Interval Algebra – an Effective Means of Scheduling Surveillance Radar Networks

Richard W. Focke[a,b], J. Pieter de Villiers[a,c], Michael R. Inggs[b]

[a] *Council for Scientific and Industrial Research, Pretoria, South Africa, {rfocke, jdvilliers1}@csir.co.za*
[b] *University of Cape Town, Cape Town, South Africa, {richard.focke, michael.inggs}@uct.ac.za*
[c] *University of Pretoria, Pretoria, South Africa, {pieter.devilliers}@up.ac.za*

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 was done in terms of actual performance as well as in terms of computation time required. The 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 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.
Keywords:
Interval Algebra, multistatic radar, scheduling, surveillance, radar networks, resource management, process refinement

1. Introduction

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 [1, 2]. 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 [3], 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 are the sensor measurements. Recently, the updated Data Fusion Information Group (DFIG) data fusion model was proposed [3–5], 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.

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 [6]. 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 Markov decision processes (POMDPs) [7, 8] and multiple-armed bandits (MABs), a simplification of the Markov decision process (MDP) [9, 10]. 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 so as 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 [2, 11, 12]. 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 [13, 14], 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 [2], and can be sub-divided into three functions [15]. 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 work.

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.

1.1. 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 [2], Musick and Malhotra [15], as well as a more recent work by Ding [16]. 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 [17]. 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 [14, 18–20].

As research in the field broadens to encompass additional functions of sensor coordination, this aspect is sometimes handled intrinsically by task prioritisation algorithms [9, 21–24]. 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 [25]. These optimisation algorithms fall into two categories: mathematical programming [19, 26–28] and artificial intelligent search techniques [29, 30]. Simulation techniques are another possibility, where possible future timelines are investigated using multiple trials [31–33]. Sometimes random approaches are followed and these are typically used as an electronic countermeasure [34].

Artificial intelligence techniques have also been proposed in the past and have predominantly used reasoning/expert systems. Examples include fuzzy-set based reasoning [35, 36] and fuzzy decision trees [36] 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.

1.2. Multistatic radars

An interesting application of multisensor management for radars is the possibility of making multistatic measurements [37, 38], 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.

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. Receiving radars can always make a monostatic measurement by computing the monopulse angle of the target and the delay experienced by the signals it transmits. More sophisticated receiving radars can also discern the Doppler shift induced by the target motion.

For multistatic radars, where a radar can distinguish the signals of the other radars, this receiving radar can then also make a bistatic measurement. This is done by pinpointing the position of the target using intersecting ellipsoids with the receiving and transmitting radars as the ellipsoid foci. Thus, each radar is able to make more measurements of the target. Each radar has an increased probability of detection and can make more accurate measurements of the positional and kinematic attributes of the target.

1.3. 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 authors' conference paper [39], there appear to be no published papers that investigate the use of IA in information fusion.

At this point, it may be beneficial to refer to Appendix A for the theory of IA. 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 defined amount of time and starting at a defined 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.

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 [40].

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.

2. Material and Methods

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 the authors' previously published work [39]. For this comparison, the simulations also made use of a target priority queue so as to prioritise targets as depicted in Figure 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 in the previous work, the simulation environment makes use of Kalman

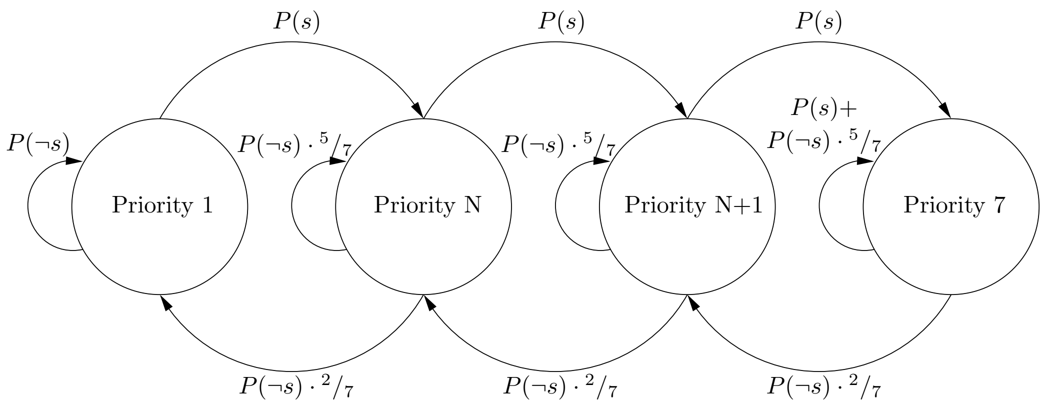
$P(s)$
$P(s)$
$P(s)$
$P(\neg s)$
$P(\neg s) \cdot {}^{5}/_{7}$
$P(\neg s) \cdot {}^{5}/_{7}$
$P(s)+$
$P(\neg s) \cdot {}^{5}/_{7}$
Priority 1
Priority N
Priority N+1
Priority 7
$P(\neg s) \cdot {}^{2}/_{7}$
$P(\neg s) \cdot {}^{2}/_{7}$
$P(\neg s) \cdot {}^{2}/_{7}$

Figure 1: Markov chain representing target priority

tracking filters [41]. The tracks of the radars are then fed to a central information fusion system. Here the tracks are associated using the auction algorithm for data association [42]. An information filter is then used to fuse the associated tracks [43]. 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.

2.1. 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 that is scanning a scene of N targets, the scheduling goal per scan can be written as:

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

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 1 but does not violate any constraints.

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

$$S(k) = \begin{cases} 1 & \text{if } k \in K, \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

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

$$C(m,n,r) = \begin{cases} \theta_r\left(m\right) \prec \theta_r\left(n\right) & \text{if } m < n, \\ \theta_r\left(m\right) \succ \theta_r\left(n\right) & \text{if } m > n, \\ \emptyset & \text{otherwise} \end{cases} \tag{3}$$

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.

2.2. Ambiguity in the sequence of targets

Ambiguities 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 [44]. Consequently, when targets are too close together in azimuth angle the radar is unable to discern their true order.

The mechanically steered multistatic surveillance radar network (MSM-SRN) scheduling algorithms must be able to exploit these ambiguities 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 2 illustrates a scenario where targets are measured ambiguously. Radar one does not need to discern the true order of targets two and five.

2.3. 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 2 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.

