11TH ICCRTS
COALITION COMMAND AND CONTROL IN THE NETWORKED ERA

Centralisation *vs* Self-Synchronisation: An Agent-Based Investigation

Topics: Cognitive Domain Issues, C2 Modeling and Simulation, C2 Concepts and Organizations

Anthony H. Dekker
Defence Science and Technology Organisation
DSTO Fern Hill
Department of Defence, Canberra ACT 2600
Australia

Telephone: +61 2 6256 6151
Facsimile: +61 2 6256 6180

dekker@ACM.org

Paper I-030

Centralisation *vs* Self-Synchronisation: An Agent-Based Investigation

Anthony H. Dekker
Defence Science and Technology Organisation
Department of Defence, Canberra ACT 2600
dekker@ACM.org

Abstract

Much NCW literature expounds the benefits of self-synchronisation, but does not clearly delineate the circumstances in which self-synchronised planning is effective. We investigate this question using agent-based simulation. Since such investigation requires an objective assessment of plan quality, we have constructed an experiment in which agents attempt to solve three well-understood computational problems of increasing difficulty: the selection problem, the target assignment problem, and the "travelling general" problem. We compare performance with a centralised organisational structure (where one "command" agent constructs a plan for solving the problem, as in the construction of a Air Tasking Order) against a decentralised organisational structure (where agents self-synchronise by exchanging messages across a network). Specifically, we examine performance under varying levels of time pressure, and varying network quality. Analysis of the experimental results provides general guidelines as to when centralised command and control is most effective, and when self-synchronisation is a more appropriate alternative.

Keywords: agent, centralisation, self-synchronisation, NCW, simulation.

1. Introduction

Much NCW literature expounds the benefits of self-synchronisation instead of centralised command and control. However, this literature does not clearly delineate the circumstances in which distributed or self-synchronised planning is effective. In this paper, we investigate this question using agent-based simulation.

Such an investigation of planning strategies requires an objective assessment of plan quality, which is difficult to do in problems simple enough to be simulated. Agent-based combat simulations [12] have limited scope for including non-trivial planning. There is scope for planning when agents play games such as chess or checkers [13], but for those cases it is difficult to evaluate plan quality, other than by playing against a wide range of opponents.

We have therefore constructed a simulation environment of simple communicating agents as an abstraction of communication and synchronisation between military entities. Using this environment, we have conducted an experiment in which agents attempt to solve three well-understood computational problems of increasing difficulty:

- the selection problem;
- the target assignment problem; and
- the "travelling general" problem.

Since these problems are well understood, we can assess the performance of different planning strategies by relating the plans they produce to the best possible plan. We compare performance of a centralised organisational structure (where one "command" agent constructs a plan for solving the problem, as in the construction of a Air Tasking Order) against a decentralised or distributed organisational structure (where agents self-synchronise by exchanging messages across a network).

We examine the difference in performance between these two structures under varying levels of time pressure, and varying network quality.

2. Experimental Method

We use the simulation toolkit included within CAVALIER, a network analysis, visualisation, and simulation tool that we have developed. A number of agents are linked in a double-ring (prism) network, as shown in Figure 1.

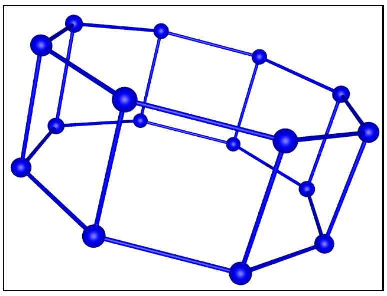

Figure 1. Double-Ring (Prism) Network Topology

The agents attempt to solve the three computational problems in two ways. One of the agents may be a "command" node, sending the solution to the other agents, or else all agents may be equal, developing a solution through self-synchronisation. We consider four levels of time pressure, requiring all agents to agree on the solution within:

- 10,000;
- 100,000;
- 1,000,000; or
- 10,000,000 time units.

We also consider four levels of network quality, with a latency (delay) on each network link of:

- 10;
- 100;
- 1,000; or
- 10,000 time units.

In addition, each agent has a handling time for each message which is 1% of the latency, so that considerable congestion may occur on the slower networks.

3. Results & Analysis: Selection Problem (Trivial Case)

The simplest problem we consider is the selection problem. Here one agent is to be assigned to a task, and the best-qualified agent for the task will be chosen. In real life, this might be the agent with the greatest supply of fuel or the largest number of available personnel. Here, each agent has a randomly assigned suitability score. The centralised process for this problem has three steps:

(a) Each agent sends suitability information to the "command" agent,
(b) The "command" agent chooses the most suitable agent.
(c) The "command" agent broadcasts its decision to the other agents.

We rate performance on this task as 1 if the process is completed, and 0 if not. Figure 2 shows the results for the centralised process involving 64 agents.

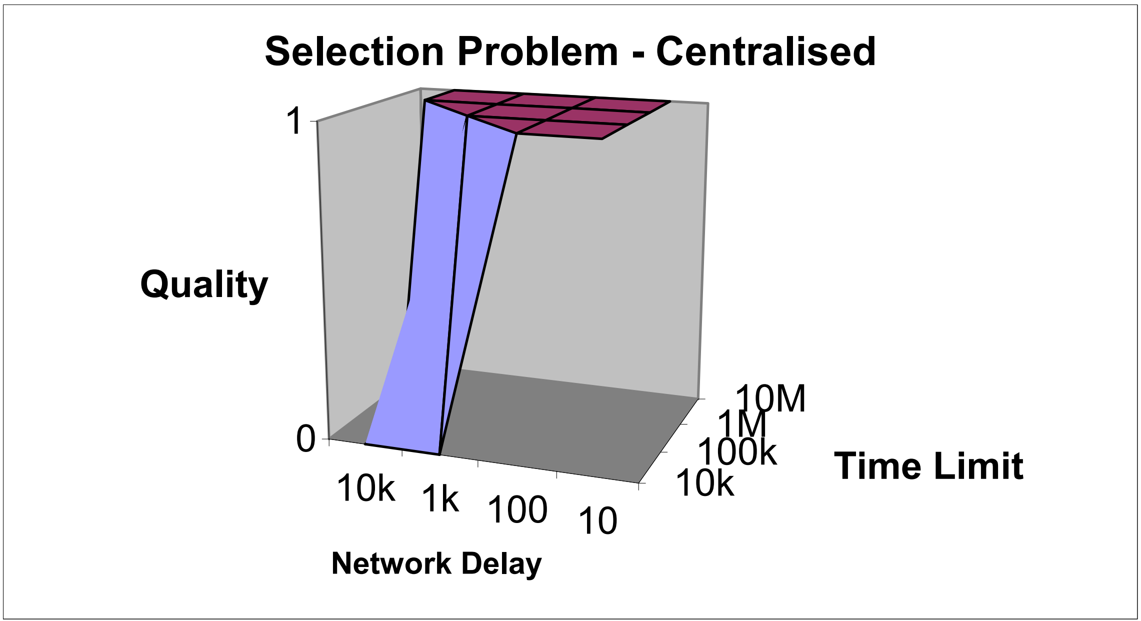

Figure 2: Performance for Centralised Organisation on Selection Problem

It can be seen that the front left corner of Figure 2 has three cases where the process fails, which correspond to a time limit less than or equal to ten times the network delay (latency):

- latency = 10,000 time units and time limit = 10,000 time units;
- latency = 1,000 time units and time limit = 10,000 time units; and
- latency = 10,000 time units and time limit = 100,000 time units.

