19th International Congress on Modelling and Simulation, Perth, Australia, 12–16 December 2011
http://mssanz.org.au/modsim2011

A Heuristic Planning Algorithm For Highly Constrained Maximum on Ground Problems

S. Shekh [a] **and L. Marsh** [a]

[a] *Defence Science and Technology Organisation, Edinburgh, South Australia, Australia*
Email: Slava.Shekh@dsto.defence.gov.au

Abstract: In military operations, logistics distribution involves the transportation of equipment, people and sustainment (resources) to various physical locations, known as distribution nodes. The quantity of items being transported is often very large and the resources need to be moved as quickly as possible using the available transportation assets. A complicating factor is that the airports at the distribution nodes typically have a limitation on the number of aircraft that can be present at one time. This is known as a Maximum on Ground (MOG) constraint.

In simple cases, the MOG problem can be overcome by waiting for aircraft spots to become free. However in larger scenarios where there are more aircraft and more movements, there is a greater likelihood of congestion. In these cases waiting can cause deadlocks, where an aircraft is unable to move because it relies on another aircraft moving first. Starvation can also arise when an aircraft is indefinitely waiting for a spot to become free, but other aircraft are continually using it. This paper presents a Maximum on Ground Algorithm (MGA) for dealing with deadlock and starvation, and resolving congestion in highly constrained scenarios.

The MGA uses a combination of heuristics to overcome congestion and find solutions in the presence of MOG constraints. For example, consider the scenario shown in Figure 1. In this case there is only one MOG spot at each airport, so movement between airports is very constrained. The MGA uses the *forced move*, *swap* and *shuffle* heuristics to assist a *task* aircraft in completing its movement. This is accomplished by reorganizing the other aircraft in the scenario, so that the task aircraft can reach its destination without violating any MOG constraints.

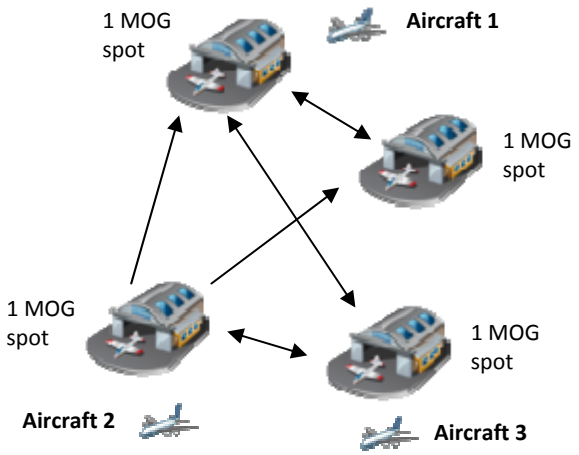

Figure 1. A scenario with numerous MOG constraints

Each of these heuristics provides a unique way of reorganizing the scenario, which is applicable in a particular set of cases. On the other hand, the *multi-shuffling* heuristic focuses on assisting the task aircraft by identifying a *utility* aircraft to provide assistance. The MGA identifies the constraints in a given scenario, and determines the best combination of heuristics to apply in order to overcome those constraints and find a solution.

A series of trials have demonstrated that the MGA is computationally efficient, capable of quickly solving moderate to large scenarios. The most computationally intensive heuristic is *multi-shuffling*, which involves each aircraft sending a request to every other aircraft. This results in a quadratic worst-case time complexity for the algorithm. In addition to rapid computation, the MGA produces efficient solutions by coordinating aircraft so that each task can be completed as quickly as possible without violating any MOG constraints. This is particularly true for the simpler cases, where the algorithm is able to achieve near-optimal efficiency.

Keywords: *Maximum on ground, military airlift, airfield capacity, aircraft scheduling*

1. INTRODUCTION

Logistics distribution in the military domain involves the rapid transportation of large quantities of equipment, people and supply items to one or more locations (Morton et al., 1995). The distribution is carried out using a combination of transport platforms, encompassing air, rail, road and sea. The nodes used to facilitate movement via these platforms become one of the key bottlenecks of distribution, and are often restricted in the number of aircraft spots available for use. This problem is known as Maximum on Ground (MOG) (Stucker and Berg, 1998). MOG can lead to conflicts arising in aircraft movement, so there need to be strategies in place for managing congested airports.