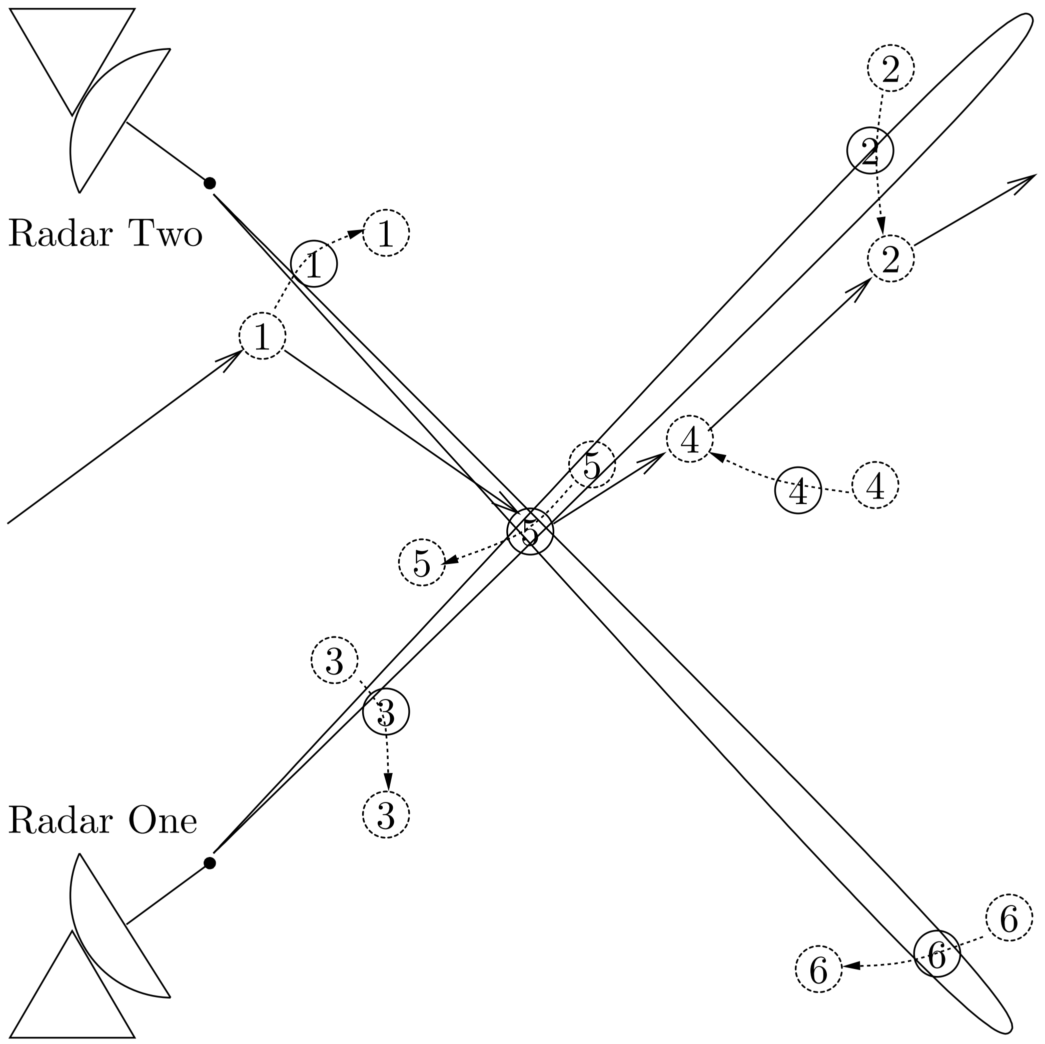

Figure 2: Ambiguity in the sequence of targets and target motion as scheduling proceeds

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. The shortest path is a straight line between the two targets and is shown in Figure 2 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 mulstistatic 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 a tracking radar 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

$$\theta_r(t_0) + t \cdot \dot{\theta}_r(t) = \arctan\left(\frac{\left(\mathbf{A}_f^t \cdot \hat{\mathbf{x}}_f(t_0) - \mathbf{x}_r\right) \cdot [0\,1\,0\,0]'}{\left(\mathbf{A}_f^t \cdot \hat{\mathbf{x}}_f(t_0) - \mathbf{x}_r\right) \cdot [1\,0\,0\,0]'}\right) \tag{4}$$

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

$$\bar{\mathbf{P}}_f(t) = \mathbf{A}_f^t \cdot \hat{\mathbf{P}}_f \cdot \mathbf{A}_f^{t\,\prime} + t \cdot \mathbf{Q}_f \tag{5}$$

where $\mathbf{A}$ is the motion model of the target, $\hat{\mathbf{P}}$ is the estimate of the tracking covariance, $\mathbf{Q}$ is the acceleration model of the target and $\bar{\mathbf{P}}$ is the prediction of the tracking covariance. However, as our simulation used discrete time, it was possible to search for the first potential solution.

2.4. Scanning, tracking and confirmation tasks

The multisensor management solution investigated here then uses a hybrid architecture consisting of two levels [13]. 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 mulstistatic 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.

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.

2.5. Implementation of scheduling algorithms

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

Table 1: Table demonstrating input format with ambiguities as multiple entries in a column

1	3	4	5	6	7
2					8
					9

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).

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 an ambiguous target azimuth ordered list.

Before a scan commences, each of the MSMSRN scheduling algorithms accepts two ambiguous target lists from the simulation environment, one list for each radar that scans over the area. An example of these lists can be found in Table 1, 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 ambiguous target lists once each interception point has been reached.

The MSMSRN scheduling algorithms select the targets measured in the

Table 2: Table demonstrating output format

1	2	5	7	8	9

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 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 1).

2.6. Choice of a comparison algorithm

For the comparison simulations it was important to compare the IA scheduling algorithm to another widely used algorithm. However, the focus of this research is not an exhaustive comparison but rather an introduction to IA for the purposes of multisensor scheduling. It includes a meaningful comparison of IA 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 optimisations, as these tend to lead to quicker convergence [2].

GRASP, as a meta-heuristic optimisation algorithm, seems to be a good representative algorithm for both of the first two choices. 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 [26]. This comparison ensures that IA is computationally feasible while achieving similar performance to that of a conventional scheduling algorithm. Interested readers can refer to Appendix B for the theory of GRASP.

Optimality is not achievable in general and is 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. Our 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.

3. Calculation

Previous work presented by the authors [39] compared IA to a heuristic algorithm that was devised to solve the MSMSRN 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, [26] 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 3.5.