Since the time for a message to travel to the "command" agent from the furthest agent in a 64-node double ring is about 17 times the network latency, it can be seen that these three cases do not leave enough time for step (a) to be completed.

For the self-synchronised or distributed process, the steps are:

(a) Each agent broadcasts suitability information to every other agent,
(b) Each agent independently makes the selection of the most suitable agent.

This decision-making strategy, which has been called the "borg" strategy [1,2], is effective for simple problems where each agent can be guaranteed to carry out step (b) in the same way (although it is not realistic for most real-world problems). For the selection problem, the process requires an agreed-upon tie-break rule for two equally qualified agents. We always break the tie by the agent with the numerically lowest internal identifier. There are other versions of distributed decision-making for the selection problem which achieve the goal with fewer messages [16].

Performance for our distributed process is the same as in Figure 2. However, simulating latency/time-limit combinations near the boundary of the failure region (for example, latency = 300 time units and time limit = 10,000 time units) shows that the distributed process can succeed where the centralised process fails. This is because the centralised process involves two communication steps, while the distributed process involves only one communication step.

Trivial problems like this one often occur at the technological level, for example in maintaining databases. The centralised approach corresponds to having a single centralised database, while the distributed approach corresponds to replicating databases in several nodes, and keeping them "in synch" with each other. There are well-understood techniques for doing this [3,4]. Where the centralised and distributed approaches work equally well, the distributed approach is often preferable for the military environment, because it has the advantages of **fault-tolerance** [4]: the distributed approach can continue if some agents are lost, while the centralised approach is vulnerable to the loss of the "command" agent.

4. Results & Analysis: Small Target Assignment Problem (Simple Case)

We next consider the target assignment problem. This involves a number of agents and targets, where each agent must be assigned to the most suitable target (in our case, the closest), as shown in Figure 3:

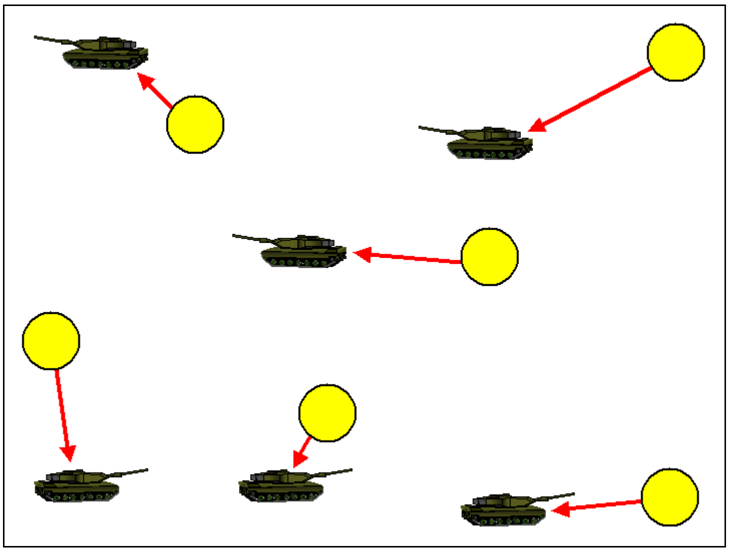

Figure 3: The Target Assignment Problem

For this experiment, we have 16 agents and 16 targets, There is a well-understood process for solving this problem, called the Hungarian algorithm [5], which takes between n^3 and n^4 steps, where n is the number of agents. There is also a faster method of solution, based on successively assigning the best available target for each agent in turn, and then looking for pairs of agents which can benefit from swapping their targets. However, this faster method may not always produce the best possible answer. The centralised process is:

(a) Each agent sends position information to the "command" agent,

(b) The "command" agent finds the best possible solution, given the available time.

(c) The "command" agent broadcasts its decision to the other agents.

Figure 4 shows the performance, averaged over 100 runs. The front left corner has a performance of 0, as in the selection problem. Most cases have a performance of 1, meaning that the best possible answer was obtained. The remaining two cases with a time limit of 10,000 show a performance of 0.99, calculated as the ratio of the sum of agent-to-target distances for the best possible answer compared to the answer actually obtained. In other words, an almost perfect answer was obtained in these two cases.

