Dynamic rescheduling heuristics for military village search environments

Paul Maxwell, Anthony A. Maciejewski, Howard Jay Siegel, Jerry Potter and Jay Smith

The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology published online 7 January 2014
DOI: 10.1177/1548512913518665

JDMS

Journal of Defense Modeling and Simulation: Applications, Methodology, Technology
201X, Vol. XX(X) 1–18

DOI: 10.1177/1548512913518665
dms.sagepub.com
SAGE

Dynamic rescheduling heuristics for military village search environments

Paul Maxwell[1], Anthony A. Maciejewski[2], Howard Jay Siegel[2], Jerry Potter[2], and Jay Smith[3]

Abstract

On the modern battlefield cordon and search missions (also known as village searches) are conducted daily. Creating resource allocations that link search teams (e.g. soldiers, robots, unmanned aerial vehicles, military working dogs) to target buildings is difficult and time consuming in the static planning environment and is even more challenging in a time-constrained dynamic environment. Conducting dynamic resource allocation during the execution of a military village search mission is beneficial especially when the time to develop a static plan is limited and hence the quality of the plan is relatively poor. Dynamic heuristics can help improve the static plan because they are able to incorporate current state information that is unavailable prior to mission execution and thus produce more accurate results than static heuristics alone can achieve. There are currently no automated means to create these dynamic resource allocations for military use. Using robustness concepts, this paper proposes and compares dynamic resource allocation heuristics that create mission plans that are resilient against uncertainty in the environment and that save valuable time for military planning staff.

Keywords

resource allocation, robustness, stochastic, robust resource allocation, dynamic rescheduling, village search

1. Introduction

A frequent challenge for military planners is the rapid creation of resource allocations for operational requirements. One of the most common mission types encountered on the contemporary battlefield is the military village search (known in the military as ''cordon and search''). At the heart of this mission is a resource allocation problem that assigns military search teams (e.g. infantry soldiers, military working dogs, search robots) to targets (e.g. buildings, cache locations, structures) in a specific order. The problem of assigning search teams to targets is a computationally complex problem akin to assigning computing tasks to processors. Typical search missions can involve up to 60 search teams and 900 targets. In the military domain, this problem is made more difficult by the introduction of constraints on the solution such as boundary lines (lines drawn on a map that demarcate allowable search areas for teams), phase lines (ground reference lines drawn on a map that act as synchronization barriers controlling the forward movement of the search teams), directions of advance (limits search direction), and time deadlines. A small example village search problem is shown in Figure 1.

In our initial work,[1] we described robust resource allocation for a static planning environment where the plan is created a priori offline and resources and time available are not a primary concern. These plans are completed before the search mission begins and no feedback from the mission execution is used to modify the plan. In our current work, the search mission is started and then feedback from the mission execution is provided to modify the plan if necessary. In this dynamic environment, robust resource allocations (i.e. allocations that have the highest probability of meeting the selected robustness criteria) can be created that account for the numerous uncertainties in the

[1]United States Military Academy, West Point, NY, USA
[2]Colorado State University, Fort Collins, CO, USA
[3]Lagrange Systems, Inc., Boulder, CO, USA

Corresponding author:
LTC Paul Maxwell, Department of Electrical Engineering and Computer Science, United States Military Academy, West Point, NY 10996, USA.
Email: paul.maxwell@us.army.mil

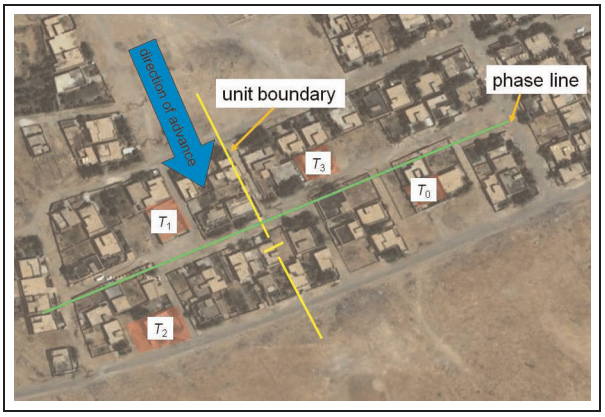

Figure 1. An example village search mission with four target buildings (T_j), a unit boundary, a restrictive phase line, and a direction of advance.

environment that include, but are not limited to, varying search rates, encounters with the enemy, the impact of weather on the search rates, and mechanical malfunctions affecting movement rates. A resource allocation that is well represented by expected values can be solved using techniques such as dynamic programming. In this environment, the distribution of the uncertainties is frequently not well modeled by expected values and thus these techniques will frequently not produce the best. Our goal is therefore to attempt the development of near-optimal resource allocations for the village search problem that are robust against these uncertainties. In general, the problem of resource allocation in heterogeneous parallel and distributed computing is NP-complete.[2,3]

Once a mission starts using an acceptable statically developed resource allocation there may be a requirement to adjust the resource allocation to improve the probability of mission success. This dynamic resource reallocation requirement could be the result of events such as the loss of a search resource, the addition of new target buildings to the target set, road blockages that prevent movement along a chosen path, and cumulative delays in building search times that result in missing the mission deadline time (MDT). The ability to identify instances when dynamic reallocation is beneficial and to create new allocations within the time constraints of a dynamic environment that enhance the mission's robustness (defined here as completing all building searches prior to the MDT) is highly desirable to military planners.

We believe that new dynamic methods are required to determine if and when a reallocation of resources is needed. When reallocations are required, then fast dynamic algorithms are needed to reallocate resources within the time constraints of an ongoing mission. Adding automated dynamic reallocation components to the Robust People, Animals, and Robots (RoPARs) tool developed by Maxwell et al.,[1] or creating a stand-alone solution (independent of RoPARS) for village search replanning in both training and operational environments would greatly assist military leaders in their decision-making process. The stand-alone solution would allow static mission plans to be evaluated for dynamic reallocation regardless of the method used to create the static plan. Thus, even non-automated static plans could be improved using our dynamic techniques. The dynamic replanning tool requires the following capabilities to be effective: model the search area using automated digital maps such as Environmental Systems Research Institute (ESRI) shapefiles; use probabilistic models to calculate search times; determine a good solution from multiple possible resource allocations; incorporate real-time mission feedback (e.g. actual search completion times of target buildings); and execute within the time constraints of an ongoing mission.

The contributions of this paper include: (1) a new framework for conducting dynamic military mission replanning; (2) new dynamic resource allocation heuristics that produce robust mission plans within mission time constraints; and (3) evaluation and analysis of these heuristics through simulation. The framework outlines how to define events that lead to reallocation and to create triggers that improve the execution time efficiency of the reallocation heuristics in a village search environment. The dynamic resource allocation heuristics select an allocation that results in an acceptable plan based upon the computation time.

The remainder of this paper is organized as follows. Section 2 presents work related to the dynamic village search problem. Section 3 provides an overview of the robustness metric developed by Ali et al.[4] and Shestak et al.[5] and its application in village searches. In Section 4, the resource allocation heuristics we developed for the village search model are described. Our simulation results are shown in Section 5. Finally, in Section 6 we present our conclusions.

2. Related work

To the best of the authors' knowledge, the village search problem has not been previously studied and addressed except in our previous work.[1] There are numerous classes of problems that share concepts with the village search problem such as the knapsack problem,[6] the traveling salesman problem (TSP), and resource allocation problems. In the interest of space, we will only describe research that closely approximates our domain in terms of the modeling framework and the resource allocation heuristics.

In terms of the modeling domain, the research areas that are most similar to our work are: combat simulations,

vehicle routing, and traveling salesmen-type problems. Research efforts in these domains may have goals similar to our work (e.g. create accurate military simulations, determine near-optimal plans) and may use similar methods to solve their problems (e.g. using genetic algorithms, simulation). However, our work differs from these solutions in substantial ways.

Many of the works relating to military combat models focus on deterministic models[7–9] or models that use stochastic information for purposes such as assigning aircraft to roles or creating probability distributions of possible outcomes.[10,11] These tools tend to focus on models for training purposes or for strategic-level simulation. In general, they are not used for real-time tactical-level combat dynamic resource allocation and decision making. Our modeling environment differs from these simulations by providing a resource allocation using stochastic methods to model uncertainty and to determine robustness, thereby assisting military leaders in making operational decisions. In addition, our work is intended to function as a decision support tool which we evaluate in this paper using simulation. Our work is not a simulation tool in the sense of a training aid such as UCOFT (Unit Conduct of Fire Trainer).

The vehicle routing problem (discussed in papers such as those by Desrochers et al.[12] and Potvin[13]) has similarities to our work. This problem can have multiple teams (vehicles) that are assigned to multiple targets (pick-up/drop-off locations) they must service. Like the village search problem, constraints can be placed on the environment such as service areas (boundaries) and time windows for service. The optimization goal in the vehicle routing problem varies such as minimizing distance traveled, minimizing completion time, and minimizing monetary cost. Though similar in many ways to our domain, our solution models uncertainty, quantifies the robustness of resource allocations (a performance metric that we find useful in our environment), and incorporates service time at the nodes which is dissimilar to the movement times due to the separate distributions for movement and search rates.

With regard to resource allocation problems, the village search problem is also similar to the multiple traveling salesmen problem (mTSP). Like the mTSP, each target building (city) must be visited once by only one search resource (salesman). One difference between the village search problem and mTSP is that the village search problem incorporates time spent searching at the nodes into the problem statement; mTSP generally does not include time spent in the visited cities. In addition, there are constraints (e.g. phase lines, boundary lines) on the problem in the village search domain that are not included in the mTSP problem.

