Investigating the Use of Interval Algebra to Schedule Mechanically Steered Multistatic Radars

Richard Wilhelm Focke
B. Eng. (Honours) University of Pretoria

Supervised by Prof. M. R. Inggs of UCT and Dr J. P. de Villiers of UP

Thesis Presented for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Electrical Engineering

UNIVERSITY OF CAPE TOWN

· February 2015 ·

Work conducted at CSIR,
DPSS, Scientia Campus, Pretoria
in collaboration with UCT

Abstract

The findings presented in this thesis support the hypothesis that Interval Algebra (IA), as a temporal reasoning language, should perform scheduling of sensor dwells efficiently and effectively. Scheduling of multistatic radars was identified as a promising research area, as it builds upon prior research into monostatic and bistatic radars in South Africa.

The hypothesis can be validated by answering three research questions. Can IA allow a multistatic radar system to make more multistatic measurements of targets? Can IA perform as well as established multisensor scheduling techniques in terms of computational requirements? Is it possible to enhance the performance of IA by making use of parallel processing architectures?

Answering the first two research questions required selecting a comparison algorithm that is already used extensively in scheduling. The Greedy Randomised Adaptive Search Procedure (GRASP) was selected as it represents two of the biggest groupings of existing scheduling algorithms. Furthermore, greedy optimisations are often preferred as they converge to optimal solutions quicker.

Two scheduling scenarios were devised which made use of a binary mechanically steered surveillance radar network. One environment made use of a very simplistic model of the information fusion system, the other implemented all the details rigorously. The first environment was used to compare IA to GRASP, while the second tested a nimble IA scheduler. For both these environments Monte-Carlo simulations were used to test random target locations and motion.

A novel IA algorithm that makes use of reduced point algebra was generated that allowed execution time to be reduced. Another, simpler novel contribution was an IA algorithm that ensured that radar tasks are only added to the IA network when required. Using these two techniques it was possible for IA to meet both the performance and execution time of GRASP, as allows for a richer set of constraints than required to perform multistatic scheduling. The Nimble IA Scheduler is a novel contribution which solves the realistic requirement of handling fast-moving and accelerating targets, and provides a small performance increase for the surveillance system.

Answering the last research question required implementing IA on a parallel processing architecture. General-Purpose Graphical Processing Units (GP-GPUs) were selected since no published research made use these architectures and they should be well suited to solving constraint satisfaction problems. A novel parallel IA path consistency algorithm was generated in OpenCL building upon parallel versions found in the literature for supercomputers. Monte-Carlo simulations were run where both the serial and parallel versions were used to solve path consistency for randomly generated IA networks.

The results for the GP-GPU identified that for large networks there was speed-up of between two to three times for consistent networks under three conditions. Firstly, the IA network must be sufficiently large to warrant copying the data to the GP-GPU. Secondly, the IA network must have a percentage of known constraints between 25% and 75%. Thirdly, the average number of IA operators should be less than 9.8.

Thus, IA can provide equivalent performance to GRASP if the constraints are reduced. Given problems that require a richer set of constraints, these can easily be handled using IA. Nimble IA scheduling can provide a means to increase the multistatic measurements made and reduce those that are missed due to prediction inaccuracies. IA path consistency can also be used on GP-GPUs but only provides speed-ups under specific conditions.

Acknowledgments

The author was funded by the Council for Scientific and Industrial Research (CSIR) through parliamentary grant funding. Special thanks go to Prof. M. R. Inggs and Dr J. P. de Villiers, who provided valuable inputs throughout the author's studies and helped to identify the focus areas of the research. Dr J. P. de Villiers suggested investigating Interval Algebra, which has turned out to be a fruitful exercise. Thanks also go to L. O. Wabeke, who helped to modify a radar simulation environment to test Interval Algebra (IA) scheduling; this ensured that our conference paper was accepted.

Special thanks to Casper van Zyl for his valuable support throughout, and for ensuring that the CSIR parliamentary grant funding was always available. I would also like to thank my wife, family and colleagues for all their support: Adré Synman, Ana M. Huenchuquir Mardones, Elke van den Berg, Emilia (Heck) Focke, Casper van Zyl, Cecilia (Vermaak) Slabbert, Chris Venter, Christian Meckelburg, Gabriela D. Focke Huenchuquir, Humberto R. Huenchuqir Mardones, Jaco C. M. van den Berg, Larissa Focke, Leon O. Wabeke, Marietta (Slabbert) Focke, Magdaleen Vierbergen, Michael Bryant, Michael R. Inggs, Michele (Bryant) Huenchuquir, Morgan van den Berg, J. Pieter de Villiers, Robert C. Focke, Seshan Govender, Sigmar Horst Focke, Stefan Cloete, Timoteus Slabbert, Tristan van den Berg, Walter W. Focke, Willie Fourie and Willie Nel. My apologies for omissions.

I thank the reviewers of all my submitted papers and manuscripts for their detailed critique; it is not often that authors receive such excellent feedback and genuine interest. In particular, I thank Ana M. Huenchuquir Mardones, Beverlie Davies, Larissa Focke, Linton Davies and Walter W. Focke. Furthermore, I acknowledge the support of the CSIR, University of Cape Town (UCT) and the South African Department of Science and Technology (DST); and the hard work and sacrifices my wife had to endure throughout.

Contributions

This research has resulted in the following manuscripts being published/created:

- a conference paper[1] [1], with the title "Implementing Interval Algebra to schedule mechanically scanned multistatic radars", presented on 7 July 2011 at the 14th International Conference on Information Fusion in Chicago, Illinois, United States of America (USA);
- a research paper[2] [2], with the title "Interval Algebra – an effective means of scheduling surveillance radar networks", published on 16 August 2014 by the Elsevier Information Fusion journal (similar content as per Sections 3.2, 3.3, 3.4 and 4.2 as well as Chapter 5);
- a review paper, with the title "State of the art in multisensor management", to be submitted to the Institute of Electrical and Electronics Engineers (IEEE) Transactions on Aerospace and Electronic Systems journal (similar content as per Chapter 2); and
- a research note, with the title "Parallel path consistency for general-purpose graphical processing units", to be submitted to the Elsevier Artificial Intelligence journal (similar content as per Sections 3.3 and 4.3 as well as Chapter 6).

"While we may not be able to control all that happens to us, we can control what happens inside us."

Benjamin Franklin

[1] Available in IEEE Xplore at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5977496
[2] Available in ScienceDirect at http://www.sciencedirect.com/science/article/pii/S1566253514000888

Contents

Figures

Tables

Glossary

C A general-purpose programming language developed at AT&T Bell Labs. The language is a procedural computing language, where algorithms are broken up into separate functions all called from one main function.

CUDA Compute Unified Device Architecture, or CUDA, is a parallel programming language developed by Nvidia for their general-purpose graphical processing units. The language enables parallel-processing algorithms to be compiled and executed on Nvidia general-purpose graphical processing units.

Matlab Matlab® or Matrix laboratory, is a numerical computing environment from MathWorks™. The enviroment includes an object-orientated and procedural language, as well as an enhanced simulation toolkit and graphical environment (called Simulink).

Moore's law An observation, made by Gordon E. Moore, that the number of transistors in dense integrated circuits doubles approximately every two years. This translates into computing performance increasing linearly while prices remain relatively constant.

OpenCL OpenCL, or the Open Compute Lanaguage, is a parallel-programming language standardised by the Khronos Group technology consortium. The language enables parallel-processing algorithms to be compiled and executed on multicore central processing units, general-purpose graphical processing units, field-programmable gate arrays and digital signal processors.

ORD-Horn A subclass of temporal relations used by Interval Algebra defined by Nebel

and Bürckert [3], which stands for Ordered (ORD) Horn.

positioner A mechanical subsystem of a mechanically steered radar that is used to point the radar antenna[3] at the target. Positioners are able to move in both azimuth and also elevation angles, in the case of 3D radars. Surveillance radars made use of rotators, a positioner than can only rotate in azimuth.

rotator A mechanical subsystem of a 2D surveillance radar that is used to rotate the radar antenna(s) in azimuth. Rotators are a simple type of radar positioner.

[3]Which can be a mechanically or electronic scanned array of antennas

Acronyms

ADC Analog-to-Digital Converter.

AESA Active Electronically Scanned Array.

AI Artificial Intelligence.

AMTI Airborne Moving Target Indicator.

ANN Artificial Neural Network.

API Adjacent Pairwise Interchange.

ATC Air Traffic Controllers.

AU Apparent Urgency.

CFAR Constant False Alarm Rate.

CP Constraint Propagation.

CPU Central Processing Unit.

CRLB Cramér-Rao Lower Bound.

CSIR Council for Scientific and Industrial Research.

CSP Constraint Satisfaction Problem.

DAC Digital-to-Analog Converter.

DBF Digital Beam-Forming.

DCSP Decentralised Constraint Satisfaction Problem.

DDC Digital Down-Converter.

DFIG Data Fusion Information Group.

DFM Data Fusion Model.

DSP Digital Signal Processor.

DST Department of Science and Technology.

DUC Digital Up-Converter.

ECCM Electronic Counter-Counter-Measure.

EKF Extended Kalman Filter.

EWR Electronic Warfare Receiver.

FPGA Field-Programmable Gate Array.

G-MS Greedy Randomised Adaptive Search Procedure Mechanically steered multistatic surveillance radar network Scheduling.

GA Genetic Algorithm.

GMTI Ground Moving Target Indicator.

GP-GPU General-Purpose Graphical Processing Unit.

GPU Graphical Processing Unit.

GRASP Greedy Randomised Adaptive Search Procedure.

HMM Hidden Markov Model.

I-MS Interval Algebra Mechanically steered multistatic surveillance radar network Scheduling.

IA Interval Algebra.

IC Integrated Circuit.

IEEE Institute of Electrical and Electronics Engineers.

IFF Identify Friend or Foe.

IFS Information Fusion System.

IG-MS Interval Algebra Growing-network Mechanically steered multistatic surveillance radar network Scheduling.

IH-MS Interval Algebra Hogge constraint propagation Mechanically steered multistatic surveillance radar network Scheduling.

II-MS Iterative Interval Algebra Mechanically steered multistatic surveillance radar network Scheduling.

IRP-MS Iterative Reduced Point Algebra Mechanically steered multistatic surveillance radar network Scheduling.

IS-MS Interval Algebra Simplified-constraints Mechanically steered multistatic surveillance radar network Scheduling.

JDL Joint Directors of Laboratories.

KB Kibibyte.

LFC Local Fusion Centre.

LSE Least Squares Error.

LTS Long Term Support.

MAB Multiarmed Bandit.

MAP Maximum A Posteriori.

MB Mebibyte.

MCPU Multicore Central Processing Unit.

MDF Multisensor Data Fusion.

MDP Markov Decision Process.

MFR Multifunction Radar.

MHT Multihypothesis Tracking.

MIMO Multiple-In Multiple-Out.

MS Mechanically steered multistatic surveillance radar network Scheduling.

MSE Mean Squared Error.

MSMSRN Mechanically Steered Multistatic Surveillance Radar Network.

NM-MAB Non-Markovian Multiarmed Bandit.

NM-NS-MAB Non-Markovian Non-Stationary Multiarmed Bandit.

NM-NS-SP Non-Markovian Non-Stationary Stochastic Process.

NMDP Non-Markovian Decision Process.

PA Point Algebra.

PCL Passive Coherent Location.

PF Particle Filter.

PID Proportional-Integral-Derivative.

POMDP Partially Observed Markov Decision Process.

PONMDP Partially Observed Non-Markovian Decision Process.

PRI Pulse Repetition Interval.

PSK Panwalkar, Smith and Koulamas.

PSO Particle Swarm Optimiser.

Q-RAM Quality of service based Resource Allocation Model.

QoS Quality of Service.

RAM Random Access Memory.

RCL Restricted Candidate List.

RCS Radar Cross-Section.

RF Radio-Frequency.

RMSE Root-Mean Squared Error.

RP-MS Reduced Point Algebra Mechanically steered multistatic surveillance radar network Scheduling.

RPA Reduced Point Algebra.

RSA Republic of South Africa.

RSP Radar Signal Processor.

RTHC Receding Time Horizon Control.

SDR Software-Defined Radar.

SEM Standard Error of the Mean.

SFS Sensor Fusion System.

SNR Signal-to-Noise Ratio.

SPIE The International Society for Optical Engineering.

SSR Secondary Surveillance Radar.

STAP Space-Time Adaptive Processing.

UAV Unmanned Aerial Vehicles.

UCT University of Cape Town.

UP University of Pretoria.

US-DOD United States Department of Defence.

USA United States of America.

WSN Wireless Sensor Network.

Chapter 1

Executive Summary

1.1 Introduction

The research documented here falls within the field of information fusion and specifically focuses on sensor management. The focus is on the scheduling of Mechanically Steered Multistatic Surveillance Radar Networks (MSMSRNs) to constrain the research within a limited time period of four years. Mechanically steered radars were chosen as it was the most prevalent type of radar systems currently employed in South Africa. This is both in the field of defence and industry, where in the latter radars are used to detect weather phenomenon. This would allow the research to make impact immediately, as we could test the theoretical ideas quickly. Furthermore, improvements to be had would immediately be able to improve the performance of these radar systems.

Section 1.2 introduces the reader to the context of the research conducted and develops the goal of introducing Interval Algebra (IA) to the sensor management community. This is followed by a summary of the current state of the art in sensor management, as well as parallel processing research applied to Interval Algebra (IA) in Section 1.3. Then the impact this research will have on this field is highlighted in Section 1.4.

At this point the hypothesis statement, research goals and objectives are highlighted in Section 1.5. Next the methodology used to design the experiments to answer these questions is provided in Section 1.6. This naturally leads to a set of methods which is further developed in Section 1.7.

The results and a discussion are then presented in Section 1.8. Then Section 1.9

highlights the measurable outputs of the research, which were submitted or published as research papers. In these documents, the results of this research have been shared with the sensor management community. Finally, conclusions are drawn about the validity of the research and the degree to which the hypothesis, questions and goals have been addressed are handled in Section 1.10.

1.2 Research Aims

This research aims to show how IA [4] can be used to address the scheduling of a MSMSRN. As such, the task generation and prioritisation steps were simplified to limit the scope of this research. Furthermore, the radar schedulers were only implemented to a required level of accuracy. This research is the first introduction of IA to scheduling a MSMSRN and is also one of few to look at multistatic radar networks.

Specifically the research has focused on optimising the generation of simultaneous multistatic measurements. Multistatic radar is also an emerging field of research, which will lead to benefits for surveillance in future. IA scheduling algorithms developed can schedule radar sensors to make as many multistatic measurements of targets as possible per scan.

This work adds onto the conference paper presented at FUSION 2011 [1]. Through the conference paper an IA scheduling algorithm was introduced to the sensor management community; this was achieved by comparing IA to a simple heuristic algorithm. IA was used to schedule multistatic measurements of targets by two multistatic search radars.

In the conference paper, it was shown that it was relatively easy to handle uncertain azimuth order in the list of targets to be measured by using IA. The heuristic algorithm, however, could not cater for the uncertain azimuth order and thus could not measure as many targets in a multistatic way per scan. Given a surveillance network of multistatic radars, which scan an area to detect, track and classify targets. Here the scan starts and ends at the boundaries of the search volume. The better algorithm should be able to schedule more multistatic measurements per scan, as it would have found a solution that is closer to optimal.

This led to the question: what is the practical value of IA? It was not immediately possible to answer this question, as the heuristic algorithm, while being a conventional approach, was a tailored algorithm not comparable to results from other researchers. Furthermore, the initial simulated runs did not record the amount of time required to generate results using either approach.

To address the shortcomings of our prior research, it was necessary to compare IA to more established approach. The comparison needed to show that IA can perform as

well as established approaches within the same processing time budget. Furthermore, the research aims to show that the nature of IA as a temporal reasoning language matches the scheduling of sensors in a natural and intuitive way. Thus IA has additional benefits for scheduling when temporal constraints need to be satisfied.

The research also tested IA scheduling against realistic problems that would be experienced by a scheduling algorithm for a MSMSRN. For IA to be an effective scheduling approach, it should be able to cater for these problems and still produce good scheduling results. Testing against realistic targets which move appreciably during the scan time of the MSMSRN was especially important.

Finally, the research aimed to show that the use of General-Purpose Graphical Processing Units (GP-GPUs) along with IA can improve the performance of IA further.

1.3 Background

1.3.1 Multistatic radar scheduling

Multisensor data fusion, a part of information fusion, formally recognised in the middle of the 1980 decade with the formation of the Data Fusion sub-panel and the publication of the Data Fusion Lexicon in 1991 [5]. Multisensor data fusion is being used frequently in the field of surveillance to improve the generation of a recognised surveillance picture.

Multisensor management deals with the task of queueing sensors to make measurements that best serve the mission that a Multisensor Data Fusion (MDF) system is intended to complete [6, 7]. In the case of a surveillance system, this means controlling the sensors so as to gather the most pertinent information about the sensed area.

Multisensor management is contained entirely in level 4, process refinement, of the Joint Directors of Laboratories' (JDL) data fusion model [8], as refining the process of information fusion can best be achieved by controlling the inputs to the process. This is when the only inputs to the Information Fusion System (IFS) are the sensor measurements.

Recently, the updated Data Fusion Information Group's (DFIG) data fusion model was proposed [8–10], where resource and mission management replaces process refinement. Resource management is a subset incorporating aspects such as tuning all information fusion functionality. Furthermore, it also looks at how to control information collected by other means, such as military intelligence. The Data Fusion Information Group (DFIG)[1] revision of the Joint Directors of Laboratories (JDL)[2] data fusion model in Figure 1.1 shows the processing and flow of information in a multisensor data fusion system.

On level one of the DFM, or object refinement, tracks are fused, often using the fusion rule/information filter generated by Bar-Shalom and Li [12]. Either measurement or track fusion can be performed to incorporate the spatial and kinematic information. Track fusion is usually more popular in centralised systems as distributed track fusion has no closed form solution. However, as most radar systems already provide the means to form tracks it may sometimes be preferable. Measurement fusion can require significant processing and communication bandwidth, which typically means that a high-performance

[1] DFIG is a subcommittee of the Joint Directors of Laboratories (JDL) that focusses on data fusion.
[2] JDL is the research committee of the United States Department of Defence.

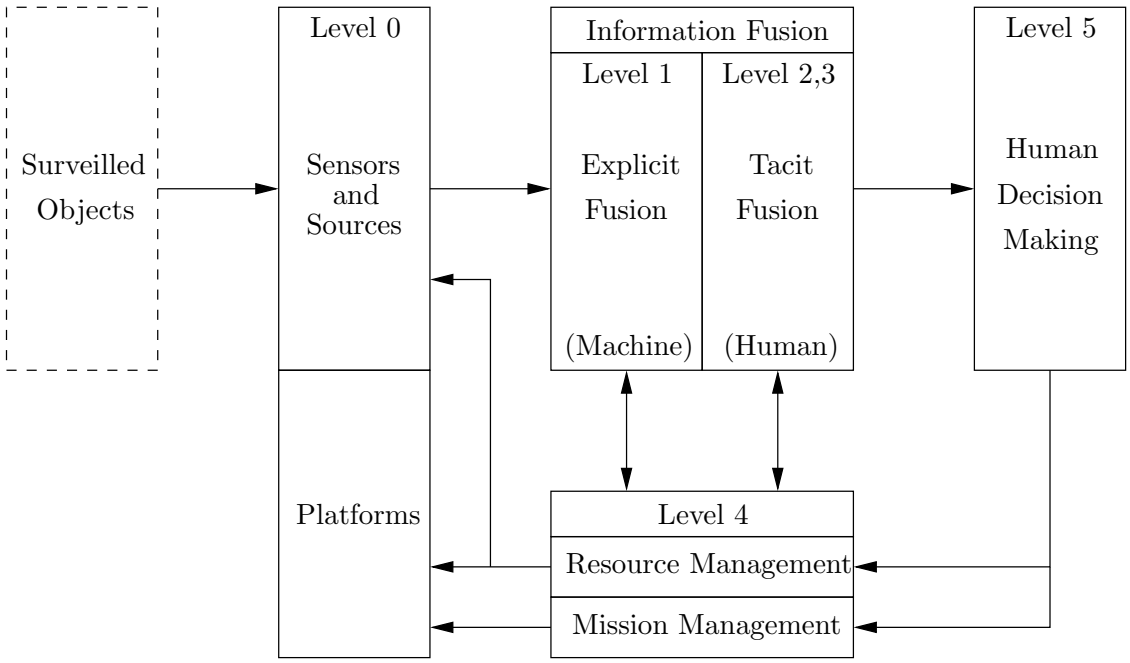


Figure 1.1: The Data Fusion Information Group revised Data Fusion Model (2004) [11]. This model is a revision of the original JDL model on data fusion. The mapping to each of the JDL levels is included in the diagram.

computing cluster is required for optimal processing. Nevertheless, it is more popular in distributed systems as it can be solved using typical data association techniques.

Similarly, classification information from various sources for the target identity are fused. Level two of the DFM, or situation assessment, groups objects and determines how the surveillance picture will progress with time. Threat assessment, DFM level three, is then used to assign a threat level/hostility indicator to moving targets. Essentially targets are prioritised by the level of threat they pose to the system or its goals. More hostile systems have a higher weight than systems that do not have the ability to make a large impact. Information is then fed to a commander or fusion operator via DFM level five or cognitive refinement.

The high-level information is also fed to processes refinement, level four of the DFM, which must control the resources of the fusion system. Recently, resource management has been proposed as a subset of process refinement. Furthermore, mission management is also incorporated as part of process refinement. Resources include: sensors, platforms, humans, as well as intelligence and contextual sources of information. Resource management can also receive commands via level five to allow human intervention. The research

documented here falls into the multisensor management aspect of resource management.

Two preliminary steps can be used to formulate a solution for multisensor management. The first is to model the sensors, their environment and the goals they must achieve. The second is the choice of architecture, which dictates how the multisensor manager will be designed and employed.

Modelling a sensor manager can be achieved using two methods [13]. The first choice is to make use of a myopic simplification and thus deal with only a very simple model of the past and the future. The alternative choice is to employ longer-term planning, which considers more historic information and generates long-term predictions. Common solutions for the latter choice include partially observed Partially Observed Markov Decision Processes (POMDPs) [14, 15] and Multiarmed Bandits (MABs), a simplification of Markov Decision Processes (MDPs) [16, 17]. These are both types of Bayesian networks and, while they are promising, they often lead to numerically difficult solutions. Thus, as longer-term planning is desirable, there is still work required to make these practically feasible in all cases.

There are various architectures used for creating a mulstisensor manager. Traditionally, a centralised architecture has been used, where a central information fusion system feeds a centralised multisensor manager with information to control all sensors. Another possibility is a decentralised architecture, where typically both the information fusion system and multisensor manager are distributed across each discrete sensor or suite of sensors [7, 18, 19]. These architectures represent the current state of the art in multisensor management, as distributed problems are difficult to solve. A hybrid of these approaches has been proposed by various authors [20, 21], which usually consists of two or more distinct levels. On the lowest level sensor management is distributed and must keep sensors busy and tune sensor parameters for high-level tasks. The higher level, sensor coordination, is centralised and ensures that collectively the sensors are optimally achieving the sensor fusion system goals.

Sensor coordination consists of planning and scheduling [7], and can be sub-divided into three functions [22] (refer to Figure 1.2). The first function of sensor coordination must solve the problem of generating tasks for each sensor. The next function of sensor coordination is to prioritise the generated tasks. Together the first two functions perform

the required planning. The final function of the sensor coordination is to place the best set of tasks in the timeline of sensing actions for each sensor. This is known as the scheduling function, which is the focus of this investigation.

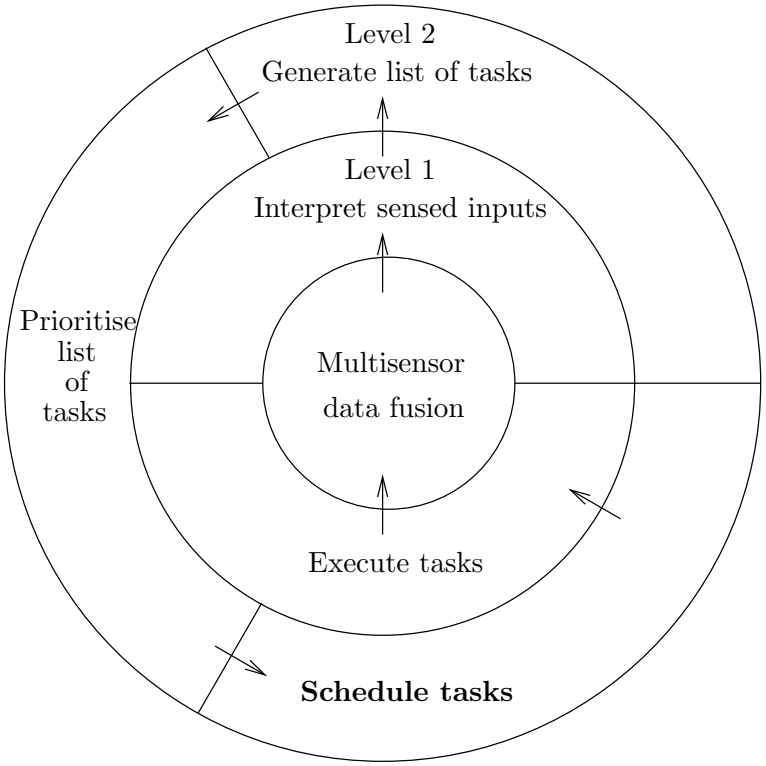


Figure 1.2: Task decomposition of multisensor management. The sensor scheduling subtask of level 2 is the focus for this research.

A sensor coordination algorithm is considered a *nimble scheduler* when it is able to rapidly adapt to changes in the surveyed area. This means that, if there are fast-moving or rapidly accelerating targets, the scheduler will incorporate the updated target locations in real-time. These targets will not only affect planning but also scheduling within very short time intervals. Thus the scheduler must be able to recalculate the schedule of tasks for the sensors concurrently with the execution of these tasks by the sensors. The updated schedule of tasks can leverage the improved situational picture as it is generated by the information fusion system.

Current scheduling solutions

Sensor scheduling treats the sensor resources as a timeline extending from the present towards the future. As such, scheduling is a combinatorial optimisation problem very

similar to the knapsack problem. The goal is to fill the timeline with tasks such that the sensor is never idle. Idle time is wasteful since this time is better spent catching up on tasks that may later cause a bottleneck.

A good overview of types of algorithms for sensor scheduling can be found in the work of Xiong and Svensson [7], Musick and Malhotra [22], as well as a more recent work by Ding [23]. There are many approaches that can be followed to schedule individual sensors. However, not all of them are directly applicable to multisensor scheduling, which is required for an MDF system.

Early in the history of sensor management scheduling was performed by human operators with only minimal assistance provided by the system that was being managed [24]. Next, heuristic approaches that captured much of the domain knowledge of human operators were employed. These approaches are still very popular today as they require minimal computation time and are easy to develop [21, 25–27].

As research in the field broadens to encompass additional functions of sensor coordination, this aspect is sometimes handled intrinsically by task prioritisation algorithms [16, 28–31]. Examples are split among those using information and decision theory. This has the benefit of not wasting computation especially if the mission of the MDF system changes on a high level. In this case, instead of handling a timeline of tasks, the sensor manager only deals with the current task to schedule. On the other hand, if a centralised fusion centre is used and stops operating or if communication is lost in a decentralised system, there will be no timeline of tasks to continue with in the interim. Just doing arbitrary tasks during this time can have detrimental effects not only on the optimality of the scheduling solution, but also on the mission of the MDF system as a whole.

The scheduling problem can be solved through the use of optimisation algorithms when information-theoretic approaches to task prioritisation are used [32]. These optimisation algorithms fall into two categories: mathematical programming [26, 33–35] and artificial intelligent search techniques [36, 37]. Simulation techniques are another possibility, where possible future timelines are investigated using multiple trials [38–40]. Sometimes random approaches are followed and these are typically used as an electronic countermeasure [41].

Artificial intelligence techniques have also been proposed in the past and have predominantly used reasoning/expert systems. Examples include fuzzy-set-based reasoning

[42, 43] and fuzzy decision trees [43] all within the context of an expert system. In these systems, the scheduling rules are captured by analysing linguistic rules of thumb provided by human sensor operators.

Multistatic radars

An interesting application of multisensor management for radars is the possibility of making multistatic measurements [44–46], which can be used to increase the probability of detection of small targets in heavy clutter scenarios. Increasingly, there is a trend where small boats are used in piracy and terrorist activities to attack larger vessels. Coordinating the measurements of multiple radars could potentially mitigate some of these problems, by ensuring that these targets are detected and tracked.

However, most radars of the affected vessels are not very sophisticated, and thus require a simple solution to benefit from multistatic measurements. The radars are not sophisticated when considering their antenna and steering mechanisms, as clearly AESA and digital beam forming require much more sophisticated antennas. However, the processing required when adding a multistatic measurement capability will make the radars more sophisticated.

Multistatic measurements are possible by adequately scheduling each radar to measure the target simultaneously. Each radar must also be able to receive the transmitted signals of the other radars, or the signals must combine coherently through constructive interference to increase the energy on the target. For multistatic radars, where a radar can distinguish the signals of the other radars, this receiving radar can then also make a bistatic measurement. A bistatic measurement is made through knowledge of the bistatic angle and delay experienced. The bistatic measurement can be represented in Cartesian space as an ellipse, with the receiving and transmitting radars as the ellipsoid foci. The radar can also make a monostatic measurement by determining the angle of arrival[3] and the delay experienced by the signals it transmits. Using Doppler processing the frequency shift of the signal can be determined for both measurements, this allows the radar to determine the range rate of the target. Thus, each radar is able to make more measurements of the target. Each radar then has an increased probability of detection and also makes

[3]The positioner angle is used and monopulse techniques can improve this angle.

more accurate estimates of the positional and kinematic attributes of the target.

In many instances, multiple radars are already deployed on the platforms used to survey an area. For example the radar systems of military and weather monitoring installations. In these instances using the radars to form a multistatic radar network will improve the probability of detection (P_d) [47]. The P_d increases if the power of each radar can be used to increase the signal-to-noise ratio over that of a single radar system. Alternatively, using orthogonal signals or frequency allocations, the radar network can make more measurements. Bistatic measurements also become possible, which is not possible without enabling the radars as a multistatic network. In the latter case, while the P_d of each detection is the same, the P_d of the cumulative number of measurements must increase as there are simply more measurements that are statistically independent. A distributed radar system also allows the P_d to increase in areas, which would have been far away from a single radar. Clutter and target Radar Cross-Section (RCS) diversity further improves the performance of a multistatic radar network.

1.3.2 Interval Algebra

Interval Algebra (IA) is an artificial intelligence technique that does not require performing searches. IA falls within the field of constraint satisfaction problems (CSP). Apart from the author's conference paper [1], there appear to be no published papers that investigate the use of IA in information fusion.

Briefly, IA reasons about the start and end points of intervals of time. Hence it is able to ensure that tasks adhere to temporal constraints (with the potential of adding duration constraints) but is not able to prioritise tasks. It can also be used to check if the intervals are consistent with each other, given fixed constraints. As sensor tasks are defined as taking a specific amount of time and starting at an absolute point in time, they can easily be represented as intervals. Temporal constraints between sensor tasks are also often required and these can be captured using IA relationships.

Temporal constraints are captured as relationships between intervals, which can consist of one of the 13 IA temporal operators. The temporal operators are 'before', 'after', 'meets', 'met by', 'overlaps', 'overlapped by', 'starts', 'started by', 'during', 'containing', 'finishes', 'finished by' and 'equals'. The size of the constraints refers to the number of IA

operators present in the relationship. The entire group of intervals and their relationships form a graph, where the labels of the graph are the relationships (containing a number of IA operators or constraints) and the nodes are the intervals. The graph is more conveniently captured as matrix called the IA relation matrix. The degree of constraints is then the amount of labels that are known, unknown labels contain relationships with all IA operators present as this is the case for maximum ambiguity in the temporal ordering.

IA as a constraint satisfaction problem (CSP) is solved using arc, path or total consistency algorithms. Arc consistency considers only the relationships between two nodes at a time. Arc consistency can only be used to fully resolve a small subset of IA, as while most IA networks can be arc consistent to be totally consistent each label must only have a single constraint. Path consistency is useful when arc consistency cannot resolve the solution of the network. It considers the relationships between three nodes at a time. However, as per arc consistency not all networks that are path consistent will be totally consistent. To fully resolve consistency of the IA network using all IA operators total consistency algorithms, such as back-tracking algorithms are required. These algorithms perform a search over the entire network and typically use along with path consistency to propagate changes.

Alone IA can also provide a schedule of tasks with a low latency and within few computation cycles. Thus, IA should be able to efficiently plan a single dwell for advanced sensor systems. In the case of the surveillance radar systems discussed here, this would entail planning the tasks for the radars during the next scan of the sensed area. IA can fill the timeline of the radars with tasks such that temporal constraints are not violated, but it cannot make decisions about which tasks are best.

IA can also be used with other algorithms such as information-theoretic algorithms to provide a holistic sensor management solution. IA could be used as a pre-processing step which eliminates tasks that violate constraints. Thereafter, all tasks not excluded by IA could be prioritised and the best tasks selected using an information-theoretic algorithm such as Shannon's entropy.

IA also does not dictate a specific multisensor management architecture. The form of IA used here will naturally work for a centralised architecture or the central part of a hybrid architecture. It can also be used for the scheduling of a single sensor and even

in a decentralised approach. Nevertheless, it is also possible to use a decentralised CSP algorithm to solve the constraints [48].

While IA is being used for scheduling here, it should be noted that IA can be used to resolve constraints wherever decisions need to be made. Thus, IA could be used in other parts of an information fusion system such as object refinement, situation assessment, threat assessment and user refinement.

1.3.3 Parallel-processing architectures and constraint satisfaction problems

The use of parallel processing architectures is increasing due to the physical difficulties encountered recently in terms of increasing clock frequency. In the past, it has been possible to gain significantly in terms of processing time by reducing the silicon size required, thereby increasing clock speed. Parallel processing architectures is one way of maintaining the applicability of Moore's law, which will eventually be impossible to maintain through reduction in silicon size alone. The use of Multicore Central Processing Unit (MCPU) architectures, General-Purpose Graphical Processing Units (GP-GPUs), Digital Signal Processors (DSPs) and Field-Programmable Gate Arrays (FPGAs) is increasingly prevalent. The ability to leverage these architectures for processing is not only desirable but required for efficient processing.

In this research, the execution of Interval Algebra (IA) [4] on GP-GPUs is investigated. OpenCL is used to implement an IA total-path consistency algorithm on a readily available GP-GPU[4]. Comparisons are drawn to serial versions of the algorithm on a host CPU[5]. To date no published research has considered the application of parallel CSP algorithms on GP-GPUs. IA is a Constraint Satisfaction Problem (CSP), and while much research has been done on parallel techniques for MCPU architectures, this work is not directly applicable to the GP-GPU architecture. Many problems solve IA networks using path consistency, an algorithm that considers three nodes in the IA network and ensures that the constraints captured are consistent.

GP-GPUs have three major limitations over the more general CPU architecture but

[4]Nvidia GeForce GT540M
[5]Intel Core i7-2670QM @ 2.2 GHz

also have many advantages. In GP-GPU processing, branching becomes very expensive as in this architecture both branches are performed sequentially by all processing nodes affected by the branch. Secondly, GP-GPUs make use of advanced synchronous dynamic memories that have high latency, making arbitrary accesses to memory expensive, which can counteract any gains in processing time reduction. Finally, there is no way to synchronise all parallel processing clusters.[6] One major advantage of GP-GPUs is the sheer number of processing units that are present within the silicon.

Thus, while it may appear that path consistency should be amenable to GP-GPU processing, this is not necessarily the case. Path consistency contains a high degree of parallelism [49, 50]. However, CSP algorithms in general access data frequently but only perform minimal computations (simple logical operations) for each access. Secondly, GP-GPU processing has an overhead required for copying the data from the host CPU to the GP-GPU and back. This will generally not be a problem if the algorithm requires a lot of computations and is highly parallel. Only the latter is true for path consistency and may result in a non-effective GP-GPU implementation.

Related work

Many parallel versions of arc consistency algorithms have been published [51–55]. This CSP algorithm checks that for any two nodes in the network the labels are consistent. It has been the preferred route of checking the consistency for many classes of CSPs. This is due to the fact that arc consistency requires much less processing that path consistency. Often arc consistency is sufficient to find solutions to the CSP. Unfortunately, arc consistency does not guarantee that a solution will be obtained for more complex CSPs. Thus, if full IA is required then arc consistency might not be sufficient, and path consistency is required. Nevertheless, arc consistency can be used as a pre-processing step to find a locally optimal arc-consistent solution.

A general consideration of arc consistency and parallel versions for each of the known serial arc consistency algorithms are proposed by Samal and Henderson [51]. Interestingly, the parallel version of the simplest serial algorithm almost achieves the maximal theoretical

[6] However, each cluster consists of 32 parallel threads where synchronisation can be applied.

speed-up. Results are obtained by running the code on a proprietary super computer[7].

Kasif [52] considers parallel versions of discrete relaxation, another arc consistency algorithm. It is shown that propositional CSPs and satisfiability of proposition Horn classes have a tight connection. Thus, discrete relaxation algorithms are inherently sequential in the general sense and cannot always gain from parallel processing. This is important to consider for future CSPs and can be satisfied by checking that the parallel version does speed up the computation in general.

Cooper and Swain [53] looks at how domain dependence can speed up computation by removing redundant computations. Theoretical results are obtained for integrated circuit designs (IC) and practical results using a proprietary super computer[8]. Future work should consider how domain dependence can be accommodated for GP-GPU processing, especially for parallel path consistency. This may be able to reduce the amount of redundant computations and thereby possibly improve the achievable speed-up. Furthermore, IA processing making use of FPGAs could make use of the documented logic transforms.

Kirousis [54] considers a special form of CSPs known as implicational CSPs. A theoretical analysis is given to show that the algorithm is indeed correct. In an implicational CSP the presence of a value at a node implies certain relationships and vice versa. The research documenter here does not consider this type of CSP, as it is not clear how sensor scheduling can be accommodated. However, it may be interesting to consider the application of implicational CSPs and how these can benefit from GP-GPU processing.

Fabiunke [55] looks at distributed agents that act in isolation to solve arc consistency. In this case, the nodes do not communicate but instead must use the information in the CSP network to decide on the next value. He effectively proposes a CSP algorithm that is modelled around parallel distributed processing as originally applied to artificial neural networks. This approach requires some randomisation and could help FPGA and CPU algorithms but would not run efficiently on GP-GPUs.

Susswein et al. [49] and Keretho et al. [50] have analysed parallel path consistency algorithms that can be used to speed up the execution of path-consistent searches on CSPs. Path consistency looks at three nodes in the CSP and ensures that the relationships

[7] BBN Butterfly Computer
[8] The Connection Machine

are consistent. Again path consistency does not guarantee total consistency but in some instances a path consistent solution is sufficient.

Path consistency is sufficient for certain types of multisensor scheduling. In particular, this is the case for mechanically-steered multistatic radar network scheduling. The reason for this is due to the fact that the target ordering for each of the radars is captured in the IA network. These must always be a physically possible configuration and, thus, the IA network of the intervals represents a consistent scenario. That is the individual network of intervals for each radar is termed n-consistent [56]. Then linking these two consistent scenarios together using the equality relationships can only result in either a consistent or inconsistent scenario. This consistency will be found using path consistency, as the resulting network is also strongly n-consistent. That is the network is consistent in all respects apart from the newly added node. The solution can be found by applying path consistency and should either find a solution or declare the network as inconsistent [56].

Ladkin and Maddux [57] also mentions the use of a parallel path consistency algorithm for IA that computes the set composition over the entire IA relation matrix. The form presented is the most general form of path consistency for any constraint programming technique. Several improvements over the basic form are used to reduce the computation required [58], including only storing half the IA relation matrix and computing only the required set compositions (as per Allen [4]). The documented algorithms do not use the improved constraint propagation algorithm of Hogge [59].

Point Algebra (PA) is another temporal language for constraint satisfaction problems similar to IA, which only has three basic operators ('before', 'equal' and 'after' in IA terminology). More recent work by Gerevini and Saetti [60] has considered how PA could be adapted for parallel processing architectures. This time the researchers have applied the field of mathematics known as metagraph closure for time series and serial-parallel (SP) metagraph representations of PA. Further work would be required to apply these techniques to IA but doing so could certainly be advantageous.

For difficult CSPs, path and arc consistency are not sufficient to provide a solution, and then total network consistency algorithms would be required. Nebel [61] looks at parallel forms of global consistency and in particular analyses the use of the ORD-Horn subclass of IA in the splitting function. The ORD-Horn is a special subset of IA [3], which is known

to be solvable using path consistency. The splitting function checks to see if a relation belongs to the ORD-Horn class. As the ORD-Horn subclass covers 10% of IA, it results in the least amount of backtracking. The author finds that in 40% of their tested cases the ORD-Horn subclass finds the solution first. While total consistency should benefit from GP-GPU processing we leave this as future work. The three IA subclasses and two methods of producing random scenarios used in this research are aligned with this work.

1.4 Impact Statement

IA is a computational intelligence algorithm and has not received any attention from the sensor scheduling community. Allen [4] devised IA as a way of capturing temporal information about intervals consistently. Each task given to a sensor or set of sensors can be seen as an interval in IA. Thus, IA should be a good tool for solving scheduling problems efficiently. This research is novel as it is the first to apply IA to the field of multisensor management.

Using multistatic radars would employ knowledge gained in monostatic radar by both the Council for Scientific and Industrial Research (CSIR) with that gained in bistatic radar of the University of Cape Town (UCT). Such a radar system has the possibility of making more accurate measurements of target kinematic attributes. This is possible through measurement fusion of the bistatic and monostatic measurements that can be made. Thus, it would be beneficial to the Republic of South Africa (RSA) for researchers to work in the emerging field of multistatic radars.

1.5 Hypothesis, Goals and Objectives

Hypothesis statement: *Interval Algebra, as a temporal reasoning language, performs scheduling of sensor dwells efficiently and effectively.*

An efficient scheduling algorithm provides good performance, while an effective scheduling algorithm caters for all real-world constraints. The author aimed to answer these specific research questions to prove the hypothesis:

1. Can Interval Algebra allow a multistatic radar system to make more multistatic measurements of targets?

2. Can Interval Algebra perform as well as established multisensor scheduling techniques in terms of computational requirements?

3. Is it possible to enhance the performance of Interval Algebra by making use of parallel processing architectures? If so, what performance increase is to be gained?

The goals, supporting the hypothesis, of the research are:

1. to make novel contributions to the field of multisensor management,

2. to build on the expertise of South African researchers in the fields of monostatic and bistatic radar by aiding the multistatic capabilities of existing radar systems in South Africa,

3. to publish the work to ensure that the research remains current and is ratified by the international research community, and

4. to use modern parallel processing architectures such that the research remains relevant.

1.6 Methodology

The methodology section gives an analysis of the methods used during the research. This includes why the methods were chosen and their applicability to the research questions. The methods section later on provides the details of the methods that were selected. This section captures the design of all experiments performed to:

1. validate the hypothesis statement,
2. answer the research questions, and
3. satisfy all the research goals.

Answering the hypothesis statement requires first selecting comparison algorithms, which is handled in Section 1.6.1. Then each research question must be answered as they each form part of the holistic view towards proving the hypothesis. This is handled in the two experiment design sections, namely Section 1.6.1 and Section 1.6.2. Finally, how the goals of publishing research outputs and thereby validating the research performed will be accomplished is captured in Section 1.6.3.

1.6.1 Design of multistatic radar scheduling experiments

This section deals with the methodology followed to answer the first two research questions. Since IA is a new technique for multisensor management, the research paradigm selected is a comparison experiment. The comparison experiment needed to be sufficiently realistic to ensure a valid comparison was drawn. However, it is not necessary to test all nuances of the scheduling environment. To limit the scope of the research, multistatic radar scheduling was selected as the sample problem. The reason for this was twofold: firstly, scheduling of multistatic radars is novel and there are few examples of research in this field; and secondly, this is a research topic that is of interest to both the funding organisations of this research. Mechanically steered radars was assumed, since both the NextRAD[9] and MECORT[10] systems employed by UCT and CSIR respectively are currently mechanically steered.

[9] A multistatic research radar facility co-developed by UCT and UCL.
[10] A tracking radar research facility owned by ARMSCOR and operated by the CSIR.

Selection of comparison algorithms

In designing the experiments for this research, the first step was to select scheduling algorithms with which to compare IA scheduling. However, the focus of this research is not an exhaustive comparison but rather an introduction to IA for the purposes of multisensor scheduling. While it is theoretically possible to use all existing scheduling algorithms to draw comparisons, this is impractical for two main reasons.

Firstly, not all existing algorithms are fully disclosed in published literature. Thus, to use the algorithm, the details required for implementation will first have to be derived from the mathematics and pseudo code found in the literature. Secondly, the simulation time and implementation time would be vast. The simulation time is limited by the computing resources available to the researchers and actual calender time allowed for the research. The long implementation time arises as there is no sharing of code in this field and the algorithms make use of various branches of mathematics.

Thus, the author and his supervisors decided to select a single comparison algorithm. However, the algorithm should satisfy two simple requirements. It should:

1. be representative of algorithms utilised within the field and
2. provide good scheduling solutions while being efficient.

For the comparison simulations it was important to compare the IA scheduling algorithm to another widely used algorithm.

GRASP was selected from the literature survey conducted to use as another comparison algorithm. The sample problem to solve is a combinatorial optimisation problem and GRASP excels at these types of problems for two reasons: firstly, it is designed to handle combinatorial inputs; and secondly, it converges to good local optimal solutions quicker than other approaches. Thus it is a computationally efficient solution for combinatorial optimisation. For all these reasons GRASP was selected.

The research includes a meaningful comparison of IA to GRASP to show that IA can be used in practical settings. Most scheduling algorithms tend to fall predominantly in three categories: heuristics, optimisation and information-theoretic approaches. Furthermore, many scheduling algorithms make use of greedy optimisation, as these tend to lead to

quicker convergence [7].

GRASP as a meta-heuristic optimisation algorithm seems to be a good representative algorithm for both of the first two categories. It is neither a full heuristic, as the search aspect is done in a rigorous manner that should find optimal solutions, nor is it very cumbersome, as it uses simple heuristics to build locally optimal solutions. GRASP was chosen as it is a heuristic search algorithm and should require both little processing time and provide good scheduling performance [33]. This comparison ensures that IA is computationally feasible while achieving similar performance to that of a conventional scheduling algorithm. Interested readers can refer to Section 3.4 for the theory of GRASP.

Optimality is not achievable in general and it not necessarily the best approach for scheduling, mainly for the reason that there are other important requirements such as meeting deadlines, resource usage, processing requirements and trading off constraints. My goal is to demonstrate IA to the sensor scheduling community and not prove optimality in all cases. IA should be seen as a tool that can be used alongside more rigorous approaches, for example with search procedures either to validate or generate solutions. IA can be used to validate temporal constraints and this is required for many problems.

Monte-Carlo simulations

Given that mechanically steered surveillance radars typically scan a volume in a defined pattern: by either rotating or through the use of scan patterns, the metric chosen to decide performance was selected as the mean number of multistatic measurements generated per scan. The location of targets within the scan volume and the placement of the radars together form a target-radar geometry. This geometry limits the amount of multistatic measurements that can be made. A better scheduling algorithm should make more multistatic measurements per scan. Multistatic measurements are beneficial to tracking the targets for various reasons given in Section 1.6.1. To ensure that biases did not result a target revisiting mechanism was employed. This ensures that the scheduler did not simply choose the longest possible sequence at the expense of measuring targets. Furthermore, the processing time required by the algorithm is also important. The scheduler processing time cannot be longer than the scan time without affecting system performance. Both these performance metrics require the use of Monte-Carlo analysis to ensure a sufficiently

representative mean is achieved.

For the multistatic measurements made, it is important to test many target geometries. Thus, many simulations were performed for each Monte-Carlo batch. To keep the simulations tractable during implementation and testing, the durations of the simulations had to be reduced. Processing time also fluctuates based on the scheduling performance of the operating system. All simulations were run on systems that were idle apart from having to run the Monte-Carlo experiments. Nevertheless, to average out all the effects of a modern multitasking operating system, multiple runs were also beneficial.

To fully answer the first two research questions it is also important to consider the other practical considerations encountered when scheduling a multistatic radar network. These practical concerns that must be addressed include: the effect of target motion and therefore the need for scheduling nimbleness; and handling of monostatic scanning, tracking and confirmation tasks. Here an exploration paradigm was followed and a few techniques for handling these concerns were devised. Again, Monte-Carlo simulations were run on a more complex simulation environment, to test the effectiveness of the techniques. The simulations would also determine how important these aspects were to scheduling a MSMSRN.

1.6.2 Design of parallel architecture experiments

This section highlights the methodology followed to answer the third and last research question. Given the prevalence of parallel processing architectures, any new research should also test whether these architectures can be employed with the algorithms developed. Using parallel processing architectures can give performance gains in terms of processing time, which can then be used to achieve better performance against other metrics, such as the mean multistatic measurements made per scan. OpenCL allows targeting MCPUs, GP-GPUs and FPGAs. This makes it ideal for transcribing algorithms intended for parallel processing. Again the research paradigm is a comparative approach, in this case between parallel processing and serial processing. Thus, C was used to implement the serial versions of the basic IA algorithms as efficiently as possible.

Monte-Carlo experiments were required to test the processing time of the serial and parallel algorithms. As mentioned in Section 1.6.1, the processing time fluctuates for

various reasons on multitasking operating systems. Furthermore, on parallel processing architectures there are often start-up conditions that take time that is not required on subsequent executions. These need to be averaged out using sufficient Monte-Carlo simulations runs per experiment.

1.6.3 Accomplishing research goals

The first research goal was achieved by generating novel algorithms. IA was selected as a novel contribution to the multisensor management field. Using IA in scheduling revealed a novel optimisation that can be used for IA algorithms in general. Nimble scheduling could also be investigated and the IA algorithm is a novel approach for a surveillance radar network. Furthermore, using IA along with GP-GPU technology has not been attempted thus far. The other research goals are achieved as follows. Interaction opportunities with other multisensor management researchers were attended. This research made use of extensive literature surveys to ensure that the published material is both in line with established research in the field and also performed as per the standards defined for the field.

1.7 Methods

1.7.1 Multistatic radar scheduling investigation

The methods followed for the comparisons of IA to GRASP is the same as the experimental set-up of the author's previously published work [1]. For this comparison, the simulations also made use of a target priority queue to prioritise targets as depicted later in Figure 4.1. These simulations also simplify the information fusion system by only simulating fused tracks. This was done to keep the simulations tractable such that many simulations could be executed for the Monte-Carlo analysis.

The nimble scheduler simulations require a more realistic approach. Thus, unlike the previous work, the simulation environment makes use of Kalman tracking filters [62]. The tracks of the radars are then fed to a central information fusion system. Here the tracks are associated using the auction algorithm for a global nearest neighbour association [63]. An information filter is then used to fuse the associated tracks [64]. The details of these algorithms are given in Appendix C.

Finally, the covariance of the information filter is used to prioritise tracks. When managing real sensors this is a better approach for prioritising targets than simply using the priority mechanism (as in Figure 4.1 later). It is possible to prioritise tracks which are about to be lost or where the detections for a target are sporadic using the information filter covariance. However, it may still be desirable to consider a mechanism such that all targets are revisited frequently. Using only the covariance of the information filter may degrade system performance, as the system will only focus on a few bad tracks at the detriment of all the other tracks. Thus a hybrid approach between using the two approaches would lead to a better overall system performance.

Scheduling problem

The sensor management problem to solve is selecting the maximum number of targets to measure with multiple radars. It is assumed that the mechanically steered positioners of the radars are not allowed to change direction during the scan, as doing so would increase the cost and complexity of the radars. Thus, each of the radars will temporally scan over

all targets in a specific sequence.

Uncertain azimuth order in the sequence of targets

Uncertainty arises in the azimuth ordering of the targets for either radar. This is due to the azimuth angle measurement accuracy of the radar, which cannot be infinitesimally small. This is an unavoidable situation even if monopulse measurements are made, as in both cases the angular accuracy is determined by the beam width [65]. Consequently, when targets are too close together in azimuth angle the radar is unable to discern their true order.

The MSMSRN scheduling algorithms must be able to exploit these uncertainties, so that they can be used to schedule more multistatic measurements per scan. Exploiting the uncertainties means that the scheduler is able to handle uncertain azimuth order in the sequence without imposing a strict azimuth ordering on the targets. It is possible for the scheduler to assume a certain order, even if one cannot be easily discerned but doing so reduced the freedom of the scheduler when choosing targets to measure. For instance, if it is assume that the targets are in a specific sequence, and this sequence differs for the sequence for the other radar, then only one of the two targets would be scheduled. Allowing for the uncertainties, the scheduler can schedule both targets and simply move the radar beam for which the target ordering is known. Furthermore, no tracking degradation will be experienced in the case where the target ordering is uncertain. This is due to the fact that the targets are sufficiently close to the centre of the radar beam in order to receive maximum antenna gain.

Figure 1.3 illustrates a scenario where targets are measured with uncertain azimuth ordering. Radar one does not need to discern the true order of targets three and five, as both targets are simultaneously present in the beam of the radar at the azimuth angle depicted. Thus, the true ordering of the two targets may not be clear, especially if the azimuth angle measurement is not sufficiently accurate. Nevertheless, a scheduling algorithm can use this information to simply allow the radar to dwell on this azimuth while the beam of radar two is steered over both targets

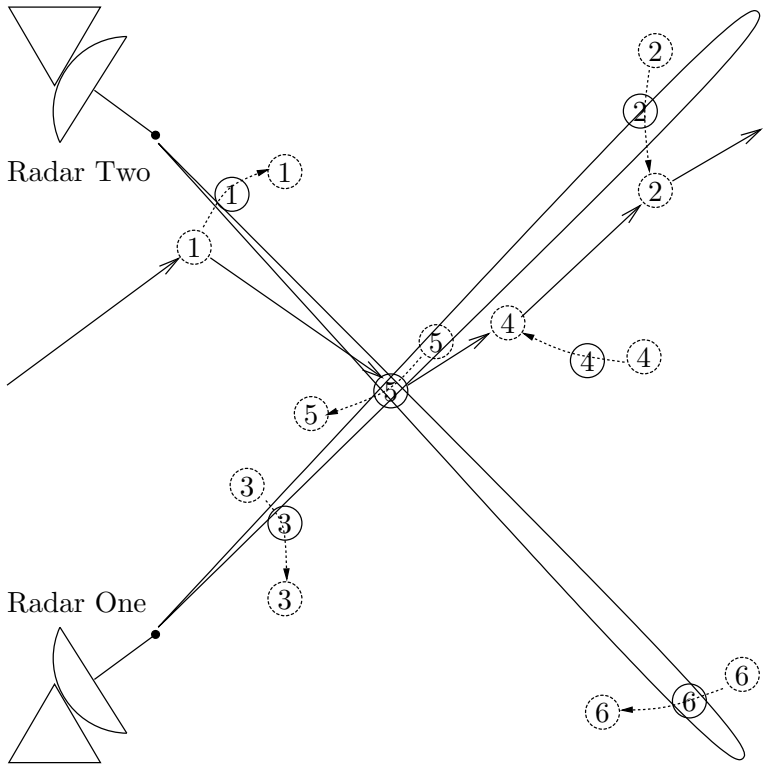


Figure 1.3: Uncertainties in the sequence of targets and target motion as scheduling processes.

Target motion and scheduler nimbleness

Since the targets are moving, the geometry of the targets relative to the two radars changes over time, as depicted in Figure 1.3 by the curved dotted arrows. If the targets move fast enough and scheduling is performed without any future prediction, then it is possible that the target will not lie in the beam of one or both radars and a multistatic measurement will not be made. Similarly, targets that have an uncertain azimuth ordering for a radar will only remain in this state for a brief period of time.

As a result, the scheduling problem to solve is dependent on the targets that are scheduled. Each time a target is scheduled, the intersection point of beams of the radars must move from the current target to the next target. The shortest path is a straight line between the two targets and is shown in Figure 1.3 as the solid straight arrows. Thus, the minimum time that will pass is the angular difference between the two targets divided by the maximum scan rate. This value will be largest for the radar which has the greatest angular difference to rotate.

During this time, all targets that must still be measured by the radars later in the

scan will have moved. The distance required to move out of the beam of the radar is dependent on the target distance from the radar, the beam width of the radar and the direction of motion. Assuming straight line motion, the last parameter can be simplified to an aspect angle of the motion. From the distance and the scheduling time between the two multistatic measurements the minimum velocity can be calculated, which will cause multistatic measurements to be missed.

Scanning, tracking and confirmation tasks

The multisensor management solution investigated here then uses a hybrid architecture consisting of two levels. The higher-level radar coordinator attempts to schedule multistatic measurements for as many targets as possible. The lower-level radar manager is replicated for each radar and operates independently of the radar coordinator. The radar manager could also have been implemented using IA; however, it is not the focus of this research. In this research, the radar manager makes a decision to track, confirm or search independently.

Thus far, only the multistatic aspect of the radar system has been discussed. However, nothing stops the radars from scanning for new targets while they are not making multistatic measurements. Furthermore, targets that will not be measured in the multistatic mode should still be measured in a monostatic mode by the radar to maintain a stable track. Finally, it is also possible to confirm the presence of new targets during the same time.

Performing these types of tasks will not affect the multistatic measurement capability of the radars for slow targets, as they only require one radar to make the measurement. Also, they always temporally occur between two multistatic measurements for the radar performing these tasks. During the time that these tasks are performed the other radar may only be required to slow down.

These tasks will, however, impact on the maximum scan time, and thus if there is an upper bound on the scan time or a requirement for a fixed scan time, only a certain number of these tasks will be possible. This can easily be accommodated in any scheduling algorithm by keeping a sum of the duration of all tasks. As long as the sum of task durations is less than the maximum scan time, more of these tasks can be added.

Simulation set-up

For each of the comparison simulations[11] and nimble simulations[12] performed a square area was selected. One radar is situated at coordinate bottom left corner of the area, while the other radar is situated at the top left corner of the area. Both radars are assumed to have a maximum detection range that can observe the entire area. The maximum slew rate of the positioners of the radar was selected to be 90° rotation every second. The radars always scan over the area in opposing directions, therefore if one radar is scanning clockwise the other will scan anti-clockwise.