3.1. Basic IA scheduling

As per Section 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:

$$I_{r,n}\{\text{before}\}I_{r,m} \mid \forall\,(r, n \neq m, T_r(n) \prec T_r(m)) \tag{6}$$

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'.

Algorithm 1 IASchedulerInit ($\boldsymbol{A}:$ target order for each radar)

```
create IA network $\boldsymbol{N} = \emptyset$
for all radars $r$ do
  create handled set $H = \emptyset$
  for all bins $b \in \boldsymbol{A}_r$ in sequence do
    for all targets $t \in \boldsymbol{A}_{r,b}$ in sequence do
      create interval $I_t$
      add row and column $t$ to $\boldsymbol{N}$ for $I_t$
      for all intervals $I_n$ in $H$ do
        if $n \in \boldsymbol{A}_{r,b}$ then
          PC $(\boldsymbol{N}, [n, t], \{\text{all}\})$
        else
          PC $(\boldsymbol{N}, [n, t], \{\text{before}\})$
        end if
      end for
      $H = \{H, I_n\}$
    end for
  end for
end for
return $\boldsymbol{N}$
```

$$I_{r,n}\{\text{equal}\}I_{s,n} \,|\, \forall\, r \neq s \tag{7}$$

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.

3.2. IA scheduling with ambiguities

IA can easily manage the case where the azimuth angle of the targets cannot be determined. As per Section 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:

$$I_{r,n}\{\text{before}\}I_{r,m} \,|\, \forall\, (r, n \neq m, n \in \boldsymbol{A}_{r,k}, m \in \boldsymbol{A}_{r,l}, k < l) \tag{8}$$

Algorithm 2 IASchedulerRun ($\boldsymbol{N}$: IA network, $\boldsymbol{A}$: target order for each radar, Q : target priority queue)

```
create schedule $S = \emptyset$
for all targets $n \in Q$ select $n$ randomly according to priority do
  for all radars $r$ do
    $\mathbf{p}_r$ = index of $n$ in $\boldsymbol{A}_r$
  end for
  set consistency $c$ = true
  for all unique combinations $a$ and $b$ $|a \neq b$ in index of $\mathbf{p}$ do
    $c = c \wedge \text{PC}\ (\boldsymbol{N}, [\mathbf{p}_a, \mathbf{p}_b], \{\text{equal}\})$
  end for
  if $c$ = true then
    $S = \{S, n\}$
  end if
end for
return schedule $S$
```

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

$$I_{r,n}\{\text{all}\}I_{r,m} \mid \forall\,(r, n \neq m, \{n, m\} \in \boldsymbol{A}_{r,k}) \tag{9}$$

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

The IA scheduling algorithm, which can handle the ambiguities in the target sequence, is presented as Algorithm 2. This algorithm requires using Algorithm 1 to initialise the IA network once per scan.

3.3. Growing the IA network

Algorithm 3 RpaNetGrow ($\boldsymbol{n}$: Reduced PA network, $\boldsymbol{A}$: radars' target order, Q : target priority queue)

```
for all $t \in Q$ select $t$ randomly according to priority do
  if $t \notin \boldsymbol{n}_k \mid \forall\, k$ then
    $\boldsymbol{n}$ = RpaNetAdd ($\boldsymbol{n}, \boldsymbol{A}, t$)
  end if
end for
return $\boldsymbol{n}$
```

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 [45], 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 3 captures the essence of the growing technique by only adding intervals as they are required.

3.4. Constraint ordering

Ordering heuristics could then be considered from the work of Van Beek and Manchak [45] to be applied to the IG-MS algorithm. Constraint ordering is described in Appendix A.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

Algorithm 4 RpaNetAdd ($\boldsymbol{n}$: Reduced PA network, $\boldsymbol{A}$: radars target order, t target number)

```
$d$ = length of vector $\boldsymbol{n}$
for all $s = 1 \to d$ do
  for all $m \in \boldsymbol{n}_s$ do
    $R = \{\text{all}\}$
    for all radars $r$ do
      $R_r$ = 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 $\boldsymbol{n}$
```

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.

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.

3.5. Reduced-point algebra networks

Algorithm 5 RpaGenRel ($\boldsymbol{A}$: target order, n : target one, m : target two)

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

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 intersection 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 PA 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 PA 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 3–5 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 ambiguous azimuth angle ordering from a radar r. 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 6.

Algorithm 6 ReducedPAScheduler ($\boldsymbol{A}$: target order for each radar, Q : target priority queue)

```
create reduced PA network $\boldsymbol{n} = \varnothing$
create schedule $S = \emptyset$
$\boldsymbol{n}$ = RpaNetGrow ($\boldsymbol{n}, \boldsymbol{A}, Q$)
for all targets $i \in \boldsymbol{n}$ in sequence do
    $S = \{S, n\}$
end for
return schedule $S$
```

3.6. Nimble IA scheduling

As discussed in Section 2.3, 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 4 for each target before building the list of ambiguous targets. In

