MASCOT: Manycast Architecture for Service-Oriented Tactical Operations

Neal Charbonneau, Vinod M. Vokkarane and Ramprasad Balasubramanian
Department of Computer and Information Science, University of Massachusetts, Dartmouth, MA
E-mail: {u_ncharbonne, vvokkarane,r.bala}@umassd.edu
Address: 285 Old Westport Rd, N. Dartmouth, MA - 02747
Phone: (508) 910-6692

Abstract—*In this paper we present our work on the Manycast Architecture for Service-Oriented Tactical Operations, or MASCOT, a project between the University of Masschusetts, Dartmouth and the United States Marine Corps (USMC). MASCOT is a service that can be used by the Combat Operation Centers (COCs) to efficiently allocate tactical field resources given a task request, or mission, and set of constraints. We have implemented a prototype as a standalone Java application that is able to allocate resources for tactical missions. COCs deal with a large amount of dynamic input from other systems and must manage a number of tactical field resources. Finding good resources given a specific request involves examining a large solution space which is difficult for human operators. MASCOT can make this task faster and more accurate.*

I. INTRODUCTION

MASCOT is a tool to allocate tactical field resources, such as humvees, marines, and UAVs given a mission and set of constraints, called service priorities. The efficient allocation of tactical field resources by Combat Operation Centers (COCs) is important to provide battle rhythm and to the success of a mission or set of missions. The operators at the COC must deal with a huge amount of input from various tactical data systems (TDSs) and based on this data must choose resources to carry out specific missions. The solution space for sets of resources to carry out a particular mission is very large. This makes the task difficult for a human operator. MASCOT is designed to provide some automation for this process while still allowing the operator to make changes to the resource recommendations or service priorities.

MASCOT is a tactical planning tool to be used by COCs to manage field resources. We have currently developed a prototype application and are working with the USMC to determine how the prototype will evolve. MASCOT works with data that can be collected from other resources available to the COC and its own data store. MASCOT utilizes a communication paradigm called *manycasting* [1] to select a set of field resources based on a request. In manycast communication, the specific recipients are not specified, as they are in unicast communication. A manycast request simply specifies a set of possible recipients and a number of recipients that must be reached. Let the set of all possible recipients be V. Manycasting is concerned with a subset of destinations called the candidate set, or D_c. The candidate set is specified along with the number of recipients that must be reached in that candidate set, denoted by k. The particular destinations to reach is not specified, any k destinations from D_c will satisfy the request. Formally, $D_c \subseteq V$, $|D_c| \geq k$. The manycast request can be denoted as (D_c, k). For MASCOT, the manycast request would state that the COC mission needs six Cougars and five HUMVEES [2], for example. MASCOT would then use an optimization algorithm to choose the Cougars and HUMVEES of a larger set (the candidate destination set) instead of directly specifying the exact resources.

There has been previous research on mission planning using different types of computer algorithms [3] [4]. These applications focus on using genetic algorithms to generate courses of action. These courses of action are the plans that will direct the resources into battle. MASCOT is similar, but does not do planning of courses of action. Instead, MASCOT focuses only on tactical field resource allocation. Given a type of mission and set of priorities, it finds the best resources to use. The generation of the plan of action for these resources is still left to the COC operator.

This paper will provide details about the current implementation of MASCOT. We will first discuss the overall blackbox model and then go into how the functionality and optimization algorithms are implemented. We will also discuss the future of the prototype based on our continuing collaboration with the USMC. MASCOT started as a request for proposals from the USMC [5] and now with a completed prototype is entering phase two.

The paper is organized as follows, Section II will discuss in more detail the problem MASCOT is designed to solve and its importance. Section III will describe the architecture of MASCOT, both how it works as a service and its implementation, Section IV discusses our current prototype, Section V will discuss where our work with the USMC is heading and how MASCOT will evolve as far as functionality, and

$$\begin{aligned} Specification &= [\{(Exp, .65, H), (Delay, .2, M), (Readiness, .15, L)\}, \{(Humvee, 2), (Cougar, 3)\}, (bearing, x/y, speed)] \\ Response &= [\{H_1, H_3, C_8, C_3, C_{12}\}] \end{aligned} \tag{1}$$

Section VI will conclude the paper.

