Robust static planning tool for military village search missions: model and heuristics

Paul Maxwell, Anthony A Maciejewski, Howard Jay Siegel, Jerry Potter, Gregory Pfister, Jay Smith and Ryan Friese

The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology 2013 10: 31 originally published online 15 July 2012
DOI: 10.1177/1548512911430218

Original article

Journal of Defense Modeling and Simulation: Applications, Methodology, Technology
10(1) 31–47
© 2012 The Society for Modeling and Simulation International
DOI: 10.1177/1548512911430218
dms.sagepub.com

Robust static planning tool for military village search missions: model and heuristics

Paul Maxwell[1,3], Anthony A Maciejewski[1], Howard Jay Siegel[1,2], Jerry Potter[1], Gregory Pfister[1], Jay Smith[1,4], and Ryan Friese[1,2]

Abstract
In the contemporary military environment, making decisions on how to best utilize resources to accomplish a mission with a set of specified constraints is difficult. A Cordon and Search of a village (a.k.a. village search) is an example of such a mission. Leaders must plan the mission, assigning assets (e.g. soldiers, robots, unmanned aerial vehicles, military working dogs) to accomplish the given task in accordance with orders from higher headquarters. Computer tools can assist these leaders in making decisions, and do so in a manner that will ensure the chosen solution is within mission constraints and is robust against uncertainty in environmental parameters. Currently, no such tools exist at the tactical or operational level to assist decision makers in their planning process and, as a result, individual experience and simplistic data tables are the only tools available. Using robustness concepts, this paper proposes a methodology, a mathematical model, and resource allocation heuristics for static planning of village searches that result in a decision-making tool for military leaders.

Keywords
decision making, resource allocation, robustness, stochastic simulation, village search

1. Introduction

Colorado State University's Information Science and Technology Center, ISTeC (http://ISTeC.ColoState.edu), is organizing the PAR (People–Animals–Robots) multi-disciplinary research laboratory[1] to study how teams of people, animals, and robots can be used together in new, synergistic ways in a variety of environments, including health care, search and rescue, and military operations. This effort is part of that PAR Lab.

On the modern battlefield, village searches are a frequent mission for military ground forces. In the current Global War on Terrorism environment, these searches are conducted daily to clear villages, capture insurgents, confiscate contraband, etc. In a village search problem, there are one or more target buildings that require searching. The search resources that conduct the search can consist of soldiers, *military working dogs* (*MWDs*), *explosive ordinance detachments* (*EODs*), military aircraft, *unmanned aerial vehicles* (*UAVs*), and electronic surveillance. The problem of assigning search resources to target buildings is in itself a computationally complex problem. Often this problem is made more difficult by the introduction of constraints on the solution, such as boundary lines (lines that demarcate allowable search areas for teams), phase lines (ground reference lines that act as synchronization barriers controlling the forward movement of the search teams), directions of advance (limits search direction), and time

[1]Electrical and Computer Engineering Department, Colorado State University, Fort Collins, USA
[2]Computer Science Department, Colorado State University, USA
[3]West Point, United States Army, USA
[4]Lagrange Systems, Boulder, 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

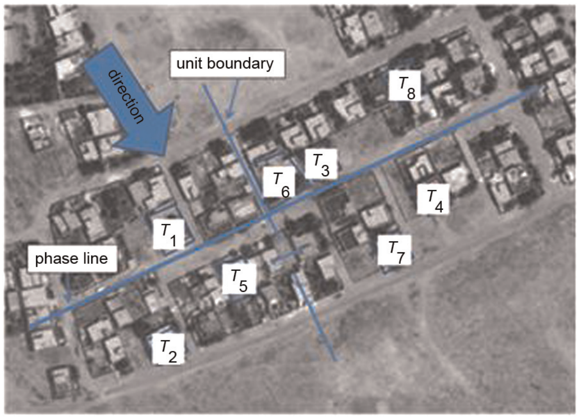


Figure 1. An example village search mission with eight target buildings (T_j), a unit boundary, a restrictive phase line, and a direction of advance.

deadlines. An example village search problem is shown in Figure 1.

When planning a village search, military staff officers must analyze the problem, allocate resources to the mission, estimate the amount of time required to complete the mission, plan for contingencies, and publish a mission plan. Given the diversity and the unpredictability of the battlefield along with the constraints of the mission, doing all of these things is an arduous task. In addition, the operating environment for these searches contain numerous uncertainties including, but not limited to, varying search times, encounters with the enemy, weather impacts on the search, and mechanical malfunctions. These uncertainties make finding exact solutions impractical. A resource allocation based on expected values will frequently not produce the best solution when these uncertainties are incorporated. It is therefore desirable to develop a near-optimal resource allocation for the village search problem that is robust against these uncertainties.

Despite the high frequency of this mission type, no comprehensive automated tools currently exist to assist military leaders in planning the execution of the searches. To develop their plans, officers must rely on the experience they have gathered during their years of service and the limited data tables provided in military Field Manuals[2,3] for factors such as ground movement rates. As a result, the quality of a plan, its ability to account for uncertainty, and the resulting confidence in its success varies dramatically.

An automated tool for village search planning for both training and operational purposes would greatly assist military leaders in their decision-making process in this environment where lives are at risk. To be effective, the tool requires the following capabilities: the ability to model the search area using automated digital maps of the search area, such as *Environmental Systems Research Institute* (*ESRI*) shapefiles; use probabilistic models to calculate search times; determine a good solution from multiple possible resource allocations; and execute within the time constraints of the planning process.

Here we introduce the *Robust People, Animals, and Robots Search* (*RoPARS*) planning tool. It improves mission planning by incorporating: (1) mathematical models that represent PAR and account for uncertainty in the environment; (2) robustness metrics that assist decision makers in managing complex processes with a high degree of certainty regarding tactical mission completion; and (3) heuristics based on those models and metrics that result in near-optimal, robust resource allocations (i.e. those that complete a village search by its deadline with high probability). The output of the tool is a recommended resource allocation for the village search mission that is expected to meet the provided constraints in a timely manner and reduce the risk for those involved in the mission.

The RoPARS tool automates many functions of the mission planning process, thus freeing planners for other tasks. Whether in training or during operational deployments, the planning staff could use the tool to automate select aspects of the time-consuming course of action development and wargaming portions of the decision-making process. Through a *graphical user interface* (*GUI*), the planners employ the tool to import standard ESRI shapefiles (http://www.esri.com) of the search area (many of which are currently available in the Urban Tactical Planner library – http://www.erdc.usace.army.mil), accept inputs regarding the plan (e.g. phase lines, boundary lines, search team types and compositions, target buildings), create a resource allocation using static allocation heuristics (i.e. minimum search heuristic, genetic algorithm, beam search heuristic), evaluate the performance of the allocation using quantifiable measures, and graphically display the resulting plan at a user-selected rate. The tool requires no additional operator training beyond the basic operational planning and graphics production skills used in the non-automated method and is thus easy to field. In addition, the tool can operate on standard issue laptops, preventing expensive procurement issues. Finally, the automated mission analysis tool uses stochastic information, is faster than human generated solutions, and is more reliable in terms of the robustness of its results than existing methods. Inevitably this will contribute to better, more informed decisions for military commanders in contemporary combat operations.

The contributions of this paper include: (1) robustness concepts for village searches; (2) a methodology with mathematical models for village searches; (3) resource allocation heuristics that produce robust mission plans; (4) evaluation and analysis of these heuristics through simulation; and (5) the integration of the model, robustness, and heuristics into the RoPARS tool along with a user interface. The methodology describes how to integrate uncertainty into a model of a village search. Its mathematical models allow for the objective evaluation of resource allocations. Finally, the resource allocation heuristics select an allocation that results in an acceptable plan based upon the computation time required and the amount of computing resources used.

With these goals in mind, Section 2 of this paper presents work related to the village search problem. Section 3 provides a discussion of the robustness metric developed by Ali et al.[4] and Saleh and Chelouah.[5] In Section 4, our model for a basic village search mission is discussed. The resource allocation heuristics we developed for the village search model are described in Section 5. An overview of our RoPARS tool GUI is provided in Section 6. Our simulation results are shown in Section 7 and, finally, in Section 8 we present our conclusions.

2. Related work

There are three categories of research that possess similarities to our work: combat simulations, vehicle routing, and traveling salesmen-type problems. These similarities can include the goals of the research (e.g. create accurate military simulations, determine near-optimal plans) and the methods used in the research (e.g. using genetic algorithms, simulation). However, as discussed in the following paragraphs, our work differs from these works in substantial ways.