There has been extensive research into heuristic solutions for the NP-complete[14] TSP and mTSP but here we review only works that use genetic algorithms to find a solution because those are the research efforts that are most similar to our approach. The research by Tang et al.[15] is a constrained problem domain that considers time spent at a node but not distance between nodes. Unlike our work, their genetic algorithm uses deterministic values instead of stochastic information and is not concerned with the uncertainties and the robustness of the solution. In the research by Sabuncuolgu and Bayiz,[16] a global satellite survey network problem was transformed into an mTSP problem to find a minimal cost route between survey points using a cost matrix to define the cost of the edge links between nodes in the graph. In this domain, it is restricted to deterministic values and is not concerned with uncertainty and robustness. In addition, the problem does not have limiting constraints on the solution such as the boundary lines of the village search problem. Finally, the research of Yu et al.[17] transforms a multiple robot mine clearing problem into an mTSP problem that is solved using a genetic algorithm. Like other papers surveyed, this work uses deterministic values, the mine removal time (equivalent to the search time of target buildings) is not considered, and it does not consider uncertainty and robustness in its calculations.

In the field of dynamic resource allocation there are efforts that focus on modifying heuristics to adapt to dynamic changes during heuristic execution and works that attempt to account for uncertain environments before heuristic execution. Both types attempt to account for environmental uncertainty but do so using different techniques. A brief description of these techniques is contained in the following paragraphs.

The research of Mavrovouniotis and Yang[18,19] are examples of dynamic resource allocation heuristics that account for uncertainty in an environment by modifying elements of traditional evolutionary heuristics (e.g. ant colony optimization, genetic algorithms). They use concepts that deliberately introduce diversity into the heuristic instead of increasing pressure towards the current best solution. For example, instead of replacing a generation's worst individual with a copy of the best individual, it is replaced with a randomly generated individual. These methods do dynamically adjust to uncertainty during heuristic execution using new scalar information but they do not consider stochastic information like our heuristics.

An example of research that attempts to account a priori for uncertainty is Powell et al.,[20] which examines aircraft allocation to satisfy airlift demands. Their study uses dynamic programming methods to develop resource allocations. Unlike our work, they model the uncertainty in the environment as the set of all possible aircraft (search team) states and then solve the allocation problem using that information. In our model, we do not attempt to define these states because it would be computationally

intractable in our problem and instead use probability mass functions (pmfs) to describe the uncertainty.

The work of Chung et al.[21] attempts to optimize the search paths of multiple agents that are tasked with detecting multiple targets. In their environment, they incorporate uncertainty in the targets' location into their model and update this information dynamically as the search progresses. In our problem domain, the uncertainty is present in the search rates and movement rates of our teams (agents). In addition, our target locations (buildings) are known a priori and their detection is not required. Finally, their work does not consider elements such as boundary lines and phase lines. These elements have a major impact on the functioning of our heuristics.

Kanoh and Hara[22] also try to account for uncertainty by using a genetic algorithm to solve a vehicle routing problem where the travel time over road segments varies dynamically. For a given run of the heuristic, the genetic algorithm produces a solution using currently available data and predicted data for segments that do not have current data. This approach does not consider the full stochastic range of data values during its processing like our method does, but it does account for updated scalar information that is not available during static planning.

3. Background on robustness and the village search mission

3.1 Robustness concepts

The quality of a resource allocation is an important metric to planners. Developing a quantifiable value for this quality metric in an environment containing uncertainties is difficult. A simple, quantifiable metric such as completion time does not always result in robust resource allocations as shown by Ali et al.[4] The term robustness is used to describe how well a system can perform given such uncertainties. We define the allocation as robust if it meets its goals despite perturbations in specified parameters. Robustness can have a variety of meanings that are heavily dependent upon the problem domain and the problem solver. A standardized framework for defining the term robustness that results in a quantifiable metric was developed by Ali et al.[4] This robustness metric has been adapted to the problem of village search planning and used as the foundation for a mathematical model of the static village search environment by Maxwell et al.[1] In this section we provide a brief summary of the robustness procedure and how it is applied to this domain. For a detailed description, the reader is referred to Maxwell et al.[1] or Appendix A.

The robustness metric for a given resource allocation can be developed using the FePIA (Features, Perturbation parameters, Impact, Analysis) method,[4] where the following are identified: (1) the performance features that determine whether the system is robust; (2) the perturbation parameters that characterize the uncertainty; (3) the impact of the perturbation parameters on the performance features; and (4) the analysis to quantify the robustness. The FePIA method provides a formal mathematical framework for modeling the village search environment.

For the purposes of this study, there are m performance features, where each feature is the amount of time it takes a team to finish searching its assigned buildings. For the system to be robust, all search teams must complete before the mission deadline time.

The exact values of the perturbations parameters (system uncertainties) are unknown during the planning phase of a village search mission. There are many uncertainties that effect the village search completion time, but for this work, the perturbations parameters that are modeled are the team search rates and team movement rates for each search resource. Factors that affect these rates are coalesced into pmfs that represent their probability distribution. For example, the physical fitness of a group of soldiers will have a positive or negative impact on their team movement rate over a given terrain path. The bins in this pmf represent a range of dismounted ground movement rates. The value associated with each bin is the probability that the actual movement rate is within the bin's range.

Currently the necessary data to represent the search rates of particular buildings and accurately model the numerous uncertainties that influence the village search model have not been gathered because there is no existing mechanism to use this data; it is easier for humans to use scalar estimates than pmfs for manual calculations. If historical data was available for these rates, then the pmfs could be modeled more accurately. The result is that field validation of our model is not possible at this time. Accordingly, these pmfs (search rate and movement rate) are created using normal distribution functions with mean values based on rates found in military field manuals. Future work may consider other validation techniques, such as those discussed by Banks.[23]

The impact of the perturbation parameters on the performance features can be described mathematically. These values can have a positive or negative effect on the search completion time of a search resource. The combined effects of the perturbation parameters define the possible outcomes (e.g. the completion time of a search team) captured by the performance feature.

For the analysis step of the FePIA process, stochastic (probabilistic) information about the values of these parameters whose actual values are uncertain is used to quantify the degree of robustness. In brief, the search completion time for a search team i (ST_i) can be calculated by convolving the pertinent pmfs (i.e. ST_i's assigned building search completion time pmfs, ST_i's designated path

movement completion time pmfs) within each phase line area. It is assumed that the search teams have adequate supporting elements to operate independently within a phase line area. In addition, the perturbation parameters considered are independent with respect to the search teams and therefore the search team completion times are independent when evaluated within a phase line area. The completion time for a particular phase line area is a joint cumulative mass function (cmf) that is the probability of all search resources completing the search of their assigned buildings for that phase line area. The probability that all search teams have completed searching their buildings by time T (where T is the upper limit of the time bin) is $\mathrm{P}(CT_1 \leq T) \cdot \mathrm{P}(CT_2 \leq T) \cdot \ldots \cdot \mathrm{P}(CT_n \leq T)$, where CT_i is the completion time of search team i. Finally, the convolution of the phase line area completion time pmfs (all phase line areas combined) is performed resulting in a single completion time pmf that represents the distribution of mission completion times. This completion time pmf is transformed into a cumulative mass function and evaluated at the MDT using integration resulting in the probability that all search teams will complete prior to the MDT. This probability is the stochastic robustness metric (SRM) for this domain.

Thus, for a given resource allocation of search teams to target buildings, the SRM provides a quantitative value for the robustness of the allocation. Therefore, a set of possible allocations can be searched to determine the allocation that is most robust via the comparison of SRM values. The application of this metric in the dynamic village search environment is discussed in the next section.

3.2 Dynamic village search

The dynamic village search environment consists of search teams, target buildings, a statically created resource allocation plan, and the completion time pmfs for the search teams. Frequently the MDT is fixed by higher headquarters and therefore a SRM of 1 is often not obtainable. It is assumed that an implemented static resource allocation will meet a minimum SRM threshold value (e.g. 40%). Allocations with a SRM below the threshold SRM would realistically have their MDTs adjusted or the mission would be cancelled due to its lack of feasibility. Once a mission begins, dynamic events can occur that cause the SRM to increase or decrease. Some events (e.g. the addition of a target building, loss of a search resource) necessitate the dynamic development of a new plan. Some events (e.g. change in MDT, weather factors that decrease the movement rates of the search teams) have less obvious effects and the need for a dynamically created new plan must be evaluated before deciding to implement the new plan. In this paper we focus on two cases where evaluation must be done prior to implementation: first (Case 1) we test our dynamic heuristics against the effects of cumulative delays; and then (Case 2) we consider a situation where a significant delay occurs at a single point.

Regardless of the case, our method involves evaluating potential resource reallocation plans each time a building search is completed. At that moment, completion time pmfs can be updated using actual time values for portions of the plan that have already been conducted providing a more accurate overall completion time pmf for the system. In the reallocation process, buildings not considered for reallocation are: all completed buildings; all buildings currently being searched; and the next building to search for the triggering search team (i.e. the search team that just completed a building search). The remaining unsearched buildings are considered for reallocation. We also assume that there is an overhead cost in terms of time for implementing a new allocation. This overhead factor accounts for time due to factors such as plan dissemination and subordinate unit planning. This overhead cost is considered in our evaluation of the efficacy of a proposed reallocation. New allocations that do not have a better SRM than the current allocation when the overhead cost is added are not implemented.

A major consideration in our heuristic design is its computation time. Because this is a dynamic environment, the time available to run reallocation heuristics is significantly shorter than what is available in the static planning domain. The heuristics' execution time is limited to the time available between building search completions across all search teams. Building searches generally take 30 minutes to several hours. As the problem size grows in terms of the number of search teams, the time between teams completing buildings will decrease further. As a result, our heuristics are designed for rapid execution (they run concurrent with building searches) and the ability to produce a robust result within the time available.

4. Dynamic resource allocation heuristics

4.1 Overview

In this section, we describe three new heuristics we have designed for use in the dynamic village search environment. These heuristics are derived from concepts known in the computing systems research literature.[3,19,24] The simulation results and the exact values for heuristic variables for evaluating these heuristics are found in Section 5.