II. Problem Statement

In this section we describe the goal of MASCOT and the problem it is intended to solve. MASCOT is not a mission planner in the sense of generating actions and predicting results. Instead, MASCOT focuses on choosing the "best" tactical field resources for a given mission. The COC is still responsible for mission planning, but can rely on MASCOT to choose a good set of resources to use for a specific mission. For example, the COC may know that it needs two Cougars and two Abrams. The COC already has the mission planned, but now needs to choose which field resources to use for the mission. This is where MASCOT is used. MASCOT is designed to allocate tactical field resources given a mission and set of *service priorities*. The service priorities are a key aspect of MASCOT. They are used by MASCOT as constraints to solve the optimization problem. Service priorities are mappable to the attributes of individual field resources. An example of a service priority would be past mission *experience*. Each resource can have an experience level and this can be used in determining which are the "best" resources. The current MASCOT prototype uses experience, readiness, agility, and delay sensitivity as service priorities. Experience is defined as how many times a resource has executed a particular mission. Missions completed successfully are given more weight than others. Readiness is defined as the inverse of the resource's load, where the load is how many missions are currently assigned to the resource. A high level of readiness implies that the resource does not have prior responsibilities. Agility is an attribute of the type of resource. It describes how well the resource can move over various terrain. A marine has higher agility than an Abrams. Lastly, delay sensitivity is used to describe how delay sensitive the mission is. If it is highly delay sensitive, this means that the mission must commence as soon as possible, which requires resources that can reach the destination soon. When applying delay sensitivity to resources, it is a function of their distance and the type of resource since certain resources can move faster than others. New service priorities can be added as long as they are mappable to attributes of resources.

MASCOT will be integrated as a service that is accessible by the COC. One possible solution is to implement MASCOT as a web service, for example. As previously mentioned, if the search space to find a set of good resources is very large, then it is difficult for human operators to find the best resources. Another important reason for MASCOT is that it will aide the USMC in integrating more technology in their COCs. MASCOT will introduce the need to collect more information that can then be used for other services such as mission tracking and path planning.

III. MASCOT Architecture

MASCOT is a service that outputs a set of field resources given a specific task request. The details of the task request and output will be described in the following subsection. We will then describe the actual implementation MASCOT, which involves the Manycast Service Engine (MSE) in Section III-B.

A. Blackbox Model

This section will describe the system in terms of inputs and outputs while the next subsection will discus actual implementation details. MASCOT takes as input a *task specification* and outputs a *task response*.

The task specification takes three parameters: p, which is a set of three-tuples describing the service priorities, t, which is a set of two-tuples describing the resources requested, and l, which describes the target. The three-tuples in p consist of the service priority, the weight (importance) of that priority, and the value of that priority which is on a scale of High, Medium, or Low. The weight of a service priority is typically defined by the type of mission. For example, experience may be the most important for a hostage rescue mission, so this would have the highest weight. The value of the service priority (High/Medium/Low) is used to filter resources. For example, we may only be interested in resources with high experience. We will discuss how this works in detail in the following subsection.

A task response is simply a set of field resources that satisfy the task specification. Along with the task response, a number of metrics are available that will be discussed in Section IV. An example of a task specification and task response is shown in (1).

In the task specification, the first set contains the three-tuples for service priorities. The first element in the tuple is the service priority name, the second is the weight, and the last is the value. So here the first service priority is experience with a weight of 0.65 and a value of High. The second set contains the tuples of requested resources. Here the request is specifying that two HUMVEES and three cougars are required. Lastly the target information is provided. To specify a target, the target's bearing, coordinates, and speed must be specified.

