Intelligent task scheduling in sensor networks

Wilbert van Norden[*,***] **Jeroen de Jong**[**] **Fok Bolderheij**[***] **Léon Rothkrantz**[*]

[*]Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Man-Machine Interaction group
P.O. box 5031, 2600 GA Delft, The Netherlands

[**]Thales Research & Technology
Delft Cooperation on Intelligent Systems
P.O. box 90, 2600 AB Delft, The Netherlands

[***]Royal Netherlands Naval College
Combat Systems Department
P.O. box 10000, 1780 CA Den Helder, The Netherlands

Abstract: *Ever more complex sensors have become available to create and maintain situational awareness during missions. Choosing the most suited sensor for the execution of a sensor function is based on sensor capabilities and function attributes. When these characteristics change rapidly, sensor allocation for sensor functions will shift. To increase performance of the entire sensor network, the total set of sensors should be scheduled in a single system. This paper puts forward and compares three new methods for scheduling prioritised tasks in sensor networks. The first is based on fuzzy Lyapunov synthesis. The other two use a genetic algorithm (GA) to optimise the set of schedules. The second scheduler uses this set to (re)train a neural network to be used online. The third approach is a novel online use of the GA. Tests showed that the novel online GA leads to a robust scheduling algorithm with high overall performance.*

Keywords: Sensor Management, Task Scheduling, Command & Control, Online Genetic Algorithms, fuzzy Lyapunov

1 Introduction

On the new Air Defence and Command Frigates of the Royal Netherlands Navy the APAR (Active Phased Array Radar) Multi Function Radar (MFR) is used for anti air and surface warfare. This system is able to perform multiple sensor functions like search, track and weapon guidance. Since more sensor functions can now be performed by a single sensor the underlying management structure has increased in complexity.

Looking for ways to improve MFR performance, it is obvious to look at the scheduling algorithm that decides which sensor task will be performed when (or which task should be dropped). This scheduling problem is known as the 'radar scheduling problem'. In light of the increasing interest in Network Centric Warfare (NCW), the properties of a scheduling algorithm for a single MFR should be extended to schedule sensor functions in a sensor network.

Currently, all tasks are generated by the combat management system and sent to the sensor chosen by the operator. Scheduling is done on the sensor level, using operator input to define the priority of sensor tasks. An undesired side effect of this way of scheduling is that sensor tasks might be dropped while they could have been executed by a less appropriate sensor. Making schedules for the entire set of sensors instead of optimising for each sensor separately seems to be a more fitting approach.

To do so, we propose three methodologies. The first is based on the scheduling algorithm by Margialot and Langholz [7]. They use fuzzy Lyapunov synthesis for solving a single machine job shop scheduling problem.

The other two algorithms are both based on a genetic algorithm (GA). We developed a new chromosome mapping and a fitness function enabling the use of genetic operators. The second algorithm uses the GA offline to find optimal solutions of past problems. These optimal solutions are used to (re)train a neural network (NN) that is used online for the actual scheduling. Such an approach results in an adaptive scheduler that combines the optimisation power of GAs and the speed of NNs.

The last scheduler is solely based on our GA. We developed a way to use the GA online without the setback of needing too much computation time.

Current schedulers and some developments in this field are discussed in section two. Section three of this paper describes a novel view on the command and control process used aboard navy ships. This view is expanded to the sensor scheduling level in section four. Using this approach, the scheduling algorithms can be discussed. Section five discusses the scheduler based on fuzzy Lyapunov synthesis.

Our genetic algorithm is discussed in section six. After this the NN approach and the online use are discussed in sections seven and eight. Some tests were done on all schedulers and are described in section nine. Finally, conclusions are drawn in section ten.

2 Related work

Much work has already been done in the field of radar (sensor) scheduling, especially since the introduction of the MFR. Some of these schedulers are based on a rotating MFR [2] and [4]. Scheduling tasks within a phased array MFR however is very different. Huizing and Bloemen proposed a scheduler based on queuing of dwell requests [5]. Our goal is to find a scheduling methodology that can be expanded to handle multiple sensors. For that reason this queuing strategy will not work directly.

Another new approach is to use a multi agent system for task scheduling, proposed by Thaens in [13]. Much work needs to be done on the negotiating scheme, rendering this approach not yet applicable for implementation in current sensor networks.

A promising scheduler is the OGUPSA algorithm, from McIntyre and Hintz, [8]. Most of the requirements for a good scheduler (like dealing with multiple sensors and real-time demands) are considered in their approach. A downside however is the sensor allocation: they look only at the ability of a sensor to perform a certain task, whereas the quality of the execution should also be considered.

