DISSERTATION

# ROBUST RESOURCE ALLOCATION HEURISTICS FOR MILITARY VILLAGE SEARCH MISSIONS

Submitted by

Paul Maxwell

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Summer 2012

## ABSTRACT

## ROBUST RESOURCE ALLOCATION HEURISTICS FOR MILITARY VILLAGE SEARCH MISSIONS

On the modern battlefield, cordon and search missions (a.k.a. village searches) are conducted daily. Creating resource allocations that assign different types of search teams (e.g., soldiers, robots, unmanned aerial vehicles, military working dogs) to target buildings of various sizes is difficult and time consuming in the static planning environment. Efficiently and effectively creating resource allocations when needed during mission execution (a dynamic environment) is even more challenging. There are currently no automated means to create these static and dynamic resource allocations for military use. Military planners create village search plans using reference tables in Field Manuals and personal experience. These manual methods are time consuming and the quality of the plans produced are unpredictable and not quantifiable. This work creates a mathematical model of the village search environment, and proposes static and dynamic resource allocation heuristics using robustness concepts. The result is a mission plan that is resilient against uncertainty in the environment and that saves valuable time for military planning staff.

## DEDICATION

This dissertation is dedicated to my wife, Rene, and my children: Samuel, Sophie, and Oliver. Their unwavering support enabled me to accomplish this task. I am extremely thankful to them for their unselfishness and their freely given assistance. I hope that one day they will reflect upon this document and the time period in which it was created with fond memories.

TABLE OF CONTENTS

# INTRODUCTION

Making decisions in a static or dynamic environment on how to best utilize resources to accomplish a task with a set of specified constraints is difficult. These resource allocation problems take many forms, such as computer task assignment, travelling salesmen planning, post-disaster search and rescue team allocation, and military search team allocation for a Cordon and Search of a village (a.k.a., village search). In all of these scenarios, it is desirable to create an allocation that is optimal with respect to some performance metric. However, in general, the problem of resource allocation is NP-complete (for example in heterogeneous parallel and distributed computing [Cof76], [IbK77]) and therefore heuristic solutions are often used to find near-optimal solutions. This dissertation establishes a framework model for search-type environments (e.g., village search), and presents techniques and heuristics for finding near-optimal resource allocations in both static and dynamic environments.

At its most basic level, a village search consists of a set of targets (buildings) that require searching. To accomplish the mission, search teams (resources) such as human search teams, military working dog teams, Explosive Ordinance Detachments, and robots, are tasked to search the buildings. An example village search environment is shown in Figure 1. A village is composed of a set of *target buildings* ($T = \{T_1, T_2, \ldots\}$). The environment also is defined by the road network that connects the target buildings. This road network is a set of *road segments* ($RS = \{RS_1, RS_2, \ldots\}$) that are located at road intersections, where target buildings intersect a road, and at evenly spaced intervals (e.g., 200 meters) if no other criteria are met.  Added to the description of the physical environment are human imposed constraints that limit the set of possible solutions to the resource allocation problem.  First, the search area can be divided into sub-areas by *unit boundary lines*.  These lines are drawn by military leaders to restrict movement by units. Units cannot cross these lines and thus are limited to searching buildings within their boundary area.  Next, unit movement can be controlled by a synchronization measure known as a *phase line*.  These user inserted lines restrict unit movement across the line until all units are ready to proceed across the line.  This in effect limits all unit progress to the rate of the slowest unit.  The *direction of advance* indicates the search team heading when crossing phase lines.

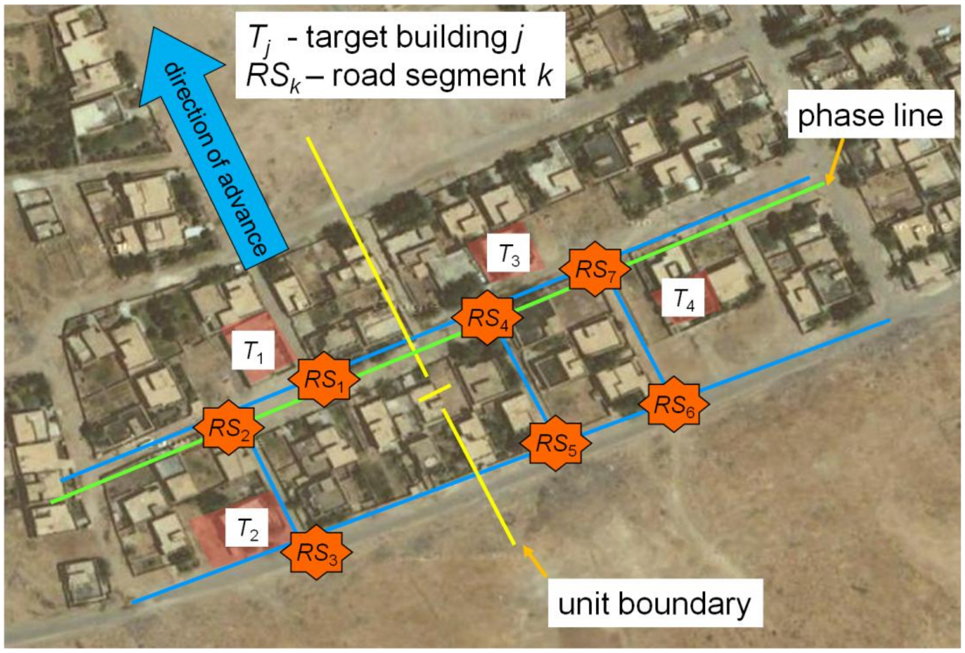


**Figure 1**. Example village search scenario with four target buildings, seven road segments, a unit boundary line, a phase line, and a direction of advance.

Once the environment is defined, military planners attempt to allocate the resources (search teams) to the tasks (building searches) in a manner that will meet the given performance requirement(s).  For this dissertation, the performance requirement is defined as the maximum time allowed to complete the village search (village search mission deadline time).  An example resource allocation for a village search is shown in Figure 2.  For a search scenario there is a set of *search teams* ($ST = \{ST_1, ST_2, \ldots\}$).  In general, searches may be limited to only certain team types, and the search rates are dependent on the type.  To move between target buildings, *movement paths* ($M_{ijk}$) that have associated *probability mass functions* (pmfs) for the time to travel between buildings *j* and *k* for *search team i* are designated.  For movement between any two target buildings a set of movement paths may exist.  For this work, the movement path with the minimum mean movement time for a given search team *i* is used.  To meet the performance criteria, the resource allocation must account for factors such as the search rate of the search teams, the movement path time between structures, the ordering of the structure searches, and uncertainties in the environment.

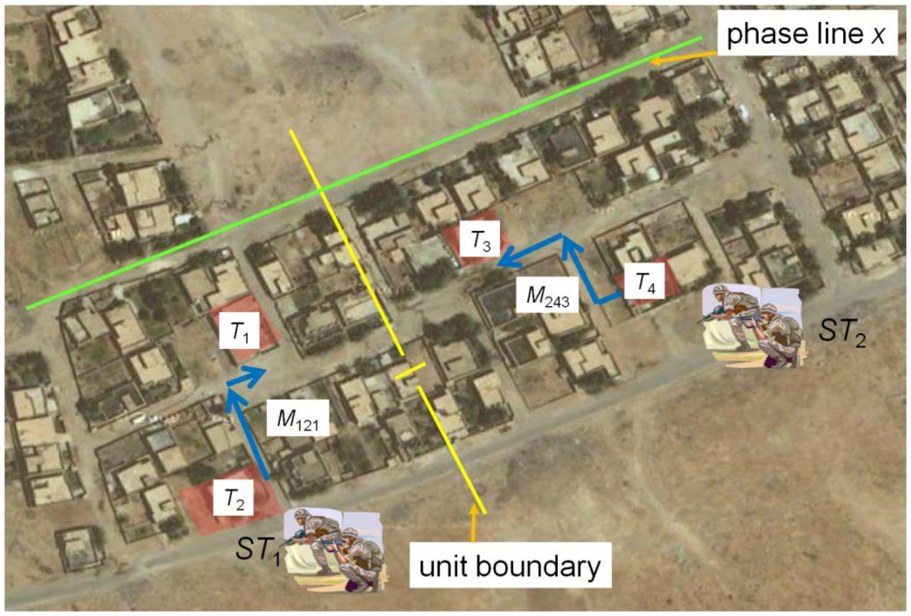


**Figure 2**.  An example village search resource allocation plan with two search teams, four target buildings, two movement paths, a unit boundary line, and a phase line.

Assigning search teams to target buildings and picking the movement paths is done by military leaders during their static planning phase and dynamically, as needed, during mission execution. Computer tools could 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 to assist decision makers in their planning process and, as a result, individual experience and published military data tables are the only tools available. The result is that the quality of a plan, its ability to account for uncertainty, and the resulting confidence in its success will vary dramatically based on the decision-maker that produced it.

Due to the lack of automated tools, the *Robust People, Animals, and Robots* (*RoPARS*) tool was developed to assist with the planning for the static village search problem. Its robustness concepts, mathematical models, and resource allocation heuristics are presented here. The result is an automated decision-making aid for military leaders to use during the static mission planning phase of an operation.

The RoPARS tool improves mission planning by incorporating: (1) mathematical models that represent people, animals, and robots (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 search for near-optimal static resource allocations (i.e., completing a village search by its deadline with maximum 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 static planning process, thus freeing planners for other tasks. Through a *graphical user interface* (*GUI*), the tool: (1) imports *Environmental Systems Research Institute* (*ERSI*) standard shapefiles of the search area (many of which are currently available in repositories such as the Urban Tactical Planner library); (2) accepts user inputs regarding the plan (e.g., phase lines, boundary lines, search team types and compositions, target buildings); (3) creates a resource allocation using static allocation heuristics (e.g., Minimum Search heuristic, Village Search Genetic Algorithm); (4) evaluates the performance of the allocation using quantifiable measures; and (5) graphically displays at a user-selected rate the resulting plan. This 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 combat operations in the contemporary operating environment.

In the static resource allocation domain, the contributions of the dissertation 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.

In addition to the RoPARS static tool, an alternative modeling and static resource allocation method that uses Petri Nets was explored. Specifically, timed, stochastic, colored Petri Nets provide the ability to model concurrency in the mission and to introduce stochastic information into the simulation. Petri Nets can model the stochastic variables of the mission and incorporate constraints typically used in the missions (e.g., unit boundary lines, phase lines, direction of attack) into the solution. Then, using Monte Carlo methods, the Petri Net simulation results can be used to build probability mass functions (pmfs) for the search mission and these pmfs can provide information about the robustness of a given plan (resource allocation) to military leaders. Though this method is a viable means of creating and evaluating resource allocations, it was abandoned for the RoPARS method due to the difficulty of creating the village models in the Petri Net representation.

Automated tools also are needed in the dynamic village search environment in which 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 team, 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 the ability 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. Adding automated dynamic reallocation components to the RoPARS tool or creating a stand-alone dynamic solution (independent of *RoPARS*) for village search replanning in both training and operational environments will greatly assist military leaders in their decision-making process. The stand-alone solution will 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 my dynamic techniques. Like the static case, the dynamic replanning tool requires the following capabilities to be effective: model the search area using automated digital maps; use probabilistic models to calculate search times; determine a good solution from multiple possible resource allocations. It also requires the ability to incorporate real-time mission feedback (e.g., actual search completion times of target buildings) and execute within the time constraints of an ongoing mission.

In the dynamic environment, the contributions of this work include: (1) a framework for conducting dynamic military mission replanning; (2) 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. Based upon the computation time available, the dynamic resource allocation heuristics select an allocation that results in the best acceptable plan.

The remainder of this dissertation is organized as follows. Chapter 2 presents the work on the static village search problem. Chapter 3 introduces the Petri Net method of static resource allocation while Chapter 4 discusses the results of the work in the dynamic village search domain. Finally, Chapter 5 contains the conclusions.

# 2. STATIC ALLOCATION HEURISTICS [1]

## 2.1 INTRODUCTION

Colorado State University's Information Science and Technology Center, ISTeC (ISTeC.ColoState.edu), is organizing the PAR (People-Animals-Robots) multi-disciplinary research laboratory [MaS09] 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 teams that conduct the search can consist of soldiers, *military working dogs* (*MWD*), *explosive ordinance detachments* (*EOD*), military aircraft, *unmanned aerial vehicles* (*UAVs*), and electronic surveillance. The problem of assigning search teams to

[1] The contents of this chapter have been accepted for publication in P. Maxwell, A. A. Maciejewski, H. J. Siegel, J. Potter, G. Pfister, J. Smith, and R. Friese, "Robust static planning tool for military village search missions: model and heuristics," *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, to appear.