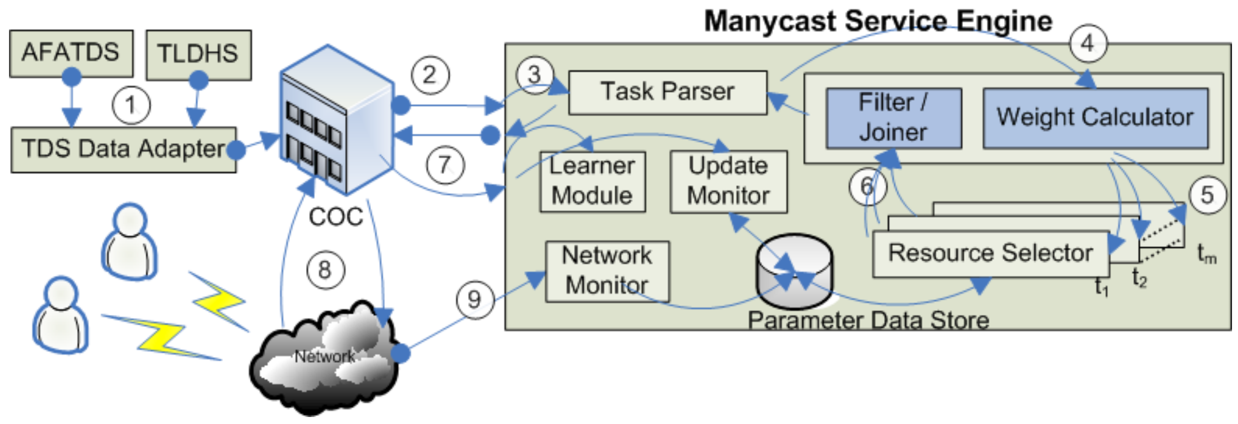


Fig. 1. MASCOT architecture.

Given this request, MASCOT will search the solution space (combinations of resources) using an optimization algorithm and will provide a list of ten recommendations. In (1) we only show one of the recommendation. The recommendation is simply a set of the specific resources that satisfy the task specification. Here the recommendation contains the two specific HUMVEES and three specific cougars. A more precise definition of a request is given in (2).

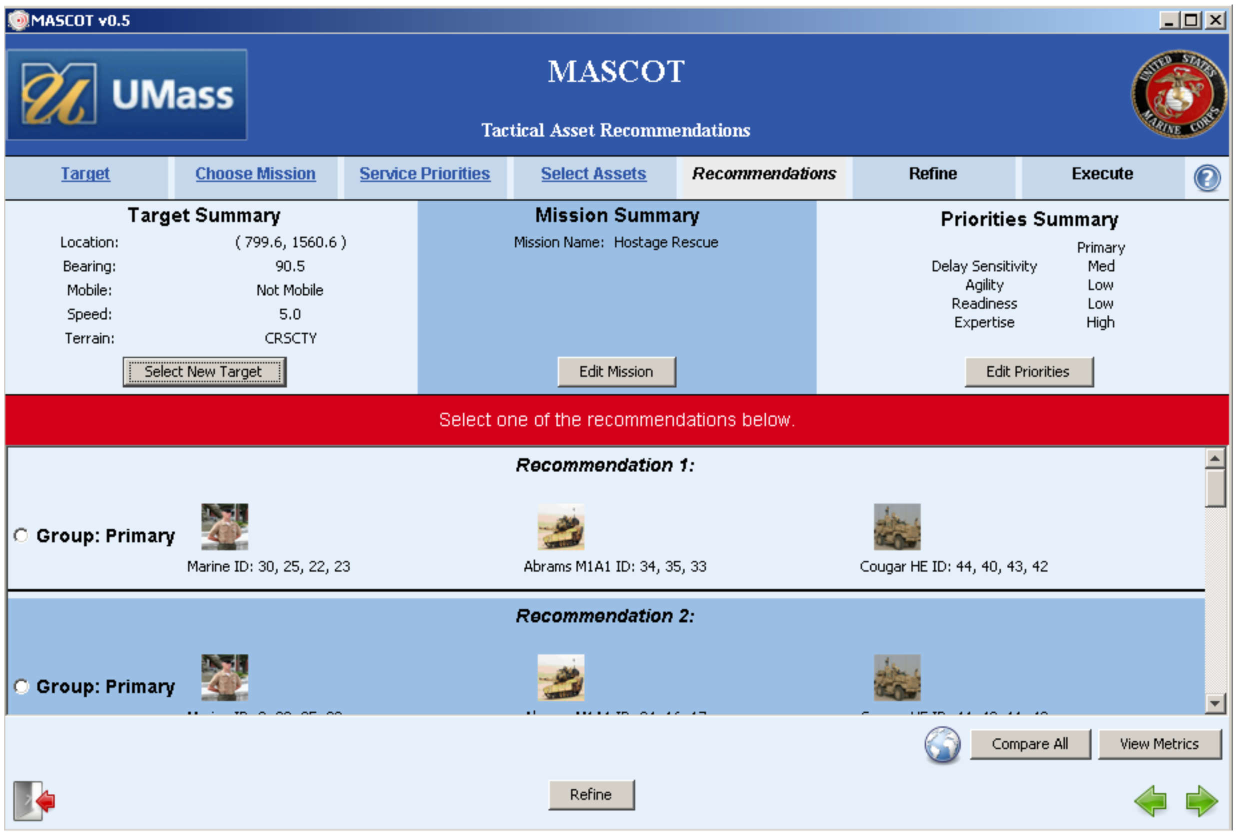
$$\begin{aligned}
Specification &= (p, t, l) \\
p &= \{p_0, p_1, ..., p_n\} \\
p_i &= (service\,priority_i, weight_i, level_i) \\
where &\sum_{i=0}^{n} weight_i = 1 \\
level_i &= \{High|Medium|Low\} \\
t &= \{resource_0, resource_1, ...resource_x\} \\
resource_i &= (name_i, requested_i) \\
l &= (bearing, coordinates, speed)
\end{aligned} \tag{2}$$