To find a scheduler for the sensor network problem we therefore looked at another scheduling domain, namely the job shop scheduling problem. Algorithms used in that domain can be adapted to fit the sensor domain. For speed and optimality reasons we chose to implement a scheduler based on fuzzy Lyapunov synthesis (Margialot and Langholz [7]) and NN scheduling (Alifantis and Robinson [1]).

Since GAs are a strong optimisation tool in large search spaces they are chosen and adapted to generate the training data for the NN, an example of GAs in scheduling is e.g. Thilakawardana and Tafazolli [14].

3 Command & Control

Generally the command and control (C2) processes are modelled based on the well known OODA-loop. In [3], Bolderheij proposes to prioritise sensor tasks using risk estimation on threat objects. Both search and track tasks can then be scheduled based on the same principles. Introducing this concept in C2 leads to parallel processes instead of a loop, illustrated in Figure 1.

This new approach shows that sensor scheduling is only dependent on an ‘object store’ containing the available information on all detected and expected objects. All objects have a priority based on the risk they pose to the mission.

Figure 1 shows that the objects are the central point in the C2 process. All processes use some of the information held by the objects’ attributes update parts of it. The classification process for instance, uses the characteristics of an object to calculate the probability of this object belonging to certain

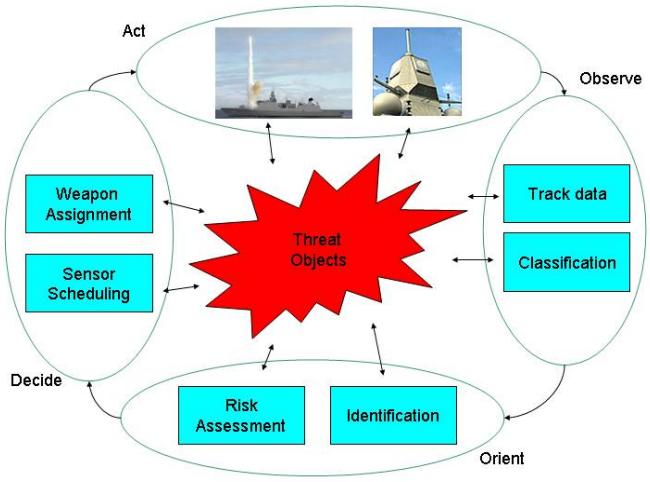

Figure 1 Command and control process aboard navy ships

class, see Mertens [9]. If the sensor manager notices that an object has a high uncertainty in classification, it can request sensor functions to update the relevant information about the object in order to help the classification process. This example shows that even though the different processes are decoupled, interdependency is maintained through the ‘object store’.

An important attribute of an object is the risk it poses with respect to the completion of a mission. Requests for sensor functions get a priority equal to the risk posed by the object the task refers to.

4 Sensor management

Since the C2 processes are considered as a set of parallel systems, we can look at the sensor scheduling in detail without looking at other processes. This leads to the resulting architecture of sensor network scheduling shown in Figure 2.

Optimising the different processes in Figure 2 one at a time, might not lead to the best picture compilation of the environment or to minimisation of risk during the mission. To reach an optimum the entire chain of resource allocation, task scheduling and dwell scheduling needs to be optimised. Dwell scheduling however is difficult to include since it is very sensor dependent and because the manufacturer of the sensor develops this part. We therefore only consider the combined problem of resource allocation and task scheduling. Since tasks are generated from the object store they are immediately available to the scheduler. The complexity of the ‘release date’ is therefore eliminated. The task attributes of importance to the scheduler are:

i) type of sensor function;
ii) priority;
iii) due date;
iv) duration;
v) position of the object to determine if it is in range for the sensors.

In order to develop an optimal scheduler we first look at the single MFR scheduling problem. If a successful scheduler for this type of sensor can be constructed we will extend this scheduler for a sensor network on a single ship. The ultimate goal, of course, is to schedule a distributed set of sensors, which is the case in NCW.

Scheduling a sensor network requires performance indicators to determine which sensor is best fitted for a sensor task at a given time. This is necessary for the single ship, multi sensor case as well as in NCW. By adding a range component the single ship problem can be expanded to a multi ship problem. Therefore the assumption can be made that this final expansion isn't very complex and that most problems will arise in expanding from single sensor to a sensor network.