Much work has been done in the field of military combat simulations. Many simulations are based on deterministic models,[6,7] although work has been done on stochastic models.[8–11] The purpose of these simulations generally fall into two categories: training aids for troops or strategic-level simulations for theater-level planning. Within some of these simulations, urban movement and combat at the soldier level is modeled, but it is not for the purpose of resource allocation and decision making. In addition, many of the urban or village simulation models rely on deterministic look-up tables for their input data or stochastic models that account for randomness in only the movement direction and combat strategy. The RoPARS tool differs from these simulations by providing a resource allocation that assists the military leader in making operational decisions. The RoPARS tool utilizes stochastic methods to model uncertainty and to determine robustness.

The vehicle routing problem (discussed in works such as Desrochers et al.,[12] Laportea et al.[13] and Potvin[14]) has similarities to our work. This problem can have multiple resources (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 travelled, minimizing completion time, and minimizing monetary cost. The problem is different from our domain in that we model uncertainty, quantify the robustness of resource allocations, and incorporate the service time at the nodes.

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; the 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.

There has been extensive research into solutions for the TSP[14,15] and the mTSP[16] due to their wide applicability and complexity. Here we review only works that use genetic algorithms to find a solution, because those are the closest comparisons.

The work of Tang et al.[17] models a steel rolling factory as a mTSP and uses a genetic algorithm to produce solutions. Similar to the village search problem, the steel rolling problem has constraints that reduce the number of valid solutions. The steel rolling problem considers time 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 robustness of the solution.

In Sabuncuoglu and Bayiz,[18] a genetic algorithm is used to produce solutions for a global satellite survey network problem. This problem domain was transformed into a mTSP where the salesmen are satellites and the cities are survey jobs for the satellites. The objective of the research is to find a minimal cost route between survey points using a cost matrix to define the edge cost. In this domain, it is restricted to deterministic values and is not concerned with robustness. In addition, the problem does not have limiting constraints on the solution, such as the boundary lines of the village search problem.

The work of Yu et al.[19] transforms a multiple robot mine clearing problem into a mTSP. In this research, multiple robots must move to multiple mines and remove those mines in a cooperative manner. The authors use a genetic algorithm to find a solution that minimizes the paths the robots traverse while removing all the mines. Like other works surveyed, this work uses deterministic values and does not consider uncertainty in its calculations. In addition, the mine removal time (equivalent to the search time of target buildings) is not considered in the problem.

There is much TSP and mTSP research to provide near-optimal solutions in a particular domain. The solution techniques vary in heuristic style and in choices, such as sequential versus parallel execution of the heuristics, but none of the works surveyed address robustness.

3. Defining robustness

We have defined robustness and a methodology to calculate the robustness of a resource allocation in Ali et al.[4] and studied it within a variety of systems.[20,21,33] We have adapted the concept of robustness to the problem of village search planning.

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 if 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.

The performance features are those measurable system attributes that can be compared against the robustness criteria. For a village search, this can be the time required for a team to finish searching its assigned buildings. That is, if there are m search teams, there are m performance features, where each feature is the time a team finishes searching its assigned buildings. For the system to be robust, all search teams must complete before the mission time constraint.

Perturbation parameters are the system uncertainties that may affect the actual mission completion time. These may include weather, estimation error in search area dimensions, variability in search resource movement rates, frequency and number of casualties, equipment losses, and number of enemy combatants encountered. Each of these elements can impact the actual mission completion time in a positive or negative manner, but the key point is that their actual values at the time of the mission are uncertain when planning is conducted. The system must account for these perturbations and recommend a resource allocation that is robust with respect to these uncertainties.

The impact of the perturbation parameters on the performance features can be described mathematically. For example, a stochastic model may be used to describe the effects of enemy combatants on the search of a given building. The collective effects of the uncertain perturbation parameters on the performance features must then be evaluated to find the most robust allocation of assets.

For the analysis step, stochastic (probabilistic) information about the values of these parameters whose actual values are uncertain is used to quantify the degree of robustness. The resulting *stochastic robustness metric* (*SRM*) is the probability that a user-specified level of system performance can be met. In this domain, the performance metric is the completion time for the search of all the target buildings.

Using these FePIA steps, the ***SRM*** for a village search can be determined. Once this is done, heuristics for planning robust resource allocations can be designed.

4. Village search robustness model

4.1 Overview

A quantitative mathematical model for a village search is presented in this section. To illustrate the problem, Figure 2 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 MWD team, an EOD, 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 deadline time (***MDT***)). 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 in Ali et al.[20] Namely: (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?

4.2 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.

4.3 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,

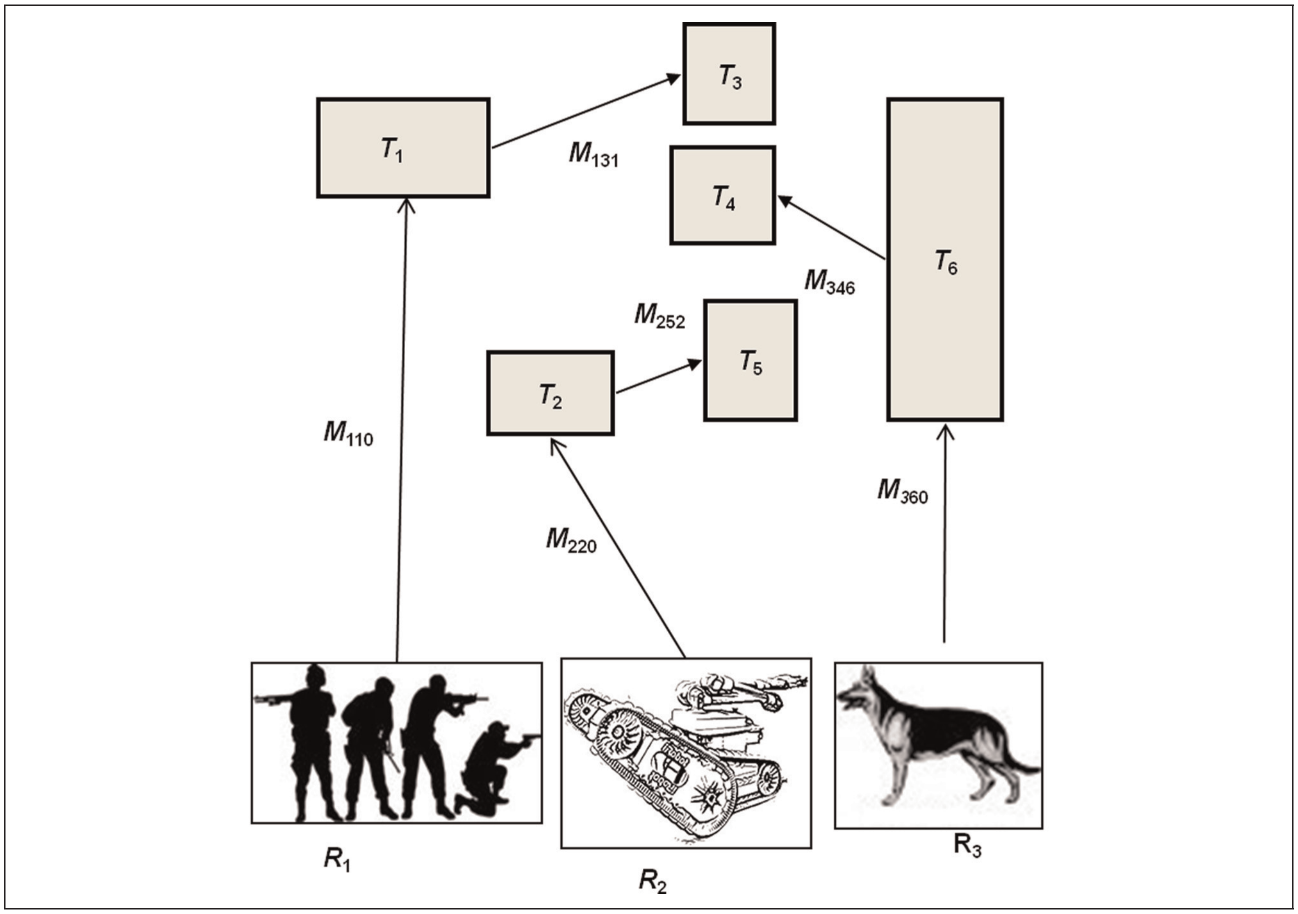


Figure 2. An example resource allocation for a village search with three search resources (human team, robot team, and a military working dog team) allocated to six tasks (building searches) with six movement paths.