The MOG problem has many similarities to combination puzzles, particularly Rush Hour. The goal of Rush Hour is to move a target car outside of a grid, but movement through the grid is restricted by the presence of other cars (Flake and Baum, 2002). Similarly with a MOG scenario, an aircraft must move to a target airport and there are restrictions on where it can move. A generalized version of Rush Hour has been proven to be NP-hard (Flake and Baum, 2002), implying that there is no algorithm for finding an optimal solution in polynomial time. Furthermore, we are not aware of any algorithms in the literature that can optimally solve our specific MOG problem. For these reasons we have chosen to develop a heuristic algorithm.

Although minimal congestion can be resolved using simple techniques, such as waiting for an aircraft spot to become free, the same methods fail to provide solutions for highly constrained scenarios. When there are multiple congested airports and aircraft needs to make a delivery to one of those airports, simply waiting is unlikely to solve the problem and could result in deadlocks, where an aircraft is unable to move because it relies on other aircraft moving first. The heuristic algorithm described in this paper provides techniques for avoiding deadlock and overcoming other problems that arise in highly congested scenarios. The scope of the MOG problem is limited to aircraft, but other transport modes, such as maritime assets, could potentially face similar problems. The approaches and techniques discussed may be applicable to these other platforms.

2. RELATED WORK

The problem of aircraft flight scheduling is an ongoing area of research. Recent work continues to examine better ways of scheduling aircraft to maximise the efficiency of aircraft movement. For instance, Grosche (2009) explores the use of computation intelligence to improve airline scheduling, while Saraf and Slater (2008) use dynamic scheduling in an attempt to better manage the arrival of aircraft at congested airports. These approaches focus on improving the efficiency of airline schedules and maximizing throughput at airports.

The military faces a similar scheduling problem when planning airlift operations as part of distribution. The goal of these operations is to deliver resources (equipment, people and supply items) to a distribution node as quickly as possible, which means utilising all available air assets. However, the node's airport will potentially have limited capacity for aircraft (Stucker and Berg, 1998) and thus, the simultaneous use of many air assets can create a significant MOG problem (Gordon and Orletsky, 2003).

A number of mathematical models and algorithms have been developed to assist with planning airlift operations and resource distribution (Morton et al., 1995; Crino et al., 2004). These algorithms are deterministic and strive to find an optimal solution to the problem to inform the planner on how transport assets could be used to complete the distribution (Wu et al., 2009). Maximum on Ground is acknowledged as a significant problem in constructing a distribution model (Morton et al., 1995), and some models provide strategies for dealing with MOG. For example, Wu (2005) describes several MOG congestion functions that provide mechanisms for assigning airbases to aircraft in the presence of *hard* and *soft* MOG constraints.

3. PROBLEM OVERVIEW

The approach described in this paper has been developed and tested in a simulation environment for distribution planning. Unlike a mathematical model, the environment is founded on fewer assumptions, replacing them with real-world data to potentially improve the accuracy of the planning. Additionally, the simulation tool does not strive to find the optimal solution, only a reasonable solution that represents one possible realistic outcome.

In this simulation environment, Maximum on Ground is modeled as a *hard* constraint, where each location has a physical limit on the number of spots available for aircraft, and this limit cannot be exceeded. Consequently, decisions made in the earlier parts of the simulation can significantly influence the overall outcome. For instance, an aircraft movement at the start of the simulation can congest an airport and prevent aircraft from being able to land at that location during later movements.

This representation of MOG leads to many questions regarding resource distribution and aircraft movement. Firstly, how does one determine the best aircraft to use for distributing a resource? An aircraft that initially seems like the best option may result in a significantly worse overall solution due to a MOG constraint. Furthermore, what happens when an aircraft needs to land at a base that is full? The aircraft could potentially wait, but there is no guarantee that a spot will become free, so the aircraft may be forced to wait indefinitely. Another possibility is that an aircraft from the congested base moves to another base to free-up a spot, but where exactly does this aircraft go?