5 Fuzzy Lyapunov

Figure 2 shows that the scheduling process for sensor(s) starts with a task request. This view on sensor scheduling can be compared with solving JSSPs. Margialot and Langholz, [7], introduced a scheduler based on fuzzy Lyapunov synthesis. This approach resembles a heuristic solution with the advantage of the stability offered by the Lyapunov synthesis.

The general idea here is to create different buffers for the different types of sensor functions. Whenever a sensor becomes available to execute a new task, the scheduler decides from which buffer the new task should come. To do so, it calculates the weight of the buffers (B_W) and then chooses the buffer with the highest weight. Since Margialot and Langholz assumed that all tasks have the same priority the formula used in [7] is unsuited for the domain of sensor scheduling. We therefore redefined the buffer weights as given in Eq. (1).

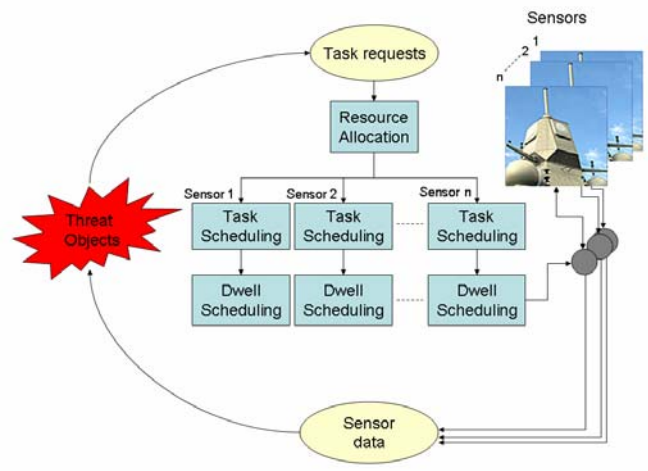

Figure 2 Architecture of control loop for a sensor network

$$B_W = k \cdot \sqrt{X} \cdot \sum_{\forall i} \rho_i P_i \left(\frac{(1-\alpha)\cdot \tau_i + \alpha}{\gamma \cdot \left(t_{dd,i} - t_{ct} - t_{td,i}\right)} \right) \qquad (1)$$

In Eq. (1) the following denotations are used:

k	:	Weight factor of the buffer;
X	:	Number of tasks in the buffer;
ρ_i	:	Range factor of object i, 1 if in range of the sensor else 0;
P_i	:	Priority of task i in the buffer, $0 \le P_i \le 1$;
τ_i	:	Relative processing time of task i in the buffer, $0 \le \tau_i \le 1$;
$t_{dd,i}$	:	Due date of task i in the buffer;
t_{ct}	:	Current time;
$t_{td,i}$	:	Task duration of task i in the buffer;
α	:	Weight factor to map relative processing time to a different domain[1], $[0,1] \rightarrow [\alpha,1]$, of course $0 \le \alpha \le 1$ holds;
γ	:	Weight factor for due date.

Extending the weight allocation from a single machine (single MFR) solution to multiple machines (sensor network) is relatively easy. The scheduler calculates the buffer weights whenever a sensor is available to receive a new task. By using the sensor specific values for k, τ and $t_{td,i}$ each sensor receives the task type for which it is most suited at that time.

After finding the right buffer, a task has to be selected from it. This mechanism is also straightforward: select the task that has the highest contribution to the buffer weight is selected. Since Lyapunov synthesis is based on stability analysis and we choose the task with highest contribution to this weight the number of tasks in the buffers is guaranteed to be limited.

6 Genetic Algorithm

Genetic algorithms are often used to find good solutions in search spaces that are too big to be searched exhaustively. Given the size of the search space[2] for scheduling in sensor networks GAs look promising. There are some issues when using a GA that need to be solved prior to applying GA for any problem.

[1] This makes it possible to make the priority more important than the required processing time.

[2] E.g.: 3 tasks on two sensors can be scheduled in 24 different ways, 4 tasks on two sensors in 120 ways

Two of these issues are inherent to using GAs: defining a fitness function and a chromosome mapping. The third is the speed of the algorithm. Although it doesn't take as much time as searching through the entire solution space, it is still computational complex, rendering it inapplicable for direct use in sensor scheduling due to the near-real-time requirements of sensor scheduling. Dealing with this last issue is discussed in sections six and seven.