B. Implementation Details

In this section we will discuss the architecture and implementation details of MASCOT. The heart of MASCOT is the Manycast Service Engine (MSE) that takes the task specifications as input and outputs the task responses. The overall architecture is shown in Fig. 1.

We will go over each of the numbered steps as seen in the figure. Step 1 shows input coming in from various tactical data systems (TDSs). MASOT utilizes this information using the TDS data adapter. This information is used for updates to target information. The COC operator then inputs the necessary data into an interface to MASCOT in step 2 based on this information. Our Task Parser creates a task specification from this data in step 3. The Weight Calculator, which we will discuss further when we explain the optimization algorithm, creates a weight function that is used to evaluate resources in step 4. In step 5 the Resource Selectors will provide a list of weighted resources of each type. The weights are based on the resource's current status and the service priorities of the mission. The better the resource satisfies the priorities, the higher the weight it will have. The Resource Selector uses information in the Parameter Data Store which contains data collected on resources from other tactical data systems and also data that MASCOT creates on its own. This is then sent to the Filter/Joiner is step 6. The Filter first filters out resources that do not meet the service priority values in order to reduce the search space and then the Joiner will combine the results into recommendations which are turned to the user. The results are combinations of the individual weighted resources. The remaining steps are not implemented in our prototype, but may be included in future versions. Once the task is executed, it goes over the network as shown in step 8. Lastly, in step 9 we receive feedback from resources that is used to update the Parameter Data Store. We also show a Learning Module in the architecture, which is an area of future work that will be described later.

The MSE performs optimizations in order to select the best resources. We will discuss steps 4, 5, and 6 in more detail here. The general idea behind the algorithm is to use the service priorities as objective functions, so each one gives a score to the resource. We then use a weight function to combine the scores of the individual priorities into a single score for the resource. Next, we use a heuristic to combine individual resources into groups that match the task specification. The algorithm will find a minimum of ten recommendations and then return them to the user. When the user is creating the request, each of our four service priorities: Delay Sensitivity, Readiness, Expertise, and Agility are given one of three values: Low, Medium or High. A value of High for Expertise means the user is requesting resources with only a high level of expertise for the given mission. Each resource in the field can be given a



Target Summary

Location:	(799.6, 1560.6)
Bearing:	90.5
Mobile:	Not Mobile
Speed:	5.0
Terrain:	CRSCTY

Mission Summary

Mission Name: Hostage Rescue

Priorities Summary

	Primary
Delay Sensitivity	Med
Agility	Low
Readiness	Low
Expertise	High

Select one of the recommendations below.

Recommendation 1:

Group: Primary

Marine ID: 30, 25, 22, 23

Abrams M1A1 ID: 34, 35, 33

Cougar HE ID: 44, 40, 43, 42

Recommendation 2:

Group: Primary

Fig. 2. Example of a set of recommendations provided by the MASCOT prototype.

score for each of the service priorities to determine which category it falls in. Once each resource has a score for each service priority, the scores are combined based on a weight function. The weight function is based on the current mission and gives higher weights to certain Service Priorities. For example, some missions may require high levels of agility. In this case, the agility service priority will be given a higher weight in the weight function compared to the other service priorities. With this information, each resource now has a single weighted value. Simply choosing the top resources from each resource type may not be the a good solution because we must also consider the score of the combined group. To do this, there are two remaining phases of the algorithm: the first phase filters available resources while the second phase joins the results together to form the final recommendations.

The Filter uses the service priority values specified by the user to order the results by how well each resource satisfies those service priorities. The Filter starts by searching for resources that match the users original request and then relaxes each of the service priority values one by one and recording the additional resources that now match the settings. When this phase is complete, the best resources and the corresponding filter values those resources meet will be stored in sorted order from best to worst.