4. THE MAXIMUM ON GROUND ALGORITHM

The Maximum on Ground Algorithm (MGA) provides a general approach for dealing with MOG focused on solving highly congested scenarios. In a congested scenario, the number of aircraft approaches the total number of available MOG spots, making it very difficult for aircraft to move. The MGA uses a combination of heuristics to avoid deadlock and overcome congestion in these scenarios. Although the heuristics do not solve every single MOG problem, they can be used effectively in a large number of cases. Given a scenario, the MGA takes the following steps each time an aircraft needs to reach an airport:

1. If the aircraft is trying to reach an airport that is full, apply the *Forced Move* heuristic.
2. If the *Forced Move* heuristic doesn't find a solution, apply the *Swapping* heuristic.
3. If the *Swapping* heuristic doesn't find a solution, apply the *Shuffling* heuristic.
4. If the *Shuffling* heuristic doesn't find a solution, apply the *Multi-shuffling* heuristic.
5. If the *Multi-shuffling* heuristic doesn't find a solution, then the MGA has failed to find a solution.

The details of each heuristic are described in the following sub-sections. The order in which the heuristics are applied reflects their overall impact on the scenario. For example, *Multi-shuffling* is considered to have the most impact, because a solution produced by *Multi-shuffling* will generally involve more aircraft movement than solutions produced by the other heuristics. Therefore, *Multi-shuffling* would be applied as a final option.

The MGA heuristics are described in reference to a series of scenarios. Each scenario is modeled as a weighted directed graph, with airports represented as nodes, and routes represented as links. The weight of the links reflects the distance between the airports, and each aircraft has a maximum range that it can fly. Each time an aircraft arrives at an airport, its refills its fuel tank. Therefore, an aircraft can continue to fly indefinitely between airports, assuming that it has enough range to traverse the links. The model that we have adopted is based on our interpretation of the MOG problem, as derived from abstractions in the literature.

4.1. Forced Move

When an airport is full and an incoming aircraft needs to land there, the incoming aircraft can force one of the other aircraft to move. This approach can be used to solve the scenario in Figure 2, where Aircraft 1 needs to get to Sydney, but the only spot available in Sydney is being used by Aircraft 2. The solution is for Aircraft 1 to force Aircraft 2 to move out to Brisbane. However, if there was another aircraft in Brisbane and that aircraft had nowhere to move, then Aircraft 2 couldn't move to Brisbane, and consequently, Aircraft 1 would not be able to get to Sydney.

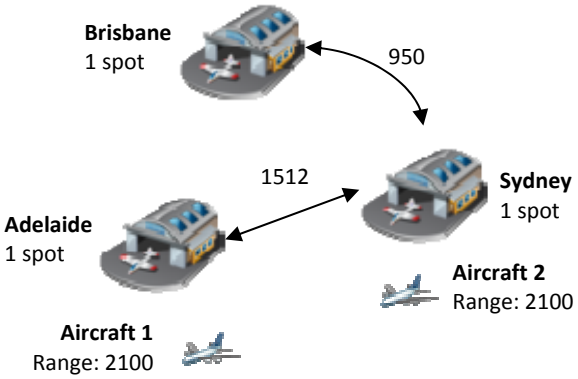

Figure 2. Aircraft 2 is forced to move to Brisbane

4.2. Swapping

Another approach that can be used even when all airports are full is known as *swapping*. The aircraft that needs to move can attempt to swap with one of the other aircraft already located at the destination airport. In order to perform the swap, both aircraft must be able to traverse the route between the airports and successfully land at the other airport.

This approach can be used to solve the scenario in Figure 3, where Aircraft 1 needs to get to Sydney, but the only spot available in Sydney is being used by Aircraft 2. The solution is for Aircraft 1 to swap with Aircraft 2, enabling it to reach Sydney.