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 deadlines.  An example village search problem is shown in Figure 3.

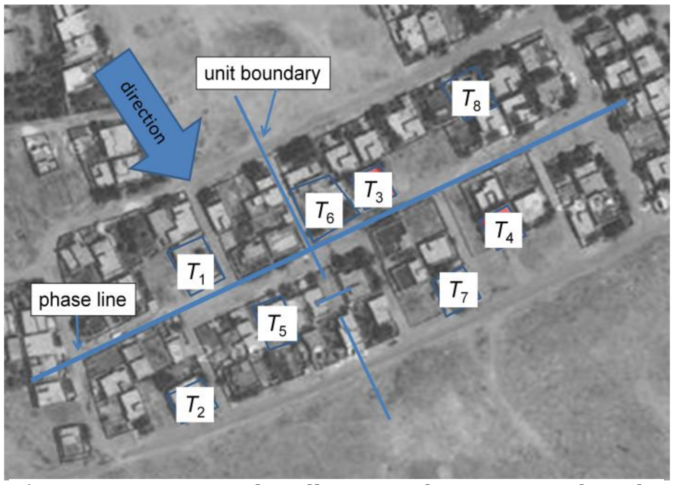


**Figure 3.** An example village search mission with eight target buildings (Tj), a unit boundary, a restrictive phase line, and a direction of advance.

When planning a village search, military staff officers must analyze the problem, allocate teams 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.  Additionally, 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 (e.g., [MFM01], [MFM98]) 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 *people, animals, and robots* (*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 war gaming portions of the decision-making process. Through a *graphical user interface* (*GUI*), the planners employ the tool to import standard ESRI shapefiles (www.esri.com) of the search area (many of which are currently available in the Urban Tactical Planner library - 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. Additionally, 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 chapter 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 2.2 of this chapter presents work related to the village search problem. Section 2.3 provides a discussion of the robustness metric developed in [AlM04], and [SaC04]. In Section 2.4, the model for a basic village search mission is discussed. The resource allocation heuristics developed for the village search model are described in Section 2.5. An overview of the RoPARS tool GUI is provided in Section 2.6. The simulation results are shown in Section 2.7 and finally, in Section 2.8 I present the conclusions.

## 2.2 RELATED WORK

There are three categories of research that possess similarities to my work: combat simulations, vehicle routing, and travelling 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, my 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 (e.g., [Ayd04], [Bab05]) though work has been done on stochastic models (e.g., [Cho83], [FuL10], [Luc00], [PuC04]). 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. Additionally, 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 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 [DeD92], [LaG00], [Pot96]) has similarities to my 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 my domain in that I 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; 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 solutions for the TSP (e.g., [LaK99], [Pot96]) and mTSP problem (e.g., [Bek06]) due to their wide applicability and complexity. Here I review only works that use genetic algorithms to find a solution because those are the closest comparisons.

The work in [TaL00] models a steel rolling factory as an mTSP problem 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 my work, their genetic algorithm uses deterministic values instead of stochastic information and is not concerned with the robustness of the solution.

In [SaB99], a genetic algorithm is used to produce solutions for a global satellite survey network problem. This problem domain was transformed into an mTSP problem 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. Additionally, the problem does not have limiting constraints on the solution such as the boundary lines of the village search problem.

The work in [YuJ92] transforms a multiple robot mine clearing problem into an mTSP problem. 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. Additionally, 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.

## 2.3 DEFINING ROBUSTNESS

I have defined robustness and a methodology to calculate the robustness of a resource allocation in [AlM04] and studied it within a variety of systems (e.g., [AlM08], [SaC04], [SmS08]). I 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 [AlM04], 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 modelling 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 team 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 stochastic robustness metric for a village search can be determined. Once this is done, heuristics for planning robust resource allocations can be designed.

## 2.4 VILLAGE SEARCH ROBUSTNESS MODEL

### 2.4.1 OVERVIEW

A quantitative mathematical model for a village search is presented in this section. To illustrate the problem, Figure 4 provides an example allocation for a village search scenario. Conducting the search are search teams (resources) ($ST = \{ST_1, ST_2, \ldots\}$) where $ST_i$ 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 team i*, ($ST_i$). Military planners attempt to allocate the resources (search teams) to the tasks (building searches) in a manner that will meet the given performance requirement (village search mission deadline time). A model of this scenario must account for factors such as the search rate of the search teams, 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 [AlM08]. Namely: (1) What behavior of the system makes it robust? (2) What uncertainties is the system robust against? and (3) How is robustness of the system quantified?

### 2.4.2 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

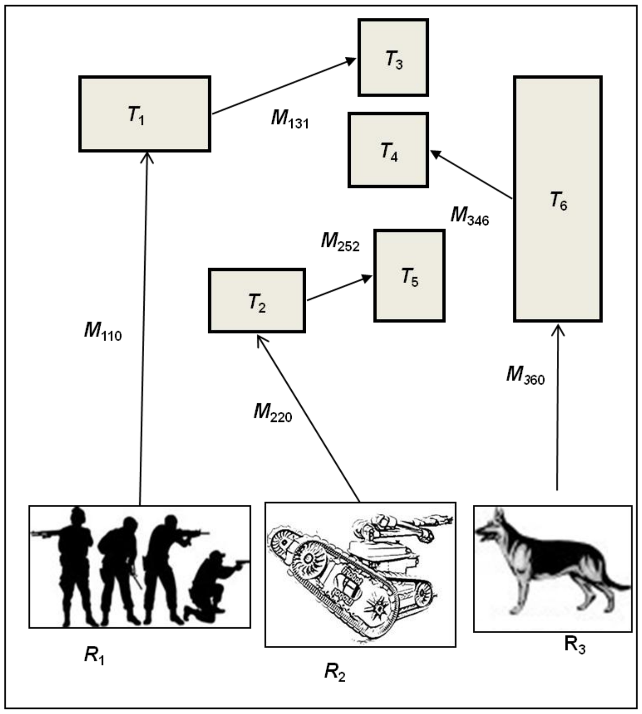


**Figure 4.** An example resource allocation for a village search with three search teams (human team, robot team, and a military working dog team) allocated to six tasks (building searches) with six movement paths

criterion considered in this study is the *mission deadline time* (*MDT*) or time by which the mission must be completed.

### 2.4.3 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, weather, engagement with explosive hazards, treatment and evacuation of casualties, changes to the availability of teams, and unanticipated animal (MWD) behaviour. 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 modelled using a variety of distributions functions (e.g., [BiM02], [BrF80], [HiM08], [WaM97]). This chapter considers the variability in the *team search rate*, $\sigma_i$, and variability in the *team ground movement rate*, $\gamma_i$ for search team $i$ as the perturbations. These values are random variables with a distribution of rates. The base rates, $\sigma_i$ and $\gamma_i$, 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 chapter, the model can support the modification of $\sigma_i$ and $\gamma_i$ 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.

### 2.4.4 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 [SaC04], 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 $RCT_{max} \leq MDT$.

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* |
| $\sigma_i$ | search rate for search team *i* |
| $C_{ijk}$ | completion time for team *i* on building *j* moving from building *k* |
| $CT_p$ | completion time for team *i* on the $p^{th}$ building in its target set |
| $\gamma_i$ | ground movement rate for search team *i* |
| $MDT$ | mission deadline time |
| $M_{ijk}$ | movement path from building *k* to building *j* for search team *i* |
| $n_{ix}$ | number of target buildings for search team *i* in phase line area *x* |
| $P$ | index into set of target buildings for search team *i* in phase line area *x* |
| $ST_i$ | search team *i* |
| $T_j$ | target building *j* |
| $\Phi$ | number of phase lines |
| $\Theta_{ix}$ | ordered set of target buildings for search team *i* in phase line area *x* |

A set of target buildings, has a corresponding set of *areas*, $A = \{A_1, A_2, \ldots\}$, that may include multiple floors. The team's ground movement rate, $\gamma_i$, is the rate that the team 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 teams using the same path is negligible. Therefore, *completion time*, $C_{ijk}$, for search team *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 team 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 [Pri04]. This algorithm identifies the minimum traversal time road segment(s) between all building pairs for search team *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 $\sigma_i$ and $\gamma_i$. 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:

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

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 teams have adequate supporting elements to operate independently within a phase line area. Assume there are $\Phi$ phase lines. This results in $\Phi$+1 phase line areas. The effect of the phase line is barrier synchronization. Additionally, the perturbation parameters considered are independent with respect to the search teams and therefore the team 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 $\Theta_{ix}$ of target buildings for search team $i$ in phase line area $x$. Then, the $p$ represents the $p^{\text{th}}$ entry in the set. In Equation 2, I sum the building completion times for a search team 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 $p^{\text{th}}$ building in the set $\Theta_{ix}$.

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

Because I am working with discrete random variables to express the uncertainty in the system, the completion time is a probability mass function. 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 teams 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 teams.

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

To find the completion time pmf for all search teams over all $\Phi + 1$ phase line areas, I convolve the phase line distribution functions as shown in Equation 5.

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

I then define the stochastic robustness metric as the probability that all search teams finish searching their target sets by the *MDT* (Equation 6).

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

Thus, for a given resource allocation of search teams 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 in [IbK77], 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 chapter, I only exam the first scenario though it is easy to convert the heuristics to accomplish the goals of the second scenario.

## 2.5 RESOURCE ALLOCATION HEURISTICS

### 2.5.1 ENVIRONMENT

The village search mission environment is defined by its boundary lines and the assignment (grouping) of search teams 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 teams to boundary line area assignments to explore. For example, if there are five search teams and two boundary line areas, then there exist four valid grouping possibilities (one *ST* on the east side, four *STs* on the west side (1,4), two *STs* on the east side, three *STs* on the west side (2,3), etc.). If more boundary lines exist (i.e., two boundary lines) and five *STs* are used then a grouping tuple may be (2,2,1).

### 2.5.2 MINIMUM SEARCH HEURISTIC

The minimum search heuristic was inspired by the original two-phase greedy heuristic in [IbK77]. 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 team. This forces the heuristic to explore other solutions. The pseudocode for the minimum search heuristic is shown in Figure 5.

```
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 5.** Minimum search heuristic pseudocode.

### 2.5.3 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 6. 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 team assignment combination is represented. During chromosome generation, a look-up table is created that cross-references *ST* index values to *ST* absolute reference values (Figure 7a). 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 [BlT95]. 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 stochastic robustness metric 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 7b. The strand is an array composed of target building number/search team index pairs. The strand represents target buildings, their assigned search teams, and the scheduling order of the target buildings. The building number/*ST* 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.

```
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 6.** 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 7.** Village search genetic algorithm chromosome components: (a) search team look-up table, and (b) chromosome "strand."

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 8, 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.

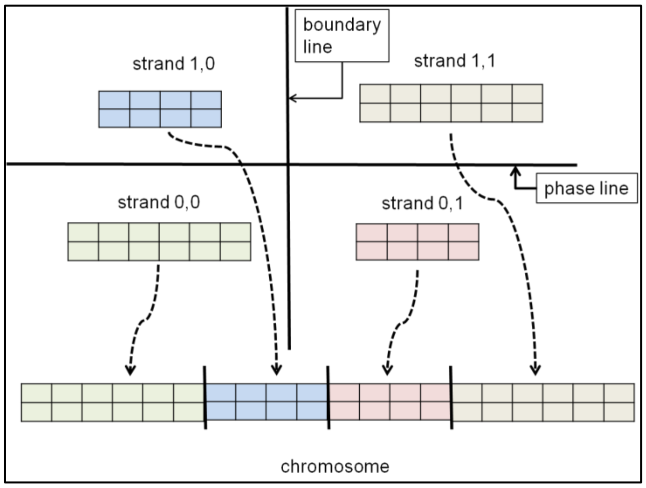


**Figure 8.** Village search genetic algorithm chromosome.

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 9 and 10. Similar to classic genetic algorithms, the crossover operators use two parent chromosomes in their operation. Additionally, 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 8) 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 in [WaM97] 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.