Algorithm 7 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 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 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
$\mathbf{N} = \text{IASchedulerInit}(\boldsymbol{A})$
$S = \text{IASchedulerRun}(\mathbf{N}, \boldsymbol{A}, Q)$
send target $S_f$ with minimum $|\boldsymbol{\delta}_{f,r}|$ to radar schedulers
```

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 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 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.

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 7.

4. Results

4.1. Simulation set-up

For each of the comparison simulations[1] performed a 40×40 km area was selected, while a 10×10 km area was selected for the nimble simulations[2] 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 [13].

Three Monte-Carlo simulation batches were performed to analyse the performance of IA scheduling. The first two batches compared IA to GRASP,

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

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

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 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.

Table 3: Results comparing IA to GRASP by varying the number of targets in the scenario. Mean (μ) and standard error (ϵ), in factors of 10^{-2}, 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	6.0	2	6.1	2	6.2	1
S_I	5.9	3	6.0	2	6.1	1	6.2	1
M_G	3.7	7	5.5	7	8.3	7	13	8
M_I	3.7	7	5.5	7	8.3	7	13	8
E_G	5.2	4	18	8	66	20	260	60
E_I	4.9	3	16	7	60	20	240	60

Table 4: Results comparing IA to GRASP by varying the radar beam width of both radars simultaneously. Mean (μ) and standard error (ϵ), in factors of 10^{-2}, 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 $10^{-6\circ}$, 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
S_I	6.0	6.0	6.0	6.1	6.2	6.5	1.8
M_G	5.3	5.4	5.6	5.8	6.4	7.3	7.6
M_I	5.3	5.4	5.6	5.8	6.4	7.3	7.7
E_G	18	17	18	17	17	16	8.1
E_I	16	16	16	16	16	15	7.0

4.2. IA comparison to GRASP

4.2.1. Varying the number of targets

Refer to Table 3 for the results of the simulation batch that compared IA to GRASP when varying the number of targets.

4.2.2. Varying the beam width of the radars

Refer to Table 4 for the results of the simulation batch that compared IA to GRASP when varying the beam width of the radars.

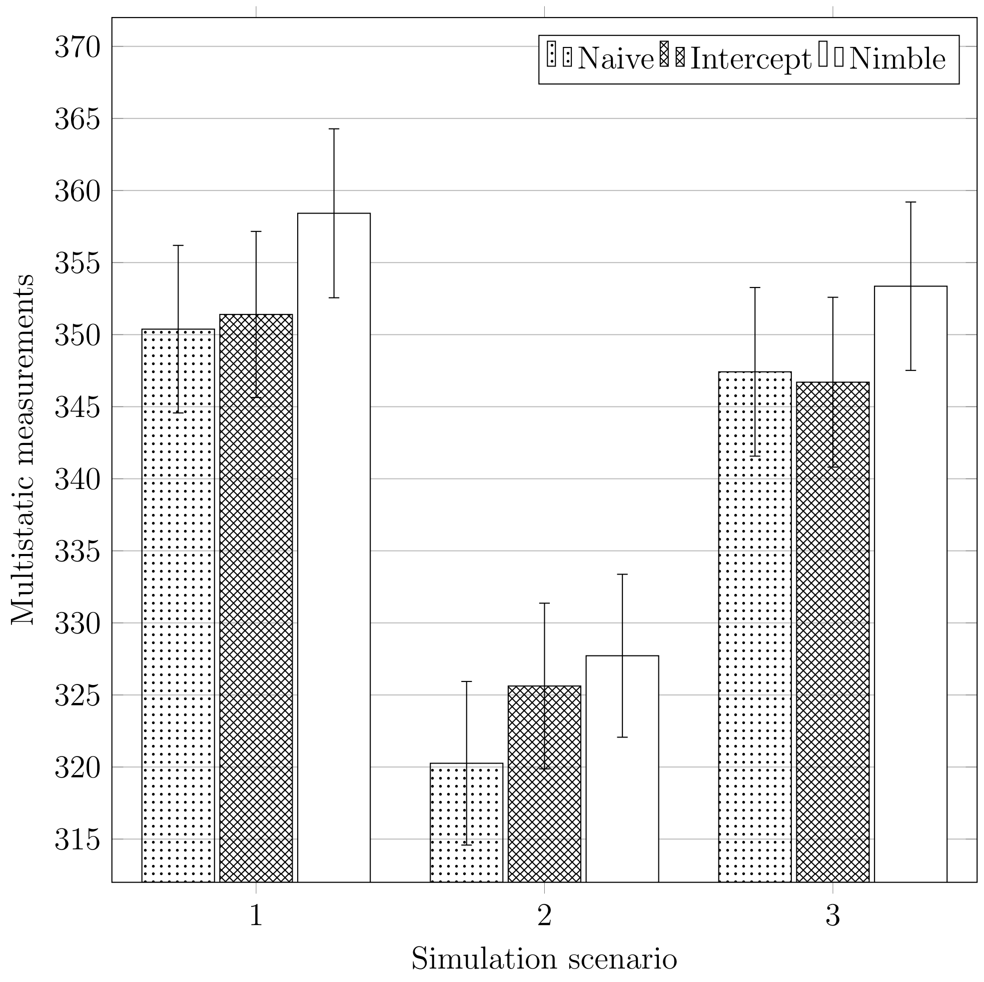

Figure 3: Effect of target initial velocity and acceleration on number of measurements made using IA scheduling. Average number of multistatic measurements made for 900 nimble IA scheduling Monte-Carlo simulation runs. Three different simulation scenarios were employed to investigate the effect of highly 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 3.6.

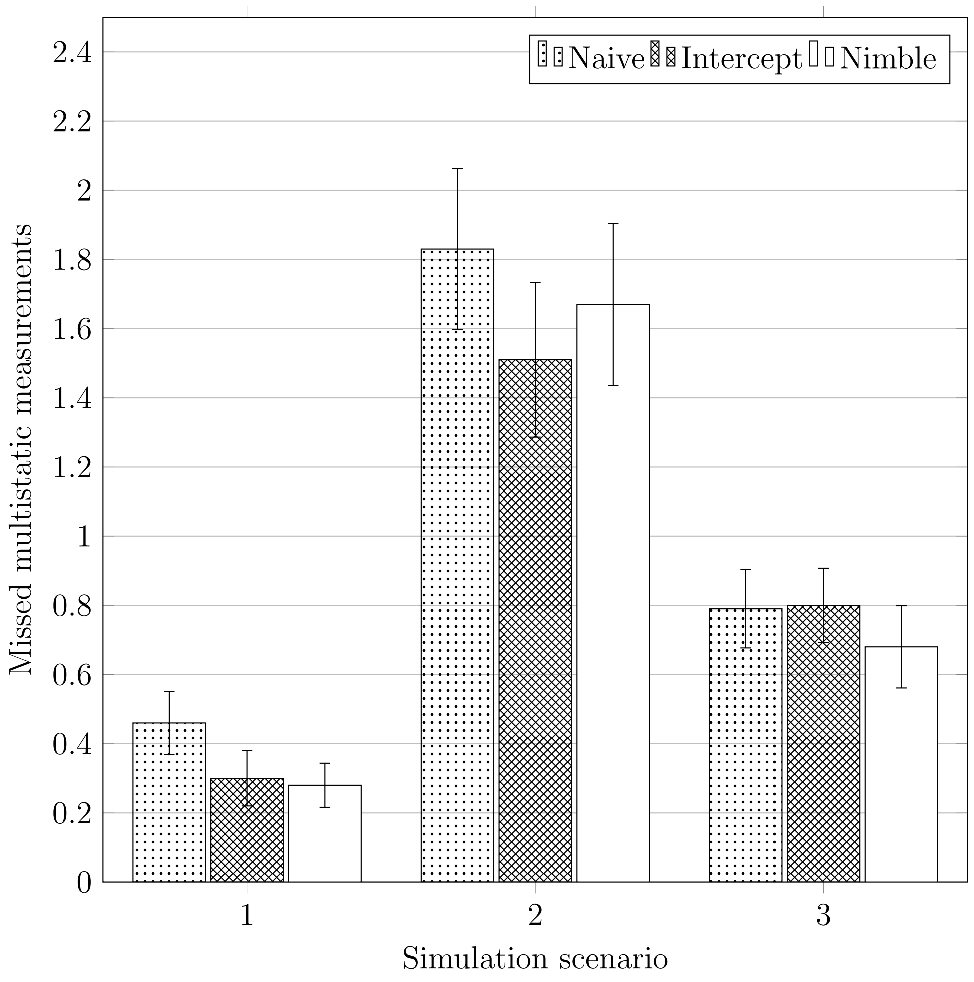

Figure 4: Effect of target initial velocity and acceleration on number of measurements missed using IA scheduling. Average number of multistatic measurements missed for 900 nimble IA scheduling Monte-Carlo simulation runs. Three different simulation scenarios were employed to investigate the effect of 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 3.6.

4.3. Nimble scheduling with IA

Refer to Figures 3 and 4 for the results of the nimble scheduling simulation batch.

5. Discussion

5.1. 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 [46].

5.1.1. 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

Table 5: 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

Table 6: Scheduling processing time versus number of targets

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

at the expense of more computations. Table 5 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 6.