A number of targets are then generated with random starting points and random motion profiles. The relative positions of the targets in relation to the two radars represents a target geometry. During the scan, the radars are not allowed to change scan direction until they have reached the boundaries of the surveyed area. With these constraints and any given target geometry there is a different upper limit of multistatic measurements that can be generated. The target geometry will change slowly as the targets move during the simulation. Target motion, for the nimble simulations, is modelled according to the Singer random-walk acceleration model [20]. The details of this model are given in Appendix C.1.

Three Monte-Carlo simulation batches were performed to analyse the performance of IA scheduling. The first two batches compared IA to GRASP, where one batch varied the number of targets, and the other varied the radar beam width. The final batch of simulations determined the nimbleness of the IA scheduler when used in highly dynamic scenarios. During a simulation run, the target geometries vary slowly due to the target movements. The initial target geometry is generated randomly and varies greatly between two different random seeds.

A global counter was used as the random seed and was only incremented after testing all scheduling algorithms. A 100 different global counter values were used for each simulation run in the batch of simulations. The random seed determines the initial placement and paths of the targets, as well as the priority change of each target within the priority queue when used. The random seed was reinitialised with the same value every time the MSMSRN scheduling algorithm changed.

[11] https://code.google.com/p/multiple-radar-fusion-simulation-simple/
[12] https://code.google.com/p/multiple-radar-fusion-simulation-realistic/

1.7.2 Parallel processing architecture investigation

Testing was performed by generating random Constraint Satisfaction Problem (CSP) networks by two of the three methods given by Nebel [61]. The methods are denoted by $\mathrm{A}(n, d, s)$ and $\mathrm{S}(n, d, s)$. The three parameters are number of nodes n, degree d of constraints that are known and average constraint size s.

The first method, $\mathrm{A}(n, d, s)$, generates random instances of n nodes with degree d. The degree d randomly selects $d/(n-1)$ relations that are not the set of all IA operators[13] of the number of independent relations. The constraints size s denotes the average number of intervals in a selected relation. The second method, $\mathrm{S}(n, d, s)$, generates random instances that are guaranteed to be consistent.

This research used the known set of subclasses of IA mentioned by Nebel [61] to select specific values for the average constraint size s. Furthermore, this research also employs Point Algebra (PA) (or $\mathcal{D}$) and IA ($\mathcal{I}$), which gives five different values for s. The strict hierarchy is $\mathcal{D} \subset \mathcal{C} \subset \mathcal{P} \subset \mathcal{H} \subset \mathcal{I}$.

Monte-Carlo simulations

Parallel processing results were obtained by executing the OpenCL algorithm given in Section 6.2. The parallel algorithms execute on a GP-GPU[14] using a MCPU[15] to execute the host code. The serial version runs on a single core of the same MCPU. The execution time is measured using the Linux system timer[16], which measures the time accurate to around 1 ms. Two Monte-Carlo simulations were run that consisted of three independent batches. The first simulation used the $\mathrm{A}(n, d, s)$ method, while the second used the $\mathrm{S}(n, d, s)$ method. Each batch varied by one independent variable, which were either the network size n, the degree of constraints d or the size of constraints s.

The first batch of a simulation tested the effect of increasing the number of nodes in the CSP network. The second batch of a simulation tested the effect of changing the degree d of constraints, i.e. those relationships that are not the set of all operators. Finally, the third batch of a simulation tested the effect of increasing constraint size s in the CSP

[13] The set of all IA operators denotes no prior knowledge of the relationship between two intervals.
[14] Nvidia™ GeForce™ GT540M
[15] Intel™ Core™ i7-2670QM @ 2.20 GHz
[16] Ubuntu Linux Long Term Support (LTS) 12.04

network.

1.8 Results and Discussion

1.8.1 Multistatic radar scheduling investigation

In Chapter 5 IA is compared to GRASP for scheduling MSMSRNs. This comparison was made to show that IA is not computationally cumbersome and can provide equivalent performance to GRASP. Furthermore, a realistic tracking environment was employed to show how IA could be used to perform nimble scheduling. In the latter environment, three different scheduling modes were tested: no prediction, simple prediction and nimble scheduling.

Three Monte-Carlo simulations for a binary radar system were run to produce results, where targets are randomly placed and move along random trajectories. The first two simulation runs were used to vary the number of targets and the radar beam width, so as to compare IA to GRASP. The final simulations runs compared the three scheduling modes used along with IA scheduling against slow, fast and rapid accelerating targets.

While IA scheduling performed as well as GRASP in the binary radar simulations, indications are that it would be easier to adapt this scheduling algorithm to cater for more radars. IA is also more amenable to cope with a richer set of temporal constraints that may be imposed. Since GRASP is based on heuristics, its implementation is more adversely affected by changes in imposed constraints than IA.

IA guarantees that a maximum-length solution will be found for the target selections that are made as it performs local optimum examinations. Instead, conventional optimisation algorithms perform a global search of the solution space. Thus, combinations of IA with an optimisation algorithm should lead to more effective scheduling algorithms.

While the processing time required is within the real time available for both algorithms, adding many more iterations or more targets would make this difficult to sustain. Thus, while performing a global search is desirable, it would not always be possible. Employing parallel processing architectures such as multicore central and general-purpose graphical processing units could make the use of a global search technique practically feasible.

In most cases, the IA scheduling algorithm already provides good performance and only a small incremental improvement is possible by utilising a conventional optimisation

algorithm alongside IA. Thus, the performance given by the IA algorithm may be sufficient and it then provides a computationally effective solution to the radar network scheduling problem. The IA algorithm can be used as implemented in Matlab® for around 10 targets without requiring parallel processing.

IA path consistency can be run for 16 384 targets in about 0.5 s of execution time using the C programming language[17]. The research documented in Chapter 6 considers how parallel architectures can be leveraged to reduce the computation time required by IA path consistency. Doing so together with running the separate GRASP iterations in parallel may make it possible to produce better results within realistic scan durations.

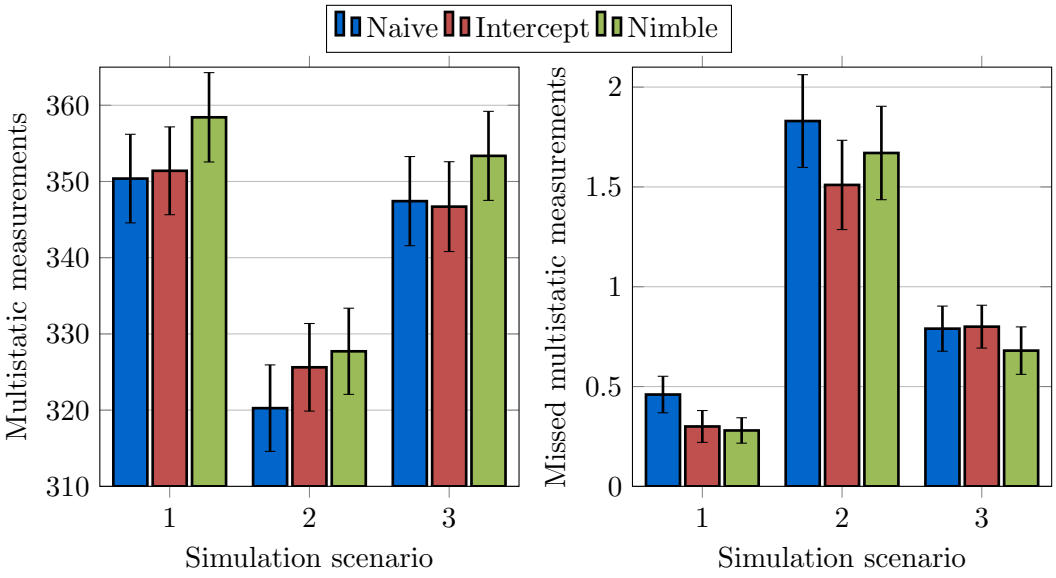


Figure 1.4: Results for nimble IA scheduling. These results were obtained by averaging over 900 nimble IA scheduling Monte-Carlo simulation runs. For each, the acceleration variance σ_a of the Singer random-walk motion model and initial target velocity, a Gaussian random value with mean μ_v, was set as follows: $\sigma_a = 2\,\mathrm{m}\cdot\mathrm{s}^{-2}$ and $\mu_v = 25\,\mathrm{m}\cdot\mathrm{s}^{-1}$; $\sigma_a = 2\,\mathrm{m}\cdot\mathrm{s}^{-2}$ and $\mu_v = 50\,\mathrm{m}\cdot\mathrm{s}^{-1}$; or $\sigma_a = 20\,\mathrm{m}\cdot\mathrm{s}^{-2}$ and $\mu_v = 25\,\mathrm{m}\cdot\mathrm{s}^{-1}$. Three MSMSRN scheduling modes were tested: once per scan using the current target geometry, thus, ignoring highly dynamic targets (series one in blue); once per scan simply predicting the location of the target using the fastest beam intercept time (series two in red); or at each intercept point where all target locations are predicted using the fastest beam intercept time as per Section 5.2.6 (series three in green).

Given the present results (refer to Figure 1.4) it would appear that nimble scheduling is not a major problem for a surveillance radar network that is tracking realistic targets that are neither ballistic nor supersonic. However, the simple prediction strategy does

[17]These are results from the author's C environment run on an Intel® Core™ i7-2670QM CPU at 2.2 GHz.

provide a minor improvement and may be worth considering if there is spare processing capacity in the system.

In the case of scheduling a multifunction radar, nimbleness is often significantly more important as usually either the platform housing the radar is moving quickly or it is likely that ballistic and supersonic targets may be encountered. Thus, the nimble IA scheduler could be adapted for use in multifunction radars and this would be an interesting topic to pursue.

Benefits of IA

IA as a constraint satisfaction problem adapted for the use of solving temporal reasoning is a suitable tool for solving the scheduling aspect of sensor management. It allows temporal constraints to be satisfied easily and by doing so it orders tasks temporally. Therefore, once IA has completed, the order tasks must be scheduled in is already known without requiring further processing.

Accommodating other types of constraints can be handled and simply requires defining a list of constraints (or operators) and a constraint propagation algorithm. The structure of the algorithm and method of solving the solution would remain the same.

More radars can easily be handled as tasks for each sensor are handled generically. Relationships between tasks of radars also follow a predefined set of rules that insures consistency. Other techniques rely on the structure of the problem to be efficient. For example, in GRASP the construction algorithm must take into account the number of radars to build the solutions piecewise. The construction will become more complex as more constraints need to be handled.

Drawbacks of IA

IA cannot easily handle task priorities as part of the algorithm, as it is a pure scheduling algorithm. Thus, it would not be as efficient as algorithms that can perform both task prioritisation and scheduling. Task priorities must be handled as a separate pre-processing step before using IA.

In the current form of IA, constraints between tasks are always only further constrained. Until techniques and algorithms are added to relax the constraints of the prob-

lem, IA would not efficiently be used to solve highly dynamic problems. Using IA in these scenarios would require reconstructing the IA network for each instance of time. Linking time instances to each other would require a high-level algorithm to set the boundaries of the individual time instances.

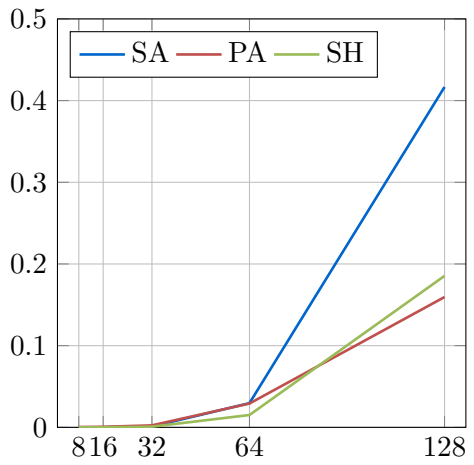
Reacting to future and past timeline events is not easily catered for as IA does not generate tasks. Task generation is separate from IA. It might be more cumbersome to handle the effect of past and future tasks on the current schedule with a more integrated algorithm. This is not to say it would be impossible but there would definitely be a challenge to achieve this with IA in the current proposed form.

1.8.2 Parallel processing architecture investigation

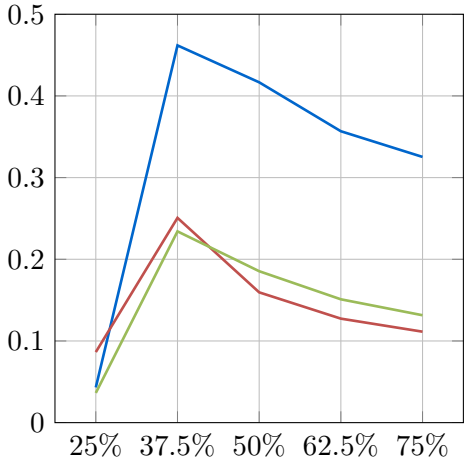
In Chapter 6 the execution time of parallel path consistency on GP-GPUs is investigated. Pseudo code for the novel OpenCL parallel path consistency algorithm implemented during this research is provided. The OpenCL algorithm was based on the original work of Ladkin and Maddux [57] and was optimised for execution on a GP-GPU. As the prevalence of parallel processing architectures such as MCPUs and GP-GPUs increases, it will become increasingly important to make use of parallel algorithms on these architectures.

The computation time of the author's OpenCL algorithm is compared to that of a serial algorithm running on a single core of a MCPU. This was accomplished by using three Monte-Carlo simulation runs, where the following three parameters were varied: number of intervals in the network, allowable IA operators and number of known IA relationships. The results capture the mean value of the computation time for the serial and parallel algorithms for each set-point of the variables as defined in each figure caption.

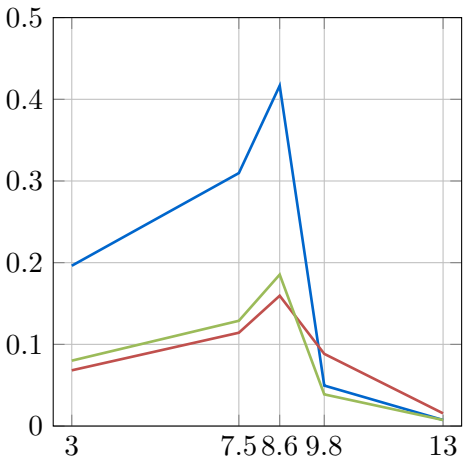
Given the results in Figure 1.5, it is clear that the parallel form of path consistency should be preferred for temporal constraint satisfaction problems where the network is more likely to be consistent. This is due to the fact that in most cases when the random scenarios generated were consistent the parallel path consistency led to a decrease in total execution time. While the author of this work was not able to verify a parallel path consistency algorithm making use of Hogge [59] constraint propagation, the results should still hold. For networks that may be inconsistent, the serial version provides better performance.



(a) Varying the number of intervals



(b) Varying the percentage of defined relations



(c) Varying the average number of operators

Figure 1.5: Mean execution time for parallel versus serial IA path consistency. The mean is taken over 1 000 Monte-Carlo runs for each point of the plots. Method $\mathrm{S}(n, d, s)$ was used to generate random consistent scenarios, as defined in Section 4.3.1. The blue line plots the results for serial path consistency using Allen constraint propagation (SA). The red line plots the results for parallel path consistency using Allen constraint propagation (PA). The green line plots the results for serial path consistency using Hogge constraint propagation (SH). The error bars, denoting the standard error of the mean for each of the points, are not depicted as they are too small to be visible at this scale. Figure 1.5(a) is obtained by varying the network size n between 8, 16, 32, 64 and 128. For these results the normalised degree of relations is $d/(n-1) = 0.5$ and the average constraint size is $s = 8.6$. Figure 1.5(b) is obtained by varying the normalised degree of relations $d/(n-1)$ between $\frac{1}{4}$, $\frac{3}{8}$, $\frac{1}{2}$, $\frac{5}{8}$ and $\frac{3}{4}$. For these results the network size is $n = 127$ and the average constraint size is $s = 8.6$. Figure 1.5(c) is obtained by varying the average constraint size n between 3, 7.5, 8.6, 9.8 and 13. For these results the network size is $n = 127$ and the normalised degree of relations is $d/(n-1) = 0.5$.

There are two strategies that can lead to quicker execution time in all cases. The first requires an idle CPU core and GP-GPU. In this case, launching both the serial and parallel algorithm at the same time is desirable, as whether or not the network is consistent one of the algorithms will have a lower execution time. Alternatively, it is possible first to run a few iterations on a CPU core. If no path-consistent scenario or inconsistency is found after a few iterations then the GP-GPU processing can be launched.

Considering only consistent networks the following should be a guide for using the parallel GP-GPU algorithm. The GP-GPU path consistency should can be preferred for all network sizes greater than 64 nodes. However, the GP-GPU memory space will limit the maximum network size that can be supported as there is typically less memory available to the GP-GPU. When the network is constrained with fewer than $\frac{1}{4}$ of the constraints being known then the serial path consistency can always be preferred. For all other cases, the GP-GPU version will provide a decrease in execution time. When the average constraint size is greater than 9.8 operators, then the serial path consistency can always be used. For average constraint sizes less than 9.8 operators the GP-GPU path consistency again provides better performance for consistent networks.

Further research would be required for other processing architectures such as MCPUs and FPGAs. However, from these results it does not appear that using MCPUs will lead to very good speed-ups. This is due to the fact that the thread launch time on these architectures becomes a significant contributor to the update processing time, whereas for GP-GPUs these are relatively small. Even with these small parallel overheads the GP-GPUs speed-up is currently only between 2 and 3 times as fast. Furthermore, even using optimisations to the OpenCL code to mitigate the above problem, MCPUs also have fewer computation cores than GP-GPUs. Nevertheless, in cases where the copy overheads dominate, the MCPU path consistency would benefit. Finally, it appears that FPGAs might provide better parallel execution for path consistency, as efficient path consistency circuits could reduce the update processing time [53].

1.9 Novel Contributions

The initial research performed by the author et al. to formulate how IA would be used to schedule mechanically steered multistatic radars was presented at the 14th International Conference on Information Fusion [1]. The work compared IA to a simple heuristic algorithm. IA scheduling was able to perform significantly better than the heuristic and it was shown that adapting IA to handle more complex scenarios is easier. The presentation was well received with critical questions being raised, and the most important of the questions revolved around the practical applicability of IA. This work has subsequently been published in the proceedings of the conference[18].

The current state of the art in sensor management up to 2014 is discussed in Chapter 2; this is a more current review than can be found in published literature[23]. The review considers the both of knowledge that has been dedicated towards solving multisensor scheduling. It focusses on what problems have been solved, modelling approaches for generating solutions and then on algorithms for each of the sub-tasks of sensor management. The sub-tasks are task generation, task prioritisation and sensor scheduling. Sensor management architectures are also highlighted. A paper generated from this content will be submitted to IEEE Transactions on Aerospace and Electronic Systems journal.

Furthermore, the theory of IA given in Section 3.3 has also been reviewed as part of the paper published by the Elsevier Information Fusion journal. The research outputs discussed in Chapter 5 were included in the publication. This submission validates the completeness of the research conducted towards answering the first two research questions through international peer review. The work focuses on the research conducted into the computational performance of IA in relation to GRASP as this algorithm is an established technique. The work presents three novel IA scheduling algorithms: the first two represent optimisations to the processing and storage of intervals within the IA network, while the third algorithm considers how the scheduler can be made to react to a fast changing surveillance picture

Next, the research outputs discussed in Chapter 6 will be converted into a research note. This submission will validate the completeness of the parallel processing research

[18] Available in IEEE Xplore at http://ieeexplore.ieee.org

conducted with answers to the final research question. The work focusses on a comparison of parallel IA implemented in OpenCL on GP-GPU against serial versions implemented on a CPU. Random networks are created using the constructs defined by prior research by [3]. Five different subclasses of IA are evaluated for completeness; these are Point Algebra, the Continuous End-Point subclass, the Pointizable subclass, the ORD-Horn subclass and complete IA. It is found that parallel processing on GP-GPUs can provide performance advantages for consistent networks only. The speed-up is in the order of 2 to 3 times that of the same serial algorithm. The network needs to be either larger than 64 intervals, have between 0.25 and 0.75 of the constraints unknown, or have interval relationships that contain between 3 and 9.8 of the basic IA constraints.

Appendix A gives a more detailed overview of these papers, which includes the abstract of each paper.

1.10 Conclusion

After all optimisations, IA was able to match the performance and computational requirements of GRASP. This was supported by the author et al.'s journal paper [2], which was published by the Elsevier Information Fusion journal. IA using the novel reduced-point algebra format can compete with GRASP both in terms of performance and execution time. Furthermore, IA does not require modifications to handle a richer set of constraints in temporal ordering. This is due to the fact that IA captures the intrinsic physical possibilities of time intervals in relation to others.

IA performs equally well as mathematical programming techniques. However, it is often not possible to use more rigorous techniques to solve sensor scheduling problems as real-time performance is necessary. Thus, IA should be preferred for its simplicity as it captures the intrinsic essence of temporal information between time intervals, which can easily represent sensor tasks. While only mechanically steered multistatic radar scheduling was investigated, there is nothing to stop IA from being used to schedule heterogeneous sensors of different types. Given the findings, IA should provide better performance within realistic sensor scheduling time frames, which answers the first and second research questions.

Using OpenCL the IA algorithms were adapted to be used with GP-GPU. An algorithm that can readily be used on all modern parallel processing architectures through the use of OpenCL programming language was generated.

In some cases, it was found that the parallel form has an improved execution time over that of the serial form. This was for the case of the parallel form executing on a GP-GPU and the serial form using a single CPU core. Furthermore, the types of IA networks which will benefit from parallel execution were enumerated. These findings prove the third research question, namely that IA can be used with modern parallel processing architectures and will thus remain relevant into the future.

All three research questions have been answered and each has yielded a positive result. IA can compete with other scheduling algorithms both in terms of performance and execution time. Furthermore, IA can make use of parallel processing architectures to speed up execution of total path consistency checks.

Finally, through the various publications, it was possible to interact with many international and South African radar researchers. The conference paper on IA and mechanically steered multistatic research was presented in Chicago to an international audience [1]. Each of the journal publications generated allowed three important research institutes, namely UCT, the CSIR and UP, to collaborate in the emerging field of multistatic radar. The IA scheduling algorithms and parallel forms of IA implemented allow all three institutes to utilise radar networks for further research in this important field. The algorithms can be used for future research in information fusion.

Chapter 2

State of the Art in Multisensor Management

2.1 Introduction

This chapter was the basis for the author's review paper to be submitted to the IEEE Transactions on Aerospace and Electronic Systems journal (Section A.2).

A Multisensor Data Fusion (MDF) system/Sensor Fusion System (SFS) is a type of Information Fusion System (IFS), which can fuse the information from multiple sensors to achieve some objective. Multisensor management deals with the task of queuing sensors to make measurements that best serve the mission that an IFS is intended to complete [12, 20, 66, 67]. In the case of a surveillance system, this means scheduling the sensors to gather the most pertinent information about the sensed area.

Recall from Section 1.3 that multisensor management can be subdivided into two distinct levels [20]: multisensor coordination and a sensor manager, which are depicted in Figure 2.1.

Multisensor management can also be sub-divided into three goals [22]: sensor task generation, sensor task prioritisation and sensor scheduling as mentioned in Section 1.3.

The first goal is to generate tasks for each sensor. Tasks should represent feasible actions for the sensor that aid the goals of an IFS. Examples include detection, tracking, classification, stopping emissions (to remain covert) and searching. Each of these tasks can have different attributes such as dwell time, transmit and receive frequency, repetition

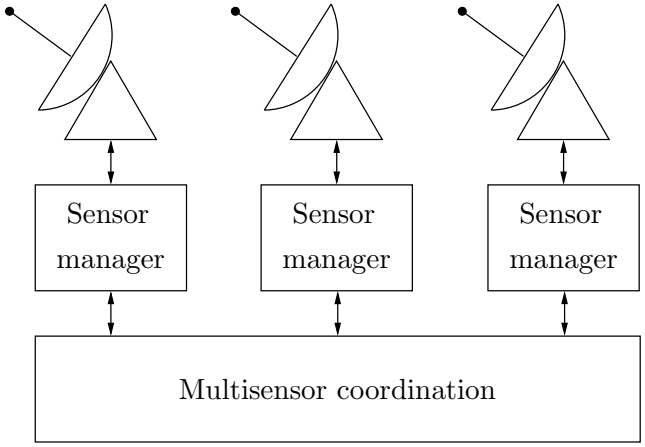


Figure 2.1: Architectural levels of a hybrid multisensor management solution. In a decentralised approach, the sensor coordination layer would also be distributed. A fully centralised approach would no longer require a sensor manager.

interval, number of repetitions, valid times, etc.

The next goal for the sensor manager is to prioritise the generated tasks. Each task provides a different information gain to the IFS. Some tasks may have very explicit valid times and durations.

Once tasks have been prioritised, the last goal for the sensor manager is to place the best set of tasks in the timeline of sensing actions for each sensor. Typically, it is impossible either to fill the timeline of a sensor completely, or to perform all the required sensor tasks due to the limitations of the IFS.

2.1.1 Categories of solutions

Traditionally, sensor management has been tasked to human operators [24]. There are now various solutions to the sensor management problem found in published literature. Figure 2.2 depicts the categories of solutions to multisensor management that were found in the literature. Four broad categories of algorithms that solve the three goals of sensor management are given by Musick and Malhotra [22]. Xiong and Svensson [7] have also looked at sensor management issues and approaches to solving these for an IFS. More recently, Hero and Cochran [13] have discussed high-level planning tools that solve multisensor management by modelling the problem as a decision process.

One of the most promising categories given by Musick and Malhotra [22] is information-theoretic approaches. Examples include Shannon's information entropy, Kullbach-Leibler

divergence, Fisher information matrices, covariance matrices of tracking filters, Mahalanobis distance, etc. These approaches leverage information that can be gained from analysing the measurement data from the sensor. An important aspect of information theoretic algorithms is that there is no bias in the algorithm for the different choices of tasks. For example, an information-theoretic approach has no bias towards search versus tracking in a multifunction radar. Information-theoretic approaches while showing promise should be used with care [68]. It is possible to use performance metric that does not adequately capture the problem [69, 70]. The near universal proxy argument on information theory has been refuted [68] and, thus, there can be cases where using an information theoretic approach will not lead to good sensing actions. For instance in tracking multiple targets, reducing only the variance of a multimodal Gaussian distribution can lead to a poor estimate of the state of both targets [68]. In this case, using information entropy would provide better results, as it is more important to differentiate the targets than necessarily reducing the variance in general.

Decision-theoretic approaches such as Bayesian networks are also covered by Musick and Malhotra [22]. One drawback of these approaches is that a decision is made up front about which tasks are the most favourable; this contrasts with the information-theoretic approaches. However, as shown by Hero and Cochran [13], these approaches can cater for longer-term planning within their usual framework. Notable approaches include the Markov Decision Process (MDP), where the Partially Observed Markov Decision Process (POMDP) is applicable to an IFS used in surveillance. Another approach is Multiarmed Bandits (MABs), borrowed from machine learning, which is also well-suited to surveillance as it can allow non-Markovian rewards and state progressions. More recently the theory of social choice has been advocated and shown to provide better computation and communication usage than Hierarchical Markov Decision Processes for Target Tracking (HMTT) [71]. Bayesian Decision Theory has been used to determine performance metrics for systems [72].

Mathematical programming techniques such as linear, non-linear and dynamic programming are also highlighted by Musick and Malhotra [22]. Recently, stochastic programming has also been widely applied. These techniques are readily applicable to solving Bayesian networks. Unfortunately, for all of these techniques to be optimal, a lot of pro-

cessing power is required. Thus, to allow a solution to be formed, the problem needs to be simplified. Furthermore, detailed analysis and mathematical rigour need to be performed before applying these techniques. Dynamic programming[1] for mathematical optimisation may be more suitable when the scheduling problem can be divided into smaller and simpler problems [22]. This is only true for sensor management problems where the decision criteria can be broken down into sub-problems. Then, in this case the solution using dynamic programming algorithms for mathematical optimisation, which subdivide the problem into smaller problems, is more efficient than approaches that do not exploit the subdivision of the problem.

Finally, computational/Artificial Intelligence (AI) techniques are also considered by Musick and Malhotra [22]. Most notably Artificial Neural Networks (ANNs) are highlighted. ANNs allow online learning. Unfortunately, this means that such techniques must first be calibrated before deploying the system. This has limited the use of ANNs in practical systems, which has led to less interest from academic researchers. AI optimisations such as genetic algorithms [73], particle swarm optimisation [74] and ant colony optimisation are more widely employed.

Modelling of sensor management systems and each of the three goals of multisensor management will be dealt with in Section 2.2. Exploiting both of these aspects is crucial to finding a holistic solution to the sensor management problem. Since modelling exists on a higher level than each of the sub-tasks, it will be covered first. Thereafter the three main goals will be dealt with in the order that they are performed in an actual sensor manager.

Lastly, Section 2.5 draws conclusions about the current state of research within the field and presents future directions that could be pursued.

[1]Dynamic programming as used here is not to be confused with dynamic programming languages. Here dynamic programming refers to the mathematical optimisation technique that can be applied to problems that can be compartmentalised.

2.2 Sensor Management Research

2.2.1 Sensor management problems

A wide variety of sensor management problems have been investigated in published literature. Early work considered the selection of an operational frequency band for an over-the-horizon radar by human operators [24].

Mechanically steered radars have received less attention due to the advent of phased-array, Digital Beam-Forming (DBF), Multiple-In Multiple-Out (MIMO) and Software-Defined Radars (SDRs). Research has looked at controlling the mode [75, 76], beam position [75, 77], power [76] and dwell time [76]. Surveillance radars have also been considered for ground [78, 79], airborne [37, 80], maritime [37, 81] and space [37, 82] deployments where controlling the mode between detection and non-cooperative target recognition is important.

Scheduling of naval, airborne and space phased-array radars has received much attention by various researchers [23], where controlling the radar mode (detect, track or image/classify) [15, 27, 37, 76, 83], due time of tasks [14, 39, 78, 79, 84], transmit power [15, 29, 76, 85, 86], beam position [82, 87–90], dwell time [27, 76, 91], detection threshold [87], sampling rate [86, 92], communication bandwidth [93], computation resources [86, 92, 93], platform motion [36] and task hand-over [94] are considered.

Published research has considered the control of a single generic sensor [95] and more frequently sensor networks in general [16]. Sensor network research includes literature that looks at controlling due time of tasks [95, 96], task selection [97], sensor selection [98–102], platform motion [103], transmit power [104] and tracking accuracy [33, 105] for any sensor. Others have focused on non-radar sensors such as selecting a transmitting sonar in a sonar network [26], synthetic aperture radars [41], infrared cameras [40, 106], video cameras [40], electronic-warfare receivers/communications intelligence surveillance [28, 40], human scouts [106], etc.

Control of radar networks containing both passive radars alongside and/or active radars, has also been investigated [15, 29, 32, 37, 41, 107]. Only recently has multiple AESA radar scheduling been addressed [108]. Multistatic/MIMO radars have also gar-

nered much interest where researchers have considered the placement of receivers [109], avoidance of Doppler blind zones [109], simultaneous transmission [1, 110], waveform selection [111–113], radar selection [35, 106, 112, 114, 115], transmit power [30, 116] and radar location [117].

Related fields include, but are not limited to: mission planning for aircraft [118], control of production machines [119–121], control of queuing networks [122], control of (wireless) communication networks [17, 123–125], intrusion detection in communication networks [126], inventory control [127], biometric access control [128], etc.

2.2.2 Sensor management modelling

There are two options to choose from when modelling a sensor manager [13]. The first choice is to make use of a myopic simplification and thus deal with only a very simple model of the past and the future. The alternative choice is to employ longer-term planning, which considers more historic information and generates longer-term predictions.

One benefit of using a myopic simplification is that solutions are mathematically tractable, and therefore easier and cheaper to solve [13]. Another reason for using a myopic simplification is to focus on one of the three goals of sensor management. In this case, it is best to have a simple model to ensure that solving and verifying the solution of the specific goal becomes easier. Naturally, once a solution is found, it then becomes important to enhance the results by considering longer-term planning to determine the utility of the solution.

One myopic approach that has a solid foundation in control theory is the control metaphor for software [28]. The idea is to treat the entire sensor manager as the 'plant'. The 'plant' therefore also includes the sensor and elements of the environment. Controllers are then designed to stabilise the 'plant'. A natural choice is the Proportional-Integral-Derivative (PID) controller.

Several techniques from the field of Bayesian networks are able to model sensor management [13]. Most notable is the MDP. Since an MDP requires complete observation of the system state, researchers in sensor management often have to turn to a POMDP. This is due to the fact that the sensors do not measure the true state of the target, and can only provide noisy measurements. A recent overview of POMDP applications to sensor

management is given by Hero and Cochran [13]. POMDPs have also been used to solve sensor management problems in various published literature [14, 15, 38, 80, 107, 127]. POMDPs can also be used to solve problems that can only be described using HMMs as long as the MAP estimator is used as a reward function [99].

MAB is a method that was developed in the machine-learning field of study [13]. The name of the method comes from the colloquial term for a casino lever-armed slot machine. Essentially, at each time step, there are multiple actions (arms) and the controller must decide which arm to play. Playing the lever of the slot machine (arm) only leads to a reward if the slot machine (bandit) pays out. For the case of tracking, a target is a bandit and the estimation of the target state is the action of playing the arm to receive a reward. Usually, for MABs, rewards are only gained for a single arm and all other rewards remain stationary. This, however, does not accurately depict the true nature of multiple target tracking, as targets not tracked can still move and their tracks must, therefore, be predicted. However, the effect of this can be reduced by having a sufficiently fast update cycle for the algorithm. MABs can be seen as a superset of MDPs, as an MAB can be modelled as an MDP. An overview of MAB applications to sensor management is given by Hero et al. [13]. Various types of MABs also feature frequently in sensor management literature [16, 17, 29, 96, 126, 129].

In many cases, the rewards obtained through sensor selection are not Markovian, meaning that the reward obtained is not only dependent on the current state but also on past and even possibly future states of the modelled sensor management system. Thus, non-Markovian methods have also been considered in the literature. However, it must be noted that these types of stochastic processes, known as a Non-Markovian Decision Processs (NMDPs), are notoriously difficult to solve. This can be seen from the state space explosion that occurs when converting an NMDP into an MDP. Furthermore, a more accurate model for a sensor management system is a Partially Observed Non-Markovian Decision Process (PONMDP), which leads to even more computational complexity. Most published literature using NMDPs comes from related fields [123–125] and more work needs to be done in sensor management in this domain, as the sensor management problem is often non-Markovian [36, 103].

2.2.3 Sensor task generation

Fully automatic generation of tasks by computers, to be performed by sensors, is by far the most difficult goal of sensor management to achieve. This is due to the fact that it not only requires creativity to think up a set of tasks but also requires sensor-specific domain knowledge. In the past, this was predominantly performed by the human operator [24].

Most current literature has focused on either completely predefining tasks [35, 95, 98, 121, 128, 130] or on only creating a set of tasks within a predefined framework [28, 33, 37, 77, 81]. Alternatives are to completely ignore this aspect of sensor management [16, 23, 25, 32, 124, 127], incorporate it within other aspects of the system [36, 78, 85, 92, 94, 112] (such as a number of target estimations for tracking) or to use random approaches [17, 126]. Solving this problem will require computers to have human cognition and, due to this need, has not been pursued as widely as the latter two goals of sensor management.

2.2.4 Sensor task prioritisation

Once tasks have been generated using the appropriate task generation algorithm, the sensor manager must decide on a prioritisation for each of the generated tasks [22]. Task priority is used to decide which tasks to discard when sensor resources are limited. As limited sensor resources are usually the case, most scheduling algorithms use the highest priority tasks first. Task prioritisation is a middle ground between creativity and pure computation. Deciding on the prioritisation requires creativity and domain knowledge to create a good strategy. Once the strategy has been determined, however, prioritising the tasks is pure mathematics. Computers have been designed to solve mathematical equations and excel at number crunching.

There are several approaches that can be followed for task prioritisation. As per task generation, task priorities can be predefined and several examples exist in the literature [81, 91, 94, 131, 132]. Heuristics allow a trade-off between completely predefining task priorities and online computation [28, 41, 118, 130, 133, 134]. Information-theoretic approaches exploit domain knowledge algorithmically to prioritise tasks in an unbiased manner [32, 35, 95, 104, 106, 135]. Decision-theoretic approaches allow choices about priorities to be made up front, while still allowing some room for manoeuvre online [14, 27, 29, 107, 136].

Artificial intelligence techniques attempt to mimic nature by providing computer-cognitive ways of prioritising [18, 23, 42, 43, 90, 128]. At other times this aspect is completely ignored [25, 77, 110, 121, 137, 138] and when this is the case, all tasks usually have equal priorities.

In certain literature, the task priorities are decided up front and not computed on line [81, 91, 94, 131, 132]. Occasionally, the task priorities are decided upon during the execution of the scheduling algorithm [23, 33, 76, 84, 93]. This has the benefit of potentially reducing the number of computations for task prioritisation. Within mathematical programming schedulers, the task priorities are decided by the objective function [34, 37, 83, 98, 127]. When using MDPs, the reward function is an alias for the task priorities and is usually an information-theoretic part of the MDP design [14, 38, 80, 107]. For MAB approaches, the reward function also represents task priorities [16, 17, 29, 124–126]. The reward function of an MAB is often approximated using the Gittin's indices, which define vectors for the long-term reward/valuation function. Similarly, the reward functions of NMDPs are another form of task prioritisation [36, 103].

2.2.5 Sensor scheduling

Only after a list of tasks and their associated priorities have been decided can the goal of scheduling tasks for the sensor be initiated [22][2]. Sensor scheduling treats the sensor resources as a timeline extending from now until the future. The goal is to fill the timeline with tasks such that the sensor is never idle. Idle time is uneconomical given that this time could have been better spent catching up on tasks that may later cause a bottleneck. Sensor scheduling is an easier task for computers to manage, as it essentially requires only simple calculations and checks. However, in the case of multisensor scheduling or scheduling sensors capable of multiple simultaneous tasks, the problem becomes more complex. Nevertheless, there is still room for creativity in scheduling sensors as this often leads to better solutions.

A good overview of types of algorithms for sensor scheduling can be found in the work of Musick and Malhotra [22], Xiong and Svensson [7] and Ding [23]. There are many approaches that can be followed to schedule sensors. Not all of them are directly applicable

[2]This section forms part of the introduction of the author et al.'s research paper published by the Elsevier Information Fusion journal (Section A.3).

to multisensor scheduling, which is definitely required for an IFS. However, most should be adaptable in some form to cope with the dimensionality increase caused by having multiple sensors. Early in the history of sensor management this task was performed by human operators, with only minimal assistance provided by the system that was being managed [24]. Next, heuristic approaches that captured much of the domain knowledge of human operators were employed. These approaches are still very popular today as they require minimal computation time [26, 41, 104, 130, 134, 135].

As research in the field broadens to encompass the other goals of sensor management, this aspect is sometimes handled intrinsically by the task prioritisation algorithms [28–30, 71, 85, 95, 101, 111, 126]. Some task prioritisation algorithms also fall in the decision-theoretic category [71]. This has the benefit of not wasting computation time if the mission of the IFS changes on a high-level, as instead of a timeline of tasks, the sensor manager is dealing with only the current task to schedule. On the other hand, if a centralised fusion centre is used and stops operating or communications are lost in a decentralised system, there will be no timeline of tasks to continue with in the interim. Simply doing arbitrary tasks during this time can have detrimental effects not only on both the optimality of the scheduling solution but also on the mission of the IFS as a whole.

When information-theoretic approaches to task prioritisation are used [32], there is a tendency to solve the scheduling problem through the use of optimisation algorithms. These optimisation algorithms fall into three categories: heuristic search techniques [102, 108, 115], mathematical programming search techniques [98, 99, 101, 117, 128] and artificial intelligence search techniques [37, 73, 74, 118, 139]. Simulation techniques are another possibility, where conceivable future timelines are investigated using multiple trials [38–40]. Other approaches use stochastic job scheduling [140], where service times and deadlines are modelled as random variables. The solution is found by finding the control input that maximises the expectation ($\mathrm{E}(\cdot)$) of the gain [140–142]. It should be noted that many of the solutions to POMDPs and MABs also fall into this category.

Sometimes random approaches are followed and these are frequently used as an Electronic Counter-Counter-Measure (ECCM) [41, 125, 135]. This aspect is rarely ignored but there have been some examples in the literature where this has been the case, as researchers have focused on other aspects of sensor management [13, 32, 77, 122, 138].

Artificial intelligence techniques have also been proposed in the past and have predominantly used reasoning/expert systems. Examples include fuzzy-set based reasoning [42, 43] and fuzzy decision trees [43] all within the context of an expert system. In these systems, the scheduling rules are captured by analysing linguistic rules of thumb provided by human sensor operators. More recently learning automata [100] have also been used to adaptively schedule sensors through an online learning process. Automata is an adaptive control technique, where the nodes the control policy is changed when the node is rewarded and penalised for control actions taken.

2.2.6 Sensor management architectures

There are various architectures used for creating a multisensor manager[3]. Traditionally, a centralised architecture has been used, where a central information fusion system feeds a centralised multisensor manager with information to control all sensors. Another possibility is a decentralised architecture, here typically both the information fusion system and the multisensor manager are distributed across each discrete sensor or suite of sensors [7, 18, 19, 100]. These architectures represent the current state of the art, as distributed problems are difficult to solve. A hybrid of these approaches has been proposed by various authors [20, 21], which usually consists of two or more distinct levels. On the lowest level, sensor management is distributed and must keep sensors busy and tune sensor parameters for high-level tasks. The higher-level sensor coordination is centralised and ensures that collectively the sensors are optimally achieving the sensor fusion system goals.

2.2.7 Identified gaps

Current literature has not considered the scheduling of Mechanically Steered Multistatic Surveillance Radar Networks (MSMSRNs). This provided an easy means for the research captured here to make a novel contribution to the field. While it would be nice to address all aspects of scheduling multistatic radar networks, the scope of the research needed to be adequately constrained. Thus, it was decided to first consider only the scheduling aspect, as IA a scheduling algorithm was also identified. Future work can build on the research

[3]This section forms part of the introduction of the author et al.'s research paper published by the Elsevier Information Fusion journal (Section A.3).

documented here by addressing other aspects.

2.3 Detailed Analysis of Sensor Scheduling Research

Referring back to Figure 1.1 of the DFIG DFM, resource management (which includes multisensor management) falls under the original JDL DFM level 4 called processes refinement. The key purpose of this level is to control not only the sensors that are attached to the MSDF system but also the deployment of forces, weapons and countermeasures. All aspects of MSDF that seeks to optimise the usage of the sensors belong on this level. Interactions between this level and all the lower levels can therefore be envisaged, as each lower level generates vital information about the sensed environment. Most published research, including this thesis, only considers the interaction between level 4 and the outputs of JDL DFM level 1 called object refinement and JDL DFM level 5 called cognitive refinement. The ideal multisensor management system would incorporate data from the higher-levels of the information fusion system including levels 2, 3 and 4 (the other functions of this level: refinement of the other levels and mission management). When a scheduler makes use of *data from the higher DFM levels* it indicates that data from levels 2–4 are also being used.

2.3.1 Heuristic approaches

Barbato and Giustiniani [81] define a heuristic scheduling algorithm. The scheduler is for a naval-based Active Electronically Scanned Array (AESA) Multifunction Radar (MFR), and must therefore deal with the complexity of a non-stationary platform. More specifically, it has to cater for the roll and pitch of the vessel caused by waves. The scheduler uses antenna, ship attitude and off-boresight scan angle of the AESA beam as inputs to create radar timeline schedules. Using this information along with the task priorities the scheduler packs tasks into the current scan cycle. Tasks not serviced in this cycle are added as higher priority tasks to the next scan cycle. This procedure ensures that tasks are not constantly starved of radar resources. However, the paper does not capture all the complexities involved. For instance, how the antenna, ship attitude and off-boresight scan angle of the AESA beam are used is never mentioned. The scheduling algorithm builds long-term plans which are static, which is good if the sensor manager cannot supply tasks for brief periods. The algorithm is not sufficiently validated and tested in the paper.

The graphs also do not convincingly prove that this is the best approach to follow. The scheduler is for a single radar and would need considerable modification for multisensor management. It is also coupled to radar sensors; thus, it would need to be redesigned for any other sensor type.

Ho and Chang [119] present four heuristic scheduling algorithms. As mentioned earlier, their research falls in the field of operation research, but focuses on the general problem of assigning multiple tasks to multiple homogeneous machines. Thus, their work should be adaptable to multisensor management, as long as the sensors used are homogeneous. Three of the heuristic algorithms follow a job-focused methodology. In this approach, tasks are taken according to their earliest due date and are assigned to the first available machine. Long-term plans, including all tasks, are formed but it should be possible to build shorter plans using the techniques. These heuristics make use of the unbalanced machine rule, where the total load of the machines is not necessarily kept balanced. This is applicable for tasks with a long processing time and tight due dates. Only the latter constraint is applicable to multisensor management. The last heuristic they propose makes use of a machine-focused methodology. In this approach, tasks are first assigned to one machine before considering the next. This approach is unlikely to be suitable for a multisensor manager, as any failures during execution of the algorithm would leave many sensors idle. Although the approaches are sound and follow scientific principles, they unfortunately do not compare to the original algorithms that were modified. The researchers discount more rigorous approaches and claim heuristic approaches are better due to their lower processing time.

Koulamas [120] also presents three heuristic scheduling algorithms in the operation research field. The problem solved is that of assigning multiple tasks to a single machine and therefore is a generic problem. Thus, the algorithms should be adaptable to single sensor management systems with similar constraints. They consider tasks with not only a due date but also an earliest start time. This is highly applicable to sensor management, where there is often an ideal period to perform a task. The first heuristic is a modified apparent urgency (AU) algorithm. This heuristic makes use of several iterations trying to minimise the apparent slack, average job processing time and machine processing time. The second heuristic makes use of the adjacent pairwise interchange (API). Jobs are

sorted according to a selected criterion, and thereafter the scheduling algorithm attempts to swap tasks so as to lower their early/tardy penalties. The last heuristic is based on previous work by Panwalkar, Smith and Koulamas [143] (PSK method). They adapt this algorithm in two ways to work with the scheduling time windows problem. Results show that their algorithm is not optimal in five percent of cases, when compared to a complete enumeration for a simplistic problem. As noted by the present author, it is well known that extrapolating such results to larger more complex problems is not valid. Testing for outlier tasks[4] is not performed, but would have been required for multisensor management. Since outlier tasks would need to be prioritised in the next scheduling interval, so as to ensure that these tasks get serviced eventually. It would have been helpful to compare the algorithm to a more rigorous approach. Despite this, the algorithms do appear to be promising, although some work would be required to adapt these algorithms to multisensor management. This is because the problem addressed is assigning jobs to machines, which is similar to multisensor management but not exactly the same. For example, mutlisensor management sometimes requires the coordination of multiple sensors.

Orman et al. [131] define one tracking and five scanning heuristic scheduling algorithms for a single AESA MFR. Radar tasks are treated as a transmit and receiver pair with a fixed delay between the tasks. Thus, tasks can be scheduled to have either segment occur within the dead time between the pairs. Their tracking heuristic scheduler attempts to place all tracking tasks to begin at their due starting times. If this is not possible, they move the tracking job forward and backwards in small step sizes. The step sizes are determined by the overlap between the conflicting task and the task to be scheduled. If this process fails, they search forward until a scheduling gap is found, thereby making the task late. The five scanning schedulers range from very restrictive[5] to very flexible, where the tracking schedule is modified freely. Only the scanning schedulers are compared; furthermore, they are only compared to each other. Thus it is hard to gauge if these are good schedulers in practise. Overall, they find that the most flexible scanning heuristics utilise the radar timeline the best, but there is a slight impact on tracking performance. Naturally, they find that the restrictive scanning schedule gives the best tracking performance. They

[4]Tasks that are too early or too late.
[5]Thus allowing no change in the tracking schedule.

conclude by stating that the radar operator should make the final choice between scanning heuristic schedulers. This, however, is no longer practical any more since systems of multiple radars can be heavily affected by local decisions. In addition, ignoring other algorithms only due to the fact that they do not use coupled tasks is not valid. Moreover, these schedulers are used for a single radar and would need considerable modification for multisensor management. The schedulers rely on a single timeline, whereas, multisensor management has a separate time line of tasks for each of the sensors that cooperate. The heuristics are also coupled to scanning sensors, and so they may not be applicable or would need to be redesigned for any other sensor type. For example, an Electronic Warfare Receiver (EWR) does not necessarily scan and instead the angle of arrival is estimated using other techniques.

Watson [88] specifies a very simple scheduler for an AESA radar. The scheduler attempts to fill the radar timeline after tasks have been prioritised. Tasks of lower priority levels are only serviced if the scheduler has exhausted all tasks of higher priority. The only rule that breaks this mould is the handling of confirmation of track tasks. These are performed as soon after a search task has generated and only when a detection as possible. Tasks are assigned in a one-to two-second time frame, where tracks are performed at the start of this time frame. Unfortunately, the scheduler is not fully disclosed as no pseudo code is given. Simulations performed do not justify their approach, and seem to be chosen ad hoc for a very particular scenario. Thus, applying the algorithm to multisensor management would be difficult and much rework would be required. It is also unclear whether simplistic approaches like this would yield good results for multiple sensors. Given more details on how the scheduler is envisaged to operate may mitigate these concerns.

Coetzee et al. [132] also investigated a heuristic scheduler for an AESA radar. They consider a scheduler that alternates between looking at a target (staring) and looking away (search). The algorithm must trade off estimation errors against searching for new targets. It must also make sure the update rate does not become too large. While the application is for AESA radar, it should be modifiable to different sensor types. One minor issue is that the algorithm appears to cater only for a single target. Furthermore, handling multiple radars in this way does not take into account the extra complexity. Simulations focus on the different Kalman filters used, and do not sufficiently test their scheduling

algorithm. A more advantageous approach would have been to test their algorithm against previous approaches. This algorithm would not be able to cater for the inertia required for mechanically steered sensors when the scan must occur far away from the target angles and still meet the tracking requirements.

Hansen et al. [93] make use of a modified Quality of service based Resource Allocation Model (Q-RAM) sensor manager. Q-RAM is a meta-heuristic algorithm as it makes use of analytical procedures; yet it is not mathematically rigorous [144]. The scheduler must control the central processing unit (CPU) usage, radar sampling bandwidth and energy utilisation for a four-faced AESA radar system. Their major contribution is to speed up computation of the task prioritisation algorithm. Their modified Q-RAM sensor management algorithm makes use of fewer computations. The Q-RAM approach seems to be adequate to handle radar resources for simplistic requirements. However, dealing with the complexities of the higher levels of the DFIG Data Fusion Model would be difficult. The approach requires having a definitive resource allocation function. However, higher level inputs are often subjective and their resource requirements are interconnected. For example, deciding on whether a target is a threat relies on multiple classification criteria. The combination of multiple tasks need to be factor in to decide whether a better estimate of threat is required. Deciding on which set of tasks will be serve this purpose is not straightforward and cannot easily be defined as simple objective functions. Afterwards a non-qualitative decision needs to be made on whether the target is threatening. Thus, only if a suitable resource allocation function could be defined and if the approach can be made more dynamic would it be suitable for handling higher-level information inputs. Q-RAM should, however, be a suitable choice for the low-level scheduler of a single radar system.

Lee et al. [92] make use of a Q-RAM sensor manager. An AESA radar system is used as the example problem to solve. For each tracking task, they control the sampling frequency and weigh this off against processing resources. They define service classes to deal with heterogeneous tasks that place different resource constraints on the radar. To reduce computation, only service classes on the diagonal of the loading per task type is handled. In their radar example, the heterogeneous tasks track different classes of targets (plane, missile, etc). So it seems that they only optimise for an equal number of targets

of different types; because this is rarely the case, their algorithm will not perform well in practise.

Miranda et al. [25] compare the Butler and Orman radar resource scheduling algorithms. Both of these algorithms are applied to a myopic sensor management problem of controlling the beam for an AESA radar. The algorithm trades off between tracking targets already known and searching for new targets to track. The simulations they perform show that under stressful conditions the Bulter algorithm makes better use of the radar resource timeline. In unloaded conditions both algorithms perform equally well. The paper flows very logically and comparisons are drawn under very realistic scenarios. It would appear that the Butler resource scheduler is a good choice for scheduling a radar in a myopic manager. However, handling higher-level DFM information and multiple sensors would be problematic. Similarly to the Q-RAM approach, there are no quick rules of thumb for the higher level information and often information from these levels can be contradictory. A simple rule would not be able to handle such contradictions by easily determining which information should be actioned on. Furthermore, if the myopic simplification were changed for longer-term planning, neither algorithm would be suitable. These heuristic approaches are built on the assumption that only a single scan of data is entered. Rules for how longer term data would be incorporated are difficult if not impossible to achieve, as the state space and corresponding rules will increase faster than the linear increase in data for each extra scan of data incorporated.

Xun et al. [145] make use of the control metaphor for software. The control loop tunes an electronic-warfare receiver (EWR) to a specific frequency and for a certain dwell time. This is done to detect all emitters in the field of regard of the EWR. Few sensor management problems make use of control theory; thus it is interesting to note that control theory can be applied to sensor management. The scheduler attempts to fill a timeline of tasks for the EWR by providing a certain level of QoS for each frequency and dwell time pair. A feedback controller on the scheduler ensures that the entire workload for the EWR is always unity. The feedback controller either extends or reduces the start times of the tasks, thereby inserting idle time into the sensor workload. This paper is also one of the few to address all three functions of a sensor manager[6]. The flow of ideas is logical in this

[6]Task creation, prioritisation and scheduling

paper, but unfortunately the control aspects of the paper are a little under-represented. Readers could have benefited from a more rigorous treatment of these in the paper. It would be interesting to see how this method can incorporate multiple sensors, higher-level DFIG Data Fusion Model information and a non-myopic scenario.

Gillespie et al. [91] propose a heuristic scheduling algorithm for an airborne AESA radar. The algorithm is an amalgamation of the best aspects of previously published heuristic algorithms. Twelve rules are defined that place constraints on the scheduler for tracking and searching. Simulations under varying degrees of load for the radar are presented and show that performance gradually tapers off. This algorithm is tightly coupled with a single radar and would be difficult to expand to multiple radars. To cater for multiple radars all the rules of the heuristic algorithm would need to be revisited. Furthermore, extra rules for cases that would not exist with a single radar, such as making multistatic measurements, would also need to be added. Sensors that do not perform search and tracking, or perform more tasks, would not be able to be handled. An emergent property of the algorithm is to limit searches close to bore-sight locations of the radar[7]. The authors claim that this emergent property is a desirable feature, but this would only be true at the beginning of conventional warfare. It would certainly not be true for unconventional warfare, where the enemy can attack from any vector. The algorithm is never compared to any of the algorithms from which ideas were borrowed.

Irci et al. [86] modify the Q-RAM sensor management algorithm. The modified Q-RAM is applied to a single AESA radar and controls the computation time, sampling frequency and transmitter power of the radar. Computation time is controlled by selecting the tracking algorithm used for each target. These authors define a track quality metric as a function of vectors, and thus collapse multidimensional space into a single dimension. Vector values in the track quality function are sorted such that the resulting curve is concave. They compare their algorithm to the Q-RAM approach followed by Lee et al. [92]. Their modifications give better performance, as they only evaluate the algorithm for targets with maximal derivative values. Their results show that 65% of the time they achieve better performance. The fact that the algorithm performs very poorly in other cases is not quantified or analysed. The only shortcomings of this body of work are its

[7]The locations directly in front of the aircraft.

brevity and the fact that the track quality metric is never wholly presented.

Miranda et al. [90] evaluate the Bulter and Orman heuristic algorithms. The problem of controlling the beam of an AESA radar is addressed. They present ideas for solving the task prioritisation problem as discussed in their previous paper. Butler and Orman are equivalent for low load but Bulter excels in high-load scenarios. This paper appears to be a direct follow-up of their conference paper. Adding the complexity of task prioritisation using AI techniques really adds value. Unfortunately, the schedulers are still only applicable to a single radar and would be difficult to modify. Heuristic algorithms are tightly coupled to the problem they solve. Again all rules of these heuristic algorithms are only applicable to the single radar case, optimising them for multiple radars would be necessary. Furthermore, extra rules would be required for tasks that can only be performed by multiple radars would also be required. The best way of incorporating these and not affecting the rest of the algorithms performance is not trivial.

Tharmarasa et al. [26] make use of several scheduling algorithms. They apply the algorithms to controlling a distributed network of sonar buoys. The heuristic scheduling algorithm controls the selection of an active set of sensors. An iterative local search is performed to derive nearly optimal solutions to this problem. A key point to raise is that they break down the multisensor management problem into several smaller problems to solve in parallel. The search algorithm they define is similar to GRASP with only one candidate solution generated and manipulated in each iteration. Unfortunately, testing of the algorithm is not performed adequately; they only compare the algorithm to a simple clustering method. The performance metric used is the Root-Mean Squared Error (RMSE) and this does not capture the full complexity of the problem they are attempting to solve. This only rewards tracking accuracy but does nothing to ensure that the area is also properly searched for new targets The entire scheduling algorithm is implemented as a heuristic flow chart, which performs each of the steps in sequence. At each point in the heuristic candidate solutions are checked for validity before proceeding.

Barbaresco et al. [27] define a simple heuristic scheduling algorithm. The algorithm is again applied only to a single AESA radar, but this time selects the radar waveform and target dwell time. Scheduling is performed according to the following policies: highest priority first, earliest deadline first, latest release time first and highest delay from due

time. Other scheduling aspects such as waveform and dwell time selection are not fully specified. In general, the paper neglects to explain how the algorithm works internally. This is possibly due to the secret nature of the work, which was done for the French Department of Defence at Thales. However, this does not justify the insufficient testing that was done to warrant the effectiveness of the algorithms. In short, the scheduling policy ideas are the only contribution in terms of scheduling.