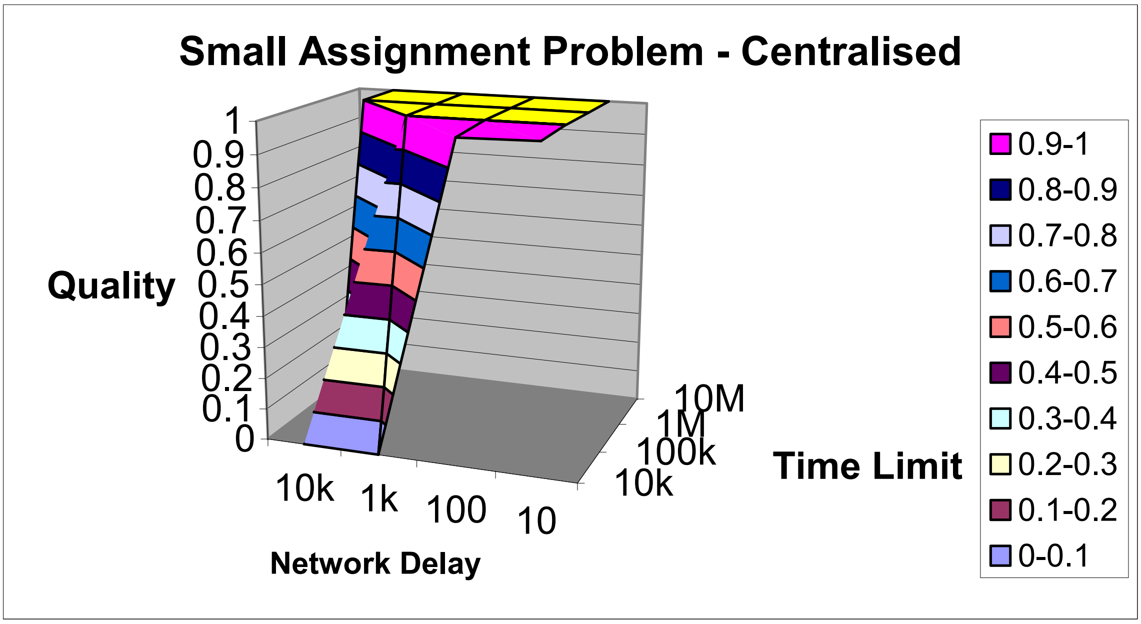

Figure 4: Performance for Centralised Organisation on Small Target Assignment Problem

The distributed (self-synchronised) process involves starting each agent with an agreed-upon poor solution (in this case, assigning the agent with internal numerical identifier *i* to the target with identifier *i*), and repeatedly exchanging messages between pairs of agents which can benefit from swapping their targets. Figure 5 illustrates this process. The initial information exchange involves each agent broadcasting its position and the location of nearby targets. The final agreement phase ensures that, if the agents have produced slightly different solutions, they select the best one (that is, it is a version of the selection problem).

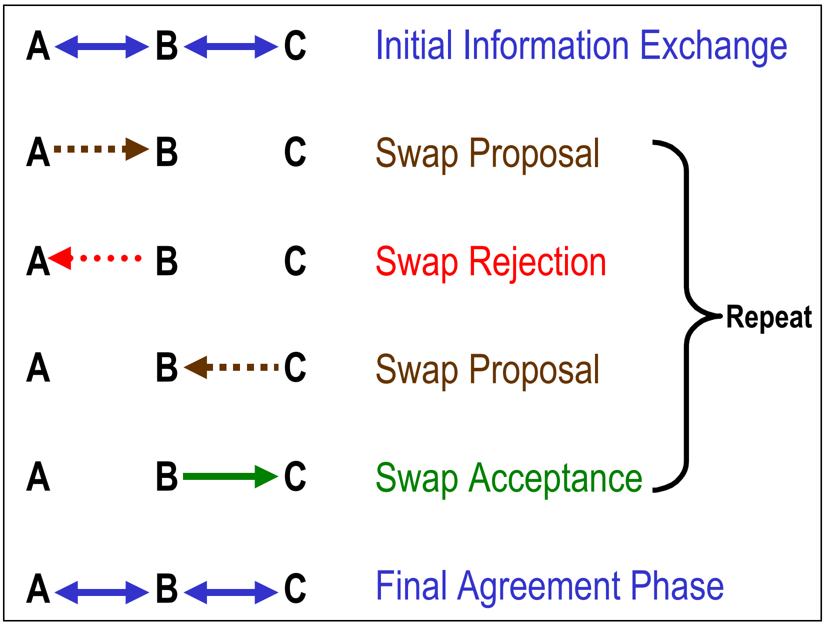

Figure 5: Example of Distributed (Self-Synchronised) Method for Target Assignment Problem with Three Agents

Results for the distributed process are shown in Figure 6. These are averages over 10 runs (except for the front right, where they are over 100 runs). Table 1 gives a comparison of the two techniques, calculated using ANOVA. The distributed process is slightly inferior to the centralised process, except when they both fail, and the difference is statistically significant, as shown in Table 1.

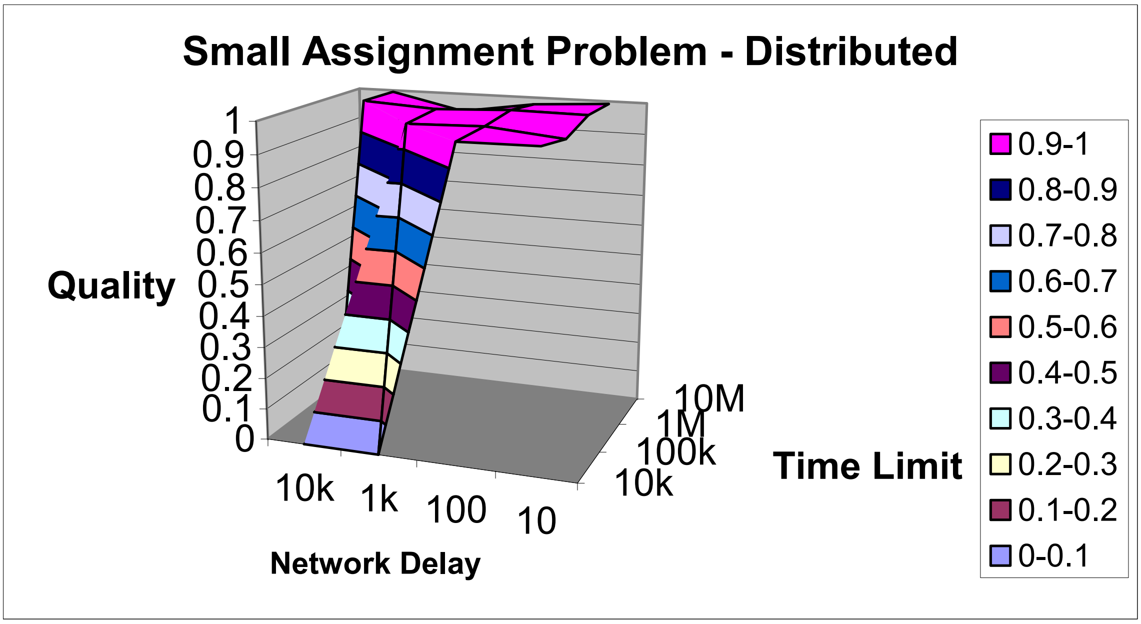

Figure 6: Performance for Distributed Organisation on Small Target Assignment Problem

Table 1: Means, Standard Deviations, and Performance Comparison for Small Target Assignment Problem