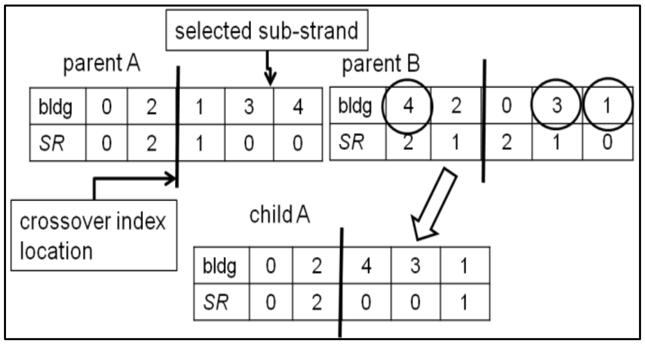


**Figure 9.** Village search genetic algorithm scheduling crossover example**.**

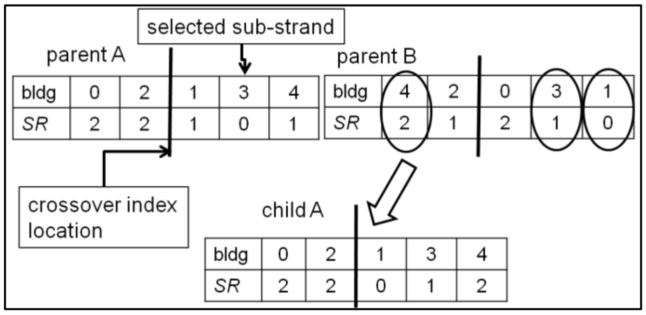
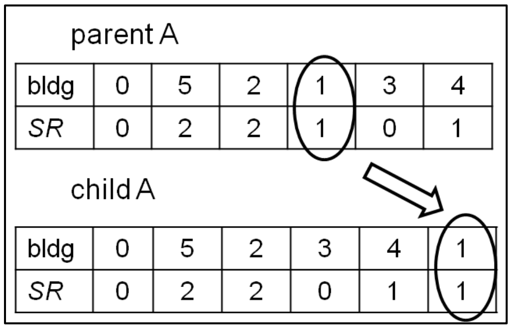


**Figure 10.** Village search genetic algorithm matching crossover example.

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 team it has in parent *B*. The operation is then repeated with the parent chromosomes reversed (see Figure 10).

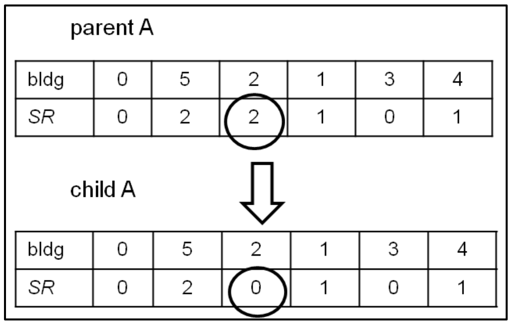
Similar to the crossover operators, the mutation operators function on a strand within a chromosome. Again, the operators can be performed on more than one strand but as with the crossover operators it has been limited to one strand for the example in this chapter. Examples for the scheduling and matching mutation operators are shown in Figures 11 and 12.



| | | | | | | |
|---|---|---|---|---|---|---|
| bldg | 0 | 5 | 2 | 1 | 3 | 4 |
| *SR* | 0 | 2 | 2 | 1 | 0 | 1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| bldg | 0 | 5 | 2 | 3 | 4 | 1 |
| *SR* | 0 | 2 | 2 | 0 | 1 | 1 |

**Figure 11.** Village search genetic algorithm scheduling mutation example.



| | | | | | | |
|---|---|---|---|---|---|---|
| bldg | 0 | 5 | 2 | 1 | 3 | 4 |
| *SR* | 0 | 2 | 2 | 1 | 0 | 1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| bldg | 0 | 5 | 2 | 1 | 3 | 4 |
| *SR* | 0 | 2 | 0 | 1 | 0 | 1 |

**Figure 12.** Village search genetic algorithm matching mutation example.

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.

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. Additionally, 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 previously, through use of a look-up table (Figure 7a), the *ST* indices are associated with absolute *ST* identification numbers. Chromosome *ST* look-up tables are inherited from parent to child during crossover and mutation operations. I use index tables so that I can search multiple combinations of search teams 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 teams 1 and 4 are on the “eastern” side of a boundary line while teams 0 and 3 are on the “western” side (in group notation - <1,4|0,3>). In parent chromosome *B*, absolute search teams 0 and 3 are on the “eastern” side and absolute search teams 1 and 4 are on the “western” side (<0,3|1,4>). A matching crossover operation could attempt to assign search team 3 to the “eastern” side of a child chromosome of parent *A* and *B* that has search team 3 already assigned to the “western” side (resulting in a group <1,3|0,3>). Because a search team can only search on one side of a boundary line, this is an illegal chromosome. With my index representation, both search team 4 and 3 can be represented in the strand as *ST 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 *ST Index* 1 during matching crossover, the valid absolute search team number 3 is maintained from parent *A* and a valid chromosome is the result (<1,4|0,3>).  Additionally, the generic representation allows us to explore the search area for a given team assignment grouping (e.g., (3,2)) with less machine time than searching each team assignment ordering individually.

### 2.5.4 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 team pair is added to the parent's partial mapping.  For each parent node, each possible combination of search team 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 like 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.  It then uses 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 and bound algorithm is a modification of a basic breadth-first branch and bound heuristic.  As described in works such as [NaT95] and [Qui04], 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 I describe the *village search variable beam* (*VSVB*) heuristic. This heuristic is similar to the heuristic used in [VaA08] except that the 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 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 the 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, I 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/$ST$ pairs (Figure 5, 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 teams 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 *ST* 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 teams' completion time pmf (Figure 13a). The highest value of these minimum completion times ($Mcomp_{max}$) is identified (Figure 13a). Let $UBA_{fill_i}$ be an upper bound on the area of the *UBA* that can be searched by a non-$Mcomp_{max}$ search team $i$ 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 13b), 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 teams 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 ($f_{imp}$, Figure 13a) located at *Mcomp*$_{max}$ that represents a shift of $f_{rem_i}$, that results in the combined completion time pmf, $f_{UB_i}$,

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

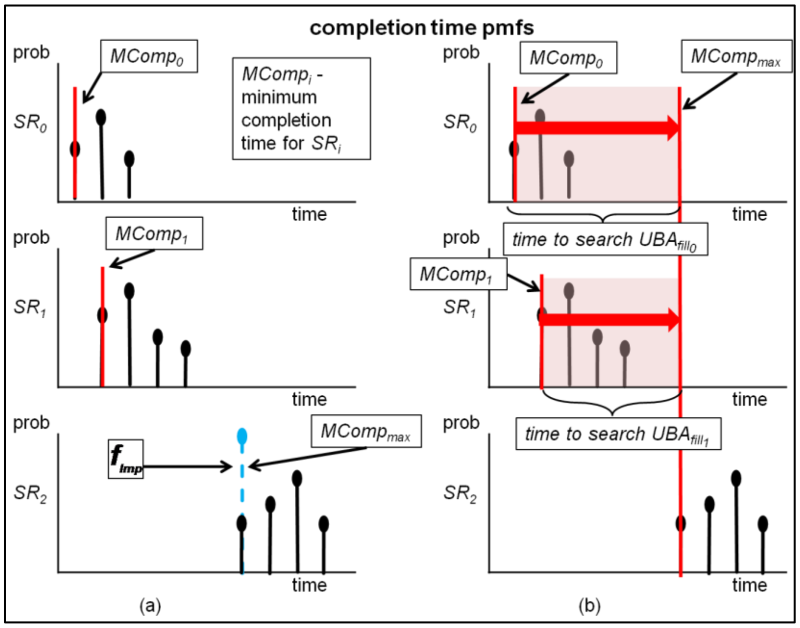


**Figure 13.** (a) Example search resource completion time pmfs for a node with *Mcomp*$_0$, *Mcomp*$_1$, *Mcomp*$_{max}$, and $f_{imp}$ pulses identified. (b) Example search resource completion time pmfs for a node with *UBA*$_{fill0}$ and *UBA*$_{fill1}$ identified.

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

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

## 2.6 RoPARS TOOL

The heuristics of the previous section are a component of the RoPARS GUI tool as shown in Figure 14. This tool preprocesses the data and allows a user to visualize the layout of a specific village and manipulate search teams 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.

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 15.

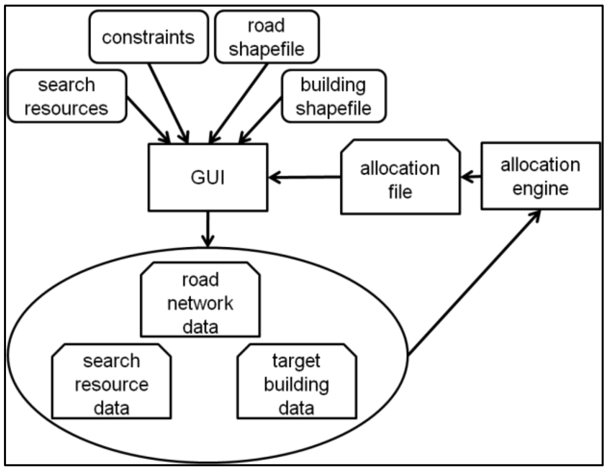
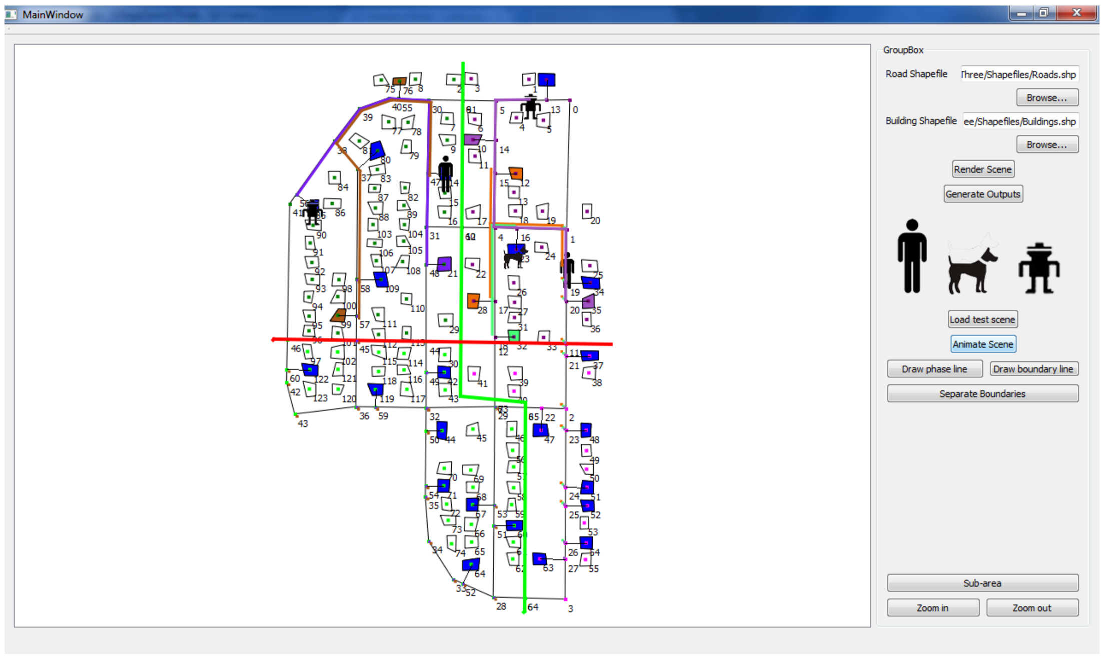


**Figure 14.** Block diagram of RoPARS tool software.

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 teams available, the user selects from three types of search resource teams: human, military working dog, 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 team 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 team data file contains: information about the types of search teams 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 [MaF10] in playback mode is shown in Figure 15.


Road Shapefile Three/Shapefiles/Roads.shp

Building Shapefile ee/Shapefiles/Buildings.shp

**Figure 15.** RoPARS tool GUI screen capture of an example resource allocation in playback mode with five search teams and their associated colored movement paths.