Benaskeur et al. [94] define a higher-level architecture for multisensor management in a military setting. For sensor scheduling they consider target hand-off between several naval vessels. As targets move from one radar's surveillance region into the next, the scheduler of the radar on the neighbouring vessel is notified and tracking is commenced as soon as possible. This paper for the first time considers multisensor radar environments. The problem solved is generic and would need to be considered in any multisensor system. Their hierarchical approach to multisensor management should scale well as it breaks the system into manageable portions. One critique is that in one of the target hand-off scenarios tested, the glitch in tracking performance could have been eliminated by terminating tracks from the queueing sensor, only once the track of the queued sensors has similar or greater accuracy.

Baoquan et al. [79] define a heuristic multisensor scheduling algorithm. The algorithm forms part of their multisensor management algorithm for a constellation of missile early-warning radars. They make use of the Kalman filter to perform multitarget tracking, which is a suboptimal approach as the IMM performs better for multitarget tracking. However, using the Kalman filter allows the system to be computed on much more cheaper processing hardware which is desirable in cases where the cost of the system is important. Since their approach is to schedule a constellation of early-warning radars, cost of the system would an important factor. Their heuristic attempts to assign groups of sensors optimally to track targets. Sensor teams are assigned by forecasting the Kalman filter covariance matrix if the sensors were used. The team that performs the best is assigned to track the target. This, however, does not seem to cater for the multitarget case as each target is treated in isolation. The algorithm would need to consider global affects in scheduling the sensors in order to provide optimal tracking for all targets.

Gao et al. [115] consider the antenna placement problem for distributed statistical MIMO radar network. The goal is to increase the probability of detection for a number of targets by assigning the best transmit and receive antennas. They formulate the problem using Kullback-Lieber divergence as a performance metric for the entropy in the measurements of the targets. The goal is to minimise the entropy for each target by using the maximum number of transmitters and receivers. The authors also consider the impact of dwell time by relating this loosely to the desired probability of detection. How the final illumination time is selected is not present, although it is mentioned that this can neither be sufficiently long nor too short. They implement a heuristic search algorithm based on a modification of an algorithm in published research. The algorithm is compared to the branch-and-bound heuristic search algorithm. The results indicate that using both of their techniques produces the best results. This work would be difficult to extend to mechanically steered radars since each scan is treated in isolation and, thus, the beams of the radars need to be placed at random relatively quickly. Furthermore, the beam steering time and further reduction in performance for off-boresight scanning is not considered. Finally, since this is a detection strategy and all cells would eventually need to be scanned, the comparisons should have shown the processing time of the algorithms. Unfortunately, all simulations only considered two targets in the scene and no processing time performance results are given.

2.3.2 Mathematical programming

Castañon [80] consider the use of mathematical programming to solve sensor management of an airborne surveillance radar. They solve a POMDP problem in the case of classification of targets. Sensor inaccuracies and target aspect angles to the sensor lead to noisy classification data, which must be handled. They discovered that sensor management is an extension of the dynamic hypothesis testing problems studied earlier and is also related to search theory. Using the POMDP architecture, they can cater for longer-term planning by taking past and future actions into account. They use Lagrangian relaxation techniques to solve the non-convex reward function of the POMDP. Thus, they simplify the overall POMDP by decomposing it hierarchically into smaller POMDP problems. Stochastic programming is used to provide solutions of the smaller POMDP reward functions, which

in turn generates the current scheduling action for the sensor. The strong point of this paper is the fact that it considers factors beyond target tracking. Extension to multiple sensors should be possible, but will require modification of the POMDP. Though the approach appears to be computationally intensive, they do provide results for classification of 100 000 objects in under three seconds, which shows promise.

Krishnamurthy [107] looks at a myopic and non-myopic POMDP-based sensor manager. Scheduling is performed through stochastic dynamic programming for both. The first sensor manager uses a one-step lookahead and the second uses Lovejoy's approximation method to reduce computation. The scheduler controls both active and passive radars and decides whether to use active, passive or predictive sensing of the kinematic information of a target. The radars are used to detect incoming aircraft within three zones (far, medium and near). The objective of the sensor manager is to covertly track the aircraft, and thus active sensors should only be favoured in the 'near' zone. The results show the 'belief space' of the POMDP for two test scenarios. The 'belief space' represents the problem that given uncertain location of the aircraft a decision about which sensor to use must be made. The first is for a single active sensor case choosing between active measurement and prediction of the target. Active sensing is only preferred when the aircraft is near or there is uncertainty between the 'near' and the other two states. For the two-sensor test scenario, they present cost functions of using the various sensors as a function of probability of detection. It is shown that the cost function of their approach is lower than when using any of the sensors in isolation. The paper does not test other sensor management algorithms. One anomaly in his results, which is not explained, is that when there is uncertainty between the 'medium' and 'far' states, active sensing is selected. Intuitively, the correct choice (also shown in his paper) is to prefer prediction.

Guettier et al. [137] consider a wireless sensor network (WSN) subnet selection problem. The goal is to track a single target using the best subnet of the WSN. They make use of a myopic simplification that does not consider long-term trends. The problem is formulated using constraint programming and then solved using a hybrid search algorithm. Their scheduling algorithm is interruptible. If the algorithm is interrupted it will produce sub-optimal results. Regardless, the algorithm is not guaranteed to produce optimal results due to the searching nature. Decision criteria of the algorithm are fault

tolerance, latency of reliable broadcast in the subnet, as well as network and processing utilisation of the nodes. The hybrid search algorithms they employ include backtracking, repair algorithms and valued constraint satisfaction. Constraints included are allocation, connectivity, duration, fault-tolerance and maximising ad hoc sensor gain. The algorithm is not adequately explained; however, it shows promise for the wider field of multisensor management. Multisensor management is closely related to WSN management.

Kushner [123, 124] uses a Non-Markovian Multiarmed Bandit (NM-MAB) for his communications management problem. The problem solved is stability in a queueing network of distributed wireless nodes. Due to the non-Markovian nature of the queues, the existence of a stabilising policy is not obvious. Thus, he defines stability using a uniform mean recurrence time. Further reasons for requiring non-Markovian reward is to ensure robustness to changes in the inputs of the system. He uses the perturbed first-order Lyapunov function to solve the stability problem. The Lyapunov and perturbation functions must be designed for their example problem. The perturbed Lyapunov function can then be solved using stochastic dynamic programming, which is not performed as part of the paper. Instead theorems for stability are proven using the function. While the technique used should be adaptable to multisensor management it, may be very computationally costly. NM-MAB describes most multisensor management problems accurately, as the Markovian assumption is not valid when targets respond to sensing actions. The assumptions made are very realistic, apart from the assumption of an infinite buffer, which in a distributed realtime system is not achievable.

Walker et al. [110] look at the problem of managing multiple multistatic radars. The goal is to improve the detection of a single target at all the receivers by using constructive interference of signals. Thus, the transmission from the multiple receivers must be accurately coordinated by the scheduler. They define simple mathematical equations, which are then solved in a distributed fashion. Communication of the results in the network ensures that consensus of transmit times is reached. The scheduling algorithm uses two steps: first the simultaneous transmissions are scheduled and then time-division scheduling for the sampling of received waveforms is performed. The paper delves into the communications requirements around the scheduler, which is an appropriate requirement in a realistic system. The communication delays place a constraint on the maximum Pulse Repetition

Interval (PRI) that the radars can handle. They analyse the PRI under a varying number of nodes, receiving frequencies and maximum radar range. Then they compare their tracking performance to a simple target hand-off scenario (similar to Benaskeur and Rhéaume [21]). Comparisons are also drawn for the performance of their algorithm based on the receive power at each receiver, and this is always seen to be greater for their algorithm. The paper offers a solution that is more effective than simple target hand-off, but it requires multistatic receivers, which are not always available.

Saraf and Slater [78] define an aircraft management system for Air Traffic Controllers (ATC), which includes the final destination airport control. The problem is formulated using an objective function that has several constraints. The constraints ensure that aircraft dynamics are never violated and that aircraft are always a fixed distance from each other. The scheduling algorithm then builds a search tree where each node represents one of the aircraft. Each level of the tree represents an arrival slot in the sequence of aircraft arrivals at the airport. Nodes, therefore, store the aircraft number, parent nodes and a node number. The search tree is initialised and a heuristic algorithm is then defined to add, remove and move around nodes within the tree. After aircraft enter the scheduling horizon, their position in the arrival sequence for each ATC/airport is fixed. Simulations are done comparing the algorithm to a simple first-come first-served algorithm (currently used) and shows better results. The algorithm is also found to execute in 10 seconds at the runway radar using Matlab and its realtime workshop toolbox. One critique is that the authors define a new meta-heuristic search but do not check for relevant algorithms in the literature. Comparisons are only drawn to a very simple algorithm, so it is difficult to gauge how well it will perform. The paper does not include a sufficient number of graphs, and results are only given in tabular format. While speed-up of the algorithm is mentioned, converting the algorithm into a machine language instead of an interpreted language is never mentioned, which seems to be the best option for speeding up the algorithm.

Sutton et al. [84] define sensor management algorithms for tracking space debris with a single AESA ground-based radar. A catalogue of space debris must be maintained with accurate tracking information. As there is a window of best tracking accuracy, two schedulers are defined: one to track on the peak where tracking accuracy is likely to be the best and another where a more relaxed approach is followed. Stochastic search algorithms

are employed for both scheduler builders. They consider adaptive priority weighting of their expected success sum function. Also considered are bias functions for the priorities. Five bias functions are used to weight different priorities. Results show great improvement for the adaptive priorities, although no algorithm devised could track all objects within the catalogue. At least 4.2% of requests were statistically unlikely to be fulfilled. It should be mentioned that a realistic space debris data set is used for testing. The bias functions generate a Pareto front in the function of filling the catalogue versus mean days without updates. Adaptive priorities tend to fill more catalogue requests, thereby increasing the mean days without updates. The algorithms were previously defined and were not designed by the authors. They appear to be promising to use in multisensor management; however, the more realtime requirements of multisensor systems might make the algorithms impossible to solve optimally. The adaptive techniques effectively turn the myopic approach into a non-myopic approach by incorporating past information into the current schedule to adapt the target priorities.

Tharmarasa et al. [26] define several mathematical programming schedulers. They look at the problem of managing a distributed network of sonar buoys. The entire scheduling problem is a multiple objective optimisation problem and is broken down into smaller more tractable problems. The sensor scheduling algorithms in this paper must control the selection of local fusion centre (LFC), frequency allocation to the sensor buoys and transmit power for the active buoys. Sensors are activated using a heuristic algorithm. Measurements formed must first be processed by a LFC and integer programming is used to solve this problem. The use of integer programming lies in the fact that there is a finite discrete number of local fusion centres. Frequency allocation must be performed such that nodes do not interfere with each other when communicating with the LFC. However, the same is likely to be true for the actual sonar pings but the system they use seems to be passive sonar. The frequency allocation is solved using linear integer programming, since the transmitters and receivers can only be tuned to a finite discrete number of frequencies. The linear simplification is due to the objective function only being linear in terms of the frequency assignment. Finally, the transmit power of the sensors must be controlled and is determined by the results of a linear convex programming algorithm called CPLEX. The entire scheduling algorithm is implemented as a heuristic flow chart that performs each

of the steps in sequence. There are checks for validity of the solutions at each point in the heuristic. As the only performance metric tested is the tracking accuracy, it is hard to determine whether their frequency and power allocation strategies work well. A metric that considers how frequently sensors jam each other by using the wrong frequency or too much power would be required. Also, metrics on sensor lifetime would further test if the power allocation is optimal.

Barbosa and Chong [15] look at scheduling multiple radars. Selection of transmit waveform as well as the active sensing platform is the task of the sensor manager. The idea is to have only a single active sensing platform, as this is more desirable in modern warfare as it reduces the threat to all platforms. This seems to be uneconomical use of all the other active sensors, seeing that they should be present on every modern military platform. It seems a more effective approach would have been to select a subset of active sensors. Nevertheless, the sensor scheduling algorithm makes use of a nominal belief-state optimisation. As the threat posed to all platforms is modelled using a POMDP, the actual state of the MDP is unknown. Their approach is known as receding time horizon control (RTHC). First Bellman's optimality principle is used to derive an approximate the solution. Then they use Q-value approximation which is usually solved using Monte-Carlo approximation techniques. Due to the computational intensity of the Monte-Carlo approximation techniques they opt for nominal-belief state approximation. This is essentially a Maximum A Posteriori (MAP) estimator of the state the POMDP is believed to be in, given a chosen control sequence. Interestingly their results only show the root-mean squared error in tracking position and do not measure a threat for each platform.

Krishnamurthy and Djonin [14] look at another non-myopic sensor management problem. They make use of a POMDP to model the problem of selecting a target to measure by a single AESA radar. Owing to the computational intensiveness of POMDP, they hierarchically divide the sensor manager into a macro- and micro-level manager. This approach is in agreement with the architecture from Blackman and Popoli [20]. The macro-level manager determines the next target to track, whilst the micro-level manager decides when to stop tracking the current target. Each of these problems is a much simpler POMDP to solve. The scheduling algorithm in this case makes use of stochastic dynamic programming to determine the maximum reward function value. Therefore, how long the current target

should be tracked can be decided. Though it is not solved in the paper, they prove that the macro-level manager can be solved using a concave objective function over a convex polytope. It is claimed that target measurements between different targets are correlated; however, this claim is never substantiated. This may be due to the fact that targets belong to a single enemy. Stochastic dynamic programming is very costly computationally, due to the exponential growth in the state space for each independent variable added to the state vector. However, approximation techniques can be used to mitigate this problem at the cost of performance. Also, while the POMDP is simple for a single sensor, it may become intractable once more when expanding this algorithm to multisensor management.

Manjunath et al. [112] consider the problem of sensor and waveform selection in a multiple-in multiple-out (MIMO) radar system. They consider two sub-problems within the MIMO management problem. First, they consider minimising the number of radars used to provide a fixed mean-squared error (MSE) in target localisation. They find that this problem is a mixed Boolean convex problem. Secondly, they consider minimising the MSE with constraints on the maximum number of transmitters and receivers that can be selected. This problem is a convex optimisation problem that can be solved using branch and bound searches. The waveform selection problem is defined as a search over a finite set of waveforms for each transmitter. They choose the cost function to minimise the trace of the tracking error covariance matrix. Furthermore, the waveform selection is found to be a quadratic optimisation problem. Hence, this problem can therefore be solved using quadratic programming. The approach is sound and provides improved tracking accuracy. The solution would be difficult to augment with other information from the various levels of the DFIG Data Fusion Model. Since their approach relies on a simplistic cost function and data from the higher levels of the DFIG Data Fusion Model typically cannot easily be reduced in this way. Furthermore, the information is often incomplete and contradictory. The computational complexity of their scheduling algorithms makes adapting their solution to the non-myopic case problematic. Tests to compare against conventional algorithms should show the value of their approach.

Thunemann et al. [33] give an overview of resource management algorithms found in the literature up until 2009. They differentiate papers based on their degree of mathematical rigour and decentralisation. In the centralised category they mention: set par-

titioning, GRASP and heuristic algorithms. For decentralised algorithms, they mention the Lagrangian dual problem solved by a bundle algorithm, spatial decomposition, matrix decomposition, standard negotiation algorithm and the quorum algorithms. Though hybrid approaches are mentioned no algorithms are cited. They determine that algorithms closer to the best case make use of more mathematical rigour and that decentralised algorithms use fewer resources. It would have been beneficial if these claims were tested with accompanying graphs and discussions.

Huang and Ahmed [127] solve the problem of inventory control, which is a common production and logistics problem. Sufficient conditions are presented for optimal solutions to the capacity problem using stochastic programming. The assumptions made are that: there is a single resource, there is linear cost and there are discrete distributions for cost and demand data (thus non-Markovian and non-stationary). They also prove that there is a planning horizon for such problems. This paper considers non-Markovian non-stationary stochastic process (NM-NS-SP) making it interesting for multisensor management researchers. NM-NS-SP most closely models a SFS which incorporates all aspects of the DFIG Data Fusion Model. However, the assumption of a single resource and linear cost would have to be lifted. The paper only gives proofs, so much more work would have to be done before applying NM-NS-SP. There are likely to be many complexity hurdles in such an application, and suitable approximations would be required to reduce the computations.

Rajkumar et al. [121] make use of GRASP to solve the problem of assigning multiple tasks to multiple machines. In the paper, tasks consist of multiple operations that must be performed in sequence. This is similar to the radar or active-sonar tracking problem, which requires a transmit period followed by a receive period. Tasks must be routed and scheduled on the machines. They test several criteria to minimise: completion time of all the jobs, maximal machine workload and total workload of all machines. GRASP, a meta-heuristic from integer programming, is used to solve the scheduling problem. GRASP first constructs solutions from a restricted candidate list (RCL). Then, a local search is performed around the candidate solutions for better solutions. A linear bias function is applied to the search procedure, which in this case is a non-standard GRASP search procedure. Simulations show that the biased GRASP converges on a solution in much

fewer iterations than a genetic algorithm (GA). GRASP algorithm should be useful in solving objective functions for multisensor management.

Tharmarasa et al. [35] define an adaptive search procedure, which is very similar to GRASP, to schedule multiple multistatic radars (in a WSN). Local fusion centres (LFCs) must select their receiving nodes from the set of all receivers. Fusion centres share information via tracklet fusion (uncorrelated unlike track fusion). To further ensure data from different LFCs are not correlated, an LFC must not select a receiver used by any other LFC. The sensor manager must improve the tracking accuracies of existing targets; detect new targets quickly and accurately; and preserve the lifetime of the sensors. A GRASP-like search procedure is formulated whereby each fusion centre makes an initial selection in turn. Sensors are added by continuing to select receivers until either each LFC is saturated or all sensors are selected. In each cycle every LFC selects only one sensor then a local search is performed to try to find a better solution. The novelty of this paper is not in the actual algorithm but rather in the number of factors that are optimised. Taking non-myopic planning into account would be the next step for this algorithm.

Narykov et al. [108] looks at scheduling two AESA radars tracking a single target. A linear Kalman filter is employed despite the non-linear motion of the target, as it is assumed that the target only moves with constant velocity. The measurement model takes into account signal-to-noise losses for both beam pointing inaccuracy and target range signal degradation typical to radar sensors. However, the measurement model does not incorporate off-bore-sight beam distortions that are normal in AESA radars. The radar management algorithm assumes that only one radar should measure the target and attempts to balance the sensor load with measurement accuracies. Non-linear optimisation is applied to solve the sensor load first based on the target dwell time, which is related to the pulse width and signal delay. Using the selected loads a global search is then performed to find the configuration that minimises the measurement error. Unfortunately, given that the simulations are for only a single target, the sensor load is always minuscule and selection of the radar appears to be only dependent on the measurement capability.

Nguyen et al. [146] consider the waveform selection for multistatic radars tracking a single target. They use an extended Kalman filter and simple track combination through a weighted sum. The scheduler then minimises the mean squared error of the track by

selecting an appropriate waveform, this requires reducing the trace of the covariance matrix of the tracking filter. Since there is clutter in their simulations, the predicted covariance matrix cannot be pre-computed for each waveform and instead needs to be estimated. Here the Cramér-Rao Lower Bound (CRLB) for bistatic radars is employed along with Monte-Carlo integration to determine the degradation factor caused by clutter. Unfortunately, no scheduling algorithm is presented and a brute-force search of all the waveforms is instead employed. Thus, the only contributions for this paper are the CRLB for bistatic radars based on bistatic range and bisector velocity and the tracking metric for bistatic radars based on this CRLB. This paper could have benefited from aligning itself closer to traditional fusion approaches, such as using either measurement or track fusion. Not doing so limits the scope of the work somewhat, as the tracking metric is only applicable for the track combination that is not used elsewhere.

2.3.3 Artificial intelligence

Popoli and Blackman [42] make use of a fuzzy-set expert system to control an AESA radar. Simple rules of thumb are transcribed to decided on search, track or identify (Secondary Surveillance Radar (SSR)/Identify Friend or Foe (IFF)). The rule system gives good results for a single simplistic simulation environment. Fuzzy rules are sometimes dependent on each other, however, this is not the case in system employed by the authors. Expert systems have seen limited use as they require a good understand of the operator doctrine. Changes in doctrine would require the capturing of rules to be repeated. Operators of advanced systems are however not comfortable changing the rules sets on their own. Nevertheless, expert systems are a good choice for heuristic task prioritisation. They employ a definitive method of capturing the way humans interact with sensors. Unfortunately, without proper tuning they cannot readily perform sensor scheduling well, as typically operators do not necessarily have all the information to make scheduling decisions.

Molina López et al. [43] also use a knowledge based system (or expert system). Fuzzy set theory and possibility theory are used to decide on system level tasks. Situation Assessment and object refinement inputs are used. The decision is reached using fuzzy necessity and a desired figure of merit. Low-level search trees are constructed from sensor configurations that can perform a system level task. Tables of accuracy translate from

high-level tasks to low-level tasks. A heuristic then assigns tasks to sensors by searching the tree for tasks that minimises sensor load and maximises suitability. This is an interesting AI heuristic technique but is not able to easily take into account changing constraints, as these would necessitate changes to many of the operator rules captured in the fuzzy-sets.

Molina López et al. [18] look at a hybridized sensor management solution. This solution employs fuzzy reasoning to reach decisions. A fusion centre creates the initial sensor tasks, which can then be completed by a single sensor or a group. Consensus is reach via sharing fuzzy priorities amongst sensors. Therefore, the fusion centre is centralised but the management is decentralised. This sensor management approach is similar to the authors' prior conference paper; however, it is now being applied to a hybrid sensor management solution. Thus, expert systems and fuzzy-set reasoning can be used in a decentralised management approach. This is useful when sensors are able to make decisions and operate in isolation.

Veeramachaneni and Osadciw [76] look at managing multiple mechanically steered radars. Though the radars are AESA radars this fact is never exploited, and therefore the problem they solve is equivalent for conventional mechanically scanned radars. This paper provides the closest link to the research conducted. In their case, the sensor manager must control the sensor dwell time per target (presumably by slowing down the angular rotation), as well as the power and surveillance range. They can either track a target or survey the area using a search waveform. Search can still be performed in the near field when tracking a target. They do not exploit the fact that it is also possible to search in the far field, through adequate pre-scheduling of the AESA beam. Unlike previous papers, the authors do not treat the search aspect of surveillance as secondary and weigh off probability of detection against tracking accuracy. They make use of a particle swarm optimiser (PSO) to determine the schedule for the radars, an artificial intelligence technique that is inspired by the swarming nature found in creatures such as bees. Results for their algorithm show that they always cover at least 50% of the area. Probability of detection is also always above 60% and increases as beams are assigned to far-out targets more frequently. This result could be improved upon by considering far-field search waveforms as discussed. Nevertheless, the paper is of good quality and the results are well presented. More work would be required to handle a non-myopic expansion of the sample problem.

Martin and Newman [36] also look into using a particle swarm optimiser (PSO). They control the platform course for a squadron of unmanned aerial vehicles (UAV). This problem is closely related to multisensor management, and thus should be easy to translate under equivalent constraints. They make use of a receding time horizon controller (RTHC) through adequate modelling of the problem using a non-Markovian decision process (NMDP). UAV motion is modelled using a simpler manoeuvre automaton. A non-Markovian reward is required as the UAVs are rewarded for visiting previously viewed terrain from different angles. Therefore, this requires taking all previously visited states into account. The PSO algorithm solves the scheduling problem of the NMRDP, and, in this case, makes use of suffix tries to reduce computation. Suffix tries attempt to find an optimal starting sequence for the UAVs and freeze it for several iterations of the PSO. Unfortunately, they only compare the algorithm to a simple open-loop loiter algorithm. PSO is definitely viable for taking into account many factors for multisensor management. However, it remains to be seen how well it converges and at what differential computational cost, and against other artificial intelligent and mathematical programming search techniques.

Newman et al. [103] again look at using a particle swarm optimiser (PSO). In terms of the scheduling problem solved it is equivalent to the previous paper by Martin. The novelty added by the paper is the fact that they use multiple hypothesis tracking (MHT) instead of the extended Kalman filter (EKF) of the previous paper. MHT leads to better target tracking than the EKF but requires more computation time. It is good to see that their approach is amenable to modern tracking techniques.

Ralph and Jones [118] consider the problem of automated mission planning for a fleet of aircraft. Thus they assign tasks: either surveillance requests or prosecution requests to the aircraft that will be most capable of performing the task. As aircraft contain different weapons platforms, some may be more suited to solving the problem. This scheduling problem is shown to be non-deterministic polynomial time complete. They present a multiple objective search algorithm known as simulated annealing. Simulated annealing is a probabilistic search algorithm that mimics the annealing of metals. They define six heuristics to make an initial selection for the simulated annealing. They only compare the simulated annealing approach using their six heuristics. Unfortunately, this does not

show whether simulated annealing, in general, is a good approach for scheduling. Testing against more conventional search algorithms and other AI techniques would be beneficial. While long-term effects are not catered for it should be possible to use simulated annealing in conjunction with non-myopic approaches.

Shea et al. [37] make use of a GA to schedule several radar systems. They look at an airborne ground-moving-target-indicator (GMTI) radar, ship-based airborne-MTI (AMTI) radar and space-based constellation of optical sensors. While this paper focuses on automatic task generation, they make use of a GAs from a prior paper to solve the scheduling problem. Tasks are prioritised using a benefit function and the GAs tries to schedule tasks to maximise the overall benefit. They compare the GAs approach to a baseline wide-area search algorithm. Unfortunately the baseline algorithm used is not specified, and therefore it is hard to interpret whether the GAs scheduler is a good approach. Nevertheless, GAs should be compatible with non-myopic planning as well, though in the paper a myopic simplification is performed.

2.3.4 Statistical techniques

He and Chong [38] look at the general multisensor problem of selecting a sensor to make measurements of a target, given sensor usage costs. They make use of a statistical technique known as Monte-Carlo simulations to track the belief state of the POMDP. Usually the analytical hidden Markov model (HMM) filter is used. The key attribute of statistical techniques is that they are simple to compute and therefore allow more sophistication in the modelling of the system. While they are simple, they do however utilise a lot of processing power, a fact that is overlooked by the researchers. To schedule the sensors, they make use of quality value (Q-value) approximation combined with a particle filter (PF). Particle filtering is a Monte-Carlo technique that is used to estimate non-linear dynamic Bayesian models, in this case the HMM. Thus, they estimate the cost of each sensing action using Q-value approximation along with a PF. The PF provides the Q-value approximation with a set of POMDP belief states. In return, Q-value delivers updated actions to the PF belief state estimation. Simulations show that their policy performs slightly worse than the closest point of approach policy in terms of tracking accuracy. It does, however, outperform such an approach in terms of power usage for the sensors.

These results are for the case where one sensor consumes more power than the other two. Their approach provides better tracking accuracy than the comparison algorithm when all sensors consume equal power, but only if there is a more accurate sensor available. It also consumes the same amount of total power from the network. Exploring individual sensor power usage seems to be overlooked, as it seems clear from their results that the accurate sensor is likely have a depleted battery. Despite this, the technique gives merit to POMDP approaches, as reducing computation time and complexity for POMDP is an unsolved problem. Monte-Carlo techniques also lend themselves well to parallel processing architectures.

Kreucher et al. [39] also consider Monte-Carlo techniques for sensor scheduling. In this case, they look at deciding when to measure targets with multiple UAV-mounted radars. Instead of a more general non-myoptic approach, a two-step look-ahead is performed to limit the computation. The two-step look-ahead is a slightly longer-term myopic sensor management algorithm. Tasks are prioritised using the Rényi entropy between the predicted probability distribution and update probability distribution. Two Monte-Carlo search techniques are presented to solve the scheduling problem. The first makes use of the two-step look-ahead and performs simulations to determine the best set of actions, or those giving the largest Rényi entropy. The second method adaptively searches for the best path to follow. It approximates the gain by looking down paths a few times. Both techniques should easily be adaptable to non-myopic approaches. However, they only compare their algorithms to each other and find that method two converges faster. This is due to the fact that it does not interrogate as many paths before finding a near-optimal solution. Incorporating more information from the different levels of the DFIG Data Fusion Model (DFM) should be possible, as this merely translates into a new task prioritisation scheme.

Wang et al. [125] make use of a random scheduling algorithm, where the probabilities of selecting different actions is weighted by their task-prioritisation algorithm. Their application area is cognitive radio links in the presence of a single intelligent jammer. Secondary users make use of spectrum assigned to a primary user and are actively jammed by the jammer. The problem is to get information sent from the transmitter to the receiver despite the efforts of the jammer. Nevertheless, the scheduling algorithm should be applicable to multisensor management and may help in scenarios where jamming is

present as well. The problem is modelled as a non-stochastic MAB, which seems counter-intuitive. Perhaps the intention was to imply a NM-NS-MAB. The non-stochastic name does, however, appear to relate to the fact that channel statistics are unknown. Thus, the algorithm does not take into account any prior knowledge of the primary user.

Farokhi and Johansson [142] consider the problem of scheduling a network of sensors. As decisions are made in a distributed fashion the problem must be solved using stochastic control theory such as those found in published books on this topic [140]. They model the sensor scheduling policy as a continuous-time Markov chain. From this an iterative differential equation is generated that models the control inputs as a Poisson counter process. An optimisation problem is then formulated as the expectation of impact the control decisions will have on the sensor operation. The optimal control policy is then derived using stochastic mathematics and partial differential equations.

2.3.5 Identified gaps

IA is a scheduling algorithm that is used in robotics and gene sequencing. No published research has determined how IA would be applied to the scheduling of multiple sensors. It was thus decided that applying this algorithm to the scheduling of multistatic radar networks would add another degree of novelty to the research. The research also considers how IA interacts with the other aspects, including the chosen architecture of the sensor management solution.

2.4 Impact on this Research

For the research conducted, the focus was on the scheduling aspect of multisensor management. On the multisensor management modelling level, to reduce the scope of the research, it was decided to opt for a myopic simplification for the multisensor management aspect. Non-myopic approaches are very interesting and are definitely the future of multisensor management. However, as work conducted at this level is also cutting edge, it is better suited for an independent study focusing on this aspect alone.

The myopic simplification, in the case of the research described in this document, considers only the current scan performed by a MSMSRN. The set of sensors was thus constrained to include only radars, as various sensors have different sensing requirements and they would add extra complexity. Nevertheless, the approach was designed to be generic and adaptable to other sensor types in the future. The multisensor manager architecture of Blackman and Popoli [20] was employed, and only the high-level manager was the focus of the research.

Task generation was defined for the current research to be implicit, meaning that it performed for each of the fused tracks, which are the outputs of the track-fusion subsystem of the SFS. The SFS generates task for all targets that are recognised on the highest level of the system. Therefore, no task-generation algorithm was required in the multisensor managements used in this research.

Task prioritisation for the current research uses the information filter covariance matrix as the target priority mechanism. This approach was taken directly from the literature, as suggested by a reviewer. Although taking into account more information from all the levels of the DFIG Data Fusion model is desirable, this can lead to considerable complexity and will make results obtained hard to analyse. In some simulations performed, task prioritisation was simplified using a simple Markov chain. There is a fixed probability that tasks increase in priority, which is related to the maximum number of scans before a target would be lost. The priority of the target decreases when targets are measured in a multistatic manner.

This framework allows the focus of the research to be on MSMSRN scheduling. From the multisensor literature, it can be seen that sensor scheduling is essentially a temporal

task organisation problem. Multisensor scheduling must fill the timeline of each of the sensors with tasks that add value to the operation of the SFS.

2.5 Conclusion

As can be seen from Figure 2.2, the field of multisensor management has a diverse set of algorithms to solve the various aspects of sensor management. It is often hard to gauge which algorithms are better to use than others, as not only do the researchers solve very different problems, but often there has been very little work has been done to compare the algorithms in the existing literature. More work should be done to perform good comparisons of algorithms when they can solve the same problem. In addition, which problems the algorithms solve the best should also be explored more thoroughly.

Researchers often employ myopic simplifications for the sensor management model. Non-myopic approaches, which seem to be the future of multisensor management modelling, are more interesting and should be applied to further enhance algorithms after a myopic simplification has been used. In all cases, using the multisensor manager architecture of Blackman and Popoli [20] not only leads to a more well-defined approach, but also makes comparison to work conducted by other researchers easier. Future research should not necessarily only use non-myopic approaches, but algorithms must eventually be validated for these types of sensor management models.

While the literature contains very complex and interesting task-generation algorithms, this field appears to require a good background in human cognition and artificial intelligence. This aspect would also be very suitable as a separate doctoral level study. This is so that a computer can truly define/generate tasks with no human intervention. In such a system, there are no hard-coded rules and constraints, where the computer algorithm learns the capabilities of the IFS as a human operator would. Fully automated task generation will definitely add value in future IFSs and might lead to new applications that have not been conceived of at present.

Task prioritisation seems to have a very good set of algorithms that solve it adequately. These algorithms should, however, take into account more information from all the levels of the DFIG data fusion model. This can lead to considerable complexity, however, and will make results obtained harder to analyse. Similar to task generation, performing this task in a fully automated fashion is not an easy task but should, nonetheless, be the goal of the sensor management community.

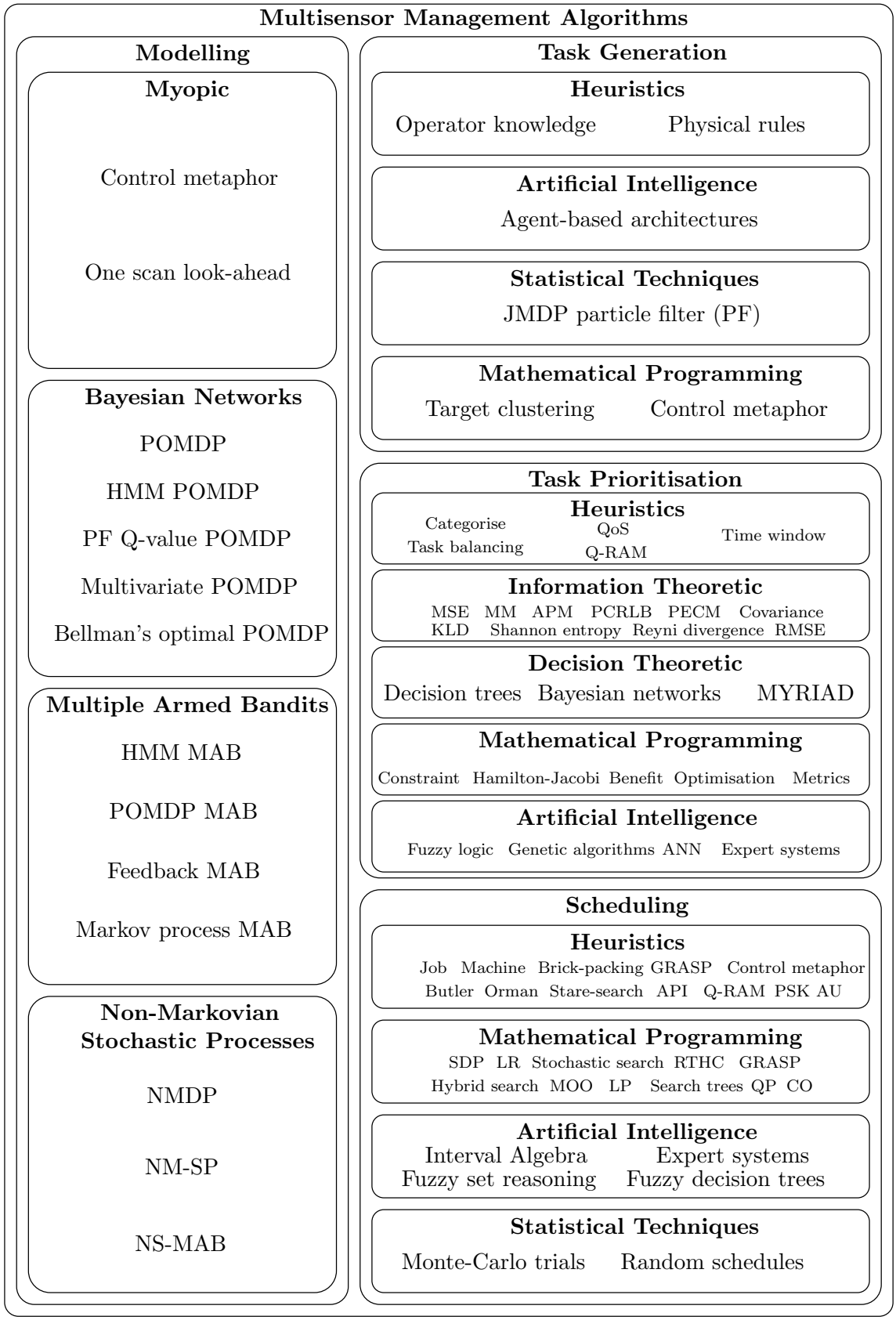


Figure 2.2: Map of multisensor management algorithms. All terms are defined in the glossary and were obtained from the referenced literature.

While the other aspects of sensor management are noteworthy, apart from scheduling, most require a good background in many fields of research. For this research, it was decided to focus on the scheduling aspect of mechanically steered multistatic radar management. Sensor scheduling appears to have many heuristic approaches, which, while computationally efficient, are tightly coupled to the problems they solve. Other scheduling approaches make use of optimisation and Monte-Carlo algorithms which can be very computationally costly. Therefore the approach followed in this work leads to a simpler and more intuitively scheduling algorithm through the use of Interval Algebra. Time will tell if this algorithm will be adopted more widely by the sensor management community. Certainly, a more automated scheduler should also be a goal to be considered by the sensor management community.

Chapter 3

Theory

3.1 Introduction

This chapter looks at the theory used throughout this research and is broken up into three sections. The first section is an overview of radar and information fusion preliminaries required to understand Mechanically Steered Multistatic Surveillance Radar Network (MSMSRN) scheduling. The second section covers the theory of Interval Algebra (IA), which is the algorithm I investigated while conducting my research. The third and final section covers the theory of the Greedy Randomised Adaptive Search Procedure (GRASP), a conventional scheduling approach that was compared to IA scheduling. Much of this chapter was the basis for the theory sections of the author et al.'s research paper accepted by the Elsevier Information Fusion journal (Section A.3).

Multistatic radars are a type of Sensor Fusion System (SFS) that employs geographically distributed radar receivers and transmitters. Section 3.2 starts off with preliminary radar background, covering only the detail necessary to understand the workings of monostatic and bistatic radars in the context of multistatic radars. It then progresses to multisensor information fusion theory, a part of which encompasses the need for multisensor management. The latter part is useful to understand how to control a multistatic radar network.

Section 3.2.2 begins with an introduction to the MSMSRN scheduling environment for a simplistic surveillance radar network. Next, the benefits of multistatic measurements are presented. As will be seen, these multistatic measurements are advantageous and are

enabled through the use of multiple radars. The focus area of the research documented in this thesis is then how to schedule a multistatic radar network, but not how to process the gathered information.

This chapter also highlights the practical concerns of a realistic MSMSRN that must be addressed by scheduling algorithms that can be used for a realisable MSMSRN. These include: uncertainties in the angular sequence of targets (Section 3.2.3); the effect of target motion and therefore the need for scheduling nimbleness (Section 3.2.4); and handling of monostatic scanning, tracking and confirmation tasks (Section 3.2.5).

Next, this chapter gives a detailed introduction to Interval Algebra in Section 3.3, starting off with the fundamental concepts. The usage of intervals, their relationships in IA and the creation of IA networks is discussed (Sections 3.3.1 and 3.3.2). This includes a description of constraint propagation, path and total/network consistency algorithms. Interesting aspects such as duration algebra, fuzzy IA and Bayesian IA are also introduced but not delved into in much detail (Sections 3.3.3 and 3.3.4). These topics could be used to conduct further research on the use of IA for other scheduling scenarios. The author of this thesis implemented MSMSRN scheduling algorithms utilising Interval Algebra to test the feasibility of IA as a generic scheduling algorithm. Section 3.3.5 gives an introduction of the current research that is being performed using IA algorithms on parallel processing architectures.

Finally, this chapter introduces GRASP in Section 3.4 and provides pseudo code for the MSMSRN scheduling, which is the sample problem to solve that was selected for this research. GRASP is compared to IA to ensure that IA can compete with conventional algorithms. The results of the comparisons performed are captured in Chapter 5.

3.2 Multistatic Radar Scheduling

This research was conducted for a wide-area surveillance SFS. The goal of the system is to detect, track and classify maritime targets. Radars are used as the primary sensors for all three of the SFS modes. In particular, the radars are mechanically steered pencil-beam monopulse Doppler radars. Multistatic measurements are possible because multiple distributed radars are used [44–46]. A sample problem needed to be devised to test the MSMSRN scheduling algorithms that were implemented. The research focused on the SFS scheduling task of generating multistatic measurements for the targets.

Only the required radar background for multistatic measurements is presented. This is followed by the Data Fusion Information Group (DFIG)[1] Data Fusion Model (DFM), which includes multisensor management. This is followed by a definition for the need to make multistatic measurements. Then the sample problem used to test the algorithms is described. Finally, a description of the simulation environment is given. In particular, the way simulations were conducted is shown.

3.2.1 Radar preliminaries

The radar information given here is an overview of the radar book by Richards et al. [147]. There are numerous books and papers available dealing with the details of radar sensors and an exhaustive list will not be given here.

The word radar was originally an acronym and stands for radio detection and ranging. In a monostatic radar, the receiver and transmitter are co-located (Figure 3.1), whereas for bistatic radar the transmitter and receiver are located some distance apart (Figure 3.2). For bistatic radar the angle between the transmitter and receiver dish is known as the bistatic angle.

Monostatic radars transmit electromagnetic energy on the target and use the delay between the transmission and reception of the pulse to measure range. The target must lie within the beam of the radar to make monostatic measurements. Bistatic radars use an ellipsoid with the transmitter and receiver as the foci of the ellipsoid, the actual location

[1]The DFIG is part of the Joint Directors of Laboratories (JDL), which in turn is part of the United States Department of Defence (US-DOD).

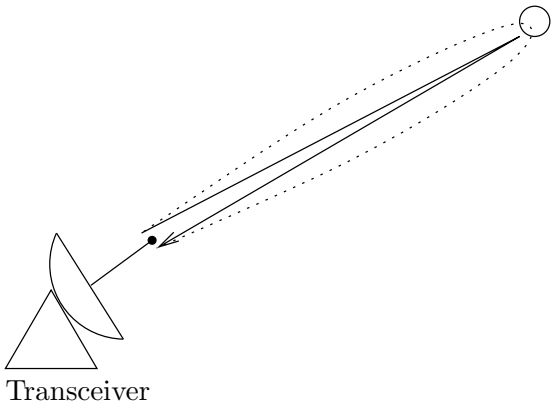


Figure 3.1: Monostatic measurement of a target by a single radar. The transmitted radar signals reflect off the sphere and are received at the same radar. The dotted line represents the round-trip delay of the electromagnetic signal, which is used to determine the slant range of the target. Half of the round trip time determines the slant range, which is the range of the target within the beam and differs from the true ground range due to the elevation angle of the beam. The angle of arrival is determined by a measurement of the position of the positioner, the beam width. Monopulse processing can improve the angle of arrival measurements to half the beam width.

of the target is then determined by delay as well as the angle of arrival (azimuth angle) measured by the receiver to pinpoint the target. Multistatic radars make use of both monostatic and bistatic techniques. Monostatic measurement is depicted in Figure 3.1 while bistatic measurement is depicted in Figure 3.2.

Monopulse radars send out or sample the received beam as four discrete parts [65] (Figure 3.3). Two beams are adjacent to the bore-sight of the radar in elevation angle, and the other two beams are adjacent to these in the azimuth angle. This allows the radar to accurately pinpoint the target within the beam. The computation requires taking the difference between the azimuth pairs and elevation pairs. The probability of detection is maintained by taking the sum of all beams to increase the Signal-to-Noise Ratio (SNR) ratio. These computations are generally performed using analog Radio-Frequency (RF) comparator circuits. Thus, the radar makes use of three channels of data: the summation/sum, azimuth and elevation channel. Signal processing must be applied on all three channels but detections are only formed for the sum channel. When targets are exactly in the radar bore-sight, only the sum channel will contain target energy. Offsets in azimuth and elevation will cause energy to appear in the relevant channels as well. Monopulse calculations take the ratio of the channels into account to estimate the azimuth and elevation offset that the target has from the bore-sight of the radar beam.

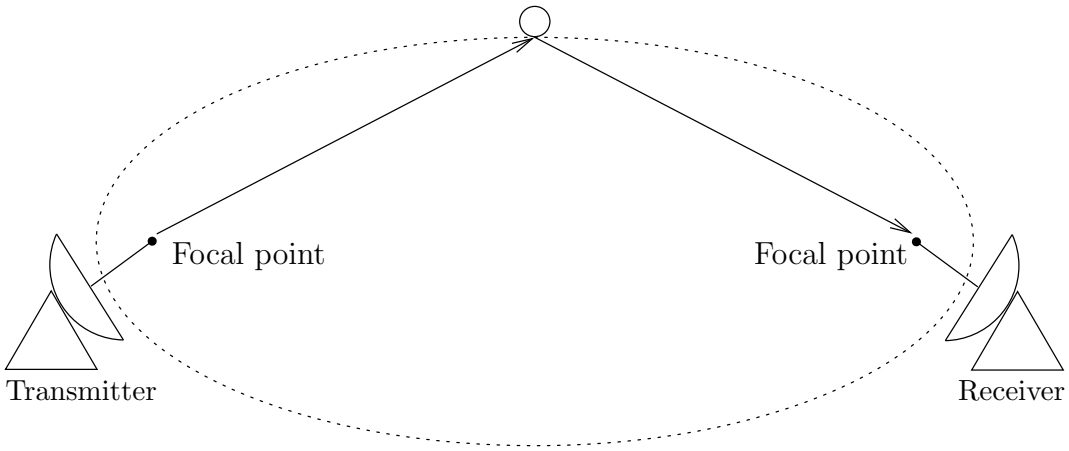


Figure 3.2: Bistatic measurement of a target by two radars. Signals transmitted by one radar reflect off the sphere and are received at the second radar. The ellipse in the figure depicts the cross-section of the bistatic range measurement ellipsoid; all points on the ellipsoid satisfy the round trip delay time from the transmitter to the receiver. The focal point of the dishes of the two radars will be the foci of the ellipse for the bistatic radar. Thus, to have a pinpointed position the transmission delay as well as the cone-angle (azimuth and elevation angles) of arrival is required.

Figure 3.4 shows the typical building blocks of a tracking and search radar. For search radars the positioner control is not controlled by the output of the tracking component. Bistatic radars split the receive and transmit paths geographically and can sometimes do without transmissions by utilising other radio-frequency signals. The latter type of bistatic radars are usually referred to as Passive Coherent Location (PCL) radars. Transmitted signals are generated digitally and then converted to analog form. These analog signals are mixed up to the frequency band of the radar and amplified before transmission. The circulator is used in monostatic radars to protect the receiver during transmission time when one antenna is shared for receive and transmit or two antennas are too close. Gain control attenuates nearby reflected signals so as not to saturate the radio-frequency components. A low-noise amplifier boosts signals sent down from the receiver to ensure a better signal-to-noise ratio. The signal is mixed down to an intermediate frequency from the radar operating frequency in the RF receiver. Digital sampling is performed and finally radar signal processing techniques are applied digitally in the radar signal processor.

Measurements from radars are passed into a Radar Signal Processor (RSP) (Figure 3.4). The data are usually processed as discrete packets of information known as bursts. The bursts consist of range and Doppler information, and each sample in time represents

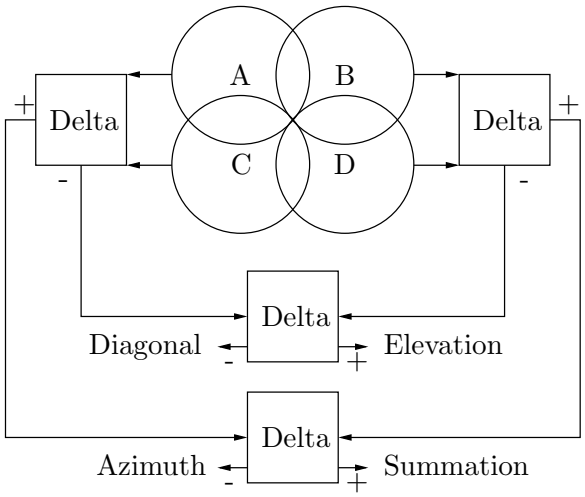


Figure 3.3: Monopulse radar measurements using four distinct transmit and receive beams/antennas using analog radio-frequency comparator circuits. Transmit beams are formed such that they can be discerned from each other on receive. The receive beams formed are passed through four comparators to generate the elevation, azimuth and summation signals. Diagonal signals, while physically created, are not used. Elevation signals are formed from the difference between the sum of the received echo on the upper two beams and the sum of the received echo on the lower two beams. Azimuth signals are formed from the difference between the sum of the received echo on the right two beams and the sum of the received echo on the left two beams. Summation signals are formed from the sum of the received echo on all the beams.

a range sample. An arbitrary range is decided as the maximum and is related to the pulse repetition interval. This selection, however, cannot be made independently of the required Doppler resolution. The details will be discussed along with Doppler processing later. Subsequent range samples therefore form part of the next pulse. Consequently, the entire burst is therefore a range-Doppler map handled as a matrix of samples. The RSP applies the following signal-processing functions: intermediate-frequency/digital-down sampling, equalisation, pulse compression, matrix transpose/corner-turning memory, Doppler filtering and detection using a constant false alarm rate (CFAR) detector. The outputs of the detector are known as first detections. These detections are then passed into a detection and estimation processor to form second detections, using clustering and censoring, and also better estimates of target range, angle and radial velocity.

Extended targets, spanning multiple range and Doppler cells, are larger than the range resolution of the radar (Figure 3.5). First detections of such targets must be clustered into a single second detection. Small targets may have only single first detection associated to the second detection. Estimations of target kinematic attributes, usually in spherical

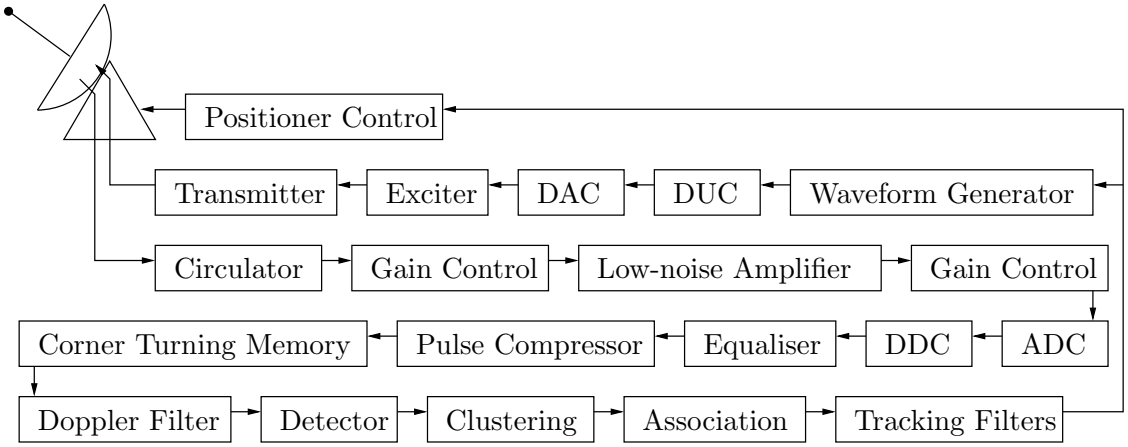


Figure 3.4: Processing chain of a typical radar sensor. Functions performed before the Digital-to-Analog Converter (DAC) and after the Analog-to-Digital Converter (ADC) form part of the Radar Signal Processor (RSP). Signals to and from the RSP are mixed up to a suitable intermediate frequency (usually lower than 250 MHz for non-imaging radars) using a Digital Up-Converter (DUC) and Digital Up-Converter (DUC). All other functions apart from the positioner control form part of the Radio-Frequency (RF) components to mix up and down from an intermediate frequency to the operating band (e.g. 8.5 GHz to 10 GHz for X-band). Not depicted is the radar timing control and local oscillator as these components attach to most RF components.

coordinates, are then made using statistical techniques. Afterwards, both the detections and estimations are fed into a tracking filter to form a track on the target. The tracking filter sequentially incorporates past and current information to track the spatial and kinematic properties of a target (Figure 3.7). A popular filter is the Kalman filter, which has also been used for fusion of multiple tracks through track fusion [12].

Doppler radars measure the Doppler frequency caused by the motion of the target (Figure 3.6). Successive phase-coherent pulses are transmitted over a short dwell time by pulse Doppler radars. Continuous wave radars transmit continuously but still sample the receive data into several PRIs. The received data broken up into PRIs are sampled coherently and adjacent range cells (or bins) of the range-Doppler map are passed through a Fourier transform.

The dwell time required is related to the velocity of the target. The number of pulses sent determines the Doppler sampling period. Longer periods can allow unambiguous velocity measurements of greater target speeds, but also reduce the range sampling capability of the radar. This is due to the fact that the dwell time requirement must still be met. If the dwell time is too long, the signals will decorrelate with each other and no

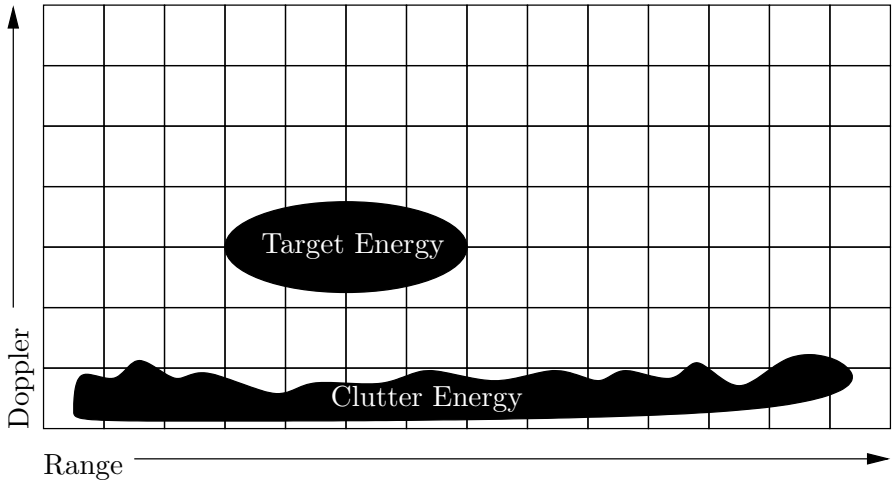


Figure 3.5: Radar burst measuring an extended target and clutter. The figure shows how radar data is sampled for an extended target as a range-Doppler burst of data. Each echo for each transmitted pulse is sampled sequentially generating range samples. The sampling speed determines the range resolution of the radar. Successive pulses are treated as distinct Doppler cells. This leads to a range-Doppler matrix per radar burst of data. Targets larger than the radar range resolution will be present in multiple range cells. Parts of the target that move at different velocities will extend into different Doppler cells.

Doppler measurement will be formed. Short dwell times can result in insufficient target energy being returned and no radar measurement being made.

These opposing requirements will result in ambiguities in either of the domains. A longer range will reduce the Doppler resolution and result in a slower upper bound on the measured unambiguous target radial velocity component (towards the radar). Having a higher upper bound for the unambiguous target radial velocity requires having a shorter unambiguous range resolution. This is due to the fact that the dwell time must be short enough that the target signals do not become uncorrelated to prior returns.

Owing to the cyclical nature of the Fourier transform and finite number of pulses, the velocity is ambiguous and modulus mathematics is applicable. Knowledge of the transmitted frequency and number of pulses helps to calculate the radial velocity. It is important to note that the radar can only measure the radial component of the velocity of the target relative to the radar. The 0 Hz Doppler line can usually be ignored by the detection logic of a stationary radar, as it will contain large returns from clutter (background interference from trees, buildings, etc.). This is provided that the target is not moving tangential to a hypothetical radial line connecting it to the radar.

However, a Doppler frequency is induced on signals returned by stationary clutter for

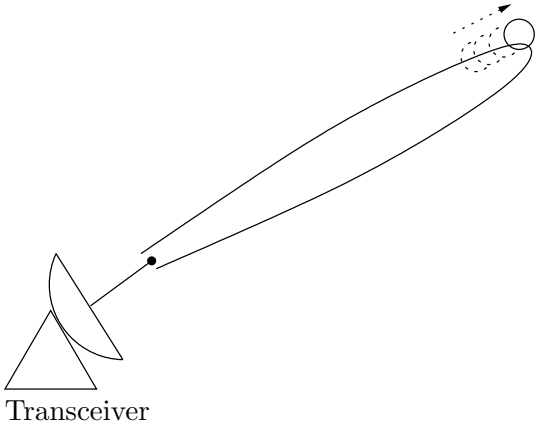


Figure 3.6: A radar Doppler measurement generated using successive coherent pulses over a short dwell time. The radar beam is depicted as a black lobe that intersect a target. The spheres with dotted lines depict prior locations of the target. Here the movement of the target is exaggerated. The actual motion of the target during the measurements is small; however, even small movements induce a measurable Doppler frequency on returned signals.

moving platforms. This means that the clutter will appear in a variety of Doppler lines. Thus, the Doppler line that contains stationary clutter varies based on the speed of the platform. This can usually only be fully resolved using Space-Time Adaptive Processing (STAP).

If the speed of the target is greater than the Doppler sampling period can measure, the target can wrap back into the ignored Doppler bins/lines. This is known as a Doppler blind zone and results in the target not being measured if these Doppler lines are ignored.

Figure 3.7 shows an example track for a moving object. Each tracking sample is denoted by a full circle. The ellipse surrounding the circle depicts the tracking covariance. The covariance gets smaller when the target performs slower manoeuvres. Noise present in the received signals leads to an offset between the measured location of the target and the true location.