<table>
<tr><th colspan="2" rowspan="2"></th><th colspan="4">Network Delay</th></tr>
<tr><th>10k</th><th>1k</th><th>100</th><th>10</th></tr>
<tr><th rowspan="4">Time Limit</th><th>10M</th><td colspan="4" rowspan="2">Cent: 1 (perfect)
Dist: 0.981 (sd 0.035, 110 runs)
Difference totally significant</td></tr>
<tr><th>1M</th></tr>
<tr><th>100k</th><td colspan="2" rowspan="2">Cent: 0 (fail)
Dist: 0 (fail)
Identical</td><td colspan="2"></td></tr>
<tr><th>10k</th><td>Cent: 0.989 (sd 0.020, 100 runs)
Dist: 0.976 (sd 0.036, 100 runs)
Significant ($p < 0.005$)</td><td>Cent: 0.993 (sd 0.017, 100 runs)
Dist: 0.979 (sd 0.032, 100 runs)
Significant ($p < 0.0002$)</td></tr>
</table>

Since the centralised process finds the best possible answer in almost all cases, and the distributed process does not, the centralised process is obviously to be preferred, unless fault-tolerance is particularly necessary. Having the best possible answer provides a “knowledge edge” against an opponent. The real-life version of the centralised target assignment process is the Air Tasking Order for planning air strikes against fixed or slow-moving targets. The US Navy’s **Cooperative Engagement Capability** air defence system (CEC) uses a similar centralised process, with a “global scheduler” for target assignment [9,10].

5. Results & Analysis: Large Target Assignment Problem (Hard Case)

When we increase the number of agents involved in the target assignment problem from 16 to 64, the problem becomes more difficult. Here the centralised process only gives the best possible answer for the most relaxed time-limit (10,000,000 time units). For most other cases, the centralised process delivers a good, but not perfect, answer, with performance averaging about 0.98, as shown in Figure 7. However, the two cases with a time limit of 10,000 at the front right of Figure 7 show a reduced performance, given the tight time constraint.

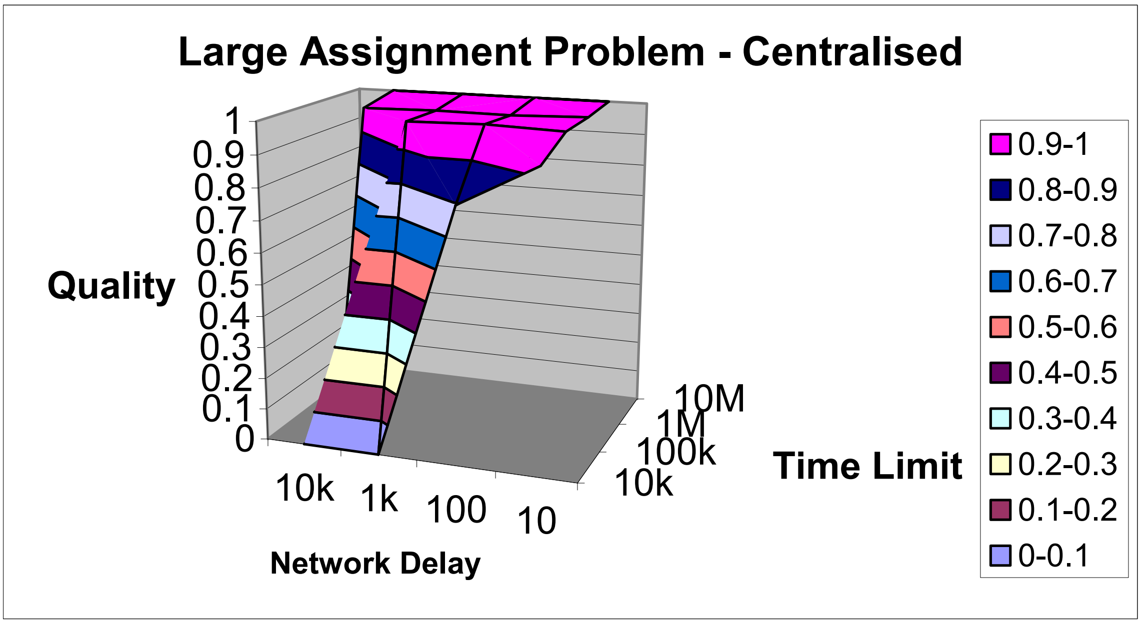

Figure 7: Performance for Centralised Organisation on Large Target Assignment Problem

Results for the distributed process are shown in Figure 8, and Table 2 gives a comparison of the two techniques, calculated using ANOVA. There are five distinct regions in the results:

- For a very relaxed time limit (10,000,000), the centralised process always returns the best possible answer, and is therefore preferred.
- For a time limit ≤ delay × 10, both methods fail, giving a performance of 0.
- For a time limit = delay × 100, the distributed process performs very poorly, since there is not enough time for the sequence of messages described in Figure 5 to be completed, and again the centralised process is preferred.
- For the worst time pressure (limit 10,000) and fastest network (delay 10), the distributed process performs about 5% better than the centralised process.
- For the remaining cases, both methods perform about the same.

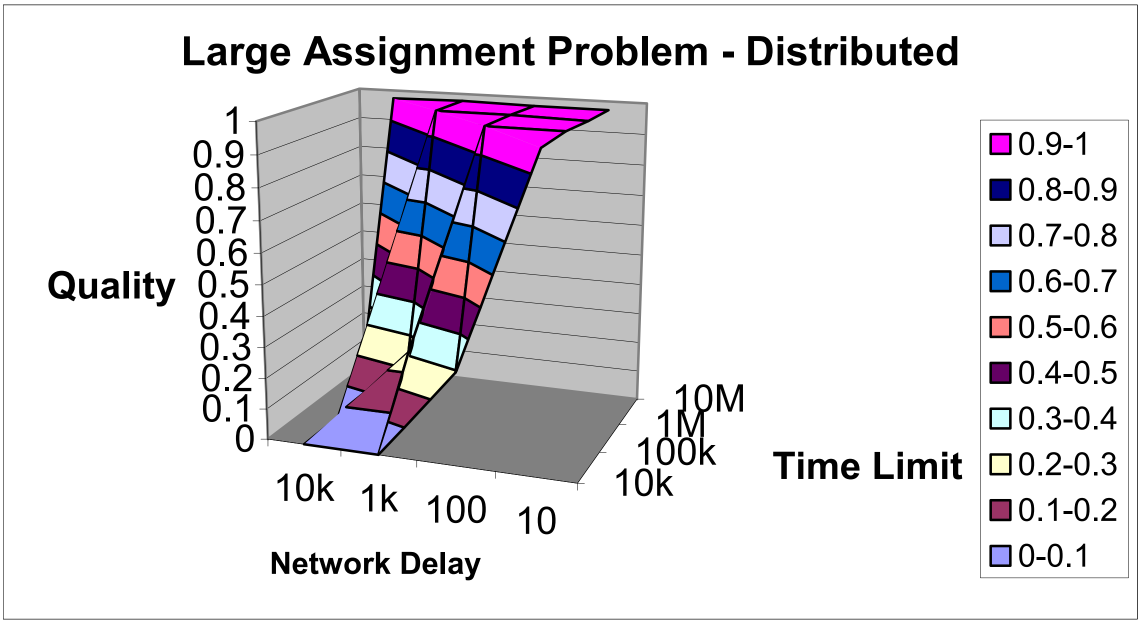

Figure 8: Performance for Distributed Organisation on Large Target Assignment Problem