4.2 Dynamic min–min heuristic

The dynamic min–min as shown in pseudocode 1 is inspired by the original two-phase greedy heuristic as developed by Ibarra and Kim.[3] In the dynamic version, only the reallocation eligible buildings in the boundary line area of interest are considered for reallocation. To

determine the boundary line area of interest, the heuristic updates the completion time pmfs for each search team by removing non-realizable bins (i.e. bins with completion times earlier than the current time) and renormalizing, then finding the search team with the highest mean completion time (ST_{max}). The boundary line area to which ST_{max} belongs is the boundary line area of interest (*BLI*). In the first phase of the heuristic, the minimum completion time building (including the time to move between target buildings) in the set of reallocation eligible buildings is identified for each search team. In the second phase of the heuristic, the *ST*/building pair with the minimum completion time based on mean values is scheduled and the building is removed from the reallocation eligible set. For this heuristic, mean values are used instead of entire pmfs to keep the heuristic fast and simple. This has been shown to be an effective heuristic in many environments as shown in the work of Braun et al.[24] The two phases of the heuristic are performed for each phase line area and search team in the *BLI*.

The dynamic min–min heuristic is fast and deterministic and thus can easily provide solutions in a time-constrained environment. Because the dynamic min–min is a greedy heuristic, it cannot compete with global search heuristics in terms of solution quality and is included in this study for the sake of comparison.

4.3 Completion time reduction heuristic

The completion time reduction heuristic (CTR) shown in pseudocode 2 attempts to improve the current allocation using iterative single building moves and single building pair swaps. The heuristic operates on reallocation eligible buildings within the boundary line area of interest. The heuristic then conducts a two-phase operation constrained by iterations limits. These limits constrain the number of building moves evaluated ($moves_{lim}$) and the number of building swaps evaluated ($swap_{lim}$) per iteration and the number of combined move/swap pairs executed ($iteration_{lim}$).

The first phase of the heuristic is the move phase where all single building moves from ST_{max} to the other *ST*s in the *BLI* are considered. As shown in Figure 2, this evaluation considers not only the reassignment of a building from ST_{max} to another *ST* but also the ordering of the building within the destination *ST*'s allocation. The single building move, if any, that results in the maximum mean completion time decrease that is less than the current mean completion time is recorded. The completion time pmfs for all *ST*s in the *BLI* are then updated and ST_{max} is revaluated. This process is repeated for $moves_{lim}$ iterations or until an iteration occurs where no building move is made. Upon completion of the phase, the resulting allocation, if any, is accepted if the completion time mean is less than the original completion time mean minus the overhead cost factor.

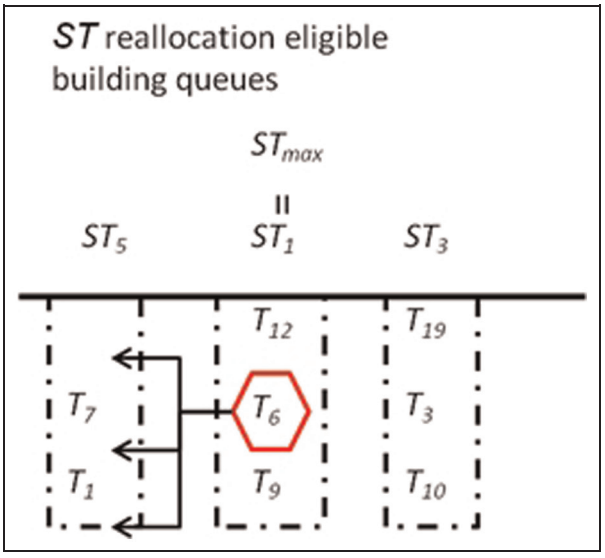

Figure 2. Completion time reduction heuristic move phase with building T_6 evaluated in ST_5's allocation queue in all possible orderings. Here ST_i represents a search team *i* and T_j represents a target building *j* in order of search for a search team.

The second phase of the heuristic exhaustively considers all single building pair swaps between ST_{max} and the remaining *ST*s in the *BLI*. To reduce the execution time of this heuristic, the swapped building is inserted into the destination *ST*'s allocation order in the position that results in the minimum time traveled between the swapped building and the building prior to it in the allocation ordering (Figure 3). The single building pair swap, if any, that results in the maximum mean completion time decrease that is less than the current mean completion time is recorded. The completion time pmfs for all *ST*s in the *BLI* are then updated and ST_{max} is revaluated. This process is repeated for $swap_{lim}$ iterations or until an iteration occurs where no building swap is made. Upon completion of the phase, the resulting allocation, if any, is accepted if the completion time mean is less than the original completion time mean minus the overhead cost factor.

4.4 Dynamic Genetic Algorithm

The Dynamic Genetic Algorithm (DGA) is a modified version of the village search genetic algorithm (VSGA) presented by Maxwell et al.[1] It differs from the static VSGA in its construction of its chromosomes, the population size, and the usage of the operators on the chromosomes. Though there are many research efforts such as that by Alcaraz and Maroto[25] that use genetic algorithms in a variety of environments, the operators for the static

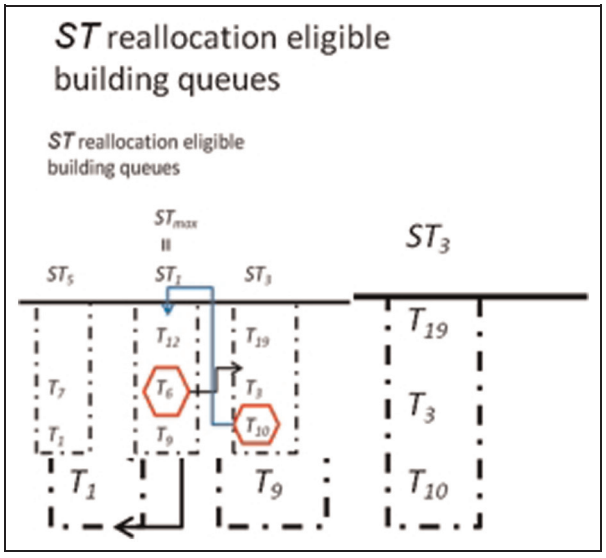

Figure 3. Completion time reduction heuristic swap phase with buildings T_6 and T_{10} swapping search teams and inserting into the destination ordering that results in the minimum movement time between buildings. Here ST_i represents a search team i and T_j represents a target building j in order of search for a search team.

bldg	1	14	0	28	31
ST	0	4	4	0	4

Figure 4. DGA strand representing search team to building assignments and the scheduling order. The scheduling order for an *ST* is read from left to right (e.g. ST_0 first searches building 1 and then building 28).

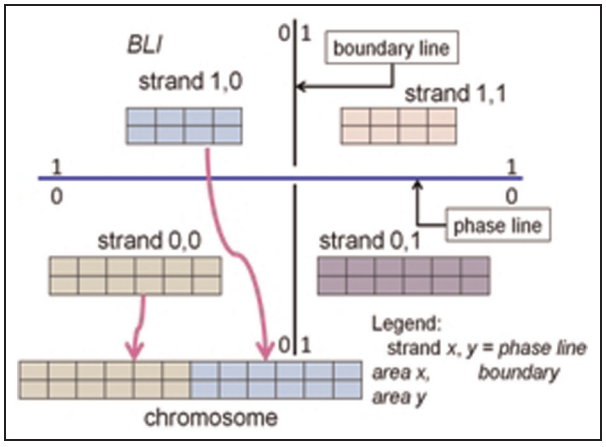

Figure 5. Example DGA chromosome with the *BLI* indicated and the current state of the search in phase line area 0. For the example *BLI*, only strands 0,0 and 0,1 are used for the DGA chromosome operations.

and DGAs are inspired by the work of Wang et al.[26] that demonstrated their efficacy in computing resource allocation environments. In the DGA, the population's chromosomes are composed of ''strands''. A strand contains information about a search plan that represents reallocation eligible target buildings, their assigned search teams, and the scheduling order of the target buildings for a given boundary/phase line area (Figure 4). The number of strands in a chromosome is determined by the number of phase line areas from the current state of the search until the end of the search inclusive (Figure 5). For purposes of evaluating the fitness of an allocation, strands from all boundary line areas are used. However, the DGA heuristic only uses the strands in the *BLI* when conducting its crossover and mutation operations. Finally, the size of the population for the heuristic is determined experimentally to be 40.

The basics of the DGA are outlined in Pseudocode 3 and described here. First, the number and size of the strands is calculated based upon the current state of the search (i.e. the current phase line area and the buildings eligible for reallocation). The heuristic then uses the current allocation (mission plan) as the basis for a seed chromosome. The other $p-1$ chromosomes in the population are generated randomly. The DGA uses stochastic universal sampling (SUS) during the selection phase (Pseudocode 1, step 6) for the next population. In this technique, each member of the population is allocated a section of a virtual roulette wheel in proportion to its fitness and the next population is selected in one ''spin'' of the virtual roulette wheel using p uniformly placed markers.[27] In each generation, offspring chromosomes are generated by subjecting the current population of chromosomes (using a chosen probability) to crossover and mutation operators that operate on the matching and scheduling of the search team/target building pairs (Pseudocode 1, steps 7–17). These are defined as the probability of scheduling crossover (P_{SC}), the probability of scheduling mutation (P_{SM}), the probability of matching crossover (P_{MC}), and the probability of matching mutation (P_{MM}). Then each chromosome in the population is evaluated using the stochastic robustness metric as the fitness function. Finally, the set of offspring is merged with the current population (Pseudocode 1, step 18) and the next generation is evaluated starting with SUS selection to limit the population size to a fixed value. The crossover operators in the DGA are the scheduling and the matching crossovers. The operators function on two randomly selected parent chromosomes from the population and they produce two offspring chromosomes. The crossover operation is

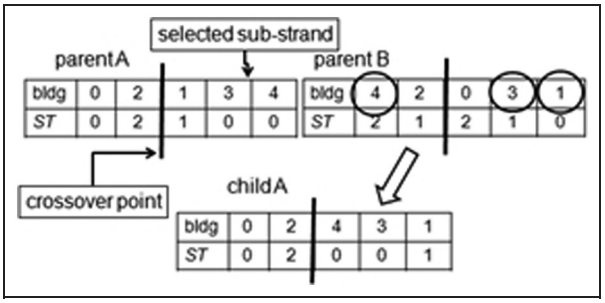

Figure 6. DGA scheduling crossover example.