3.2.2 Multistatic radars

Now recall from Section 1.3 and Chapter 2 the details of multistatic radars and multi-sensor management. Multistatic radars are a type of Multiple-In Multiple-Out (MIMO) radar system, which provide many advantages over monostatic and bistatic radar systems. Multistatic measurements can help reduce occlusions that would occur due to for example

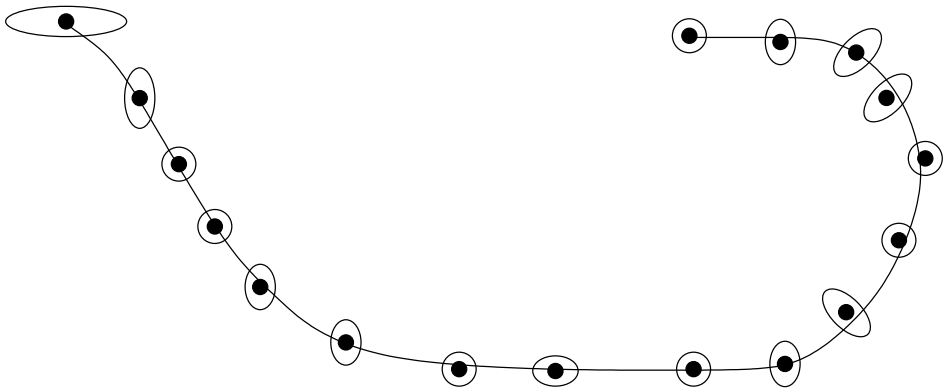


Figure 3.7: Example of a radar track for a moving target. Radar measurements are depicted as black dots along the solid black line depicting the motion of the target. Each measurement is not exact and the measurement 2σ covariance is plotted as black ellipses around the radar measurement.

the sea state for maritime targets. Waves and other sea conditions can cause targets to be occluded or experience electromagnetic multi-path effects. Multistatic measurements are formed by using both monostatic and bistatic radar techniques. First, the transmitting radar can form monostatic measurements from the direct skin echo of the electromagnetic signals from the signals it has sent. Next, from indirect reflections of signals transmitted by other radars bistatic measurements can be formed. Measurement fusion is then performed on the monostatic and bistatic measurement to form a multistatic measurement. As the measurements are made during the same time at the same radar, computation and communication requirements are minimal.

Multistatic measurements have several advantages over making only either bistatic or monostatic measurements. Firstly, the SNR is raised as there is more energy on the target. This means that the probability of detection (P_d) will be greater, as P_d is directly proportional to the SNR [147]. Secondly, the probability of detection also goes up as clutter effects and target Radar Cross-Section (RCS) vary for the two radars. To be precise, the clutter interfering with the target signals received at the two radars is statistically independent. Furthermore, as most targets are not symmetrical in all spatial dimensions, the RCS is related to the aspect angle of the target in relation to the radar. In fact, even for stealth aircraft, the scattering of signals is more likely to cause detections in an MIMO radar.

In terms of information fusion, multistatic measurements also provide numerous ad-

vantages. Firstly, there are simply more radar measurements made of the target, which provides better information about the spatial and kinematic information gathered on the target. This can especially help mitigate the multipath distortions in some geometries of radars to targets. Secondly, given certain geometries, the intersection of the track covariances will produce more accurate fused information. The best scenario in a binary radar system is where the beams of the radars intersect at 90°when making measurements. In this case, the angle inaccuracy of the two radars are orthogonal to each other. Since this is the largest uncertainty for both monostatic and bistatic measurements, the resulting inaccuracy of the target position will be the lowest possible. For multistatic radar systems where more than two radars are used the best scenario will change based on the location of the radars. Lastly, the RCS and clutter diversity in the fused information ensure that the track will be more accurate.

The radars must transmit at different frequencies and be able to receive on all radar transmit frequencies to make a multistatic measurement, which consists of a monostatic and bistatic measurement at each radar. Another possibility is to make use of the constructive interference of the waveforms sent by both radars. In this case, the radars can still receive on a single frequency but the SNR is likely to be higher. Furthermore, target occlusion at all radars in both monostatic and bistatic modes is also less likely. For both types of multistatic radar systems, the radars must measure the targets simultaneously. Therefore, the multisensor manager must apply closed loop control of the SFS. Every time a multistatic measurement is required, the corresponding radars must be scheduled to measure the target simultaneously.

Management

The goal of the MSMSRN management algorithm is to ensure that the radars measure the targets periodically in a multistatic mode (refer to Figure 3.8 for an example). While simultaneous scheduling of search modes can also be advantageous, this was not investigated in the current study. Multistatic measurements should ensure that tracks of the targets will be more stable. Searching for new targets within the field must also still be performed, seeing as the primary task of the SFS is surveillance. This leads to the constraint that the radars are disallowed from turning around during scans. Therefore, the

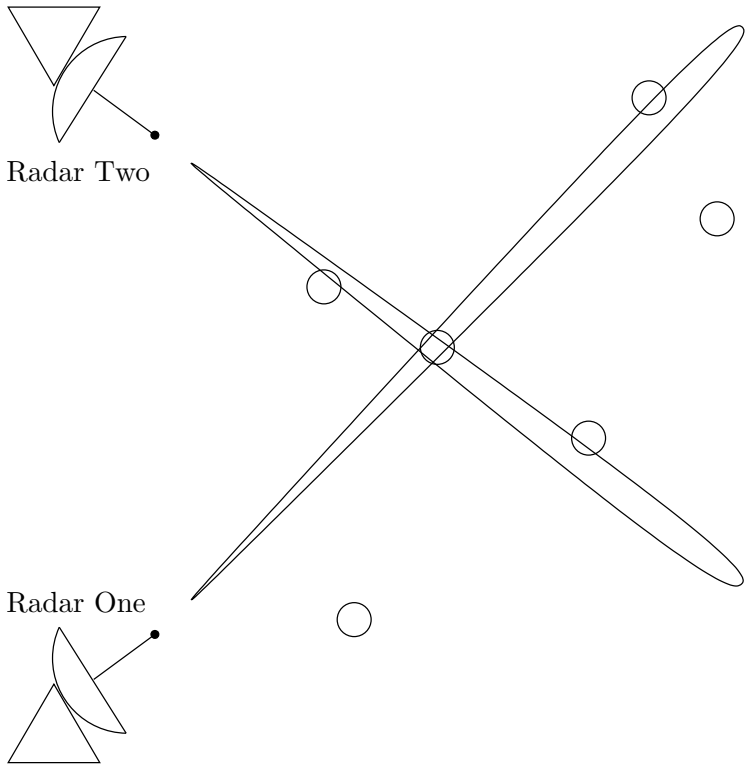


Figure 3.8: Example scene for MSMSRN management of a binary radar system. Radar beams are depicted as black lobes extending from the radar dishes. Targets are circles and a multistatic measurement is being made of a single target. Three other targets are being measured using monostatic measurements.

MSMSRN management algorithm will, on each scan, have a fixed maximum number of targets that can be measured in multistatic mode. In general, this number will be lower than the total number of targets. An example scene is depicted in Figure 3.8.

The MSMSRN management algorithms implemented are sub-divided into a high-level and low-level scheduler. A multistatic scheduling task is created per target. Task prioritisation is then achieved through the use of the tracking covariance of the targets or a simple priority mechanism. The low-level scheduler adjusts the azimuth scan rate of the mechanically steered positioners of the radar to ensure that the radars measure targets simultaneously. Typically, one radar must slow down while another speeds up its azimuth scan rate. The high-level scheduler decides which multistatic scheduling tasks will be performed from scan to scan. These decisions are made based on the target priorities constrained by positioner movement and target locations.

A sensor coordination algorithm is considered a *nimble scheduler* when it is able to rapidly adapt to changes in the surveyed area. This means that, if there are fast-moving

or rapidly accelerating targets, the scheduler will incorporate the updated target locations in real-time. These targets will not only affect planning but also scheduling within very short time intervals. Thus the scheduler must be able to recalculate the schedule of tasks for the sensors concurrently with the execution of these tasks by the sensors. The updated schedule of tasks can leverage the improved situational picture as it is generated by the information fusion system.

3.2.3 Scheduling problem

The sensor management problem to solve is selecting the maximum number of targets to measure with multiple radars. It is assumed that the mechanically steered positioners of the radars are not allowed to change direction during the scan, as doing so would increase the cost and complexity of the radars. Thus each radar will temporally scan over all targets in a specific sequence.

Given a multistatic radar network with R radars that are scanning a scene of N targets, the scheduling goal per scan can be written as:

$$\begin{gathered} \operatorname{argmax} \sum_{k=1}^{N} Q\left(k\right) \cdot S(k) \\ \text{subject to: } C(m,n,r) \, \forall \, \{m,n\} \in K, r \in R \end{gathered} \tag{3.1}$$

where k, m and n are target indexes; r is the radar index, K is the set of scheduled targets and R is the set of radars. $Q(k)$ is the priority of target k. $S(k)$ denotes whether target k has been scheduled. The full enumeration of $C(m,n,r)$ for all values of m and n in K, as well as radars r in R are the constraints that need to be satisfied. In other words, the goal is to find the largest set K, which maximises the value of Equation 3.1 but does not violate any constraints. The optimisation variable in Equation 3.1 is $S(k)$ when considering a single scan. However, good selections of $S(k)$ can also affect the quality of the measurements and, thus, the priority of the targets $Q(k)$.

The target selection function $S(k)$ is then simply:

$$S(k) = \begin{cases} 1 & \text{if } k \in K, \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

Given an arbitrary reference line and radar positioner rotation, $C(m, n, r)$ returns constraints as follows:

$$C(m,n,r) = \begin{cases} \theta_r(m) \prec \theta_r(n) & \text{if } m < n, \\ \theta_r(m) \succ \theta_r(n) & \text{if } m > n, \\ \emptyset & \text{otherwise} \end{cases} \tag{3.3}$$

Here $\theta_r(m)$ is the function that determines the angle from the radar r taken from the arbitrary reference line to target m. This reference line is selected such that the angles for all targets increase monotonically. For a dual radar case used in this research, the best selection for the reference is the line that connects the radars. The $C(m, n, r)$ function imposes the constraint that all targets must either appear in sequence or simultaneously for a specific radar. The precede '$\prec$' and succeed '$\succ$' operators are used as the ordering depends on the scan direction. Thus, all the radars should be able to measure the targets in sequence without requiring a change in scan pattern.

Equation 3.3 will always hold given a specific reference line. The angles for a target to a radar is determined by θ_r, which is not only dependent on the target but also the radar. The function θ_r will certainly change if the reference line is changed. However, the choice of reference line is arbitrary as all that is required is for the targets to either increase monotonically or decrease monotonically for the next scan. A different reference line can be selected per radar but this will likely make the application of the technique slightly more complex

Uncertainty in the angular sequence of targets

Uncertainties arise in the azimuth ordering of the targets for either radar. This is due to the azimuth angle measurement accuracy of the radar, which cannot be infinitesimally small. This is an unavoidable situation even if monopulse measurements are made, as in both cases the angular accuracy is determined by the beam width [65]. Consequently, when targets are too close together in azimuth angle the radar is unable to discern their true order.

The MSMSRN scheduling algorithms must be able to exploit these uncertainties so that they can be used to schedule more multistatic measurements per scan. Furthermore,

no tracking degradation will be experienced in the ambiguous case. This is due to the fact that the targets are sufficiently close to the centre of the radar beam in order to receive maximum antenna gain.

Figure 3.9 illustrates a scenario where targets are measured ambiguously. Radar one does not need to discern the true order of targets three and five.

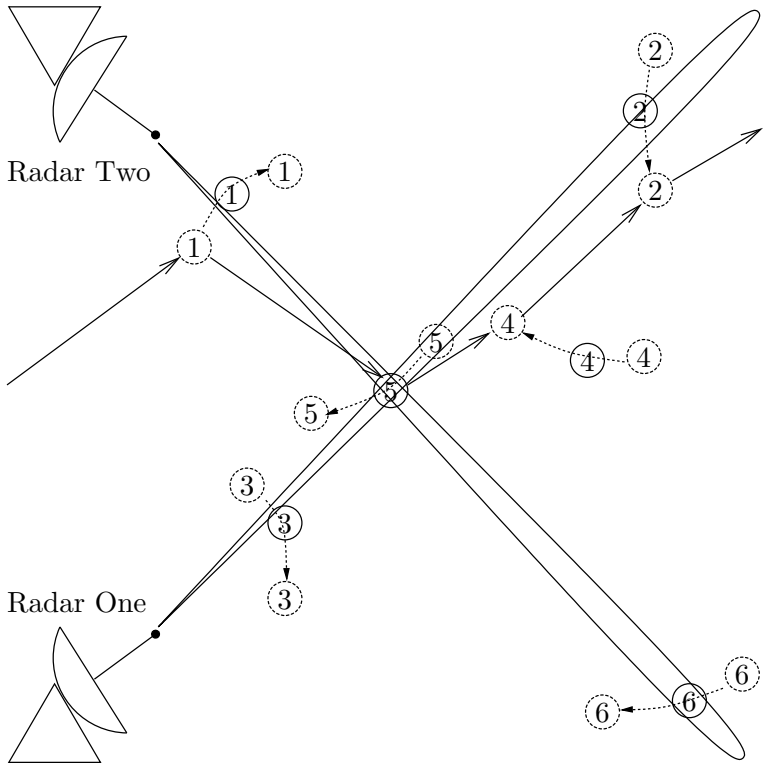


Figure 3.9: Uncertainties in the sequence of targets and target motion as scheduling processes.

3.2.4 Target motion and scheduler nimbleness

Since the targets are moving, the geometry of the targets relative to the two radars changes over time, as depicted in Figure 3.9 by the curved dotted arrows. If the targets move fast enough and scheduling is performed without any future prediction, then it is possible that the target will not lie in the beam of one or both radars and a multistatic measurement will not be made. Similarly, targets that are ambiguous for a radar will only remain in this state for a brief period of time.

As a result, the scheduling problem to solve is dependent on the targets that are scheduled. Each time a target is scheduled, the intersection point of the radar beams

must move from the current target to the next target. To do this, the scheduler merely requires knowing the target angles from both radars and moving the positioners at the correct rate to intersect at the new point. The strategy followed is to move the radar that must rotate by the biggest angle at the maximum rotation rate and slow down the other radar appropriately. The shortest path is a straight line between the two targets and is shown in Figure 3.9 by the solid straight arrows. Thus the minimum time that will pass is the angular difference between the two targets divided by the maximum scan rate. This value will be largest for the radar which has the greatest angular difference to rotate.

During this time, all targets that must still be measured by the radars later in the scan will have moved. The distance required to move out of the beam of the radar is dependent on the target distance from the radar, the beam width of the radar and the direction of motion. Assuming straight line motion, the last parameter can be simplified to an aspect angle of the motion. The minimum velocity can be calculated from the distance and the scheduling time between the two multistatic measurements, which will cause multistatic measurements to be missed.

The degree to which multistatic measurements will be missed is thus dependent on the velocity of the target as well as the scanning rate of the radars. Targets that will be measured closer to the end of the scan will also be the worst affected, as more time would have passed before these multistatic measurements were attempted. Fast vehicles, helicopters, airplanes and fast-moving boats, such as go-fast boats, will cause the largest problems for any surveillance radar scheduler. Ballistic and supersonic targets are usually not tracked by surveillance systems and are instead tracked by either a tracking radar or more frequently by the tracking aspect of a Multifunction Radar (MFR) due to their extremely fast motion. For all these types of target, the scene unfolds quickly at each time step and it is not possible to rely on targets falling within the edge of the beam to ensure detections.

In order to handle fast targets, a scheduling algorithm must therefore predict target locations after each multistatic measurement is made. However, as the goal is to measure as many targets as possible, this leads to two interacting requirements. Firstly, the schedule of targets to visit that will be generated is dependent on the geometry of the targets. However, the geometry changes over time and the degree of change depends on which

targets have been scheduled. There is no simple solution to this problem that will ensure that no multistatic measurements are missed.

Nevertheless, any forward predictions made should result in a better scheduling performance than making no predictions. One simple approach would be to determine the fastest time the two radar beams could be made to intersect at any target in isolation. Thus all other multistatic measurements that may be made before this target is visited are ignored. This will once again lead to a static target geometry, with errors that increase over the scan, which can be used to select the targets for multistatic measurements. A better approach would be to calculate the schedule of targets to visit on the fly, while still attempting to measure as many high-priority targets as possible. This scheduler would be nimble in terms of reacting to the changes in the target geometry.

Both approaches require solving

$$\theta_r(t_0) + t \cdot \dot{\theta}_r(t) = \arctan\left(\frac{\left(\mathbf{A}_f^t \cdot \hat{\mathbf{x}}_f(t_0) - \mathbf{x}_r\right) \cdot [0\ 1\ 0\ 0]'}{\left(\mathbf{A}_f^t \cdot \hat{\mathbf{x}}_f(t_0) - \mathbf{x}_r\right) \cdot [1\ 0\ 0\ 0]'}\right) \tag{3.4}$$

for each radar r, where t_0 denotes the current time, t denotes the prediction time, f signifies the target, θ and $\dot{\theta}$ is the current positioner movement profile of the radar, $\mathbf{A}$ is the motion model of the target, $\hat{\mathbf{x}}$ is the current filter estimate of the target Cartesian location and velocity, and $\mathbf{x}_r$ is the Cartesian location and velocity of the radar (assumed to be stationary). This equation can be solved for t using the Newton-Raphson method along with a numerical differentiation technique.

Equation 3.4 is solved by the scheduler for t to make a forward prediction of the location of each target based on the schedule it builds. The forward prediction of targets can alter the geometry. At each time step the scheduler forward predicts all targets from its current location to the intercept time for all targets. The scheduler selects one positioner to move at the maximum rate thus the theta rate is known. The intercept time is not known up front, as the target moves while both radars are rotating. Thereafter, all other targets need to be forward predicted as well in order for the scheduler to decide if this is a good choice of target sequences to use.

Similarly, the tracking covariance matrices of the targets must also be updated:

$$\bar{\mathbf{P}}_f(t) = \mathbf{A}_f^t \cdot \hat{\mathbf{P}}_f \cdot \mathbf{A}_f^{t\,\prime} + t \cdot \mathbf{Q}_f \tag{3.5}$$

where $\mathbf{A}$ is the state transition matrix of the target, $\hat{\mathbf{P}}$ is the estimate of the tracking covariance, $\mathbf{Q}$ is the process noise covariance matrix (caused by the target manoeuvring) and $\bar{\mathbf{P}}$ is the prediction of the tracking covariance. However, as the implemented simulation used discrete time, it was possible to search for the solution.

3.2.5 Scanning, tracking and confirmation tasks

The multisensor management solution investigated here then uses a hybrid architecture consisting of two levels [20]. The higher-level radar coordinator attempts to schedule multistatic measurements for as many targets as possible. The lower-level radar manager is replicated for each radar and operates independently of the radar coordinator. The radar manager could also have been implemented using IA; however, it is not the focus of this research. In this research, the radar manager makes a decision to track, confirm or search independently.

Thus far, only the multistatic aspect of the radar system has been discussed. However, nothing stops the radars from scanning for new targets while they are not making multistatic measurements. Furthermore, targets that will not be measured in the multistatic mode should still be measured in a monostatic mode by the radar so as to maintain a stable track. Finally, it is also possible to confirm the presence of new targets during the same time.

Performing these types of tasks will not affect the multistatic measurement capability of the radars for slow targets, as they only require one radar to make the measurement. Also, they always temporally occur between two multistatic measurements for the radar performing these tasks. During the time that these tasks are performed the other radar may only be required to slow down. If either radar dwells longer, the other must be slowed down such that the two radars will still intercept at the next rendezvous point. Effectively the dwell time is reducing the average rotation rate of the radar to below what it would have achieved if it was not required to dwell.

These tasks will, however, impact on the maximum scan time, and thus if there is an upper bound on the scan time or a requirement for a fixed scan time, only a certain number of these tasks will be possible. This can easily be accommodated in any scheduling algorithm by keeping a sum of the duration of all tasks. As long as the sum of task durations is less than the maximum scan time, more of these tasks can be added.

3.2.6 Target dynamics

A MSMSRN scheduling algorithm must take into account both the scan time and spatial constraints. The scan time is limited by the speed of the radar positioners and also how quickly the radars need to react to target movement. Under an upper limit on the scan time and quick target revisit times, there is a reduced number of multistatic measurements that can be made.

For the comparison simulations the target and radar dynamics were not as important. Both scheduling algorithms as implemented only account for the target geometry. A soft limit was set on the scan time by not allowing the positioners to turn around and requiring that the positioners move as fast as possible between multistatic measurements. Consequently, the comparison simulations of IA and GRASP only looked at the spatial constraints. Given the same operating environment, the comparison is fair. Thus, the scheduling algorithms need to be tested only for the same target scenarios. Testing diverse target geometries using many short simulation runs would ensure no biases.

Incorporating revisit times and radar rotator speeds can be achieved by merely setting an upper limit on the total scan duration. Each task that gets scheduled will have a required task time. The upper limit will impose a maximum number of tasks that can be performed during a scan. The highest priority tasks would still need to be scheduled and lower priority tasks would be sacrificed. However, to not severely impact system performance the tasks prioritisation mechanism would need to be changed. The upgraded tasks prioritisation mechanism will ensure that tasks never get starved of radar performance, and that high priority tracks are maintained. Thus, simulations testing the requirements of the schedulers to respond quickly were also incorporated. Here targets could manoeuvre quickly and would impact the scheduling.

Nevertheless, target dynamics limit the maximum scan time for the radars when con-

sidering both monostatic and multistatic measurements. For monostatic measurements, the only consideration is the accuracy of the prediction when the targets manoeuvre. The Kalman filter used assumes a certain amount of acceleration from the target, which opens the tracking gate. However, forward predictions will only take the prior motion into account and, thus, the track will follow the similar trajectory as the last measurement. If the measurements are not updated in a short enough interval, new measurements will fall outside of the gate and consequently the track will be lost. Furthermore, when making multistatic measurements, the beams of the radars intersect to form a kite-like quadrilateral and only in a single configuration a triangle (when the beams are pointing such that their edges just point at the other radar). The size of the quadrilateral depends on the azimuth angles of the two radar beams. Thus in certain geometries of the target and beam intersects, small inaccuracies in the prediction of the future target locations can cause multistatic measurements to be missed (one or more beam does not intersect while illuminating the target).

Simulation assumptions

In the comparison simulations, the targets are assumed to be boats. This selection was made as multistatic radars are already being employed in maritime environments through research conducted by both the CSIR and UCT. The boats are assumed to start at random points and end at random points. The boats steer themselves to the end point by changing their heading only. Three forces act on the boats: the first force is the boats own power, the second force is from a constant wind and the third is from the drag they experience from the sea. The boats can apply 1.42 ms^{-2} maximum accelerations. The wind is assumed to blow constantly from a south-western direction at 5 ms^{-2} but only affect the boats by 0.5 ms^{-2} due to the size of the ships. Finally, the drag force retards the boats motion to ensure that the boats after being manoeuvred travel at constant velocity of 3 ms^{-1}. For these simulations the boats were assumed to travel in a straight line. Given these constraints, the target should never be missed and there is no need to set an upper scan time, since the tracking filters will be able to accommodate for all the constant velocities. This was crucial for comparing the algorithms fairly. If the scan time was a factor in the comparisons, it could unfairly affect one of the schedulers and bias the comparison results.

For the nimble simulations a more realistic target motion model was assumed. The targets were assumed to be able to accelerate and decelerate at the acceleration rates specified using the Singer random walk model. In each simulation batch, the acceleration variance σ_a of the Singer random-walk motion model and initial target velocity, a Gaussian random value with mean μ_v, were set differently. In batch one $\sigma_a = 2$ ms^{-2} and $\mu_v = 25$ ms^{-1}, in batch two $\sigma_a = 2$ ms^{-2} and $\mu_v = 50$ ms^{-1}, and in batch three $\sigma_a = 20$ ms^{-2} and $\mu_v = 25$ ms^{-1}. Targets were however only modelled to move in a curved line, which means that only one manoeuvre was allowed. However, while moving on these trajectories, the target could speed up and slow down at random. For these simulations the scan time affects the number of missed multistatic measurements and loss of track events.

For loss of track events there are two possibilities. The first is if the radar beam no longer illuminates the target due to sufficient angular error in the predicted target state. The second is if the radar beam still illuminates the target but the measurement no longer falls within the tracking gate. This can be caused by both angular and range errors. For missed multistatic measurements, the same limits apply but now both radars must have sufficiently reduced angular and range errors.

To determine the bounds of the upper bound for scan time the time delay that causes the error in predicted target state to be greater than the Kalman gate requires to be calculated:

$$i = y - \boldsymbol{H}\boldsymbol{A}^t\hat{\boldsymbol{x}} \tag{3.6}$$

$$\boldsymbol{S} = \boldsymbol{H}\boldsymbol{P}\boldsymbol{H}' + \boldsymbol{R} \tag{3.7}$$

$$\mathrm{E}\left[i'\boldsymbol{S}^{-1}i\right] \geq 9 \tag{3.8}$$

where t is the predicted time y is the observation, $\boldsymbol{H}$ is the observation matrix, $\boldsymbol{A}$ is the target state transition matrix, $\hat{\boldsymbol{x}}$ is the estimated target state and $\boldsymbol{P}$ is the zero-mean Gaussian estimation error and $\boldsymbol{R}$ is the measurement covariance. Using numerical methods the greatest value for t can be calculated using Equations 3.6 and 3.7 under the target motion model assumed, the initial target velocity and target acceleration variance that still satisfies Equation 3.8.

For missed multistatic measurements the angular error in the fused track must be

less than half the beam width for both radars. This can be solved using the following equations.

$$\theta_x = \arctan\left(\frac{\boldsymbol{x}[1] - \boldsymbol{p}[1]}{\boldsymbol{x}[0] - \boldsymbol{p}[0]}\right) \tag{3.9}$$

$$\theta_f = \arctan\left(\frac{\boldsymbol{A}^t\hat{\boldsymbol{x}}[1] - \boldsymbol{p}[1]}{\boldsymbol{A}^t\hat{\boldsymbol{x}}[0] - \boldsymbol{p}[0]}\right) \tag{3.10}$$

$$\mathrm{E}\left[|\theta_f - \theta_x|\right] \geq \frac{\phi}{2} \tag{3.11}$$

In Equations 3.9–3.11, t is the predicted time, $\boldsymbol{p}$ is the radar position in Cartesian coordinates, $\boldsymbol{x}$ is the target state, $\hat{\boldsymbol{x}}$ is the fused target state estimate, $\boldsymbol{A}$ is the target state transition matrix, θ_x is the angle computed for target position $\boldsymbol{x}$ to the radar, θ_f is the angle computer for target predicted position $\hat{\boldsymbol{x}}$ to the radar and ϕ is the radar beam width. Again numerical methods are required to find the maximum value for t that satisfies Equation 3.11.

Thus, the scan limits for missing multistatic measurements of targets determined through three Monte-Carlo simulations batches are as follows[2]. The tracks must be updated in each iteration or at the rate of $\frac{1}{90}$ seconds for targets with profiles that are tangential to the beam progression or are manoeuvring tightly regardless of the Singer model parameters. On average, an upper scan time limit of 12.2 seconds is sufficient for batch one. In simulation batch the upper scan time limit reduces to 6.63 seconds. Finally, simulation batch three requires an upper scan times of 12.1 seconds. In certain cases, the target motion is predictable and placing these targets in the beam even after long periods between measurements was possible and thus no upper scan limit was required.

Target tracking requirements on revisit time however were tighter than those required for multistatic measurements. It is important to maintain stable tracks, so that the fusion system can input the scheduler with accurate target state vectors. A track update interval of $\frac{1}{90}$ seconds is required in the worst case for maintaining stable tracks. The mean target revisit time to ensure no track lost events occurred was 0.363 seconds for simulation

[2]One simulation batch was used for each of the Singer model parameters defined in paragraph two of this section and the batch numbers match those of the nimble simulations.

batch one. In simulation batch two, the target revisit time required reduces to 0.222 seconds. Finally, simulation batch three requires a target revisit times of 0.383 seconds. In cases where the target follows a predictable trajectory the target revisit times reduces to 53.2, 35.2 and 35.2 seconds for each of the simulation batches respectively. Most of the time, targets do not follow predictable trajectories and when they manoeuvre tightly, accelerate/decelerate rapidly or cross the beam of the radar tangentially then the worst case comes into play.

3.3 Interval Algebra

3.3.1 Allen's IA with improvements

The details of IA presented here is a rework of a conference paper [1], which was presented by the authors to the sensor management community, along with some new material that covers further interesting aspects.

IA is a method of reasoning about the temporal ordering of intervals in a consistent manner [4]. Tasks can be represented by intervals in IA and, in the case of the MSMSRN scheduling dealt with here, they either represent target measurement for a single radar or a multistatic measurement for the group of radars. The relationships capture the temporal ordering of intervals between an arbitrary start and end point in time. Here, the relationships are the temporal ordering constraints that the MSMSRN scheduler must satisfy when scheduling tasks, and these arise due to physical constraints on positioner movement and target geometries.

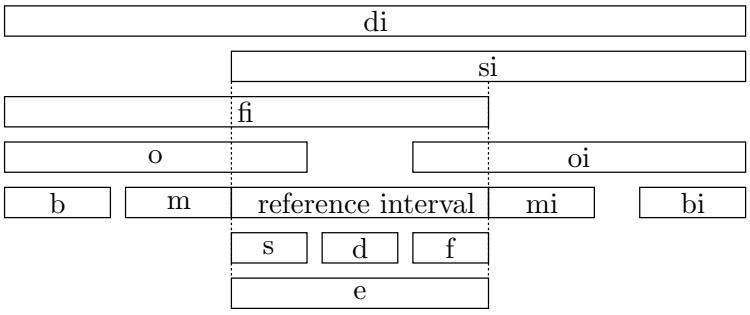


Figure 3.10: Temporal ordering of intervals in IA

Intervals are related to each other using relationships that can contain any set of the 13 temporal operators: 'before' (b), 'after' (bi), 'meets' (m), 'met by' (mi), 'overlaps' (o), 'overlapped by' (oi), 'finishes' (f), 'finished by' (fi), 'starts' (s), 'started by' (si), 'during' (d), 'contains' (di) and 'equals' (e). Figure 3.10 depicts what each of the temporal operators means in isolation when referenced to a common reference interval. The suffix 'i' associates two operators as inverse pairs, where the inverse operator imposes exactly the opposite temporal ordering. Logically, the 'equals' operator is the inverse of itself.

When no information is known about the temporal ordering between two intervals, then the relationship is the set {all} that contains all 13 temporal operators. The null set,

$\emptyset$, denotes an inconsistency, as all intervals must have some temporal ordering in relation to each other. Any other combination of operators, within a relationship, captures some degree of certainty in the ordering of intervals. Relationships also always have an inverse relationship, which can be found by placing into the inverse relationship the inverse of each operator present in the relationship to invert. The inverse operation is called 'Inverse $(\cdot)$' in the pseudo code. When considering the relationship between two intervals, there are always two reciprocal relationships that exist. Each relationship uses one of the intervals as a reference, and thus the inverse relationship denotes the relationship using the alternate interval as the reference interval.

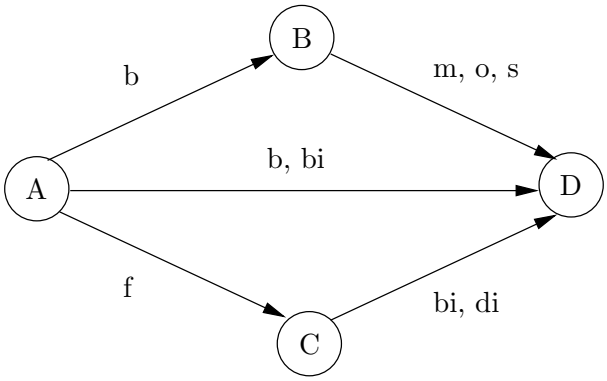


Figure 3.11: An example IA network with four intervals

Table 3.1: Resultant IA relationship matrix for the example network given in Figure 3.11

$\boldsymbol{N}$	**A**	**B**	**C**	**D**
A	e	b	f	b, bi
B	bi	e	all	m, o, s
C	fi	all	e	bi, di
D	b, bi	mi, oi, si	b, d	e

A specific scenario is then captured in an IA network, which consists of a set of intervals and all the relationships between each interval pair. The relationships of an IA network are stored in a matrix $\boldsymbol{N}$, where each cell $\boldsymbol{N}_{r,c}$ contains a relationship between an interval defined by the row r and an interval defined by the column c. The row or column number in the matrix matches the index of the interval when they are stored in a vector. An example network is shown in Figure 3.11 and the corresponding relationship matrix is captured in Table 3.1.

Cells in the upper and lower triangle of the relationship matrix, where the column

and row numbers are swapped, are always the set inverse of each other. The reason for this is that these two relationships represent the relationship between the same intervals from the two possible reference points. Relationships on the diagonal of the matrix must always contain only the 'equals' operator (the 'equals' relationship), as they determine the temporal ordering of an interval referenced to itself. For our research, each scan for the radars is planned by the MSMSRN scheduler using a single IA network, or in the case of the nimble scheduler one network per intercept.

Algorithm 1 CP ($R_{A\to B}$, $R_{B\to C}$)

```
R_{A→C} = ∅
for all n ∈ R_{A→B} do
  for all m ∈ R_{B→C} do
    R_{A→C} = R_{A→C} ∪ T_{n,m}
    if R_{A→C} = {all} then
      return {all}
    end if
  end for
end for
return R_{A→C}
```

Constraint propagation, or set composition for relationships, is captured as Algorithm 1, which is the algorithm in IA that keeps the relationships amongst any three intervals temporally consistent. Consider three intervals 'A', 'B' and 'C', where knowledge about the relationship between 'A' and 'B' ($R_{A\to B}$) as well as 'B' and 'C' ($R_{B\to C}$) is known. Constraint propagation can then determine the possible operators in the relationship between 'A' and 'C' ($R_{A\to C}$), which is consistent with the two known relationships. This is done by pairing one operator, n, from $R_{A\to B}$ with another, m, from $R_{B\to C}$ and then determining the operators permissible in $R_{A\to C}$. The permissible operators for each combination of basic operators are stored as a transitivity matrix $\boldsymbol{T}$.

The details for the contents of the transitivity matrix can be found in Allen's paper on IA [4], and has been repeated here in Table 3.2. The transitivity matrix has 13 columns and rows, where the row and column number matches one of the 13 operators, and the contents are the set of permissible operators for the unknown relationship.

Path consistency is presented as Algorithm 2, where set L contains each of the cells that have been updated. When possible, by using constraint propagation, this algorithm allows an interval to be added to an IA network such that all relationships remain consistent.

Table 3.2: IA transitivity matrix, $\boldsymbol{T}$

$\boldsymbol{T}$	**b**	**bi**	**d**	**di**	**o**	**oi**	**m**	**mi**	**s**	**si**	**f**	**fi**	**e**
b	b	all	b, d, o, m, s	b	b	b, d, o, m, s	b	b, d, o, m, s	b	b	b, d, o, m, s	b	b
bi	all	bi	bi, d, oi, mi, f	bi	bi, d, oi, mi, f	bi	bi, d, oi, mi, f	bi	bi, d, oi, mi, f	bi	bi	bi	bi
d	b	bi	d	all	b, d, o, m, s	bi, d, oi, mi, f	b	bi	d	bi, d, oi, mi, f	d	b, d, o, m, s	d
di	b, di, o, m, fi	bi, di, oi, mi, si	d, di, o, oi, e	di	di, o, fi	di, oi, si	di, o, fi	di, oi, si	di, o, fi	di	di, oi, si	di	di
o	b	bi, di, oi, mi, si	d, o, s	b, di, o, m, fi	b, o, m	d, di, o, oi, e	b	di, oi, si	o	di, o, fi	d, o, s	b, o, m	o
oi	b, di, o, m, fi	bi	d, oi, f	bi, di, oi, mi, si	d, di, o, oi, e	bi, oi, mi	di, o, fi	bi	d, oi, f	bi, oi, mi	oi	di, oi, si	oi
m	b	bi, di, oi, mi, si	d, o, s	b	b	d, o, s	b	f, fi, e	m	m	d, o, s	b	m
mi	b, di, o, m, fi	bi	d, oi, f	bi	d, oi, f	bi	s, si, e	bi	d, oi, f	bi	mi	mi	mi
s	b	bi	d	b, di, o, m, fi	b, o, m	d, oi, f	b	mi	s	s, si, e	d	b, o, m	s
si	b, di, o, m, fi	bi	d, oi, f	di	di, o, fi	oi	di, o, fi	mi	s, si, e	si	oi	di	si
f	b	bi	d	bi, di, oi, mi, si	d, o, s	bi, oi, mi	m	bi	d	bi, oi, mi	f	f, fi, e	f
fi	b	bi, di, oi, mi, si	d, o, s	di	o	di, oi, si	m	di, oi, si	o	di	f, fi, e	fi	fi
e	b	bi	d	di	o	oi	m	mi	s	si	f	fi	e

Furthermore, path consistency can be used to alter a relationship in the IA network to determine the constraints this will place on the rest of the relationships. In both cases, the temporal ordering constraints can result in an inconsistent network or they will be maintained so as to be consistent with each other. However, even when consistent, it is not guaranteed that the network as a whole will yield a consistent scenario through a single use of path consistency. Total path consistency can only be maintained through iterative use of path consistency, where each relationship in the network, in turn, is treated as though it were updated (populating L with all cells).

Initially, when adding the interval, the relationship between the newly added interval and every other interval already in the network is {all}. Next, one relationship between the newly added interval and an existing interval is updated with a known set of operators. Updates must always be done by taking the set intersection of the existing relationship and the requested update, as the IA algorithms only constrain relationships. Then every relationship in the same row and column as the updated relationship is further constrained by using constraint propagation. In this case, the relationship for the cell under test is treated as the unknown relationship and the two other relationships include the updated

Algorithm 2 PC $(\boldsymbol{N}, [r, c], R)$

```
N_{r,c} = N_{r,c} ∩ R
N_{c,r} = Inverse (N_{r,c})
add [r, c] to set L
while L ≠ ∅ do
  select and delete first [r, c] from L
  for all cells [k, l] in same row or column as [r, c] do
    if l = c then
      if not Skip (N_{k,r}, N_{r,c}) then
        R = N_{k,l} ∩ CP (N_{k,r}, N_{r,c})
      end if
    else
      if not Skip (N_{r,c}, N_{c,l}) then
        R = N_{k,l} ∩ CP (N_{r,c}, N_{c,l})
      end if
    end if
    if R = ∅ then
      return false
    end if
    if R ≠ N_{k,l} then
      N_{k,l} = R
      N_{l,k} = Inverse (R)
      add [k, l] to L
    end if
  end for
end while
return true
```

relationship and the relationship that completes the CP triangle. This process is repeated for each subsequent update that may occur during any iteration of path consistency.

Algorithm 3 Skip (R_1, R_2)

```
if (b ∈ R_1 and bi ∈ R_2)
or (bi ∈ R_1 and b ∈ R_2)
or (d ∈ R_1 and di ∈ R_2) then
  return true
else
  return false
end if
```

Some improvements to the IA algorithms as given by Allen are now listed. Algorithm 3 is used by path consistency to skip computations that will yield {all} [148]. Algorithm 1 includes an improvement that stops computation if the resultant set becomes {all} [148]. Furthermore, Algorithm 2 can return the consistency of the algorithm whenever a set becomes $\emptyset$ [148, 149].

3.3.2 Hogge's constraint propagation

Each time constraint propagation algorithm is invoked, it computes the set composition that results for the two input relationships. It is also possible to pre-compute all combinations of the set composition by executing Allen's constraint propagation using all possible inputs. The result would be a square lookup matrix where each dimension is 2^{13} long. Each cell of this matrix contains the set composition of the relationship matching the row versus the relationship matching the column. This would, however, require storing the result of 67 108 864 compositions (128 MB for 16-bit relationships). However, doing so would also regain most of the constraint propagation processing time, as only one memory lookup would be required.

Hogge devised a compromise between the ideal case above and Allen's constraint propagation [59][3]. The approach splits relationships into two lookup parameters. Seven IA operators are chosen to form the first lookup value and will be referred to as the lower operators. The remaining six IA operators yield the second lookup value and will be referred to as the upper operators. Ideally, the choice of IA operators should match their packing in a 13-bit field such that minimal encoding and decoding is required to generate these values (preferably no encoding or decoding).

Using this definition, four inputs are derived from the two known relationships and are then used to look up a value in four different matrices.

1. Matrix one is a 128 × 128 matrix, which is populated with the results of set composition of all combinations of relationships that contain only the lower operators.

2. Matrix two is a 128 × 64 matrix, which is populated with the results of set composition of all combinations of relationships that contain only the lower operators along with all combinations of relationships that contain only the upper operators.

3. Matrix three is a 64 × 128 matrix, which is populated with the results of set composition of all combinations of relationships that contain only the upper operators along with all combinations of relationships that contain only the lower operators.

4. Finally, matrix four is a 64 × 64 matrix, which is populated with the results of

[3]It is difficult to obtain this manuscript hence the full treatment of the technique here.

set composition of all combinations of relationships that contain only the upper operators.

The four retrieved relationships are then combined using a set union to give the final relationship.

This method requires four times the computation as per the ideal case (simple logical conjunction, $\wedge$, operations) but only requires storing 36 864 compositions (72 KB for 16-bit relationships). Although memory requirements are becoming less important, the Hogge method is a good compromise between storing all set compositions and computing them every time.

3.3.3 IA constraint ordering

Constraint ordering is a technique where the intervals are added in an order that is determined by the weight of each of the relationships associated with that interval to others [148]. Thus, each relationship that must be applied to the IA network is first scored based on the impact that it may have on the IA network.

As the relationship is composed of up to 13 IA operators, each IA operator can potentially be given a different score. The final score of the relationship is the combination, usually the sum, of each of the scores of the IA operators that are present in the relationship.

For example, in the weighted scoring method [148], the score is determined by the extent to which each IA operator will restrict the allowable operators in the rest of the IA network. The IA operator that has the most weight in this case is the IA 'equals' operator.

Constraint ordering is used to reduce the amount of computation required for the IA path consistency algorithm. If the intervals whose relationships have the most impact on the network are added first the computation required will be less than if they were added last. This is because relationships with a higher score will result in multiple updates on each iteration of the IA path consistency algorithm. Thus more iterations of IA path consistency are required before the network stabilises.

The number of relationships that need to be updated per iteration of IA path consistency is potentially twice that of the number of intervals present in the IA network. Thus

it is better to execute updates requiring multiple iterations of the IA path consistency algorithm while the IA network is still small.

3.3.4 Fuzzy and Bayesian IA

As mentioned in our conference paper [1], there are also two extensions to IA that increase the amount of information that can be captured in the IA network. The first uses fuzzy-logic constructs on the relationships and sometimes on durations of intervals, and the other uses probabilistic extensions to the relationships only.

Fuzzy IA has been examined by various authors and seems to be the most promising extension of IA to employ [150–153]. An advantage of fuzzy IA is that it can easily capture ordinary IA networks within the proposed framework. In Allen's IA, each operator is either fully present within the relationship or absent. However, for fuzzy IA the degree of membership is allowed to vary, as a real number, over the interval $[0, 1]$. The degrees of membership for all the operators in a relationship can therefore capture the certainty with which it may be the true relationship, and this allows fuzzy IA to capture more information than Allen's IA. Fuzzy mixing is required to update the degrees of membership during constraint propagation, path consistency and total consistency checking.

Some authors also make the duration of the intervals fuzzy [150, 152], which can be useful when the interval durations are not precisely known. The duration is captured as a trapezoid in fuzzy IA where the ramp up and ramp down are again the degree to which each point in time forms part of the interval. This extension is useful in most practical applications, as tasks generally only have a desired length, which can usually increase or decrease given the current load a system performing those tasks is experiencing.

Bayesian IA has also been proposed, where a probability is assigned to each of the operators in a relationship [154, 155]. In this case, the probability of all operators in the set must add up to one, which is a much more difficult constraint to maintain. Capturing ordinary IA networks in this framework would require keeping the probabilities equal for operators present in the relationship. Bayesian IA also requires much more complex consistency algorithms, as during all updates the probabilities must add up to one. Furthermore, updates to the relationships would require using Bayesian mathematics to ensure that the probabilities are maintained consistently, which is more computationally

expensive than fuzzy-logic mixing. Much of the finer details of Bayesian IA have not been published yet.

A good example of where fuzzy or Bayesian IA could be employed, is for the case where targets are not exactly in the middle of the beam of the radars. In this case, there is a degree of uncertainty both in the ordering of the tasks for the radars as well as their duration. The ordering uncertainty occurs when multiple targets are closer in the azimuth angle than the azimuth angle resolution of the radar measurements. Allen's [4] IA is able to handle this case through the inclusion of multiple operators, but fuzzy IA could be used to determine the true ordering by tracking the change in degree of membership over time. Such a technique will be considered in future work. Finally, the uncertain duration occurs because the actual time the beams of two or more radars would simultaneously measure the targets cannot be determined, as the radars cannot measure azimuth angle to infinite accuracy.

Another example where Bayesian and Fuzzy IA could prove beneficial is in handling random service times and deadlines for tasks [140]. In these instances, Bayesian IA may be more readily applicable. Using the tasks start time and task duration probability density functions, equations can be derived for estimating the probability for each IA operator in the relations between intervals. Hard constraints about task orders can then override these probabilities making certain operators more likely. Using both these sources of information, a Bayesian IA network can be built and then temporal reasoning can be performed by a scheduler. Each time the scheduler decides on a specific task service time and the corresponding deadline, the Bayesian IA network must be updated. The usual rules of IA apply in removing constraints that are no longer possible. However, the probabilities of the remaining constraints must be updated according to the derived probability estimation equations. At each point, it is still important to ensure that the IA network remains consistent. Inconsistent networks effectively discount certain scheduler actions and would require maintaining a backup of the network prior to scheduling decisions. Finally, when there is no more scheduling freedom, the task ordering will be given directly by the relationships remaining in the IA network.

3.3.5 Using IA with parallel processing architectures

Many parallel versions of arc consistency algorithms have been published [51–55]. This CSP algorithm checks that for any two nodes in the network the labels are consistent. It has been the preferred route of checking the consistency for many classes of CSPs. This is due to the fact that arc consistency requires much less processing that path consistency. Often arc consistency is sufficient to find solutions to the CSP. Unfortunately, arc consistency does not guarantee that a solution will be obtained for more complex CSPs. Thus, if full IA is required then arc consistency might not be sufficient, and path consistency is required. Nevertheless, arc consistency can be used as a pre-processing step to find a locally optimal arc-consistent solution.

A general consideration of arc consistency and parallel versions for each of the known serial arc consistency algorithms are proposed by Samal and Henderson [51]. Interestingly, the parallel version of the simplest serial algorithm almost achieves the maximal theoretical speed-up. Results are obtained by running the code on a proprietary super computer[4].

Kasif [52] considers parallel versions of discrete relaxation, another arc consistency algorithm. It is shown that propositional CSPs and satisfiability of proposition Horn classes have a tight connection. Thus, discrete relaxation algorithms are inherently sequential in the general sense and cannot always gain from parallel processing. This is important to consider for future CSPs and can be satisfied by checking that the parallel version does speed up the computation in general.

Cooper and Swain [53] looks at how domain dependence can speed up computation by removing redundant computations. Theoretical results are obtained for Integrated Circuit (IC) designs and practical results using a proprietary super computer[5]. Future work should consider how domain dependence can be accommodated for GP-GPU processing, especially for parallel path consistency. This may be able to reduce the amount of redundant computations and thereby possibly improve the achievable speed-up. Furthermore, IA processing making use of FPGAs could make use of the documented logic transforms.

Kirousis [54] considers a special form of CSPs known as implicational CSPs. A theoretical analysis is given to show that the algorithm is indeed correct. In an implicational

[4]BBN Butterfly Computer
[5]The Connection Machine

CSP the presence of a value at a node implies certain relationships and vice versa. The research documenter here does not consider this type of CSP, as it is not clear how sensor scheduling can be accommodated. However, it may be interesting to consider the application of implicational CSPs and how these can benefit from GP-GPU processing.

Fabiunke [55] looks at distributed agents that act in isolation to solve arc consistency. In this case, the nodes do not communicate but instead must use the information in the CSP network to decide on the next value. He effectively proposes a CSP algorithm that is modelled around parallel distributed processing as originally applied to artificial neural networks. This approach requires some randomisation and could help FPGA and MCPU algorithms but would not run efficiently on GP-GPUs.

Susswein et al. [49] and Keretho et al. [50] have analysed parallel path consistency algorithms that can be used to speed up the execution of path-consistent searches on CSPs. Path consistency looks at three nodes in the CSP and ensures that the relationships are consistent. Again path consistency does not guarantee total consistency but in some instances a path-consistent solution is sufficient. Path consistency is sufficient for certain types of multisensor scheduling. In particular, this is the case for MSMSRN scheduling.

Ladkin and Maddux [57] also mention the use of a parallel path consistency algorithm for IA that computes the set composition over the entire IA relation matrix. The form presented is the most general form of path consistency for any constraint programming technique. Several improvements over the basic form are used to reduce the computation required [58], including only storing half the IA relation matrix and computing only the required set compositions (as per Allen [4]). The documented algorithms do not use the improved constraint propagation algorithm of Hogge [59].

Point Algebra (PA) is another temporal language for constraint satisfaction problems similar to IA, which only has three basic operators ('before', 'equal' and 'after' in IA terminology). More recent work by Gerevini and Saetti [60] has considered how PA could be adapted for parallel processing architectures. This time the researchers have applied the field of mathematics known as meta-graph closure for time series and serial-parallel meta-graph representations of PA. Further work would be required to apply these techniques to IA but doing so could certainly be advantageous.

For difficult CSPs, path and arc consistency are not sufficient to provide a solution, and

then total network consistency algorithms would be required. Nebel [61] looks at parallel forms of global consistency and in particular analyses the use of the ORD-Horn[6] subclass of IA in the splitting function. The ORD-Horn is a special subset of IA [3], which is known to be solvable using path consistency. The splitting function checks to see if a relation belongs to the ORD-Horn class. As the ORD-Horn subclass covers 10% of IA, it results in the least amount of backtracking. The author finds that in 40% of his tested cases the ORD-Horn subclass finds the solution first. While total consistency should benefit from GP-GPU processing we leave this as future work. The three IA subclasses and two methods of producing random scenarios used in this research are aligned with this work.

[6]Ordered Horn refer to ORD-Horn in the glossary.

3.4 Greedy Randomised Adaptive Search Procedure

GRASP is a meta-heuristic search procedure that comprises three main steps, as per Algorithm 4:

1. Generate a Restricted Candidate List (RCL) that will be used to generate solutions. In the case of the G-MS algorithm the RCL is the priority queue of targets, and is the Q input to Algorithm 4. The priority queue is explained in Section 4.2.

2. Next, construct an initial solution from the RCL that adheres to the solution space. Each radar supplies the azimuth angle ordering of targets as L_1 and L_2. The azimuth angle orderings can be seen as a set of vectors, where each vector contains a cluster of targets with uncertain order, and the sequence of vectors denotes the true ordering of the clusters in relation to each other. The construction algorithm used by the G-MS algorithm is presented as Algorithm 5, making use of Algorithms 6 and 7.

3. Search the solution space by performing a local search around the candidate solution from the previous step. The local search algorithm is defined as Algorithm 8, making use of Algorithms 6, 7, 5 and 9. When compared to genetic algorithms, the local search of GRASP can be likened to the mutation operation used for genetic algorithms.

Algorithm 4 GraspSchedule (Q : target priority queue, $\boldsymbol{A}$: target order for all radars, k : iteration)

```
create best solution b = ∅
for i = 1 → k do
    create candidate solution c = ∅
    c = GraspConstruction (Q, A, c)
    c = GraspLocalSearch (Q, A, c)
    b = GraspUpdateSolution (Q, c, b)
end for
for all radars r do
    GraspCheckSolution (b, A_r)
end for
return b
```

The last two steps are repeated iteratively and the number of iterations to perform, k, is a parameter that can be tuned for the GRASP algorithm. At the end of the last step,

during each iteration, the best solution found during the local search is then compared to a stored-best solution. If the current solution is better than the stored-best solution, the current solution becomes the stored solution.

When the GRASP algorithm completes, the best solution found after all iterations will be returned as the sequence of targets to measure using the multistatic mode of the radars. The low-level scheduler will then ensure that the radars will make simultaneous measurements of the targets contained in this list by speeding up or slowing down either of the radar positioners.

Algorithm 5 constructs an initial solution that will be used as the starting point during the GRASP local search algorithm. The algorithm receives the priority queue of targets (Q), the azimuth angle ordering of targets ($\boldsymbol{A}$) for all radars and a list of already scheduled targets ($\boldsymbol{s}$). During the execution the algorithm will attempt to add targets randomly selected from the priority queue to the list of scheduled targets using Algorithm 6. If the azimuth angle ordering of neither radar is violated the target will be added, which is checked in Algorithm 7. The final updated list of scheduled targets is returned to the calling algorithm and this parameter can be seen as an input and output parameter.

Algorithm 5 GraspConstruction (Q : target priority queue, $\boldsymbol{A}$: target order for all radars, $\boldsymbol{s}$: scheduling vector)

```
for all t ∈ Q select randomly according to priority do
  if t ∉ s then
    for all radars r do
      N_r = GraspAddToSolution (t, s, A_r)
    end for
    if GraspFindMatchingSolutions (N, m) then
      s = m
    end if
  end if
end for
return s
```

Algorithm 6 generates a list of possible solutions of how a target (t) could be added to a list of scheduled targets ($\boldsymbol{s}$) adhering to the azimuth ordering ($\boldsymbol{A}$) specified by a radar low-level scheduler. The algorithm first finds the location of the target in the azimuth ordering of the radar, and it will determine the scan sector (r) and the element number (c) in the scan vector. Only the scan sector ordering must be adhered to as targets within a scan

sector will be illuminated simultaneously by the radar beam regardless of the scheduling sequences. Then the algorithm iterates through the list of scheduled targets determining the first insert point and last insert point that will adhere to the scan sector ordering. The first insert point will be just before the first target that is in the same sector or a subsequent sector. The last insert point will be just after the last target that is in the same sector. Using these two insert points a number of solutions are generated. The number of solutions generated will depend on the difference between the two insert points plus one.

Algorithm 6 GraspAddToSolution (t : target number, $\boldsymbol{s}$: scheduled targets vector, $\boldsymbol{A}$: target order for a radar)

find row and column $[r, c]$ of $t \in \boldsymbol{A}$
$f = 0$
$l = 0$
sort $\boldsymbol{s}$ in target order $\boldsymbol{A}$ for radar
for all $k \in \boldsymbol{s}$ in sequence **do**
  find row and column $[i, j]$ of $k \in \boldsymbol{A}$
  if $i < r$ **then**
    $f =$ index of k in $\boldsymbol{s}$
    $l =$ index of k in $\boldsymbol{s}$
  end if
  if $i = r$ **then**
    $l =$ index of k in $\boldsymbol{s}$
  end if
end for
$R = \emptyset$
for $k = f \rightarrow l$ **do**
  $\boldsymbol{v} = [\boldsymbol{s}_{1 \rightarrow k}, t, \boldsymbol{s}_{k+1 \rightarrow \text{length}(\boldsymbol{s})}]$
  $R = \{R, \boldsymbol{v}\}$
end for
return R

Algorithm 7 receives two inputs: a set of scheduling vectors for radar one (S_1) and a set of scheduling vectors for radar two (S_2). It also receives a matched scheduling vector output parameter ($\boldsymbol{m}$) that will contain a matching solution from the two scheduling vector sets if found. The algorithm searches within the two scheduling vector sets for a matching scheduling vector. If the vector is found the algorithm returns Boolean 'true' and sets the matched scheduling output parameter appropriately. Alternatively, the algorithm returns Boolean 'false' and does not modify the matched scheduling output parameter.

Algorithm 8 performs a local search around the initial scheduling solution found by Algorithm 5. The algorithm receives three inputs: a priority queue of targets (Q), the

Algorithm 7 GraspFindMatchingSolutions (S : set of integer vectors per radars, $\boldsymbol{m}$: scheduled target vector)

```
for all s_1 ∈ S_1 do
  create match flag f = true
  for all radars r ≠ 1 do
    for all s_r ∈ S_r do
      if length (s_1) = length (s_r) then
        for i = 1 → length (s_1) do
          if s_{1,i} ≠ s_{r,i} then
            f = false
          end if
        end for
      else
        f = false
      end if
      if f = true then
        continue to next radar
      end if
    end for
  end for
  if f = true then
    m = s_1
    return true
  end if
end for
return false
```

azimuth angle scan ordering for all radars ($\boldsymbol{A}$) and the initial scheduled list of targets ($\boldsymbol{s}$). The algorithm returns the best scheduling solution found ($\boldsymbol{s}$). The set of scheduling solutions is generated during the execution of the algorithm by randomly selecting targets to add to the initial scheduled list of targets. The GRASP search algorithm generates solutions by using a modified version of the GRASP construction algorithm. After the randomly selected target is added, the GRASP construction algorithm (Algorithm 5) is once again called to add any remaining targets to the solution generated. Each of the generated solutions is then compared to each other using the GRASP fitness algorithm (Algorithm 9). Only the solution generated or the initial solution with the highest fitness score is kept and returned as the new list of scheduled targets.

Algorithm 8 GraspLocalSearch (Q : target priority queue, $\boldsymbol{A}$: target order for all radars, $\boldsymbol{s}$: scheduled targets vector)

```
R = {s}
for all t ∈ Q select randomly according to priority do
  if t ∉ s then
    v = ∅
    M = {t, k ∈ s}
    for all i ∈ M do
      for all radars r do
        N_r = GraspAddToSolution (i, v, A_r)
      end for
      if GraspFindMatchingSolutions (N, u) then
        v = u
      end if
    end for
    v = GraspConstruction (Q, A, v)
    R = {R, v}
  end if
end for
m = −∞
for all k ∈ R do
  f = GraspFitness (Q, k)
  if f >= m then
    s = k
    m = f
  end if
end for
return s
```

Algorithm 9 simply consults the target priority queue (Q) and sums all the priorities of all the targets present in the scheduled list of targets ($\boldsymbol{s}$). The result is returned as

the fitness value for the list of scheduled targets. Thus longer target sequences as well as target sequences with high priority targets are preferred.

Algorithm 9 GraspFitness (Q : target priority queue, $\boldsymbol{s}$: integer vector)

```
f = 0
for t ∈ s do
  p = priority of t in Q
  f = f + p
end for
return f
```

Algorithm 10 is used to keep a globally optimal solution during each iteration of the GRASP scheduling algorithm. The algorithm receives the priority queue of targets (Q) which is used to determine the fitness of each of the scheduled list of targets ($\boldsymbol{s}_1$ and $\boldsymbol{s}_2$). The scheduled list of targets with the highest fitness value is returned as the solution to keep.