Table 2: Means, Standard Deviations, and Performance Comparison for Large Target Assignment Problem

<table>
<tr><th colspan="2" rowspan="2"></th><th colspan="4">Network Delay</th></tr>
<tr><th>10k</th><th>1k</th><th>100</th><th>10</th></tr>
<tr><th rowspan="4">Time Limit</th><th>10M</th><td colspan="4">Cent: 1 (perfect)
Dist: 0.973 (sd 0.018, 40 runs)
Difference totally significant</td></tr>
<tr><th>1M</th><td>Included in 100k/1k ↘</td><td colspan="3">Cent: 0.983 (sd 0.016, 50 runs)
Dist: 0.979 (sd 0.018, 50 runs)
Difference Not Significant ($p = 0.30$)</td></tr>
<tr><th>100k</th><td></td><td>Time Limit = Delay × 100
Cent: 0.918 (sd 0.098, 30 runs)
Dist: 0.259 (sd 0.043, 30 runs)
Very Significant ($p < 10^{-39}$)</td><td colspan="2"></td></tr>
<tr><th>10k</th><td colspan="2">Cent: 0 (fail)
Dist: 0 (fail)
Identical</td><td>↖ Included in 100k/1k</td><td>Dist: 0.975 (sd 0.020, 100 runs)
Cent: 0.925(sd 0.031, 100 runs)
Very Significant ($p < 10^{-29}$)</td></tr>
</table>

For this more difficult problem, the distributed process can perform as well as the centralised process in many cases, and actually performs better when there is great time pressure and a fast network. This combination of fast network and extreme time pressure is of course exactly the situation where Network Centric Warfare (NCW) is appropriate [14]. The distributed process also has the advantage of fault-tolerance: it can continue to operate even when some agents go off-line.

There are other factors determining the choice of centralised *vs* distributed decision-making [11]. For example, in terms of decision-making facilities and staff, some agents may be better-equipped than others. Nevertheless, distributed (self-synchronised) military decision-making should be explored for scenarios where the combination of tight time pressure and fast networking occurs.

6. Results & Analysis: Travelling General Problem (Hard Case)

The "Travelling General" problem (more traditionally known as "travelling salesman") involves a General visiting all agents in turn, and returning to his starting point as shown in Figure 9. To reduce threats to his safety, the General should travel for the smallest possible distance.

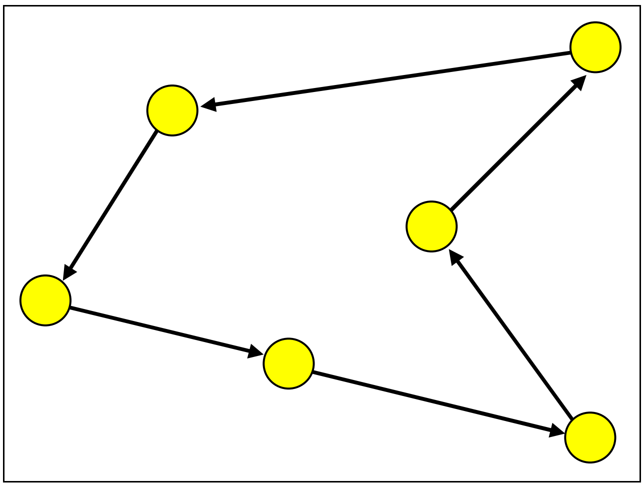

Figure 9: The Travelling General Problem

This is a more difficult problem than it appears: for n agents there are $(n–1) \times (n–2) \times \ldots \times 4 \times 3$ possible solutions, and finding the best possible answer requires checking all of them. Table 3 illustrates the scale of the problem.

Table 3: Number of Potential Solutions for Travelling General Problem

Number of agents *n*	Number of solutions to check
3	1
4	3
5	12
10	181,000
15	44,000,000,000
20	61,000,000,000,000,000
25	310,000,000,000,000,000,000,000
30	4,000,000,000,000,000,000,000,000,000,000

The exponential growth in the number of potential solutions means that finding the best possible answer for the Travelling General Problem is unrealistic for even moderate values of n. Many real-life problems share this property [6,7]. They include the target assignment problem if there are interactions between one assignment and the suitability of other assignments. The target assignment problem also falls into this category if it is extended to involve a three-way matching between a target, a "shooter," and a targeting sensor. Many logistics problems are also in this category. However, for many of these problems, finding a "good enough" solution is much easier than finding the best possible answer.

Figure 10 shows performance of the centralised process for 12 agents (the difficulty of the problem makes it infeasible to use more agents). The centralised process obtains a fairly good, but not perfect, solution by using the simple "greedy" process of successively finding the closest agent not yet chosen. If possible in the time available, this is then improved, using "Branch and Bound" [8]. This process gives a path for the General on average 10% longer than the optimum, corresponding to a performance of 0.91. As in the case of the large target assignment problem, the centralised process cannot deliver the best possible answer.