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 multistatically 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.

5.1.2. 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 ambiguities in the radar target sequences. Both algorithms efficiently handled the ambiguities 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.

Reduced PA has some of the 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. Ambiguities 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 Reduced PA. Thus, more ambiguities did not require more processing time.

The slight decrease in processing time required can be attributed to the fact that as the number of ambiguities 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.

5.1.3. 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.

Reduced PA and any other form of IA will easily handle the addition of multiple radars. In the case of Reduced PA, 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.

5.1.4. IA network compression using Reduced PA

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 [45] 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.

Reduced PA became evident after considering the work of Van Beek and Manchak [45]. It also builds on the initial work of Allen [46], 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 Reduced PA 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 Reduced PA form would be applicable. For more complex IA relationships, especially those that consider various types of overlapping intervals, the Reduced PA 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 Reduced PA format. Alternatively, a surrogate interval could be used in the Reduced PA 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.

5.2. 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.

5.3. 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 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 CSP (DCSP) [40]. 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. Information gain

It is important for a multisensor manager to improve the measures of effectiveness of an information fusion system [47]. 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 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.

6. Conclusions

In this paper, Interval Algebra (IA) was compared to the Greedy Randomised Adaptive Search Procedure (GRASP) for scheduling mechanically steered multistatic surveillance radar networks (MSMSRNs). 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 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 field programmable gate arrays, 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 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[3]. Future work could consider 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 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 improves 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.

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

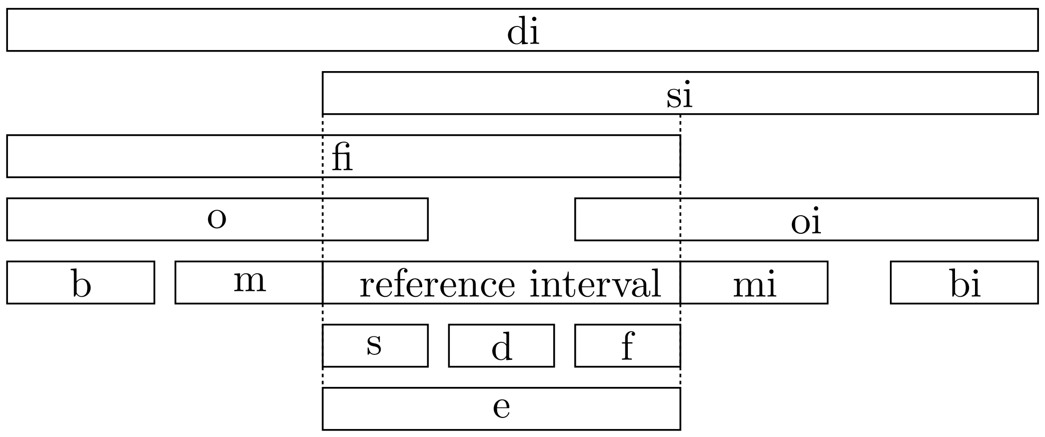

Figure A.1: Temporal ordering of intervals in IA

Appendix A. IA Theory

Appendix A.1. Allen's IA with improvements

The details of IA presented here is a rework of a conference paper [39], 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 [46]. 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.

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 A.1 depicts what each of the temporal operators means in isolation when referenced to a common reference interval. The suffix ‘i’ is used to associate 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 $\{\text{all}\}$ 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.

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.

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