Algorithm 10 GraspUpdateSolution (Q : target priority queue, $\boldsymbol{s}_1$: integer vector, $\boldsymbol{s}_2$: integer vector)

```
f1 = GraspFitness (Q, s1)
f2 = GraspFitness (Q, s2)
if f1 > f2 then
  b = s1
else
  b = s2
end if
return b
```

Algorithm 11 checks the solutions generated by the GRASP scheduling algorithm for validity. The algorithm receives the scheduled list of targets ($\boldsymbol{s}$) as well as the azimuth angle ordering of targets for a radar (L). The ordering of targets within the scheduled list of targets is checked against the azimuth angle ordering. Only if the orderings match will the scheduled list of targets be valid. The scheduled list of targets is also checked for any duplicate targets, as a target can only appear in the list once.

Algorithm 11 GraspCheckSolution ($\boldsymbol{s}$: integer vector, L : set of integer vectors)

```
(i, j) = (0, 0)
for all s_k ∈ s do
  find location (r, c) of s_k in L
  if r < i then
    s = ∅
    return false
  end if
  for all s_l ∈ s | k ≠ l do
    if s_k = s_l then
      s = ∅
      return false
    end if
  end for
  (i, j) = (r, c)
end for
return true
```

3.5 Conclusion

Multistatic radar is a type of sensor system that employs the strengths of both monostatic and bistatic measurement capabilities. Through either the use of electromagnetic interference or orthogonal signals, better or more measurements can be made of a target of interest. Information fusion can be used to combine the information into better tracks of the targets. This means that surveillance systems employing multistatic radars can better discern the location, velocity and acceleration of these targets.

This chapter introduces the necessary concepts in radar theory required to schedule multistatic radars. Furthermore, the more practical concerns that must be addressed when scheduling a mechanically steered surveillance radar network to make multistatic measurements of targets are given. These include: uncertainties in the azimuth sequence of targets; the effect of target motion and therefore the need for scheduling nimbleness; handling of monostatic scanning, tracking and confirmation tasks; and the implementation of scheduling algorithms.

Interval Algebra is an interesting temporal reasoning language for tasks (intervals) that need to be performed. This means that a timeline of tasks can be built that consist of a realisable scheduling plan. Thus IA is useful for scheduling sensors, as the goal in sensor scheduling is to use limited sensor resources to perform as many tasks as possible. Examples of tasks for multistatic radars includes detection, tracking and imaging of a target of interest. Interval Algebra contains a rich set of tools, including temporal and durational organisation, and fuzzy and Bayesian logic, to enable the scheduling of tasks for any sensor.

Finally, GRASP was introduced so that it may be compared to IA. GRASP, a meta-heuristic optimisation algorithm, is an algorithm that is representative of the more conventional scheduling approaches found in the literature.

Chapter 4

Approach

4.1 Introduction

This section builds on the methodology defined in Section 1.6 and is an expanded version of Section 1.7 to prove the hypothesis as outlined in Section 1.5. All material and methods used during the research are described in this chapter and the appropriate sections here should be read before consulting the relevant discussion chapters: Chapters 5 and 6. This chapter is thus divided into two main method sections, where the first introduces the Mechanically Steered Multistatic Surveillance Radar Network (MSMSRN) scheduling problem used to compare Interval Algebra (IA) to Greedy Randomised Adaptive Search Procedure (GRASP) and the second introduces the Monte-Carlo simulation environment used to compare parallel and serial total IA consistency on modern parallel processing architectures.

Section 1.5 captures the research hypothesis, goals and objectives. The research questions that would need to be answered in proving the hypothesis are also outlined. This is followed by the methodology that was followed in proving the hypothesis, reaching the goals and meeting the objectives in Section 1.6. These sections form part of the executive summary and should be read before continuing with this chapter.

Section 4.2 begins with a recap of the MSMSRN scheduling environment for a simplistic surveillance radar network[1], as mentioned in Chapter 3. Multisensor management (Section

[1]This section was the basis for the method sections of the author et al.'s research paper published by the Elsevier Information Fusion journal (Section A.3).

1.3.1), multistatic radar (Section 1.3.1) and IA (Section 1.3.2) are then also mentioned in the context of the investigations to be performed. How IA and GRASP MSMSRN scheduling algorithms are evaluated against this surveillance radar environment is given in Section 4.2. Comparisons are drawn in Chapter 5 by testing exactly the same set of test scenarios. Next the implementation of scheduling algorithms is discussed in Section 4.2.1. Note the reason for choosing GRASP was formulated in Section 1.6.1.

Section 4.3 begins with an introduction of the current research that is being performed using IA algorithms on parallel processing architectures[2]. It compares the research documented in Chapter 6 with prior research. It then gives the details for the three sets of Monte-Carlo simulations that were run. Each set of simulations was designed to investigate the effect of changing one independent variable on the processing time of both parallel and serial IA. The first set of simulations varies the number of intervals in the network, the second set varies the expressiveness of the operator set used within the algebra and the final set varies the number of unknown relationships in the IA network.

Finally, Section 4.4 summarises the key points of the chapter.

[2]This section was the basis for the method sections of the author et al.'s research paper published by the Elsevier Information Fusion journal (Section A.3).

4.2 Multistatic Radar Scheduling Investigation

The results of two different sets of simulations are presented to achieve the following goals. Firstly, a comparison is drawn between IA and the Greedy Randomised Adaptive Search Procedure (GRASP). Secondly, an investigation into a nimble scheduler, where targets are highly manoeuvrable, is presented.

The methodology followed for the comparisons of IA to GRASP is the same as the experimental set-up of my previously published work [1]. For this comparison, the simulations also made use of a target priority queue to prioritise targets as depicted in Figure 4.1. These simulations also simplify the information fusion system by only simulating fused tracks. This was done to keep the simulations tractable such that many simulations could be executed for the Monte-Carlo analysis.

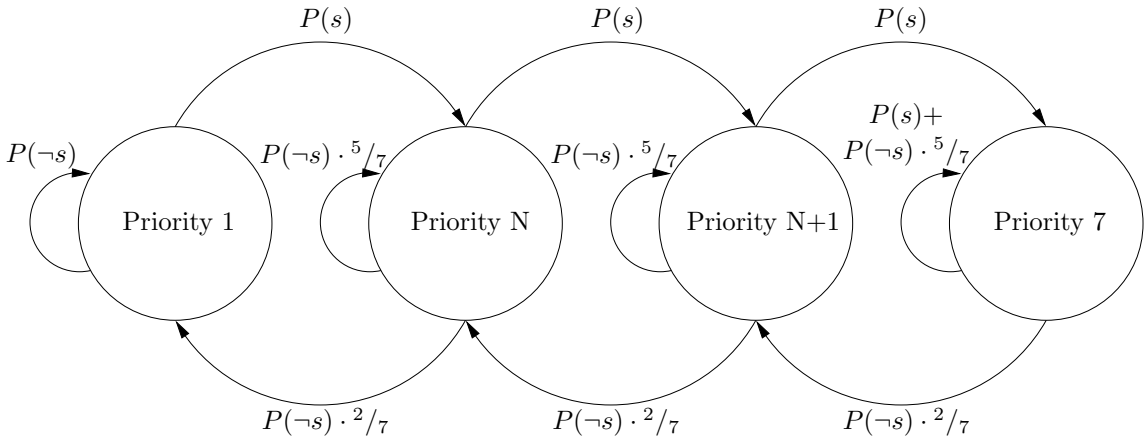


Figure 4.1: Markov chain target priority state transition diagram. This Markov chain represents the transition of target priorities. This target priority mechanism is used to ensure targets are visited frequently enough and that problematic targets never cause the system to focus solely on these targets. The states represent the priority of the target, where 1 is a high priority and 7 a low priority. Limiting the priority levels ensures that targets cannot become starved of measurements nor can then prohibit the system from making measurements of other targets. Each transition has an associated probability of occurring and is shown in the figure as the function $P(\cdot)$. In each of these transitions the scheduling event is shortened as s, whereas $\neg s$ is the event where the target is not scheduled.

The nimble scheduler simulations require a more realistic approach. Thus, unlike in the previous work, the simulation environment makes use of Kalman tracking filters [62] (Appendix C.2). The tracks of the radars are then fed to a central information fusion system. Here the tracks are associated using the auction algorithm [63] for data associa-

tion (Appendix C.3). An information filter [64] is then used to fuse the associated tracks (Appendix C.4). Finally, the covariance of the information filter is used to prioritise the tracks. When managing real sensors this is a better approach for prioritising targets. However, it may still be desirable to consider a mechanism whereby all targets are frequently revisited.

4.2.1 Implementation of scheduling algorithms

Quality of the measurements affects the scheduling but will only make a sensor management approach more or less efficient. The most common method of catering for this with sensor management is to incorporate a task prioritisation step, which is not the focus of this study. Nevertheless, two scheduling approaches should perform equally well with the same task prioritisation approach. Thus, to remove the effects of task prioritisation, the probability of detection was set to one for the comparison simulations.

Later to test nimble scheduling a more realistic assumption of probability of detection other than one was required, as this will have an impact on whether a nimble approach is required. For these simulations the probability of detection (P_d) was set to 0.9 for monostatic/bistatic measurements and 0.99 for simultaneous monostatic and bistatic measurements using Bayesian probabilities. Either one (0.9×0.1) or both measurements (0.9×0.9) detect the targets leading to $2 \times 0.9 \times 0.1 + 0.9 \times 0.9 = 0.99$. The decision for setting the probability of detection to 0.9 was made arbitrarily and the assumption was that all targets are the same and the radar performance is uniform over the detection range.

In a realistic system, this would naturally not be the case and a more holistic optimisation approach should include the fidelity of the measurements. Again, however, it was not the focus of the research to look at task prioritisation techniques. Task prioritisation along with task generation algorithms can certainly be considered for follow on research and will be mentioned in the future research section.

The target priority queue is initialised with all targets having the highest priority level for the comparison simulations when the simulation starts. Targets are then placed at their initial 2D positions and will travel on their selected 2D paths. In all the comparison simulations, targets were always visible to both radars when they were scanning over the

area. For the nimble scheduling simulations, the probability of detection (P_d) was set to 0.9 for monostatic measurements and 0.99 for multistatic measurements. Furthermore, the probability of false alarm (P_{fa}) was set to 10^{-6} per radar cell (radars were assumed to have a 3.75 m resolution and a 1°azimuth resolution).

False alarms can cause false tracks but due to the fact that there will be no corresponding false track from the other radars at the same location, the scheduler will not be affected by these. False alarms will only create tracks if the M out of N rule passes, where M was set to 3 and N set to 5. Furthermore, false tracks are starved of measurements and do not last long. Only fused tracks are used as inputs to the scheduler and thus in the event that two radars generate a false track at the same location will the scheduling performance be affected. In this case, no multistatic measurement will be made but track deletion is also accelerated.

If the target is not detected, the track is coasted and potentially, a multistatic measurement can be missed due to the resulting target location accuracy decreasing. This can be mitigated by allowing the radars to dwell longer in the location, provided the target has not moved to far off nor is occluded for both radars.

A recognised surveillance picture is assumed to be perfectly generated by a fusion system for the comparison simulations. However, for the nimble simulations the recognised surveillance picture is built by first detecting targets, then forming tracks, associating the tracks of the two radars and finally fusing the associated tracks. The recognised surveillance picture is used by the radar schedulers to generate a target azimuth ordered list, which may contain uncertainties in the ordering of some targets.

The surveillance volume is then divided into azimuth sectors for each radar. The simplest approach is to merely divide the volume into regions that are the width of the radar beam. This will mean that some targets that have angular uncertainty in their measurements for either radar the scheduler would be unable to make use of this to optimise the multistatic measurements. Nevertheless, for the comparison simulations this approach was followed as both algorithms would be affected equally and no bias in the results would be created. Doing so allowed the simulations to be executed more efficiently and, thus, allow more simulation executions to be performed.

For the nimble simulations, this simple division would lead to a bias in the amount of

missed multistatic measurements, as effectively this scenario represents multistatic that could have been made but were ignored. Thus, for these simulations, a more dynamic approach was taken towards generating the azimuth sectors. Azimuth sectors were adjusted so that when targets are predicted to be closer in azimuth than the beam width of the radar, where the centre point of the azimuth sector would lie on the angle that was between the two targets. Using the predicted location of the targets to the time of the beam intersection ensured that fewer multistatic measurements would be missed if the targets moved sufficiently far apart to no longer reside in a single azimuth sector.

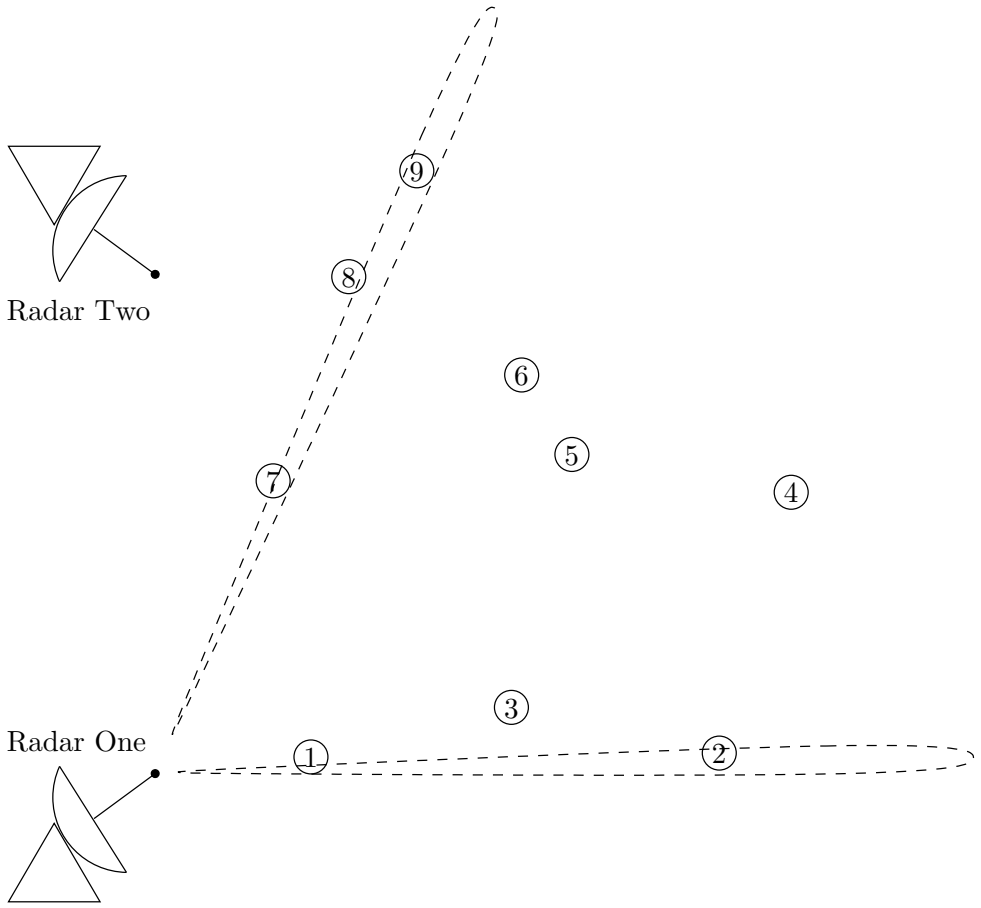


Figure 4.2: Scheduling example scene: The target locations are formed through creation of tracks by the two radars, which are then fused at the fusion centre. Then the scheduler determines which azimuth bin contains each of the target in relation to both radars. Targets that are close together in azimuth angle for one radar can be placed in the same azimuth bin for that radar. Azimuth sectors for the research were chosen to be the scan angle divided by the beam width. Certainly, a potentially better approach would be to choose sectors based on the clustering of targets. Note that targets 1 and 2, as well as 7, 8 and 9 reside in the same azimuth bin for radar one, where the beam of the radar is shown in dotted lines to clarify the uncertainty in these angular measurements.

Table 4.1: Table demonstrating input format where multiple entries in a column indicate an uncertainty in the order

1	3	4	5	6	7
2					8
					9

Table 4.2: Table demonstrating output format

1	2	5	7	8	9

Before a scan commences, each of the MSMSRN scheduling algorithms accepts two target lists with uncertainties from the simulation environment, one list for each radar that scans over the area. An example of these lists can be found in Table 4.1 for radar one in the scene depicted in Figure 4.2, where the columns indicate discrete azimuth sectors and the rows are individual targets within the sector. The nimble scheduler is also supplied with two updated target lists once each interception point has been reached.

The MSMSRN scheduling algorithms select the targets measured in the multistatic mode by taking into account the priorities and the azimuth angle orderings per radar for each target. This list of multistatic measurements is then returned to the simulation environment (refer to Table 4.2). Finally, the simulation environment executes the next scan as quickly as possible under the positioner slew rate constraints. This is achieved by ensuring that the radars illuminate the targets in the list simultaneously.

If used, the target priority queue is updated for the next scan at the end of the current scan. This is done using the final list of targets that was selected for multistatic measurement. Targets that were scheduled decrease in priority, and unscheduled targets can either stay at the same priority level or increase in priority (refer to Figure 4.1).

4.2.2 Rationale for research decisions

All algorithms implemented for the multistatic radar scheduling investigation made use of Matlab®. This does not include the parallel processing versions where the underlying IA implementation used C and OpenCL. Matlab is very good for prototyping and comparison/exploration but is not necessarily the best choice for reducing computation time. Nevertheless, results obtained are valid as the research only needs to compare the algorithms. It is already known that GRASP can be used in a realistic system. Thus in

a comparison where IA matches or exceeds the performance within reasonable execution times, it can be proven to be an efficient and effective scheduling algorithm. Using a more non-scripted programming language would provide even better scheduling performance for the same computation time.

Secondly, whether IA can handle and improve realistic requirements of a Mechanically Steered Multistatic Surveillance Radar Network (MSMSRN) scheduler is to be evaluated. This second test is also necessary as any effective scheduling algorithm should handle all real-world requirements, which the sensors and environment impose on the algorithm. Research on this aspect need not be realistic in execution time, as the requirement is to prove that the algorithm can handle the complexity of the solution space. Proving that the algorithm can also execute in real time completes the proof of the research question. Although the focus is on a MSMSRN, the results should be relevant to other types of sensor networks or systems. The scheduling algorithms would, however, need to be adapted to handle the peculiarities of any new sensor employed.

Initial results of a fair comparison of IA to GRASP showed that IA requires much more processing time to achieve similar scheduling performance. The performance metric is the mean number of multistatic measurements made during a scan. This is for the case where the sensors only require a subset of the constraints that IA can handle. Therefore, to evaluate the effectiveness of IA for simpler constraint satisfaction problems, IA had to be optimised so that it no longer computed using all available constraints. The literature was consulted to see if good published techniques could be adapted for reducing the complexity of IA. There was no need to optimise GRASP as it is a meta-heuristic search procedure, which quickly converges to good results.

For all simulations it was sufficient to make use of a binary radar network, as this problem was already complex enough to test the algorithms against realistic constraints. The results obtained are, however, in no way limited to such a binary radar network. It was necessary to highlight the way the results can be extrapolated to a multisensor network. For all simulations, multiple Monte-Carlo runs had to be executed to arrive at suitable mean values when varying each independent variable. Each run made use of a random seed to generate a completely random scenario. In doing so, all biases towards specific set-ups could be averaged out for the scheduling algorithms. Furthermore, slight

variations in execution time could also be handled. The comparisons all used the Standard Error of the Mean (SEM) to ensure that there was statistical significance in the results.

4.3 Parallel Processing Architecture Investigation

Testing is performed by generating random CSP networks by two of the three methods given by Nebel [61]. The methods are denoted by A(n, d, s), H(n, d) and S(n, d, s) and are repeated here for completeness. The three parameters are the number of nodes 'n', the degree of constraints known 'd'[3] and the average constraint size 's'.

The A(n, d, s) method generates random instances of n intervals with degree d. The network size also determines the number of independent relations as $\frac{n(n-1)}{2}$. The degree 'd' randomly selects $\frac{d}{n-1}$ relations of the independent relations. Thus, the degree can vary over the interval $(0, n-1]$. The selected intervals will contain randomly generated constraints. Relations not selected are the set of all operators or {all}.

The average constraints size 's' denotes the average number of intervals in a selected relation. For the selected relations, a uniform random number from 1 to 13 is used to select an initial operator. One operator is then randomly selected for exclusion uniformly from the remaining operators. Finally, the rest of the operators are selected uniformly using the value $\frac{s-1}{13}$. Thus s can vary over the interval $[3, 13]$.

The H(n, d) method generates instances as per A(n, d, s), but instead of randomly generating constraints, the edges are selected from a set of 3 006 particularly difficult constraints. These constraints have three or more clauses when considering their logic form [3]. This method was not used as it is a subset of the other two methods. Furthermore, the region of complex solutions is similar enough to the other methods such that results can be extrapolated.

The S(n, d, s) method generates a random instance that is guaranteed to be consistent. This is done by generating a random network as per method A(n, d, s). Then a consistent scenario is generated by generating n random intervals by creating defined start and stop points. The constraints for this scenario are then extracted and added to the randomly generated network.

The known set of subclasses of IA mentioned by Nebel [61] were used to select specific values for the average constraint size 's'. These are the continuous-endpoint class ($\mathcal{C}$),

[3]Relations not known are the set of all operators or {all}

the Pointisable class[4] ($\mathcal{P}$) and the ORD-Horn class ($\mathcal{H}$). Furthermore, we also consider Point Algebra (PA) (or $\mathcal{D}$) and IA ($\mathcal{I}$), which gives five different values for s. The strict hierarchy is $\mathcal{D} \subset \mathcal{C} \subset \mathcal{P} \subset \mathcal{H} \subset \mathcal{I}$.

4.3.1 Monte-Carlo simulations

Parallel processing results were obtained by executing the OpenCL algorithm given in Section 6.2. The parallel algorithms execute on a GP-GPU[5] using a MCPU[6] to execute the host code. The serial version runs on a single core of the same MCPU. The execution time is measured using the Linux system timer[7], which measures the time accurate to around 1 ms. The timer value before and after execution of either the serial or parallel algorithm is differenced. In practice, parallel processing has overheads as specified in Section 6.2. All overheads should not exceed the serial execution time less the parallel execution time for the method to be effective.

Two Monte-Carlo simulations were run that consisted of three independent batches. The first simulation generates random scenarios using the A(n, d, s) method, while the second uses the S(n, d, s) method. Each batch varied by one independent variable, which are either the network size n, the degree of constraints d or the size of constraints s.

The first batch of a simulation tested the effect of increasing the number of nodes in the CSP network. For this batch the ratio $\frac{d}{n-1}$ of constraints was set to $\frac{1}{2}$. The average constraint size in terms of operators in the randomly generated relations was the set to 9.8. The number of intervals was then varied between 8, 16, 32, 64 and 128. The maximum size of 128 was limited by the external memory of the GP-GPU used.

The second batch of a simulation tested the effect of changing the degree d of constraints that are not the set of all operators. The ratio was varied between $\frac{1}{4}$, $\frac{3}{8}$, $\frac{1}{2}$, $\frac{5}{8}$ and $\frac{3}{4}$. In this batch the number of nodes in the CSP network was 128. This gives the values for d as 31.75, 47.625, 63.5, 79.375 and 95.25. Furthermore, the constraint size s in terms of operators in the randomly generated relations was 9.8.

The third and final batch of a simulation tested the effect of increasing constraint size

[4]The Pointisable class considers the end points of intervals as points and is not to be confused with point algebra, where nodes are points and not intervals.

[5]Nvidia GeForce GT540M

[6]Intel Core i7-2670QM @ 2.20 GHz

[7]Ubuntu Linux Long Term Support (LTS) 12.04

s in the CSP network. The five subclasses of IA were used leading to the values for s as: 3, 7.5, 8.6, 9.8 and 13. In this batch the number of nodes in the CSP network was 128. Furthermore, the ratio $\frac{d}{n-1}$ was kept at $\frac{1}{2}$.

A single iteration of the batch then executes both algorithms on the same random scenarios. The random scenario is generated once per iteration using the same method and the values for n, d and s selected. Thus six Monte-Carlo batches were run in total, where each made use of 1 000 iterations. The results for these simulations are captured in separate figures in Section 6.3.

4.3.2 Rationale for research decisions

This investigation only considers the parallel processing performance enhancement that IA path consistency can achieve. The reason for this is that our prior investigation into MSM-SRN scheduling indicates that only path consistency is required. Arc consistency, a subset of path consistency, seems to be less important, but total consistency may be required for more complex multisensor scheduling. Nevertheless, research has been published into both of these consistency algorithms, and these should also be able to be implemented similarly for GP-GPUs.

OpenCL was chosen as the computing language to implement the parallel path consistency algorithm. This programming language readily enables the use of MCPUs, GP-GPUs and FPGAs. However, it must be noted that even so any algorithm implemented in OpenCL is not guaranteed to be efficient or even provide speed-ups on any of the supported parallel processing architectures. GP-GPUs were then selected as they seem to be the most promising architecture for path consistency to exploit. They have thousands of computation units and each unit has a very small thread launch overhead. FPGAs could also have been considered, but these are less prevalent as co-processing units and are much more expensive.

Monte-Carlo simulations had to be run to remove the variation in execution time achieved given that most systems use a modern operating system. Nevertheless, variation in computation time was also constrained by ensuring that no other computational load was present on the system. Published research into total path consistency was used to identify methods of generating random IA networks. Two techniques were used from the

work of Nebel [61]: one generates mostly inconsistent networks and the other consistent networks. Furthermore, four sub-algebras of Interval Algebra was evaluated alongside IA. The sub-algebras used are PA, the continuous-endpoint class, the Pointisable class and the ORD-Horn class [3]. These algebras can be useful in multisensor scheduling even though the multistatic radar scheduling investigation made use of Reduced PA[8].

[8]Only using the 'before' and 'after' IA operators

4.4 Conclusion

The hypothesis that *Interval Algebra, as a temporal reasoning language, should perform scheduling of sensor dwells efficiently and effectively* is outlined. Three research questions that if answered will prove the hypothesis are given. Can Interval Algebra allow a multistatic radar system to make more multistatic measurements of targets? Can Interval Algebra perform as well as established multisensor scheduling techniques in terms of computational requirements? Is it possible to enhance the performance of Interval Algebra by making use of parallel processing architectures, and if so, what performance increase is to be gained? Next, the methodology followed in arriving at these research questions and the method in which they are answered is given.

The first two research questions are answered by considering Mechanically Steered Multistatic Surveillance Radar Network (MSMSRN) scheduling. The simulation environments, defined in Section 4.2, give a suitable statistical framework to test the MSMSRN scheduling algorithms[9]. Both define a binary multistatic surveillance radar network and a rectangular observed area. The radars are mechanically steered and the rotators do not change direction until a scan is completed. Using these simulations allows the performance of scheduling algorithms utilising IA to be compared to those utilising GRASP. Furthermore, nimble scheduling can be investigated for surveillance radars. The results of the simulation runs are provided and discussed in Chapter 5.

The last research question is answered by comparing parallel path consistency executing on General-Purpose Graphical Processing Unit (GP-GPU) against serial path consistency executing on a single Central Processing Unit (CPU) core. Section 4.3 provides a framework within which the serial and parallel IA network consistency can be compared to each other in terms of processing time required. This framework is compared to published research that considers how IA algorithms could be utilised on parallel processing architectures. Using this simulation environment, multiple runs were executed for each of the varied independent variables: number of intervals in the network, expressiveness of the IA relationships and number of unknown relationships. These simulations were performed to compare GP-GPU parallel processing of IA path consistency to that of a serial algorithm

[9]The simulation environments was also used to generate the conference paper [1] and journal paper.

utilising only a single CPU core. The results of these simulation runs are captured and discussed in Chapter 6.

Chapter 5

Multistatic Radar Scheduling Investigation

5.1 Introduction

This chapter captures the details of the research conducted to answer the first two research questions. Answers to the first two research questions (Section 1.5) made the most impact on supporting the hypothesis statement. The methodology (Section 1.6) and methods (Section 4.2) were followed to ensure the research provided valid results. The rationale for the research decisions made was discussed in Section 4.2.2. This chapter[1] forms the basis for a research paper, as mentioned in Section A.3, which was published by the Elsevier Information Fusion journal [2].

Section 5.2 provides the details of the algorithms developed to compare Interval Algebra (IA) to Greedy Randomised Adaptive Search Procedure (GRASP) Mechanically Steered Multistatic Surveillance Radar Network (MSMSRN) scheduling and test it against a nimble scheduling problem. More details can also be found in Appendix B. This begins with Section 5.2.1, which contains an overview of the initial IA scheduling algorithm [1]. This is followed by a treatment of the most basic real-world constraint of uncertainties in the angular order of target (Section 3.2.3) in Section 5.2.2. These uncertainties in the angular order of targets result from the fact that no sensors can have an infinitesimally small measurement accuracy. Next, Sections 5.2.3, 5.2.4 and 5.2.5 capture the improvements

[1] Along with the supporting Sections 3.2, 3.3, 3.4 and 4.2

and optimisations that were made to the initial IA scheduler. Two of these optimisations resulted in novel contributions. Finally, Section 5.2.6 looks at how IA can be used to solve the nimble scheduling problem, a potentially important real-world constraint.

Section 5.3 the captures the results and how they were obtained by following the methods in Section 4.2. This begins with Section 5.3.1, which contains the details of the simulations executed to answer the research questions using the algorithms defined in the previous section. Next, Section 5.3.2 gives the results of the comparison of IA to GRASP. Finally, Section 5.3.3 gives the results of the nimble scheduling investigation.

In Section 5.4, an academic discussion around the meaning and significance of the results is given. Section 5.4.1 is divided into four subsections, and discusses how well IA compares to GRASP. The first three subsections look at how the comparison is impacted when varying one independent variable: number of targets in the scene, radar beam widths and number of radars in the multistatic network. The final subsection considers the impact of the novel Reduced Point Algebra (RPA) format that was developed in Section 5.2.5. Section 5.4.2 then discusses the nimble scheduling results and the impact thereof on surveillance sensor networks in general. Next, Section 5.4.3 looks at the importance of sensor architecture and how other architectures can be accommodated with IA. Finally, Section 5.4.4 considers the information gain that is achieved by using IA.

This investigation is concluded by Section 5.5, where all the most important results are highlighted and future work is identified. The concluding section reiterates the research that was conducted so that it can stand on its own.

5.2 Calculation

Previous work presented by the author [1] compared IA to a heuristic algorithm that was devised to solve the Mechanically Steered Multistatic Surveillance Radar Network scheduling (MS) problem. This comparison did not demonstrate the effectiveness of IA against established techniques. Thus, the IA MS (I-MS) algorithm was compared to a conventional algorithm to evaluate the suitability of IA as a solution to the MS problem. GRASP, a mathematical programming technique [33], was selected for the comparisons.

The GRASP MS (G-MS) algorithm, unlike the I-MS algorithm, performs a global search by repeatedly performing multiple local searches. In each iteration of GRASP, a solution is generated and then a local search is performed, which continually adds and removes targets in order to find a better solution. In contrast, the I-MS algorithm generates a single locally optimal solution and so is expanded by using it as the local search for the GRASP algorithm to draw fair comparisons. This form of the I-MS algorithm was referred to as the Iterative IA MS (II-MS) algorithm.

The comparison of the II-MS algorithm to the G-MS algorithm was then evaluated from two different perspectives to ensure that a fair comparison was performed. The first perspective compared the algorithms in terms of the computational gain they offer. The algorithms were optimised such that they performed equally well against the tested metric of mean multistatic measurements per scan. Algorithms that require less computation time provide a better solution to the scheduling problem.

Conversely, the second perspective compared the algorithms in terms of the performance gain they offer. The algorithms were optimised such that they perform in the same amount of processing time. Algorithms that make more multistatic measurements of targets on average per scan of the radars provide better solutions. While the II-MS algorithm compared well to the G-MS algorithm when only considering the performance of the algorithms, it also required three orders of magnitude greater processing time. This is due to the fact that IA provides a much richer set of constraints than that which was being used by the MSMSRN scheduler. To address this shortcoming the Iterative Reduced Point Algebra MS (IRP-MS) algorithm was developed that can be regarded as an equivalent to the G-MS algorithm in both respects, and is discussed in Section 5.2.5.

5.2.1 Basic IA scheduling

Algorithm 12 IASchedulerInit ($\boldsymbol{A}$: target order for each radar)

```
create IA network N = ∅
for all radars r do
  create handled set H = ∅
  for all bins b ∈ A_r in sequence do
    for all targets t ∈ A_{r,b} in sequence do
      create interval I_t
      add row and column t to N for I_t
      for all intervals I_n in H do
        if n ∈ A_{r,b} then
          PC (N, [n, t], {all})
        else
          PC (N, [n, t], {before})
        end if
      end for
      H = {H, I_n}
    end for
  end for
end for
return N
```

As per Section 4.2, the IA scheduler receives an ordered lists of targets T_r from each of the radars r forming a multistatic radar network, where $T_r(n)$ is the location of the target n in the list. The IA scheduler then creates an interval $I_{r,n}$ per target for each of the radars, which represents a task for the radar to track the target during the scan. Next, the IA network is initialised by adding each of the created intervals and only populating the IA relationships for tasks of the same radar.

This is done by consulting the two order lists and generating suitable IA relations. If the relationship is unambiguous then it will only contain either the IA operator 'before' or 'after'. The order list is consulted for each target and the relationship for a corresponding interval is set as follows:

$$I_{r,n}\{\text{before}\}I_{r,m} \mid \forall\,(r, n \neq m, T_r(n) \prec T_r(m)) \tag{5.1}$$

All other relationships are kept at their initial value of {all}.

Next, the scheduler must decide which targets will be measured by both radars simultaneously, thus a multistatic measurement can be made. For the IA scheduler, this

requires setting the relationship between the intervals for the selected target n of all the radars r and s to contain only 'equal'.

$$I_{r,n}\{\text{equal}\}I_{s,n} \mid \forall\, r \neq s \tag{5.2}$$

Each time a target is scheduled for a multistatic measurement, the IA scheduler first makes a backup of the current IA network. Then the scheduler randomly selects a target with the highest priority and sets the corresponding intervals to the 'equal' relationship. If the network remains consistent the target is added to the list of targets that will be measured simultaneously and the resulting IA network is retained. Alternatively, when an inconsistency is generated, the IA network is reinstated to the backup and the target is discounted from being measured simultaneously.

The final list of targets to measure simultaneously is the list that is generated once each target has been attempted. Targets of a lower priority are only selected once every target of a higher priority has been attempted. If there are two targets with the same priority a random selection is made.

5.2.2 IA scheduling with uncertainties

Algorithm 13 IASchedulerRun ($\boldsymbol{N}$: IA network, $\boldsymbol{A}$: target order for each radar, Q : target priority queue)

```
create schedule S = ∅
for all targets n ∈ Q select n randomly according to priority do
  for all radars r do
    p_r = index of n in A_r
  end for
  set consistency c = true
  for all unique combinations a and b in index of p do
    c = c ∧ PC (N, [p_a, p_b], {equal})
  end for
  if c = true then
    S = {S, n}
  end if
end for
return schedule S
```

IA can easily manage the case where the azimuth angle of the targets cannot be determined. As per Section 4.2, the scheduler now receives an order list of azimuth sectors

$\boldsymbol{A}_r$ for each radar r. Each azimuth sector $\boldsymbol{A}_{r,k}$ can contain one or more targets, where sectors containing zero targets need not be reported.

The same IA operators are used in the relationships of targets from different azimuth sectors, which will again match their ordering in the list as follows:

$$I_{r,n}\{\text{before}\}I_{r,m} \mid \forall\,(r, n \neq m, n \in \boldsymbol{A}_{r,k}, m \in \boldsymbol{A}_{r,l}, k < l) \tag{5.3}$$

where n is the first target number, m is the second target number, k is the sector number corresponding to target n and l is the sector number corresponding to target m.

However, the relationships between targets in the same azimuth sector are not updated and remain {all}, as per

$$I_{r,n}\{\text{all}\}I_{r,m} \mid \forall\,(r, n \neq m, \{n, m\} \in \boldsymbol{A}_{r,k}) \tag{5.4}$$

Targets are selected as per the case where the ordering is known exactly and the list of targets to visit is generated again by using 'equal' relationships. Scheduling with uncertainties then progresses exactly as per basic IA scheduling.

The IA scheduling algorithm, which can handle the uncertainties in the target sequence, is presented as Algorithm 13. This algorithm requires using Algorithm 12 to initialise the IA network once per scan.

5.2.3 Growing the IA network

Algorithm 14 RpaNetGrow ($\boldsymbol{n}$: Reduced PA network, $\boldsymbol{A}$: radars target order, Q : target priority queue)

```
for all t ∈ Q select t randomly according to priority do
  if t ∉ n_k | ∀ k then
    n = RpaNetAdd (n, A, t)
  end if
end for
return n
```

Here the way IA is used to schedule is reconsidered by revisiting the way intervals are added to the IA network. The I-MS algorithm first adds all intervals to the IA network before performing any constraint processing. Thus each iteration of the IA path

consistency algorithm executes on a maximum-sized constraint matrix.

A better method to perform the IA computations would be to only add the intervals as they are required, thereby growing the IA network as processing continues. This is a simple contribution, which has parallels with the work by van Beek and Manchak [148], as this work focuses on adding intervals in the order of maximum impact to least impact. This highlights the importance of adding intervals to the IA network in the best possible order.

In the IA Growing-network MS (IG-MS) algorithm, multistatic measurement intervals from each radar are added for one target at a time to the IA network. In this way, the IA network only grows as more multistatic measurements of the targets are scheduled. Algorithm 14 captures the essence of the growing technique by only adding intervals as they are required.

Algorithm 15 RpaNetAdd($\boldsymbol{n}$: Reduced PA network, $\boldsymbol{A}$: radars target order, t target number)

d = length of vector $\boldsymbol{n}$
for all $s = 1 \rightarrow d$ **do**
  for all $m \in \boldsymbol{n}_s$ **do**
    $R = \{\text{all}\}$
    for all radars r **do**
      $R_r = \text{RpaGenRel}\ (\boldsymbol{A}_r, t, m)$
      $R = R \cap R_r$
    end for
    if $R = \emptyset$ **then**
      return $\boldsymbol{n}$
    else if $R = \{\text{before}\}$ **then**
      $\boldsymbol{n}_{s-1} = \{\boldsymbol{n}_{s-1}, t\}$
      return $\boldsymbol{n}$
    else if $R = \{\text{before, after}\}$ **then**
      $\boldsymbol{n}_s = \{\boldsymbol{n}_s, t\}$
      return $\boldsymbol{n}$
    end if
  end for
end for
$\boldsymbol{n}_{d+1} = \{t\}$
return $\boldsymbol{n}$

5.2.4 Constraint ordering

Ordering heuristics could then be considered from the work of van Beek and Manchak [148] to be applied to the IG-MS algorithm. Constraint ordering is described in Section 3.3.3.

However, the ordering heuristics cannot be used directly for the IA scheduling algorithm discussed. This is because the constraints with the 'equals' relation have the largest impact and these are determined during the operation of the algorithm. These represent the multistatic measurements, which must be selected by the scheduling algorithm.

The ordering heuristics did, however, lead to the notion of using a single interval per multistatic tracking task. Doing so removes the need for the 'equals' constraints as there is only one interval per target, which now represents the multistatic measurement task for both radars.

All IA scheduling algorithms up until the IG-MS algorithm used an interval for each target per radar; thus there were also two times more intervals than targets for these algorithms. Using a single interval for a multistatic measurement task leads to a further improvement in performance, as the IA constraint matrix size is reduced by three quarters.

Scheduling with IA allows either an interval per track or an interval per target to be used. The single interval is then chosen to represent a group of tracks that are believed to belong to the same target. Track-to-track association is performed in the research using the auction algorithm, which is typically applied to detection-to-track association.

While targets move the interval stays the same but the relationship of the interval to others changes. This requires rebuilding the IA network on each scan or only when the targets have sufficiently moved on from their current location to require such a change. For the research the network was rebuilt on each scan, but applying a more dynamic approach could potentially alleviate much processing requirements, if an efficient way for checking the scope of target movements has required a change in the network structure. Such a technique was deemed outside the scope of the current research. I will add this to the list of potential future work.

Using a single interval per target requires performing the intersect of the constraints from the radar in the multistatic cluster. If the set is not null then the interval can be

added to the IA network. The resulting algorithm was called the IA Simplified-Constraints MS (IS-MS) algorithm.

5.2.5 Reduced PA networks

After the preceding optimisations, only two IA operators were used in interval relations, namely the 'before' and 'after' IA operator. This is a subset of Point Algebra (PA) which uses only 'before', 'after' and 'equals'. This means that the relationships between the interval for a single radar would either be {before}, {after} or {before, after}. The intersect of the relationships from both radars would then be similarly constrained.

Using Matlab® code analyser revealed that, in all cases, this PA network remained consistent. This was as long as the intersect of the constraints from either radar was not null. Looking at the PA relational matrix for the resulting PA network showed that each row of the PA network matrix, for MSMSRN scheduling, is merely a permutation of every other row if set inversion is applied. That is, one row captured all the information about the relationships for the entire matrix.

Thus the entire PA network matrix could be captured more efficiently as a vector of sets containing intervals. Intervals from different elements in the vector, a set, are temporally organised as per their occurrence in the vector. However, intervals found in the set have an ambiguous temporal relationship to each other. In this case, a surrogate interval could be used in the vector to represent this ambiguous grouping of intervals.

The Reduced Point Algebra MSMSRN Scheduling (RP-MS) algorithm therefore uses a vector of sets, containing intervals, to capture and maintain the relationships between all the intervals. The Iterative RP-MS (IRP-MS) Algorithm again uses RP-MS as the local search for GRASP. This leads to savings in memory and processing time, as the Reduced Point Algebra (RPA) form does not require storing or maintaining all the relationships between every interval. This algorithm is a novel contribution which demonstrates that under certain circumstances the relationships in a PA network can better be captured as a vector of sets than a matrix.

Algorithms 14-16 implement the Reduced PA defined as part of this research. All set operations have their usual meaning, while subscripts denote the indexing operation. Sets are in upper case, matrices are in bold upper case while vectors are in bold lower case.

Finally, the target order matrices $\boldsymbol{A}_r$ are the azimuth angle ordering from a radar r, which can incorporate uncertainties in this ordering. Each column represents the known angular sequence and the true order of targets in the same column (on different rows) cannot be discerned. The pseudo code for the Reduced PA scheduler is provided as Algorithm 17.

Algorithm 16 RpaGenRel ($\boldsymbol{A}$: target order, n : target one, m : target two)

```
c_n = column of n in A
c_m = column of m in A
if c_n < c_m then
   R = {before}
else if c_n = c_m then
   R = {before, after}
else
   R = {after}
end if
return R
```

Algorithm 17 ReducedPAScheduler ($\boldsymbol{A}$: target order for each radar, Q : target priority queue)

```
create reduced PA network n = ∅
create schedule S = ∅
n = RpaNetGrow (n, A, Q)
for all targets i ∈ n in sequence do
   S = {S, n}
end for
return schedule S
```

5.2.6 Nimble IA scheduling

As discussed in Section 3.2.4, in certain circumstances it is not sufficient to perform scheduling only when the scan commences. In these cases, the scheduling is performed at scan boundaries as well as during the time the multistatic measurement is being made. The Nimble IA Scheduler solves Equation 3.4 for each target before building the list of targets that may contain uncertainties. In doing so, it assumes that the target will be visited by moving the radar with the largest azimuth delta at the fastest positioner rate. Furthermore, the assumption is made that no other target will be visited. The target priority is also updated using Equation 3.5.

The selection of targets for multistatic measurement is then performed as per all the other IA scheduling algorithms, except that the target priority is the area of the 2σ ellipse

defined by the predicted tracking covariance. Thereafter, the low-level scheduler is tasked with ensuring that the positioners of the two radars are moved so that the beams of the radars intercept on the target. The low-level scheduler also solves Equation 3.4 but instead of assuming no other targets are visited, it must compute the times as determined for the entire sequence of targets. When reaching interception points, targets that can no longer be measured in the multistatic mode are not considered by the nimble scheduler. These targets are those that lie in the opposite direction from the radar beam scan direction and have either been measured already using the multistatic mode or were measured using only monostatic measurements. They can no longer be measured using any mode during the current scan and will have to be revisited on the next scan.

Each time the scheduler is invoked at the boundaries of the scan or at target intercepts, an IA network must be built from the remaining targets that can be visited. As the target relationships change during the time it takes the radar beams to reach an interception point, the IA network from previous iterations must be discarded as it contains invalid information. In the current form IA can only further constrain relationships; however, future work could address this by allowing IA also to permit more ambiguity in relationships. Nevertheless, generating the IA network for subsequent rounds gradually requires less computation so that total processing time increases logarithmically. The pseudo code for the Nimble IA Scheduler is presented as Algorithm 18.

Algorithm 18 NimbleIAScheduler ($\hat{\mathbf{x}}$: target state estimate vector for all targets, t_0 : current time)

for all targets f where $\theta_f(t_0) \succ \theta_r(t_0)$ **do**
  for all radars r **do**
    $\boldsymbol{\delta}_{f,r} = \theta_f(t_0) - \theta_r(t_0)$
  end for
  find radar r with maximum $|\boldsymbol{\delta}_{f,r}|$
  solve Equation 3.4 for t using r
  $\bar{\mathbf{x}}_f = \mathbf{A}_f^t \cdot \hat{\mathbf{x}}_f(t_0)$
  compute $\bar{\mathbf{P}}_f(t)$ using Equation 3.5
  add target f and priority $\bar{\mathbf{P}}_f(t)$ to priority queue Q
  add target f to $\boldsymbol{A}_r$ bins as per $\boldsymbol{\delta}_{f,r}$
end for
$\mathbf{N} = \text{IAScheduleInit}(\boldsymbol{A})$
$S = \text{IAScheduleRun}(\mathbf{N}, \boldsymbol{A}, Q)$
send target S_f with minimum $|\boldsymbol{\delta}_{f,r}|$ to radar schedulers

Interested readers can refer to Appendix B for the pseudo code all the scheduling algorithms implemented during this research. Furthermore, the Matlab code for all algorithms can be found on the accompanying disc, the contents of which are outlined in Appendix D.

5.3 Results

5.3.1 Simulation set-up

For each of the comparison simulations[2] performed a 40×40 km area was selected, while a 10×10 km area was selected for the nimble simulations[3] to reduce simulation times. One radar is situated at coordinate $(0, 0)$ km or the bottom left corner of the area. The other radar is situated at the top left corner of the area (coordinate $(0, 40)$ km or $(0, 10)$ km). Both radars are assumed to have a maximum detection range greater than 57 km. The maximum slew rate of the positioners of the radar was selected to be $\Pi/2$ radians per second, or a 90° rotation every second. The radars always scan over the area in opposing directions; therefore, if one radar is scanning clockwise the other will scan anti-clockwise.

A number of targets are then generated with random starting points and random motion profiles. The relative positions of the targets in relation to the two radars represent a target geometry. During the scan, the radars are not allowed to change scan direction until they have reached the boundaries of the surveyed area. With these constraints and any given target geometry, a different upper limit of multistatic measurements can be generated. The target geometry will change slowly as the targets move during the simulation. Target motion for the nimble simulations is modelled according to the Singer random-walk acceleration model [156]. The details of this model are given in Appendix C.1.

Three Monte-Carlo simulation batches were performed to analyse the performance of IA scheduling. The first two batches compared IA to GRASP, where one batch varied the number of targets, and the other varied the radar beam width. For these batches the simulation time, for each random run, was kept low at 1 000 seconds, which is not necessarily equivalent to execution time. The final batch of simulations determined the nimbleness of the IA scheduler when used in highly dynamic scenarios. Simulation time had to be limited to 60 seconds due to the computational complexity of the nimble IA scheduling simulation runs.

In all cases, the simulation time is realistic when considering the radar positioner slew

[2]Refer to Appendix D or https://code.google.com/p/multiple-radar-fusion-simulation-simple/
[3]Refer to Appendix D or https://code.google.com/p/multiple-radar-fusion-simulation-realistic/

rates and target motion, but the simulation does not take into account the processing time of the algorithms. During a simulation run, the target geometries vary slowly due to the target movements. The initial target geometry is generated randomly and varies greatly between two different random seeds. Therefore using more simulation runs tests a greater diversity of the target geometries than using fewer simulations that run for a longer simulation time.

The random seed was reinitialised every time the MSMSRN scheduling algorithm changed. This was done for all the batches of simulations to ensure that the different scheduling algorithms operated under exactly the same conditions. A global counter was used as the random seed and was only incremented after testing all scheduling algorithms. A 100 different global counter values were used for each batch of simulations. The random seed determines the initial placement and paths of the targets, as well as the priority change of each target within the priority queue when used.

5.3.2 IA comparison to GRASP

Varying the number of targets

Table 5.1: Results comparing IA to GRASP by varying the number of targets in the scenario. Mean (μ) and standard error (ϵ) of scan duration (S), targets measured (M) and total execution time (E) as the number of targets is varied for IA (I) and GRASP (G). This table captures the results of 800 Monte-Carlo simulation runs, where the number of targets in the scenario were varied between 10, 20, 40 and 80 targets. The beam width of the radars was kept at a constant value of 1°. The G-MS algorithm is abbreviated to G while the IRP-MS algorithm is abbreviated to I.

	Targets							
	10		20		40		80	
	μ	ϵ	μ	ϵ	μ	ϵ	μ	ϵ
S_G	5.9	$3 \cdot 10^{-2}$	6.0	$2 \cdot 10^{-2}$	6.1	$2 \cdot 10^{-2}$	6.2	$1 \cdot 10^{-2}$
S_I	5.9	$3 \cdot 10^{-2}$	6.0	$2 \cdot 10^{-2}$	6.1	$1 \cdot 10^{-2}$	6.2	$1 \cdot 10^{-2}$
M_G	3.7	$7 \cdot 10^{-2}$	5.5	$7 \cdot 10^{-2}$	8.3	$7 \cdot 10^{-2}$	13	$8 \cdot 10^{-2}$
M_I	3.7	$7 \cdot 10^{-2}$	5.5	$7 \cdot 10^{-2}$	8.3	$7 \cdot 10^{-2}$	13	$8 \cdot 10^{-2}$
E_G	5.2	$4 \cdot 10^{-2}$	18	$8 \cdot 10^{-2}$	66	$20 \cdot 10^{-2}$	260	$60 \cdot 10^{-2}$
E_I	4.9	$3 \cdot 10^{-2}$	16	$7 \cdot 10^{-2}$	60	$20 \cdot 10^{-2}$	240	$60 \cdot 10^{-2}$

Varying the beam width of the radars

Table 5.2: Results comparing IA to GRASP by varying the radar beam width of both radars simultaneously. Mean (μ) and standard error (ϵ) of scan duration (S), targets measured (M) and total execution time (E) as the beam width is varied for IA (I) and GRASP (G). This table gives the results of 1 200 Monte-Carlo simulation runs, where the beam width of the radars was varied between 10^{-6°, 1°, 2°, 4° and 8°. The number of targets was kept at a constant value of 20 targets. The G-MS algorithm is abbreviated to G while the IRP-MS algorithm is abbreviated to I.

	Beam Width °						
	μ						ϵ
	10^{-6}	0.5	1	2	4	8	
S_G	6.0	6.0	6.0	6.1	6.2	6.5	$1.8 \cdot 10^{-2}$
S_I	6.0	6.0	6.0	6.1	6.2	6.5	$1.8 \cdot 10^{-2}$
M_G	5.3	5.4	5.6	5.8	6.4	7.3	$7.6 \cdot 10^{-2}$
M_I	5.3	5.4	5.6	5.8	6.4	7.3	$7.7 \cdot 10^{-2}$
E_G	18	17	18	17	17	16	$8.1 \cdot 10^{-2}$
E_I	16	16	16	16	16	15	$7.0 \cdot 10^{-2}$

5.3.3 Nimble scheduling with IA

Note that for both Figure 5.1 and 5.2 better performance could have been gained for the Naive, Intercept and Nimble IA Scheduler by allowing the GRASP-like outer loop of the algorithms to run more iterations.

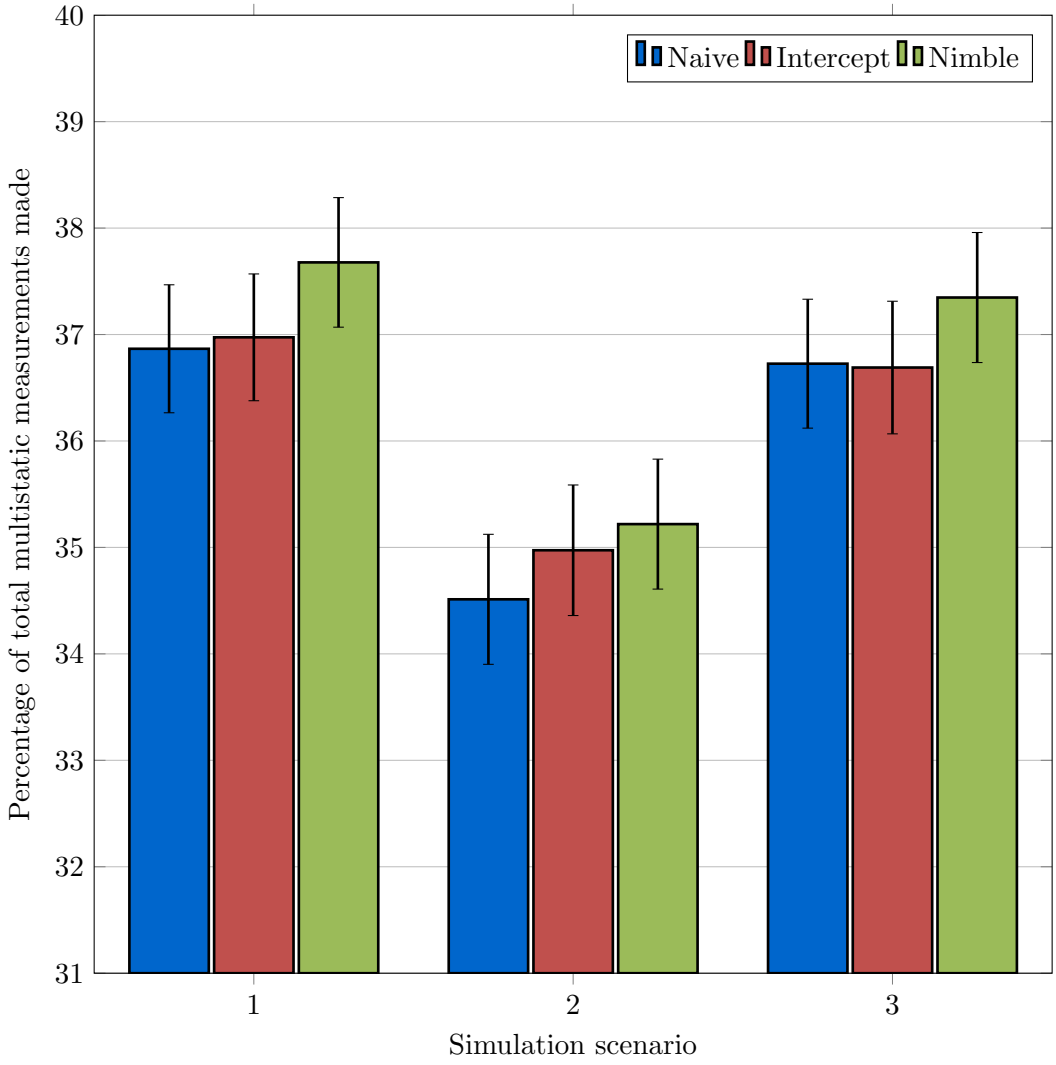


Figure 5.1: Effect of target initial velocity and acceleration on number of measurements made using IA scheduling. Total measurements possible is assumed to be two multistatic measurements per target, where 100% is only possible with a perfect AESA radar (i.e. off-bore-sight scanning does not require longer dwells) and 40% is the theoretical maximum for an MSMSRN scheduler. This is assuming the measurement dwell is 17 ms on average and a measurement is made as the target enters both beams and as it leaves the beams. Average number of multistatic measurements made for 900 nimble IA scheduling Monte-Carlo simulation runs. Three different simulation scenarios were employed to investigate the effect of highly dynamic targets. For each, the acceleration variance σ_a of the Singer random-walk motion model and initial target velocity, a Gaussian random value with mean μ_v, were set as follows: $\sigma_a = 2\ \mathrm{m} \cdot \mathrm{s}^{-2}$ and $\mu_v = 25\ \mathrm{m} \cdot \mathrm{s}^{-1}$, $\sigma_a = 2\ \mathrm{m} \cdot \mathrm{s}^{-2}$ and $\mu_v = 50\ \mathrm{m} \cdot \mathrm{s}^{-1}$ or $\sigma_a = 20\ \mathrm{m} \cdot \mathrm{s}^{-2}$ and $\mu_v = 25\ \mathrm{m} \cdot \mathrm{s}^{-1}$. Three MSMSRN scheduling modes were tested: naive, once per scan using the current target geometry, thus ignoring highly dynamic targets; intercept, once per scan simply predicting the location of the target using the fastest beam intercept time; or nimble, at each intercept point where all target locations are predicted using the fastest beam intercept time as per Section 5.2.6.