## 2.7 SIMULATION RESULTS

The results discussed here consist of two different studies that are both based upon three test village search scenarios using five search teams (four human teams, one military working dog team). Mission constraints include one phase line and one boundary line. Scenario 1 is based on the village in Figure 15, with 30 target buildings and 66 road nodes. Scenario 2 uses the same village, 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 different village (Appendix A, Figure 42), and has 24 target buildings and 64 road nodes.

To compare my heuristics, I used an expert solution created by co-author Lieutenant Colonel Maxwell, who has 20 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 teams of the same type. This is by necessity because specific information about the individual team rates does not currently exist. To calculate the robustness of the expert solution, I 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, I compare the performance of the resource allocations generated by my heuristics with the same team search and movement rate means the expert assumes. First, 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 team 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 team rates. Figure 16 shows the stochastic robustness for the heuristics versus the expert solution.

Study 2 focuses on the overall system I propose, where I simulate having collected data about individual teams' search and movement rate pmfs. Here, search teams 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 teams' specific movement and search pmfs. Then, the *SRM* for the resource allocation generated for each trial by a given heuristic was evaluated.

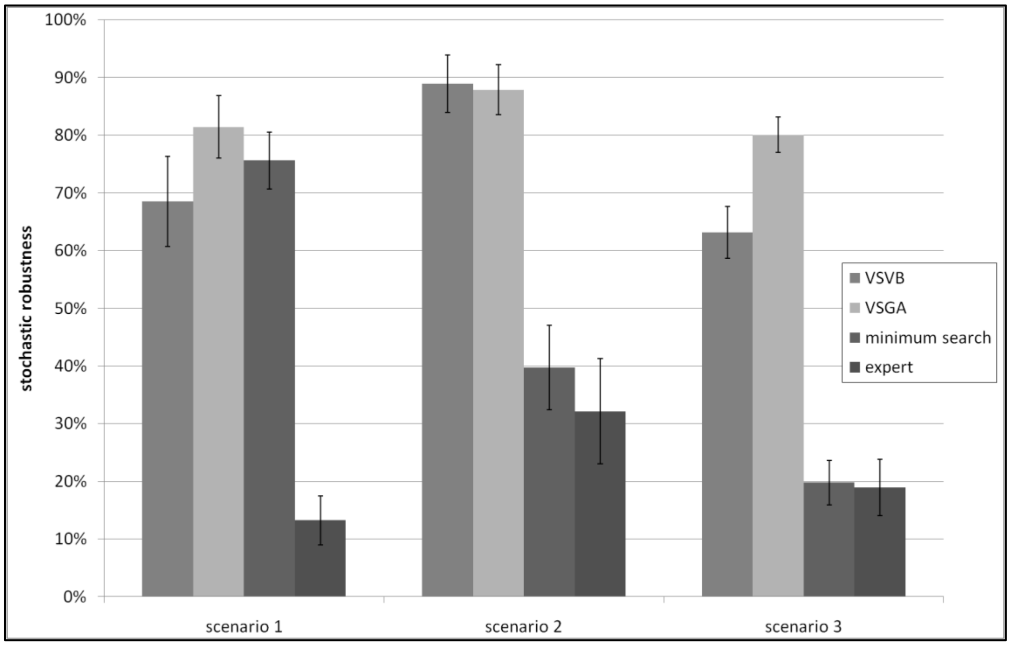


**Figure 16.** Study 1 average heuristic stochastic robustness for minimum search allocation, VSGA allocation, VSVB allocation, and expert allocation using a single Gaussian pmf for search and movement rates for each team type. Error bars show a 95% confidence interval.

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 the proposed, new overall system where my computer executed heuristics can deal 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 my heuristics compare to an expert when given the same information; while Study 2 shows how my 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. I conducted experiments (Appendix A, Figure 43) with 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 (Appendix A, Figure 44) varying the probability of crossover and probability of mutation to determine the effect on the *SRM*. These tests indicated that a cross-over probability of 0.8 and a mutation probability of 0.05 gave the best robustness. Tests were also conducted where the number of strands operated on by the VSGA varied and the total number of chromosomes varied (Appendix A, Figure 45). VSGA configurations with one strand operations produce better quality results overall but the with less than one percent difference in the robustness metric. 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 thirty-six 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 16. 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 the results in Figures 16 and 17. This makes using even a simple heuristic like minimum search valuable.

The average improvement in the stochastic robustness versus the expert allocation for Study 2 is shown in Figure 17. 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.

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 I 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 (Appendix A, Figure 46). 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 ten increase in the average heuristic execution time.

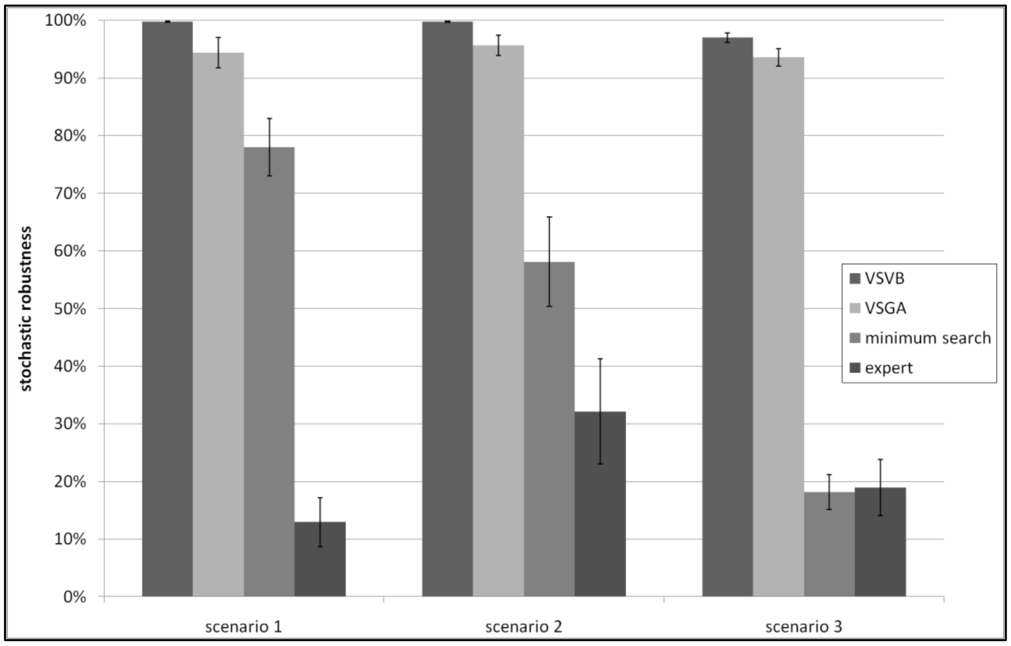


**Figure 17.** Study 2 average heuristic stochastic robustness values for minimum search allocation, VSGA allocation, VSVB allocation, and expert allocation using known values for search team search and movement rates. Error bars show a 95% confidence interval.

## 2.8 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. I 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 teams.  Additionally, 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.

# 3. STATIC ALLOCATIONS USING PETRI NETS[2]

## 3.1 Introduction

In the fast paced, dynamic world of military operations, cordon and search missions (a.k.a. village search) are conducted daily to capture insurgents, secure villages, and confiscate contraband. These missions are complex, and the planning for them involves allocating search teams to target buildings in a manner that efficiently achieves the mission objectives. It is a difficult task to make accurate, feasible plans. This requires knowledge of the search teams involved (e.g., size of the team, team composition, team search rate), the target buildings (e.g., area, location, disposition of inhabitants), the weather (e.g., precipitation, temperature, humidity), and other parameters. Frequently, a unit tasked with a village search mission is given a time constraint for the completion of the mission. This constraint affects the development of the plan and the manner in which it is conducted. To add to the complexity of the planning, participating elements can include: soldiers, Military Working Dogs (MWD), Explosive Ordinance Detachments (EOD), military aircraft, Unmanned Aerial Vehicles (UAV), and electronic surveillance. Given these difficult planning factors, it is a complex problem to create plans for a cordon and search mission that are robust against environmental uncertainties. This

[2] The contents of this chapter are published in P. Maxwell, A. A. Maciejewski, H. J. Siegel, and J. Potter, "A cordon and search model and simulation using timed, stochastic, colored Petri Nets for robust decision-making," *2010 Spring Simulation Multiconference (SpringSim10)*, Orlando, FL, Apr. 2010.

problem is being addressed by the ISTeC People, Animals, Robots (PAR) laboratory [MaS09].

Despite the frequency and difficulty of developing cordon and search plans, military planners currently rely on simplistic data tables from military Field Manuals (e.g., [FM01], [FM98]) and personal experience to conduct analysis of the mission during the planning phase. The result is plans that vary dramatically in accuracy and quality and therefore the risk to service members is higher than desired by leaders. The variability in accuracy is due to dependence on the quality of the inputs and the calculation methods used by the user. The variability in quality is due to users' cognitive abilities and past experiences that can have a positive or negative effect on the quality of the result.

An automated tool using Petri Nets [JeK09] can improve the mission planning process and can help produce more robust plans. The proposed automated tool would allow: (a) the input of the search area via imagery or shape files, (b) the automatic generation of the Petri Net village model, (c) the input of user selected data, and (d) the analysis of the search mission using stochastic models. The tool would accomplish these tasks within the time constraints of the planning process. The long term goal for this tool is to allocate search teams (e.g., humans, MWDs, EOD, UAVs) to tasks (e.g., building searches), and to calculate the robustness for a resource allocation (the probability of mission completion within the time constraint).

The use of Petri Nets to model and simulate the cordon and search environment can result in more accurate, robust mission plans with reduced risk to service members and can provide valuable data to decision-makers during the planning process. Petri Nets are capable of creating detailed models of a cordon and search mission that account for the uncertainties in the environment. Specifically, timed, stochastic, colored Petri Nets provide the ability to model concurrency in the mission and to introduce stochastic information into the simulation. The choice of parameters represented by stochastic information can vary depending on user preference but can include any quantifiable influence on the mission (e.g., temperature, probability of enemy contact, precipitation). Current military combat simulations are in general knowledge or rules based and deterministic ([MoR06], [SaZ06]). Combat is inherently stochastic and thus many current simulations have limitations due to these design choices. Simulations that use stochastic information exist but they do not use that information to derive robust resource allocations. Petri Nets can model the stochastic variables of the mission and incorporate constraints typically used in the missions (e.g., unit boundary lines, phase lines, direction of attack) into the solution. Then, using Monte Carlo methods, the Petri Net simulation results can be used to build probability mass functions (pmfs) for the search mission and these pmfs can provide information about the robustness of a given plan (resource allocation) to military leaders.

The main contribution of this chapter is a novel timed, stochastic, colored Petri Net model for cordon and search missions that can simulate the search environment including its uncertainties. Additionally, I demonstrate how the results of the simulations can be used via Monte Carlo methods to analyze the robustness of resource allocations and provide robust information to decision-makers. Specifically, the pmfs that result from the Monte Carlo simulations allow an objective comparison of specific resource allocations and thus the selection of an acceptable mission plan.

This chapter assumes that the probability distribution functions (pdfs) for the quantities that are uncertain in the cordon and search environment can be developed. The definition of these pdfs is a separate research problem and is not addressed here. Additionally, the model assumes that only one search team is used per target building. Future research on this problem will remove this assumption.

The remainder of the chapter is organized as follows. Section 3.2 reviews related work in the fields of Petri Nets and military combat simulations. A review of Petri Net basics and a detailed description of the Petri Net cordon and search model is in Section 3.3. Background on robustness research and its application to the Petri Net model is in Section 3.4. In Section 3.5, the simulation set-up is described and results are presented. I present my conclusions in Section 3.6.

## 3.2 RELATED WORK

Numerous models and simulations have been created for combat and other military requirements. These range from theater level simulations that train high-level staffs to game-type simulations that provide realistic simulations for squad/crew-level service members. These simulations are generally deterministic and are focused on simulating combat for analysis of different plans or for basic tactics training. As a result, the field of stochastic simulations has a lot of room to explore.

The work of [SaZ06] focuses on simulations of basic tactical scenarios. The goal of the authors is to use artificial intelligence algorithms to create human agents that behave "realistically" in a computer simulation of a Military Operations in Urban Terrain

(MOUT) environment. They use hierarchical Finite State Machines to generate the agent behavior and thus the model is deterministic.