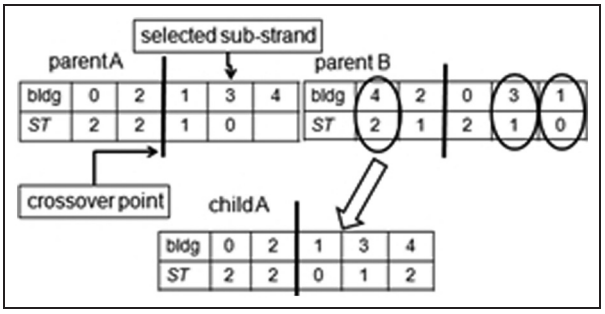

Figure 7. DGA matching crossover example.

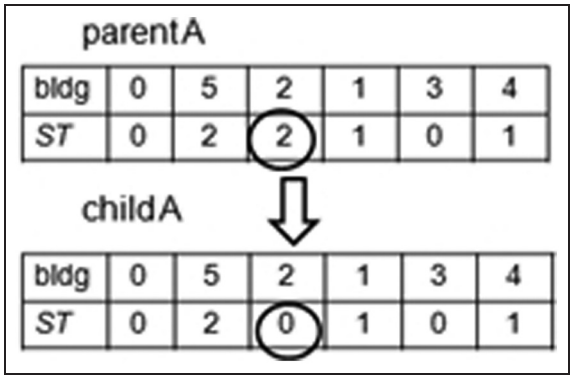

Figure 8. DGA matching mutation example.

performed on the same strand within each parent chromosome. For example, a crossover operation could be performed on strand 0,0 (Figure 5) on both parent chromosome *A* and parent chromosome *B*. Crossover (and mutation) operations can be defined to process multiple strands in a chromosome in a strand simultaneously, however in this work only one strand is operated upon. However, in this work, only one randomly selected strand per chromosome pair is operated upon in a given generation because studies by Maxwell et al.[1] show a significant

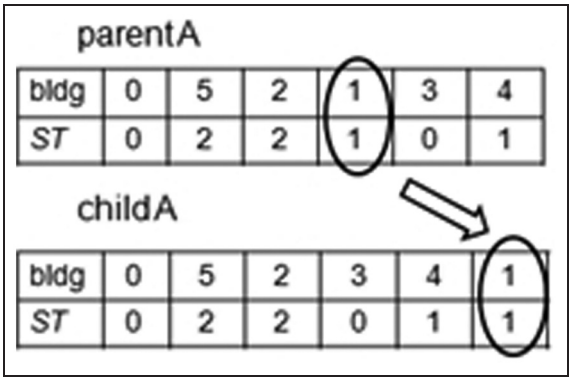

Figure 9. DGA scheduling mutation example.

increase in the computational load for a small increase in the robustness of the solution.

The scheduling crossover operation operates as follows (Figure 6). A single crossover point is randomly chosen and then the sub-strand to perform the crossover on is randomly chosen. Unlike crossover operators described by Wang et al.[26] that function on the ''right'' portion of the parent chromosomes, the VSGA crossover operator chooses the ''left'' or ''right'' sub-strand of the strand for crossover. Next, the scheduling order of the target buildings in the selected sub-strand of parent chromosome *A* are re-ordered to match the scheduling order of parent chromosome *B*. The operation is performed again with the parents' roles reversed.

The matching crossover operates similarly to the scheduling operator. A single crossover point is randomly chosen and then a sub-strand is randomly chosen. Each target building within the chosen sub-strand of parent *A* is assigned the search resource it has in parent *B* (see Figure 7). The operation is then repeated with the parent chromosomes reversed.

Similar to the crossover operators, the mutation operators function on one strand within a chromosome. Again, the operators can be performed on more than one strand but as with the crossover operators it has been limited to one strand for the work described in this paper. Examples for the matching and scheduling mutation operators are shown in Figures 8 and 9. The scheduling mutation operator begins by randomly selecting a target building/search team pair to reschedule. Next, it randomly selects a new order position in the strand. It then inserts the target building/search team pair at the newly selected destination creating a new scheduling order for that strand.

The matching mutation operator begins by randomly selecting a target building/search team pair to mutate. The operator then randomly selects a new search team from the set of search teams operating within the given boundary line area. The selected search team is then assigned to the target building creating a new matching.

5. Simulation results

5.1 Overview

In this section we first discuss the operation of our simulation environment and the parameters used in its execution. Following that discussion, we present the results of our simulations. These results include an evaluation of the heuristics with and without attempts to reduce the computational load through use of triggers that we describe later.

5.2 Simulation set-up

The results discussed here are based on Monte Carlo simulations of village search missions. An event-driven simulator was created that uses statically created allocations from Maxwell et al.[1] to conduct trials where search teams search their assigned target buildings. One complete trial simulates all search teams completing in order the search of their assigned target buildings with an associated search completion time. In addition to the static allocations, information about the search teams, target buildings, and road network is provided to the simulator. The simulator operates by randomly sampling, using a uniform distribution, the search teams' movement rate, and search rate pmfs to determine the duration of an event. Monte Carlo simulations are used as follows.

When a building search is completed by a search team and a trigger condition (discussed later) is met, one of the dynamic replanning heuristics described in Section 3 is invoked. The full information contained in the input pmfs is only used (through convolution) when a dynamic replanning event occurs. If the new dynamic allocation is rejected, then the simulation continues as normal. If the new dynamic allocation is accepted, then the simulation creates a branch of the simulation that uses the new allocation while continuing to simulate the execution of the static allocation. The simulation continues until all teams have completed searching their assigned buildings in both the original static allocation and any branch dynamic allocations.

The simulation results are based on four different test village search scenarios with five (scenarios 1, 2, and 3) or six (scenario 4) search teams, one phase line, and one boundary line. The scenarios were selected to exhibit the characteristics of real world village search missions using the combat experience of one of the present authors. We did not generate random search patterns or random number of search teams because they may not generate problem scenarios that are typical of real searches. The set of search teams consists of one military working dog team and with the rest being human only teams. In our experience, the dog teams conduct searches faster and thus have a higher search rate. This difference is captured in the generation of search team search rate pmfs, i.e. dog team pmfs are generated using a higher mean search rate than the human search teams.

Figure 10. Screen shot from the RoPARs animation tool[1] showing the village structure used for scenarios 1 and 2 (city a) generated using ERSI shapefiles. The solid colored buildings shown are the target buildings for scenario 1.

The complexity of the search mission is a function of the number of target buildings and their location, along with the connectivity of the road network (i.e. the more path choices available to move between two buildings, the more complex the problem). The road network of a city is modeled as a graph where a node represents: (a) road intersections, (b) points where building entranceways meet a road, and (c) points at predefined intervals (e.g., every 200 m) for long uninterrupted roads. Scenario 1 is based on the city (city a) in Figure 10, with 30 target buildings and 66 road nodes. Scenario 2 uses the same city, but with a different phase line location, different boundary line location, and different target buildings, resulting in 50 target buildings and 80 road nodes. Scenario 3 is based on a second city (city b) in Figure 11, and has 24 target buildings and 64 road nodes. Finally, scenario 4 uses the same city as scenario 3 but with 40 target buildings and 85 road nodes. Table 1 summarizes this data.

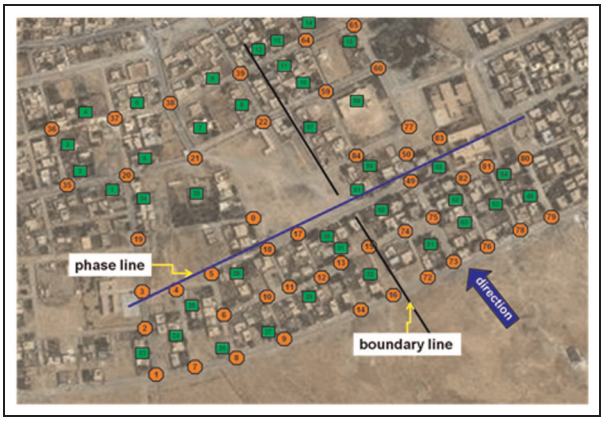

Figure 11. Village structure used for scenarios 3 and 4 (city b) based on satellite imagery of Kubaysa, Iraq. The actual target buildings (squares) and road nodes (octagons) shown are used in scenario 4.

Static allocations were created for each of the scenarios using the three static resource allocation heuristics used by Maxwell et al.[1] (minimum search heuristic, VSGA, and village search variable beam heuristic) and a new two phase greedy static resource allocation heuristic (maximum–minimum search heuristic (max–min search)) that is a variant of the minimum search heuristic. The minimum search heuristic is a two-phase greedy heuristic with random target building start points that attempts to minimize the mean search completion time of an allocation. The VSGA is similar in concept to the DGA but it searches a larger solution space to find good solutions in the static domain. The DGA has more information to conduct its search. For example, it knows the boundary area assignments for each search team. The village search variable beam heuristic is a branch and bound heuristic variant that searches for a solution using a reduced search set in each level of the search tree. The max–min search finds the maximum completion time building/*ST* pair for each *ST* in the first phase of the heuristic and then finds the minimum pairing from the first phase during the second phase of the heuristic. The building assigned is removed from the list of eligible buildings to be searched. Then the heuristic continues until all buildings are assigned. The four static allocation heuristics can be grouped into the simple or complex heuristic category. The minimum search heuristic and the max–min search heuristic are members of the simple category due to their greedy approach. The VSGA and the village search variable beam search heuristic are classified as complex heuristics. This grouping is done to provide a mechanism to evaluate the dynamic heuristics' performance. Each of the four scenarios had 100 trials run for each of the four static allocations for a total of 1600 trials per dynamic heuristic (800 simple trials, 800 complex trials).