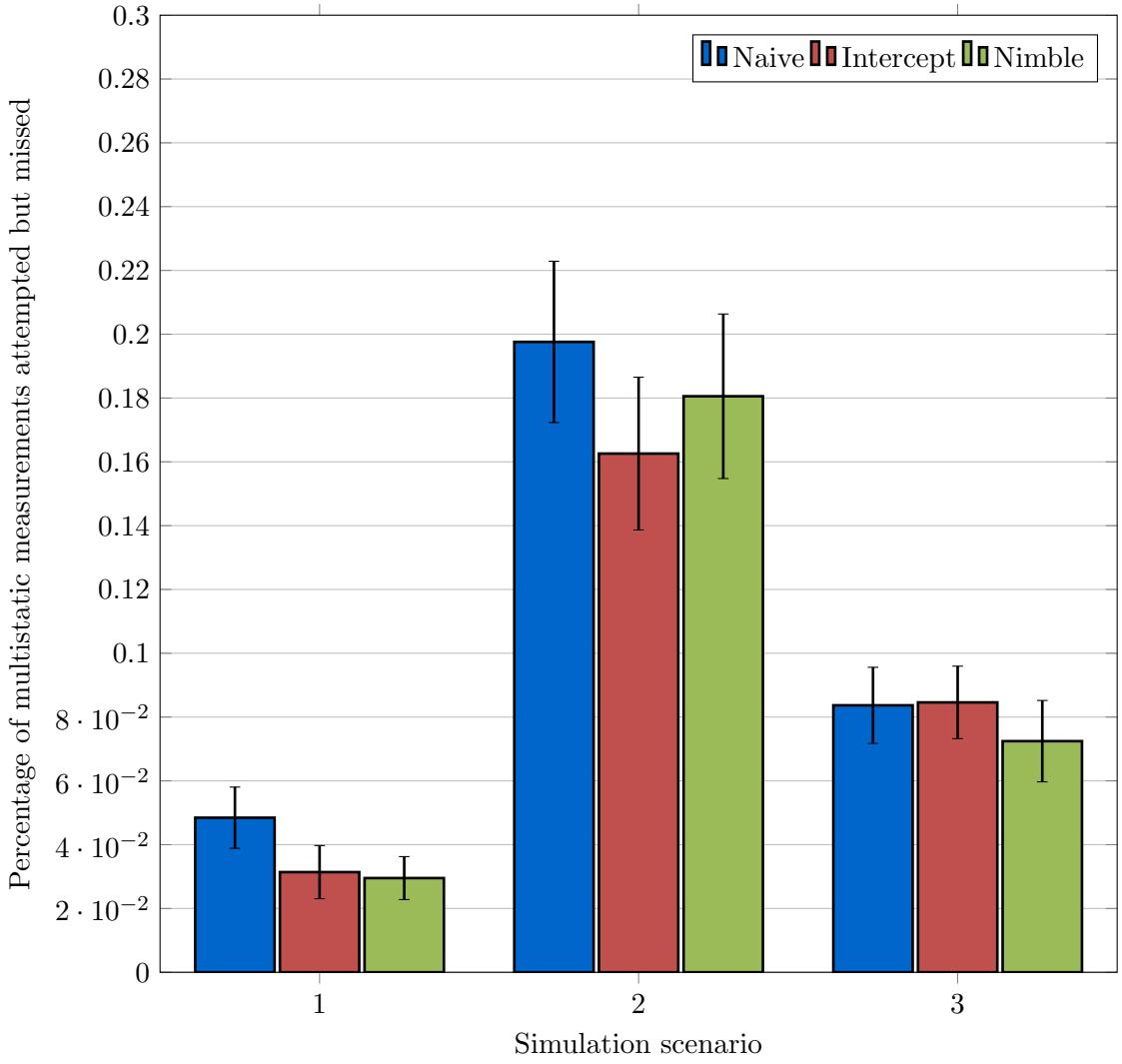


Figure 5.2: Effect of target initial velocity and acceleration on number of measurements attempted but missed using IA scheduling. Average number of multistatic measurements attempted but missed for 900 nimble IA scheduling Monte-Carlo simulation runs. Total measurements possible is assumed to be two multistatic measurements per target, where 100% is only possible with a perfect AESA radar (i.e. off-bore-sight scanning does not require longer dwells) and 40% is the theoretical maximum for an MSMSRN scheduler. This is assuming the measurement dwell is 17 ms on average and a measurement is made as the target enters both beams and as it leaves the beams. Three different simulation scenarios were employed to investigate the effect of highly dynamic targets. For each, the acceleration variance σ_a of the Singer random-walk motion model and initial target velocity, a Gaussian random value with mean μ_v, were set as follows: $\sigma_a = 2\ \mathrm{m \cdot s^{-2}}$ and $\mu_v = 25\ \mathrm{m \cdot s^{-1}}$, $\sigma_a = 2\ \mathrm{m \cdot s^{-2}}$ and $\mu_v = 50\ \mathrm{m \cdot s^{-1}}$ or $\sigma_a = 20\ \mathrm{m \cdot s^{-2}}$ and $\mu_v = 25\ \mathrm{m \cdot s^{-1}}$. Three MSMSRN scheduling modes were tested: naive, once per scan using the current target geometry, thus ignoring highly dynamic targets; intercept, once per scan simply predicting the location of the target using the fastest beam intercept time; or nimble, at each intercept point where all target locations are predicted using the fastest beam intercept time as per Section 5.2.6.

5.4 Discussion

5.4.1 IA comparison to GRASP

Running Monte-Carlo simulations comparing the IRP-MS algorithm to the G-MS algorithm revealed that the two algorithms not only compute in a similar amount of time but also perform equally in terms of multistatic measurements made. In the case of the IRP-MS algorithm, the RP-MS algorithm would be run once per target in the scene. Thus, in these respects the RP-MS algorithm is equivalent to the GRASP construction algorithm used by the G-MS algorithm.

However, IA can cater for a richer set of temporal constraints without modifying the underlying algorithm. The GRASP construction algorithm would need to be redesigned to handle the new set of constraints and the modifications would become increasingly complicated matching the complexity of the constraints. IA would be able to handle the new constraints without modification, as long as these constraints can be expressed using the temporal relationships defined by Allen [4].

Effect of increasing the number of targets

As the targets in the scene were increased, both the IRP-MS algorithm and the G-MS algorithm could make more multistatic measurements per scan. However, the increased number of multistatic measurements came at the expense of more computations. Table 5.3 shows that the number of multistatic measurements made decreases as a percentage of the number of targets present in the scene. Furthermore, processing time required per target in the scenario also increases as per Table 5.4.

Table 5.3: Scheduling performance versus number of targets

Targets	Mean multistatic measurements	Percentage
10	3.7148	37.148
20	5.5500	27.750
40	8.3025	20.756
80	12.519	15.649

The reason for these trends is due to the fact that as the number of targets increases linearly, the number of different choices of sequences of targets that can be measured in

Table 5.4: Scheduling processing time versus number of targets

Targets	Processing time	Time per target
10	5.0505	0.5051
20	16.919	0.8460
40	63.406	1.5852
80	251.92	3.1490

the multistatic mode increases much quicker. The positions of the targets relative to the two radars determines a unique geometry. The number of unique geometries of targets that can be generated was the key factor and the longer sequences became less likely when increasing the number of targets. Thus, on average, the total length of each of the sequences of targets that solved the scheduling problem increased slowly. The statistical distribution of the geometry of the targets within the area under surveillance affected both of these factors. Each of the simulation runs within a batch of simulations, using the same scheduling algorithm, tested a different geometry.

Effect of increasing the radar beam width

When the radar beam width increased this did not cause much change in the required computation for either algorithm. This was due to the fact that the computation time for both algorithms is not dominated by the number of uncertainties in the radar target sequences. Both algorithms efficiently handled the uncertain order present in the azimuth ordering of targets.

G-MS employed a very simple construction algorithm, which was also used to generate mutated solutions. The mutated solutions were generated during the local search from the first solution constructed. In both these algorithms, a single target was added at a time and simple checks ensured that the target was only added if the sequence remained consistent. Thus, this simple searching nature keeps the computational requirements low.

RPA has some benefits of normal IA, but with fewer processing requirements. This is because the IA relationship matrix was collapsed from a matrix to a single row (or column) vector of interval sets. Uncertainties in the list of targets from either radar changed the input relationships whose intersection generated the IA relationships contained in the vector. The same computation was performed regardless of the content of the relationships for RPA. Thus, more uncertain orderings did not require more processing time.

The slight decrease in processing time required can be attributed to the fact that as the number of uncertainties in the ordering increased, the average length of the solutions generated also increased. This means that, on average, more targets can be added to the list of simultaneous measurements, and thus, on average, fewer mutated solutions need to be generated in the search phase of GRASP.

Effect of increasing the number of radars

No simulations were performed to test the effect of increasing the number of radars; however, it is easy to extrapolate what will occur from the results given and the workings of these algorithms.

G-MS will require a redesigned construction and mutation algorithm to handle the dimension increase in the number of radars used. This will lead to more complicated checks that need to be performed and will lead to a more complicated solution. The processing time for the G-MS algorithm should increase linearly, as is the case when the number of targets increases. This will be true as long as a sufficiently simple construction and mutation algorithm can be generated for GRASP.

RPA and any other form of IA will easily handle the addition of multiple radars. In the case of RPA, this only requires intersecting more relationships for the multistatic target measurement intervals. Adding more sensors should also cause computation time of the IRP-MS algorithm to increase linearly. The IRP-MS makes use of a vector of interval sets to store the relationships between intervals, and the size of this vector bounds the computation. Each time a radar is added, one more intersect per target addition will be required, thus the processing time should double. Thus, a linear increase in computation is expected for the IRP-MS algorithm as well.

IA network compression using RPA

The two novel contributions were key to allowing the IRP-MS algorithm to perform as well as the G-MS algorithm. Without these modifications IA algorithms required more processing than GRASP for the same sized MSMSRN scheduling problems. It is important to note, however, that these modifications could be used in any IA algorithm. Thus they are not exclusive to the MSMSRN scheduling problem.

The technique of growing the IA matrix was similar to the work of van Beek and Manchak [148] as the performance improvement results from the order that intervals are added. Unlike the previous work, however, the key was that intervals should only be added to the IA network when they are necessary to generate a solution. Adding the interval prior to this time leads to wasted computations, which, even though they are initially small, adds up over multiple iterations of the IA path consistency algorithm.

RPA became evident after considering the work of van Beek and Manchak [148]. It also builds on the initial work of Allen [4], in which he noted that a set of intervals could be summarised using a surrogate interval. However, details of when using a surrogate interval would be beneficial were not clear. In the RPA algorithm, it is clear that surrogate intervals could be used to represent a set of IA intervals that have simple relationships.

When the IA relationships amongst a set of IA intervals were purely of an ordering nature (containing 'before', 'after', 'meets' and 'met') then the RPA form would be applicable. For more complex IA relationships, especially those that consider various types of overlapping intervals, the RPA format is insufficient and the matrix format must be used.

The relationships of the set of intervals with simple constraints can then be captured in the more computationally efficient RPA format. Alternatively, a surrogate interval could be used in the RPA relationship vector to represent portions of the network that could only be represented using an IA relationship matrix. Which of these two approaches will work best depends on the IA network, and thus both techniques may be usable to solve future problems more efficiently. Similarly, a surrogate PA network can also be used to reduce the size and computations of a large IA network and vice versa. This holds true for any two constraint programming languages where parts of a network only require the simpler constraint language.

Richer temporal constraints

So far richer temporal constraints have not been required to build schedules for the problem at hand. However, there are scheduling scenarios where these constraints could be used and then having a tool that incorporates a mechanism to allow for this would be beneficial. One example would be to further optimise the dwell times of the radars. Currently, these are seen as one interval starting from the first reception until the target echo for both the

monostatic and bistatic measurements have been received. Alternatively, this task could be further subdivided into three tasks for each radar: transmit energy, receive monostatic and receive bistatic. The radars could then be allowed to transmit new waveforms while the other radar is receiving the bistatic or monostatic echo. This scenario would require the overlapping constraints of IA to solve correctly and would optimise the dwell time of the radars on a particular target.

5.4.2 Nimble IA scheduling

None of the nimble scheduling results obtained are statistically significant due to the large variance of the two tested parameters: multistatic measurements made and missed. On average, however, the nimble scheduling strategy is able to make the most multistatic measurements, approximately ten more than non-predictive scheduling strategy for all simulation modes. This is due to the fact that the nimble scheduler is acting based on an updated recognised surveillance picture and not on a single sample of the target geometry.

Unfortunately, it also requires much more processing than either of the other strategies, as at each interception point all target locations are predicted for targets that can still be visited. The updated recognised surveillance picture is then fed to the nimble scheduling algorithm to recompute. Thus it appears that this simple intercept prediction strategy offers the best alternative. It performs as well as the naive/non-predictive scheduling strategy for both slow-moving targets and those with high accelerations, but performs better for fast-moving targets.

In terms of missed multistatic measurements, both of the predictive scheduling strategies provided a small improvement. However, once again, the results are neither statistically significant nor give a large improvement, and thus do not warrant the extra processing.

5.4.3 Multisensor manager architecture

For this research a hybrid architecture was employed, where the multisensor manager is centralised and the radar managers are decentralised. However, there are three ways that this architecture could be decentralised.

One approach would be to simply replicate the fusion centre and multisensor manager

at each radar. In this scheme, each radar would supply the tracks that occur in a region of overlap to other radars. The tracks would be associated, fused and a scheduling decision would be made. Each radar potentially arrives at a different locally optimal solution. Then to reach consensus, the radars share their chosen schedule with each other. The final choice of targets to schedule is the sequence that maximises Equation 3.1. This approach is equivalent to running iterations of GRASP in parallel using IA as the construction algorithm.

Another approach would be to divide up the sensed area into zones. Again the information fusion system and multisensor manager are replicated at each radar. However, the radars now only share tracks with other radars managing the zone in which the track lies. A radar with a decentralised multisensor manager then solves the IA constraints for their zones. Each partial solution is then fed to the other radars collaborating in the multistatic network. The final choice of targets to schedule is the concatenation of each partial solution in the correct sequence. This approach is only practical when the search volume can be divided up into zones that each radar scans in exactly the same sequence.

Finally, it is also possible to solve the scheduling with a Decentralised Constraint Satisfaction Problem (DCSP) [48]. As per the previous approach, the sensed area would still be divided up into zones. Each radar would be responsible for a zone and would need to receive tracks from the other radars that fall in the zone. Each radar then associates and fuses tracks for the zone it controls. These are then entered into the DCSP, where the radars collaborate to solve the CSP as per our usage of IA in this research. For this approach the choice of zones becomes completely arbitrary as a solution will be found taking all constraints into account.

5.4.4 Information gain

It is important for a multisensor manager to improve the measures of effectiveness of an information fusion system [157]. Performing multistatic measurements for radars achieves this in three ways, and the Nimble IA scheduler offers the best improvement.

Firstly, the amount of information flowing into the fusion system is increased. This is a direct consequence of the radars being able to make both bistatic and monostatic measurements. Thus it is desirable to increase the number of multistatic measurements

that can be made and reduce the ones that are missed. This is indeed the case for the Nimble IA Scheduler and also the simple prediction scheduling. It is important to note that given our comparison of IA to GRASP, similar results can be expected for GRASP.

Secondly, the quality of tracks formed for targets gaining multistatic measurement should improve due to the increased information gain from the multistatic measurements. Even when a multistatic measurement cannot be made due to partial occlusion, the target will still be measured. More accurate tracks should provide better information to all processes.

Finally, robustness of the information fusion system is improved. It is far less likely that when a target is occluded for one radar that it will be occluded for all radars. Therefore it is more likely that the target will be measured regardless.

5.5 Conclusion

In this chapter, IA was compared to GRASP for scheduling MSMSRN. This was done to show that IA is not computationally cumbersome and can provide equivalent performance to that of GRASP. Furthermore, a realistic tracking environment was employed to show how IA could be used to perform nimble scheduling. In the latter environment, three different scheduling modes were tested: naive or no prediction, simple intercept prediction and nimble scheduling.

Three Monte-Carlo simulations for a binary radar system were run to produce results, where targets are randomly placed and move along random trajectories. The first two simulation runs were used to vary the number of targets and the radar beam width in order to compare IA to GRASP. The final simulation runs compared the three scheduling modes used along with IA against slow, fast and rapidly accelerating targets.

While IA scheduling performed as well as GRASP in the binary radar simulations, indications are that it would be easier to adapt this scheduling algorithm to cater for more radars. IA is also more amenable to coping with a richer set of temporal constraints that may be imposed. Since GRASP is based on simpler heuristics, its implementation is more adversely affected by changes in imposed constraints than IA.

IA guarantees that a maximum length solution will be found for the target selections that are made as it performs locally optimal examinations. Instead, conventional optimisation algorithms carry out global searches. Thus, combinations of IA with conventional optimisation algorithms such as IA should lead to even more effective scheduling algorithms.

While the processing time required is within the real time available for both algorithms, adding many more iterations or more targets would make this difficult to sustain. Thus, while performing a global search is desirable, it would not always be possible. Employing parallel processing architectures such as field programmable gate arrays, multi-core central and general-purpose graphical processing units could make the use of a global search technique practically feasible.

In most cases, the IA scheduling algorithm already provides good performance and only a small incremental improvement is possible by utilising a conventional optimisation

algorithm alongside IA. Thus, the performance given by the IA algorithm may be sufficient and it then provides a computationally effective solution to a radar network scheduling problem. The IA algorithm can be used as implemented in Matlab®for around 10 targets without requiring parallel processing.

IA path consistency can be run for 16 384 targets in about 0.5 s of execution time using C[4]. Chapter 6 considers how parallel architectures can be leveraged to reduce the computation time required by IA. Doing so together with running the separate IA iterations in parallel may make it possible to produce better results within realistic scan durations.

Given the present results it would appear that nimble scheduling is not a major problem for a surveillance radar network that is tracking realistic targets that are neither ballistic nor supersonic. However, the simple intercept prediction strategy does provide a minor improvement and may be worth considering if there is spare processing capacity in the system. Nevertheless, both multistatic measurements and nimble scheduling improve the information gain of an information fusion system.

In the case of scheduling a multifunction radar, nimbleness is significantly more important as the platform housing the radar is usually moving quickly and it is likely that ballistic and supersonic targets may be encountered. Thus, the Nimble IA Scheduler could be adapted for use in multifunction radar networks and this would be an interesting topic to pursue in the future.

IA can also be used in decentralised multisensor management solutions and does not impose a specific architecture to the multisensor manager. Furthermore, IA is not only useful in scheduling sensors but can be used to ensure that constraints are satisfied in other information fusion functions.

[4]These are results from my C environment run on an Intel® Core™ i7-2670QM CPU at 2.2 GHz.

Chapter 6

Parallel Processing Architecture Investigation

6.1 Introduction

This chapter captures the details of the research conducted to answer the third and final research question (Section 1.5). Answering this question ensures that the hypothesis, if supported by the evidence, will be valid into the future. This is due to the fact that modern parallel processing architectures are increasingly becoming the primary way of ensuring that Moore's law can be maintained. Ubiquitous parallel processing technologies include Multicore Central Processing Units (MCPUs), General-Purpose Graphical Processing Units (GP-GPUs), Field-Programmable Gate Arrays (FPGAs) and Digital Signal Processors (DSPs). The methodology (Section 1.6) and methods (Section 4.3) were followed to ensure that the research provided valid results. This chapter[1] forms the basis for a research note, as mentioned in Section A.4, which was submitted to the Elsevier Artificial Intelligence journal.

The study in this chapter considers the impact of using path consistency with GP-GPUs in general, whereas prior chapters have focussed on the multistatic scheduling problem. Doing so does not limit the applicability of the solutions given in this chapter to any particular problem solved with IA. However, the algorithms can still be readily applied to any scheduling algorithm that makes use of the basic IA scheduler.

[1] Along with the supporting Sections 3.3 and 4.3

Section 6.2 provides the details of the algorithms developed. An OpenCL form, of Interval Algebra (IA) path consistency, allows modern parallel processing architectures to be compared to a single Central Processing Unit (CPU) core using a serial form. The parallel form was optimised for execution on a GP-GPU, and the rationale for this choice was discussed in Section 4.3.2. Next, Section 6.3 captures the results obtained from running three Monte-Carlo simulation batches. Each batch was split into two partitions, one operating on mostly inconsistent IA networks and the other on consistent IA networks. Section 6.4 then represents an academic discussion of the results obtained and their significance. Finally, Section 6.5 highlights the most important findings and lists some areas of future work.

6.2 Calculation

The parallel form of Constraint Satisfaction Problem (CSP) path consistency presented here was derived from the most general form of parallel path consistency [57]. An OpenCL parallel path consistency algorithm that is optimised for GP-GPU is presented. The algorithm may not be the optimal version achievable but does achieve speed-ups in execution. The aim is to test whether there is any merit in using GP-GPU for path consistency algorithms of CSPs. The goal is not to compare parallel processing on the GP-GPU to that of a MCPU, FPGA or DSP. However, OpenCL was chosen as this should allow the algorithm to be easily ported to MCPUs, FPGAs and DSPs.

The general form applies constraint propagation to the entire CSP relational matrix [57]. This allows the algorithm to easily execute in parallel by performing the computation for each cell simultaneously. The reason for this choice and not using other algorithms [49, 50] is due to the basic structure of the general form. These structures generally give good initial performance on GP-GPU as they contain the least amount of branching in the computation.

The general form performs unnecessary computations, as all cells are updated even if there is no prior update for a related cell. While this does not affect the total computation time it does affect device utilisation. Other optimisations to reduce the unnecessary computations of the serial version are testing for the set of all operators {all} and quitting when the empty set $\emptyset$ is generated [58]. For the GP-GPU process it is preferable to perform the same computation on all nodes. This is because any branching unwinds into sequential parallel computations, and therefore these optimisations are not used.

In the pseudo code, upper case bold face is used for matrices, lower case bold face is used for vectors, upper case plain typeface for sets and lower case plain typeface is used for scalar types. Constraint propagation is denoted using the mathematical '$\circ$' operator. The mathematical operator '$\cup$' denotes set union,'$\cap$' denotes set intersection and '$\subset$' denotes a proper subset of CSP relations. A set inversion operation for CSP relations is defined as the 'inverse' algorithm, and for each operator present in the input relation places only the inverse operator into the resulting set [4].

Algorithm 19 represents the pseudo code for the host portion of OpenCL total path

Algorithm 19 ParallelPC ($\boldsymbol{N}$: relation matrix)

```
copy N to GP-GPU
allocate relation matrix M same dimensions as N on GP-GPU
allocate Boolean matrix U same dimensions as N on GP-GPU
set path-consistent Boolean p = True
set update Boolean u = True
while p ∩ u do
  execute PCKernel (N, M, U) on GP-GPU
  execute CheckContinueKernel (M, U, p, u) on GP-GPU
  copy p and u from GP-GPU
  swap N and M
end while
copy N from GP-GPU
return p
```

consistency. The algorithm receives one input: the input CSP relational matrix. The algorithm returns the status of the network consistent as 'true' and inconsistent as 'false'. After the algorithm has completed the input CSP relational matrix is modified.

This part of the algorithm executes on the host CPU but invokes two OpenCL kernels on the GP-GPU. Kernels are Graphical Processing Unit (GPU) native code generated by either an OpenCL or a CUDA compiler from the applicable code for the algorithm. These two kernels are presented as Algorithms 20 and 21. To execute these kernels and continue computation for as long as necessary, it is necessary to copy the CSP network to the GP-GPU and after each iteration copy back the consistency and update flags. All copy operations, the launching of the kernels and the execution of the code on the host are overheads that must be taken into account as part of the total execution time for the parallel algorithm.

The algorithm sequentially computes successive updates on the outer loop [57]. There is no way to execute this sequential loop in parallel, as each round requires the outputs of the prior round as inputs. The algorithm presented here also stores the entire IA matrix and not only the upper or lower triangle.

Now recall from Section 3.3.2 that Hogge constraint propagation reduces the processing overhead of preforming constraint propagation, as instead of computing the values on the fly a lookup table is used. The size of the lookup table does not currently fit into GP-GPU internal memory. Using the external memory is possible but would reduce the efficiency of the algorithm, as accesses to external memory are much more expensive due to the small

caches for compute nodes and large latency of the synchronous dynamic Random Access Memory (RAM). Thus, it was not possible to use the Hogge [59] constraint propagation algorithm on the GP-GPU utilised for this study. However, these limitations do not exist on the CPU and the results for the parallel implementation will also be compared to that of the serial version using Hogge [59] constraint propagation on a CPU. The Hogge [59] constraint propagation algorithm should be usable on a GP-GPU in future that has a cache able to store the 72 KB look up tables.

Algorithm 20 PCKernel ($\boldsymbol{N}$: input relation matrix, $\boldsymbol{M}$: output relation matrix, $\boldsymbol{U}$: output Boolean matrix)

for all rows r of matrix $\boldsymbol{N}$ **in parallel do**
  allocate relation row vector $\boldsymbol{v}$
  for all columns c of matrix $\boldsymbol{N}$ **do**
    $\boldsymbol{v}_c = \boldsymbol{N}_{r,c}$
  end for
  for all columns c of matrix $\boldsymbol{N}$ **in parallel do**
    relation $R = \boldsymbol{N}_{r,c} \cap \text{inverse } (\boldsymbol{N}_{c,r})$
    Boolean $f = \text{False}$
    for all columns k of matrix $\boldsymbol{N}$ **do**
      relation $C = R \cap (\boldsymbol{v}_k \circ \boldsymbol{N}_{k,c})$
      Boolean $s = (C \subset R)$
      $R = R \cap C$
      $f = f \cup s$
    end for
    $\boldsymbol{M}_{r,c} = R$
    $\boldsymbol{U}_{r,c} = f$
  end for
end for

Algorithm 20 is used on each iteration of Algorithm 19 to update the CSP relation matrix. This algorithm receives three inputs: a relation matrix to check, and a relation matrix to store the updated network and a Boolean matrix to store which cells were updated. Constraint propagation is applied to all cells in the same row or column as the cell under test.

The structure of Algorithm 20 is very important to correct operation on the GP-GPU. GP-GPU consist of a cell of compute nodes and each compute node can compute up to 32 parallel threads. In OpenCL, work-groups are assigned to individual cells of compute node array and a work-item to a single compute, where each parallel run of the work-item is executed on the threads (or warps). The algorithm breaks up the relation matrix

into work-groups for each row and a work-item for each column. Work-groups cannot be synchronised and thus there is no attempt to collate copies from the network in columns rows. However, work-items assigned to a single work-group can be synchronised, and thus each work-group makes a local copy of all elements in the same row. This should improve memory access times by reducing the number of accesses on a single row.

Furthermore, there are no conditional statements in the algorithm. Where conditional logic is required, it is performed by using masks and simple Boolean operations. This ensures that the work-group does not compute the branches sequentially. All work-items complete processing in the same amount of time as a side-effect of removing the branching, but this behaviour is not required.

Each work-item then updates a single-row cell in both the output relation and update matrix. As only work-items within a single work-group can be synchronised, multiple accesses to the same location can be performed in any sequence. It is not safe to have multiple write accesses to the same location nor write and read accesses at the same time. This is the reason for the output relation matrix.

Algorithm 21 CheckContinueKernel ($\boldsymbol{N}$: relation matrix, $\boldsymbol{U}$: Boolean matrix, p : path-consistent Boolean, u : update Boolean

p = true
u = false
allocate Boolean row vector $\boldsymbol{q}$
allocate Boolean row vector $\boldsymbol{v}$
for all columns c of matrix $\boldsymbol{N}$ **in parallel do**
  $\boldsymbol{q}_c$ = true
  $\boldsymbol{v}_c$ = false
  for all rows r of matrix $\boldsymbol{N}$ **do**
    $\boldsymbol{q}_c = \boldsymbol{q}_c \cap (\boldsymbol{N}_{r,c} \neq \emptyset)$
    $\boldsymbol{v}_c = \boldsymbol{v}_c \cap \boldsymbol{U}_{r,c}$
  end for
end for
for all columns c of matrix $\boldsymbol{N}$ **do**
  $p = p \cap \boldsymbol{v}_c$
  $u = u \cup \boldsymbol{f}_c$
end for

Algorithm 21 then collates the consistency of the network and whether any updates have occurred. This must be performed in a single work-group as synchronisation is required to update the returned consistency and update flags. A work-item is assigned

per column, where each work-item tests consistency across all rows. Updates in the rows are also checked across all rows. Finally, the work-group will collate all the results outside the parallel loop.

This algorithm, while not optimal at using the GP-GPU resources, is required to avoid copying either matrix to the host processor. This is the only alternative as the host CPU can also guarantee that checks are done atomically. Nevertheless, more recent GP-GPUs can execute multiple kernels in parallel so this may not be such a big problem. On the GP-GPU used, the two kernels (Algorithms 20 and 21) were run sequentially. It would be possible to achieve greater performance with a GPU that allows parallel execution of kernels by masking off the overhead time of Algorithm 21 with a parallel execution of Algorithm 20.

The sequential algorithm is given by Allen [4] but changes to improve the operation of the sequential algorithm [58] are used. To ensure fair comparisons are drawn we assume the host CPU cannot use the Hogge [59] constraint propagation algorithm. This would be the case if the memory required to store the lookup matrices are not available. It also allows conclusions to be drawn without a bias towards the CPU implementation. More advanced GPUs should allow the Hogge [59] constraint propagation algorithm to be used and the results should still hold true but the speed-up will likely decrease. Results are included for the serial path consistency algorithms using both Allen and Hogge constraint propagation. Furthermore, the serial algorithm treats each cell as updated on the first round as we generate a randomly generated scenario as a batch of nodes. The pseudo code for the serial path consistency is presented as Algorithm 22 and Algorithm 23.

The C and OpenCL code for both algorithms can be found on the accompanying disc, the contents of which are outlined in Appendix D.

Algorithm 22 SerialPC ($\boldsymbol{N}$: relation matrix)

```
allocate relation matrix M same dimensions as N
allocate Boolean matrix U same dimensions as N
allocate Boolean matrix V same dimensions as N
set path-consistent Boolean p = true
set update Boolean u = true
set U_{r,c} = true ∀r, c
while p ∧ u do
  u = false
  for all rows r in N do
    for all columns c in N do
      if r = c then
        M_{r,c} = N_{r,c}
        V_{r,c} = false
      else
        p = SerialPCIteration (N, U, r, c, M, V)
        if ¬p then
          quit both loops
        end if
        if (¬u) ∧ V_{r,c} then
          u = true
        end if
      end if
    end for
  end for
  swap pointer to N and M
  swap pointer to U and V
end while
if p∧ pointer to N is set to M then
  copy M into N
end if
return p
```

Algorithm 23 SerialPCIteration ($\boldsymbol{N}$: input relation matrix, $\boldsymbol{U}$: input Boolean matrix, r : row, c : column $\boldsymbol{M}$: output relation matrix, $\boldsymbol{V}$: output Boolean matrix)

relation $R = \boldsymbol{N}_{r,c} \wedge Inverse(\boldsymbol{N}_{c,r})$
Boolean $f = (R \neq \boldsymbol{N}_{r,c})$
for all k in the range 1 to number of intervals in $\boldsymbol{N}$ **do**
  if $(k \neq r) \wedge (k \neq c) \wedge (\boldsymbol{U}_{r,k} \vee \boldsymbol{U}_{k,c})$ **then**
    if $(\mathrm{b} \in \boldsymbol{N}_{r,k} \wedge \mathrm{bi} \in \boldsymbol{N}_{k,c}) \vee (\mathrm{bi} \in \boldsymbol{N}_{r,k} \wedge \mathrm{b} \in \boldsymbol{N}_{k,c}) \vee (\mathrm{d} \in \boldsymbol{N}_{r,k} \wedge \mathrm{di} \in \boldsymbol{N}_{k,c})$ **then**
      skip this loop iteration
    end if
    relation $T = R \cap \mathrm{CP}\ (\boldsymbol{N}_{r,k}, \boldsymbol{N}_{k,c})$
    if $T = \emptyset$ **then**
      return false
    else if $T \subset R$ **then**
      $R = T$
      $f = true$
    end if
  end if
end for
$\boldsymbol{M}_{r,c} = R$
$\boldsymbol{V}_{r,c} = f$
return true

6.3 Results

Figures 6.1–6.6 plot the results for all three Monte-Carlo simulation batches.

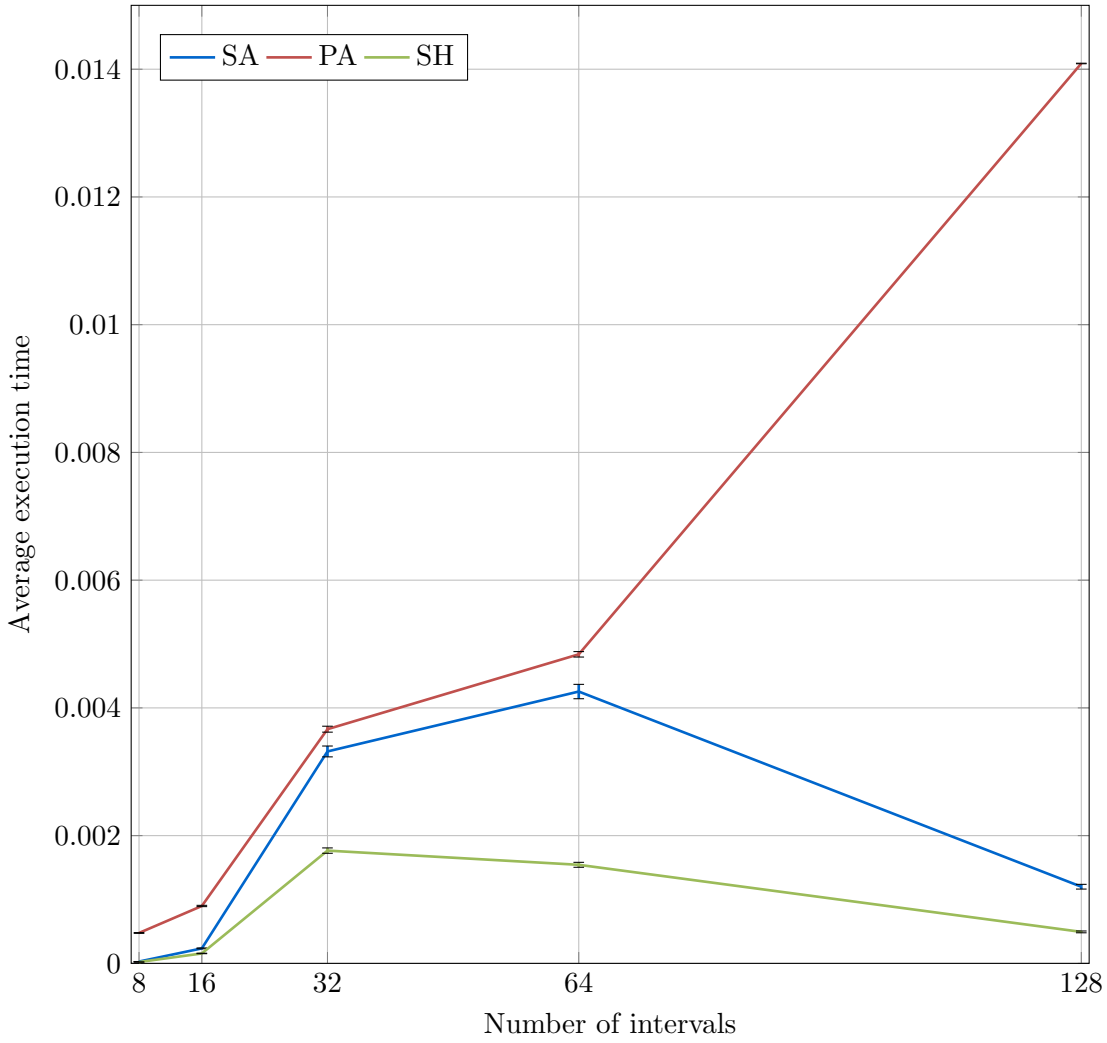


Figure 6.1: Mean execution time in seconds for path consistency when varying the network size n between 8, 16, 32, 64 and 128; the mean is taken over 1 000 Monte-Carlo runs for each point of the plots. Random CSP networks are generated using method $\mathrm{A}(n, d, s)$. Three algorithms were evaluated: serial IA using Allen constraint propagation (SA), parallel IA using Allen constraint propagation (PA) and serial IA using Hogge constraint propagation. For these results the normalised degree of relations is $d/(n-1) = 0.5$ and the average constraint size is $s = 8.6$. The blue line plots the results for serial path consistency using Allen constraint propagation (SA). The red line plots the results for parallel path consistency using Allen constraint propagation (PA). The green line plots the results for serial path consistency using Hogge constraint propagation (SH). The error bars denote the standard error of the mean for each of the points.

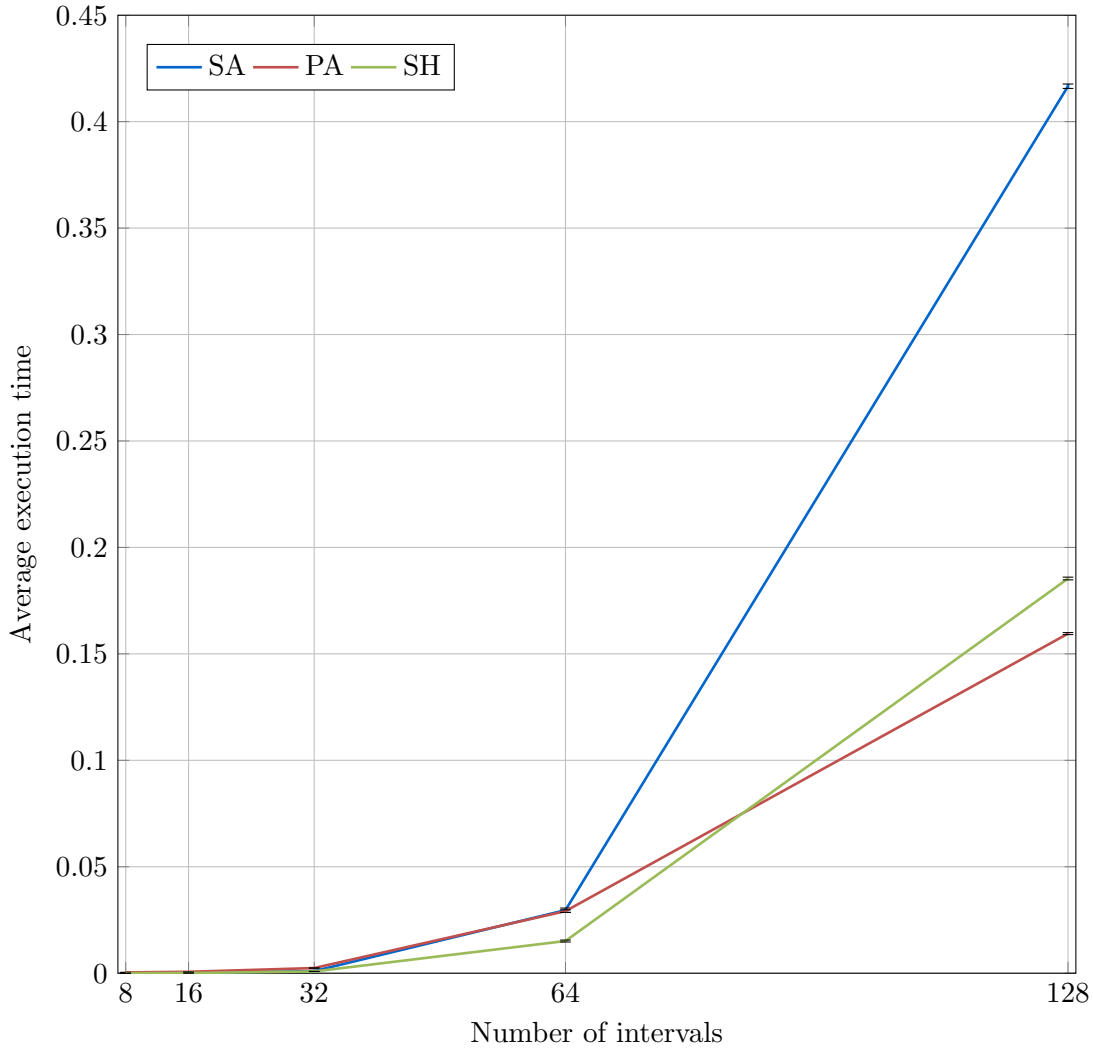


Figure 6.2: Mean execution time in seconds for path consistency when varying the network size n between 8, 16, 32, 64 and 128; the mean is taken over 1 000 Monte-Carlo runs for each point of the plots. Random CSP networks are generated using method $\mathrm{S}(n, d, s)$. Three algorithms were evaluated: serial IA using Allen constraint propagation (SA), parallel IA using Allen constraint propagation (PA) and serial IA using Hogge constraint propagation. For these results the normalised degree of relations is $d/(n-1) = 0.5$ and the average constraint size is $s = 8.6$. The blue line plots the results for serial path consistency using Allen constraint propagation (SA). The red line plots the results for parallel path consistency using Allen constraint propagation (PA). The green line plots the results for serial path consistency using Hogge constraint propagation (SH). The error bars denote the standard error of the mean for each of the points.

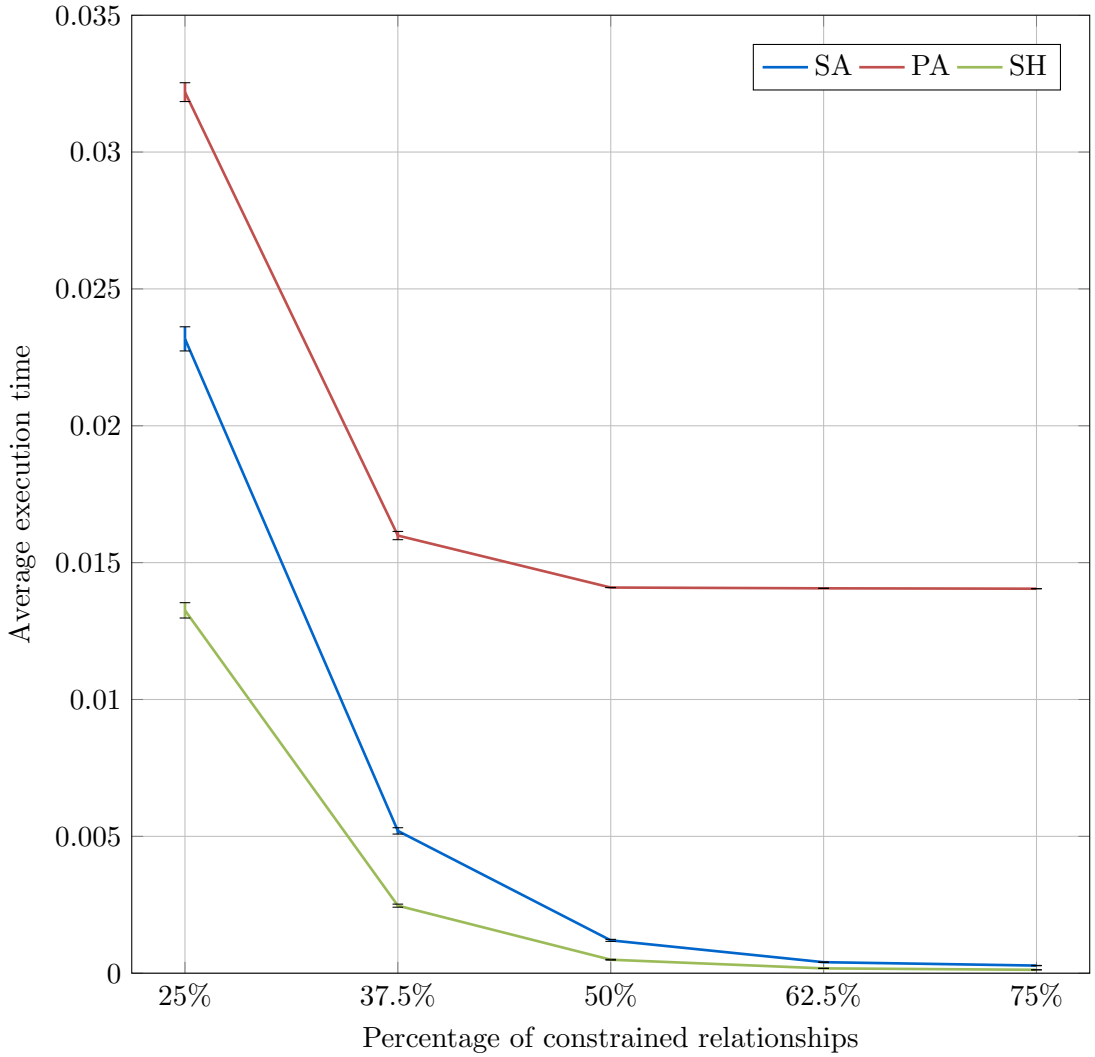


Figure 6.3: Mean execution time in seconds for path consistency when varying the normalised degree of relations $d/(n-1)$ between $\frac{1}{4}$, $\frac{3}{8}$, $\frac{1}{2}$, $\frac{5}{8}$ and $\frac{3}{4}$; the mean is taken over 1 000 Monte-Carlo runs for each point of the plots. Random CSP networks are generated using method $\mathrm{A}(n, d, s)$. For these results the network size is $n = 127$ and the average constraint size is $s = 8.6$. The blue line plots the results for serial path consistency using Allen constraint propagation (SA). The red line plots the results for parallel path consistency using Allen constraint propagation (PA). The green line plots the results for serial path consistency using Hogge constraint propagation (SH). The error bars denote the standard error of the mean for each of the points.

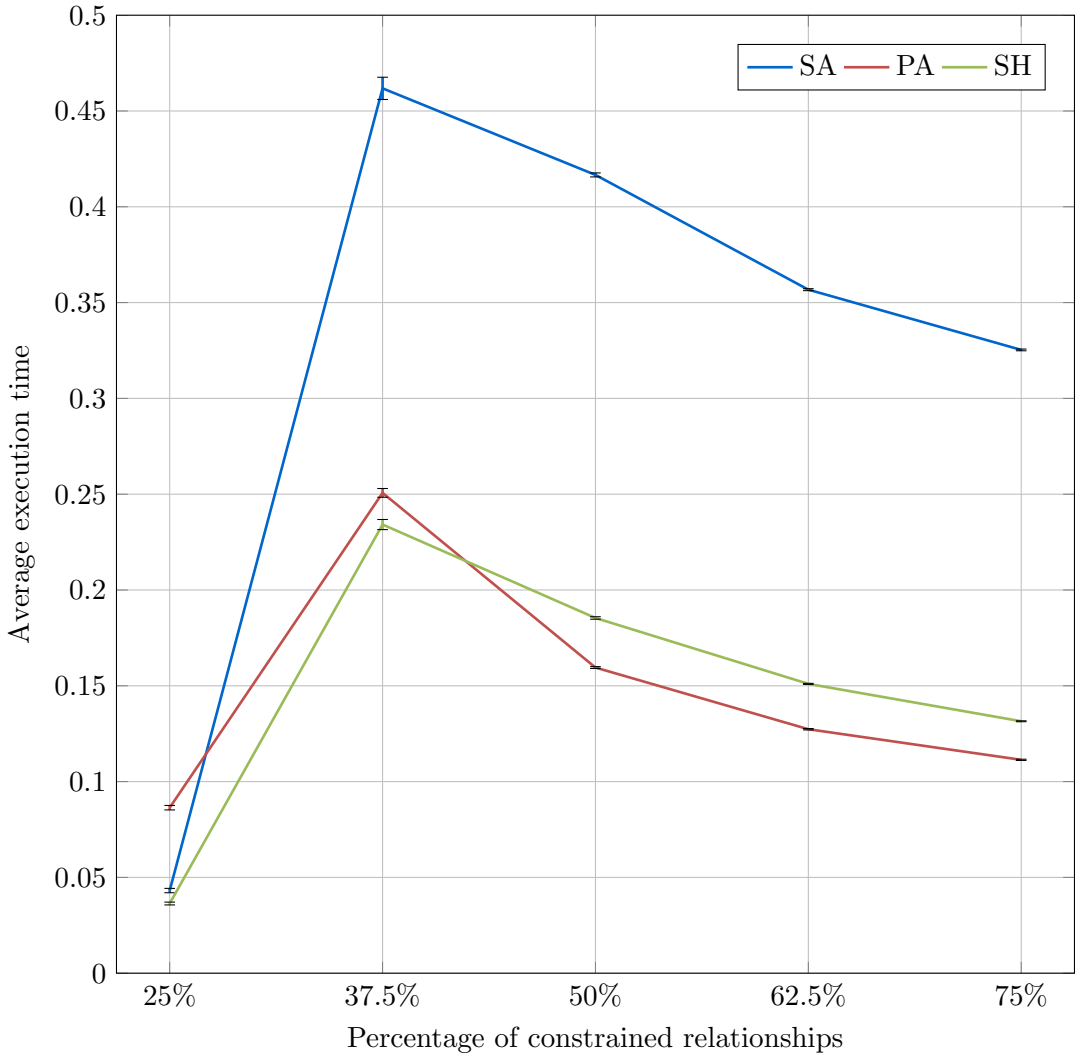


Figure 6.4: Mean execution time in seconds for path consistency when varying the normalised degree of relations $d/(n-1)$ between $\frac{1}{4}$, $\frac{3}{8}$, $\frac{1}{2}$, $\frac{5}{8}$ and $\frac{3}{4}$; the mean is taken over 1 000 Monte-Carlo runs for each point of the plots. Random CSP networks are generated using method $\mathrm{S}(n, d, s)$. For these results the network size is $n = 127$ and the average constraint size is $s = 8.6$. The blue line plots the results for serial path consistency using Allen constraint propagation (SA). The red line plots the results for parallel path consistency using Allen constraint propagation (PA). The green line plots the results for serial path consistency using Hogge constraint propagation (SH). The error bars denote the standard error of the mean for each of the points.

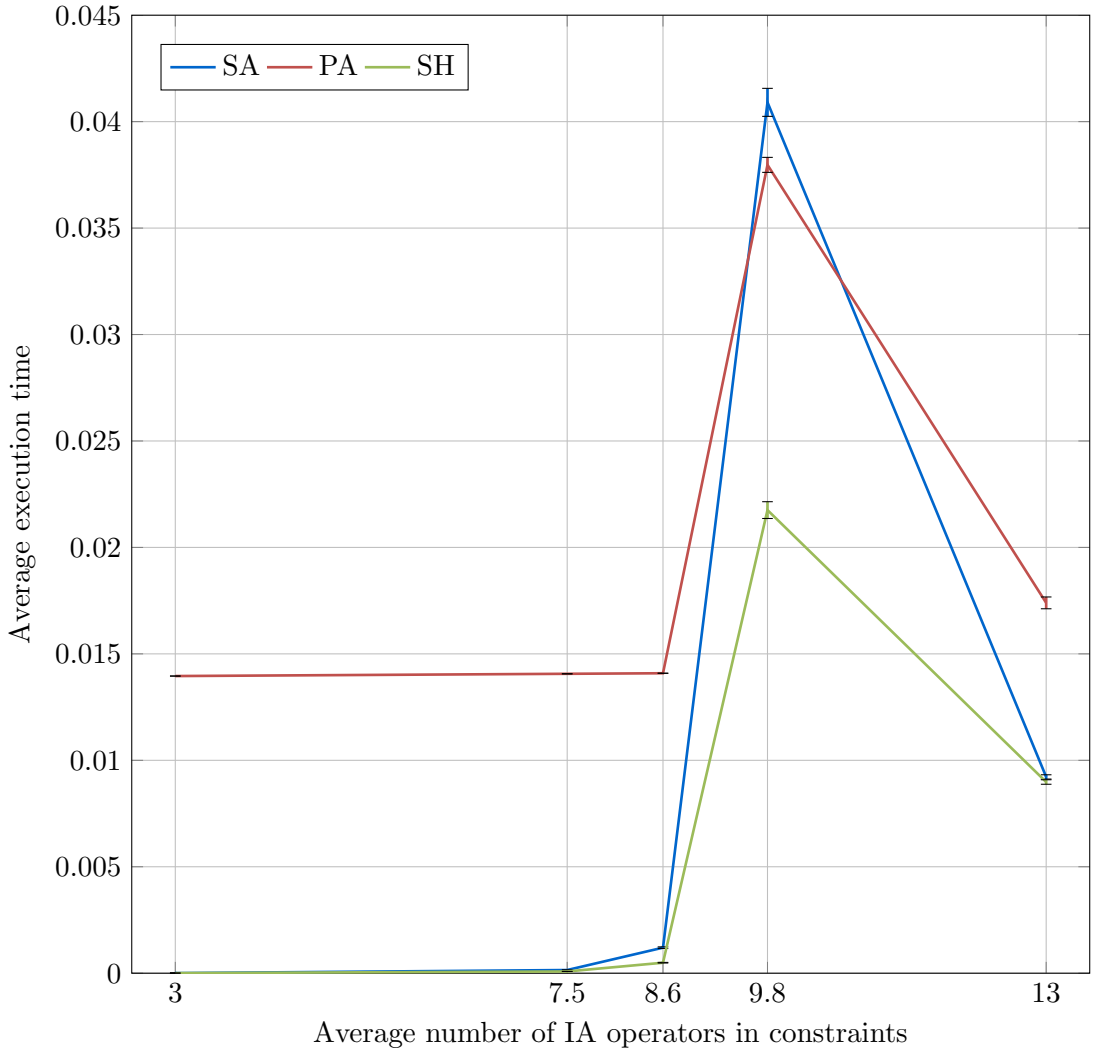


Figure 6.5: Mean execution time in seconds for path consistency when varying the average constraint size n between 3, 7.5, 8.6, 9.8 and 13; the mean is taken over 1 000 Monte-Carlo runs for each point of the plots. Random CSP networks are generated using method $\mathrm{A}(n, d, s)$. Three algorithms were evaluated: serial IA using Allen constraint propagation (SA), parallel IA using Allen constraint propagation (PA) and serial IA using Hogge constraint propagation. For these results the network size is $n = 127$ and the normalised degree of relations is $d/(n-1) = 0.5$. The red line plots the results for serial path consistency using Allen constraint propagation (SA). The blue line plots the results for parallel path consistency using Allen constraint propagation (PA). The green line plots the results for serial path consistency using Hogge constraint propagation (SH). The error bars denote the standard error of the mean for each of the points.

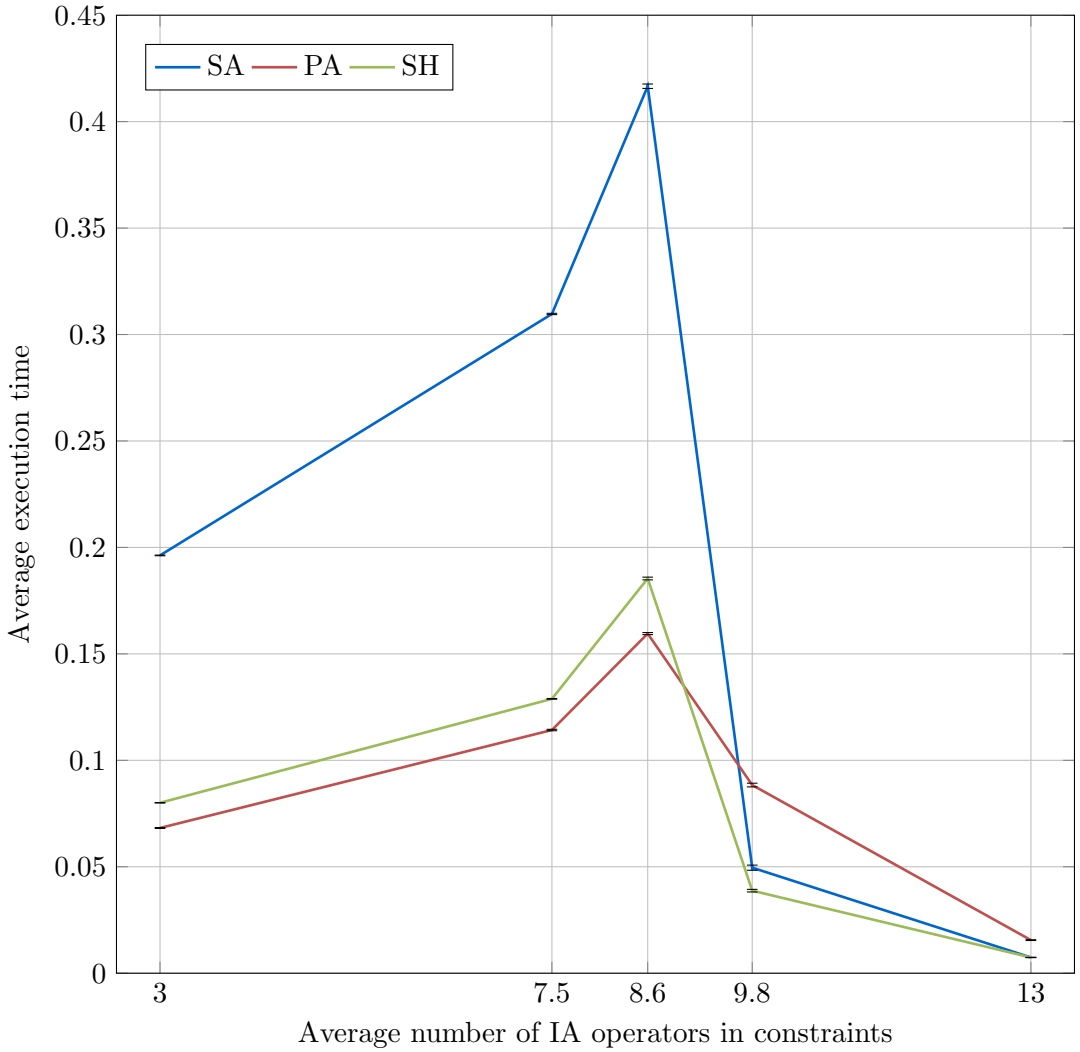


Figure 6.6: Mean execution time in seconds for path consistency when varying the average constraint size n between 3, 7.5, 8.6, 9.8 and 13; the mean is taken over 1 000 Monte-Carlo runs for each point of the plots. Random CSP networks are generated using method $\mathrm{S}(n, d, s)$. Three algorithms were evaluated: serial IA using Allen constraint propagation (SA), parallel IA using Allen constraint propagation (PA) and serial IA using Hogge constraint propagation. For these results the network size is $n = 127$ and the normalised degree of relations is $d/(n-1) = 0.5$. The red line plots the results for serial path consistency using Allen constraint propagation (SA). The blue line plots the results for parallel path consistency using Allen constraint propagation (PA). The green line plots the results for serial path consistency using Hogge constraint propagation (SH). The error bars denote the standard error of the mean for each of the points.

6.4 Discussion

The results indicate that the parallel path consistency algorithm in some cases requires between $\frac{1}{3}$ and $\frac{1}{2}$ of the execution time of the serial path consistency algorithm. That is for the serial path consistency algorithm using Allen constraint propagation. The gain is achieved by performing multiple updates in parallel per iteration. Conversely, when the network is inconsistent, small, sparsely constrained or the constraints are highly ambiguous[2], the parallel algorithm requires more execution time than either of the serial algorithms.

The main reasons for the worse performance in some cases are due to the speed discrepancies between the GP-GPU and CPU processing nodes and overheads. The GP-GPU clock rate is 1.344 GHz, while the CPU is clocked at 2.2 GHz, thus an update to a single node in the CSP takes longer. Therefore, the GP-GPU must compute more required updates per iteration for the processing time to be the same as or lower than that of the CPU.