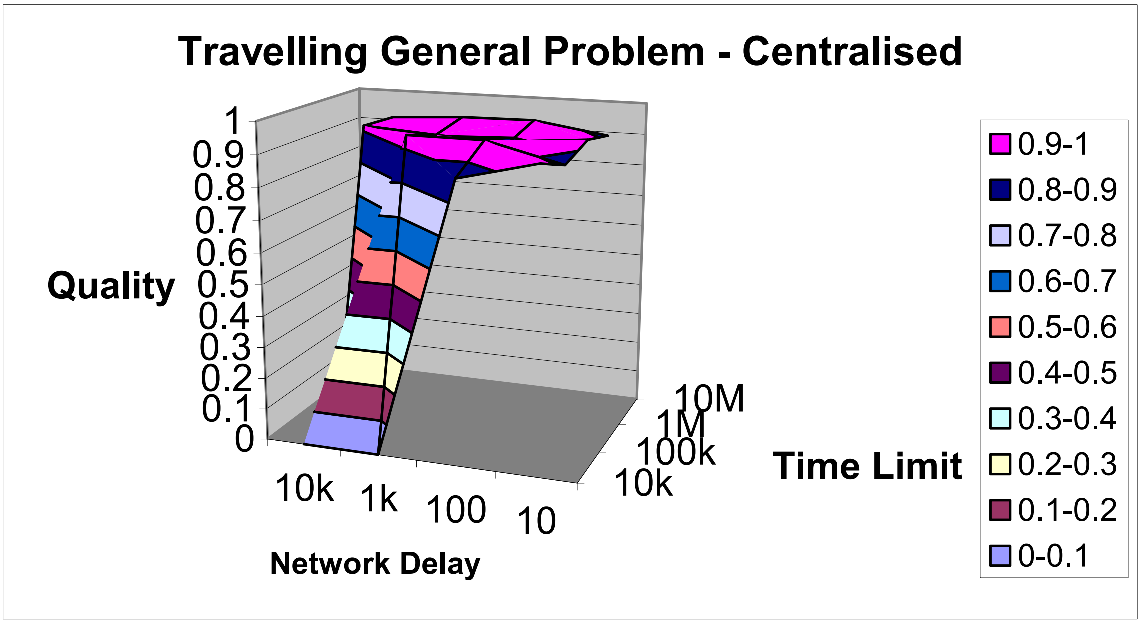

Figure 10: Performance for Centralised Organisation on Travelling General Problem

The distributed process mimics the “greedy” centralised process of successively finding the closest agent not yet chosen. Each agent in turn chooses the closest “next agent” not yet chosen, and sends a message instructing that agent to make the next decision. There is no significant performance difference between this and the centralised process, as shown in Figure 11 and Table 4.

The serial “one after another” nature of this greedy process does cause problems with slow networks and large numbers of agents. However, other distributed solutions are applicable to those cases [18], and we have found them to perform as well as the greedy process. The Travelling General problem can thus be solved equally well by centralised and self-synchronised techniques.

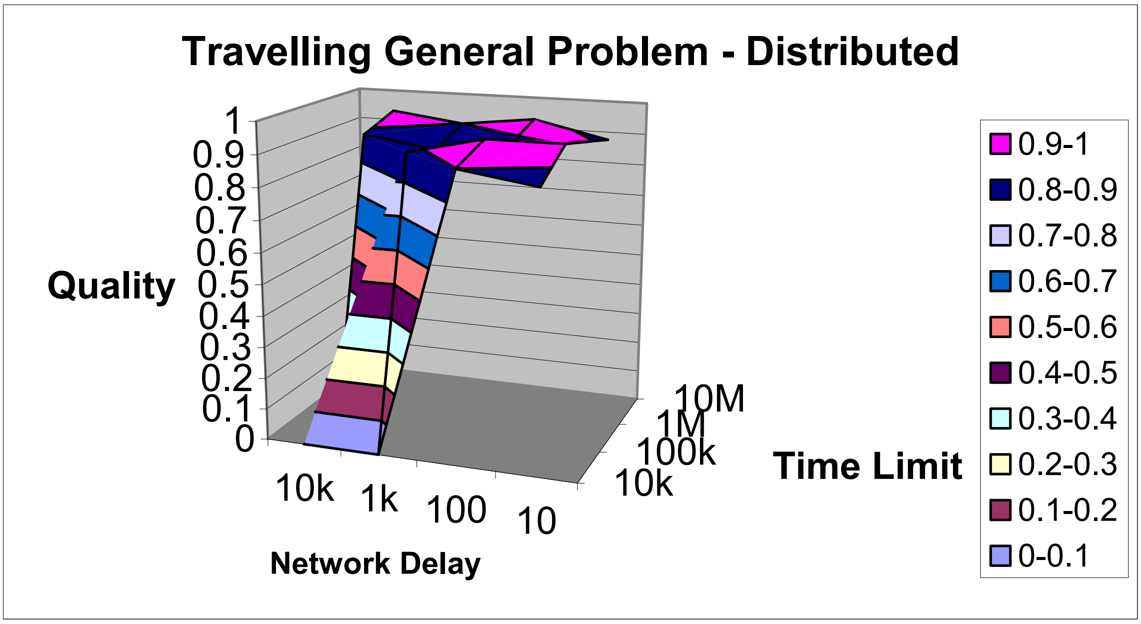

Figure 11: Performance for Distributed Organisation on Travelling General Problem

Table 4: Means, Standard Deviations, and Performance Comparison for Travelling General Problem

<table>
<tr><th colspan="2" rowspan="2"></th><th colspan="4">Network Delay</th></tr>
<tr><th>10k</th><th>1k</th><th>100</th><th>10</th></tr>
<tr><th rowspan="4">Time Limit</th><th>10M</th><td colspan="4" rowspan="2">Cent: 0.910 (sd 0.075, 130 runs)
Dist: 0.901 (sd 0.071, 130 runs)
Not Significant (p = 0.28)</td></tr>
<tr><th>1M</th></tr>
<tr><th>100k</th><td></td><td></td><td></td><td></td></tr>
<tr><th>10k</th><td colspan="2">Cent: 0 (fail)
Dist: 0 (fail)
Identical</td><td></td><td></td></tr>
</table>

7. Hierarchies: A Comparison

For comparison purposes, we also examined a hierarchical approach to solving the large target assignment problem. This used the structure shown in Figure 12. The 64 target-engaging agents were organised into 16 teams, and 5 additional command agents completed the hierarchical structure. The network topology in this case was the tree in Figure 12, rather than the “double ring.”

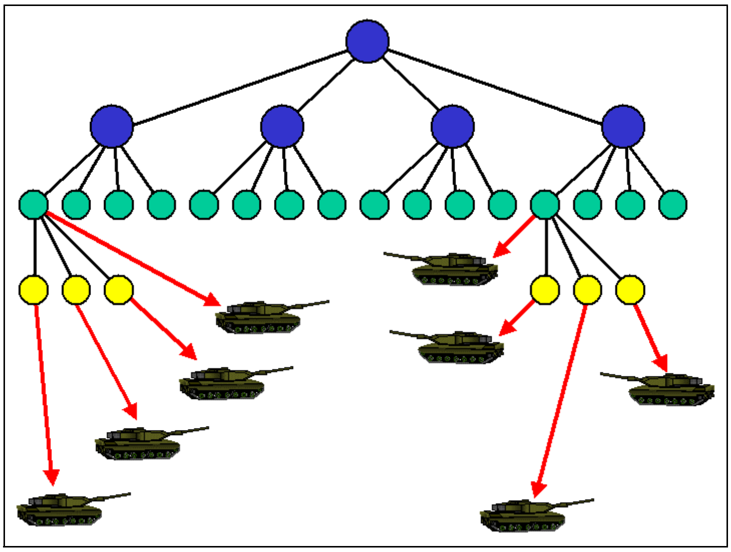

Figure 12: Hierarchy for the Target Assignment Problem

Hierarchical structures such as the one in Figure 12 are traditional in both military and bureaucratic organisations. This is because they are effective: they can solve problems by "divide and conquer." In this case, the higher-level agents divide the area of operations into rectangles containing only four targets, as in Figure 13. This reduces the large target assignment problem to sixteen trivial target assignment problems, in which a team assigns itself to a rectangle of four targets.