6.1 Fitness function

Before we can use the GA for optimisation we have to define the fitness function C_F, Eq. (2). This function has to be maximised to find the optimal schedule. The downside of maximisation here is that the maximum is unknown, meaning that when we stop after a certain amount of generations we can't say how (sub) optimal the solution is.
The choice for maximisation was made since the goal is to execute as many tasks with high priority on the most suited sensor. A minimisation function could be made that should go to zero. However, it still couldn't be said how close to zero the most optimal schedule can get since reaching zero is not always possible.

$$C_F = \sum_{j=1}^{N} \sum_{i=1}^{n_j} \kappa_{j,i} \delta_{i,j} S_{j,i} P_i \left((1-\alpha) \cdot \tau_{i,j} + \alpha \right) \qquad (2)$$

In Eq. (2) the following denotations are used:

C_F	:	Chromosome fitness;
$\kappa_{j,i}$	:	Appropriateness of sensor j for task i, $0 \le \kappa_{j,i} \le 1$;
$\delta_{i,j}$	:	Feasibility of task i on sensor j, 1 if $t_{ct} + t_{td} < t_{dd}$ else 0;
$S_{j,i}$	:	Set-up factor in sensor j for task i, 1 if no set-up is needed else 0.8;
P_i	:	Priority of tasks i, $0 \le P_i \le 1$;
$\tau_{i,j}$	:	Relative processing time of task i on sensor j, $0 \le \tau_{i,j} \le 1$;
α	:	Weight factor to map relative processing time on a different domain, $[0,1] \rightarrow [\alpha,1]$, with $0 \le \alpha \le 1$;
n_j	:	Number of tasks for sensor j;
N	:	Number of sensors in the network.

6.2 Chromosome mapping

In order to be able to use genetic operators, the mapping between solution and chromosome should be well defined. Chromosomes should facilitate the use of operations like cross-over and mutation. After these operations the schedules embedded in the new chromosomes should still be valid. To obtain a good mapping we need to find a representation of the problem at hand. First we look at a representation for the single sensor problem. This mapping will then be extended to also map the multiple sensor case.

Assume that at a given time all task requests are placed in a single buffer with size *K* and the GA finds the optimal schedule for this buffer. A schedule can then be represented as a string of integers with values between 1 and *K*. In such a schedule each value can only appear once since each value in the schedule refers to a position in the buffer.

This mapping has a major advantage. Looking at the constraints placed on the representation of the schedule a good comparison can be made with the well known travelling salesman problem. Much research has already been conducted in solving this type of problem, so a mapping strategy already exists. The mapping we use is the swapping representation from De Jong [6].

This however solves only part of the mapping problem. In the multi sensor context we want to assign a sensor to certain tasks. This problem is solved by adding values to the schedule.

Consider ten tasks to be scheduled, $K = 10$, on three sensors. The schedule is mapped with swapping representation to chromosome *Chr*. For each sensor an element is added that says which part of *Chr* is to be executed by what sensor. The schedule, *S*, for the set of sensors is obtained by using the swapping representation denoted by $S = swapping(Chr)$. For multiple sensors, S_j denotes the schedule for sensor *j* with $S = \langle S_1 \oplus S_2 \oplus ... \oplus S_N \rangle$.

$$chromosome = \langle [4 \quad 3 \quad 3] \quad [Chr] \rangle$$
$$\begin{cases} S_1 = elements \quad 1 \rightarrow 4 \quad of \quad swapping(Chr) \\ S_2 = elements \quad 5 \rightarrow 7 \quad of \quad swapping(Chr) \\ S_3 = elements \quad 8 \rightarrow 10 \quad of \quad swapping(Chr) \end{cases}$$

This representation however isn't a good mapping since cross-over and mutation are still impossible since the added values must equal *K* when summed. Looking at the additional elements as weights rather than numbers solves this problem.

$$chromosome = \langle [2 \quad 7 \quad 4] \quad [Chr] \rangle$$
$$\begin{cases} S_1 = elements \quad 1 \rightarrow a \quad of \quad S \\ S_2 = elements \quad a+1 \rightarrow a+b \quad of \quad S \\ S_3 = elements \quad b+1 \rightarrow a+b+c \quad of \quad S \\ a+b+c \equiv K \end{cases}$$

$$a = floor\left(\frac{2 \cdot K}{2+7+4}\right) = floor\left(\frac{20}{13}\right) = 1$$

$$b = floor\left(\frac{7 \cdot K}{2+7+4}\right) = floor\left(\frac{70}{13}\right) = 5$$

$$c = floor\left(\frac{4 \cdot K}{2+7+4}\right) = floor\left(\frac{40}{13}\right) = 3$$

while $a+b+c < K$

add 1 *to smallest value*

end

$\rightarrow \quad a = 2 \quad b = 5 \quad c = 3$

This type of mapping enables cross-over and mutation. Since this mapping is highly dependent on the problem domain, the construction of new generations should be made to fit the problem.