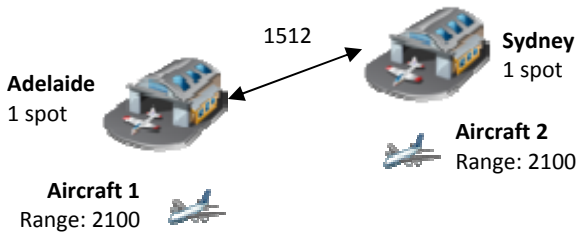

Figure 3. A scenario requiring the aircraft to swap

4.3. Shuffling

Although the Figure 3 scenario can be solved with swapping, this heuristic doesn't always work. For instance consider Figure 4, which contains a scenario with a three-way deadlock. In this case, Aircraft 1 needs to get to Sydney, but Aircraft 2 is using the only spot available in Sydney. Aircraft 1 and 2 can't swap because the routes in this scenario are unidirectional. In the military context, one-way routes can exist in the presence of a diplomatic clearance, which creates restrictions on aircraft movement.

Due to these unidirectional routes, the only way for a transport to move in this scenario, is for all transports to move simultaneously (referred to as *shuffling*). In the Figure 4 scenario, the shuffle involves Aircraft 1 moving to Sydney, Aircraft 2 moving to Brisbane and Aircraft 3 moving to Adelaide. This results in Aircraft 1 reaching its destination and exactly one aircraft at each airport, meaning that MOG has not been exceeded.

4.4. Multi-shuffling

The shuffling heuristic can resolve deadlocks that have a one-step solution, such as the scenario in Figure 4, but in some cases (see Figure 5) deadlock can only be resolved through a series of steps. In this scenario, Aircraft 1 needs to reach Sydney, but Sydney is full. Aircraft 1 would normally attempt to swap with Aircraft 2, but Aircraft 2 lacks the range to perform the swap. However, there is another aircraft in the scenario (Aircraft 3) which has sufficient range to swap with Aircraft 1, but is not located in Sydney and thus is unable to perform the swap.

One solution to this problem would be for Aircraft 3 to swap with Aircraft 2, and then once Aircraft 3 is in Sydney, it can swap with Aircraft 1, allowing Aircraft 1 to reach its destination. Although this seems like an intuitive solution, the action of Aircraft 3 swapping with Aircraft 2 has no direct utility in satisfying the goal, which is for Aircraft 1 to move to Sydney.