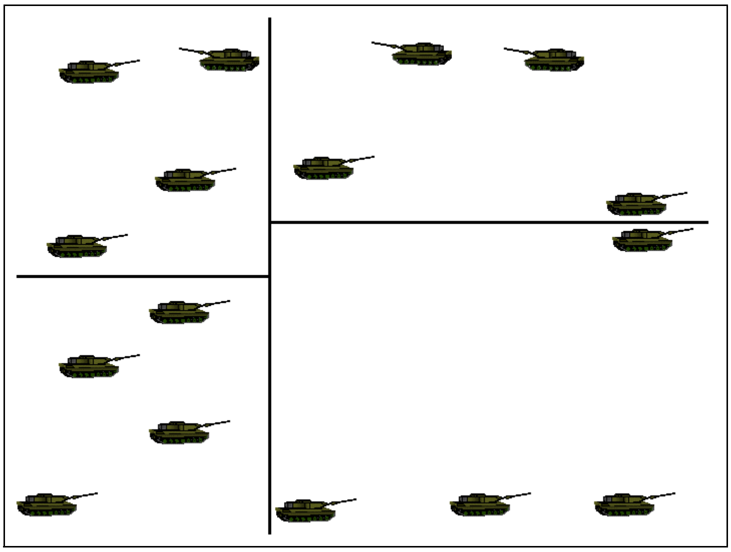

Figure 13: "Divide and Conquer" for the Target Assignment Problem

The problem is thus decomposed in a way that matches the organisational hierarchy. This hierarchical problem decomposition process has been traditionally used because it is effective for many problems, although it is not always fast. For real military problems, situational awareness information is collated and fused going up the hierarchy. The commanding node then produces a "big picture" plan (often called an "intent"). This is passed down the hierarchy, and tactical detail is added by subordinate nodes. This avoids over-straining the planning capability of the agents. Hierarchies thus combine elements of the centralised and distributed approaches.

A major weakness of hierarchical problem decomposition is that it may produce solutions which are far from perfect. For our experiment, the hierarchical approach succeeds for all but the usual three time-constrained cases, and produces solutions with an average quality of 0.851, independent of network delay and time pressure (with a standard deviation of 0.056 over 220 runs). Figure 14 illustrates the performance. For all but the three cases with time limit = delay × 100, the hierarchical approach performed worse than the centralised and distributed approaches ($p < 10^{-27}$).

For the case with time limit = delay × 100 (where the distributed process performed very poorly), the hierarchical approach was worse than the centralised approach ($p < 0.0000002$), but better than the distributed approach ($p < 10^{-49}$). However, this comparison is not entirely fair, since the tree topology in Figure 12 allows faster message transmission than the double-ring (prism) topology.

In general, the traditional hierarchical approach performs worse than the centralised and distributed approaches, since it allows only limited opportunity for coordination between the "leaves" of the tree. Performance can be improved either by centralising to greater extent, or by allowing some self-synchronisation between the "leaves" of the tree. To be effective, both of these possible changes require technologies which have only become available in recent decades. The implications of such variations to traditional military hierarchies are therefore still being explored.

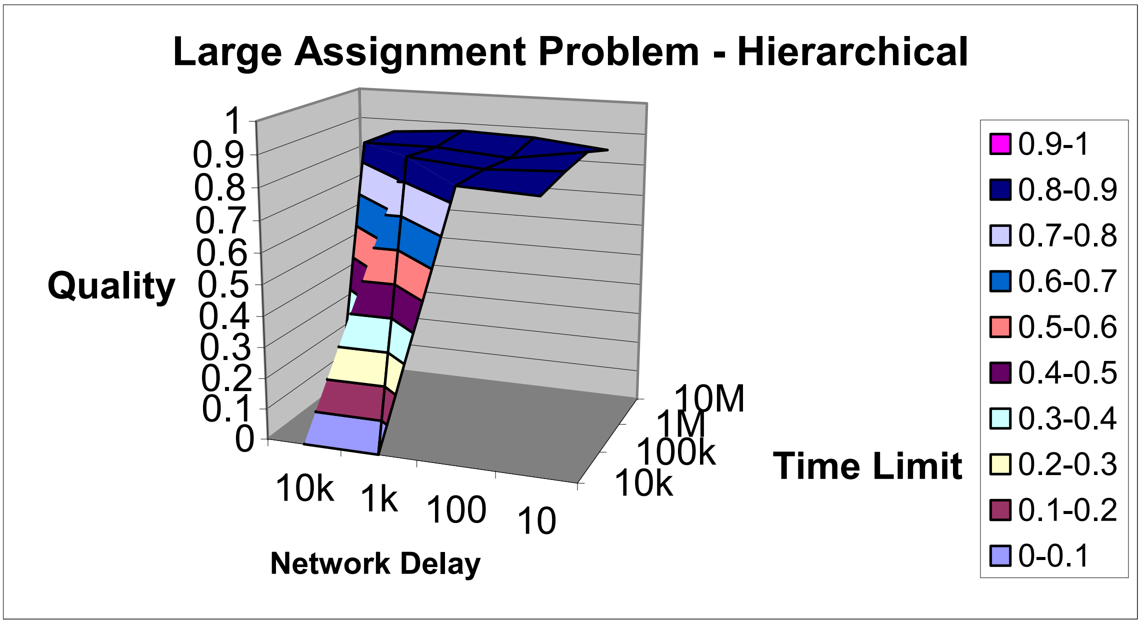

Figure 14: Performance for Hierarchical Organisation on Large Target Assignment Problem

8. Discussion

Our simulation results have demonstrated some important points about centralisation *vs* self-synchronised decision-making:

- For **trivial** problems, which often occur at the technological level, the centralised and self-synchronised approaches work equally well, but the self-synchronised approach is preferable because it is **fault-tolerant**.