A graphical simulation is found in [MoR06] that simulates combat at the operational level. Here the tool uses a graphical interface that allows planners to input mission parameters and then observe the results of the simulation. The purpose of the tool is to allow a quick comparison of different courses of action and thus provide planners and leaders with valuable data for the decision-making process. The tool uses a rules based system in its simulation engine and does not account for stochastic variables.

The research of [PoC04] is another planning/decision support tool. This simulation environment is a strategic/operational-level analysis tool that uses mixed linear programming and stochastic sampling of pdfs to evaluate mission plans for the Air Force. The tool attempts to optimize a mission plan through cycles of simulation and evaluation. This simulation does incorporate stochastic elements to account for uncertainty but only for the purpose of determining point values for model elements (e.g., damage, combat durations, force ratios).

Similarly, the authors of [JaS95] use stochastic information in their mission plan simulation engine. Their work samples pdfs to calculate values including the duration of combat, the probability of winning, and the expected number of casualties. The goal of the work was to create a combat simulation that differed from the standard deterministic models and that would provide more accurate results. Their work does not attempt to develop resource allocations or to compare courses of action.

The field of Petri Net research is well developed. Many researchers have applied Petri Nets to the simulation of production environments, computers and their networks, and even resource allocation [JeK09]. The ability of Petri Nets to model concurrency, interdependencies, and conflict are very useful in these and many other domains.

The optimization of production lines is an important area of research (e.g., [KuG98], [RiN05]). Determining the most productive allocation of resources (e.g., machines, workers, raw materials) is valuable to industry. Petri Nets can be used to study the optimum levels of resources required and the effects of changing these levels as in [KuG98]. The work of [RiN05] proceeds beyond these basic simulations and uses heuristics to prune Petri Net coverability trees for the purpose of optimizing production systems. In contrast with my work, these studies are deterministic or use a mixture of deterministic and stochastic information.

Petri Nets also have been used to schedule tasks in military scenarios in work such as [ZhK02]. In that work, colored Petri Nets are used to schedule operational/strategic level tasks that meet mission goals and the commander’s intent. Unlike my work though, this tool uses deterministic methods to compare courses of action. The simulation checks resource requirements against mission types and incorporates timing values to determine plan feasibility. It is a decision-making aid for staffs but it does not evaluate robust resource allocations.

## 3.3 PETRI NET CORDON AND SEARCH MODEL

### 3.3.1 PETRI NET BASICS

Basic Petri Nets are a graphical tool to formally describe systems. They are good at modeling concurrency, synchronization, causality, and mutual exclusion. A basic Petri Net is a four-tuple (*P*, *I*, *O*, *T*) where $\underline{P}$ is the set of places, $\underline{I}$ is the input function, $\underline{O}$ is the output function, and $\underline{T}$ is the set of transitions. If transitions are removed from a Petri Net, then the result is a directed graph.

Places can describe system state information in Petri Nets, while transitions describe events that modify the system state. Arcs, described by the *I* and *O* functions, specify relationships between states and transitions. They are directed and exist only between places and transitions. Tokens specify state (marking), physical data, or information.

Transitions are said to be "enabled" if a token exists at every input place to the transition. Transitions "fire" atomically once they are enabled. Tokens are consumed by transitions from the input places and created at the output places when fired. The number of tokens consumed/created is determined by the number of input/output arcs connected to a transition.

Basic Petri Nets have been extended to include hierarchy, time, stochastic elements, and color. Hierarchy allows the reuse of system components and facilitates simpler representations. The extension of time allows Petri Nets to model systems that require timing to properly represent them. The addition of stochastic elements provides the ability to represent uncertainty in models. Finally, color allows tokens to have information associated with them, places to have data types, and for transitions and arcs to have expressions (e.g., conditionals, rules). Together, these extensions allow Petri Nets to model a variety of systems and properties.

A timed, stochastic, colored Petri Net is a nine-tuple, $CPN = (P, A, T, \Sigma, V, C, G, E, I)$ [JeK09]. Rather than use this formal Petri Net notation, I will discuss the model in terms of the application domain.

## 3.3.2 CORDON AND SEARCH MODEL

A cordon and search mission can be modeled by a timed, stochastic, colored Petri Net. I will refer to this Petri Net model as the cordon and search model (CSM). The power of the Petri Net system allows for uncertainty, heterogeneous search teams, graphical control measures, and additional elements to be modeled and simulated in the system. The CSM assumes (though it is not reliant on the assumption) that the underlying Petri Net structure can be automatically generated from input images or shape files of the target area and user inputs (e.g., mission constraints, number and type of search teams, intelligence data).

In the CSM, the set of *places* contains the search teams' starting points, the target buildings ($T_j$), and portions of the village road structure denoted as road segments ($RS_k$). The set of *transitions* contains the stochastic timing and decision elements that calculate the time to search target buildings, the time to move along a road segment, expressions for randomly selecting target buildings, etc. These elements are described in more detail in subsequent paragraphs. The CSM consists of simple data types (e.g., integer, real, string) along with a constructed data type *Search Team* that contains information on the modeled search teams (e.g., team type, mean search rate, mean movement rate). Each place in this model is associated with the data type *Search Team*. There is a "guard function" that contains Boolean conditions for transitions, such as a check that enables only road segment transitions that lead to the desired target building. The arc expressions in the CSM are conditional statements that make decisions on factors such as the building search time for different search team types (e.g., human vs. MWD). This description of the CSM elements is a necessarily brief overview because a complete description of the CSM is beyond the space limitations of this dissertation.

The CSM is a hierarchical model that utilizes several modules. The highest level module is the *village* module. In this module (shown in Figure 18), search team starting places, target buildings, and the road network are connected to create the village environment. Search team starting places model the teams' initial starting location and can account for the time of travel to the target village if desired. Target buildings are connected to the road network at the appropriate road segment. Logic conditions are added to ensure each building is only searched once. The wet bulb temperature (a measurement combining temperature and humidity) is incorporated into the module by sampling the wet bulb temperature pdf and then performing a look-up table operation in the heat factor function. This factor is generated from military work/rest tables for wet bulb temperature [FM98]. The heat factor function provides a value in the interval [0.1, 1.0] that represents work/rest ratios [FM98] for forces at a given wet bulb temperature. It is important to understand that a variety of uncertainties and modeling elements can be captured in the CSM. The CSM's strength is only limited by the detail requirements of the user. Finally, control functions are incorporated into the *village* module to allow multiple simulation trials and output of trial data.

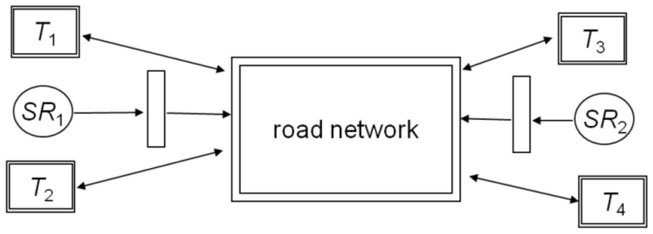


**Figure 18**. Example village module models village shown in Figure 2 with two search teams ($SR_i$), four target buildings ($T_j$), and a road network module.

A sub-module of the village module is the *target building*. This element models the search of a target building by a search team. The module accounts for the impact of uncertainties, such as the search team search rate, the actual size of the building, and the effect of the wet bulb temperature on the search rate. Each of these uncertainties is represented by a pdf. To calculate the search time, the search team's simulation search rate is determined by sampling the search rate pdf. Then the simulation building size is determined by sampling the building size pdf. The search time is calculated by dividing the simulation size of the building by the combination of the search rate multiplied by a heat factor that incorporates the wet bulb temperature. If the team is a MWD team, then the MWD search rate is used assuming the dog team has not already worked for more than one hour. The module is designed to account for the dog's work/rest schedule. If the dog's work allowance is exceeded, then the team searches the building at the human search rate. If required, other uncertainties can be captured in this module (e.g., enemy contact, casualty evacuation, improvised explosive devices).

The *road network* module contains all the systems' road segments required to model the environment. The road segments are connected to other road segments via arcs and transitions.  Additionally, to prevent cycles, the road segments have guard expressions that ensure a token moves to an adjacent road segment only if it is a practical path to the destination target building and it is not the road segment just traversed.  A search team's simulation movement rate is determined by sampling the movement pdf. Thus the time to traverse a road segment is computed by dividing the length of the road by the combination of the simulation movement rate multiplied by the heat factor

A sub-module of the *road network* module is the *road segment*.  A road segment is a section of road as determined by the image processing software.  The segments may be defined by natural breaks (e.g., intersections, dead-ends) or by a predefined length. The road segment module models the time required for a search team to traverse the road segment.  Uncertainties modeled in this element include the movement rate of the search team and the effect of the wet bulb temperature upon the movement rate.  As with the target building module, other factors such as road conditions, obstacles, and enemy contact can easily be added to the model if desired.

The CSM outlined in the paragraphs of this sub-section to this point describe an unconstrained CSM.  Military planners may choose to constrain the mission by adding graphical control measures such as unit boundaries and restrictive phase lines.  Elements such as these sub-divide the graph and limit the set of possible solutions to the mission. <u>Unit boundaries</u> prevent specific search teams from searching target buildings outside their boundary.  <u>Restrictive phase lines</u> control search team movement by not allowing teams to proceed beyond the phase line until all teams reach the line.  An example village with constraints is shown in Figure 19 and its corresponding road network is shown in Figure 20.  The CSM can effectively capture these user constraints.

For example, a unit boundary is implemented by removing arcs and transitions in the road network to segment the network into sectors while maintaining the requirement that each sector has at least one search team assigned to it. A restrictive phase line requirement is incorporated by the addition of a *phase line place* in the village module that holds search team tokens until all tokens in the mission are present. Other constraints (e.g., maximum distance between teams, initial target building assignments, direction of advance) can be added to the model as desired by the user.

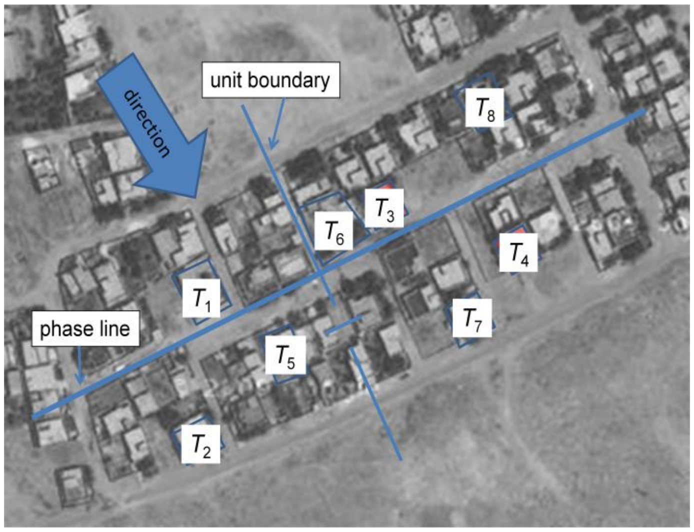


**Figure 19.** Example cordon and search mission with eight target buildings ($T_j$), a unit boundary, a restrictive phase line, and a direction of advance.

The CSM is able to capture uncertainties in the combat environment and user constraints. In a simulation trial of the model, the paths the tokens traverse and the buildings they search constitute one possible solution to the mission. In the next section, I will discuss how the CSM can be used via Monte Carlo simulations to determine robust resource allocations.

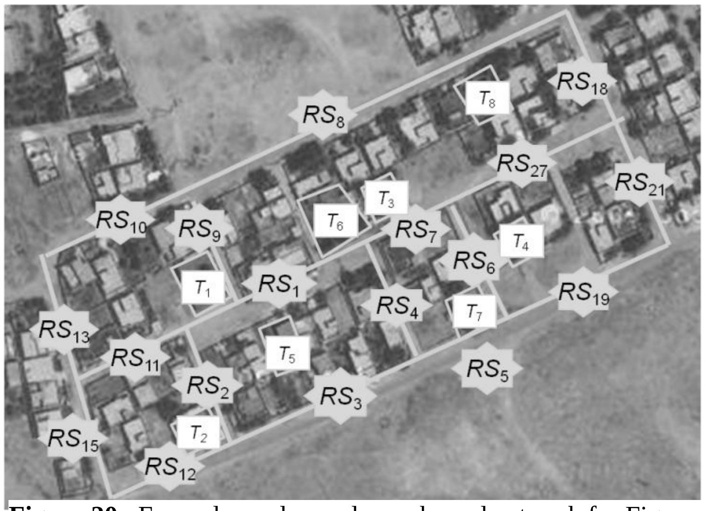


**Figure 20.** Example cordon and search road network for Figure 17. $RS_j$ is road segment $j$.

## 3.4 ROBUSTNESS

### 3.4.1 ROBUSTNESS BACKGROUND

It is often desired that the solutions of resource allocation problems be robust. This is true of the cordon and search mission domain. Military leaders desire a solution to the problem that is able to withstand the effects of uncertainty. This is evidenced by the common military saying, "the plan goes out the window as soon as we cross the line of departure." However, the definition of the term 'robustness' is frequently unclear.