Note that this is a simple mapping that works. Other mappings, e.g. the mapping used in vehicle routing problem (see [11]), might prove to result in a faster convergence to maximal chromosome fitness. We will not discuss other mappings since the goal here is only to give a conceptual proof of using GA in sensor scheduling.

6.3 Creating the new population

The new population is generated from the old population with the use of cross-over and mutation. The biggest part of the new population, 65%, is made by using cross-over on the chromosomes with highest fitness. On this part a mild form of mutation is applied. The next part is constructed by copying the fittest chromosomes from the old population into the new population. On the fittest chromosomes in this group no mutation is applied. On the others the elements for sensor allocation are mutated.

The final part of the new population is made by adding new randomly generated chromosomes. These random chromosomes build up 5% of the new population and are added to avoid local optima and to introduce new genetic material for cross-over in the next generation.

The values for these percentages were determined by tests and seemed to give good results. Before GAs can be used for sensor scheduling more tests should be done to optimise the construction of new populations. This however is also dependent on the optimal mapping solution.

7 Training a neural network

As mentioned in section five, the down side of a GA is the computation time needed to find optimal solutions. This problem can be solved by running the GA in the background and use its solutions to train a feed-forward NN, Figure 3.

To train a NN, we used the back propagation algorithm. Different network configurations can be used depending on user input. The number of in- and output nodes are of course defined by the size of the sensor network and the number of tasks in the buffer.

For the hidden layers the user can choose the number of nodes freely. The number of hidden layers can be chosen with a maximum of three. This choice was made since most practical systems never use more than three hidden layers, see [10]. The number of nodes and hidden layers can be chosen but the network structure is of course based on the general theories discussed in e.g. [12].

Figure 3 shows the general outline of the resulting scheduler. Incoming task request are placed in a buffer. This buffer is then scheduled by the NN. Over-training is prevented by using the GA offline to find optimal schedules. These solutions are used to retrain another NN that replaces the NN used online when it is fully trains. The result is a fast and adaptive scheduler.

8 Online use of GA

Using our GA for training NNs is not the only option to overcome the problem of time constraints. Online use of the GA was accomplished by using a buffering scheme similar to the one used in the Lyapunov based scheduler. Here only one buffer is used with a fixed size *K*. This buffer is filled with default tasks until task requests are received. These requests then replace default tasks in the buffer.

Whenever a sensor becomes available to execute a new task the scheduler looks at the current best solution given by the GA. It identifies the correct schedule for the sensor and only uses the first entry, meaning that only one task is sent to the sensor. Doing so means the buffer only changes slightly. Since the GA is constantly running, only few generations are needed to find a new optimal solution.

In this setting the GA does not have any stop criteria. We therefore can say nothing about the optimality of the schedule that is used. Although this might seem to be a large setback, it most likely is not due to the real-time demands on the system. In the military domain a sub-optimal solution in time is better than an optimal solution that is too late.

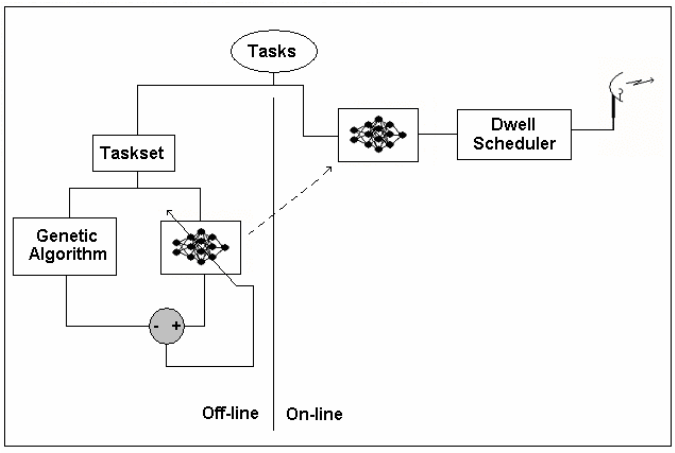

Figure 3 Architecture of the genetic-neural network

The solutions that are found should be as good as possible, given the available amount of computation time. The maximum fitness in consecutive populations should therefore increase as fast as possible. Some techniques for obtaining this result are:

i) injecting heuristic solution into the initial population;
ii) make the initial population large enough so all genetic material is available;
iii) use parallel processing to speed up the generation process;
iv) find optimal values for the mutation probabilities.