The overheads arise from the need to copy the network to and from the GP-GPU. These we call the copy overheads. These are incurred at the start and end of the execution of the parallel algorithm. Note that there are also overheads that arise from launching the algorithms on the GP-GPU and from copying the two flags to the CPU. One flag is used to check if computation must continue and the other flag is used to check for inconsistencies. These we call the iteration overheads, which accumulate for the entire duration of execution. They contribute to the execution time of a single iteration as does the processing of an update to a single node, as all the updates for all nodes, in one iteration, occur in parallel on the GP-GPU.

There must be sufficient required updates per iteration to yield a lower total execution time on the GP-GPU than executing serially on the CPU. Furthermore, the gains achieved for all iterations run on the GP-GPU must also exceed the copy overheads.

The execution time gained by using the parallel version is almost equivalent to using Hogge constraint propagation for serial path consistency on the CPU. This can largely be attributed to the overheads of copying data to and from the GP-GPU, which means that

[2]The constraints contain many operators.

even though the processing time is still faster, there is no gain in overall execution time. For the remainder, we mostly compare the two equivalent algorithms.

Using more modern GP-GPUs should make it possible to also improve the execution time of parallel path consistency using Hogge constraint propagation. All the GPUs available to my research did not have sufficient cache sizes to store the Hogge lookup matrices. The achievable speed-up of the parallel algorithm should decrease as Hogge constraint propagation would require less processing per update. Furthermore, the ranges where the parallel algorithm can achieve better performance than the equivalent serial algorithm would also decrease. The remainder of this section will mainly compare the two equivalent parallel and serial path consistency algorithms.

6.4.1 Effect of increasing the network size

Increasing the number of intervals, n, when using method A(d, n, s) leads to poor performance for the parallel path consistency algorithm versus the serial path consistency algorithms. This is due to the fact that the large networks (64 and 128) that are generated are all inconsistent and this inconsistency is found quickly by path consistency algorithms. This again confirms that large random CSP networks are less likely to be consistent [57]. For the smaller sizes the networks are increasingly consistent, which leads to the downward concave curve in Figure 6.1.

In all cases, the parallel version requires two significant overheads, copying the network to and from the GPU, that are not required for serial path consistency. Effectively, the processing time contributes little to the execution time of the parallel path consistency algorithm and the copy overhead dominates for large inconsistent networks. This overhead increases quadratically for each network size increase and cannot be regained from processing time improvements, as few checks are required to find the inconsistency.

In some cases, the large increase in execution time for the parallel version on network sizes of 128 can be attributed to multiple iterations each with minimal updates. An iteration of the parallel version has three overheads: launching the parallel path consistency algorithm, launching the check continue algorithm and copying the two flags back to the host. These overheads only become a problem when there are insufficient updates performed in an iteration.

Switching to the S(d, n, s) scenario generation methods leads to the parallel path consistency having mostly superior execution time when the network size increases. This is due to the fact that consistent networks require more updates to be performed by the path consistency algorithms [49]. This is evident from the fact that in Figure 6.2 the processing time for the serial version now exceeds that of the parallel version in 6.1. The overheads are efficiently overcome for all network sizes, leading to a smaller total execution time on average. Even for small networks, the parallel path consistency algorithm has a more deterministic execution time and should be preferred for real-time applications. This can be seen from the smaller variation in total execution time depicted by the bars showing standard error.

6.4.2 Effect of increasing the normalised degree of relations

The results for varying the normalised degree of relations $\frac{d}{n-1}$ when using method A(d, n, s) leads to poor performance for the parallel path consistency algorithm versus the serial path consistency algorithm. In all these cases, the networks generated were inconsistent and the number of required updates is low. For most of the cases the root cause of the increased execution time on the GP-GPU is the iteration overheads. This can be seen in Figure 6.3, where the parallel execution time exceeds the serial execution time in all cases.

When the network is under-constrained[3], there are more iterations that require multiple updates. Thus, the processing time accounts for more of the execution time and the iteration overheads on the GP-GPU do not impact performance as badly. However, the gains are not enough to account for both the iteration and copying overheads. As the number of relations that are constrained is increased, the number of iterations required to find the inconsistency also decreases. Thus, the iteration overheads for launching the parallel algorithm on the GP-GPU and the copy overheads dominate the total execution time.

Using method S(d, n, s), scenario generation leads to cases where superior execution time for the parallel algorithm is achieved. For sparsely constrained networks, a locally consistent solution is found quickly and the overheads for the parallel algorithm dominates. This can be seen in Figure 6.4, where the total execution time is almost equivalent to that

[3]There are few constraints that are not the set {all}.

of the serial path consistency algorithms. Thus, for networks where fewer than a quarter of their independent relations known, it is better to use the serial algorithm. The serial algorithm can skip the processing of combinations that will lead to the set {all}, leading to fewer serial checks required per iteration.

Figure 6.4 shows that the processing time for all algorithms decreases as the degree of uncertainty in the network decreases. The reason for this behaviour comes from the method of generating the constraint networks. The method $\mathrm{S}(d, n, s)$ used to generate Figure 6.4, generates mostly consistent networks. First a consistent network is generated and subsequently altered to have some random constraints. The higher the normalised degree of relations, the less random constraints are present in the network. Thus, at about 37.5% the path consistency algorithms begin to be able to detect this original consistent solution to the problem quickly. The speed of detecting the consistent scenario increased as fewer random constraints are present.

When the constrained relations increase slightly, the number of iterations to find a path-consistent solution also increase. The relationships interact more frequently, leading to more cases where the processing cannot be skipped by the serial version. The gradual decrease in execution time, from the maximum at a normalised degree of relations of roughly 0.375, is due to the fact that gradually fewer iterations are required to find the consistent network. This will mean that above a certain limit close to a normalised degree of relations of 1.0, the execution time on the GP-GPU should again approach that of the serial algorithm.

Interestingly, between a normalised degree of relations of roughly 0.44 and 0.75, the parallel version takes on average less time to execute than the serial version using Hogge constraint propagation. This can be attributed to the fact that there are many cells that require updates in a large percentage of the iterations of both the serial and parallel algorithms. Thus, even though Hogge constraint propagation is superior to Allen constraint propagation, the parallel processing advantage dominates.

6.4.3 Effect of increasing the average constraint size

The results for varying the average constraint size when using method $\mathrm{A}(d, n, s)$ lead to poor performance for the parallel path consistency algorithm versus the serial path

consistency algorithms. The only exception to this can be seen in Figure 6.5 at an average constraint size of 9.8.

For networks with constraints that have an average constraint size of less than 9.8 and those that have more, the parallel version has a worse execution time. The copy overheads are again the largest contributor to this decrease in performance. In some cases, the iteration overheads also contribute significantly to the execution time and this leads to the larger variation in execution time for the parallel algorithm.

When the $S(d, n, s)$ method is used to generate the random scenarios, processing time peaks for all the algorithms at an average constraint size of around 8.6 in Figure 6.6. This can be attributed to the fact that for consistent networks with average size constraints, the path consistency algorithm must iterate many more times to find the path-consistent solution. Each of these iterations also require many updates to be performed. While this is also the case for lower average constraint size, for these cases the amount of processing required in an update and the number of iterations decreases. Thus, the parallel algorithm execution time is better on average than both serial algorithms for networks with average constraint sizes smaller than or equal to 8.6.

Once the average constraint size increases, however, the execution time for all the path consistency algorithms again decreases. This is due to the fact that there are now many fewer updates required to find a path-consistent solution. In these cases, the number of updates per iteration is still large. Thus, the parallel path consistency algorithm total execution time is within that of the serial algorithms. However, the copy overheads lead to the situation that the total execution time for the parallel version does not improve over that for the serial algorithms.

Nevertheless, these results are very positive for the work of Rodriguez and Anger [158], who propose defining a new algebra that has relations that can cater for different temporal reference points as per the theory of general relativity. My results seems to indicate that such an algebra should be computationally feasible, and would likely not require much more processing that is currently required by Interval Algebra.

6.5 Conclusion

In this chapter, the execution time of parallel path consistency on GP-GPUs was investigated. Pseudo code for the OpenCL parallel path consistency algorithm was provided. This OpenCL algorithm was based on the original work of Ladkin and Maddux [57] and was optimised for execution on a GP-GPU. As the prevalence of parallel processing architectures such as MCPUs and GPUs increases, it will become ever more important to make use of parallel algorithms on these architectures.

Given the results it is clear that the parallel form of path consistency should be preferred for temporal constraint satisfaction problems where the network is more likely to be consistent. This is due to the fact that in most cases when the random scenarios generated were consistent, the parallel path consistency led to a decrease in total execution time. While it was not possible to verify a parallel path consistency algorithm making use of Hogge [59] constraint propagation, these results should still hold for GP-GPUs that are able to use it. For networks that may be inconsistent, the serial version provides better performance.

There are two strategies that can lead to quicker execution time in all cases. The first requires an idle CPU core and GP-GPU. In this case, launching both the serial and parallel algorithms at the same time is desirable, because whether the network is consistent one of the algorithms will have a lower execution time. Alternatively, it is possible to first run a few iterations on a CPU core. If no path-consistent scenario or inconsistency is found after a few iterations, then the GP-GPU processing can be launched.

Considering only consistent networks the following should be a guide for using the parallel GP-GPU algorithm. The GP-GPU path consistency should be preferred for all network sizes greater than 64 nodes. However, the GP-GPU memory space will limit the maximum network size that can be supported as there is typically less memory available to the GP-GPU. When the network is constrained with fewer than $\frac{1}{4}$ of the constraints being known, the serial path consistency can always be preferred. For all other cases, the GP-GPU version will provide a decrease in execution time. When the average constraint size is greater than 9.8 operators, the serial path consistency can always be used. For average constraint sizes of less than 9.8 operators, the GP-GPU path consistency again

provides better performance for consistent networks.

Additional research is still required for other processing architectures such as MCPUs and FPGAs. However, from the results of this study it does not appear that using MCPUs will lead to very good speed-ups. This is due to the fact that the thread launch time on these architectures becomes a significant contributor to the update processing time, whereas for GP-GPUs this is relatively small. Even with these small parallel overheads the GP-GPUs' speed-up currently is only between 2 and 3 times as fast. Furthermore, even using optimisations to the OpenCL code to mitigate the above problem, MCPUs also have fewer computation cores than GP-GPUs. Nevertheless, in cases where the copy overheads dominate, the MCPU path consistency would benefit. The same arguments hold for DSPs as per MCPUs. Finally, it appears that FPGAs might provide better parallel execution for path consistency, as efficient path consistency circuits could reduce the update processing time [53].

Future work should consider what the impact of a more expressive algebra such as the relativistic algebra considered by Rodriguez and Anger [158] is on the behaviour of path consistency, as my results seems to indicate that such an algebra should be computationally feasible. Other work can first apply the more sophisticated sub-graph closure techniques of Gerevini and Saetti [60] to the full Interval Algebra. Thereafter, an analysis can be performed for parallel algorithms utilising this technique on GP-GPUs, MCPUs, FPGAs or DSPs.

Chapter 7

Closing Remarks

7.1 Introduction

In the field of multisensor management, there is a diverse set of algorithms to solve for the various aspects. Multisensor management includes modelling the sensor management problem to devise a holistic solution. A realised sensor management system must then perform task generation, task prioritisation and scheduling of all sensors. In this research, the focus was on the scheduling aspect of managing a Mechanically Steered Multistatic Surveillance Radar Network (MSMSRN)[1].

MSMSRN was chosen as the sensors to manage in this evaluation of IA: as multistatic radars are a type of sensor system that employs the strengths of both monostatic and bistatic radar measurement capabilities to make better or more measurements of a target of interest. Using multistatic radar enabled me to address realistic problems, such as small boat detection, applying technologies being researched by both the Council for Scientific and Industrial Research (CSIR) and the University of Cape Town (UCT).

Information fusion can be used to combine the information gathered by a multistatic radar into better tracks of the targets. Multistatic measurements add value to information fusion systems, not only at the higher level, but also at the lowest level in terms of measurement fusion, which can happen on the receiving radar. This method of measurement fusion is practical and should result in better detection, tracking and classification.

Sensor scheduling appears to have many heuristic approaches, which, while compu-

[1] After discussions with Prof. M. R. Inggs

tationally efficient, are tightly coupled to the problems they solve. Other scheduling approaches make use of optimisation and Monte-Carlo algorithms which can be very computationally costly. Initial research into Interval Algebra (IA) showed it to be a simpler and equally intuitive scheduling algorithm[2].

IA is an interesting temporal reasoning language for intervals, or distinct periods of time. IA contains a rich set of tools to enable the scheduling of tasks for any sensor type, including temporal and durational organisation, and fuzzy and Bayesian logic extensions. Thus IA is useful for scheduling sensors, as the goal in sensor scheduling is to use limited sensor resources to perform as many tasks as possible. Using the set of tools in IA, a timeline of tasks can be built that consist of a realisable scheduling plan. Examples of tasks for radars include detection, tracking and imaging of targets of interest. In this work, only tracking tasks were utilised, but the results should be extendible to the other two radar functions. A published conference paper, created by the author along with L. O. Wabeke and his supervisors, showed the initial merits of IA [1] against a simpler heuristic algorithm.

This research hypothesis that, *IA, as a temporal reasoning language, performs scheduling of sensor dwells efficiently and effectively*, was then formulated. The hypothesis could be validated by answering the following three research questions. Can IA allow a multistatic radar system to make more multistatic measurements of targets? Can IA perform as well as established multisensor scheduling techniques in terms of computational requirements? Is it possible to enhance the performance of IA by making use of parallel processing architectures?

[2]After discussion with Dr J. P. de Villiers

7.2 Multistatic Radar Scheduling Investigation

The simulation environments defined in Section 4.2 give a suitable statistical framework to test the MSMSRN scheduling algorithms[3]. These were created to answer the first two research questions. They define a binary multistatic surveillance radar network and a rectangular observed area. The radars are mechanically steered and the rotators do not change direction until a scan is completed. Greedy Randomised Adaptive Search Procedure (GRASP) (meta-heuristic mathematical optimisation algorithm) and IA scheduling algorithms that can schedule the binary radar system are provided.

The more practical concerns that must be addressed when scheduling a mechanically steered surveillance radar network include: uncertainty in the azimuth sequence of targets; the effect of target motion and therefore the need for scheduling nimbleness; handling of monostatic scanning, tracking and confirmation tasks; and the implementation of scheduling algorithms[4].

Three Monte-Carlo simulations batches for a binary radar system were executed to produce results, where targets were randomly placed and move along random trajectories. The first two simulation batches were used to vary the number of targets and the radar beam width to compare IA to GRASP. This was done to show that IA is not computationally cumbersome and can provide equivalent performance to GRASP. The final simulation batch compared the three scheduling modes used along with IA scheduling against slow, fast and rapidly accelerating targets and looked at scheduling nimbleness for surveillance radars. This was done to determine if nimble scheduling is required for surveillance radars. The results of the simulation runs are provided and discussed in Chapter 5.

While IA scheduling performed as well as GRASP in the binary radar simulations, indications are that it would be easier to adapt this scheduling algorithm to cater for more radars. IA is also more amenable to cope with a richer set of temporal constraints

[3]The simpler simulation environment was also used to generate the conference paper [1]. The basic elements of which were modified by L. O. Wabeke from his single radar scheduling simulations. This enabled implementing the multistatic scenario and scheduling algorithms used in this research.

[4]The more complex simulation environment made use of some information fusion functions and a simple multitarget tracking single-radar environment first developed by Dr J. P. de Villiers. In this environment, the Nimble IA Scheduler along with the other two simple scheduling strategies were implemented. This was achieved by generating a multistatic radar network feeding a centralised information fusion system (or connecting the information fusion functions appropriately).

that may be imposed. Since GRASP is based on heuristics, its implementation is more adversely affected by changes in imposed constraints than IA.

IA guarantees that a maximum length solution will be found for the target selections that are made as it performs local optima examinations. Instead, conventional optimisation algorithms carry out global searches. Thus, combinations of IA with conventional optimisation algorithms such as GRASP should lead to even more effective scheduling algorithms.

While the processing time required is within the real time available for both algorithms, adding many more iterations or more targets would make this difficult to sustain. Thus, while performing a global search is desirable, it would not always be possible. Employing parallel processing architectures such as multi-core central and general-purpose graphical processing units could make the use of a global search technique practically feasible.

In most cases, the IA scheduling algorithm already provides good performance and only a small incremental improvement is possible by utilising a conventional optimisation algorithm alongside IA. Thus, the performance given by the IA algorithm may be sufficient and it then provides a computationally effective solution to the radar network scheduling problem. The IA algorithm can be used as implemented in Matlab® for around 10 targets without requiring parallel processing.

IA algorithms can be run for 16 384 targets in about 0.5 s of processing time using the C programming language[5]. The research documented in Chapter 6 considers how parallel architectures can be leveraged to reduce the computation time required by IA. Doing so together with running the separate GRASP iterations in parallel makes it possible to produce better results within realistic scan durations.

Given the results obtained, it would appear that nimble scheduling is not a major problem for a surveillance radar network that is tracking realistic targets that are neither ballistic nor supersonic. However, the simple prediction strategy does provide a minor improvement and may be worth considering if there is spare processing capacity in the system.

IA can make more measurements than simple heuristic algorithms but performs equally

[5]These are results from the algorithms, implemented in the C programming language, run on an Intel® Core™ i7-2670QM CPU at 2.2 GHz.

as well as mathematical programming techniques. It is often not possible to use more rigorous techniques to solve sensor scheduling problems as real-time performance is necessary. Thus, IA should be preferred for its simplicity, which captures the intrinsic essence of temporal information between time intervals, which can easily represent sensor tasks. While only mechanically steered multistatic radar scheduling was investigated, there is nothing that stops IA from being used to schedule heterogeneous sensors of different types. Given the findings, IA should provide better performance within realistic sensor scheduling time frames, which answers the first and second research questions.

Thus, after all optimisations IA was able to match the performance and computational requirements of GRASP. This was proven in the journal paper, by the author of this work et al., that was published in the Elsevier Information Fusion journal [2]. As shown in the conference paper, the heuristic algorithm was no comparison to either IA or GRASP as it required a lot of computational requirements to match the performance achieved. Furthermore, IA does not require modifications to handle a richer set of constraints in temporal ordering. This is due to the fact that IA captures the intrinsic physical possibilities of intervals of time in relation to others.

7.3 Parallel Architecture Investigation

Section 4.3 provides a framework within which the serial and parallel IA network consistency algorithms can be compared to each other in terms of processing time required. This framework was created to answer the third and final research question. This framework is compared to published research that looks at how IA algorithms could be utilised on parallel processing architectures. Using this simulation environment, multiple runs were executed for each of the varied independent variables: number of intervals in the network, expressiveness of the IA relationships and number of unknown relationships. The results of these simulation runs are captured and discussed in Chapter 6.

In Chapter 6 pseudo code for the parallel path consistency algorithm defined originally by Ladkin and Maddux [57] ported to OpenCL is provided. This algorithm can make use of Hogge [59] constraint propagation and can be run on General-Purpose Graphical Processing Units (GP-GPUs), Multicore Central Processing Units (MCPUs), Field-Programmable Gate Arrays (FPGAs) and Digital Signal Processors (DSPs), unlike the algorithm of Ladkin and Maddux [57]. It will become ever more important to use parallel algorithms that are capable of using these parallel processing architectures as their prevalence increases.

Given the results, it is clear that the parallel form of path consistency should be preferred for temporal constraint satisfaction problems where the network is more likely to be consistent. This is due to the fact that in most cases when the random scenarios generated were consistent, the parallel path consistency led to a decrease in total execution time. While it was not possible to verify a parallel path consistency algorithm making use of Hogge [59] constraint propagation, these results should still hold for GP-GPUs that are able to use it. For networks that are inconsistent, the serial version provides better performance.

There are two strategies that can lead to quicker execution time in all cases. The first requires an idle CPU core and GP-GPU. In this case, launching both the serial and parallel algorithms at the same time is desirable, because whether or not the network is consistent one of the algorithms will have a lower execution time. Alternatively, it is possible to first run a few iterations on a CPU core. If no path-consistent scenario or inconsistency is found after a few iterations, then the GP-GPU processing can be launched.

Considering only consistent networks the following should be a guide for using the parallel GP-GPU algorithm. The GP-GPU path consistency should be preferred for all network sizes greater than 64 nodes. However, the GP-GPU memory space will limit the maximum network size that can be supported as there is typically less memory available to the GP-GPU. When the network is constrained with fewer than $\frac{1}{4}$ of the constraints being known, the serial path consistency can always be preferred. For all other cases, the GP-GPU version will provide a decrease in execution time. When the average constraint size is greater than 9.8 operators, the serial path consistency can always be used. For average constraint sizes of less than 9.8 operators, the GP-GPU path consistency again provides better performance for consistent networks.

Additional research is still required for other processing architectures such as MCPUs, FPGAs and DSPs. However, the findings do not suggest that using either MCPUs or DSPs will be useful for parallel path consistency. Nevertheless, these findings prove the third research question, that *IA can be used with modern parallel processing architectures*, and thus ensure that IA will remain relevant into the future.

7.4 Future Work

It is often hard to gauge which scheduling algorithms are better to use than others, as not only do the researchers solve very different problems, but often very little work has been done to compare the algorithms in existing literature. More work is needed to perform good comparisons of scheduling algorithms when they can solve the same problem. In addition, which problems the algorithms solve best should also be explored more thoroughly.

In this research, the radar positioners/rotators were not allowed to change the direction of rotation during a scan. Scheduling the antenna positioner rotation changes for the radars could produce better solutions. This is due to the fact that the solution space will drastically change if the radars change direction. Furthermore, it may produce more interesting constraints, thereby proving the usefulness of the other IA relations. Thus, an investigation into adding such tasks at the right time should be performed.

When the radars scan in an opposing direction the geometry of the targets seen by that radar will produce one optimal sequence. On the other hand, if the radars scan in the same direction, a different sequence of targets will become the optimal sequence. This is because the geometry of the targets in the scene determines the maximum length sequence of targets that can be measured in the multistatic mode. The turn-around event requires more IA relations. This is because it can occur during, at the start or at the end of a tracking task.

In this research, only mechanically steered radars were considered, as these systems are currently employed more widely and such systems are available for further research at the CSIR and UCT. However, Active Electronically Scanned Array (AESA) and Digital Beam-Forming (DBF) radars are becoming cheaper and therefore more prevalent. AESA or DBF radars would be ideal for multistatic measurements as their beams can be positioned very quickly. This is due to the fact that the beams are positioned electronically and therefore experience no inertia. This is very different behaviour to the mechanically steered radars that were considered for this research. Thus, there is a need to investigate the use of IA in AESA/DBF multistatic radar scheduling.

Furthermore, both AESA and DBF are multifunction radars, which are better suited for tracking and searching as the beam can be scheduled much quicker than a mechani-

cally steered radar. In the case of scheduling a multifunction radar, nimbleness is often significantly more important as usually either the platform housing the radar is moving quickly or it is likely that ballistic and supersonic targets may be encountered. Thus, the nimble IA scheduler could be adapted for use in Multifunction Radar (MFR), and this would be an interesting topic to pursue in the future.

However, for these types of radars, the beam width also increases as the beam is pointed off bore-sight of the radar. This non-linear increase in beam width of the radar would produce more uncertainties in the target sequences for off-bore beam locations. The receive power of the signal also decreases as the beam is pointed off bore-sight of the array face. Thus, this type of radar would also have different duration constraints, which would need to be investigated.

The duration algebra of IA was alluded to by Allen [4] and should be investigated as future work. However, no published works describing this in detail seem to exist. Investigating Allen's duration algebra or other duration algebras to add constraints around scan times would be meaningful for all the above-mentioned radar systems. This is because the scan duration is often fixed or tasks require a specific duration.

The duration of each task that needs to be performed during the scan will impact the total scan duration. Radar tasks have duration requirements as well, for example so as to meet a minimum probability of detection in a scanning task. Furthermore, moving the positioners can also be seen as tasks that have a duration limited by the maximum inertia the positioner can withstand. Thus, using a duration algebra would leverage this information to build schedules for time-constrained systems that have scan duration requirements. The allowable scan duration is very short for a time-constrained system, and thus the duration of each executed task is important.

It was seen in this research that a parallel architecture, namely the GP-GPU, can be used to speed-up IA processing. However, it was also found that the computation time required would also increase if either the number of targets in the scene or the beam width of the radar were to increase. Although not tested conclusively in the research, the computation time must also increase when more radars are used in a multistatic radar nework. The reason for the increased computation time is due to the fact that the parallel form of the IA algorithm executes redundant computations. Thus, ways to reduce the

amount of redundant computations are required to ensure that this form of IA will still be usable under all conditions.

Without these optimisations more computational hardware will be required either under increasing target load on the mechanically steered multistatic radars or if their beam widths are increased. Usually systems with larger beam widths are designed to be cheaper, and thus this would impose systems engineering problems with increasing cost on the scheduling subsystem.

Further research could also formulate the algorithms defined by Gerevini and Saetti [60], a more sophisticated sub-graph closure technique, for parallel processing of IA. Allowing this technique designed for PA to be used for IA would be beneficial. An investigation of the use of these adapted algorithms along with MCPUs, GP-GPUs, FPGAs or DSPs would then also be beneficial.

Finally, the formulation of a general task-organising algorithm using IA could be an interesting research topic. IA should be able to solve efficiently any problem requiring temporal organisation. This research would, however, require background research into other fields where task organisation is required.

Future work should consider the impact of a more expressive algebra such as the relativistic algebra considered by Rodriguez and Anger [158] on the behaviour of path consistency, as these results seems to indicate that such an algebra should be feasible.

7.5 Conclusion

IA can provide equivalent performance to GRASP if the constraints are reduced. Given problems that require a richer set of constraints, these can easily be handled using IA. Nimble IA scheduling can provide a means to increase the multistatic measurements made and reduce those that are missed due to prediction inaccuracies. IA path consistency can also be used on GP-GPUs but only provides speed-ups under specific conditions.

Finally, through the various publications, it was possible to interact with many international researchers and South African radar researchers. The initial work on IA and mechanically steered multistatic research was presented in Chicago to an international audience. Each of the journal publications, created from this thesis, allowed three important research institutes namely the CSIR, UCT and the University of Pretoria (UP) to collaborate in the emerging field of multistatic radar. The IA scheduling algorithms and parallel forms of IA implemented in IA should allow all three research institutions to utilise radar networks to continue working in this important field. Multistatic radar is a subset of Multiple-In Multiple-Out (MIMO) radar, and thus future work is enabled.

Bibliography

[1] R. W. Focke, L. O. Wabeke, J. P. de Villiers, M. R. Inggs, Implementing Interval Algebra to schedule mechanically scanned multistatic radars, in: Proceedings of the 14th International Conference on Information Fusion (FUSION), vol. 1, (2011), pp. 1–7.

[2] R. W. Focke, J. P. de Villiers, M. R. Inggs, Interval Algebra – An effective means of scheduling surveillance radar networks, Information Fusion (2014). http://www.sciencedirect.com/science/article/pii/S1566253514000888. doi:http://dx.doi.org/10.1016/j.inffus.2014.08.002.

[3] B. Nebel, H.-J. Bürckert, Reasoning about temporal relations: A maximal tractable subclass of Allen's Interval Algebra, Journal of the ACM 42 (1995) 43–66. http://doi.acm.org/10.1145/200836.200848. doi:10.1145/200836.200848.

[4] J. F. Allen, Maintaining knowledge about temporal intervals, Communications of the ACM 26 (1983) 832–843. http://doi.acm.org/10.1145/182.358434. doi:10.1145/182.358434.

[5] F. E. White, Data fusion lexicon, Technical Report, DTIC Document, 1991.

[6] G. W. Ng, K. H. Ng, Sensor management – What, why and how, Information Fusion 1 (2000) 67–75. http://www.sciencedirect.com/science/article/pii/S1566253500000099. doi:http://dx.doi.org/10.1016/S1566-2535(00)00009-9.

[7] N. Xiong, P. Svensson, Multi-sensor management for information fusion: Issues and approaches, Information Fusion 3 (2002) 163–186. http://www.sciencedirect.com/science/article/pii/S1566253502000556. doi:10.1016/S1566-2535(02)00055-6.

[8] J. Llinas, C. Bowman, G. L. Rogova, A. Steinberg, E. Waltz, F. White, Revisiting the JDL data fusion model II, in: Proceedings of the 7th International Conference on Information Fusion (FUSION), (2004), pp. 1218–1230. http://isif.org/fusion/proceedings/fusion04CD/IF04/IF04-1218.pdf.

[9] A. N. Steinberg, C. L. Bowman, F. E. White, Revisions to the JDL data fusion model, in: Proceedings of SPIE, vol. 3719, (1999), pp. 430–441. http://dx.doi.org/10.1117/12.341367. doi:10.1117/12.341367.

[10] E. Blasch, A. Steinberg, S. Das, J. Llinas, C. Chong, O. Kessler, E. Waltz, F. White, Revisiting the JDL model for information exploitation, in: Proceedings of the 16th International Conference on Information Fusion (FUSION), (2013), pp. 129–136.

[11] E. Blasch, Level 5 (user refinement) issues supporting information fusion management, in: Proceedings of the 9th International Conference on Information Fusion (FUSION), (2006), pp. 1–8. doi:10.1109/ICIF.2006.301581.

[12] Y. Bar-Shalom, X. R. Li, Multitarget Multisensor Tracking: Principles and Techniques, YBS Publishing, Storrs, CT, 1995.

[13] A. O. Hero, D. Cochran, Sensor management: Past, present, and future, IEEE Sensors Journal 11 (2011) 3064–3075. doi:10.1109/JSEN.2011.2167964.

[14] V. Krishnamurthy, D. V. Djonin, Optimal threshold policies for multivariate POMDPs in radar resource management, IEEE Transactions on Signal Processing 57 (2009) 3954–3969. doi:10.1109/TSP.2009.2022915.

[15] P. R. Barbosa, E. K. P. Chong, Adaptive sensing for target tracking in covert operations, in: Proceedings of SPIE, vol. 7336, (2009), pp. 73360A–73360A–10. http://dx.doi.org/10.1117/12.818357. doi:10.1117/12.818357.

[16] S. Guha, K. Munagala, Approximation algorithms for partial-information based stochastic control with Markovian rewards, in: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS), (2007), pp. 483–493. doi:10.1109/FOCS.2007.23.

[17] H. Su, Q. Wang, K. Ren, K. Xing, Jamming-resilient dynamic spectrum access for cognitive radio networks, in: Proceedings of the IEEE International Conference on Communications (ICC), (2011), pp. 1–5. doi:10.1109/icc.2011.5962525.

[18] J. M. Molina López, J. García Herrero, F. J. Jiménez Rodríguez, J. R. Casar Corredera, Cooperative management of a net of intelligent surveillance agent sensors, International Journal of Intelligent Systems 18 (2003) 279–307. http://dx.doi.org/10.1002/int.10089. doi:10.1002/int.10089.

[19] C. Kreucher, K. Kastella, A. O. Hero, Sensor management using an active sensing approach, Signal Processing 85 (2005) 607–624. http://www.sciencedirect.com/science/article/pii/S0165168404003068. doi:http://dx.doi.org/10.1016/j.sigpro.2004.11.004.

[20] S. Blackman, R. Popoli, Design and Analysis of Modern Tracking Systems, Artech House Radar Library, Artech House, Norwood, MA, 1999.

[21] A. R. Benaskeur, F. Rhéaume, Adaptive data fusion and sensor management for military applications, Aerospace Science and Technology 11 (2007) 327–338. http://www.sciencedirect.com/science/article/pii/S1270963807000120. doi:10.1016/j.ast.2007.01.005.

[22] S. Musick, R. Malhotra, Chasing the elusive sensor manager, in: Proceedings of the IEEE National Aerospace and Electronics Conference, vol. 1, (1994), pp. 606–613. doi:10.1109/NAECON.1994.332850.

[23] Z. Ding, A survey of radar resource management algorithms, in: Proceedings of the Canadian Conference on Electrical and Computer Engineering, (2008), pp. 001559–001564. doi:10.1109/CCECE.2008.4564804.

[24] G. Earl, B. Ward, Frequency management support for remote sea-state sensing using the JINDALEE skywave radar, IEEE Journal of Oceanic Engineering 11 (1986) 164–173. doi:10.1109/JOE.1986.1145165.

[25] S. L. C. Miranda, C. J. Baker, K. Woodbridge, H. D. Griffiths, Phased array radar

resource management: A comparison of scheduling algorithms, in: Proceedings of the IEEE Radar Conference, (2004), pp. 79–84.

[26] R. Tharmarasa, T. Kirubarajan, J. Peng, T. Lang, Optimization-based dynamic sensor management for distributed multitarget tracking, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 39 (2009) 534–546.

[27] F. Barbaresco, J. C. Deltour, G. Desodt, B. Durand, T. Guenais, C. Labreuche, Intelligent M3R radar time resources management: Advanced cognition, agility & autonomy capabilities, in: Proceedings of the International Radar Conference – Surveillance for a Safer World, (2009), pp. 1–6.

[28] Y. Xun, M. M. Kokar, K. Baclawski, Control based sensor management for a multiple radar monitoring scenario, Information Fusion 5 (2004) 49–63. http://www.sciencedirect.com/science/article/pii/S1566253503000484. doi:10.1016/j.inffus.2003.08.001.

[29] V. Krishnamurthy, Emission management for low probability intercept sensors in network centric warfare, IEEE Transactions on Aerospace and Electronic Systems 41 (2005) 133–151. doi:10.1109/TAES.2005.1413752.

[30] H. Godrich, A. P. Petropulu, H. V. Poor, Power allocation strategies for target localization in distributed multiple-radar architectures, IEEE Transactions on Signal Processing 59 (2011) 3226–3240. doi:10.1109/TSP.2011.2144976.

[31] H. Aftab, N. Raj, P. Cuff, S. Kulkarni, Mutual information scheduling for ranking, in: Proceedings of the 14th International Conference on Information Fusion (FUSION), (2011), pp. 1–8.

[32] Y. Cheng, X. Wang, M. Morelande, B. Moran, Information geometry of target tracking sensor networks, Information Fusion 14 (2013) 311–326. http://www.sciencedirect.com/science/article/pii/S1566253512000310. doi:http://dx.doi.org/10.1016/j.inffus.2012.02.005.

[33] P. Z. Thunemann, R. Mattikalli, S. Arroyo, P. Frank, Characterizing the tradeoffs between different sensor allocation and management algorithms, in: Proceedings of the 12th International Conference on Information Fusion (FUSION), (2009), pp. 1473–1480.

[34] Q. Ling, Y. Fu, Z. Tian, Localized sensor management for multi-target tracking in wireless sensor networks, Information Fusion 12 (2011) 194–201. Special Issue on Information Fusion in Future Generation Communication Environments. http://www.sciencedirect.com/science/article/pii/S1566253511000042. doi:10.1016/j.inffus.2011.01.003.

[35] R. Tharmarasa, T. Kirubarajan, A. Sinha, T. Lang, Decentralized sensor selection for large-scale multisensor-multitarget tracking, IEEE Transactions on Aerospace and Electronic Systems 47 (2011) 1307–1324.

[36] S. R. Martin, A. J. Newman, The application of particle swarm optimization and maneuver automatons during non-Markovian motion planning for air vehicles performing ground target search, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), (2008), pp. 2605–2610. doi:10.1109/IROS.2008.4651133.

[37] P. J. Shea, J. Kirk, D. Welchons, Adaptive sensor management for multiple missions, in: Proceedings of SPIE, vol. 7330, (2009), pp. 73300M–73300M–12. http://dx.doi.org/10.1117/12.818892. doi:10.1117/12.818892.

[38] Y. He, E. K. P. Chong, Sensor scheduling for target tracking in sensor networks, in: Proceedings of the 43rd IEEE Conference on Decision and Control (CDC), vol. 1, (2004), pp. 743–748. doi:10.1109/CDC.2004.1428743.

[39] C. Kreucher, A. O. Hero, K. Kastella, D. Chang, Efficient methods of non-myopic sensor management for multitarget tracking, in: Proceedigns of the 43rd IEEE Conference on Decision and Control (CDC), vol. 1, (2004), pp. 722–727. doi:10.1109/CDC.2004.1428735.

[40] S. Ahlberg, P. Hörling, K. Johansson, K. Jöred, H. Kjellström, C. Mårtenson, G. Nei-

der, J. Schubert, P. Svenson, P. Svensson, J. Walter, An information fusion demonstrator for tactical intelligence processing in network-based defense, Information Fusion 8 (2007) 84–107. Special Issue on the Seventh International Conference on Information Fusion (FUSION) – Part II. http://www.sciencedirect.com/science/article/pii/S1566253505000928. doi:10.1016/j.inffus.2005.11.002.

[41] K. Sycara, R. Glinton, B. Yu, J. Giampapa, S. Owens, M. Lewis, C. Grindle, An integrated approach to high-level information fusion, Information Fusion 10 (2009) 25–50. Special Issue on High-level Information Fusion and Situation Awareness. http://www.sciencedirect.com/science/article/pii/S1566253507000437. doi:10.1016/j.inffus.2007.04.001.

[42] R. Popoli, S. Blackman, Expert system allocation for the electronically scanned antenna radar, in: Proceedings of the American Control Conference, (1987), pp. 1821–1826.

[43] J. M. Molina López, S. J. Jimenez Rodríguez, J. R. Casar Corredera, Fuzzy reasoning for multisensor management, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC), Intelligent Systems for the 21st Century, vol. 2, (1995), pp. 1398–1403.

[44] V. S. Chernyak, Fundamentals of Multisite Radar Systems: Multistatic Radars and Multistatic Radar Systems, CRC Press, Boca Raton, FL, 1998.

[45] N. J. Willis, H. Griffiths, Advances in Bistatic Radar, SciTech Publishing, Raleigh, NC, 2007.

[46] J. Li, P. Stoica, MIMO Radar Signal Processing, Wiley, Hoboken, NJ, 2009.

[47] C. Baker, An introduction to multistatic radar, NATO SET-136 Lecture Series – Multistatic Surveillance and Reconnaissance: Sensor, Signals and Data Fusion (2009).

[48] E. Shakshuki, A. Trudel, Y. Xu, D. C. Rine, A multi-agent temporal constraint satisfaction system based on Allen's Interval Algebra and probabilities, in: A. I.

Ghazi (Ed.), Agent Technologies and Web Engineering, IGI Global, Hershey, PA, 2009, pp. 56–76.

[49] S. Y. Susswein, T. C. Henderson, J. L. Zachary, C. Hansen, P. Hinker, G. C. Marsden, Parallel path consistency, International Journal of Parallel Programming 20 (1991) 453–473. http://dx.doi.org/10.1007/BF01547895. doi:10.1007/BF01547895.

[50] S. Keretho, R. Loganantharaj, V. N. Gudivada, Parallel path-consistency algorithms for constraint satisfaction, in: Proceedings of the 3rd International Conference on Tools for Artificial Intelligence, (1991), pp. 516–517. doi:10.1109/TAI.1991.167040.

[51] A. Samal, T. Henderson, Parallel consistent labeling algorithms, International Journal of Parallel Programming 16 (1987) 341–364. http://dx.doi.org/10.1007/BF01407901. doi:10.1007/BF01407901.

[52] S. Kasif, On the parallel complexity of discrete relaxation in constraint satisfaction networks, Artificial Intelligence 45 (1990) 275–286. http://www.sciencedirect.com/science/article/pii/000437029090009O. doi:http://dx.doi.org/10.1016/0004-3702(90)90009-O.

[53] P. R. Cooper, M. J. Swain, Arc consistency: Parallelism and domain dependence, Artificial Intelligence 58 (1992) 207–235. http://www.sciencedirect.com/science/article/pii/000437029290008L. doi:http://dx.doi.org/10.1016/0004-3702(92)90008-L.

[54] L. M. Kirousis, Fast parallel constraint satisfaction, Artificial Intelligence 64 (1993) 147–160. http://www.sciencedirect.com/science/article/pii/000437029390063H. doi:http://dx.doi.org/10.1016/0004-3702(93)90063-H.

[55] M. Fabiunke, Parallel distributed constraint satisfaction, in: Proceendings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), vol. 99, (1999), pp. 1585–1591.

[56] V. Kumar, Algorithms for constraint-satisfaction problems: A survey, AI Magazine 13 (1992) 32.

[57] P. B. Ladkin, R. D. Maddux, On binary constraint problems, Journal of the ACM 41 (1994) 435–469. http://doi.acm.org/10.1145/176584.176585. doi:10.1145/176584.176585.

[58] P. B. Ladkin, A. Reinefeld, Effective solution of qualitative interval constraint problems, Artificial Intelligence 57 (1992) 105–124. http://www.sciencedirect.com/science/article/pii/0004370292901068. doi:10.1016/0004-3702(92)90106-8.

[59] J. C. Hogge, TPLAN: A Temporal Interval-based Planner with Novel Extensions, Technical Report, University of Illinois at Urbana-Champaign, Department of Computer Science, 1987.

[60] A. E. Gerevini, A. Saetti, Computing the minimal relations in point-based qualitative temporal reasoning through metagraph closure, Artificial Intelligence 175 (2011) 556–585. http://www.sciencedirect.com/science/article/pii/S0004370210001682. doi:10.1016/j.artint.2010.10.004.

[61] B. Nebel, Solving hard qualitative temporal reasoning problems: Evaluating the efficiency of using the ORD-Horn class, Constraints 1 (1997) 175–190. http://dx.doi.org/10.1007/BF00137869. doi:10.1007/BF00137869.

[62] Y. Bar-Shalom, Tracking and Data Association, Academic Press Professional, Inc., San Diego, CA, 1987.

[63] I. Kadar, E. R. Eadan, R. R. Gassner, Comparison of robustized assignment algorithms, in: Proceeding of SPIE, vol. 3068, (1997), pp. 240–249. http://dx.doi.org/10.1117/12.280802. doi:10.1117/12.280802.

[64] K. C. Chang, R. K. Saha, Y. Bar-Shalom, On optimal track-to-track fusion, IEEE Transactions on Aerospace and Electronic Systems 33 (1997) 1271–1276. doi:10.1109/7.625124.

[65] S. M. Sherman, Monopulse Principles and Techniques, Artech House Radar Library, Artech House, Norwood, MA, 1984.

[66] D. L. Hall, S. A. H. McMullen, Mathematical Techniques in Multisensor Data Fusion, 2nd ed., Artech House Radar Library, Artech House, Norwood, MA, 2004.

[67] L. A. Klein, Sensor and Data Fusion: A Tool for Information Assessment and Decision Making, SPIE Press, Bellingham, WA, 2004.

[68] E. Aoki, A. Bagchi, P. Mandal, Y. Boers, On the "near-universal proxy" argument for theoretical justification of information-driven sensor management, in: Proceedings of the IEEE Statistical Signal Processing Workshop, (2011), pp. 245–248. doi:10.1109/SSP.2011.5967671.

[69] E. H. Aoki, A. Bagchi, P. Mandal, Y. Boers, A theoretical look at information-driven sensor management criteria, in: Proceedings of the International Conference on Information Fusion, (2011), pp. 1–8.

[70] J. Aughenbaugh, B. La Cour, Metric selection for information theoretic sensor management, in: Proceedings of the International Conference on Information Fusion, (2008), pp. 1–8.

[71] S. Chatterjee, S. Misra, Target tracking using sensor-cloud: Sensor-target mapping in presence of overlapping coverage, IEEE Communications Letters 18 (2014) 1435–1438. doi:10.1109/LCOMM.2014.2336839.

[72] C. J. Willis, Meeting performance and sensing-cost requirements for detection and recognition systems, Proceedings of SPIE 8899 (2013) 889912–889912–13. http://dx.doi.org/10.1117/12.2027582. doi:10.1117/12.2027582.

[73] P. Chen, W. Hu, Sleep-wake up scheduling with probabilistic coverage model in sensor networks, International Journal of Parallel, Emergent and Distributed Systems 29 (2014) 1–16. http://dx.doi.org/10.1080/17445760.2013.766733. doi:10.1080/17445760.2013.766733.

[74] Z. Chen, Y. Fu, W. Xu, Sensor scheduling with intelligent optimization algorithm based on quantum theory, Mathematical Problems in Engineering 2013 (2013) 8 pages. doi:10.1155/2013/853430.

[75] R. L. Popp, A. W. Bailey, J. N. Tsitsiklis, Dynamic airborne sensor resource management for ground moving target tracking and classification, in: Proceedings of the

IEEE Aerospace Conference, vol. 3, (2000), pp. 405–415. doi:10.1109/AERO.2000.879868.

[76] K. Veeramachaneni, L. A. Osadciw, Multiple sectors, multi function, multi radar dwell time management using particle swarm optimization (M3RTM), in: Proceedings of the IEEE Conference on Radar, (2006), pp. 7–7. doi:10.1109/RADAR.2006.1631835.

[77] M. K. Schneider, Computing maximal track clusters for sensor resource management, in: Proceedings of the 12th International Conference on Information Fusion (FUSION), (2009), pp. 78–84.

[78] A. P. Saraf, G. L. Slater, An efficient combinatorial optimization algorithm for optimal scheduling of aircraft arrivals at congested airports, in: Proceedings of the IEEE Aerospace Conference, (2006), pp. 11–11. doi:10.1109/AERO.2006.1655877.

[79] L. Baoquan, L. Weimin, H. Guangjun, F. Gang, G. Gang, Research on management method of area aerial defense antimissile multi-sensor, in: Proceedings of the International Conference on E-Product E-Service and E-Entertainment (ICEEE), (2010), pp. 1–4. doi:10.1109/ICEEE.2010.5661380.

[80] D. A. Castañon, Approximate dynamic programming for sensor management, in: Proceedings of the 36th IEEE Conference on Decision and Control, vol. 2, (1997), pp. 1202–1207. doi:10.1109/CDC.1997.657615.

[81] A. Barbato, P. Giustiniani, An improved scheduling algorithm for a naval phased array radar, in: Proceedings of the International Radar Conference, (1992), pp. 42–45.

[82] P. E. Berry, D. A. B. Fogg, On the use of entropy for optimal radar resource management and control, in: Proceedings of the International Radar Conference, (2003), pp. 572–577. doi:10.1109/RADAR.2003.1278804.

[83] F. Barbaresco, Radar resources optimization by adaptive search domains priority assignment based on most threatening trajectories computation, in: Proceedings

of the 3rd Institution of Engineering and Technology Seminar on Intelligent Sensor Management, (2007), pp. 1–10.

[84] A. M. Sutton, A. E. Howe, L. D. Whitley, Using adaptive priority weighting to direct search in probabilistic scheduling, in: Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS), (2007), pp. 320–327. http://www.aaai.org/Papers/ICAPS/2007/ICAPS07-041.pdf.

[85] C. Kreucher, K. Kastella, A. O. Hero, Sensor management using an active sensing approach, Signal Processing 85 (2005) 607–624. http://www.sciencedirect.com/science/article/pii/S0165168404003068. doi:http://dx.doi.org/10.1016/j.sigpro.2004.11.004.

[86] A. Irci, A. Saranli, B. Baykal, On optimal resource allocation in multifunction radar systems, in: Proceedings of the IEEE Conference on Radar, (2006), pp. 8–8. doi:10.1109/RADAR.2006.1631875.

[87] K. Kastella, Discrimination gain to optimize detection and classification, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 27 (1997) 112–116.

[88] R. Watson, Radar resource management modelling, in: Proceedings of the International Radar Conference, (2002), pp. 562–566. doi:10.1109/RADAR.2002.1174783.

[89] J. P. Hui, K. C. Chang, Improved representations of sensor exploitation for automatic sensor management, in: Proceedings of the 6th International Conference on Information Fusion (FUSION), vol. 1, (2003), pp. 688–694. doi:10.1109/ICIF.2003.177513.

[90] S. L. C. Miranda, C. J. Baker, K. Woodbridge, H. D. Griffiths, Knowledge-based resource management for multifunction radar: A look at scheduling and task prioritization, IEEE Signal Processing Magazine 23 (2006) 66–76. doi:10.1109/MSP.2006.1593338.

[91] B. Gillespie, E. Hughes, M. Lewis, Scan scheduling of multi-function phased array

radars using heuristic techniques, in: Proceedings of the IEEE International Radar Conference, (2005), pp. 513–518. doi:10.1109/RADAR.2005.1435880.

[92] C.-G. Lee, C.-S. Shih, L. Sha, Online QoS optimization using service classes in surveillance radar systems, Real-Time Systems 28 (2004) 5–37. http://dx.doi.org/10.1023/B%3ATIME.0000033377.75148.d0. doi:10.1023/B:TIME.0000033377.75148.d0.

[93] J. P. Hansen, S. Ghosh, R. Rajkumar, J. Lehoczky, Resource management of highly configurable tasks, in: Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS), (2004), pp. 116–116. doi:10.1109/IPDPS.2004.1303070.

[94] A. Benaskeur, A. Khamis, H. Irandoust, Multisensor cooperation in military surveillance systems, in: Proceedings of the 3rd International Conference on Signals, Circuits and Systems, (2009), pp. 1–6. doi:10.1109/ICSCS.2009.5412286.

[95] J. Yu, F. Zhang, A. Zhang, W. Jin, Y. Tian, Motion parameter optimization and sensor scheduling for the sea-wing underwater glider, IEEE Journal of Oceanic Engineering PP (2013) 1. doi:10.1109/JOE.2012.2227551.

[96] V. Krishnamurthy, J. Mickova, Finite dimensional algorithms for the hidden Markov model multi-armed bandit problem, in: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 5, (1999), pp. 2865–2868. doi:10.1109/ICASSP.1999.761360.

[97] D. Strömberg, F. Lantz, Operator control of shared resources in sensor networks, in: Proceedings of the 6th International Conference on Information Fusion (FUSION), vol. 1, (2003), pp. 575–582. doi:10.1109/ICIF.2003.177498.

[98] D. Shi, T. Chen, Approximate optimal periodic scheduling of multiple sensors with constraints, Automatica 49 (2013) 993–1000. http://www.sciencedirect.com/science/article/pii/S0005109813000253. doi:http://dx.doi.org/10.1016/j.automatica.2013.01.024.

[99] D. Jun, D. Cohen, D. Jones, A direct algorithm for joint optimal sensor scheduling and MAP state estimation for hidden markov models, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, (2013), pp. 4212–4215. doi:10.1109/ICASSP.2013.6638453.

[100] H. Mostafaei, M. Meybodi, Maximizing lifetime of target coverage in wireless sensor networks using learning automata, Wireless Personal Communications 71 (2013) 1461–1477. http://dx.doi.org/10.1007/s11277-012-0885-y. doi:10.1007/s11277-012-0885-y.

[101] S. Liu, M. Fardad, P. Varshney, E. Masazade, Optimal periodic sensor scheduling in networks of dynamical systems, IEEE Transactions on Signal Processing 62 (2014) 3055–3068. doi:10.1109/TSP.2014.2320455.

[102] X. Wu, K. Zhang, C. Sun, Optimal scheduling of multiple sensors in continuous time, {ISA} Transactions 53 (2014) 793–801. http://www.sciencedirect.com/science/article/pii/S0019057813002346. doi:http://dx.doi.org/10.1016/j.isatra.2013.12.024.

[103] A. J. Newman, S. R. Martin, J. T. DeSena, J. C. Clarke, J. W. McDerment, W. O. Preissler, C. K. Peterson, Receding horizon controller using particle swarm optimization for closed-loop ground target surveillance and tracking, in: Proceedings of SPIE, vol. 7336, (2009), pp. 73360M–73360M–12. http://dx.doi.org/10.1117/12.818535. doi:10.1117/12.818535.

[104] L. Shi, P. Cheng, J. Chen, Sensor data scheduling for optimal state estimation with communication energy constraint, Automation 47 (2011) 1693–1698. http://www.sciencedirect.com/science/article/pii/S0005109811001385. doi:10.1016/j.automatica.2011.02.037.

[105] O. E. Drummond, D. Dana-Bashian, Track covariance consistency compensation performance, in: Proceedings of SPIE, vol. 7445, (2009), pp. 74450N–74450N–15. http://dx.doi.org/10.1117/12.830649. doi:10.1117/12.830649.

[106] P. Chavali, A. Nehorai, Managing multi-modal sensor networks using price theory, IEEE Transactions on Signal Processing 60 (2012) 4874–4887.

[107] V. Krishnamurthy, Algorithms for optimal scheduling and management of hidden Markov model sensors, IEEE Transactions on Signal Processing 50 (2002) 1382–1397. doi:10.1109/TSP.2002.1003062.

[108] A. Narykov, O. Krasnov, A. Yarovoy, Algorithm for resource management of multiple phased array radars for target tracking, in: Proceedings of the International Conference on Information Fusion, (2013), pp. 1258–1264.

[109] P. R. Horridge, M. L. Hernandez, Multistatic radar resource management, in: Proceedings of SPIE, vol. 5204, (2003), pp. 583–594. http://dx.doi.org/10.1117/12.504526. doi:10.1117/12.504526.

[110] T. O. Walker, M. Tummala, J. B. Michael, Pulse transmission scheduling for a distributed system of cooperative radars, in: Proceedings of the IEEE/SMC International Conference on System of Systems Engineering, (2006), pp. 6–6. doi:10.1109/SYSOSE.2006.1652310.

[111] P. R. Barbosa, E. K. P. Chong, S. Suvorova, B. Moran, Multitarget-multisensor tracking in an urban environment: A closed-loop approach, in: Proceedings of SPIE, vol. 6969, (2008), pp. 69690W–69690W–12. http://dx.doi.org/10.1117/12.777147. doi:10.1117/12.777147.

[112] B. Manjunath, J. J. Zhang, A. Papandreou-Suppappola, D. Morrell, Sensor scheduling with waveform design for dynamic target tracking using MIMO radar, in: Proceedings of the 43rd Asilomar Conference on Signals, Systems and Computers, (2009), pp. 141–145.

[113] T. Qureshi, M. Zoltowski, High resolution MIMO radar with unitary waveform matrix scheduling, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, (2013), pp. 4110–4114. doi:10.1109/ICASSP.2013.6638432.

[114] H. Godrich, A. P. Petropulu, H. V. Poor, A combinatorial optimization framework for subset selection in distributed multiple-radar architectures, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), (2011), pp. 2796–2799.

[115] H. Gao, J. Wang, C. Jiang, X. Zhang, Antenna allocation in MIMO radar with widely separated antennas for multi-target detection, Sensors 14 (2014) 20165–20187. http://www.mdpi.com/1424-8220/14/11/20165. doi:10.3390/s141120165.

[116] H. Godrich, A. Petropulu, H. V. Poor, Power allocation schemes for target localization in widely distributed MIMO radar systems, in: Proceedings of the Military Communications Conference (MILCOM), (2010), pp. 846–851. doi:10.1109/MILCOM.2010.5680193.

[117] A. Gorji, R. Tharmarasa, T. Kirubarajan, Optimal antenna allocation in MIMO radars with collocated antennas, IEEE Transactions on Aerospace and Electronic Systems 50 (2014) 542–558. doi:10.1109/TAES.2013.120384.

[118] J. F. Ralph, D. M. Jones, Automatic task assignment for mixed aircraft formations, in: Proceedings of SPIE, vol. 7307, (2009), pp. 73070N–73070N–9. http://dx.doi.org/10.1117/12.818519. doi:10.1117/12.818519.

[119] J. C. Ho, Y.-L. Chang, Minimizing the number of tardy jobs for m parallel machines, European Journal of Operational Research 84 (1995) 343–355. http://www.sciencedirect.com/science/article/pii/0377221793E0280B. doi:http://dx.doi.org/10.1016/0377-2217(93)E0280-B.

[120] C. Koulamas, Single-machine scheduling with time windows and earliness/tardiness penalties, European Journal of Operational Research 91 (1996) 190–202. http://www.sciencedirect.com/science/article/pii/0377221795001166. doi:http://dx.doi.org/10.1016/0377-2217(95)00116-6.

[121] M. Rajkumar, P. Asokan, N. Anilkumar, T. Page, A GRASP algorithm for flexible job-shop scheduling problem with limited resource constraints, International Journal of Production Research 49 (2011) 2409–2423.

[122] A. J. Orman, A. K. Shahani, A. R. Moore, Modelling for the control of a complex radar system, Computers & Operations Research 25 (1998) 239–249. http://www.sciencedirect.com/science/article/pii/S0305054897000476. doi:http://dx.doi.org/10.1016/S0305-0548(97)00047-6.

[123] H. J. Kushner, Scheduling and control of mobile communications networks with randomly time varying channels by stability methods, in: Proceedings of the 45th IEEE Conference on Decision and Control (CDC), (2006), pp. 2967–2973. doi:10.1109/CDC.2006.376882.

[124] H. J. Kushner, Control of multi-node mobile communications networks with time-varying channels via stability methods, Queueing Systems 54 (2006) 317–329. http://dx.doi.org/10.1007/s11134-006-0304-8. doi:10.1007/s11134-006-0304-8.

[125] Q. Wang, K. Ren, P. Ning, Anti-jamming communication in cognitive radio networks with unknown channel statistics, in: Proceedings of the 19th IEEE International Conference on Network Protocols (ICNP), (2011), pp. 393–402. doi:10.1109/ICNP.2011.6089079.

[126] F. R. Yu, H. Tang, Distributed node selection for threshold key management with intrusion detection in mobile ad hoc networks, Wireless Networks 16 (2010) 2169–2178. http://dx.doi.org/10.1007/s11276-010-0250-6. doi:10.1007/s11276-010-0250-6.

[127] K. Huang, S. Ahmed, A stochastic programming approach for planning horizons of infinite horizon capacity planning problems, European Journal of Operational Research 200 (2010) 74–84. http://www.sciencedirect.com/science/article/pii/S037722170801045X. doi:http://dx.doi.org/10.1016/j.ejor.2008.12.009.

[128] S. Sengupta, S. Das, M. Nasir, P. N. Suganthan, Risk minimization in biometric sensor networks: An evolutionary multi-objective optimization approach, Soft Computing 17 (2013) 133–144. http://dx.doi.org/10.1007/s00500-012-0906-5. doi:10.1007/s00500-012-0906-5.

[129] V. Krishnamurthy, J. Mickova, R. J. Evans, Beam scheduling for electronically scanned array tracking systems using multi-arm bandits, in: Proceedings of the

38th IEEE Conference on Decision and Control (CDC), vol. 1, (1999), pp. 1039–1044. doi:10.1109/CDC.1999.832932.