The authors of [AlM08] created three questions to help correct this problem by defining robustness for a given scenario. The three robustness questions that help define robustness for a system are: (1) What behavior of the system makes it robust? (2) What uncertainties is the system robust against? and (3) Quantitatively, exactly how robust is the system? A system might be defined as robust if it meets a specific time deadline or if a maximum number of tasks from a set of tasks is completed. Uncertainties that a system could be robust against are variations between the expected completion/execution times for a task and the actual completion/execution times. The most important question in the list though is question 3. A quantitative measure for robustness is required to avoid subjective evaluations. This can be accomplished through the creation of an objective function for the system. For example, the objective function could be the probability that a resource allocation for a computing system will complete by its time deadline. The result can then be compared to other resource allocations to determine the most robust allocation (i.e., the allocation with the highest probability of meeting the time deadline) from the analyzed set of allocations.

The FePIA (Features, Perturbation parameters, Impact, Analysis) method [AlM04] is a framework for measuring robustness. The robustness metric for a given resource allocation can be developed using the FePIA method, 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. As these steps are followed, the robustness metric for a system is created. In the next sub-section, I will apply the robustness questions and the FePIA procedures to the cordon and search environment.

### 3.4.2 ROBUSTNESS AND THE PETRI NET CSM

In the cordon and search mission domain, the robustness questions can be answered differently depending on the performance objective in question. To illustrate the application of the questions to this domain, I will define the system to be robust if all the search teams complete their building searches prior to the mission deadline time

(*MDT*). This is a time constraint by which all teams must complete. The uncertainties the system is robust against include variability in the ideal search rates of the search teams, variability in the ideal movement rates of the search teams, the effects of wet bulb temperature on the search rates, and the error in the estimated size of the target buildings. The uncertainties that can be modeled by the CSM are not limited to these choices. Any uncertainty that can be modeled may be incorporated into the CSM. To answer the third robustness question, I will use the FePIA method.

First, I define the search resource completion time for team $i$ ($RCT_i$) to be the performance feature for this system. If there are $m$ search teams, then there are $m$ performance features. The perturbation parameters for step 2 are the same uncertainties outlined in the previous paragraph. The impact of the perturbation parameters on the performance feature is that they will either increase or decrease the value of $RCT_i$. For example, target building $j$ may have an estimated area of 500 $m^2$ based on an image processing software's calculations. However, the true building size may be double that due to the presence of an undetected second story. Thus, it will take longer to search the building and the value of $RCT_i$ will increase.

The analysis to quantify the robustness of the system is complex and the reader is referred to [MaM09] for a more detailed discussion. In summary, the time to search a given building is a complicated function of factors that include the building characteristics, search team characteristics, and pdfs of the relevant uncertainties. This results in a pdf that represents the building search time. To calculate $RCT_i$ it is necessary to convolve (assuming independence) the building search time pdfs for the buildings assigned to that team and the associated road segment traversal pdfs resulting in the search team's completion time pdf. With the CSM, a single trial in the simulation corresponds to one data point in the search resource completion time pmf. Thus, using Monte Carlo simulation methods and assuming a large enough sample size, the actual search resource completion time pdf for each search team can be approximated by the simulation search resource completion time pmfs.

It is assumed that the search teams have adequate supporting elements to operate independently and therefore the search resource completion times are independent in unconstrained and selected constrained scenarios. Thus, the stochastic robustness metric, *SRM*, of [MaM09] is defined as the product of the all the teams' probabilities of completing before the *MDT*. That is, the *SRM* is the probability that all teams will complete by the *MDT*. For each simulation trial of the CSM, I can determine the maximum $RCT_i$ and then use the results of multiple trials to create a pmf of the maximum search team completion times. From this pmf, I can sum the probabilities of the completion times that are less than or equal to the *MDT*. This sum is the probability that the mission will be complete by the *MDT*. The advantage of this technique over the convolution-based technique of [SmZ09] is that if the assumption of independence is removed (for example by the constraint of a phase line) then the CSM is still valid.

The Monte Carlo generated pmfs and cumulative mass functions (cmfs) of the CSM can be used in several ways to identify robust resource allocations or to provide robust decision-making data. In the first way, the cmf of a cordon and search mission could be used to determine with a given probability (e.g., 95%) the maximum length of time it would take to complete the search of all the target buildings. This would allow leaders to accurately plan for the required supporting resources. Another use is that specific resource allocations could be compared via their cmfs to determine which allocation has the highest probability of completing prior to the *MDT*. Additionally, the effects of changing aspects of the mission (e.g., varying search teams, varying placement of unit boundary lines) could be examined by comparing the resulting cmfs. In the next section, I show the results of simulations to further explain this point.

## 3.5 SIMULATION RESULTS

The CSM was created and simulated using the ExSpecT software version 6.1 [Del09]. This free package supports timed, stochastic, colored Petri Nets and allows file I/O to capture the results of simulations. The pdfs used in the simulations for the uncertainties were the PERT beta distribution [Ris09]. This closed distribution emphasizes the mean over the minimum and maximum values. The distribution uses the expected value, a minimum value, and a maximum value as parameters for the distribution. For the building area, the minimum value was 0.25 times the estimated area and the maximum was four times the estimated area. The search rate pdf minimum was 0.05 $m^2/s$ and the maximum was twice the mean search rate of the search team. For the movement rate pdf, the minimum value was set to 0.1 m/s and the maximum value was twice the mean movement rate of the search team. The pdfs used for this or any of the modules is not crucial to the correctness of the model. It is assumed that the pdfs to be used in practical application can be approximated through the use of historical data or data from a combat training center.

For this work, a model was built of an eight target building village with twenty-one road segments (shown in Figures 19 and 20). Each building was assigned an estimated area and the roads were assigned a length based on the input image's scale. Three search teams were modeled of which one was a military working dog team. Each search team was assigned mean values for its search and movement rates based on military Field Manual movement rate tables [FM01]. The military working dog team was also assigned a mean value for its search rate when using the dog team to search. An initial wet bulb temperature value of 76 degrees Fahrenheit was used as well.

In the first batch of simulation trials, three cordon and search scenarios were run: unconstrained; unit boundary constrained; and unit boundary with restrictive phase line and direction of attack constrained. A total of 1000 trials were performed for each scenario. The differences between trials are the resulting sample values used from the perturbation pdfs (e.g., building size, search rate, movement rate), the allocation of target buildings to search teams (randomly allocated using a uniform distribution for the initial building to be searched and then after each team completes a given building search), and the assignment of search teams to a particular side of the unit boundary (randomly allocated using a uniform distribution while maintaining at least one team on a side). The resulting pmfs are shown in Figure 21. It is clear that adding the constraints to the mission increased the mean of the distributions.

The second batch of simulation trials used three specific search team allocations selected from different bins of the constrained unit boundary with restrictive phase line and direction of attack simulation results. The team allocations were chosen such that all were unique allocations and search teams searched opposite sides of the unit boundary. For reference, the selected allocations are shown in Table 2. For each of the allocations, 1000 trials were conducted and the resulting cmfs are shown in Figure 22 (to construct a cmf that corresponds to a given pmf, for each time *i* on the x-axis, the cmf y-axis value is the sum of the pmf probabilities from time 0 to time *i*). The trials differed by the value of the perturbation parameters. These were generated by sampling the pdfs in the CSM. Using a *MDT* value of 4 hours, allocation 3 is the most robust of the three allocations with a 90.3 percent probability of team completion prior to the *MDT*. Thus, if a specific team allocation is provided, the CSM can evaluate it to determine the stochastic robustness metric of the allocation. This result can then be used to select the most robust allocation from a given set of allocations.

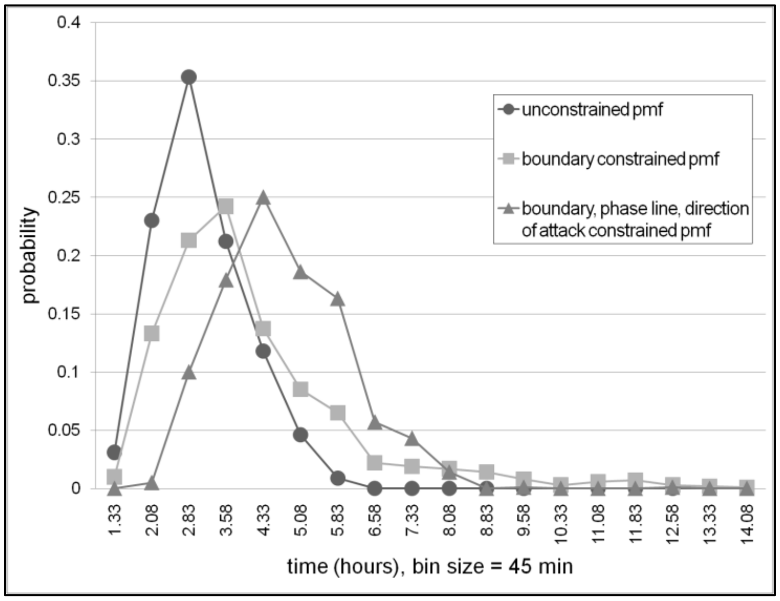


**Figure 21.** Cordon and search Petri Net model pmfs for eight buildings, three teams, and 1000 random trials each.

**Table 2.** Three team allocations with three search teams (*ST*) each and the assigned paths. In the path section, numbers represent road segment ids and $\boxed{i}$ represents a search of target building *i*.

| allocation | ST | path |
|---|---|---|
| 1 | 1 | 1-$\boxed{1}$ -1- 11 - 15 - 12 -$\boxed{2}$- 2-1-$\boxed{5}$ |
| | 2 | 1 -$\boxed{6}$ - 1 - 4 - 5 - 6 – $\boxed{4}$ |
| | 3 | 7 -$\boxed{3}$ - 7 - 27 - 18 - 8 -$\boxed{8}$ - 8 - 21 - 19 - 5 –$\boxed{7}$ |
| 2 | 1 | 1 –$\boxed{1}$ - 1 - 11 - 15 - 12 - 2 -$\boxed{2}$ - 1 -$\boxed{5}$ |
| | 2 | 1 -$\boxed{6}$ - 1 - 4 - 5 -$\boxed{7}$ |
| | 3 | 7 -$\boxed{3}$ - 7 - 27 - 18 - 8 -$\boxed{8}$ - 8 - 18 - 27 - 6 –$\boxed{4}$ |
| 3 | 1 | 1 -$\boxed{6}$ - 1 - 4 - 5 -$\boxed{7}$ |
| | 2 | 7 -$\boxed{3}$ - 7 - 27 - 18 - 8 -$\boxed{8}$ - 8 - 18 - 27 - 6 –$\boxed{4}$ |
| | 3 | 1 -$\boxed{1}$ - 1 - 11 - 15 - 12 -$\boxed{2}$ - 12 - 15 - 11 - 1 –$\boxed{5}$ |

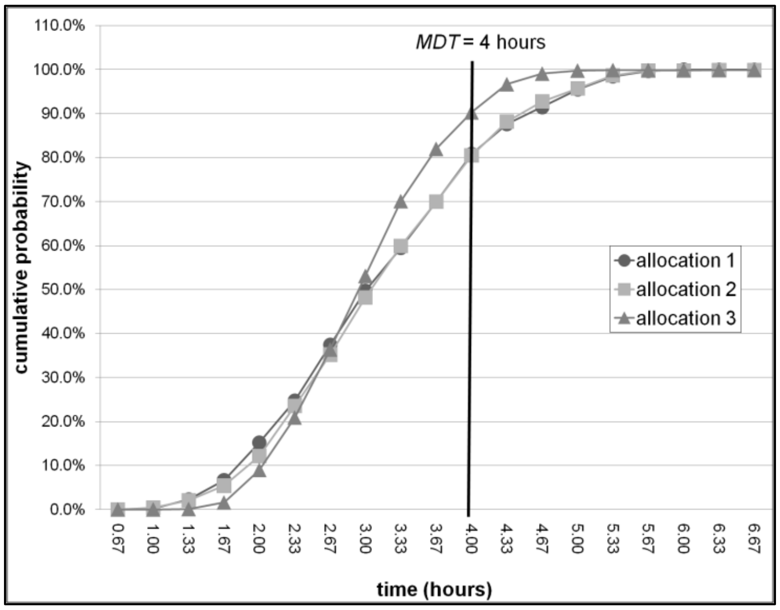


**Figure 22.** Comparison of three team allocation cmfs for 1000 trials each. MDT time is 4 hours.

## 3.6 CONCLUSIONS

In this chapter, I have proposed a novel cordon and search model (CSM) for a cordon and search mission using timed, stochastic, colored Petri Nets. The power of Petri Nets to model uncertainty, concurrency, and causality make it an excellent fit for this environment. Contrary to other military simulations, my model uses stochastic information in the form of pdfs to make the cordon and search model and then uses the results of Monte Carlo simulations to evaluate the robustness of a team allocation or provide robust information to decision-makers. The simulation trials demonstrate the validity of the CSM and highlight its power.