```
$R_{A \to C} = \emptyset$
for all $n \in R_{A \to B}$ do
  for all $m \in R_{B \to C}$ do
    $R_{A \to C} = R_{A \to C} \cup \boldsymbol{T}_{n,m}$
    if $R_{A \to C} = \{\text{all}\}$ then
      return $\{\text{all}\}$
    end if
  end for
end for
return $R_{A \to C}$
```

Constraint propagation, or set composition for relationships, is captured as Algorithm 8, 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 [46], but these can also easily be determined by considering each case in isolation. 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 permissible operators for the unknown relationship.

Path consistency is presented as Algorithm 9, 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. Furthermore, path consistency can be used to alter a relationship in the IA network so as 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).

Algorithm 9 PC $(\boldsymbol{N}, [r,c], R)$

```
$\boldsymbol{N}_{r,c} = \boldsymbol{N}_{r,c} \cap R$
$\boldsymbol{N}_{c,r} = \text{Inverse}(\boldsymbol{N}_{r,c})$
add $[r,c]$ to set $L$
while $L \neq \emptyset$ 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 $(\boldsymbol{N}_{k,r}, \boldsymbol{N}_{r,c})$ then
        $R = \boldsymbol{N}_{k,l} \cap \text{CP}\ (\boldsymbol{N}_{k,r}, \boldsymbol{N}_{r,c})$
      end if
    else
      if not Skip $(\boldsymbol{N}_{r,c}, \boldsymbol{N}_{c,l})$ then
        $R = \boldsymbol{N}_{k,l} \cap \text{CP}\ (\boldsymbol{N}_{r,c}, \boldsymbol{N}_{c,l})$
      end if
    end if
    if $R = \emptyset$ then
      return false
    end if
    if $R \neq \boldsymbol{N}_{k,l}$ then
      $\boldsymbol{N}_{k,l} = R$
      $\boldsymbol{N}_{l,k} = \text{Inverse}\ (R)$
      add $[k,l]$ to $L$
    end if
  end for
end while
return true
```

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 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 10 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 10 is used by path consistency to skip computations that will yield {all} [45]. Algorithm 8 includes an improvement that stops computation if the resultant set becomes {all} [45]. Furthermore, Algorithm 9 can return the consistency of the algorithm whenever a set becomes $\emptyset$ [45, 48].

Appendix A.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 MiB 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 [49][4]. 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 $\times$ 128 matrix, which is populated with the results of set composition of all combinations of relationships that contain only

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

the lower operators.

2. Matrix two is a 128 $\times$ 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 $\times$ 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 $\times$ 64 matrix, which is populated with the results of 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 KiB 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.

Appendix A.3. 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 [45]. 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 [45], 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.

Appendix A.4. Fuzzy and Bayesian IA

As mentioned in the authors' conference paper [39], 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 [50–53]. 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 of 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 [50, 52], 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 [54, 55]. 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 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.

Appendix B. GRASP Theory

GRASP is a meta-heuristic search procedure that is comprised of three main steps, as per Algorithm 11:

1. Generate a reduced 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 the algorithm. The priority queue is explained in Section 2.

Algorithm 11 GraspSchedule (Q : target priority queue, $\boldsymbol{A}$: target order for all radars, k : iteration)

```
create best solution $\boldsymbol{b} = \varnothing$
**for** $i = 1 \to k$ **do**
    create candidate solution $\boldsymbol{c} = \varnothing$
    $\boldsymbol{c}$ = GraspConstruction ($Q, \boldsymbol{A}, \boldsymbol{c}$)
    $\boldsymbol{c}$ = GraspLocalSearch ($Q, \boldsymbol{A}, \boldsymbol{c}$)
    $\boldsymbol{b}$ = GraspUpdateSolution ($Q, \boldsymbol{c}, \boldsymbol{b}$)
**end for**
**for all** radars $r$ **do**
    GraspCheckSolution ($\boldsymbol{b}, \boldsymbol{A}_r$)
**end for**
**return** $\boldsymbol{b}$
```

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 an ambiguous cluster of targets 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 12, making use of Algorithms 13 and 14.
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 15, making use of Algorithms 13, 14, 12 and 16. When compared to genetic algorithms, the local search of GRASP can

be likened to the mutation operation used for genetic algorithms.

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 12 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 13. If the azimuth angle ordering of neither radar is violated the target will be added, which is checked in Algorithm 14. 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 13 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

Algorithm 12 GraspConstruction (Q : target priority queue, $\boldsymbol{A}$: target order for all radars, $\boldsymbol{s}$: scheduling vector)

```
for all $t \in Q$ select randomly according to priority do
  if $t \notin \boldsymbol{s}$ then
    for all radars $r$ do
      $N_r$ = GraspAddToSolution ($t$, $\boldsymbol{s}$, $\boldsymbol{A}_r$)
    end for
    if GraspFindMatchingSolutions ($N$, $\boldsymbol{m}$) then
      $\boldsymbol{s} = \boldsymbol{m}$
    end if
  end if
end for
return $\boldsymbol{s}$
```

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 13 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 $k = f \to l$ do
    $\boldsymbol{v} = [\boldsymbol{s}_{1\to k}, t, \boldsymbol{s}_{k+1\to \text{length}(\boldsymbol{s})}]$
    $R = \{R, \boldsymbol{v}\}$
end for
return $R$
```

Algorithm 14 GraspFindMatchingSolutions (S : set of integer vectors per radars, $\boldsymbol{m}$: scheduled target vector)

```
for all $\boldsymbol{s}_1 \in S_1$ do
  create match flag $f$ = true
  for all radars $r \neq 1$ do
    for all $\boldsymbol{s}_r \in S_r$ do
      if length $(\boldsymbol{s}_1)$ = length $(\boldsymbol{s}_r)$ then
        for $i = 1 \rightarrow$ length $(\boldsymbol{s}_1)$ do
          if $\boldsymbol{s}_{1,i} \neq \boldsymbol{s}_{r,i}$ 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
    $\boldsymbol{m} = \boldsymbol{s}_1$
    return true
  end if
end for
return false
```

Algorithm 14 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 15 performs a local search around the initial scheduling solution found by Algorithm 12. The algorithm receives three inputs: a priority queue of targets (Q), the 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 12) 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 16). 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 16 simply consults the target priority queue (Q) and sums all the priorities of all the targets present in the scheduled list of targets ($\boldsymbol{s}$).

Algorithm 15 GraspLocalSearch (Q : target priority queue, $\boldsymbol{A}$: target order for all radars, $\boldsymbol{s}$: scheduled targets vector)

```
$R = \{\boldsymbol{s}\}$
**for all** $t \in Q$ select randomly according to priority **do**
  **if** $t \notin \boldsymbol{s}$ **then**
    $\boldsymbol{v} = \varnothing$
    $M = \{t, k \in \boldsymbol{s}\}$
    **for all** $i \in M$ **do**
      **for all** radars $r$ **do**
        $N_r$ = GraspAddToSolution ($i$, $\boldsymbol{v}$, $A_r$)
      **end for**
      **if** GraspFindMatchingSolutions ($N$, $\boldsymbol{u}$) **then**
        $\boldsymbol{v} = \boldsymbol{u}$
      **end if**
    **end for**
    $\boldsymbol{v}$ = GraspConstruction ($Q, \boldsymbol{A}, \boldsymbol{v}$)
    $R = \{R, \boldsymbol{v}\}$
  **end if**
**end for**
$m = -\infty$
**for all** $\boldsymbol{k} \in R$ **do**
  $f$ = GraspFitness ($Q, \boldsymbol{k}$)
  **if** $f >= m$ **then**
    $\boldsymbol{s} = \boldsymbol{k}$
    $m = f$
  **end if**
**end for**
**return** $\boldsymbol{s}$
```

Algorithm 16 GraspFitness (Q : target priority queue, $\boldsymbol{s}$: integer vector)

```
$f = 0$
for $t \in \boldsymbol{s}$ do
    $p =$ priority of $t$ in $Q$
    $f = f + p$
end for
return $f$
```

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

```
$f1 =$ GraspFitness $(Q, \boldsymbol{s}_1)$
$f2 =$ GraspFitness $(Q, \boldsymbol{s}_2)$
if $f1 > f2$ then
    $\boldsymbol{b} = \boldsymbol{s}_1$
else
    $\boldsymbol{b} = \boldsymbol{s}_2$
end if
return $\boldsymbol{b}$
```

Algorithm 17 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 18 GraspCheckSolution ($\boldsymbol{s}$: integer vector, L : set of integer vectors)

```
$(i, j) = (0, 0)$
**for all** $s_k \in \boldsymbol{s}$ **do**
  find location $(r, c)$ of $s_k$ in $L$
  **if** $r < i$ **then**
    $\boldsymbol{s} = \varnothing$
    **return** false
  **end if**
  **for all** $s_l \in \boldsymbol{s} \mid k \neq l$ **do**
    **if** $s_k = s_l$ **then**
      $\boldsymbol{s} = \varnothing$
      **return** false
    **end if**
  **end for**
  $(i, j) = (r, c)$
**end for**
**return** true
```

Algorithm 18 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.