As already said in section five, the mapping could also be optimised leading to faster convergence of chromosome fitness.

9 Test results

To evaluate the three schedulers a test case was defined for a difficult scheduling problem. The problem was defined in such a way that not all requests can be scheduled. This causes the scheduler to drop certain tasks, which tasks are dropped can be used for evaluation purposes. In this simulation four sensor functions are defined: search, track, midcourse guidance (MG) and terminal illumination (TI). To make the problem large enough all the tasks have a long duration. A set-up time is required when switching to and from TI representing the time needed to switch from waveform generator in actual MFRs.

The scheduling was done for the single sensor situation and for the duration of one minute. It is assumed that all task requests are made at the beginning of the second in which they should be performed. The problem is illustrated in Figure 4 where the required time for all requested tasks is shown for each second. Since the single sensor problem is used, the available time is, of course, one second.

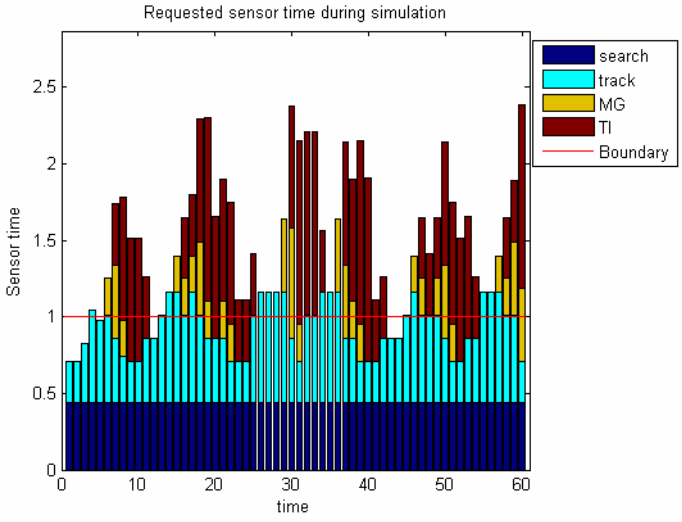

Figure 4 The simulated scheduling problem

To compare the different schedulers we used the utility as given in Eq. (3). This measure for scheduling efficiency was introduced by Thaens [13].

$$U = \frac{\sum_{\forall i} \delta_i P_i}{\sum_{\forall i} P_i} \tag{3}$$

In Eq. (3):

U : Utility;
δ : 1 if task is scheduled correctly, else 0;
P : Priority.

Before the scheduler based on fuzzy Lyapunov could be used for solving the simulation, tests were done to determine values for α and γ. The latter proved to be no factor of importance and was therefore set to 1. The test also showed that $\alpha = 0.8$ was a reasonable value (tests with our GA confirmed this value for Eq. (2) as well). These values are not surprising when looking at Eq. (1) and Eq. (3). The evaluation is based on the priorities of the executed tasks where α reduces the influence of the task duration with respect to the priority. A high value for α is therefore to be expected.

Table 1 Results of four different schedulers

Scheduler	Utility	Dropped budget requests (%)			
		Search	Track	MG	TI
FIFO	*0.5938*	*28.9*	*38.5*	*49.3*	*49.8*
Lyapunov	*0.6805*	*61.4*	*37.7*	*39.6*	*13.4*
GNN	*0.5730*	*19.0*	*42.3*	*47.6*	*46.6*
Online GA	*0.6720*	*48.5*	*41.3*	*49.0*	*10.9*

Since the scheduling problem is highly dependent on time, the utility in time is as important as the average utility. For each of the four schedulers the utility in time is given in Figure 5-8. The average utility in the simulated minute is given in table 1 for each of the three schedulers. For comparison the results of a First-In-First-Out (FIFO) scheduler are also given. In table 1 the percentage of sensor budget that is not executed is also given for each of the four sensor function types.

Looking at the results of the utility, the choice would be to use the scheduler based on Lyapunov synthesis. This scheduler shows a high average utility. The extreme points in time also show that it performs at least as good as the FIFO scheduler when the demand is higher then the available budget. An additional advantage in the military domain is the system predictability: for equal demand, the resulting schedule is the same.