Future work in this domain includes developing a software package that can automatically generate the underlying Petri Net structure from input imagery and user inputs. Additionally, the actual pdfs for the perturbation parameters are required to increase the accuracy of the model. Finally, more perturbation factors can be incorporated into the model to improve the realism of the simulation.

# 4. DYNAMIC RESOURCE ALLOCATIONS

## 4.1 INTRODUCTION

A frequent challenge for military planners is the rapid creation of mission plans (resource allocations) for operational requirements. One of the most common mission types encountered on the contemporary battlefield is the “cordon and search” or military village 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. In the military domain, 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 deadlines. An example village search problem is shown in Figure 23.

In a static planning environment as described in [MaM12], 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 environment, that include, but are not limited to, varying search rates, encounters with the enemy, weather impacts on the search rates, and mechanical malfunctions affecting movement rates. However, in general, the problem of resource allocation in heterogeneous parallel and distributed computing is NP-complete (e.g., [Cof76], [IbK77]). 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.

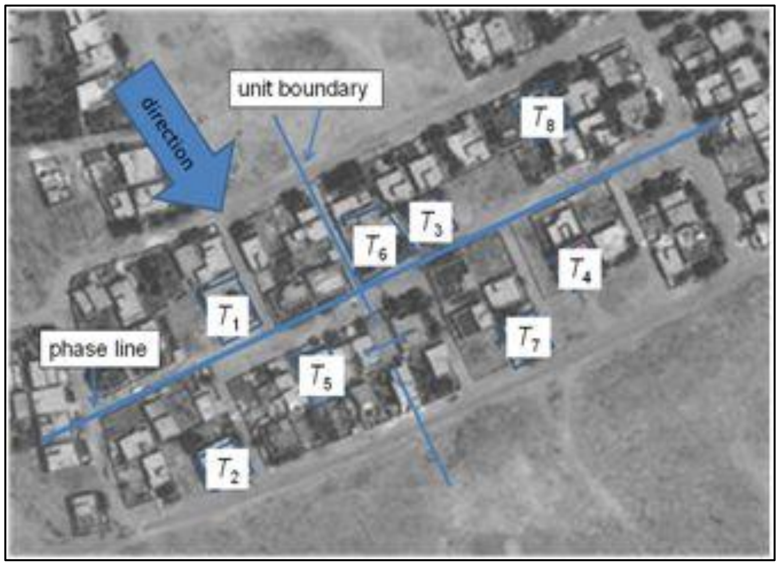


**Figure 23.** An example village search mission with eight target buildings (*Tj*), a unit boundary, a restrictive phase line, and a direction of advance.

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.

In [MaM12], the *Robust People, Animals, and Robots Search* (*RoPARS*) planning tool was introduced. This tool automated the static creation of robust mission plans. Adding automated dynamic reallocation components to the RoPARS tool 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 my 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 chapter include: 1) a framework for conducting dynamic military mission replanning, 2) 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 chapter is organized as follows. Section 4.2 presents work related to the dynamic village search problem. Section 4.3 provides an overview of the robustness metric developed in [AlM04] and [ShS08] and its application in village searches. In Section 4.4, the resource allocation heuristics I developed for the village search model are described. The simulation results are shown in Section 4.5, and in Section 4.6 I present my conclusions.

## 4.2 RELATED WORK

In terms of the modeling domain, the research areas that are most similar to my work are: combat simulations, vehicle routing, and travelling 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 my work differs from these works in substantial ways.

Many of the works relating to military models focus on deterministic models (e.g., [Ayd04], [Bab05]) or models that use stochastic information only for limited purposes such as movement direction (e.g., [FuL10], [PoC04]). These tools tend to focus on models for training purposes or for strategic-level simulation. In general, they are not used for combat resource allocation and decision making. The 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.

The vehicle routing problem (discussed in works such as [DeD92], [Pot96]) has similarities to my 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 travelled, minimizing completion time, and minimizing monetary cost. The problem is different from my domain in that I 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; 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 ([Pap77]) TSP and mTSP problem but here I review only works that use genetic algorithms to find a solution because those are the closest comparisons. The work in

[TaL00] is a constrained problem domain that considers time spent at a node but not distance between nodes. Unlike my 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 [SaB99], 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. Additionally, the problem does not have limiting constraints on the solution such as the boundary lines of the village search problem. Lastly, the work in [YuJ02] transforms a multiple robot mine clearing problem into an mTSP problem that is solved using a genetic algorithm. Like other works 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 works 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.

The works of [MaY10] and [Yan08] 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 my heuristics.

An example of research that attempts to account a priori for uncertainty is [PoB11], which examines aircraft allocation to satisfy airlift demands. Their work uses dynamic programming methods to develop resource allocations. Unlike my models, they model the uncertainty in the environment as the sets of all possible aircraft (search team) states and then solve the allocation problem using that information. In my model, I do not attempt to define these states because it would be computationally intractable in my problem and instead use *probability mass functions* (*pmfs*) to describe the uncertainty.

The authors of [KaH08] 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 my method does but it does account for updated scalar information that is not available during static planning.

## 4.3 ROBUSTNESS AND THE VILLAGE SEARCH MISSION

### 4.3.1 ROBUSTNESS CONCEPTS

Robustness can have a variety of definitions 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 in [AlM04]. 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 in [MaM12]. Provided in this section is a brief summary of the robustness procedure and how it is applied to this domain. For a detailed description, the reader is referred to [MaM12].

The robustness metric for a given resource allocation can be developed using the *FePIA* (*Features*, *Perturbation parameters*, *Impact*, *Analysis*) method [AlM04], 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 modelling 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 are unknown during the planning phase of a village search mission. For this work, the perturbations parameters that are modelled are the search rates and movement rates for each search team. For example, the physical fitness of a group of soldiers will impact positively or negatively their movement rate over a given terrain path. These values are represented as pmfs. These pmfs are created using normal distribution functions with mean values based on rates found in military field manuals.

The impact of the perturbation parameters on the performance features can be described mathematically. These values can positively or negatively affect the search completion time of a search team. 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, 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 can be calculated by convolving the pertinent pmfs (i.e., building completion time pmfs, path movement completion time pmfs) within each phase line area. The completion time for a particular phase line area is the maximum of the search team completion time pmfs for that area. Finally, the convolution of the phase line area completion time pmfs (all phase line areas combined) is performed and the corresponding cumulative mass function is evaluated at the *MDT*. The resulting *stochastic robustness metric* (*SRM*) is the probability that a user-specified level of system performance can be met. In this domain, the *SRM* is the probability that all search teams will complete the search of all their assigned target buildings prior to the *MDT* [MaM12].

### 4.3.2 DYNAMIC VILLAGE SEARCH

The dynamic village search environment consists of search teams, target buildings, a statically created team allocation plan, and the completion time pmfs for the search teams. It is assumed that an implemented static team allocation will meet a minimum *SRM* threshold value (e.g., 40%). Allocations with an *SRM* below the threshold *SRM* would realistically have their *MDT*s adjusted or the mission would be cancelled due to its lack of feasibility. Once a mission begins, events can occur that cause the *SRM* to increase or decrease. Some events (e.g., addition of a target building, loss of a search team) 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 chapter I focus on two cases where evaluation must be done prior to implementation: first (Case 1) I test my dynamic heuristics against the effects of cumulative delays; and then (Case 2) I consider a situation where a significant delay occurs at a single point.

Regardless of the case, my method involves evaluating potential team 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. I 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 the 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 my 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, the heuristics are designed for rapid execution (they run concurrent with building searches) and the ability to produce a robust result within the time available.

## 4.4 DYNAMIC RESOURCE ALLOCATION HEURISTICS

### 4.4.1 DYNAMIC MIN-MIN HEURISTIC

The dynamic min-min is inspired by the original two-phase greedy heuristic in [IbK77]. 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 evaluates the updated completion time pmfs for each search team and finds 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 travel time) 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 is scheduled and the building is removed from the reallocation eligible set. 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.

## 4.4.2 COMPLETION TIME REDUCTION HEURISTIC

The completion time reduction heuristic (*CTR*) 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 24, 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.

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 travelled between the swapped building and the building prior to it in the allocation ordering (Figure 25). 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 a $swap_{lim}$ 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.

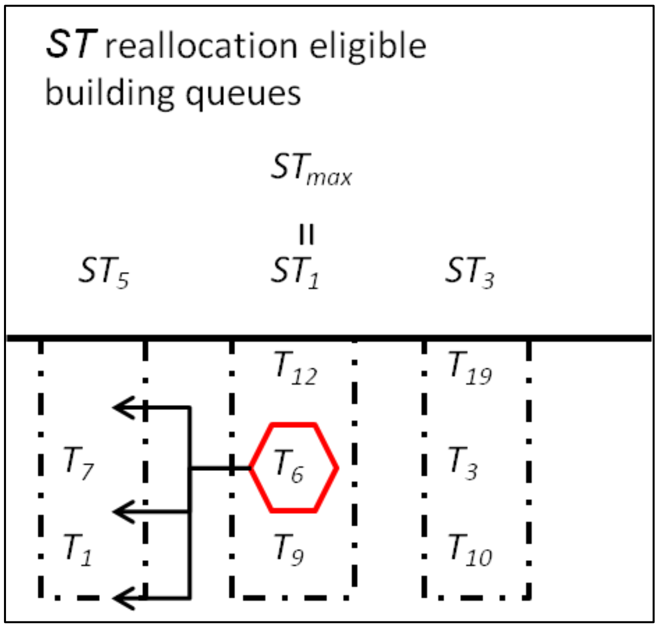


**Figure 24.** Completion time reduction heuristic move phase with building $T_5$ evaluated in $ST_5$'s allocation queue in all possible orderings.

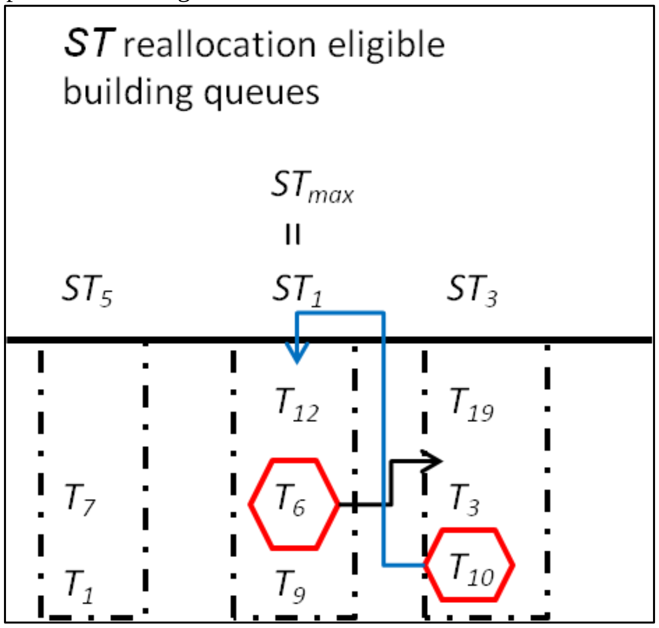


**Figure 25.** Completion time reduction heuristic swap phase with buildings $T_5$ and $T_{10}$ swapping search teams and inserting into the destination order that results in the minimum movement time between buildings.