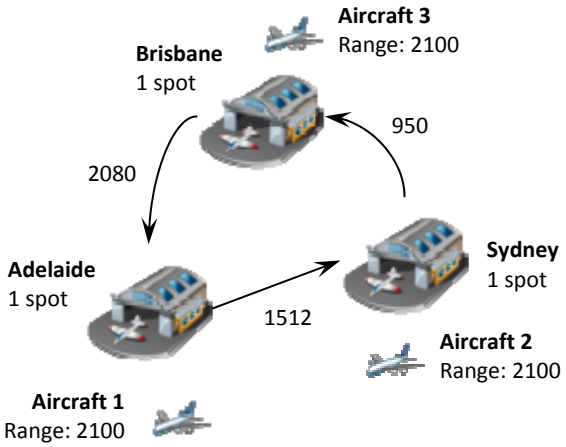

Figure 4. All three aircraft must move at once

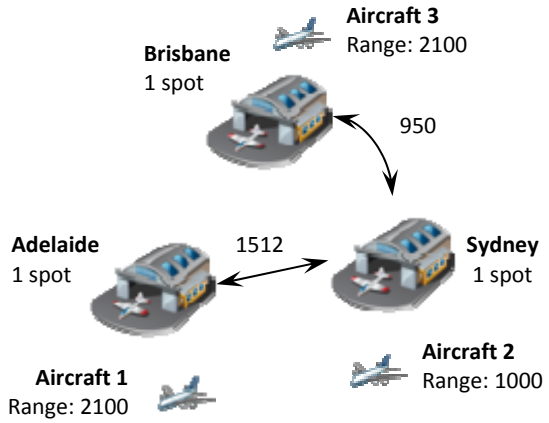

Figure 5. Aircraft need to cooperate

To overcome this problem, the *multi-shuffling* heuristic can be applied. This approach identifies a *utility* aircraft in the scenario that can assist the aircraft that is performing the movement. In the Figure 5 scenario, Aircraft 3 is identified as the utility aircraft and needs to assist Aircraft 1. The utility aircraft is then given the goal of reaching a base from which it can assist Aircraft 1. Consequently, Aircraft 3 moves to Sydney by performing a swap with Aircraft 2, from where it can assist Aircraft 1 in completing its movement.

4.5. Trapping

Some deadlocks have very specific solutions, and if that solution isn't used, the deadlock can become irreparable. For instance, Figure 6 shows a scenario that can result in a situation referred to as *trapping*.

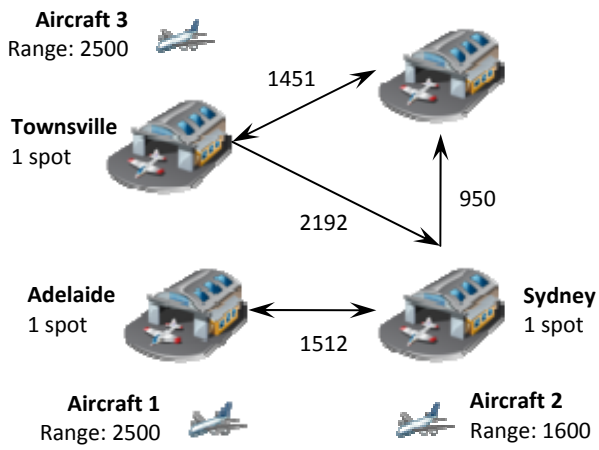

Figure 6. Aircraft 2 will become trapped by moving to Brisbane

In this case, Aircraft 1 is loading in Adelaide and Aircraft 2 is idle. Aircraft 3 needs to get to Sydney and therefore tries to force Aircraft 2 to move out of Sydney. The only free airport is Brisbane. However, if Aircraft 2 moves to Brisbane, it has no way of returning to Sydney, because it lacks the range to traverse the path from Townsville to Sydney. This situation is known as *trapping*.

To avoid trapping, Aircraft 3 must not force Aircraft 2 to move to Brisbane. It must wait for Aircraft 1 to finish loading, and then allow Aircraft 1 and 2 to swap. Once Aircraft 1 is in Sydney, Aircraft 3 can force Aircraft 1 to move to Brisbane without any possibility of trapping. In spite of this, if Aircraft 2 never needs to return to Sydney, then moving to Brisbane may not be an issue.

5. TRIALS AND RESULTS

A series of trials have been run to assess the efficiency and optimality of the Maximum on Ground algorithm. The efficiency is concerned with how quickly the algorithm finds a solution, while the optimality is a measure of how good that solution is in relation to the optimum.

5.1. Brute Force Algorithm (BFA)

In order to benchmark the MGA, a Brute Force Algorithm (BFA) has also been developed. The BFA attempts to solve the MOG problem by looking at every possible solution in order to find the optimum. At each time step, the BFA identifies every possible move that can be made by every aircraft in the scenario, and models each of them as a state. The algorithm then searches for all states that can be reached from each of the generated states and repeats this process recursively. If a generated state is deemed to be inefficient, it is pruned and the algorithm doesn't explore that branch any further. The BFA continues searching until all states have been examined or pruned.

The time step affects how frequently the state tree is expanded, which will have an impact on solution optimality and the time it takes to find that solution. Both algorithms use minutes as their base time unit, so with a time step of one minute, the BFA will expand the state tree and branch further every minute. If the scenario occurs over the course of a day, using a time step of one will result in a very large state tree and become computationally infeasible. Thus, a time step of 60 minutes has been selected for the following trials.