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 *probability mass function* (*pmf*). For example, future temperature values, future precipitation, and building sizes have been modeled using a variety of distributions functions.[22–24] This paper 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 paper, 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.

4.4 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 1. Applying the general stochastic model of robustness developed in Saleh and Chelouah,[5] 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

Table 1. 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

movement path M_{ijk} due to multiple resources using the same path is negligible. Therefore, *completion time,* C_{ijk}, for search resource i searching a given target building T_j and traversing the 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.[26] 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.

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

$$C_{ijk} = f_{c_{ijk}}(A_j, \sigma_i, M_{ijk}, \gamma_i). \tag{1}$$

Equation (1) 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 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 Θ_{ix} of target buildings for search resource i in phase line area x. Then, the p represents the pth entry in the set. In Equation (2), we sum the building completion times for a search resource to obtain the *resource completion time,* RCT_{ix}. 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 pth building in the set Θ_{ix}:

$$RCT_{ix} = \sum_{p=1}^{n_{ix}} CT_p. \tag{2}$$

Because we are working with discrete random variables to express the uncertainty in the system, the completion time is a pmf. Equation (2) can be expressed as a pmf, as shown in Equation (3), where $f_{RCT_{ix}}$ is the pmf for the completion time of SR_i in phase line area x:

$$f_{RCT_{ix}} = f_{CT_1} * f_{CT_2} * \cdots * f_{CT_{n_{ix}}}. \tag{3}$$

The completion time distribution function for all search resources in phase line area x is shown in Equation (4). The result is a pmf for phase line area x that equals the maximum of the pmfs for all search resources:

$$f_{PLx} = \max_{\forall i} \; f_{RCT_{ix}}. \tag{4}$$

To find the completion time pmf for all search resources over all $\Phi + 1$ phase line areas, we convolve the phase line distribution functions, as shown in Equation (5):

$$f_{Comp} = f_{PL(\phi+1)} * f_{PL\Phi} * \cdots * f_{PL1}. \tag{5}$$

We then define the ***SRM*** as the probability that all search resources finish searching their target sets by the *MDT* (Equation (6)):

$$SRM = P(Completion\,time \leq MDT) = \int_{-\infty}^{MDT} f_{Comp}. \tag{6}$$

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 of Shestak et. al.,[27] 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 ***MDT***. 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, although it is easy to convert the heuristics to accomplish the goals of the second scenario.

5. Resource allocation heuristics

5.1 Environment

The village search mission environment is defined by its boundary lines and the assignment (grouping) of search resources to specific boundary line areas. Assuming that at least one boundary line is present, a village search problem solution space has many combinations of the number of search resources to boundary line area assignments to explore. For example, if there are five search resources and two boundary line areas, then there exist four valid grouping possibilities (one *SR* on the east side, four *SRs* on the west side (1,4), two *SRs* on the east side, three *SRs* on the west side (2,3), etc.). If more boundary lines exist (i.e. two boundary lines) and five *SRs* are used then a grouping tuple may be (2,2,1).

5.2 Minimum search heuristic

The minimum search heuristic was inspired by the original two-phase greedy heuristic.[28] It is modified to fit the village search domain and its constraints. It is a fast, deterministic heuristic and thus can provide valid solutions in a time-constrained environment.

The minimum search heuristic is used to find a solution for one specific grouping tuple. To find the best solution with the heuristic, the heuristic must be executed for all possible tuples. The heuristic is deterministic and will assign the same starting building locations for each execution if unguided. To expand the search area and improve the solution, the heuristic was modified to randomly select starting building locations for each search resource. This forces the heuristic to explore other solutions. The pseudocode for the minimum search heuristic is shown in Figure 3.

5.3 Village search genetic algorithm

The minimum search heuristic provides valid solutions but in general does not provide good-quality solutions due to its limited exploration of the search space and its greediness. The *village search genetic algorithm* (*VSGA*) compensates for these weaknesses by exploring the space more broadly.

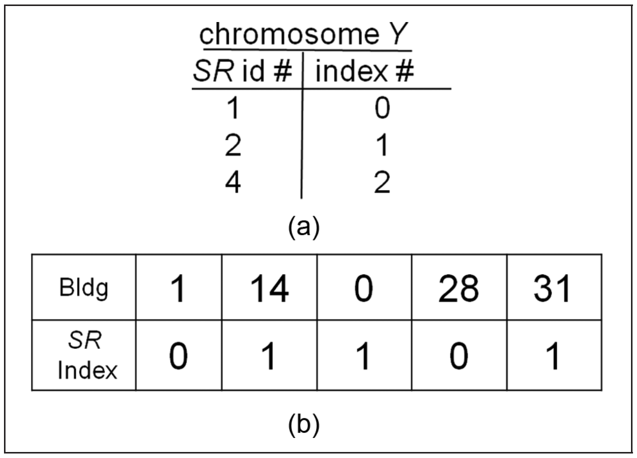
The VSGA is a modification of a classic evolutionary genetic algorithm. Its pseudocode is shown in Figure 4.

```
1.  for every combination of $y_i$ search resources assigned to boundary
    area $i$ do     (ex. <0,2,4>, <1,3,4>)
2.      for (number of trials $x$) do
3.          select random building starting location for
            each search resource
4.          for each phase line area $j$ and boundary area $i$
5.              for each search resource (SR) assigned to
                boundary area $i$
6.                  find minimum completion time (MCT) unassigned
                    building and associated road segment
7.              find MCT building/SR pair from line 6
8.              assign MCT building from line 7 to its matched SR
9.              remove MCT building from unassigned building list
10. output allocation with the highest Stochastic Robustness Metric
```

Figure 3. Minimum search heuristic pseudocode.

```
while stopping criteria not met
    1.  select next population using Stochastic Universal Sampling
    2.  for (number of chromosomes/2) do
        a.  randomly select two chromosomes
        b.  if (random < probability of crossover) do scheduling crossover
        c.  if (random < probability of crossover) do matching crossover
    3.  for each chromosome do
        a.  if (random < probability of mutation) do scheduling mutation
        b.  if (random < probability of mutation) do matching mutation
    4.  for each chromosome do evaluate fitness function
```

Figure 4. Village search genetic algorithm pseudocode.



chromosome Y

SR id #	index #
1	0
2	1
4	2

(a)

Bldg	1	14	0	28	31
SR Index	0	1	1	0	1

(b)

Figure 5. Village search genetic algorithm chromosome components: (a) search resource look-up table; (b) chromosome 'strand.'

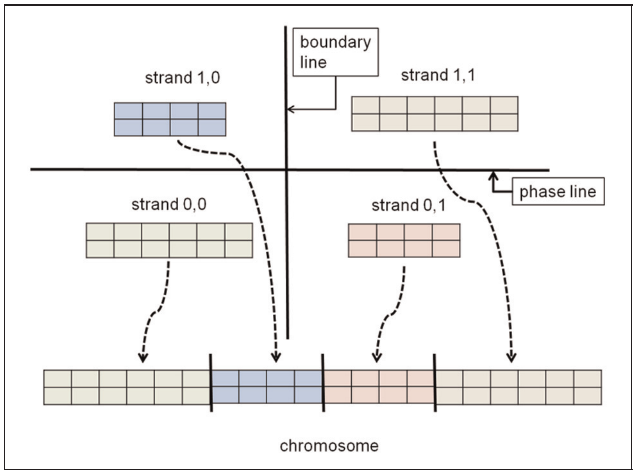


Figure 6. Village search genetic algorithm chromosome.

The VSGA uses the minimum search heuristic solution as a seed chromosome. The other chromosomes in the population are generated randomly ensuring valid chromosomes and that each possible search resource assignment combination is represented. During chromosome generation, a look-up table is created that cross-references *SR* index values to *SR* absolute reference values (Figure 5(a)). The reason for this table will be discussed later. The VSGA uses stochastic universal sampling for the selection of the next population. In this technique, selection bias with regard to the expected reproduction rate is avoided and the next population is selected in one 'spin' of the virtual roulette wheel.[29] In each generation, chromosomes are subjected (using a chosen probability) to crossover and mutation operators. Then each chromosome in the population is evaluated using the ***SRM*** as the fitness function. The details of the chromosome operators and the fitness function are described in subsequent paragraphs.

At its most basic level, a chromosome for the VSGA consists of multiple 'strands.' An example strand is shown in Figure 5(b). The strand is an array composed of target building number/search resource index pairs. The strand represents target buildings, their assigned search resources, and the scheduling order of the target buildings. The building number/*SR* index pair's position in the array defines a global ordering with pairs in the leftmost array position being first in time. The strand's length is determined by the number of target buildings within its boundary line and phase line area. The VSGA uses strands due to the constraints placed on the solution by the problem domain, such as boundary lines that limit crossover and mutation changes.