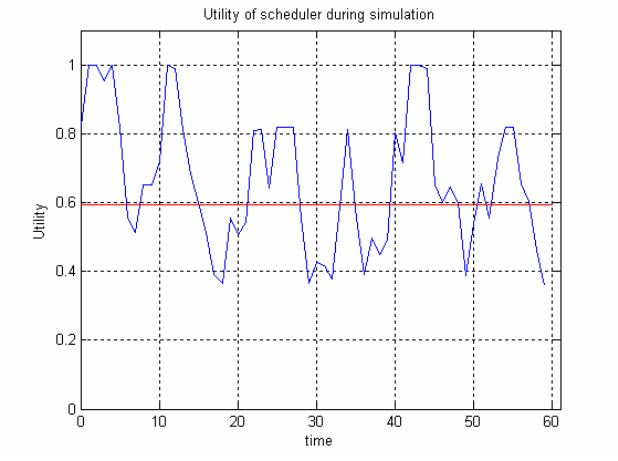

Figure 5 Utility of the FIFO-scheduler

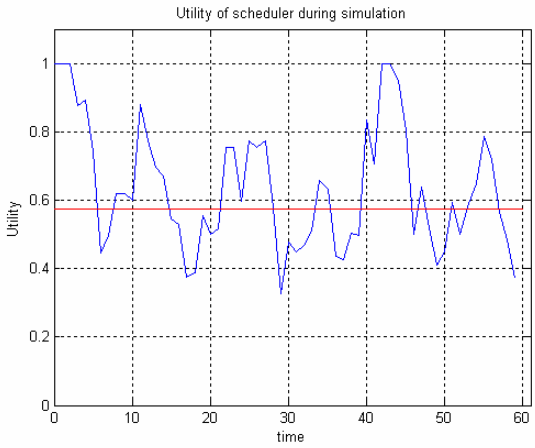

Figure 7 Utility of the GNN scheduler

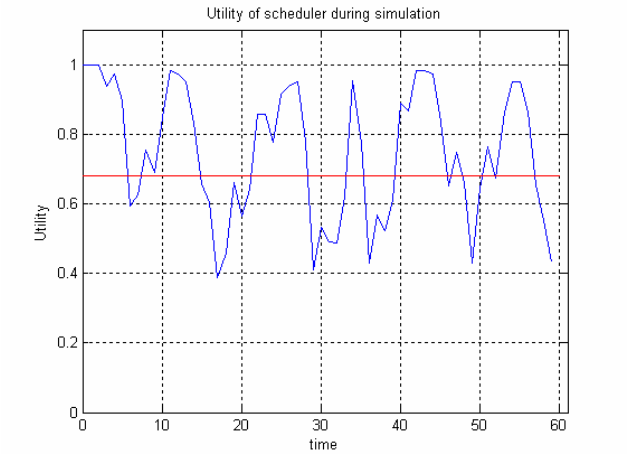

Figure 6 Utility of the fuzzy Lyapunov scheduler

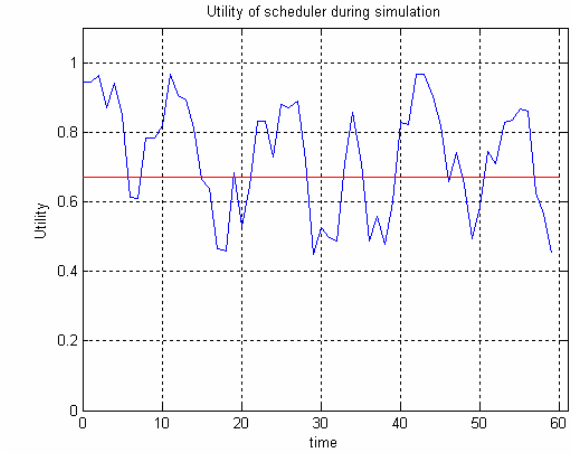

Figure 8 Utility of the online GA for scheduling

A scheduler that is perhaps faster due to fewer computations than the fuzzy Lyapunov based scheduler is the GNN. Our simulation however showed some downsides of this approach. In table 1 we can see that the scheduler drops more TI tasks than search tasks. In practise this means that some of our guided missiles will miss a target. Most likely this effect is due to the NN since the online GA doesn't have this side effect.

The online GA shows promising results. The utility is almost equal to that of the Lyapunov scheduler and it drops less TI tasks. In practise this means that more missiles will hit a target, thus minimising the risk to the mission.

Another advantage of our online GA is that the minimum utility in time is higher when compared to the Lyapunov scheduler. Again this means that it performs slightly better in difficult scheduling situations. This is probably due to the lower percentage of dropped TI tasks. A final remark for all schedulers is the high percentage of dropped MG tasks compared to track tasks. Since the first are of more importance we should consider adjusting our equations somewhat to ensure that MG tasks are dropped less. Furthermore, all three schedulers outperform the FIFO scheduler based on the percentage of dropped tasks.