5.2. Computational Efficiency

The efficiency of the MGA was compared to the BFA in six scenarios of increasing complexity. In each scenario a number of tasks need to be completed, which involve moving items between two locations. Table 1 shows the computation time of both algorithms for each scenario, along with the heuristics used by the MGA and the number of solutions examined by the BFA to find the best solution. The swapping, shuffling and multi-shuffling heuristics are abbreviated as SW, SH and MS respectively. The upper part of the table shows the number of variables, giving an indication of scenario complexity. The MOG limit of all airports is 1, indicating a high degree of congestion across all scenarios.

From Table 1, it can be seen that the MGA requires minimal computation time, even as the size of the scenario increases. Out of the MGA heuristics, *multi-shuffling* is the most computationally intensive, because it involves each aircraft requesting every other aircraft for assistance with the movement. Thus, the highest order term is the number of aircraft, and using asymptotic analysis, we derive that the MGA has a quadratic worst-case time complexity:

$$T(n) = \mathrm{O}(N^2) \quad \text{where } N \text{ is no. of aircraft}$$

Table 1. Computation time of MGA and BFA in different scenarios

	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
Airports, *A*	2	3	4	5	9	12
Aircraft, *N*	2	3	4	4	6	9
Tasks	1	1	1	1	4	9
Duration (minutes)	271	403	678	755	4358	8849
	Maximum on Ground Algorithm					
Computation Time	3.1 ms	4.2 ms	6.3 ms	6.6 ms	47 ms	182 ms
Heuristics Used	SW	SW, SH	SW, MS	SW, SH	SW	SW, SH, MS
	Brute Force Algorithm					
Computation Time	3.5 ms	22.1 ms	1.9 seconds	28.1 seconds	-[1]	-[1]
Solutions Examined	8	1,166	181,288	2,832,452	-	-

There are other variables that will influence the computational complexity of the MGA, such as the number of airports, but they are lower order terms so they will have no impact on the worst-case complexity. In contrast, the BFA requires progressively more computation time, because an increase in the number of variables causes the time complexity to increase at double exponential time. This results in larger scenarios having enormous completion times, such as Scenarios 5 and 6 which did not complete after running for 48 hours. Assuming N is the number of aircraft and A is the number of airports, the worst-case time complexity of the BFA can be expressed as:

$$T(n) = \mathrm{O}(N^{A^R}) \quad \text{where } R = \frac{Duration}{Timestep}$$

The actual time complexity of the BFA will be smaller, because the algorithm generates every possible state, but only branches further from states that are deemed to be efficient, resulting in a smaller state tree and a lower computation time. Nonetheless, as shown in Table 1, a slight increase in scenario complexity results in double exponential growth of the computation time, suggesting that the BFA is not feasible to use in moderate or larger scenarios.

5.3. Algorithm Optimality

In addition to computational efficiency, the MGA was assessed on the optimality of the solutions that it produced, by comparing it to the best solution found by the BFA, given the chosen time step of 60 minutes. The optimality is a measure of the total time that aircraft spent flying between airports. Therefore, a solution is deemed optimal if the scenario was successfully solved and the aircraft movement was minimal. Figure 7 shows a comparison of MGA and BFA for the same scenarios that were used in the efficiency analysis (Section 5.2). Scenarios 5 and 6 are not included, because the BFA wasn't able to find a solution in a reasonable time.

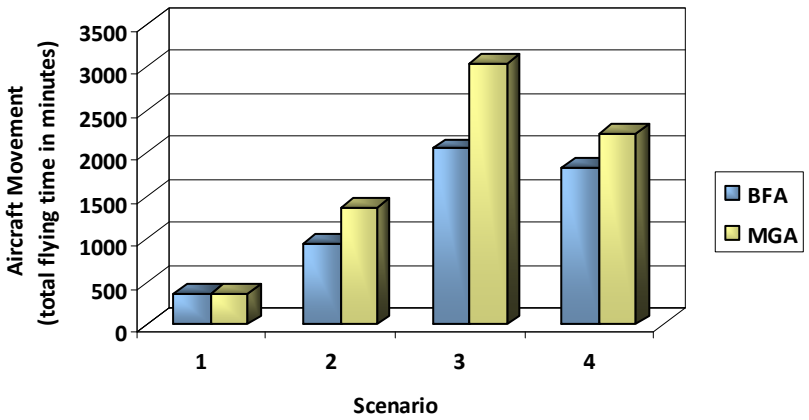

Figure 7. A comparison of algorithm optimality (less aircraft movement is better)