At the next higher level, the VSGA chromosome consists of one or more strands. The number of strands in a chromosome is determined by the number of phase line areas multiplied by the number of boundary areas. In Figure 6, an example chromosome is shown with its four component strands for a two-phase line area by two boundary line area village search. When the chromosome's fitness is evaluated, the strands are assembled as a whole into the chromosome and then the *SRM* is calculated.

The crossover operators in the VSGA are the scheduling and the matching crossovers. The operators function using two randomly selected parent chromosomes from the population and they produce two child chromosomes. In a particular generation, the number of crossover operations is less than or equal to half the number of population chromosomes. Examples for these operators are shown in Figures 7 and 8. Similar to classic genetic algorithms, the crossover operators use two parent chromosomes in their operation. In addition, the crossover operators function on the same strand within each parent chromosome. For example, a crossover operation could be performed on strand 0,0 (Figure 6) on both parent chromosome *A* and parent chromosome *B*. Crossover (and mutation) operations can be performed on more than one of the strands in a chromosome if desired. However, in this work, only one randomly selected strand per chromosome pair is operated upon in a given generation.

The scheduling crossover operation operates as follows. A single crossover point is randomly chosen and then the sub-strand to perform the crossover upon is randomly chosen. Unlike crossover operators described by Wang et al.[25] 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.

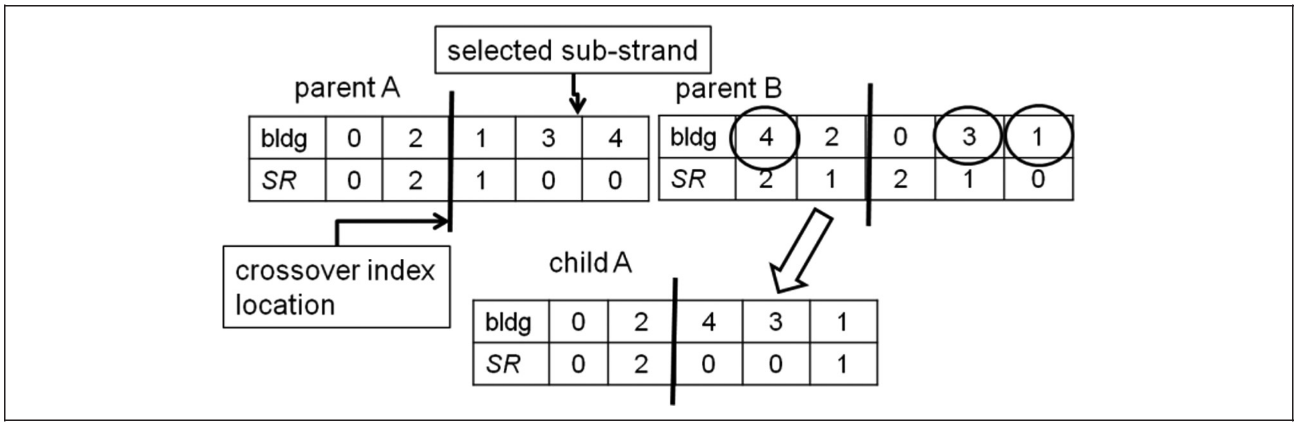


Figure 7. Village search genetic algorithm scheduling crossover example.

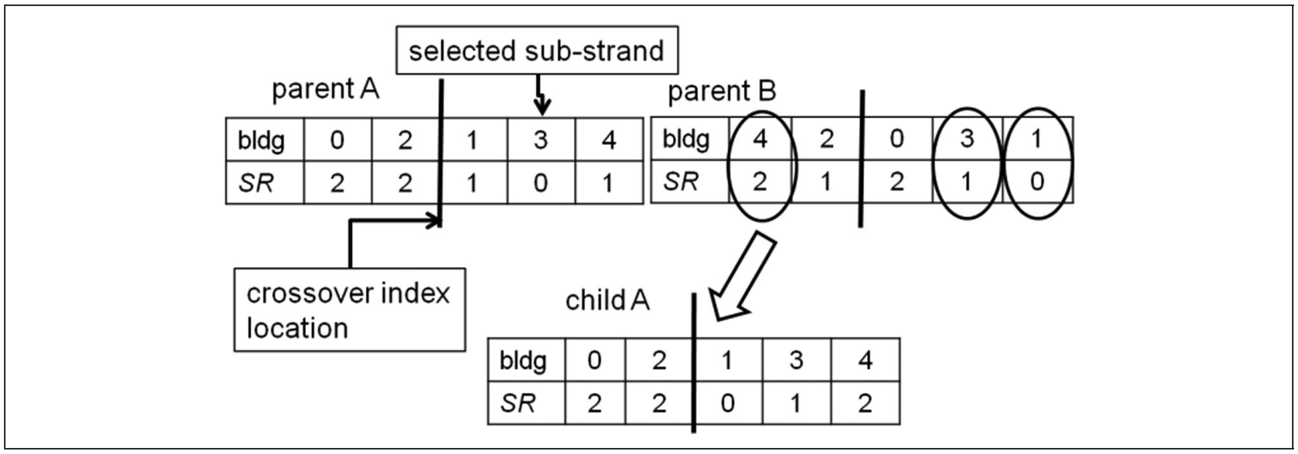


Figure 8. Village search genetic algorithm matching crossover example.

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*. The operation is then repeated with the parent chromosomes reversed (see Figure 8).

Similar to the crossover operators, the mutation operators function on a 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 example in this paper. Examples for the scheduling and matching mutation operators are shown in Figures 9 and 10, respectively.

The scheduling mutation operator begins by randomly selecting a target building/search resource pair to reschedule. Next, it randomly selects a new order position in the strand. It then inserts the target building/search resource 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 resource pair to mutate. The operator then randomly selects a new search resource from the set of search resources operating within the given boundary line area. The selected search resource is then assigned to the target building, creating a new matching.

All of the operators described in the preceding paragraphs operate in each generation. For each operator, a user-selected probability is used (i.e. probability of crossover, probability of mutation) to randomly apply the operator on the selected chromosomes. As with all genetic algorithms, an optimal solution is not guaranteed in a finite amount of time. In addition, its convergence rate and quality of solution is dependent upon implementation factors (e.g. population size, number of generations created, probability of crossover/mutation). As mentioned

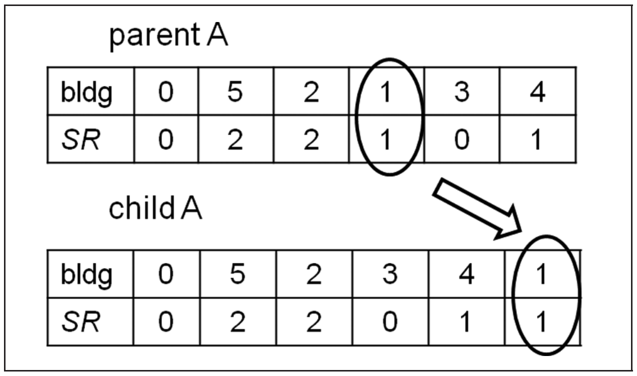


Figure 9. Village search genetic algorithm scheduling mutation example.

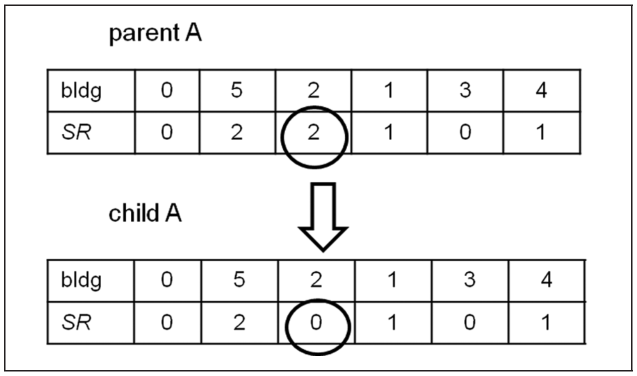


Figure 10. Village search genetic algorithm matching mutation example.

previously, through use of a look-up table (Figure 5(a)), the *SR* indices are associated with absolute *SR* identification numbers. Chromosome *SR* look-up tables are inherited from parent to child during crossover and mutation operations. We use index tables so that we can search multiple combinations of search resources in a grouping without generating invalid chromosomes during crossover operations. An example of how invalid chromosomes could occur follows. In parent chromosome *A*, absolute search resources 1 and 4 are on the 'eastern' side of a boundary line, while resources 0 and 3 are on the 'western' side (in group notation – < 1,4|0,3 >). In parent chromosome *B*, absolute search resources 0 and 3 are on the 'eastern' side and absolute search resources 1 and 4 are on the 'western' side (< 0,3|1,4 >). A matching crossover operation could attempt to assign search resource 3 to the 'eastern' side of a child chromosome of parent *A* and *B* that has search resource 3 already assigned to the 'western' side (resulting in a group < 1,3|0,3 >). Because a search resource can only search on one side of a boundary line, this is an illegal chromosome. With our index representation, both search resources 4 and 3 can be represented in the strand as *SR Index* 1 (group notation – < 0,1|0,1 >). They can then conduct crossover operations without generating invalid chromosomes. In this example, when the child chromosome *A* is assigned *SR Index* 1 during matching crossover, the valid absolute search resource number 3 is maintained from parent *A* and a valid chromosome is the result (< 1,4|0,3 >). In addition, the generic representation allows us to explore the search area for a given resource assignment grouping (e.g. (3,2)) with less machine time than searching each resource assignment ordering individually.