Table 1. Simulation scenario parameters.

Scenario	City map	Number of target buildings	Number of road nodes
1	A	30	66
2	A	50	80
3	B	24	64
4	b	40	85

All of the heuristics used an overhead cost factor of 5 minutes. In the CTR and DGA heuristics, the simulation trials were run with the following settings. The CTR heuristic results are shown with an $iteration_{lim}$ of 30, and a $moves_{lim}$ and $swap_{lim}$ limit of 80 (a total of 4800 move/swap operations). These values were chosen to make the CTR heuristic have a run time comparable with that of the DGA. For the DGA, the heuristic was executed with a generation limit of a total of 1250 iterations or 350 iterations with no improvement in the best solution found. These values were chosen experimentally by varying the total iterations limit and the no improvements limit. In addition, the generational limits are designed to limit the total algorithm execution time to less than 10 minutes. The population size was set to 40 chromosomes based on experimental results. The probability of crossover and mutation were determined experimentally as follows: probability of scheduling crossover (P_{SC}) is 25%; matching crossover (P_{MC}) is 50%; scheduling mutation (P_{SM}) is 30%; and matching mutation (P_{MM}) is 75%. The scheduling probabilities are low because simply rearranging the scheduling order of an existing plan is less likely to produce better results. The most impact is to be gained through reassigning target buildings to new *ST*s, which is a matching mutation operation and thus that probability is higher. As discussed in other evolutionary algorithm research,[18,19] the generational memory effects of the algorithms due to aspects like elitism tend to drive solutions toward one portion of the search space. In a dynamic environment, this can have adverse effects and therefore the introduction of mutated chromosomes can help find new solutions. Here the high mutation rate provides the mechanism for the heuristic to find solutions that differ from the static solution.

5.3 Simulation results

The results shown in Figure 12 demonstrate the benefits of using a dynamic reallocation tool. During the trial runs, the reallocation heuristics were executed each time a building search was completed. This test set-up is called the 'always trigger' mode. The dynamic allocation algorithms use the search and movement rate pmfs of the search teams to calculate the SRM of the allocation. The

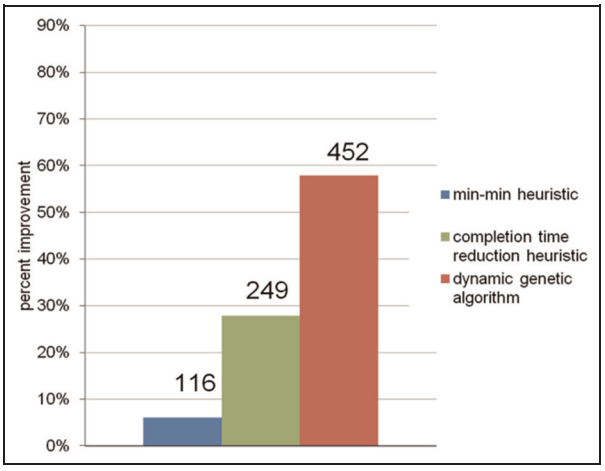

Figure 12. Percentage of trials improved (case 1) that meet the MDT using dynamically created resource allocations given that the statically created resource allocations (using all static heuristics) failed to meet the MDT. The actual number of trials with improvement out of 1600 is shown above each bar.

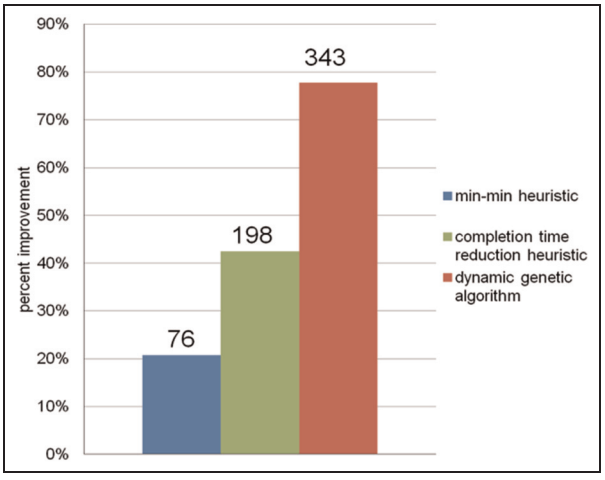

Figure 13. Percentage of trials improved (case 1) that meet the MDT using dynamically created resource allocations given that the statically created resource allocations (using greedy heuristics) failed to meet the MDT. The actual number of trials with improvement out of 800 is shown above each bar.

results depict the percentage of trials where the original static plan failed to meet the MDT but subsequently met the MDT using a dynamic reallocation plan. For example, the DGA results in Figure 12 show a percentage improvement of 57.87%. Of the 1600 trials conducted, the static allocation failed to meet the MDT in 781 trials. Of those 781 trials, the DGA successfully replanned the allocation and met the MDT in 452 of the trials. The trial results include cases where more than one reallocation plan was accepted during the course of the mission. The results shown, in most cases, accepted one reallocation plan but some had as many as three plans generated. The dynamic min–min heuristic performs poorly overall with all of its successful cases occurring when the static allocation heuristic was the max–min search heuristic. The CTR and DGA heuristics perform much better in comparison. Most of the successful cases for both of these algorithms occur when the static allocation heuristic was the minimum search or max–min search heuristic. Despite this, some success was found when the static allocation heuristic was more complex (village search variable beam search and VSGA). In these cases, the CTR had over a 14% improvement (51 successes in 347 reallocation attempts) and the DGA had a 32% improvement (109 successes in 340 reallocation attempts). The performance of the dynamic allocation heuristics against only the 800 trials from the greedy/simple static heuristics is shown in Figure 13 for comparison. The explanation for this disparity between Figure 12 and Figure 13 is that the complex static heuristics (i.e. village search variable beam, VSGA) consider the allocation's entire pmf instead of only the allocation to the evaluated point when conducting the static allocation plan and develop much better static solutions than the greedy heuristics.

As mentioned previously, there are many types of events that can occur that may cause a static allocation to miss the MDT. We tested our dynamic heuristics against a second case (case 2) for additional insight on how they would perform under more severe circumstances. In this case, we use the same scenarios as before but we randomly select a target building in the first phase line area for a long duration event (e.g. a delay due to an IED (improvised explosive device), discovery of a contraband cache). The time to search the selected building is calculated by first determining the building's search completion time cmf at 99.9% (i.e. essentially the maximum time to search this building). Then the slack in terms of time between the mean of the current allocation's updated completion time pmf and the MDT is added. The result is a long duration event at the building that forces a dynamic reallocation. As shown in Figure 14, the dynamic heuristics are still able to find solutions that meet the MDT, but are less successful than when cumulative delays are the main cause (see e.g. Figure 12). This is not a surprising result because in real-life, long duration events will dramatically affect a mission plan and will frequently cause plans to miss the MDT due to the nature of those events.

One of the main concerns in a dynamic environment is the execution time of the allocation heuristics. The dynamic min–min heuristic is a simple, greedy heuristic that executes quickly. The other dynamic heuristics are slower, as shown in Figure 15, but still are fast enough for the problem sizes explored.

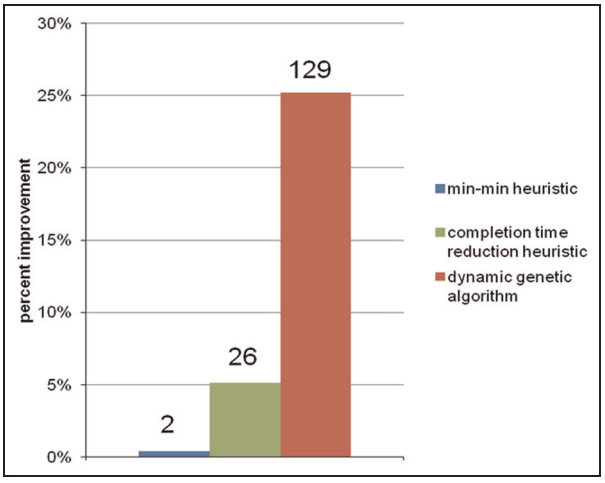

Figure 14. Percentage of trials improved (case 2) that meet the MDT using dynamically created resource allocations given that the statically created resource allocations (using all static heuristics) failed to meet the MDT. The actual number of trials with improvement out of 900 is shown above each bar.

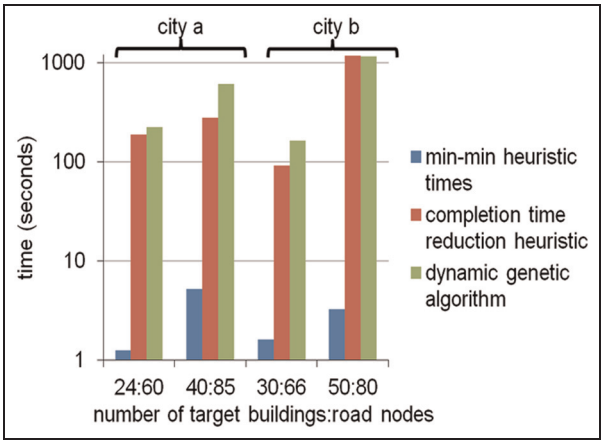

Figure 15. Average heuristic execution time (on a logarithmic scale) for an entire village search, averaged over 400 trials versus city complexity (number of target buildings: number of road nodes) for the min–min, completion time reduction, and DGA heuristics.