Acknowledgment

The authors would like to thank all the reviewers of the draft manuscripts for their detailed critique. It is not often that authors receive such excellent feedback and genuine interest. We would also like to thank Ana M. Huenchuquir Mardones, Larissa Focke and Walter W. Focke for their detailed reviews. This document was also proofread by Beverlie and Linton Davies of Stylus. Finally, we must thank the Council for Scientific and Industrial Research, the University of Cape Town and the University of Pretoria for without whose funding this research would not have been possible.

R. W. Focke acknowledges the hard work and sacrifices his wife had to endure during his studies; thanks to Adré Synman, Andre Le Roux, Adrian Adams, Casper van Zyl, Leon O. Wabeke, Magdaleen Vierbergen, Premilla Govender, Willie Fourie, Willie A. J. Nel and the co-authors for their support; and he acknowledges the South African Department of Science and Technology for their financial support. J. P. de Villiers acknowledges the Armaments Corporation of South Africa and the South African Department of Defence for their financial support. M. R. Inggs acknowledges the King Abdulasiz City for Science and Technology for their research support.

References

[1] G. W. Ng, K. H. Ng, Sensor management – what, why and how, Inf. Fusion 1 (2000) 67–75.

[2] N. Xiong, P. Svensson, Multi-sensor management for information fusion: issues and approaches, Inf. Fusion 3 (2002) 163–186.

[3] J. Llinas, C. Bowman, G. L. Rogova, A. N. Steinberg, E. Waltz, F. E. White, Revisiting the JDL data fusion model II, in: Proc. Int. Conf. Inf. Fusion, (2004), pp. 1218–1230.

[4] A. N. Steinberg, C. L. Bowman, F. E. White, Revisions to the JDL data fusion model, in: Proc. SPIE, vol. 3719, (1999), pp. 430–441. http://dx.doi.org/10.1117/12.341367. doi:10.1117/12.341367.

[5] E. Blasch, A. N. Steinberg, S. Das, J. Llinas, C. Chong, O. Kessler, E. Waltz, F. E. White, Revisiting the JDL model for information exploitation, in: Proc. Int. Conf. Inf. Fusion, (2013), pp. 129–136.

[6] A. O. Hero III, D. Cochran, Sensor management: past, present, and future, IEEE Sens. J. 11 (2011) 3064–3075.

[7] V. Krishnamurthy, D. V. Djonin, Optimal threshold policies for multivariate POMDPs in radar resource management, IEEE Trans. Signal Process. 57 (2009) 3954–3969.

[8] P. R. Barbosa, E. K. P. Chong, Adaptive sensing for target tracking in covert operations, in: Proc. SPIE Conf. Def. Secur., (2009), pp. 73360A–73360A–10. http://dx.doi.org/10.1117/12.818357. doi:10.1117/12.818357.

[9] S. Guha, K. Munagala, Approximation algorithms for partial-information based stochastic control with markovian rewards, in:

Proc. Annu. IEEE Symp. Found. Comput. Sci., (2007), pp. 483–493. doi:10.1109/FOCS.2007.23.

[10] H. Su, Q. Wang, K. Ren, K. Xing, Jamming-resilient dynamic spectrum access for cognitive radio networks, in: Proc. IEEE Int. Conf. Commun., (2011), pp. 1–5. doi:10.1109/icc.2011.5962525.

[11] 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, Int. J. Intell. Syst. 18 (2003) 279–307.

[12] C. Kreucher, K. Kastella, A. O. Hero III, Sensor management using an active sensing approach, Signal Process. 85 (2005) 607–624.

[13] S. Blackman, R. Popoli, Design and Analysis of Modern Tracking Systems, Artech House Radar Library, Artech House, Norwood, MA, 1999.

[14] A. R. Benaskeur, F. Rhéaume, Adaptive data fusion and sensor management for military applications, Aerosp. Sci. Technol. 11 (2007) 327–338.

[15] S. Musick, R. Malhotra, Chasing the elusive sensor manager, in: Proc. IEEE Natl. Aerosp. Electron. Conf., vol. 1, (1994), pp. 606–613. doi:10.1109/NAECON.1994.332850.

[16] Z. Ding, A survey of radar resource management algorithms, in: Proc. Can. Conf. Electr. Comput. Eng., (2008), pp. 001559–001564. doi:10.1109/CCECE.2008.4564804.

[17] G. Earl, B. Ward, Frequency management support for remote sea-state

sensing using the JINDALEE skywave radar, IEEE J. Ocean. Eng. 11 (1986) 164–173.

[18] S. L. C. Miranda, C. J. Baker, K. Woodbridge, H. D. Griffiths, Phased array radar resource management: a comparison of scheduling algorithms, in: Proc. IEEE Radar Conf., (2004), pp. 79–84. doi:10.1109/NRC.2004.1316399.

[19] R. Tharmarasa, T. Kirubarajan, J. Peng, T. Lang, Optimization-based dynamic sensor management for distributed multitarget tracking, IEEE Tran. Syst., Man, Cybern., Part C: Appl. Rev. 39 (2009) 534–546.

[20] 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: Proc. Int. Radar Conf., Surveillance for a Safer World, (2009), pp. 1–6.

[21] Y. Xun, M. M. Kokar, K. Baclawski, Control based sensor management for a multiple radar monitoring scenario, Inf. Fusion 5 (2004) 49–63.

[22] V. Krishnamurthy, Emission management for low probability intercept sensors in network centric warfare, IEEE Trans. Aerosp. Electron. Syst. 41 (2005) 133–151.