[130] F. Zhang, J. Chen, H. Li, Y. Sun, X. Shen, Distributed active sensor scheduling for target tracking in ultrasonic sensor networks, Mobile Networks and Applications 17 (2012) 582–593. http://dx.doi.org/10.1007/s11036-011-0311-9. doi:10.1007/s11036-011-0311-9.

[131] A. J. Orman, C. N. Potts, A. K. Shahani, A. R. Moore, Scheduling for a multifunction phased array radar system, European Journal of Operational Research 90 (1996) 13–25. http://www.sciencedirect.com/science/article/pii/037722179500307X. doi:http://dx.doi.org/10.1016/0377-2217(95)00307-X.

[132] S. L. Coetzee, K. Woodbridge, C. J. Baker, Multifunction radar resource management using tracking optimisation, in: Proceedings of the International Radar Conference, (2003), pp. 578–583. doi:10.1109/RADAR.2003.1278805.

[133] C. Liu, G. Cao, Spatial-temporal coverage optimization in wireless sensor networks, IEEE Transactions on Mobile Computing 10 (2011) 465–478.

[134] B. Wang, H. B. Lim, D. Ma, A coverage-aware clustering protocol for wireless sensor networks, Computer Networks 56 (2012) 1599–1611. http://www.sciencedirect.com/science/article/pii/S1389128612000400. doi:10.1016/j.comnet.2012.01.016.

[135] J. Li, Q.-S. Jia, X. Guan, X. Chen, Tracking a moving object via a sensor network with a partial information broadcasting scheme, Information Sciences 181 (2011) 4733–4753. Special Issue on Interpretable Fuzzy Systems. http://www.sciencedirect.com/science/article/pii/S0020025511002805. doi:10.1016/j.ins.2011.06.006.

[136] S. Sivashanmugam, C. Tsatsoulis, A Bayesian network for autonomous sensor control during polar ice sheet measurements, in: Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS), vol. 1, (2004), pp. 104–104. doi:10.1109/IGARSS.2004.1368955.

[137] C. Guettier, G. Le Lann, J.-F. Hermant, Ad hoc sensor networks, constraint programming and distributed agreement, in: Proceedings of the International Con-

ference on Information Technology: Research and Education (ITRE), (2003), pp. 286–290. doi:10.1109/ITRE.2003.1270624.

[138] F. Lantz, D. Strömberg, Task management in sensor-provided operator platforms, in: Proceedings of the International Conference on Integration of Knowledge Intensive Multi-Agent Systems (KIMAS), (2003), pp. 596–601. doi:10.1109/KIMAS.2003.1245107.

[139] P. Zhu, W. Xiong, B. Xu, A sensor management method based on an improved PSO algorithm, International Journal of Advancements in Computing Technology 4 (2012) 259–265.

[140] M. Pinedo, K. Hadavi, Scheduling: Theory, algorithms and systems development, in: W. Gaul, A. Bachem, W. Habenicht, W. Runge, W. Stahl (Eds.), Operations Research Proceedings 1991, vol. 1991 of *Operations Research Proceedings 1991*, Springer Berlin Heidelberg, 1992, pp. 35–42. http://dx.doi.org/10.1007/978-3-642-46773-8_5. doi:10.1007/978-3-642-46773-8_5.

[141] Y.-W. Hong, A. Scaglione, Group testing for binary markov sources: Data-driven group queries for cooperative sensor networks, IEEE Transactions on Information Theory 54 (2008) 3538–3551. doi:10.1109/TIT.2008.926363.

[142] F. Farokhi, K. Johansson, Stochastic sensor scheduling for networked control systems, IEEE Transactions on Automatic Control 59 (2014) 1147–1162. doi:10.1109/TAC.2014.2298733.

[143] S. S. Panwalkar, M. L. Smith, C. P. Koulamas, A heuristic for the single machine tardiness problem, European Journal of Operational Research 70 (1993) 304–310. http://www.sciencedirect.com/science/article/pii/037722179390241E. doi:http://dx.doi.org/10.1016/0377-2217(93)90241-E.

[144] R. Rajkumar, C. Lee, J. Lehoczky, D. Siewiorek, A resource allocation model for qos management, in: Proceedings of the 18th IEEE Real-Time Systems Symposium, (1997), pp. 298–307.

[145] Y. Xun, M. M. Kokar, K. Baclawski, Control based sensor management for a multiple radar monitoring scenario, Information Fusion 5 (2004) 49–63. http://www.sciencedirect.com/science/article/pii/S1566253503000484. doi:http://dx.doi.org/10.1016/j.inffus.2003.08.001.

[146] N. H. Nguyen, K. Dogancay, L. Davis, Adaptive waveform scheduling for target tracking in clutter by multistatic radar system, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, (2014), pp. 1449–1453. doi:10.1109/ICASSP.2014.6853837.

[147] M. A. Richards, J. A. Scheer, W. A. Holm (Eds.), Principles of Modern Radar: Basic Principles, SciTech Publishing, Raleigh, NC, 2010.

[148] P. van Beek, D. W. Manchak, The design and experimental analysis of algorithms for temporal reasoning, Journal of Artificial Intelligence Research 4 (1996) 1–18.

[149] P. van Beek, Reasoning about qualitative temporal information, Artificial Intelligence 58 (1992) 297–326. http://www.sciencedirect.com/science/article/pii/000437029290011L. doi:10.1016/0004-3702(92)90011-L.

[150] S. Badaloni, M. Falda, M. Giacomin, Integrating quantitative and qualitative fuzzy temporal constraints, AI Communications 17 (2004) 187–200. http://iospress.metapress.com/content/T422G91UQ4HB86XC.

[151] S. Badaloni, M. Giacomin, The algebra IA^{fuz}: A framework for qualitative fuzzy temporal reasoning, Artificial Intelligence 170 (2006) 872–908. http://www.sciencedirect.com/science/article/pii/S0004370206000403. doi:http://dx.doi.org/10.1016/j.artint.2006.04.001.

[152] S. Schockaert, M. D. Cock, Temporal reasoning about fuzzy intervals, Artificial Intelligence 172 (2008) 1158–1193. http://www.sciencedirect.com/science/article/pii/S0004370208000039. doi:http://dx.doi.org/10.1016/j.artint.2008.01.001.

[153] L. Deng, Y. Cai, C. Wang, Y. Jiang, Fuzzy temporal logic on fuzzy temporal constraint networks, in: Proceedings of the 6th International Conference on Fuzzy

Systems and Knowledge Discovery, vol. 6, (2009), pp. 272–276. doi:10.1109/FSKD.2009.464.

[154] K. Zhang, A. Trudel, Efficient heuristics for solving probabilistic Interval Algebra networks, in: Proceedings of the 13th International Symposium on Temporal Representation and Reasoning, (2006), pp. 111–120.

[155] M. Mouhoub, J. Liu, Managing uncertain temporal relations using a probabilistic Interval Algebra, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, (2008), pp. 3399–3404.

[156] R. Singer, Estimating optimal tracking filter performance for manned maneuvering targets, IEEE Transactions on Aerospace and Electronic Systems AES-6 (1970) 473–483. doi:10.1109/TAES.1970.310128.

[157] E. Blasch, P. Valin, E. Bosse, Measures of effectiveness for high-level fusion, in: Proceedings of the 13th International Conference on Information Fusion (FUSION), (2010), pp. 1–8.

[158] R. V. Rodriguez, F. D. Anger, Using constraint propagation to reason about unsynchronized clocks, Constraints 3 (1998) 191–202. http://dx.doi.org/10.1023/A%3A1009725727148. doi:10.1023/A:1009725727148.

Appendix A

Research Outputs

A.1 Implementing Interval Algebra to schedule mechanically scanned multistatic radars

The content of this conference paper [1] is used in Chapter 4. The paper appears in IEEE Xplore[1] and was presented by the author on the 7 July 2011 in the Hyatt Regency Hotel, Chicago, Illinois, USA at the 14th International Conference on Information Fusion.

A.1.1 Abstract

Resource management improves the performance of multisensor systems through efficient sensor scheduling. An algorithm that utilises Interval Algebra is presented to schedule multistatic measurements. Interval Algebra can capture temporally ordered sequences in a consistent manner. Multistatic measurements result in better tracks of targets being formed for various reasons. The algorithm is not intended to solve the problem of scheduling electronically scanned phased arrays, although using Interval Algebra may certainly be advantageous.

In this paper it is shown how Interval Algebra produces similar performance to a simpler heuristic approach for a scenario utilising narrow radar beams. The algorithm was not verified against existing resource management techniques. The intention is to present Interval Algebra as an alternative approach to solving the multiple-in multiple-out radar scheduling problem. When considering multiple targets simultaneously in the

[1] Available in IEEE Xplore at http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5977496

radar beam, the power of Interval Algebra is revealed. With no changes to Interval Algebra and only minor tweaks to how the constraints are represented, Interval Algebra can utilise this complexity to outperform the heuristic method.

A.2 State of the art in multisensor management

This review paper contains the same content as Chapter 2. It was previously submitted to the 16th International Conference on Information. A revised article may be submitted to the IEEE Transactions on Aerospace and Electronic Systems journal.

A.2.1 Abstract

This article provides a review of the current state of the art in multisensor management. Both the modelling and functional perspectives of multisensor management are evaluated. Modelling of a multisensor management problem can often lead to better solutions. Typically, models can either be myopic or prudent (considering long-term) planning. Prudent models fall into the Markovian decision processes or non-Markovian-decision/Stochastic processes. Functionally, multisensor management must generate tasks for the sensors to perform, prioritise the tasks as far as possible and schedule the sensors with as many tasks as possible. Algorithms to solve these three main aspects are either heuristic in nature or employ information theory, decision theory, rigorous mathematical programming techniques or artificial intelligence techniques. This paper also investigates the future of multisensor management.

A.3 Interval Algebra – an effective means of scheduling surveillance radar networks

This journal paper [2] contains similar content to Sections 3.3 and 4.2 as well as Chapter 5. It was first submitted to the Elsevier Information Fusion journal on the 10 December 2012 and was accepted with major revisions. The paper was revised and then resubmitted on the 4 March 2014. A second minor revision was submitted on 10 March 2014, and a third on 18 June 2014. The final minor revision was submitted on 2 August 2014 and was accepted on 4 August 2014. The manuscript was published on 16 August 2014 on ScienceDirect[2].

A.3.1 Abstract

Interval Algebra provides an effective means to schedule surveillance radar networks, as it is a temporal ordering constraint language. Thus it provides a solution to a part of resource management, which is included in the revised Data Fusion Information Group model of information fusion. In this paper, the use of Interval Algebra to schedule mechanically steered radars to make multistatic measurements for selected targets of importance is shown. Interval Algebra provides a framework for incorporating a richer set of requirements, without requiring modifications to the underlying algorithms. The performance of Interval Algebra was compared to that of the Greedy Randomised Adaptive Search Procedure and the applicability of Interval Algebra to nimble scheduling was investigated using Monte-Carlo simulations of a binary radar system. The comparison evaluates both scheduling performance and computation time required by the algorithms. The scheduling performance of the algorithms was quantified by keeping track of the number of targets that could be measured simultaneously. It was found that nimble scheduling is important where the targets are moving fast enough to rapidly change the recognised surveillance picture during a scan.

Two novel approaches for implementing Interval Algebra for scheduling surveillance radars are presented. It was found that adding targets on the fly and improving performance by incrementally growing the network is more efficient than pre-creating the full

[2]http://www.sciencedirect.com/science/article/pii/S1566253514000888

network. The second approach stemmed from constraint ordering. It was found that for simple constraint sets, the Interval Algebra relationship matrix reduces to a single vector of interval sets. The simulations revealed that an Interval Algebra algorithm that utilises both approaches can perform as well as the Greedy Randomised Adaptive Search Procedure with similar processing time requirements. Finally, it was found that nimble scheduling is not required for surveillance radar networks where ballistic and supersonic targets can be ignored. Nevertheless, Interval Algebra can easily be used to perform nimble scheduling with little modification and may be useful in scheduling the scans of multifunction radars.

A.4 Parallel path consistency for general-purpose graphical processing units

This research note contains the same content as Section 4.3 and Chapter 6. It was previously submitted on the 2 January 2013 to the Elsevier Artificial Intelligence journal but was rejected as it did not contain practical results. This shortcoming was addressed by using OpenCL to target GP-GPUs, and a major revision will be submitted to the same journal.

A.4.1 Abstract

In this paper, the execution time of parallel path consistency on general-purpose graphical processing units is investigated. Pseudo code for a novel Open Compute Language total-path consistency algorithm is provided. The algorithm was based on the original work of Ladkin and Maddux [57] and was optimised for execution on a general-purpose graphical processing unit. As the prevalence of parallel processing architectures such as general-purpose graphical processing units, multicore central processing units, digital signal processors and field programmable gate arrays increases, it will become ever more important to make use of parallel algorithms on these architectures. Open Compute Language algorithms can make use of all four of these architectures. Monte-Carlo simulations are run, on a general-purpose graphical processing unit and a single core of a multicore central processing unit, to solve randomly generated Interval Algebra networks. These random networks were generated using two techniques from the work of Nebel [61].

The results indicate that the Open Compute Language total-path consistency algorithm, executed on a general-purpose graphical processing unit in parallel, should be preferred for temporal constraint satisfaction problems where the network is more likely to be consistent. Then for consistent networks this parallel algorithm can provide execution time speed-up between 2 and 3 times that of the serial algorithm. This parallel algorithm should be preferred for the interval networks: with more than 64 intervals; with more than $\frac{1}{4}$ and less than $\frac{3}{4}$ of the constraints partially known; and with average constraint sizes greater than or equal to 3 and fewer than 9.8 operators.

Appendix B

Multisensor Scheduling Algorithms

B.1 Introduction

The Mechanically Steered Multistatic Surveillance Radar Network (MSMSRN) scheduling algorithms documented in this appendix were implemented during this research in Matlab®. Matlab was chosen as it allows fast prototyping and evaluation of alternatives. Matlab has over time become more efficient and can give results very close to those of tailored applications. Nevertheless, as the goal of this research was not to solve the real-time scheduling problem but rather to evaluate the effectiveness of Interval Algebra (IA), the actual programming environment is less important.

Matlab allows fair comparisons to be drawn between various algorithms but does not necessarily test the applicability of these algorithms to real-time systems. Furthermore, since the sensors scheduled in this research are mechanically steered surveillance radars, it could even be used to schedule systems which are used areas where not many targets are encountered. This is due to the fact that the processing requirement is in the order of seconds. For more challenging environments, my C and OpenCL algorithms have shown that IA in itself can easily be used for real-time applications for up to 16 384 targets.

This chapter provides the pseudo code for the implementation of each MSMSRN algorithm used during the research. The pseudo code is provided for engineers who wish to continue this research but may not wish to use Matlab. The scheduling algorithms

are broken down into three categories: those using IA, those using Greedy Randomised Adaptive Search Procedure (GRASP) and those using both IA and GRASP. This chapter should aid the usage of these advanced algorithms in further research endeavours.

Please refer to Section 3.3 for all the background preliminary theory required to understand this pseudo code better.

B.2 Notation Used in the Pseudo Code

This guide on notation applies in the pseudo code of all the algorithms. Matrices use upper-case bold italic characters, vectors use lower-case bold italic characters, sets use upper-case italic characters and scalar variables use lower-case italic characters. Constants and functions are given in regular type face. This includes all the normal mathematical constants.

For all the IA algorithms, the values 'b', 'm', 'o', 's', 'd', 'f' and 'e' represent the seven basic IA operators. Their inverse operators 'bi', 'mi', 'oi', 'si', 'di', 'fi' and 'e' may also appear. When the exponential function is implied it will appear outside of a set.

Sets are always contained inside the curly brackets '{' and '}'. The set operators are defined as follows: intersect '$\cap$', union '$\cup$', subset '$\subset$', superset '$\supset$', equality and assignment '$=$', subset equals '$\subseteq$' and superset equals '$\supseteq$'. For all types of variables, assignment statements are always contained outside of conditional statements, while equality checks are always performed only for conditional statements.

The contents of vectors and matrices are contained inside square brackets '[' and ']'. Both can be indexed in the number of dimensions they contain and any indexing is denoted by appropriate subscripts, where each dimension will be differentiated using a comma ','. An example is $\boldsymbol{M}_{r,c}$, where the two-dimensional matrix $\boldsymbol{M}$ is subscripted by the variables r for rows and c for columns.

A range of numbers may be represented as an interval between two numbers. For ranges the round bracket is used to show exclusion of the end point, while the square bracket implies inclusion.

Finally, the set of real numbers is denoted by '$\mathbb{R}$', integers '$\mathbb{I}$', natural numbers '$\mathbb{N}$' and natural numbers including zero '$\mathbb{N}_0$'.

B.3 Scheduling using IA

The scheduling algorithms in this section utilise Interval Algebra, which was documented in Section 3.3. In total, eight MSMSRN scheduling algorithms that utilise IA were implemented during the course of this work. This section describes each of six algorithms that only make use of IA. These six versions of the algorithm effectively perform a local search during the scheduling, as IA will ensure that a locally optimal solution is found given the target selections. The last two perform a global search and will be captured in Section B.5.

The IA schedulers make use of two underlying IA algorithms: path consistency and constraint propagation, as documented in Section 3.3. These two algorithms were originally published by Allen [4], but modifications from the work of van Beek [149] and van Beek and Manchak [148] were used to improve performance. As will be seen in each subsequent algorithm, further research in the field can be used to further optimise any IA algorithm. As both path consistency and constraint propagation were documented in Section 3.3, they will not be repeated here.

B.3.1 The I-MS algorithm (basic IA scheduling)

The Interval Algebra Mechanically steered multistatic surveillance radar network Scheduling (I-MS) algorithm was described in Chapter 4. It was utilised as the IA scheduling algorithm for the conference paper [1], which compared IA to a simple heuristic algorithm.

This was the first IA MSMSRN scheduling algorithm that was implemented. The I-MS algorithm makes use of the IA path consistency and constraint propagation algorithm as described by Allen [4] and the modifications to path consistency as per van Beek [149] and van Beek and Manchak [148].

Algorithm 24 creates an interval per target t, which represents the measurement task for radar r of target t. The algorithm receives a matrix, $\boldsymbol{A}$, which contains three dimensions. The first dimension is used to store the target azimuth ordering for a radar r; there is one row per radar. The second dimension contains the azimuth bins b for the radar r. The final dimension contains the targets t in the azimuth bin b.

The algorithm creates an interval for each target measurement task by a specific radar.

Algorithm 24 IASchedulerInit ($\boldsymbol{A}$: target order for each radar)

```
create IA network N = ∅
for all radars r do
  create handled set H = ∅
  for all bins b ∈ A_r in sequence do
    for all targets t ∈ A_{r,b} in sequence do
      create interval I_t
      add row and column t to N for I_t
      for all intervals I_n in H do
        if n ∈ A_{r,b} then
          PC (N, [n, t], {all})
        else
          PC (N, [n, t], {b})
        end if
      end for
      H = {H, I_n}
    end for
  end for
end for
return N
```

Thus, if there are T targets and R radars there will be $T \cdot R$ intervals. The intervals for all targets for a specific radar are then placed in the IA network. This is done by consulting the target azimuth ordering. Targets in the same bin have no constraints in the relationships between their intervals. Targets in different azimuth bins have the IA relationships defined by the order of their bin number.

Algorithm 25 then uses the IA network created by Algorithm 24 to schedule as many targets as possible. This algorithm receives the IA network created previously, the azimuth ordering of targets as defined above and a priority queue of targets. The priority queue is a set of targets each paired with their priority. Targets are selected by their priority to schedule the multistatic measurements, but in the case of equal priorities a random selection will apply. Then the measurement tasks for all radars are temporally set equal to each other using the IA equals operator 'e' for all radars. If the IA network is consistent, then the target is added to the set of scheduled multistatic measurements.

B.3.2 The IH-MS algorithm (Hogge constraint propagation)

The Interval Algebra Hogge constraint propagation Mechanically steered multistatic surveillance radar network Scheduling (IH-MS) algorithm makes use of Hogge [59] con-

Algorithm 25 IASchedulerRun ($\boldsymbol{N}$: IA network, $\boldsymbol{A}$: target order for all radars, Q : target priority queue)

```
create schedule S = ∅
for all targets n ∈ Q select n randomly according to priority do
  for all radars r do
    p_r = unique index of n in A_r
  end for
  set consistency c = True
  for all unique combinations a and b in index of p do
    c = c ∩ PC (N, [p_a, p_b], {e})
  end for
  if c then
    S = {S, n}
  end if
end for
return schedule S
```

straint propagation, which is an optimised implementation of [4] constraint propagation. Apart from this change the IH-MS algorithm is identical to the I-MS algorithm and thus no scheduling pseudo code will be provided. All calls to 'CP $(\cdot)$' in Algorithms 24 and 25 should be replaced with 'HoggeCP $(\cdot)$' defined here as Algorithm 31. Furthermore, one call to 'HoggeInit $(\cdot)$' defined here as Algorithm 26 should be called before using this IA scheduling algorithm. For all these algorithms, the IA relationship is assumed to be represented as a 13-bit number, where each bit is used to denote the presence or absence of one IA operator.

Algorithm 26 HoggeInit ($\boldsymbol{N}$: IA network, T_r : target order per radar r, Q : target priority queue)

```
global H_a = HoggeMatrixA ()
global H_b = HoggeMatrixB ()
global H_c = HoggeMatrixC ()
global H_d = HoggeMatrixD ()
```

Algorithm 26 creates the four required Hogge matrices as per Section 3.3.2.

Algorithm 27 creates the first Hogge matrix, which is the enumeration of [4] constraint propagation for using only the lower set of operators in both reference relationships.

Algorithm 28 creates the second Hogge matrix, which is the enumeration of [4] constraint propagation for the lower set of operators used in the first reference relationship against the upper set of operators used in the second reference relationship.

Algorithm 29 creates the second Hogge matrix, which is the enumeration of [4] con-

Algorithm 27 HoggeMatrixA ($\boldsymbol{N}$: IA network, T_r : target order per radar r, Q : target priority queue)

$\boldsymbol{H} = \varnothing$
for all $n \in \mathbb{I}\{1, \cdots, 127\}$ **do**
  for all $k \in \mathbb{I}\{1, \cdots, 127\}$ **do**
    $\boldsymbol{H}_{n,k} = \text{CP}\ (n, k)$
  end for
end for
return $\boldsymbol{H}$

Algorithm 28 HoggeMatrixB ($\boldsymbol{N}$: IA network, T_r : target order per radar r, Q : target priority queue)

$\boldsymbol{H} = \varnothing$
for all $n \in \mathbb{I}\{1, \cdots, 127\}$ **do**
  for all $k \in \mathbb{I}\{1, \cdots, 63\}$ **do**
    $\boldsymbol{H}_{n,k} = \text{CP}\ (n, k \cdot 128)$
  end for
end for
return $\boldsymbol{H}$

straint propagation for the upper set of operators used in the first reference relationship against the lower set of operators used in the second reference relationship.

Algorithm 30 creates the fourth Hogge matrix, which is the enumeration of [4] constraint propagation for using only the upper set of operators in both reference relationships.

Algorithm 31 uses the four Hogge matrices to compute the constraint propagation resulting from two known IA relationships. Each relationship is split into their upper and lower set as per the choices made for the IA operators defined in Section 3.3.2. Then if necessary, four lookups are performed into the four Hogge matrices. The result of each lookup is combined using the set union operator.

Algorithm 29 HoggeMatrixC ($\boldsymbol{N}$: IA network, T_r : target order per radar r, Q : target priority queue)

$\boldsymbol{H} = \varnothing$
for all $n \in \mathbb{I}\{1, \cdots, 63\}$ **do**
  for all $k \in \mathbb{I}\{1, \cdots, 127\}$ **do**
    $\boldsymbol{H}_{n,k} = \text{CP}\ (n \cdot 128, k)$
  end for
end for
return $\boldsymbol{H}$

Algorithm 30 HoggeMatrixD ($\boldsymbol{N}$: IA network, T_r : target order per radar r, Q : target priority queue)

$\boldsymbol{H} = \varnothing$
for all $n \in \mathbb{I}\{1, \cdots, 63\}$ **do**
  for all $n \in \mathbb{I}\{1, \cdots, 63\}$ **do**
    $\boldsymbol{H}_{n,k} = \text{CP } (n \cdot 128, k \cdot 128)$
  end for
end for
return $\boldsymbol{H}$

Algorithm 31 HoggeCP ($\boldsymbol{R_1}$: first IA relation, $\boldsymbol{R_1}$: second IA relation)

$T = \emptyset$
$low_1 = R_1 \wedge 127$
$low_2 = R_2 \wedge 127$
$high_1 = \lceil \frac{R_1}{127} \rceil \wedge 63$
$high_2 = \lceil \frac{R_2}{127} \rceil \wedge 63$
if $low_1 \neq 0 \wedge low_2 \neq 0$ **then**
  $T = T \cup \boldsymbol{H_a}_{low_1, low_2}$
end if
if $low_1 \neq 0 \wedge high_2 \neq 0$ **then**
  $T = T \cup \boldsymbol{H_b}_{low_1, high_2}$
end if
if $high_1 \neq 0 \wedge low_2 \neq 0$ **then**
  $T = T \cup \boldsymbol{H_c}_{high_1, low_2}$
end if
if $high_1 \neq 0 \wedge high_2 \neq 0$ **then**
  $T = T \cup \boldsymbol{H_d}_{high_1, high_2}$
end if
return T

B.3.3 The IG-MS algorithm (growing IA network)

The details of the Interval Algebra Growing-network Mechanically steered multistatic surveillance radar network Scheduling (IG-MS) algorithm documented here are an expanded version of Section 5.2.3. Referring back to the I-MS algorithm, the IA network is created using an initialisation algorithm. That is for both the radars in the scene an interval was created to represent the tracking task for each of the targets. Next, the intervals for both radars were added to the IA network along with the known constraints. The constraints came from the azimuth angle scan ordering in which the targets would be illuminated by the radars. This results in an IA relationship matrix that has two times the number of intervals as per targets.

The approach taken in this algorithm was instead to start off with an empty IA net-

work. An interval is added for both the radars for the target selected for multistatic measurements. Thereafter the two intervals are set equal to each other using the equals relationship {e}. Next, the constraints to all other intervals already in the IA network is populated by consulting the azimuth angle ordering of targets. This is done for each of the intervals created for each of the individual radar measurement tasks.

This approach then has the benefit that IA path consistency is not applied over an IA network that is large on every iteration. Instead, as the IA network grows, the computation required by IA path consistency gradually increases. This leads to a performance gain in the total computation time as IA path consistency is run on a matrix that is smaller than the matrix of I-MS on each iteration. Only once all targets have been added for multistatic measurements, does IA path consistency need to process a fully populated IA relationship matrix. The pseudo code is presented as Algorithm 32.

B.3.4 The IS-MS algorithm (simplified IA constraints set)

The Interval Algebra Simplified-constraints Mechanically steered multistatic surveillance radar network Scheduling (IS-MS) algorithm makes use of constraint ordering to simplify the set of constraints used [148]. Constraint ordering was first mentioned in Section 5.2.4. Here a more detailed description of how constraint ordering was applied to scheduling is given.

Constraint ordering

Constraint ordering is a technique which determines the order in which intervals are added to the IA network. Each of the IA relationships associated with that interval are given a weight using a scoring algorithm. The best approach to use is a scored based on the impact that it may have on the IA network. Several scoring algorithms can be devised [148].

As the IA relationship is composed of up to 13 IA operators, each IA operator can potentially be given a different score. The final score of the IA relationship is the sum of each of the scores of the IA operators that are present in the relationship. In the weighted constraint ordering, the score is determined by how each IA operator will restrict the allowable operators in the rest of the IA network [148]. The IA operator that has the most

Algorithm 32 IAGrowingNetworkScheduler ($\boldsymbol{A}$: target order for all radars, Q : target priority queue)

```
create IA network N = ∅, create schedule S = ∅
for all targets n ∈ Q select n randomly according to priority do
  for all radars r do
    p_{r,n} = bin of n in A_r
  end for
  set consistency c = true, create IA network M = N
  for all radars r do
    add row and column q_r to M for target n measured by radar r
    for all radars a ≠ r do
      c = c ∩ PC (M, [q_r, q_a], {e})
    end for
  end for
  for all intervals i already in N do
    for all radars r do
      p_{r,i} = bin of i in A_r
      if p_{r,i} < p_{r,n} then
        R = {b}
      else if p_{r,i} = p_{r,n} then
        R = {all}
      else
        R = {bi}
      end if
      c = c ∩ PC (M, [i, q_r], R)
    end for
  end for
  if c then
    S = {S, n}
    N = M
  end if
end for
return schedule S
```

weight in this scoring algorithm is the equals operator. The weighted constraint ordering algorithm provides the best improvement to the IA algorithms.

Constraint ordering is used to reduce the amount of processing time required by IA path consistency. It reduces the processing time by adding the intervals whose relationships have the most impact on the network first. This is due to the fact that relationships with a higher score will result in multiple updates on each cycle, thereby requiring more iterations of IA path consistency. Thus, it is better to have multiple cycles of IA path consistency while the network is still small as opposed to when it is much larger.

Simplified constraint IA scheduling

In the I-MS algorithm, the IA operators required are 'before', 'after' and 'equal'. According to the weighted constraint ordering [148], the 'equal' operator has the highest impact on the IA network. The 'before' and 'after' operators have a medium impact on the IA network. Thus, to utilise the ordering constraints it would be best to add the 'equal' operators first. However, the scheduling algorithm currently uses these to make decisions about multistatic measurements. Thus, it is not possible to use weighted constraint ordering directly.

Alternatively, all the target tracking tasks (or intervals) could be added to the network with only their 'equal' operator relationships set. Then as targets are selected for multistatic measurements their azimuth ordering constraints are applied to the IA network. If the network becomes inconsistent, the target tracking tasks that caused the inconsistency would be removed. In this way, the IA network would slowly be pruned down to the feasible list given the selections made. However, this approach would again lead to a maximum size matrix over which IA path consistency must be applied. Thus it does not lead to an algorithm that executes more efficiently than the IG-MS algorithm.

The approach taken was to remove the need for the 'equal' constraints altogether. This was done by instead having a list of multistatic measurement tasks for the radar network. Thus, there is only one interval in the IA network for every target. The matrix size is immediately reduced to half the required size of the preceding IA algorithms.

The radars still provide a relationship for each of the intervals, which are determined by the azimuth angle ordering the targets will be illuminated during the next scan. When the target is selected for a multistatic measurement, the IA relationships from both the radars are first combined using set intersection. The interval for the target is then added to the IA network using the IA relationship that results from the set intersection. If the network remains consistent the target is scheduled, otherwise the target is excluded. All targets are attempted in the order of their priority and randomly for equal priority.

This algorithm also makes use of Hogge [59] constraint propagation. Algorithm 33 represents the pseudo code for the operation of the IS-MS algorithm.

Algorithm 33 IASimpleConstraintsScheduler ($\boldsymbol{A}$: target order for all radars, Q : target priority queue)

```
create IA network N = ∅, create schedule S = ∅
for all targets n ∈ Q select n randomly according to priority do
  for all radars r do
    p_{r,n} = bin of n in A_r
  end for
  set consistency c = true, create IA network M = N
  add row and column q to M for target n measured by radar r
  for all intervals i already in N do
    R = {all}
    for all radars r do
      p_{r,i} = bin of i in A_r
      if p_{r,i} < p_{r,n} then
        R = R ∩ {b}
      else if p_{r,i} = p_{r,n} then
        R = R ∩ {all}
      else
        R = R ∩ {bi}
      end if
    end for
    c = c ∩ PC (M, [i, q], R)
  end for
  if c then
    S = {S, n}
    N = M
  end if
end for
return schedule S
```

B.3.5 The RP-MS algorithm (vectorized IA network)

The Reduced Point Algebra Mechanically steered multistatic surveillance radar network Scheduling (RP-MS) algorithm was the final optimised IA algorithm that achieved similar performance to GRASP in Chapter 5. For this IA scheduling algorithm the IA network was collapsed to a vector of interval sets. It was found that for the set of constraints used this leads to a sufficient representation. This algorithm is a further optimisation of the IS-MS algorithm of Section B.3.4. The details of the algorithm given here are an expansion of the contents of Section 5.2.5.

Reduced PA representation

Following on from the optimisations of the IS-MS algorithm, the IA networks formed were analysed. It was found that using only the 'before', 'after' and 'equal' IA operators, the entire IA network could be represented as a vector of interval sets. The order of the intervals within the vector would denote their temporal ordering. However, each element of the vector is a set of intervals. The intervals of this set have an ambiguous relationship, and thus their relationships contain the set {b, bi, e}.

Using this notion, the path consistency algorithm reduces to simply finding the insertion index in the vector or simply the set to which the interval belongs. There is no longer a need for using constraint propagation as once the location in the vector is found, the relationship to all the other intervals is known.

These findings are only true when using a subset of the IA operators similar to those above. In the scheduling algorithms used here there are only two distinct operators, in this case 'before' and 'equal'. Each also has one inverse 'after' and 'equal'. Similar findings would be made for the other IA operators when using one operator, its inverse operator along with the 'equals' operator. They also hold if none of the containment IA operators are used ('start', 'during', 'finish', 'started by', 'contains' or 'finished by') or the overlap IA operators ('overlap' and 'overlapped by').

Using only the pure temporal ordering IA operators ('before', 'meets', 'equals', 'after', and 'met by') means that the temporal sequence of the intervals is either well known or ambiguous (Figure 3.10). If they are well known then their ordering is evident from their location in the vector and can be further emphasised by placing the IA relationship in the IA network vector. If there is a local ambiguity, this ambiguity only affects the two intervals which are ambiguous, as long as their relationships to all other intervals is well known.

This type of behaviour is present in MSMSRN scheduling as typically targets can be ordered in azimuth angle for one of the scanning radars. Since as per the IS-MS algorithm, the IA relationships for both the radars are combined using set intersection, the most constrained IA relationship will dominate. Thus, unless the azimuth ordering of the target is uncertain for both radars, the resulting IA relationship from the set union

operator will not be ambiguous.

Targets that have an uncertain azimuth order for both of the radars in the geometry described in Section 3.2.3 would have to be very closely spaced within a small kite quadrilateral. Thus only a cluster of targets within this polygon will have an uncertain azimuth order for both radars. So for sparsely populated scenes most targets would have a defined azimuth order and only a small number would have an uncertain azimuth order unless the targets were all closely clustered. The ordering of intervals for targets within these clusters does not matter in the vector form of IA. While there may be multiple clusters that are adjacent, the ordering of targets between the clusters of targets with uncertain azimuth order will always be discernible.

Reduced PA scheduling

Given the details of the previous discussion, the RP-MS algorithm then only needs to maintain a vector representing the IA network entirely. Targets are selected for multistatic measurement by the two radars as per the I-MS algorithm. Then for each target in the list of those to be measured in the multistatic mode, the IA relationships to the selected target from the two radars are combined using set intersection. If the set intersection results in an empty set at any point then the target is discounted from being measured in the multistatic mode and a new target is selected. Alternatively, the target is added to the list of multistatic measurements. The pseudo code for the algorithm is repeated from Section 5.2.5 as Algorithm 34.

Algorithm 34 ReducedPAScheduler ($\boldsymbol{n}$: reduced PA network, $\boldsymbol{A}$: target order for all radars, Q : target priority queue)

```
create schedule S = ∅
n = RpaNetGrow (n, A, Q)
for all targets t ∈ n in sequence do
   S = {S, t}
end for
return schedule S
```

B.4 Scheduling using GRASP

There is only one algorithm listed in this section, which is the Greedy Randomised Adaptive Search Procedure Mechanically steered multistatic surveillance radar network Scheduling (G-MS) algorithm. This algorithm makes use of the Greedy Randomised Adaptive Search Procedure (GRASP) to perform a global search on the scheduling solution space. The algorithm consists of multiple iterations and makes use of a Restricted Candidate List (RCL) to build solutions. At the begining of an iteration, a candidate solution is built using a heuristic construction algorithm by consulting the RCL. Next, a local search is performed, where the candidate solution is altered slightly using elements not contained in the candidate solution but present in the RCL. At the end of the local search, only the best solution is maintained as the candidate solution for this iteration. Finally, this is checked against all solutions from prior iterations to determine the best solution found.

B.4.1 The G-MS algorithm (GRASP scheduling algorithm)

The G-MS algorithm was compared to both the II-MS algorithm and the IRP-MS algorithm in Chapter 5. This scheduling algorithm makes use of GRASP to generate and pick candidate solutions to the scheduling for the next scan. The GRASP algorithm for scheduling MSMSRN is documented in Section 3.4.

B.5 Scheduling using IA and GRASP

The algorithms listed in this section make use of GRASP to perform global searches. However, the construction and mutation algorithms make use of IA. Two algorithms were implemented, one using IA and the other making use of the reduced PA algebra defined in Section 5.2.5.

B.5.1 The II-MS algorithm (iterative IA scheduling)

The II-MS algorithm is the iterative form of the I-MS algorithm. It was used in Chapter 5 to compare to the G-MS algorithm. It provides equivalent performance in scheduling solution as per the G-MS algorithm but requires more execution time. The extra computation is caused by the rich set of constraints of IA.

Briefly, the algorithm creates an initial solution using the I-MS algorithm at the start of each iteration of GRASP. Furthermore, local searches are performed by first adding a random target to the IA network that was not part of the initial solution. Thereafter, targets are randomly attempted starting with those that formed part of the initial solution and followed by all the remaining unselected targets.

Local search scheduling

Up until this point, each of the IA algorithms defined have performed a local search of the scheduling solution space. Each begins by selecting a target for measuring in the multistatic mode at random. This randomly selected target will always be added to the IA network, as initially the network is empty. Therefore, the first selection reduces the solution space to only those solutions that contain this target.

All IA scheduling algorithms will then generate a maximal length scheduling solution given the random order in which the targets are selected. This is because all targets that satisfy the scheduling constraints will be added. Each target that gets added in sequence further constrains the solution space. Thus, while the solution is guaranteed to be as long as possible, the IA scheduling algorithms only find a locally optimal solution.

Global search scheduling

The GRASP algorithm attempts to find a globally optimal solution by searching the solution space using multiple iterations of local searches. Thus, to compare the IA scheduling algorithms fairly with GRASP scheduling, they needed to be adapted to perform a global search. For the II-MS algorithm this was achieved by using the I-MS algorithm as the construction and mutation algorithms for GRASP scheduling. Now on each iteration an initial solution is first found using the I-MS algorithm.

Each time a solution is generated it is tested against the best solution found. Naturally, the first solution generated will initialise the best solution. The GRASP fitness algorithm is used to generate a score for both solutions, where the best solution has the highest score. The GRASP fitness algorithm is defined in Section B.4.

During the local search a target not already part of the initial solution is added to the IA network first. Thereafter, targets that occur in the initial solution are randomly selected and added to the IA network. Finally, all remaining targets are attempted to ensure that the solution is as long as possible. Only those targets that do not violate constraints imposed by the previously added targets will result in a consistent IA network. These targets represent the schedule of targets to measure using multistatic measurements.

The best solution found after all iterations is used as the final scheduling solution. The pseudo code for this algorithm is presented as Algorithms 35 and 36.

Algorithm 35 IterativeIAScheduler ($\boldsymbol{A}$: target azimuth ordering per radar, Q : target priority queue, *iterations* : integer)

```
create best schedule B = ∅
for i = 1 → iterations do
  create current schedule C = ∅
  create IA network N = ∅
  C = IAScheduleRun (N, A, Q)
  C = IALocalSearch (A, Q, C)
  B = GraspUpdateSolution (Q, C, B)
end for
for all radars r do
  GraspCheckSolution (B, A_r)
end for
return B
```

Algorithm 36 IALocalSearch ($\boldsymbol{A}$: target azimuth ordering per radar, Q : target priority queue, S : schedule)

create best schedule $B = S$
create max fitness $m = \text{GraspFitness}(Q, S)$
for all $t \in Q, t \notin S$ select randomly according to priority **do**
  create mutated schedule $C = \{t\}$
  create IA network $\boldsymbol{N}_{1,1} = \{\text{e}\}$ for target t
  create priority queue $P = Q \forall\, t \in S$
  set mutated schedule $C = C \cup \text{IAScheduleRun}\ (\boldsymbol{N}, \boldsymbol{A}, P)$
  set mutated schedule $C = C \cup \text{IAScheduleRun}\ (\boldsymbol{N}, \boldsymbol{A}, Q)$
  set mutated fitness $f = \text{GraspFitness}\ (Q, C)$
  if $f >= m$ **then**
    set best schedule $B = C$
    set max fitness $m = f$
  end if
end for
return $\boldsymbol{s}$

B.5.2 The IRP-MS algorithm (iterative Reduced PA scheduling)

The Iterative Reduced Point Algebra Mechanically steered multistatic surveillance radar network Scheduling (IRP-MS) algorithm is the iterative form of the RP-MS algorithm. The IRP-MS algorithm makes use of GRASP to perform global searches of the scheduling solution space. It was compared in Chapter 5 to the G-MS algorithm. The IRP-MS algorithm compares well to GRASP both in terms of performance and execution time.

The RP-MS algorithm is used to generate the initial solution at the start of an iteration. Thereafter, one target not part of the initial solution is randomly selected to form part of a new solution during the local search. Targets that are part of the initial solution are then randomly selected and attempted, where only targets that maintain consistency in the temporal constraints are added to the multistatic schedule. Finally, all targets that were not part of the initial solution are also attempted in a random order. The final schedule of targets will be the maximum length possible given the random selection of targets. The GRASP fitness function is used to test which solution is better and only the best solution is kept. This process is repeated until each target not part of the initial solution has been attempted as the first target in a new solution for the local search.

The final schedule of targets to measure using multistatic measurements is the best solution found after all iterations. The pseudo code is Algorithms 37 and 38.

Algorithm 37 IterativeRpaScheduler ($\boldsymbol{A}$: target azimuth ordering per radar, Q : target priority queue, *iterations* : integer)

create best schedule $B = \varnothing$
for $i = 1 \rightarrow iterations$ **do**
  create current schedule $C = \varnothing$
  create reduced PA network $\boldsymbol{n} = \varnothing$
  $C =$ ReducedPASchedule $(\boldsymbol{n}, \boldsymbol{A}, Q)$
  $C =$ RpaLocalSearch $(\boldsymbol{A}, Q, C)$
  $B =$ GraspUpdateSolution (Q, C, B)
end for
for all radars r **do**
  GraspCheckSolution $(B, \boldsymbol{A}_r)$
end for
return B

Algorithm 38 RpaLocalSearch ($\boldsymbol{A}$: target azimuth ordering per radar, Q : target priority queue, S : schedule)

create best schedule $B = S$

create max fitness $m =$ GraspFitness (Q, S)

for all $t \in Q, t \notin S$ select randomly according to priority **do**

  create mutated schedule $C = \{t\}$

  create reduced PA network $\boldsymbol{n}_1 = \{t\}$

  create priority queue $P = Q \,\forall\, t \in S$

  ReducedPASchedule $(\boldsymbol{n}, \boldsymbol{A}, P)$

  set mutated schedule $C =$ ReducedPASchedule $(\boldsymbol{n}, \boldsymbol{A}, Q)$

  set mutated fitness $f =$ GraspFitness (Q, C)

  if $f > m$ **then**

    set best schedule $B = C$

    set max fitness $m = f$

  end if

end for

return $\boldsymbol{s}$

Appendix C

Tracking and Information Fusion Algorithms

C.1 Singer Random-Walk Acceleration Model

The Singer random-walk acceleration model [156] transforms the target (f) state $\boldsymbol{x}_f$ by a Markov process, where $\boldsymbol{x}_f$ is given by:

$$\boldsymbol{x}_f = [p_x, p_y, v_x, v_y]' \tag{C.1}$$

where p_x is the first dimension of the Cartesian position (m), p_y is the second dimension of the Cartesian position (m), v_x is the first dimension of the Cartesian velocity (ms^{-2}) and v_y is the second dimension of the Cartesian velocity (ms^{-2}).

The equation for this Markov process is of the form:

$$a\left(k+1\right) = \rho_m a\left(k\right) + \sqrt{1-\rho_m^2}\sigma_m r\left(k\right) \tag{C.2}$$

where $\rho_m = e^{-\beta T}$, $\beta = \frac{1}{\tau_m}$, τ_m is the target manoeuvre time constant, σ_m is the target manoeuvre standard deviation, $r(k)$ is a zero-mean unit-standard deviation Gaussian distributed random variable and T is the sampling time interval.

For the case where $\tau_m \ll T$ an accurate prediction of the acceleration cannot be obtained, and this leads to the following two equations. The Singer filter state transition

matrix $\boldsymbol{A}_f$ for the target is:

$$\boldsymbol{A}_f = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{C.3}$$

where T is the sampling time interval. The Singer filter process noise covariance matrix $\boldsymbol{Q}_f$ for the target is:

$$\boldsymbol{Q}_f = 2\sigma_m^2\tau_m \begin{bmatrix} \frac{T^3}{3} & 0 & \frac{T^2}{2} & 0 \\ 0 & \frac{T^3}{3} & 0 & \frac{T^2}{2} \\ \frac{T^2}{2} & 0 & T & 0 \\ 0 & \frac{T^2}{2} & 0 & T \end{bmatrix} \tag{C.4}$$

where T is the sampling time interval. In all the simulations captured in this document, $\tau_m = \frac{T}{2}$ and σ_m is chosen independently for the simulation batch to be run.

C.2 Kalman Filter

Radar measurements are first converted to Cartesian using Equation C.5 and then a linear Kalman filter is applied. While this approach is suboptimal, it produces adequate results for surveillance radars. Typically, an Extended Kalman Filter is required to produce optimal estimations of Cartesian position when the filter is provided polar radar measurements. Polar radar measurements are converted to Cartesian coordinates,

$$\boldsymbol{y}_f = \begin{bmatrix} d_{r,f}\cos\left(\theta_{r,f}\right) & d_{r,f}\sin\left(\theta_{r,f}\right) \end{bmatrix} \tag{C.5}$$

where $d_{r,f}$ is the radial distance and $\theta_{r,f}$ is the angle of target f from radar r.

The Kalman filter is an extention of recursive Least Squares Error (LSE) for tracking with random dynamics [20], where k represents the iteration for the recursive function. Many papers make use of the form of the Kalman filter described here and the Kalman derivation given here is based on the paper by Bar-Shalom [62] and reworked from Blackman and Popoli's [20] book. Given the same definition of target (f) state, $\boldsymbol{x}_f$, and target state transition matrix $\boldsymbol{A}_f$ as per Section C.1, the state transition matrix over time can be written in a discrete-time Markov form as:

$$\boldsymbol{x}_f\left(k+1\right) = \boldsymbol{A}_f\boldsymbol{x}_f\left(k\right) + \boldsymbol{q}_f\left(k\right) + \boldsymbol{f}_r\left(k+1|k\right) \tag{C.6}$$

In Equation C.6, $\boldsymbol{q}_f$ is the zero-mean white Gaussian process noise with assumed covariance $\boldsymbol{Q}_f$ and $\boldsymbol{f}_r$ is the assumed deterministic process noise caused by the motion of the sensor platform r. For the simulations used $\boldsymbol{f}_r$ was assumed to be zero for all sensor platforms.

Then the radar measurements observe the target position and velocity as follows:

$$\boldsymbol{y}_r\left(k\right) = \boldsymbol{H}_r\boldsymbol{x}_f\left(k\right) + \boldsymbol{v}_r\left(k\right) \tag{C.7}$$

where $\boldsymbol{H}_r$ is the measurement matrix for the radar (r) and $\boldsymbol{v}_r$ is the zero-mean white Gaussian process noise caused by the radar when making measurements. All simulations

in this document used:

$$\boldsymbol{H}_r = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{C.8}$$

and the maximum value for $\boldsymbol{v}_r$ as:

$$\max\left(\boldsymbol{v}_r\right) = \begin{bmatrix} 30 \\ 30 \end{bmatrix} \tag{C.9}$$

Given all the above, and dropping all subscripts for ease of notation, the Kalman filter equations are given by:

$$\hat{\boldsymbol{x}}\left(k|k\right) = \hat{\boldsymbol{x}}\left(k|k-1\right) + \boldsymbol{K}\left(k\right)\left[\boldsymbol{y}\left(k\right) - \boldsymbol{H}\hat{\boldsymbol{x}}\left(k|k-1\right)\right] \tag{C.10}$$

$$\boldsymbol{K}\left(k\right) = \boldsymbol{P}\left(k|k-1\right)\boldsymbol{H}'\left[\boldsymbol{H}\boldsymbol{P}\left(k|k-1\right)\boldsymbol{H}' + \boldsymbol{R}\right]^{-1} \tag{C.11}$$

$$\boldsymbol{P}\left(k|k\right) = \left[\boldsymbol{I} - \boldsymbol{K}\left(k\right)\boldsymbol{H}\right]\boldsymbol{P}\left(k|k-1\right) \tag{C.12}$$

$$\hat{\boldsymbol{x}}\left(k+1|k\right) = \boldsymbol{A}\hat{\boldsymbol{x}}\left(k|k\right) + \boldsymbol{f}\left(k+1|k\right) \tag{C.13}$$

$$\boldsymbol{P}\left(k+1|k\right) = \boldsymbol{A}\boldsymbol{P}\left(k|k\right)\boldsymbol{A}' + \boldsymbol{Q} \tag{C.14}$$

In most cases, Equation C.12 should rather be of the form:

$$\boldsymbol{P}\left(k|k\right) = \left[\boldsymbol{I} - \boldsymbol{K}\left(k\right)\boldsymbol{H}\right]\boldsymbol{P}\left(k|k-1\right)\left[\boldsymbol{I} - \boldsymbol{K}\left(k\right)\boldsymbol{H}\right]' + \boldsymbol{K}\left(k\right)\boldsymbol{R}\boldsymbol{K}\left(k\right)' \tag{C.15}$$

as the gain calculation is seldom exact or needs tweaking to account for modelling inaccuracies compared to that of realistic radars.

In Equations C.10–C.15, $\boldsymbol{K}$ is the Kalman gain, $\boldsymbol{P}$ is the zero-mean Gaussian estimation error:

$$\boldsymbol{P}\left(k\right) = \mathrm{E}\left(\left[\boldsymbol{x}\left(k\right) - \hat{\boldsymbol{x}}\left(k\right)\right]\left[\boldsymbol{x}\left(k\right) - \hat{\boldsymbol{x}}\left(k\right)\right]'\right) \tag{C.16}$$

and $\boldsymbol{I}$ is the identity matrix. $\mathrm{E}\left(\cdot\right)$ is the statistical expectation function of a random variable. All other variables are defined the same as in Section C.1.

The measurement covariance is computed using Equations C.17 and C.18. The matrix

$\boldsymbol{B}$ uses the sampling time T which is assumed to be fixed:

$$\boldsymbol{B} = \begin{bmatrix} \frac{T^2}{2} & 0 & T & 0 \\ 0 & \frac{T^2}{2} & 0 & T \end{bmatrix}' \tag{C.17}$$

The final measurement covariance is the product of the square of the target acceleration variance, matrix $\boldsymbol{B}$ and the transpose of matrix $\boldsymbol{B}$,

$$\boldsymbol{Q}_f = \sigma_{a,f}^2 \left(\boldsymbol{B}\boldsymbol{B}' \right) \tag{C.18}$$

where $\sigma_{a,f}$ is the assumed or estimated target acceleration variance.

C.3 Auction Algorithm

Kadar et al. [63] define an auction algorithm to assign observations to tracks of a sensor. This algorithm can also be used to associate tracks of sensors to each other. This section captures the essential details of the algorithm [20].

$$\left[\max_i \left(a_{ij} - P_i\right)\right] - \left(a_{i_j j} - P_{i_j}\right) \leq \epsilon \tag{C.19}$$

Equation C.19 can simply be interpreted as each track wants to pay the minimum price for an associated track.

Here a_{ij} is the margin by which the statistical distance d_{ij}^2 passes the gate G_{ij}. P_i is the price of a track and P_{i_j} is the price of assigning track j of sensor 2 to track i of sensor 1. The factor ϵ determines the maximal value for the assignment and is selected as:

$$\epsilon = \frac{1}{N+1} \tag{C.20}$$

for N tracks to be associated with.

The steps of the auction algorithm then are:

1. Initialise all tracks as not associated and all sensor 1 track costs P_i to zero.
2. Select a track from sensor 2 that is not associated. If none remain the algorithm completes.
3. Find the best track from sensor 1 i_j for each track of sensor 2 j.

$$a_{i_j j} - P_{i_j} = \max_{i=1,\ldots,n} \left(a_{ij} - P_i\right) \tag{C.21}$$

4. Mark the previously associated tracks as not associated and associate i_j with j.
5. Update the price of track i_j to a level at which j is almost optimally priced:

$$P_{i_j} = P_{i_j} + y_j + \epsilon \tag{C.22}$$

Here y_j is the difference between the best and second best assignment for track j.

6. Repeat from step 2.

This model is useful in simulations that require realistic manned targets.

C.4 Information Filter/Track Fusion

Chang et al. [64] gives an optimal form of the information filter (also known as the track fusion). The optimality is achieved by using the prior fusion, and can under certain conditions improve track fusion. The fusion equations for the case of two sensors is then:

$$\hat{\boldsymbol{x}}_c = \boldsymbol{P}_c \left[\boldsymbol{P}_i^{-1}\hat{\boldsymbol{x}}_i + \boldsymbol{P}_j^{-1}\hat{\boldsymbol{x}}_j - \boldsymbol{P}_{pi}^{-1}\hat{\boldsymbol{x}}_{pi} - \boldsymbol{P}_{pj}^{-1}\hat{\boldsymbol{x}}_{pj} + \boldsymbol{P}_p^{-1}\hat{\boldsymbol{x}}_p \right] \tag{C.23}$$

$$\boldsymbol{P}_c = \left[\boldsymbol{P}_i^{-1} + \boldsymbol{P}_j^{-1} - \boldsymbol{P}_{pi}^{-1} - \boldsymbol{P}_{pj}^{-1} + \boldsymbol{P}_p^{-1} \right]^{-1} \tag{C.24}$$

In Equations C.23 and C.24, $\hat{\boldsymbol{x}}_c$ is the current fused estimate of target state matrix and $\boldsymbol{P}_c$ is the current fused covariance matrix. Similarly, $\hat{\boldsymbol{x}}_p$ is the previous fused estimate of the target state matrix and $\boldsymbol{P}_p$ is the previous fused covariance matrix.

Next for the first sensor's track of the target, $\hat{\boldsymbol{x}}_i$ is the current estimate of the state matrix and $\boldsymbol{P}_i$ is the current covariance matrix. Similarly, $\hat{\boldsymbol{x}}_{pi}$ is the previous estimate of the state matrix and $\boldsymbol{P}_{pi}$ is the previous covariance matrix for the same track of sensor 1.

Finally, for the second sensor's track of the target, $\hat{\boldsymbol{x}}_j$ is the current estimate of the state matrix and $\boldsymbol{P}_j$ is the current covariance matrix. Similarly, $\hat{\boldsymbol{x}}_{pj}$ is the previous estimate of the state matrix and $\boldsymbol{P}_{pj}$ is the previous covariance matrix for the same track of sensor 2.

Appendix D

Contents of the Accompanying Disc

I. MIT

A. open course work

1. Contains useful open courses from MIT.

II. UCT

A. MSc Eng EEE

1. Conference Papers

a. Contains the LaTeX source code for the published FUSION 2011 conference paper [1].

2. Data Association

a. Literature used for data association research.

3. Dissertation

a. Draft Dissertation that was generated in 2011, and was intended to be submitted to UCT to fulfil the requirements of a M.Eng in Electrical Engineering degree. An upgrade to a Doctoral Degree was recommended after this dissertation was reviewed by Prof. M. R. Inggs and Dr J. P. de Villiers.

4. Journal Papers

a. Paper 1

(1) Contains the LaTeX source code for the draft versions of the Elsevier Information Fusion journal paper (Chapter 5) [2].

b. Paper 2

(1) Contains the LaTeX source code for the draft versions of the Elsevier Artificial Intelligence journal paper (Chapter 6).

5. Linux

a. Batch scripts that were used to execute simulations on the CSIR Insight Server.

6. Matlab

a. Simulation environment used to compare IA to GRASP, written in Matlab.

7. PhD Proposal

a. Contains the LaTeX source code for the Ph.D. proposal submitted to UCT in 2011.

8. Python

a. Contains prototype IA code for the parallel architecture investigation, written in the Python programming language.

9. Sensor Fusion

a. Literature used for sensor fusion research.

10. Sensor Scheduling

a. Contains the literature survey for multisensor scheduling as well as the results obtained when running the Matlab simulation environment.

11. Tracking

a. Literature used for radar tracking research.

B. PhD Eng EEE

1. Code

a. Contains IA code for the parallel architecture investigation, written in the C and OpenCL programming languages. This version is portable but was only tested on Ubuntu Linux LTS 12.04.

2. Conference Papers

    a. Contains the LaTeX source code for the submitted FUSION 2013 conference paper (Chapter 2).

3. Journal Papers

    a. Paper 1

        (1) Contains the LaTeX source code for the final version as well as the draft versions of the Elsevier Information Fusion journal paper (Chapter 5) [2].

    b. Paper 2

        (1) Contains the LaTeX source code for the final version as well as the draft versions of the Elsevier Artificial Intelligence journal paper (Chapter 6).

4. Literature

    a. Additional literature used during the author's Doctoral studies. Includes documents on multisensor management, Nvidia devices and OpenCL.

5. Matlab

    a. Simulation environment used to test nimble IA scheduling, written in Matlab.

6. MSVC++

    a. Contains IA code for the parallel architecture investigation, written in the C and OpenCL programming languages. This version can only be run using the Microsoft Windows operating system[1].

7. Results

    a. Contains the results obtained when running the nimble IA scheduling simulations.

8. Thesis

    a. Contains the LaTeX source code for this thesis document.

III. UP

[1]Microsoft Visual Studio 2010 and Microsoft Windows 7 was used

A. EIN732 – Introduction to Research

1. Useful course for any researcher beginning a masters or doctoral degree. The course is presented at UP by Prof. J. A. G. Malherbe The folder contains the three assignments completed for the course. This work formed the precursory work in the first year of the author's masters degree program.

· THE END ·