The dynamic heuristics would realistically be used at brigade-level units or lower and thus the number of search teams would be around 60 or fewer and the number of target buildings 900 or less. The heuristics are capable of meeting the time requirements for these real world problem sizes. In the 'always trigger' scenarios, the number of heuristic execution events equals the number of target buildings minus one. The execution times shown for the dynamic heuristics are based on non-optimized, serial code. Execution time improvement can be obtained through optimization and parallelization of the code. In addition, the CTR looping and DGA generational limits can be adjusted to conserve time. Finally, the CTR and DGA are ''anytime algorithms''[28] and maintain the current best solution found as it executes and therefore can produce that result if stopped before the heuristic completes due to time constraints.

A single invocation of a heuristic (heuristic execution time) only requires on the order of 1 minute to generate a new resource allocation. The execution time required for a heuristic is dependent on the current state of the village search when it is invoked. For example, the time to create a new allocation at the start of a search will be higher than it is to create a new allocation at the end of the search due to the number of reallocation eligible buildings. To evaluate the execution time of the heuristics (eliminating the overhead time of the simulator), it is necessary to run Monte Carlo simulations of the village search scenario and then repeat over many trials to acquire a large enough sample size to be statistically significant. Using the DGA heuristic, it currently takes over 36 hours to run 100 trials in the simulator. It is important to note that the majority of the computational expense is in the Monte Carlo simulations. Therefore, for larger scenarios, it is not currently practical to evaluate the execution time of these heuristics to a statistically significant level using Monte Carlo methods.

Another means of mitigating execution time issues is to use filters (triggers) to limit when dynamic reallocation can occur. Instead of evaluating dynamic reallocations each time the search of a single building is completed, a trigger can identify (using specific criteria) when to evaluate possible reallocation plans. This method does not decrease the execution time of a single run of a heuristic (i.e. a single mapping event), but instead reduces the number of occurrences of reallocation attempts by the heuristic. This can have an impact in situations where buildings are completed close to each other with regard to time and thus the amount of time to conduct reallocation is severely limited. The types of triggers that we explored are either based on properties of the static allocation or related to the MDT. The evaluation of a trigger must be executed quickly due to the time constraints of the dynamic environment and thus the triggers examined are simplistic in their design. Two triggers based on properties of the static allocation are: comparing the current completion time mean to the static completion time mean plus one standard deviation (*plan mean trigger*); and comparing the current SRM to 90% of the original SRM (*SRM decrease trigger*). In both of these triggers, if the current completion time pmf's mean is sufficiently worse than the static completion time pmf's mean, a dynamic reallocation is evaluated. A trigger based on the MDT was also evaluated that compares the

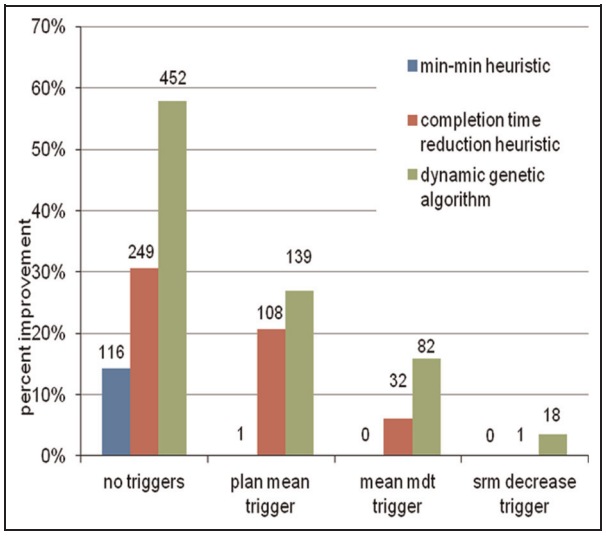

Figure 16. Comparison of dynamic allocation heuristics with no triggers and dynamic allocation heuristics with one of three trigger types: mean plus standard deviation; mean plus MDT; and SRM decrease. The actual number of trials with improvement out of 1600 is shown above each bar.

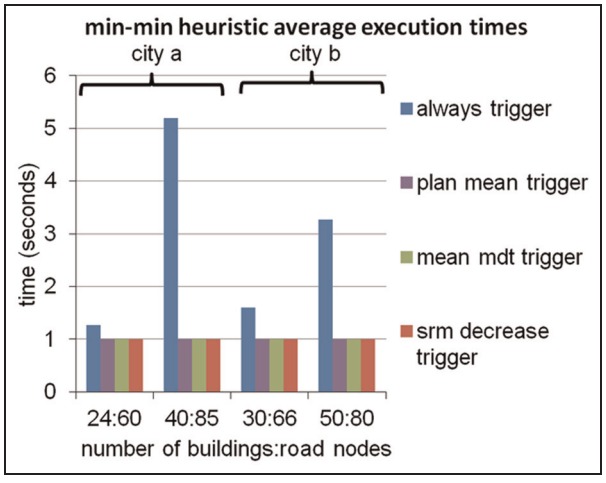

Figure 17. Execution time in seconds (on a logarithmic scale) for an entire village search, averaged over 400 trials versus city complexity (number of target buildings: number of road nodes) for the completion time reduction heuristic for four test conditions: no triggers, SRM decrease trigger, mean MDT trigger, and plan mean trigger on a logarithmic scale. The 95% confidence interval maximum values range from 3.9% of the mean for always trigger to 17.4% of the mean for the SRM decrease trigger of the mean.

current completion time pmf's mean to the MDT and conducts a dynamic reallocation attempt if the current mean is greater than the MDT.

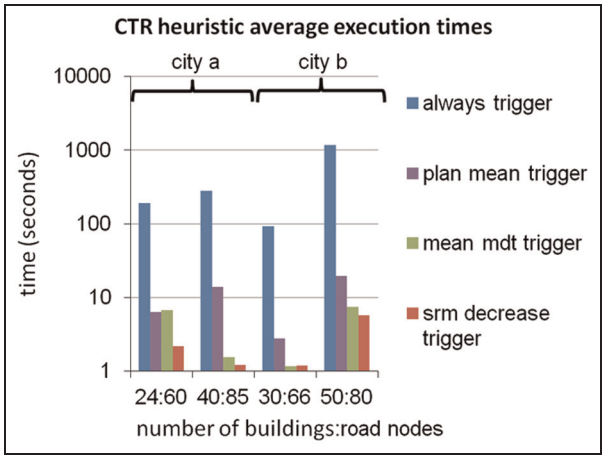

Figure 18. Execution time in seconds for an entire village search, averaged over 400 trials versus city complexity (number of target buildings: number of road nodes) for the min–min heuristic for four test conditions: no triggers, SRM decrease trigger, mean MDT trigger, and plan mean trigger on a logarithmic scale. The 95% confidence interval maximum values range from 2.0% for SRM decrease trigger to 9.9% of the mean for mean MDT trigger.

As Figures 16–19 show, there is a trade-off between the average simulation execution time for an entire village search and the heuristic's performance. Trigger-based filtering does obtain a lower computational load (most notable in the CTR and DGA algorithms) but also results in reduced heuristic performance. In Figure 16, the dynamic heuristics performance with no triggers is compared with the three triggering methods mentioned previously. The CTR algorithm experienced a net decrease in performance of over 20% compared with the no trigger scenario and the DGA heuristic had a net decrease in heuristic performance of over 30% for the same comparison. The min–min heuristic's performance decreases to zero percent indicating that triggering is not a good choice for this greedy heuristic. Figure 17 shows up to a two order of magnitude improvement in simulation execution time of the CTR algorithm using triggering. For the DGA heuristic, an order of magnitude improvement in simulation execution time was achieved (Figure 19). Figure 18 shows an order of magnitude improvement of the simulation execution time for the min–min heuristic but given the poor performance of the heuristic it is irrelevant. For the triggers tested, it is clear that triggering is not beneficial except for reducing the total system computational load. The poor performance of the triggers is due to their design in which reallocation is only considered after a designated performance metric is violated. At this point, it is often too late to recover. Tighter performance metrics can provide some trigger performance improvement for this case. Other

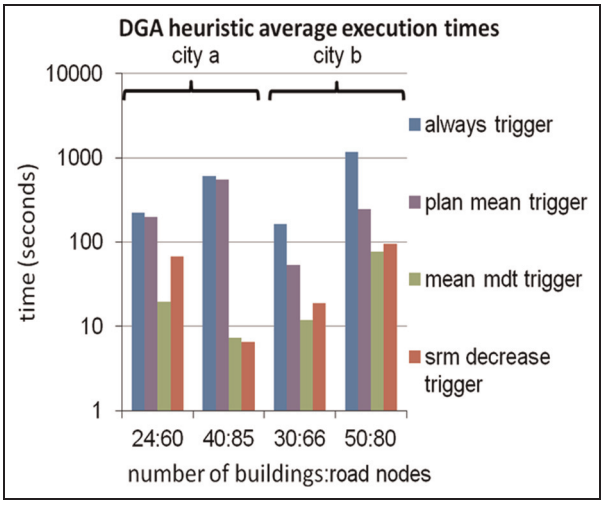

Figure 19. Execution time in seconds (on a logarithmic scale) for an entire village search, averaged over 400 trials versus city complexity (number of target buildings: number of road nodes) for the DGA for four test conditions: no triggers, SRM decrease trigger, mean MDT trigger, and plan mean trigger on a logarithmic scale. The 95 percent confidence interval maximum values range from 4.45% for mean MDT trigger to 36.9% of the mean for SRM decrease trigger.

situations where completion times are faster than the mean completion time based on the pmfs are not examined and thus an opportunity to discover a better schedule is missed. The fundamental problem with the designed triggers is that they do not take advantage of an improved situation. Differently designed triggers may improve the reallocation performance by considering these situations and by optimizing the performance metric of the trigger. Work of this nature may be considered in future work.