The Joiner then considers the users request and how many resources of each type the user requires. The Joiner begins by trying to find enough resources that satisfy the users original service priority values and by only searching possible combinations of these resources. This reduces the search space. For example, if the current service priority levels specified by the user contain enough resources that meet them to satisfy the request, the Joiner will then search combinations of these resources to create teams for the mission (we implement a limit of checking two times the number requested, to reduce the search space further). If there are not enough resources then the Joiner will relax the service priority levels to be able to search for combinations from a bigger pool of resources that may not all satisfy the users original request. All of this information is already available from the filter phase. Once the Joiner is able to find enough resources, either the first time or by relaxing the service priority levels, each of the combinations are enumerated and each of these teams of resources are ranked according to the sum of the scores of their resources (using the service priorities and weight function). If all of the resources come from a single unit then this recommendations score will be increased by 10%. We do this because we want to favor requests that can be satisfied from a single unit to increase the safety of this team. Once all of the combinations are ranked, the top ten are presented to the user. This algorithm is suitable for a relatively small number of assets (tens

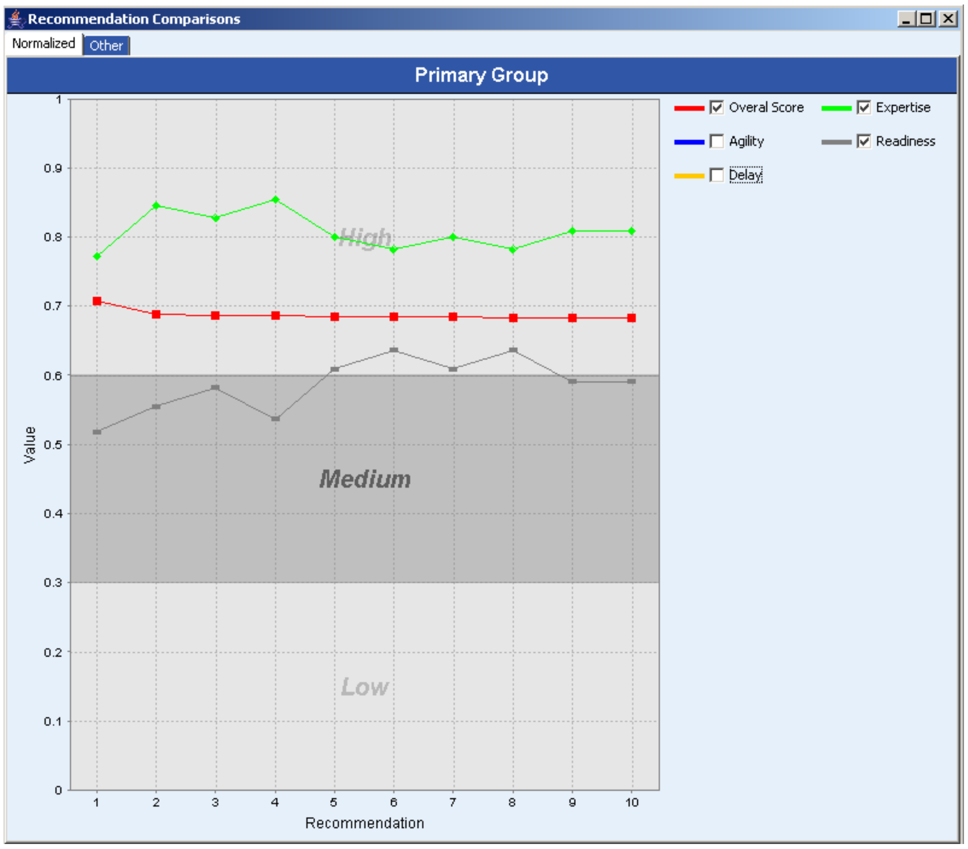


Fig. 3. Comparing recommendations with the MASCOT prototype.

or hundreds). One area of future work is exploring new optimization algorithms that are more efficient and can scale.

IV. Current Prototype

MASCOT is currently implemented as a standalone *Java* application prototype. We will briefly describe how the prototype works.

The prototype takes the user through a sequence of screens that will formulate the task specification as described in III-A when complete. First, the user is presented with a list of possible targets to choose from. This screen will gather all the target information required. Once this is chosen, the user is then presented with a list of mission types to choose from. Based on the type of mission, a set of service priorities are selected and assigned their weight values given their importance for that particular mission. Next, the user specifies the values for each of the service priorities (High/Medium/Low). The service priorities are now completely defined. The last thing that must be specified are the types and number of resources requested. The next screen will provide the user with information about how many are available and allow the user to select the required amount. At this point, the task specification is completed. The engine will run the optimization algorithm and present the user with a list of results. An example list can be seen if Fig. 2.