5.4 Village search variable beam heuristic

The village search mission problem search space can be represented as a rooted tree. Each node in the tree represents a partial mapping and associated set of unmapped target buildings. At each level of the tree, another building/search resource pair is added to the parent's partial mapping. For each parent node, each possible combination of search resource to unassigned building creates a child node. This method ensures all possible allocations are generated but does create duplicate nodes that reduce execution efficiency.

Tree search algorithms, such as branch and bound, can be used to find optimal solutions for this problem. Branch and bound algorithms use lower bound and upper bound estimates of the fitness function for a node. These estimates bound the solution fitness of all nodes that are children of the evaluated node. They then use the bounds to prune portions of the search tree that do not contain an optimal solution. Because branch and bound algorithm execution times grow exponentially with the problem size, alternative techniques are often used to find solutions.

A beam search branch ands bound algorithm is a modification of a basic breadth-first branch and bound heuristic. As described in works such as Nair et al.[30] and Quinn,[31] it expands at most a user-selected number of the best nodes (also called the *beam width*) at each level of the search tree and prunes the remainder. If the beam width is infinite, then the heuristic executes as a complete breadth-first branch and bound algorithm. While a finite beam width does not guarantee an optimal solution, it does reduce the execution time of the heuristic significantly to make the technique feasible.

Here we describe the *village search variable beam* (*VSVB*) heuristic. This heuristic is similar to the heuristic used by Valentea and Alvesb,[32] except that our beam width simply varies by tree depth instead of varying at each level according to a calculated threshold value. The VSVB is a breadth-first branch and bound algorithm that searches for the maximum *SRM*. The beam width is set to an initial value and is then incremented by a constant as the tree

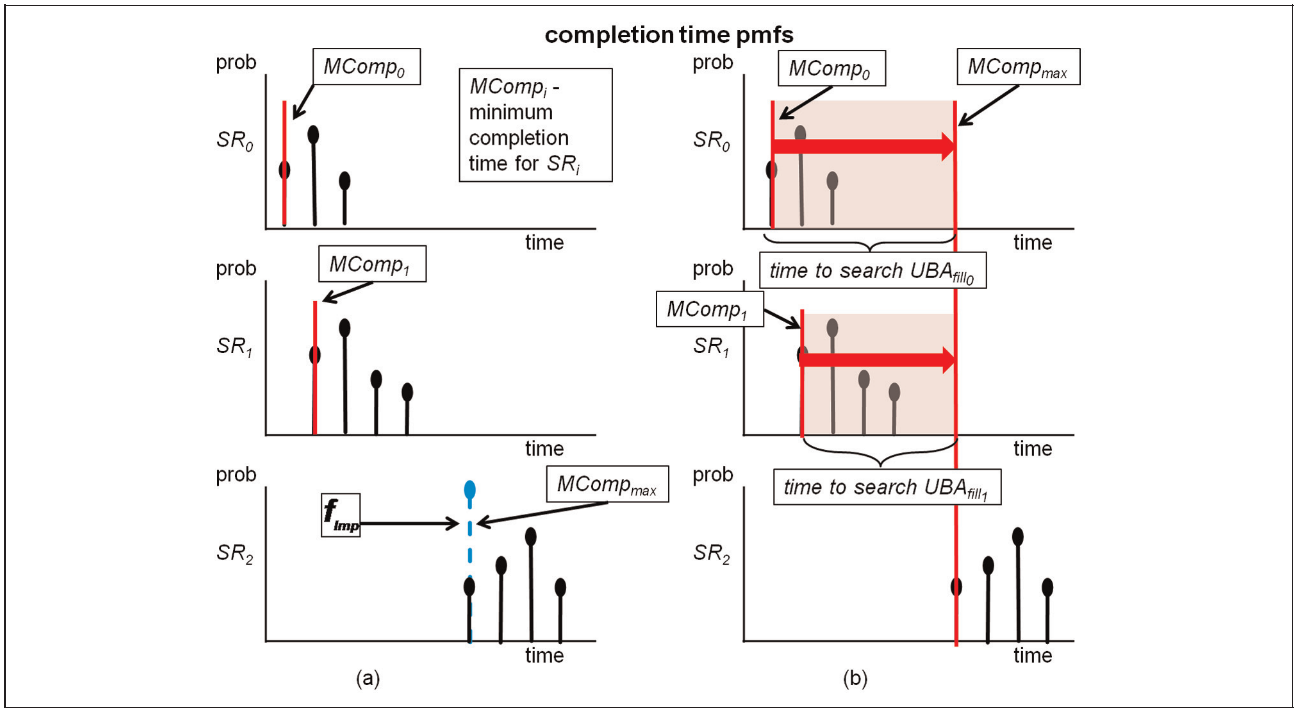


Figure 11. (a) Example search resource completion time pmfs for a node with $MComp_0$, $MComp_1$, $MComp_{\max}$, and f_{imp} pulses identified. (b) Example search resource completion time pmfs for a node with UBA_{fill0} and UBA_{fill1} identified.

depth increases. A maximum beam width is chosen prior to execution to limit the search. The reason for the variable beam width is that near the root of the tree, few actual allocations have been made and the upper bound *SRM* calculation is extremely loose. A fixed beam width decreases the percentage of total nodes searched as the depth of the tree increases. Thus, relative to the fixed beam width search, the variable beam width searches more nodes as the tree depth increases when the upper bound calculation contains more information. As our experiments showed, this modification maintains the relatively quick execution time of the beam search heuristic while providing better solutions. In the event that the upper bound calculation results in nodes with probability 1, we use the maximum lower bound as a tie breaker. This is only relevant if there are more nodes with probability 1 than the size of the beam width.

The VSVB uses a modified version of the minimum search heuristic to calculate a lower bound *SRM* for each evaluated node in the search tree. The modification removes the random starting location component of the minimum search heuristic and executes only the search for minimum completion building/*SR* pairs (Figure 3, lines 4–10). Thus for each node, the lower bound *SRM* is the fitness of the allocation to the current tree node combined with the allocation created by the min-min solution for the unassigned buildings.

The overall concept of the upper bound calculation is to first find an earliest common starting time for all search resources to begin searching the unassigned target buildings in the node. Then, the unassigned target building area is divided equally and considered to be searched in parallel. The upper bound *SRM* equals the *SRM* of the *SR* with the highest probability of completing its search of its portion of the unassigned target building area. This method ignores the path traversal times between the unassigned buildings to create a valid, but loose, upper bound.