6. Conclusions

Conducting dynamic resource allocation during the execution of a military village search mission is beneficial especially when the time to develop a static plan is limited and hence the quality of the plan is limited (e.g. when using the static min search heuristic). These dynamic heuristics succeed because they are able to incorporate current state information that is unavailable prior to mission execution and thus produce more accurate results than static heuristics alone can achieve. Our results show that there are a significant number of times when these dynamic reallocations can produce new allocations that meet the mission constraints when the static plans fail; in particular 50% for case 1 (cumulative delays cause MDT violation) and 25% for case 2 (single point delays cause MDT violation) of the trials. When the static resource allocation heuristic is relatively fast and simplistic (e.g., greedy heuristics) the dynamic heuristics' ability to improve the resource allocation increases to over 75%. This is important when the static planning is done manually by human planners or a relatively poor static heuristic is used. Finally, our two best heuristics (CTR and DGA) automatically produce robust solutions that account for uncertainty in the environment in the limited time available, thus eliminating the need for time consuming staff work by planning officers. This can save military planners time and resources, and results in a better quality plan with respect to the modeled uncertainties.

Future work for this problem includes modeling other dynamic events that may require reallocation (e.g. the addition of target buildings, loss of a search resource), expanding the size of the test scenarios, and experimenting with other triggers. Further work can be done on changes to the DGA and the data inputs from the village search environment to allow it to constantly run in the background instead of running only when triggered. In this manner, the algorithm can constantly search for the optimal solution while responding to dynamic changes in the environment. Finally, research into the application of other techniques for modeling the environment could be done. Treating the village search scenario as a knapsack problem may create new insights into solving the resource allocation problem.

Funding

This research was supported by the National Science Foundation (grant number CNS-0905399), and by the Colorado State University George T. Abell Endowment.

Disclaimer

The views expressed in this article are those of the authors and do not reflect the official policy or position of the Department of the Army, DOD, or the U.S. Government.

References

1. Maxwell P, Maciejewski AA, Siegel HJ, Pfister G, Smith J and Friese R. Robust static planning tool for military village search missions: model and heuristics. *J Defense Model Sim* 2013; 10(1): 31–47.
2. Coffman EG (ed.). *Computer and Job-Shop Scheduling Theory*. New York, NY: John Wiley & Sons, 1976.
3. Ibarra OH and Kim CE. Heuristic algorithms for scheduling independent tasks on non-identical processors. *J ACM* 1977; 24(2): 280–289.
4. Ali S, Maciejewski AA, Siegel HJ and Kim J. Measuring the robustness of a resource allocation. *IEEE Trans Parallel Distrib Sys* 2004; 15(7): 630–641.
5. Shestak V, Smith J, Maciejewski AA and Siegel HJ. Stochastic robustness metric and its use for static resource allocations. *J Parallel Distrib Comput* 2008; 68(8): 1157–1173.

6. Chekuri C and Khanna S. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM J Comput* 2006; 35(3): 713–728.
7. Ahner D, Buss A and Ruck J. Using a low-resolution entity model for shaping initial conditions for high resolution combat models. In *2007 Winter Simulation Conference*, 2007, pp. 1344–1352.
8. Aydin M. *An Exploratory Analysis of Village Search Operations*. Master's Thesis, Naval Postgraduate School, June 2004.
9. Babilot M. *Comparison of a Distributed Operations Force to a Traditional Force in Urban Combat*. Master's Thesis, Naval Postgraduate School, September 2005.
10. Fulton LV, Lasdon LS, McDaniel RR Jr, et al. Two-stage stochastic optimization for the allocation of medical assets in steady-state combat operations. *J Defense Model Sim* 2010; 7(2): 89–102.
11. Popken D and Cox L. A simulation-optimization approach to air warfare planning. *J Defense Model Sim* 2004; 1(3): 127–140.
12. Desrochers M, Desrosiers J and Solomon M. A new optimization algorithm for the vehicle routing problem with time windows. *Operat Res* 1992; 40(2): 342–354.
13. Potvin J. Genetic algorithms for the traveling salesman problem. *Ann Operat Res* 1996; 63: 339–370.
14. Papadimitriou G. The Euclidean travelling salesman problem is NP-complete. *Theor Comput Sci* 1977; 4(3): 237–244.
15. Tang L, Liu J, Rong A, et al. A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex. *Eur J Operat Res* 2000; 124: 267–282.
16. Sabuncuoglu I and Bayiz M. Job shop scheduling with beam search. *Eur J Operat Res* 1999; 118: 390–412.
17. Yu Z, Jinhai L, Gouchang G, et al. An implementation of evolutionary computation for path planning of cooperative mobile robots. In *4th World Congress on Intelligent Control and Automation*, June 2002, pp. 1798–1802.
18. Mavrovouniotis M and Yang S. Ant colony optimization with immigrants schemes in dynamic environments. In *Parallel Problem Solving from Nature (PPSN XI)* (*Lecture Notes in Computer Science*, Vol. 6239). Berlin: Springer, 2011, pp. 371–380.
19. Yang S. Genetic algorithms with memory- and elitism-based immigrants in dynamic environments. *Evol Computat* 2008; 16(3): 385–416.
20. Powell W, Bouzaiene-Ayari B, Berger J, et al. The effect of robust decisions on the cost of uncertainty in military airlift operations. *ACM Trans Model Comput Sim* 2011; 9(4): 39.
21. Chung T, Kress M and Royset J. Probabilistic search optimization and mission assignment for heterogeneous autonomous agent. In *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 939–945.
22. Kanoh H and Hara K. Hybrid genetic algorithm for dynamic multi-objective route planning with predicted traffic in a real-world road network. In *10th Annual Conference on Genetic and Evolutionary Computation*, July 2008, pp. 657–664.
23. Banks J (ed.). *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*. New York, NY: John Wiley & Sons, 1998.
24. Braun T, Siegel HJ, Beck N, et al. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *J Parallel Distrib Comput* 2001; 61(6): 810–837.
25. Alcaraz J and Maroto C. A robust genetic algorithm for resource allocation in project scheduling. *Ann Operat Res* 2001; 102: 83–109.
26. Wang L, Maciejewski AA, Siegel HJ, Roychowdjury VP and Eldrige BD. Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach. *J Parallel Distrib Comput* 1997; 47(1): 8–22.
27. Blickle T and Thiele L. *A Comparison of Selection Schemes Used in Genetic Algorithms*. Technical Report TIK-Report Number 11, Version 2, CEN Lab, Swiss Federal Institute of Technology, December 1995.
28. Zilberstein S. Using anytime algorithms in intelligent systems. *AI Mag* 1996; 17(3): pp. 73–83.
29. Ali S, Maciejewski AA and Siegel HJ. Perspectives on robust resource allocation for heterogeneous parallel systems. In Rajasekaran S and Reif J (eds), *Handbook of Parallel Computing: Models, Algorithms, and Applications*. Boca Raton, FL: Chapman & Hall/CRC Press, 2008, pp. 41-1–41-30.
30. Bilbao J, Miguel AH and Kambezidis HD. Air temperature model evaluation in the north Mediterranean belt area. *J Appl Meteorol* 2002; 41: 872–884.
31. Bruhn JE, Fry WE and Fick GW. Simulation of daily weather data using theoretical probability distributions. *J Appl Meteorol* 1980; 19: 1029–1036.
32. Hill RD, Moate CP and Blacknell D. Estimating building dimensions from synthetic aperture radar image sequences. *IET Radar Sonar Nav* 2008 2(3): 189–199.
33. Prins C. A simple and effective evolutionary algorithm for the vehicle routing problem. *Comput Operat Res* 2004; 31(12): 1985–2002.

Appendix A. Village search robustness model

A.1. Overview

A quantitative mathematical model for a village search is presented in this appendix. To illustrate the problem, Figure A1 provides an example allocation for a village search scenario. Conducting the search are search resources ($SR = \{SR_1, SR_2, \ldots\}$) where SR_i can represent a human search team, a military working dog team, an explosive ordinance detachment, a robot, etc. In general, searches may be limited to only certain team types, and the search rates are dependent on the type. As shown in the figure, a village is composed of a set of *target buildings* ($T = \{T_1, T_2, \ldots\}$). Also shown are the *movement paths* (M_{ijk}) that have associated times to travel between buildings j and k for a *search resource* i (SR_i). Military planners attempt to allocate the resources (search resources) to the tasks (building searches) in a manner that will meet the given performance requirement (village search mission