MASCOT also provides information about why the resources were selected. It provides a set of metrics that describe how well each of the recommendations matches the original task specification. The metrics provide the overall score given to the group of resources, the distance travelled, the time to target, and information about how well they satisfied the original service priorities. It also provides a graphical representation of how well the overall group performs based on the service priorities and allows the user to compare all of the recommendations together. An example of this can be seen in Fig. 3 (the High/Medium/Low in the figure correspond to High/Medium/Low of the service priorities). Here the user can look at the tradeoffs between groups and make a decision about which recommendation to choose. A certain group may have a very high score for a particular service priority and average scores for the remaining causing it to not be the first recommendation, but here the system givess the user the ability to make that tradeoff.

V. Future Work

We have recently had many discusses with the USMC about the future work of MASCOT since the completion of our initial prototype. In this section we will discuss the areas of future work that we will be investigating for phase two of the project.

A. Redesign and integration into existing systems

The first goal is to redesign MASCOT, particularly the MSE, so that it can be used as a web-service to allow easy integration into the USMC's existing infrastructure. MASCOT will also leverage many of the USMC's existing systems to gather much of the information that it needs, such as resource tracking and route selection.

MASCOT will include new resources types such as Unmanned Ariel Vehicles (UAVs), logistics and support. Our initial prototype focused on combat resources, but based on the hierarchical structure of the USMC, it is difficult to optimize at this level. The USMC will also provide an updated set of service priorities that more accurately reflect the important characteristics of resources.

B. New features

There are a number of new features that we plan to implement for MASCOT. We will briefly describe some of them here. A natural extension for MASCOT would be to integrate command logging into the allocation process. MASCOT will essentially create a mission chronology log that will log information such as who authorized the mission, the time, summary, and status information. Part of this would be to implement role-based authentication for the application.

There are a number of new features that could be added to deal with missions as a whole. We could implement different modes of Mission Planning. Currently, MASCOT is used for resource allocation, but we can provide different levels of planning. We could also provide Mission tracking and updating, which again ties into the command log described previously. MASCOT could also provide Mission Simulation based on path performance and past missions. We can leverage data available from other USMC systems to provide this and use models to predict the outcome. Another interesting area would be to implement a "War Planner", where instead of planning just a single mission, MASCOT can plan missions for an entire day or week or longer.

Another important area to investigate is evaluating alternative multi-objective optimization approaches. We used a simple heuristic for optimization. We can investigate more complex techniques like genetic algorithms or ILPs.

C. Long term research

The long term research tasks include handling multiple missions concurrently and large missions with concurrent submissions. We plan to develop accurate metrics to evaluate how well a mission was executed that can be used in the future. We will also investigate system scalability.

VI. Conclusion

In this paper we have presented MASCOT. MASCOT is our initial prototype for a tactical resource allocation system for the USMC. The current prototype allows for optimized resource allocation given a set of service priorities in a task specification. It then produces a task response consisting of the top ten recommendations.

We are currently entering phase two of the project with the USMC and will be evolving MASCOT.

While MASCOT focuses on allocation of combat resources, it can be used to handle different resource allocation problems as well.

VII. Acknowledgements

We would like to thank David Silvia of the Naval Undersea Warefare Center, Newport, RI for his guidance and feedback during the project.

References

[1] Xiaodong Huang, Qingya She, V.M. Vokkarane, and J.P. Jue, "Manycasting over optical burst-switched networks," *Communications, 2007. ICC '07. IEEE International Conference on*, pp. 2353–2358, June 2007.

[2] "Jane's information group," 2008, http://www.janes.com/; Last accessed: 08/2008.

[3] R.H. Kewley and M.J. Embrechts, "Computational military tactical planning system," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 32, no. 2, pp. 161–171, May 2002.

[4] Brad Rosenberg, Marc Richards, John T. Langton, Sofya Tenenbaum, and Daniel W. Stouch, "Applications of multi-objective evolutionary algorithms to air operations mission planning," in *GECCO '08: Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, New York, NY, USA, 2008, pp. 1879–1886, ACM.

[5] "Marine corps systems command (MCSC) product group 11, marine air/ground task force (MAGTF) command & control (C2) weapons & sensors development & integration (PG11 MC2I), PG11 effort to stimulate university research in C2 systems & enablers, request for proposals," Feb. 2008.