Let the *Unassigned Building Area* for the current node be denoted *UBA*. The minimum completion time ($MComp_i$) for each SR_i is the starting pulse (i.e. the earliest pulse with a non-zero value) for each search resources' completion time pmf (Figure 11(a)). The highest value of these minimum completion times ($MComp_{\max}$) is identified (Figure 11(a)). Let UBA_{fill_i} be an upper bound on the area of the *UBA* that can be searched by a non-$MComp_{\max}$ search resource i without increasing the $MComp_{\max}$ completion time. Assuming that each SR_i searches at its maximum search rate $Search_{\max_i}$, $MComp_{\max}$ is used along with $Search_{\max_i}$ to determine UBA_{fill_i} (Figure 11(b)), which is

$$UBA_{fill_i} = Search_{\max_i} * (MComp_{\max} - MComp_i). \quad (7)$$

The remaining unassigned building area (UBA_{rem}) is calculated by subtracting from the *UBA* the sum of UBA_{fill_i} for all i. Next, for the purposes of the upper bound calculation, it is assumed that the UBA_{rem} can be searched in parallel by all search resources and thus expressed as

$$UBA_{end} = UBA_{rem}/n. \tag{8}$$

Then a completion time pmf for SR_i searching its share of the UBA_{rem}, which is UBA_{end}, is calculated for each i using its search rate pmf. Each of these n completion time pmfs (f_{rem_i}) is then separately convolved with an impulse of probability 1 ($f_{\downarrow}imp$ "Figure 11a") located at $MComp_{max}$ that represents a shift of f_{rem_i}, which results in the combined completion time pmf, f_{UB_i}:

$$f_{UB_i} = f_{imp} * f_{rem_i}. \tag{9}$$

The f_{UB_i} with the highest probability of completing prior to the MDT is the upper bound (UB) for the node:

$$UB = \max_{\forall i}\left(\sum_{time \leq MDT} f_{UB_i}\right). \tag{10}$$

6. RoPARS tool

The heuristics of the previous section are a component of the RoPARS GUI tool, as shown in Figure 12. This tool preprocesses the data and allows a user to visualize the layout of a specific village and manipulate search resources and constraints. The GUI allows the search area to be specified without tedious user action. The GUI handles the creation of village data files to be processed by the RoPARS resource allocation engine. After an allocation is created by the RoPARS tool, a user can review the plan by viewing an animation that shows the search plan being conducted at a user-selected speed.

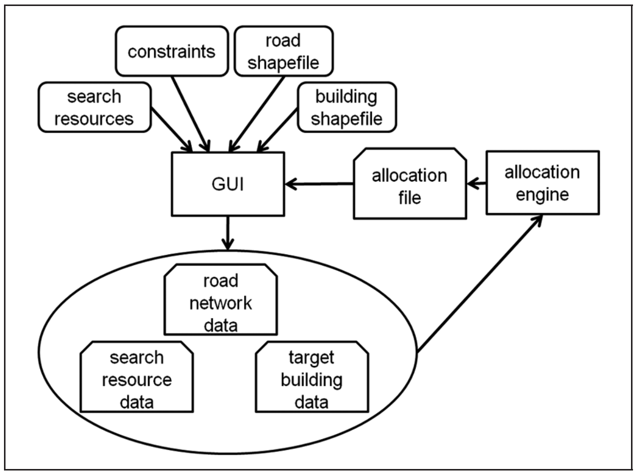


Figure 12. Block diagram of the Robust People, Animals, and Robots Search (RoPARS) tool software.

The GUI represents a village by reading in a pair of ESRI standard shapefiles that contain information on the village's road infrastructure and buildings. From these shapefiles, many important pieces of data are derived. These files allow the village to be graphically represented, as shown in Figure 13.

The GUI allows the user to input the search constraints (e.g. boundary lines, phase lines, target buildings) to define the search problem. To define the resources available, the user selects from three types of search resource teams: human, MWD, and robot. As well as being able to select the type of team, the user can edit certain descriptors of the team, such as average search and movement rates.

After all the constraints for the search plan have been input, the GUI creates three data files and sends this information to the RoPARS resource allocation engine. These files include road network, building, and search resource data files. The road network adjacency file contains connection and length data for all the road segments in the village. A road segment is demarcated by two nodes. Nodes occur where a road changes direction, at road intersections, at the closest point to a target building, and every 200 meters if no other break has occurred. Every time a change is made to the village (e.g. a new building is selected, a phase/boundary line is added), the nodes are updated. The building data file contains information about target building locations, the road segment node connected to the buildings, and the target buildings' areas. The search resource data file contains information about the types of search resources in the village, and both their movement rates and search rates.

After the RoPARS resource allocation engine creates the resource allocation, a file containing the search plan is created and interpreted by the GUI. This file contains information on the specific routes individual teams will travel, the target buildings they will search, and the timing data for each team. This information allows the GUI to display an animation of the search plan. This animation is played at a user-selected rate. A screen capture from the RoPARS GUI[33] in playback mode is shown in Figure 13.

7. Simulation results

The results discussed here consist of two different studies that are both based upon three test village search scenarios using five search resources (four human teams, one MWD team). Mission constraints include one phase line and one boundary line. Scenario 1 is based on the village in Figure 13, with 30 target buildings and 66 road nodes. Scenario 2 uses the same village, but with a different phase

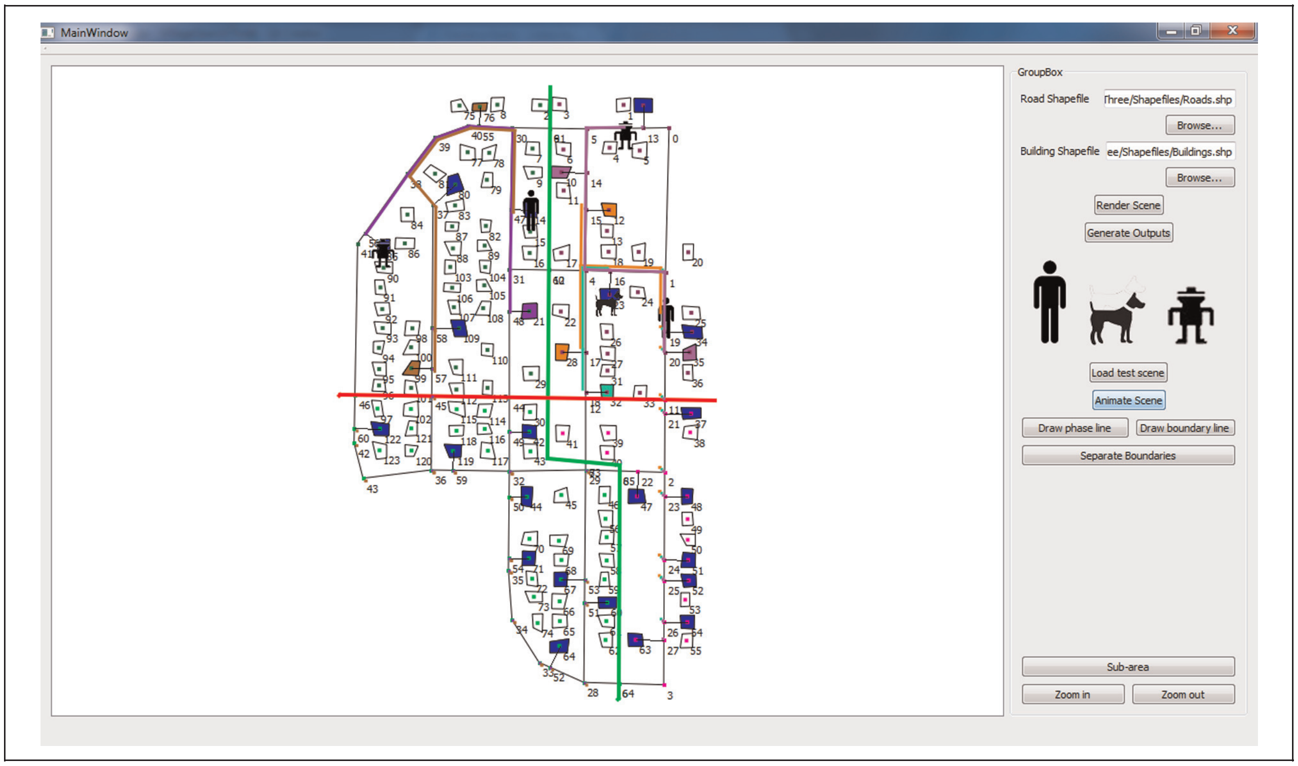


Figure 13. Robust People, Animals, and Robots Search (RoPARS) tool graphical user interface (GUI) screen capture of an example resource allocation in playback mode with five search resources and their associated colored movement paths.

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 different village (not shown), and has 24 target buildings and 64 road nodes.

To compare our heuristics, we used an expert solution created by co-author Lieutenant Colonel Maxwell, who has 19 years experience in the Army and has planned and conducted village search missions. The expert solution was created by hand using only the tools currently available to military planners. The expert solution uses the same mean search rate and movement rate for all search resources of the same type. This is necessary because specific information about the individual team rates does not currently exist. To calculate the robustness of the expert solution, we assume the search and movement rate pmfs are Gaussian with the means based on values in current Field Manuals and standard deviations of 12.5% of the mean.