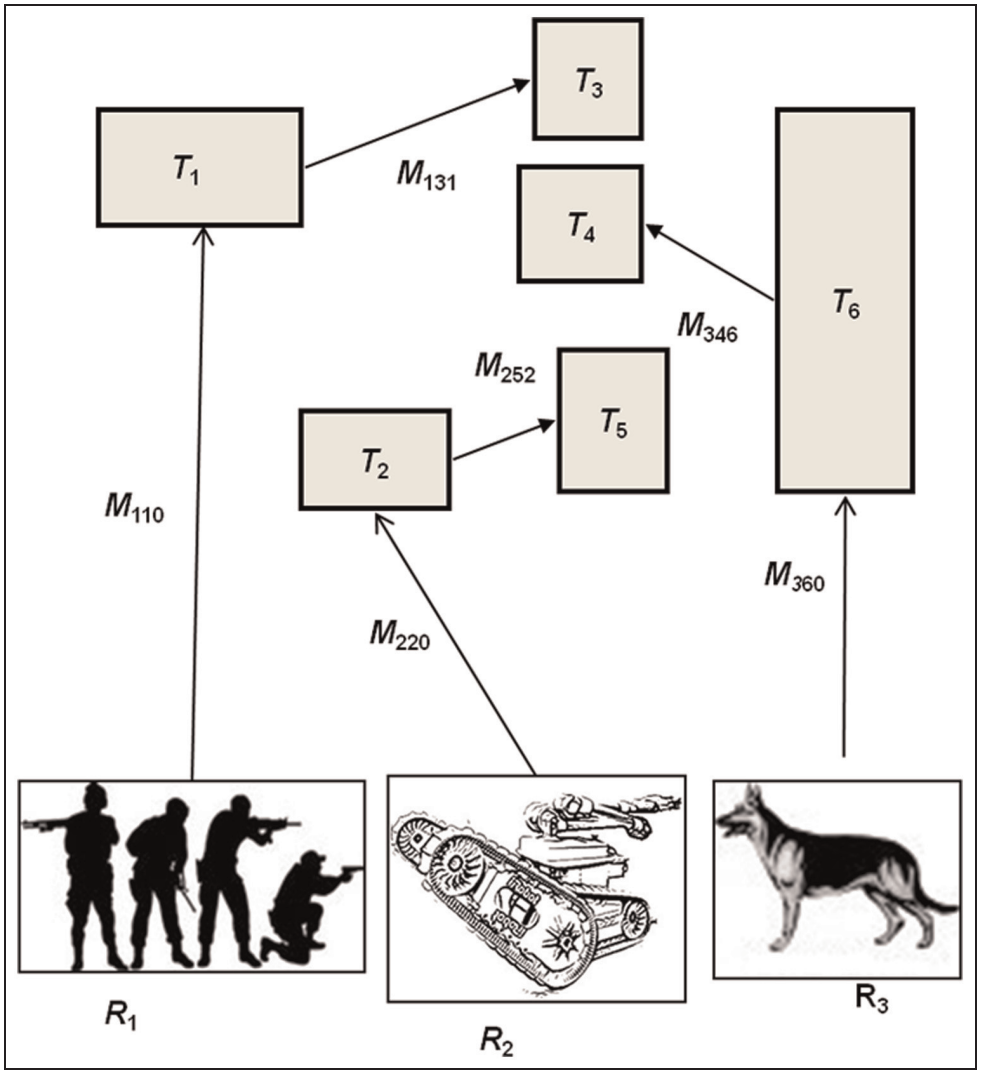

Figure A1. An example resource allocation for a village search with three search resources (human team, robot team, and a military working dog team) allocated to six tasks (building searches) with six movement paths.

deadline time). A model of this scenario must account for factors such as the search rate of the search resources, the movement time between structures, the ordering of the structure searches, and the perturbations discussed in the previous section.

To apply the robustness procedure to the village search scenario, one must answer the three robustness questions of Ali et al.:[29] (1) What behavior of the system makes it robust? (2) What uncertainties is the system robust against? (3) How is robustness of the system quantified?

A.2. Robust system behavior

The required behavior for the system to be considered robust may be one of or a combination of criteria, such as a specified time constraint is met, a specified percentage of casualties or less occurs, or no high value equipment is destroyed. The robustness criterion considered in this study is the MDT or time by which the mission must be completed.

A.3. System uncertainties

A system of this type will need to be robust against a variety of dynamic uncertainties that occur in the field, including the number of enemy combatants encountered, weather, engagement with explosive hazards, treatment and evacuation of casualties, changes to the availability of resources, and unanticipated animal (MWD) behavior. The village search model can incorporate any perturbation that can be described by a pmf. For example, future temperature values, future precipitation, and building sizes

have been modeled using a variety of distributions functions.[9,30–32] This work considers the variability in the *resource search rate,* σ_i, and variability in the *resource ground movement rate*, γ_i for search team i as the perturbations. These values are random variables with a distribution of rates. The base rates, σ_i and γ_i, are used as input variables to an overall completion time function that is defined later. The definition of these pmfs is a separate research problem and is not addressed here, but one way to develop them is by collecting data from training missions.

While not used in this work, our model can support the modification of σ_i and γ_i by perturbations such as temperature. If the temperature is higher than an accepted normal range (from which the nominal search and ground movement rates are determined) then the pmfs for searching and ground movement may shift in the negative direction reflecting a slower overall rate.

A.4. Quantifying robustness

To make determinations on resource allocations with regard to robustness, a quantitative method for calculating robustness is required. A list of notation used in this model is shown in Table A1. Applying the general stochastic model of robustness developed by Ali et al.,[4] this is defined as the probability that a user-specified level of system performance can be met. Let the *maximum search resource completion time* for the set of search teams be RCT_{max}. Then the robustness requirement is $RCT_{max} \leq MDT$.

A set of target buildings, has a corresponding set of areas, $A = \{A_1, A_2,\ldots\}$, that may include multiple floors. The resource's ground movement rate, γ_i, is the rate that the resource can move tactically along a movement path M_{ijk}. It is assumed that the waiting time for movement on movement path M_{ijk} due to multiple resources using the same path is negligible. Therefore, the completion time, C_{ijk}, for search resource i searching a given target building T_j and traversing movement path from building k is simply the area of the building divided by the search rate plus the distance of the movement path to the building divided by the ground movement rate. In this environment, multiple paths may exist between two buildings and each search resource moves along the paths at different rates. To efficiently determine the shortest traversal time path between two buildings, a stochastic all-pairs, shortest path algorithm is used.[33] This algorithm identifies the minimum traversal time road segment(s) between all building pairs for search resource i using road segment cumulative mass functions evaluated at a user selected probability level. Representing path fitness in terms of time instead of distance allows for the future incorporation of uncertainties that effect path traversal time such as encounters with improvised explosive devices.

Table A1. Village Search Notation.

Name	Description
SRM	Stochastic robustness metric, probability that the village search completion time is less than MDT
RCT_{ix}	Resource completion time for team i in phase line area x
RCT_{max}	Maximum search resource completion time for the set of search teams
A_j	Area of target building j
σ_i	Search rate for search resource i
C_{ijk}	Completion time for team i on building j moving from building k
CT_p	Completion time for team i on the pth building in its target set
γ_i	Ground movement rate for search resource i
MDT	Mission deadline time
M_{ijk}	Movement path from building k to building j for search resource i
n_{ix}	Number of target buildings for search resource i in phase line area x
P	Index into set of target buildings for search resource i in phase line area x
SR_i	Search resource i
T_j	Target building j
Φ	Number of phase lines
Θ_{ix}	Ordered set of target buildings for search resource i in phase line area x

The completion time function is subject to its input variables A_j and M_{ijk} and its perturbation parameters σ_i and γ_i. These are random variables. Given these random variables, the completion time for team i on target building j and its corresponding movement path have a distribution function defined as

$$C_{ijk} = f_{C_{ijk}}\left(A_j, \sigma_i, M_{ijk}, \gamma_i\right). \quad \text{(A1)}$$

Equation (A1) results in a random variable with a distribution consisting of building completion times. It is assumed that the pmf for this function will be created at run time using input values for the perturbation parameters (e.g. movement rates and search rates).

It is assumed that the search resources have adequate supporting elements to operate independently within a phase line area. Assume that there are Φ phase lines. This results in $\Phi + 1$ phase line areas. The effect of the phase line is barrier synchronization. In addition, the perturbation parameters considered are independent with respect to the search resources and therefore the resource completion times are independent when evaluated within a phase line area.

Let p be an index ($\{1,2,\ldots,n_{ix}\}$) into an ordered set $_{ix}$ of target buildings for search resource i in phase line area x.

Then, the p represents the pth entry in the set. In the following equation, we sum the building completion times for a search resource to obtain the resource completion time, RCT_{ix}:

$$RCT_{ix} = \sum_{p=1}^{n_{ix}} CT_p. \quad \text{(A2)}$$

Here, RCT_{ix} is the completion time for SR_i in phase line area x, where n_{ix} is the number of target buildings in its search set and CT_p is the completion time for the p^{th} building in the set $_{ix}$. Because we are working with discrete random variables to express the uncertainty in the system, the completion time is a pmf. Equation (A2) can be expressed as a pmf

$$f_{RCT_{ix}} = f_{CT_1} * f_{CT_2} * \cdots * f_{CT_{n_{ix}}}. \quad \text{(A3)}$$

Here $f_{RCT_{ix}}$ is the pmf for the completion time of SR_i in phase line area x.

The completion time distribution function for all search resources in phase line area x is

$$f_{PLx} = \max_{\forall i} f_{RCT_{ix}}. \quad \text{(A4)}$$

The result is a pmf for phase line area x that equals the maximum of the pmfs for all search resources.

To find the completion time pmf for all search resources over all $+ 1$ phase line areas, we convolve the phase line distribution functions as

$$f_{Comp} = f_{PL(\Phi+1)} * f_{PL\Phi} * \cdots * f_{PL1}. \quad \text{(A5)}$$

We then define the SRM as the probability that all search resources finish searching their target sets by the MDT:

$$SRM = P(Completion\ time \le MDT) = \int_{-\infty}^{MDT} f_{Comp}. \quad \text{(A6)}$$

Thus, for a given resource allocation of search resources to target buildings, the SRM provides the quantitative value for the robustness of the allocation. Therefore, a set of possible allocations can be searched to determine the allocation that is most robust via the comparison of SRM values.

Building on the general discussion by Shestak et al.,[5] the robustness metric can be utilized in two manners for the village search tool. In the first scenario, a military unit is tasked to conduct a village search within a given time constraint. Here the tool is used to calculate the resource allocation that has the highest probability of meeting the mission deadline time. For the second scenario, a military unit is tasked to search a village and requires an accurate estimate of the completion time to allow for the planning of supporting assets. In this case, the robustness metric is used to calculate the completion time for the mission with a given probability (e.g. 95%). In this paper, we only examine the first scenario though it is easy to convert the heuristics to accomplish the goals of the second scenario.

Author biographies

Dr Anthony A Maciejewski is the department head for the Electrical and Computer Engineering Department at Colorado State University.

LTC Paul Maxwell is an assistant professor in the department of Electrical Engineering and Computer Science at the United States Military Academy.

Dr Jerry Potter is a research faculty member in the department of Electrical and Computer Engineering at Colorado State University.

Dr HJ Siegel is the George T Abell endowed chair at the department of Electrical and Computer Engineering at Colorado State University.

Dr James Smith is a research faculty member in the department of Electrical and Computer Engineering at Colorado State University and a co-founder and chief technical officer of Lagrange Systems, Inc.