10 Conclusions

This paper introduced three new scheduling mechanisms for task scheduling in sensor networks. All of those were based on a novel view of the C2 process.

The Lyapunov scheduler proved to be a fast scheduling algorithm that can be implemented relatively easily. The results showed that it can outperform the often used first-in-first-out heuristic. We also suggested that this scheduler can be expanded to accommodate a multi sensor system. An advantage of this scheduler over the others is that it is a predictable method.

Genetic algorithms are a very strong optimisation tool. Using this power in a dedicated parallel process renders it usable in an online scheduler. This results in a high performance scheduler that can be made to be at least as good as any scheduler available today due to the copying without mutation of the fittest chromosomes. The online setting however should be developed and tested further to improve the results.

The disappointing results of NN scheduling are probably due to the complexity of the sensor scheduling problem. Based on the training data no generalisation can be made causing the network to give scheduling results that are worse than the simple FIFO heuristic. A possible solution could be the use of more sophisticated training algorithms that enable the network to generalise better. Since the online implementation seems to work well this however might be unnecessary.

One setback of the GA however remains: a prediction about the optimality of the chosen solution can't be given. Injecting heuristic solutions in the initial population could be a way to guarantee a certain level of optimality.

Further work needs to be done in testing the schedulers in simulation environments. This would show their scheduling efficiency with respect to more realistic task sets. Another important test is to find the saturation points of the schedulers. This boundary indicates the possibility to track a certain number of objects while maintaining some search tasks as well.

Other work that has to be done is to optimise the chromosome mapping in the GA and fine tune its parameters. In Lyapunov some further research could be done on the calculation the buffer weight.

Acknowledgements

This research was conducted at the Delft Cooperation on Intelligent Systems (DECIS) laboratory. This lab is supported by Thales Research and Technology Netherlands, Applied Physics Laboratory (TNO), Delft University of Technology and the University of Amsterdam.

The new approach to the C2 process was developed as part of the STATOR research program supported by Thales Naval Netherlands, the Royal Netherlands Naval College and the International Research Centre for Telecommunications-transmission and Radar of the Delft University of Technology.

References

[1] Alifantis T. and Robinson S., 'Using simulation and neural networks to develop a scheduler advisor', *Proceedings of the 2001 Winter Simulation Conference*

[2] Barbato A., Giustiniani P., 'An improved scheduling algorithm for naval phased array radar', *Alenia Defense Systems, Italy*

[3] Bolderheij F. and Genderen, van P., 'Mission driven sensor management', *7th International Conference on Information Fusion, Stockholm, Sweden, 2004*

[4] Duron C. and Proth J.M., 'Multifunction radar: Task scheduling', *Journal of Mathematical Modelling and Algorithms 1:105-116, 2002*

[5] Huizing A.G. and Bloemen A.A.F., 'An efficient scheduling algorithm for a multifunction radar', *IEEE int. symp. on Phased Array Systems and Technology. 0-7803-3232-6/96, IEEE, 1996*

[6] Jong, de J.L., 'Different representations for the travelling salesman problem using genetic algorithms', *Vrije Universiteit Amsterdam*

[7] Margialot M. and Langholz G., 'Design and analysis of fuzzy schedulers using fuzzy Lyapunov synthesis', *Engineering Applications of Artificial Intelligence, 14:2, p183-188, 2001*

[8] McIntyre G.A., Hintz J.E., 'Sensor measurement scheduling: an enhanced dynamic preemtive algorithm', *Optical Engineering, 37:2, p517-523, 1998*

[9] Mertens B.G.M., 'Reasoning with uncertainty in the situational awareness of air targets', *TNO-FEL 2004, FEL03-S211, TNO The Hague, 2004*

[10] Negnevitsky M., *Artificial Intelligence; A guide to intelligent systems*, Person Education Limited, 2002

[11] Reeves C.R., *Modern heuristic techniques for combinatorial problems*, McGraw-Hill, 1995

[12] Russell S. and Norvig P., *Artificial Intelligence: A Modern Approach 2/E*, Pretence Hall, Pearson Education Inc., 2002

[13] Thaens R., 'Sensor scheduling using intelligent agents', *7th International Conference on Information Fusion, Stockholm, Sweden, 2004*

[14] Thilakawardana S. and Tafazolli R., 'Use of genetic algorithms in efficient scheduling for multi service classes', 2004, http://research.ac.upc.es/EW2004/papers/15.pdf