[1] Did not complete after running for 48 hours

Figure 7 shows that the MGA can produce near-optimal solutions in smaller scenarios, but becomes less efficient as the size of the scenario increases. The most notable inefficiencies can be seen in Scenario 3, where the MGA employs the multi-shuffling heuristic (refer to Table 1). This suggests that the multi-shuffling heuristic could be an area for further research and development.

6. DISCUSSION AND CONCLUSIONS

The different scenarios demonstrate that the MGA is a computationally efficient algorithm with a quadratic worst-case time complexity. This also means the algorithm is very scalable, being able to find solutions to MOG-constrained scenarios of varying size without significant increases to the computation time. Furthermore, the solutions produced by the MGA are quite efficient, although there are a several areas where they could be improved.

One area of optimization is the multi-shuffling heuristic, which currently only considers the closest valid aircraft for its choice of *utility* aircraft. However there may be other aircraft in the scenario that may be more suitable for providing assistance. The choice of aircraft could improve the optimality of the solution, but finding the best aircraft will result in increases to the computation time.

Inefficiency also exists in the way the heuristics are applied. For example, the aircraft that are involved in a shuffle currently start moving at the time when the shuffle is activated. Aircraft can potentially pre-empt the actions of other aircraft and begin moving earlier, which would minimise aircraft movement in the scenario. All heuristics suffer from this problem so pre-emptive movements could provide significant benefits to overall solution optimality.

Finally, there are MOG-constrained scenarios that the MGA is unable to solve, due to the complexity of the steps required to reach the solution. Additional heuristics could be developed for these scenarios, but the cases are very specific and would rarely arise in the real-world. Given their rarity, these cases could simply be handled by a human operator. Therefore, the key area for future work is improving the solutions produced by the algorithm, so that MOG problems are resolved as efficiently as possible. This may also provide a better representation of the way in which MOG is handled in the real-world.

ACKNOWLEDGMENTS

The authors acknowledge Dr. Don Gossink for his contribution to the development of the MOG algorithm.

REFERENCES

Crino, J. R., J. T. Moore, J. W. Barnes and W. P. Nanry (2004). Solving the Theater Distribution Vehicle Routing and Scheduling Problem Using Group Theoretic Tabu Search. *Mathematical and Computer Modelling,* 39(6-8), 599-616.

Flake, G. W. and E. B. Baum (2002). Rush Hour is PSPACE-complete, or "Why you should generously tip parking lot attendants". *Theoretical Computer Science,* 270(1-2), 895-911.

Gordon, J. and D. Orletsky (2003). Moving Rapidly to the Fight. *The US Army and the New National Security Strategy,* pp. 191-216.

Grosche, T. (2009). Integrated Airline Scheduling. *Computational Intelligence in Integrated Airline Scheduling,* SCI 173, 59-171.

Morton, D. P., R. E. Rosenthal and L. T. Weng (1995). Optimization Modeling for Airlift Mobility, Naval Postgraduate School.

Saraf, A. P. and G. L. Slater (2008). Optimal Dynamic Scheduling of Aircraft Arrivals at Congested Airports. *Journal of Guidance, Control, and Dynamics,* 31(1), 53-65.

Stucker, J. P. and R. T. Berg (1998). Understanding Airfield Capacity for Airlift Operations, RAND.

Wu, T. (2005). The Optimizing Simulator for the Military Airlift Problem. Ph.D. Thesis, Princeton University.

Wu, T. T., W. B. Powell and A. Whisman (2009). The Optimizing-Simulator: An Illustration Using the Military Airlift Problem. *ACM Transactions on Modeling and Computer Simulation,* 19(3).