In Study 1, we compare the performance of the resource allocations generated by our heuristics with the same team search and movement rate means the expert assumes. Firstly, each heuristic generated a single resource allocation using these team search and movement rates. Next, using these resource allocations, 50 simulation trials were conducted where the search resource movement rate and search rate means differed (+/–9% for search rate, +/–4% for movement rate) from one trial to the next. This shows how the *SRM* for the fixed allocations is perturbed by changes in the search resource rates. Figure 14 shows the stochastic robustness for our heuristics versus the expert solution.

Study 2 focuses on the overall system we propose, where we simulate having collected data about individual teams' search and movement rate pmfs. Here, search resources possess unique pmfs. The simulated pmf data is generated using the same 50 simulation trial data sets from Study 1. The heuristics generate a new resource allocation for each simulation trial using the search resources' specific movement and search pmfs. Then, the *SRM* for the resource allocation generated for each trial by a given heuristic was evaluated.

The resource allocation of the expert remained fixed at the one derived in Study 1, which matches current practice. That is, the expert uses just the mean values from the Field Manuals. The expert does not use pmfs based on the collected data to generate the resource allocation due to the complexity of finding an optimal resource allocation in a feasible amount of time using pmfs with non-automated methods. This is in contrast to our proposed, new overall system, where our computer-executed heuristics can deal

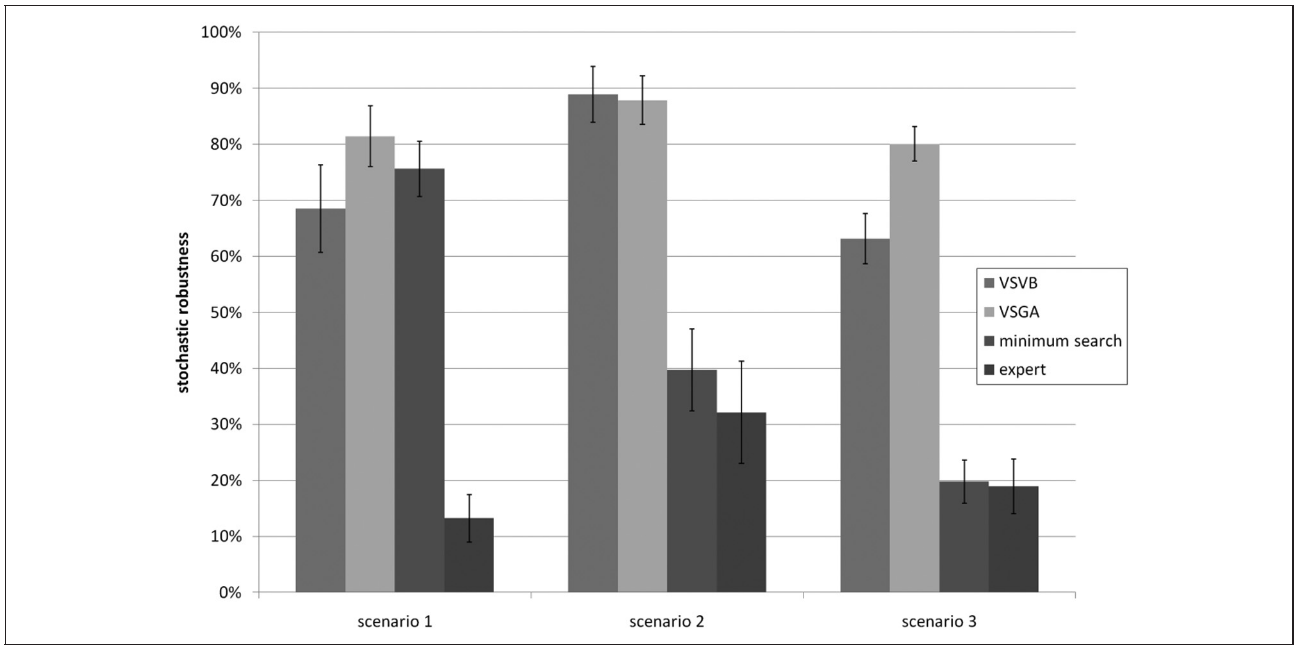


Figure 14. Study 1 average heuristic stochastic robustness for minimum search allocation, village search genetic algorithm (VSGA) allocation, village search variable beam (VSVB) allocation, and expert allocation using a single Gaussian probability mass function (pmf) for search and movement rates for each resource type. Error bars show a 95% confidence interval.

with the complexity of: (1) different means among teams of the same type; and (2) information provided by complete pmfs. Thus, Study 1 shows how our heuristics compare to an expert when given the same information, while Study 2 shows how our heuristics, using improved information, compare to the same expert.

The minimum search heuristic results are executed using 100 different groups of random building starting location assignments per trial. We conducted experiments with a varying number of random starting locations and found that the solution quality grows less than linearly with an increasing number of starting locations. Thus, the number of starting locations was chosen based upon scenario size and result quality.

Experiments for the VSGA were conducted varying the probability of crossover and probability of mutation to determine the effect on the *SRM*. These tests indicated that a crossover probability of 0.8 and a mutation probability of 0.05 gave the best robustness. Finally, the stopping conditions for the genetic algorithm were set to 1500 total generations or 350 generations with no change in the best solution. Experiments with larger number of generations did not show a significant improvement. Elitism was used to ensure that the best solution was kept in the population across generations.

The VSVB heuristic was tested using an initial beam width of 10 with the beam width incrementing by five in each level of the tree until a maximum width of 80 was reached. Even with this relatively small beam width, the execution time of the heuristic for scenario 2 was over 36 hours. By comparison, for the same scenario, the VSGA executed in 6 minutes and the minimum search heuristic executed in 10 minutes.

The results of Study 1 are shown in Figure 14. On average, the quality of the solutions produced by the VSGA was better than the other approaches. The scenario average of the VSGA was 61% better than the expert average, while the VSVB was 52% better, and minimum search was 24% better. For larger village search scenarios, the length of time to calculate an expert solution will increase and its quality will most likely decrease, as shown by our results in Figures 14 and 15. This makes using even a simple heuristic such as minimum search valuable.

The average improvement in the stochastic robustness versus the expert allocation for Study 2 is shown in Figure 15. The average over scenarios of the VSVB was 83% better than the expert average, while the VSGA was 73% better and minimum search was 30% better.

The expert solution performed well in scenario 2, which is the scenario with the smallest road network but densest target building concentration. As the scenarios increased in complexity (i.e. building density decreases or road network size increases), the expert solution quality decreased. Obviously, the quality of the expert solution will vary greatly depending on the experience and expertise of the planner.

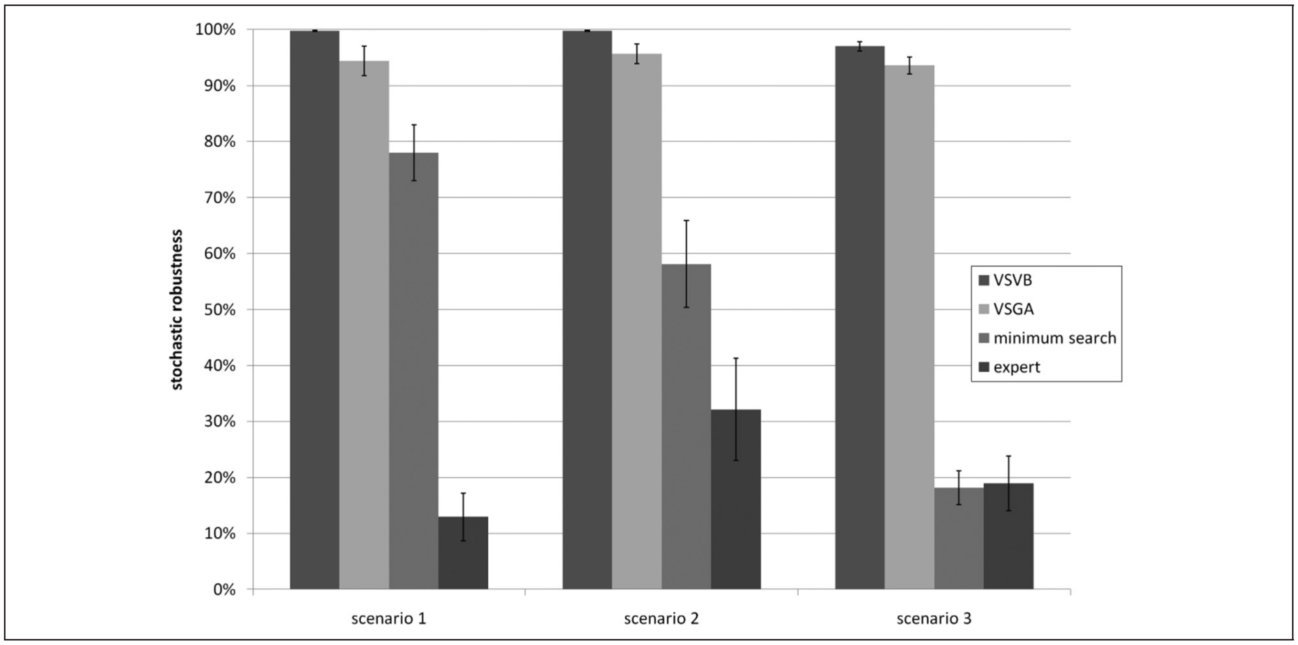


Figure 15. Study 2 average heuristic stochastic robustness values for minimum search allocation, village search genetic algorithm (VSGA) allocation, village search variable beam (VSVB) allocation, and expert allocation using known values for search resource search and movement rates. Error bars show a 95% confidence interval.