[23] H. Godrich, A. P. Petropulu, H. V. Poor, Power allocation strategies for target localization in distributed multiple-radar architectures, IEEE Trans. Signal Process. 59 (2011) 3226–3240.

[24] H. Aftab, N. Raj, P. Cuff, S. Kulkarni, Mutual information scheduling for ranking, in: Proc. Int. Conf. Inf. Fusion, (2011), pp. 1–8.

[25] Y. Cheng, X. Wang, M. Morelande, B. Moran, Information geometry of target tracking sensor networks, Inf. Fusion 14 (2013) 311–326.

[26] P. Z. Thunemann, R. Mattikalli, S. Arroyo, P. Frank, Characterizing the tradeoffs between different sensor allocation and management algorithms, in: Proc. Int. Conf. Inf. Fusion, (2009), pp. 1473–1480.

[27] Q. Ling, Y. Fu, Z. Tian, Localized sensor management for multi-target tracking in wireless sensor networks, Inf. Fusion 12 (2011) 194–201. Special Issue on Information Fusion in Future Generation Communication Environments.

[28] R. Tharmarasa, T. Kirubarajan, A. Sinha, T. Lang, Decentralized sensor selection for large-scale multisensor-multitarget tracking, IEEE Trans. Aerosp. Electron. Syst. 47 (2011) 1307–1324.

[29] 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: Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., (2008), pp. 2605–2610. doi:10.1109/IROS.2008.4651133.

[30] P. J. Shea, J. Kirk, D. Welchons, Adaptive sensor management for multiple missions, in: Proc. SPIE Conf. Def. Secur., (2009), pp. 73300M–73300M–12. http://dx.doi.org/10.1117/12.818892. doi:10.1117/12.818892.

[31] Y. He, K. P. Chong, Sensor scheduling for target tracking in sensor

networks, in: Proc. IEEE Conf. Decis. Control, vol. 1, (2004), pp. 743–748. doi:10.1109/CDC.2004.1428743.

[32] C. Kreucher, A. O. Hero III, K. Kastella, D. Chang, Efficient methods of non-myopic sensor management for multitarget tracking, in: Proc. IEEE Conf. Decis. Control, vol. 1, (2004), pp. 722–727. doi:10.1109/CDC.2004.1428735.

[33] S. Ahlberg, P. Hörling, K. Johansson, K. Jöred, H. Kjellström, C. Mårtenson, G. Neider, J. Schubert, P. Svenson, P. Svensson, J. Walter, An information fusion demonstrator for tactical intelligence processing in network-based defense, Inf. Fusion 8 (2007) 84–107. Special Issue on the Seventh International Conference on Information Fusion – Part II.

[34] K. Sycara, R. Glinton, B. Yu, J. Giampapa, S. Owens, M. Lewis, L. C. Grindle, An integrated approach to high-level information fusion, Inf. Fusion 10 (2009) 25–50. Special Issue on High-level Information Fusion and Situation Awareness.

[35] R. Popoli, S. Blackman, Expert system allocation for the electronically scanned antenna radar, in: Proc. American Control Conf., (1987), pp. 1821–1826.

[36] J. M. Molina López, S. J. Jimenez Rodríguez, J. R. Casar Corredera, Fuzzy reasoning for multisensor management, in: Proc. IEEE Int. Conf. Syst., Man., Cybern., vol. 2, (1995), pp. 1398–1403.

[37] V. S. Chernyak, Fundamentals of multisite radar systems: multistatic radars and multistatic radar systems, CRC Press, Boca Raton, FL, 1998.

[38] J. Li, P. Stoica, MIMO Radar Signal Processing, Wiley-IEEE Press, Hoboken, NJ, 2008.

[39] R. W. Focke, L. O. Wabeke, J. P. de Villiers, M. R. Inggs, Implementing Interval Algebra to schedule mechanically scanned multistatic radars, in: Proc. Int. Conf. Inf. Fusion, (2011), pp. 1–7.

[40] 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.

[41] Y. Bar-Shalom, Tracking and Data Association, Academic Press Professional, Inc., San Diego, CA, 1987.

[42] I. Kadar, E. R. Eadan, R. R. Gassner, Comparison of robustized assignment algorithms, Proc. SPIE 3068 (1997) 240–249.

[43] K. C. Chang, R. K. Saha, Y. Bar-Shalom, On optimal track-to-track fusion, IEEE Trans. Aerosp. Electron. Syst. 33 (1997) 1271–1276.

[44] S. M. Sherman, Monopulse principles and techniques, Artech House Radar Library, Artech House, Norwood, MA, 1984.

[45] P. Van Beek, D. W. Manchak, The design and experimental analysis of algorithms for temporal reasoning, J. Artif. Intell. Res. 4 (1996) 1–18.

[46] J. F. Allen, Maintaining knowledge about temporal intervals, Commun. ACM 26 (1983) 832–843.

[47] E. Blasch, P. Valin, E. Bosse, Measures of effectiveness for high-level fusion, in: Proc. Int. Conf. Inf. Fusion, (2010), pp. 1–8.

[48] P. van Beek, Reasoning about qualitative temporal information, Artif. Intell. 58 (1992) 297–326.

[49] 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.

[50] S. Badaloni, M. Falda, M. Giacomin, Integrating quantitative and qualitative fuzzy temporal constraints, AI Commun. 17 (2004) 187–200.

[51] S. Badaloni, M. Giacomin, The algebra $\mathrm{IA}^{\mathrm{fuz}}$: a framework for qualitative fuzzy temporal reasoning, Artif. Intell. 170 (2006) 872–908.

[52] S. Schockaert, M. D. Cock, Temporal reasoning about fuzzy intervals, Artif. Intell. 172 (2008) 1158–1193.

[53] L. Deng, Y. Cai, C. Wang, Y. Jiang, Fuzzy temporal logic on fuzzy temporal constraint networks, in: Proc. Int. Conf. Fuzzy Syst. Knowl. Discov., vol. 6, (2009), pp. 272–276. doi:10.1109/FSKD.2009.464.

[54] K. Zhang, A. Tradel, Efficient heuristics for solving probabilistic Interval Algebra networks, in: Proc. Int. Symp. Temporal Represent. Reason., (2006), pp. 111–120. doi:10.1109/TIME.2006.13.

[55] M. Mouhoub, J. Liu, Managing uncertain temporal relations using a probabilistic Interval Algebra, in: Proc. IEEE Int. Conf. Syst., Man and Cybern., (2008), pp. 3399–3404. doi:10.1109/ICSMC.2008.4811823.