- For **simple** problems or relaxed time limits, the centralised approach can often provide a "knowledge edge" by finding the best possible solution, which makes the centralised approach preferable, unless fault-tolerance and other factors require self-synchronisation.

- For **hard** problems, such as the Travelling General Problem, finding the best possible solution is not feasible, which means that self-synchronisation can be as good as the centralised approach. For the combination of extreme time pressure and fast networking, self-synchronisation can actually produce better answers than the centralised approach.

- Hierarchies are a traditional compromise between total centralisation and total self-synchronisation, but do not perform as well as either extreme case.

Thus, paradoxically, it is the simplest and the most difficult problems which benefit most from self-synchronisation, while for problems of intermediate difficulty centralised decision-making can sometimes provide an advantage. However, the need for fault-tolerance, and the availability of decision-making staff and facilities may influence this choice.

In the real world, many modern military problems do involve extreme time pressure. However, a key question for implementing self-synchronisation schemes is whether message transmission times (including the human processing component) have as yet fallen below the threshold where self-synchronisation can outperform centralised decision-making. The full advantages of self-synchronisation only begin to appear beyond that threshold.

In this paper we have considered **mission swapping** as one of the main self-synchronisation mechanisms (together with local "greedy" decision-making and the "borg" approach). Mission swapping was the technique that agents used in the distributed process for the target assignment problem. In real-life, mission swapping is easy to implement technologically, but is it culturally acceptable? Will a commander tolerate subordinates negotiating to swap their missions?

In addition to mission swapping, several other self-synchronisation techniques are possible. In **voting**, subordinates vote on possible courses of action. This can be faster than having subordinates explain to the commander the reasons for their preference, but such a "council of war" has traditionally been viewed as a sign of weakness, as well as breaking the principle of "unity of command."

Another possible self-synchronisation technique is **bidding** or **auctioning** [17], where subordinates bid for missions or resources using some form of "tokens" or "credits." This technique may lead to young officers preferring "glamorous" rather than necessary missions.

Self-synchronisation is also possible using **service/request** techniques [2], where some battlefield agents provide a "service," while other agents place requests for those services. This technique has historically been used for artillery support, and can therefore easily be extended to other services.

Finally, **distributed search** [15] is a self-synchronisation technique closer to information-gathering than to decision-making, and hence is perhaps culturally more acceptable.

This paper has demonstrated that self-synchronisation can be superior to centralised decision-making under some circumstances, but it remains necessary to find self-synchronisation doctrines and mechanisms which are acceptable to military culture. Having done that, it will also be necessary to adapt command and control systems to support those doctrines and mechanisms.

9. Acknowledgements

The author is indebted to Richard Taylor for discussions on this topic.

10. References

[1] Wheeler, S. & White, G., 2004, "Swarm or Borg? A comparison of Distributed and Collective Control," *Proc. 7th Asia-Pacific Conference on Complex Systems*, Cairns, Australia, 6–10 December.

[2] Dekker, A.H., 2005, "A Taxonomy of Network Centric Warfare Architectures," SETE 2005 – Systems Engineering / Test and Evaluation Conference, Brisbane, Australia, November.

[3] Ceri, S. & Pelagatti, G., 1984, *Distributed Databases: Principles and Systems*, McGraw-Hill.

[4] Mullender, S., 1993, *Distributed Systems*, 2nd edition, Addison-Wesley.

[5] Christofides, N., 1975, *Graph Theory: An Algorithmic Approach*, Academic Press.

[6] Rich, E., 1983, *Artificial Intelligence*, McGraw-Hill.

[7] Cormen, T.H., Leiserson, C.E., and Rivest, R.L., 1990, *Introduction to Algorithms*, MIT Press.

[8] Reingold, E.M., Nievergelt, J., and Deo, N., 1977, *Combinatorial Algorithms: Theory and Practice*, Prentice-Hall.

[9] Jane's Information Group, 2000, Delivering CEC potential, *Jane's Navy International*, March 01.

[10] Jane's Information Group, 2003, CEC, *Jane's Naval Weapon Systems*, **38**.

[11] Dekker, A.H., 2003, "Centralisation and Decentralisation in Network Centric Warfare**,**" *Journal of Battlefield Technology*, Volume 6, Number 2, July, pages 23–28.

[12] Dekker, A.H., 2005, "Network Topology and Military Performance**,**" in Zerger, A. and Argent, R.M. (eds), *MODSIM 2005 International Congress on Modelling and Simulation*, Modelling and Simulation Society of Australia and New Zealand, December, pp 2174–2180, ISBN: 0-9758400-2-9: www.mssanz.org.au/modsim05/proceedings/papers/dekker.pdf

[13] Scholz, J., Hengst, B., Calbert, G., Antoniades, A., Smet, P., Marsh, L., Kwok, H., and Gossink, D., 2005, "Machine Learning for Adversarial Agent Microworlds," in Zerger, A. and Argent, R.M. (eds), *MODSIM 2005 International Congress on Modelling and Simulation*, Modelling and Simulation Society of Australia and New Zealand, December, pp 2195–2201, ISBN: 0-9758400-2-9: www.mssanz.org.au/modsim05/proceedings/papers/scholz.pdf

[14] Alberts, D., Garstka, J., and Stein, F., 1999, *Network Centric Warfare: Developing and Leveraging Information Superiority*, (US) Department of Defense C4ISR Cooperative Research Program Publications Series.

[15] Marsh, L., Calbert, G., Tu, J., Gossink, D., and Kwok, H., 2005, "Multi-Agent UAV Path Planning," in Zerger, A. and Argent, R.M. (eds), *MODSIM 2005 International Congress on Modelling and Simulation*, Modelling and Simulation Society of Australia and New Zealand, December, pp 2188–2194, ISBN: 0-9758400-2-9: www.mssanz.org.au/modsim05/proceedings/papers/marsh.pdf

[16] Hromkovič, J., Klasing, R., Pelc, A., Ružička, P., and Unger, W., 2005, *Dissemination of Information in Communications Networks*, Springer Verlag.

[17] Chew, L.P., Cheah , M., Low, L., and Kong, J., 2005, "Composable and Adaptive Services Environment (CASE) – A Naturalistic System Architecture That Enables Command Staff to Organize," *Proceedings of the 10th International Command and Control Research and Technology Symposium*, MacLean, VA, June 13–16: www.dodccrp.org/events/2005/10th/CD/papers/217.pdf

[18] Platzman, L.K. and Bartholdi, J.J., 1989, "Spacefilling Curves and the Planar Travelling Salesman Problem," *Journal of the ACM*, **36** (4), October, pp 719–737.