The minimum search heuristic is a greedy heuristic and does not account for the long-term impact of its choices. This results in allocations that are relatively non-robust. The random starting location aspect of the heuristic does create the potential for better performance than we report here, but it comes at the cost of longer run times.

Finally, the VSGA heuristic was compared to a non-indexed VSGA heuristic that did not use index representations for *SR*s. In this non-indexed version, it becomes necessary to examine each possible combination of *SR*s within the selected boundary assignment grouping separately. This is required to prevent the generation of illegal chromosomes during crossover and mutation operations. The non-indexed VSGA's *SRM* was on average less than 1% better than the VSGA's *SRM* at the cost of a factor of 10 increase in the average heuristic execution time.

8. Conclusions

Determining resource allocations for military village search problems is a complex problem. Current solutions produced by planning officers are time consuming and of unquantifiable quality. We have presented the RoPARS tool for village search planning as a way of addressing these issues along with three heuristics for use in its resource allocation engine. The heuristics demonstrate that computer tools can create solutions for these problems and do so in a manner that is robust against uncertainty in the environment. This can save military planners time and resources during the planning process and improve the quality of the resulting plan.

Future work in this area includes increasing the size of the test scenarios in terms of target buildings and number of search teams. Work also can be done to determine the best locations for phase lines and boundary lines, given a set of target buildings and search resources. In addition, research is needed to develop heuristics that can quickly and dynamically re-evaluate the resource allocation plan should conditions change from those used during static resource allocation.

Acknowledgments

The authors would like to thank E Chong, L Briceño, and A Al-Qa'wasmeh for their comments and D Young for use of one of his sub-routines. Preliminary versions of portions of this paper appeared in the 2009 International Conference on Information and Knowledge Engineering and the 2010 Huntsville Simulation Conference.

Funding

This work was supported by the National Science Foundation [grant number CNS-0905399] and by the Colorado State University George T. Abell Endowment.

References

1. Maxwell P, Siegel HJ, Potter J and Maciejewski AA. The ISTeC People-Animals-Robots laboratory: robust resource allocation. In: *Proceedings in the 2009 IEEE International Workshop on Safety, Security, and Rescue Robotics*, November 2009.
2. *FM 3-31.1 Army and Marine Corps Integration in Joint Operations*. US Army Training and Doctrine Command, Ft. Monroe, VA, November 2001.
3. *FM 34-8-2 Intelligence Officer's Handbook*. US Army Training and Doctrine Command, Ft. Monroe, VA, May 1998.
4. Ali S, Maciejewski AA, Siegel HJ and Kim J. Measuring the robustness of a resource allocation. *IEEE Trans Parallel Distrib Syst* 2004; 15: 630–641.
5. Saleh H and Chelouah R. The design of the global navigation satellite system surveying networks using genetic algorithms. *Eng Appl Artif Intell* 2004; 17: 111–122.
6. Aydin M. *An exploratory analysis of village search operations*. Master's thesis, Naval Postgraduate School, June 2004.
7. Babilot M. *Comparison of a distributed operations force to a traditional force in urban combat*. Master's thesis, Naval Postgraduate School, September 2005.
8. Choe JS. *Some stochastic-duel models of combat*. Master's thesis, Naval Postgraduate School, Marche 1983.
9. Fulton LV, Lasdon LS, McDaniel RR, Jr and Coppola MN. Two-stage stochastic optimization for the allocation of medical assets in steady-state combat operations. *J Defense Model Simulat* 2010; 7: 89–102.
10. Lucas T. The stochastic versus deterministic argument for combat simulations: Tales of when the average won't do. *Mil Oper Res* 2000; 5: 9–28.
11. Popken D and Cox L. A simulation-optimization approach to air warfare planning. *J Defense Model Simulat* 2004; 1: 127–140.
12. Desrochers M, Desrosiers J and Solomon M. A new optimization algorithm for the vehicle routing problem with time windows. *Oper Res* 1992; 40: 342–354.
13. Laportea G, Gendreau M, Potvin J and Semet F. Classical and modern heuristics for the vehicle routing problem. *Int Trans Oper Res* 2000; 7: 285–300.
14. Potvin J. Genetic algorithms for the traveling salesman problem. *Ann Oper Res* 1996; 63: 339–370.
15. Larranaga P, Kuijpers CMH, Murga RH, Inza I and Dizdarevic S. Genetic algorithms for the travelling salesman problem: a review of representations and operators. *Artif Intell Rev* 1999; 13: 129–170.
16. Bektas T. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega* 2006; 34: 209–219.
17. Tang L, Liu J, Rong A and Yang Z. A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex. *Eur J Oper Res* 2000; 124: 267–282.
18. Sabuncuoglu I and Bayiz M. Job shop scheduling with beam search. *Eur J Oper Res* 1999; 118: 390–412.
19. Yu Z, Jinhai L, Gouchang G, Rubo Z and Haiyan Y. An implementation of evolutionary computation for path planning of cooperative mobile robots. In: *Proceedings of the 4th World Congress on Intelligent Control and Automation*, June 2002, pp.1798–1802.
20. 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.
21. Smith J, Siegel HJ and Maciejewski AA. Robust resource allocation in heterogeneous parallel and distributed computing systems. In: Wah B (ed.) *Wiley encyclopedia of computing*. Vol. 4. New York: John Wiley & Sons, 2008, pp.2461–2470.
22. Bilbao J, Miguel AH and Kambezidis HD. Air temperature model evaluation in the north Mediterranean belt area. *J Appl Meteorol* 2002; 41: 872–884.
23. Bruhn JE, Fry WE and Fick GW. Simulation of daily weather data using theoretical probability distributions. *J Appl Meteorol* 1980; 19: 1029–1036.
24. Hill RD, Moate CP and Blacknell D. Estimating building dimensions from synthetic aperture radar image sequences. *IET Radar Sonar Navig* 2008; 2: 189–199.
25. Wang L, Maciejewski AA, Siegel HJ, Roychowdhury VP and Eldrige BD. Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach. *J Parallel Distrib Comput* 1997; 47: 8–22.
26. Prins C. A simple and effective evolutionary algorithm for the vehicle routing problem. *Comput Oper Res* 2004; 31: 1985–2002.
27. 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: 1157 – 1173.
28. Ibarra OH and Kim CE. Heuristic algorithms for scheduling independent tasks on non-identical processors. *J ACM* 1977; 24: 280–289.
29. Blickle T and Thiele L. *A comparison of selection schemes used in genetic algorithms*. Technical Report TIK-Report No. 11, Ver. 2, CEN Lab, Swiss Federal Institute of Technology, December 1995.
30. Nair SK, Thakur LS and Wen K. Near optimal solutions for product line design and selection: beam search heuristics. *Manag Sci* 1995; 41: 767–785.
31. Quinn M. *Parallel programming in C with MPI and OpenMP*. New York: McGraw-Hill, 2004.
32. Valentea J and Alvesb R. Beam search algorithms for the single machine total weighted tardiness scheduling problem with sequence-dependent setups. *Comput Oper Res* 2008; 35: 2388 – 2405.
33. Maxwell P, Friese R, Maciejewski AA, Siegel HJ, Potter J and Smith J. A demonstration of a simulation tool for planning robust military village searches. In: *Proceedings of the Huntsville Simulation Conference (HSC'10)*, October 2010.

Author biographies

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

Dr Anthony A Maciejewski is the department head for the Electrical and Computer Engineering Department 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 Jerry Potter is a research faculty member in the department of Electrical and Computer Engineering at Colorado State University.

Dr Gregory Pfister is a research faculty member in 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 research scientist at Lagrange Systems, Inc.

Ryan Friese is a graduate student at Colorado State University. He holds a BS in Electrical Engineering and a BS in Computer Science.