## 4.4.3 DYNAMIC GENETIC ALGORITHM

The dynamic genetic algorithm (*DGA*) is a modified version of the village search genetic algorithm (*VSGA*) presented in [MaM12]. In this version, the population's chromosomes are composed of "strands." A strand 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 26). The number of strands in a chromosome is defined by the number of phase line areas from the current state of the search until the end of the search inclusive (Figure 27). 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 operations. Finally, the size of the population for the heuristic is determined experimentally.

| | | | | | |
|---|---|---|---|---|---|
| bldg | 1 | 14 | 0 | 28 | 31 |
| *ST* | 0 | 4 | 4 | 0 | 4 |

**Figure 26.** Dynamic genetic algorithm 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).

The basics of the dynamic genetic algorithm are outlined in Pseudocode 1 and described here. First, the number and size of the strands is calculated based upon the current phase line area and the buildings eligible for reallocation. The heuristic then uses the current allocation 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*) for the selection of 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 [BlT95]. 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. 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 and the next generation is evaluated starting with SUS selection to limit the population size to a fixed value.

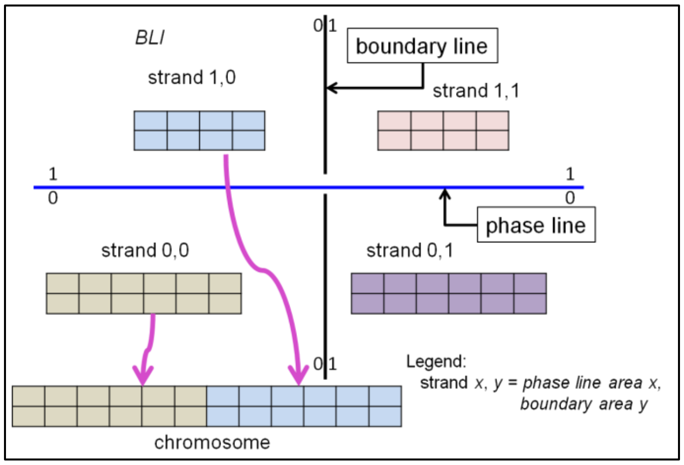


**Figure 27.** Example dynamic genetic algorithm chromosome with the boundary line area of interest (*BLI*) indicated and the current state of the search in phase line area 0. For the example BLI, only strands 0,0 and 0,1 are used for the DGA chromosome operations.

**Pseudocode 1.** Dynamic Genetic Algorithm

**Input:** Current resource allocation plan, list of current buildings being searched, the *BLI*, and updated completion time pmfs.
**Output**: New resource allocation plan.

```
1. initialize set of offspring to empty
2. determine number and length of strands
3. randomly generate initial population using the current allocation as a seed chromosome
```

```
4. while stopping criteria not met
5.    evaluate fitness function for each chromosome
6.    select next population using Stochastic Universal Sampling
7.    for (number of chromosomes/2)
8.          randomly select two chromosomes from current population
9.          with probability $P_{SC}$ do scheduling crossover
10.         with probability $P_{MC}$ do matching crossover
11.         insert new chromosomes into the set of offspring
12.   end
13.   for each chromosome in current population
14.         with probability $P_{SM}$ do scheduling mutation
15.         with probability $P_{MM}$ do matching mutation
16.         insert new chromosomes into the set of offspring
17.   end
18.   merge the set of offspring with the current population, make set of offspring empty
19. end
20. output allocation with the highest fitness
```

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. The crossover operation is performed on the same strand within each parent chromosome. For example, a crossover operation could be performed on strand 0,0 (Figure 27) 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 because studies in [MaM12] show a significant increase in the computational load for a small increase in the robustness of the solution.

The scheduling crossover operation operates as follows (Figure 28). A single crossover point is randomly chosen and then the sub-strand to perform the crossover on is randomly chosen. Unlike crossover operators described in [WaM97] 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.

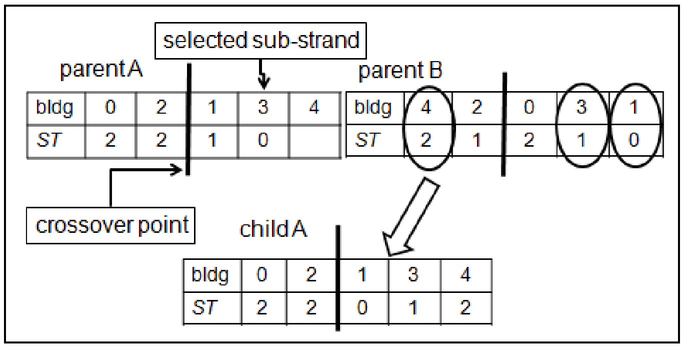


**Figure 28.** Dynamic genetic algorithm scheduling crossover example.

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 team it has in parent *B* (see Figure 29). The operation is then repeated with the parent chromosomes reversed.

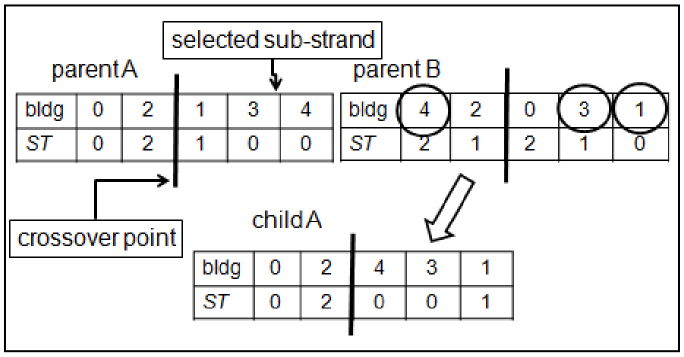


**Figure 29.** Dynamic genetic algorithm matching crossover example.

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 chapter.  Examples for the scheduling and matching mutation operators are shown in Figures 30 and 31.

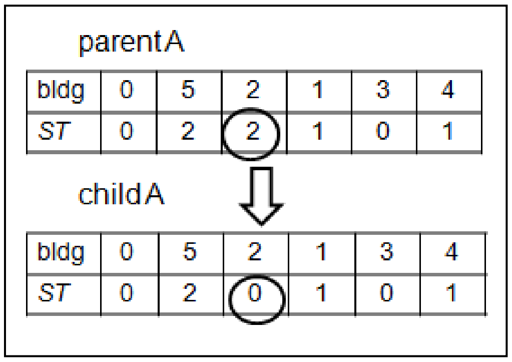


**Figure 30.** Dynamic genetic algorithm scheduling mutation example.

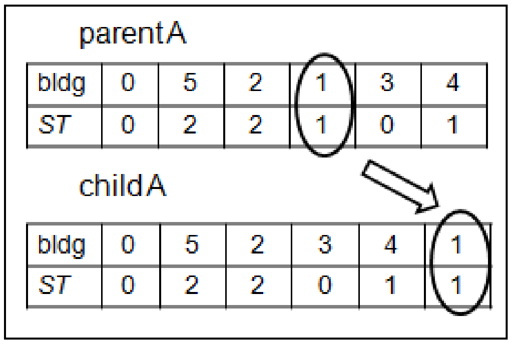


**Figure 31.** Dynamic genetic algorithm matching mutation example.

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.

## 4.5 SIMULATION RESULTS

The results discussed here are based on simulations of village search missions. An event driven simulator was created that uses statically created allocations from [MaM12] to conduct trials where search teams search their assigned target buildings. One complete trial simulates each search team completing in order the search of its assigned target buildings. 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 and search pmfs to determine the duration of an event.

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 set of search teams consists of one military working dog team and with the rest being human teams. In my experience, the dog teams conduct searches faster. This difference is captured in the generation of search team search rate pmfs, i.e., dog team pmfs are generated using a higher mean than the human search teams.

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 meters) for long uninterrupted roads. Scenario 1 is based on the city (*city a*) in Figure 32, 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 33, 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.

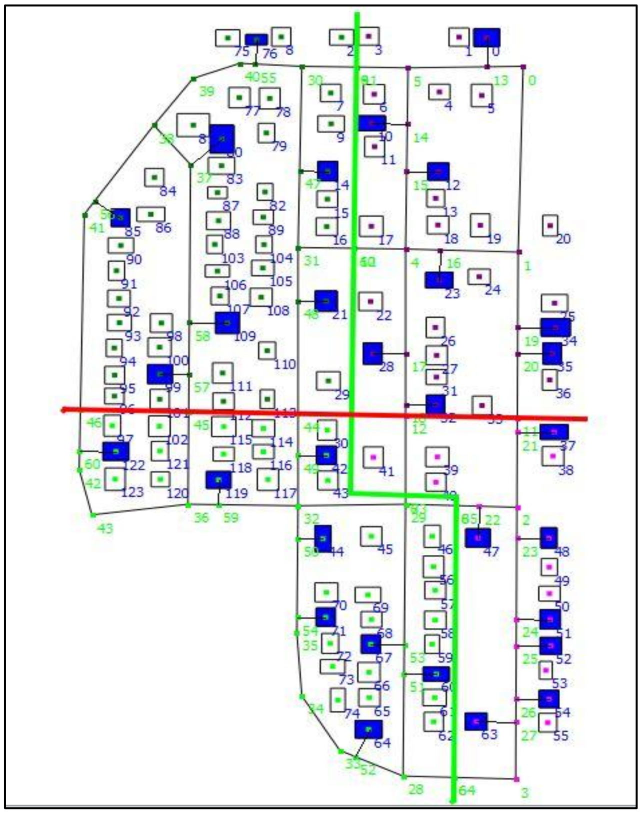


**Figure 32.** Screen shot from the Robust People, Animals, and Robots (ROPARs) animation tool [MaM12] showing the village structure used for scenarios 1 and 2 (city a) generated using ERSI shapefiles. The solid colored buildings shown are the target buildings for scenario 1.

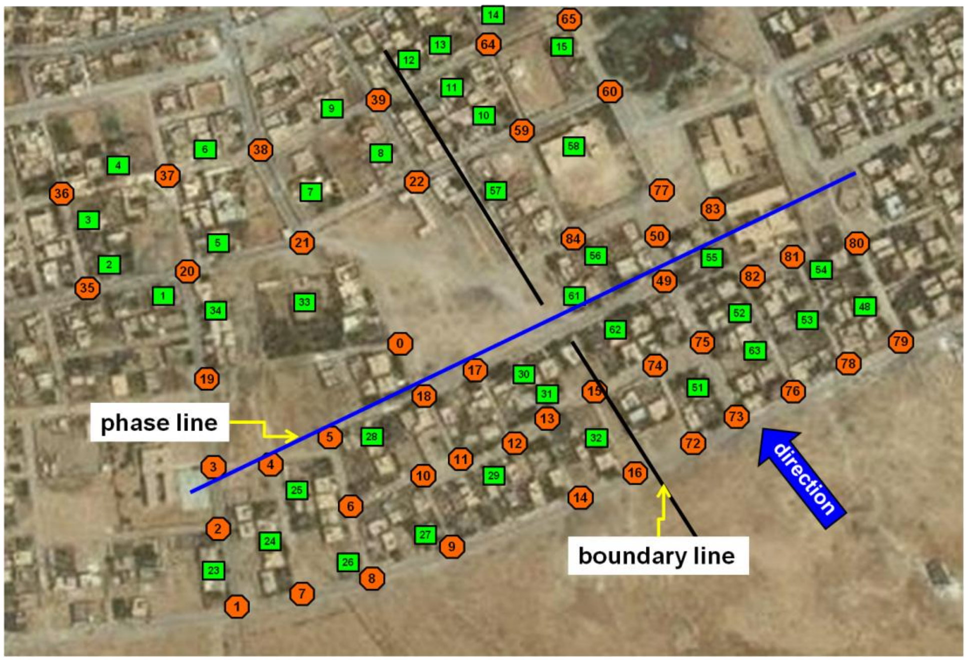


**Figure 33.** Village structure used for scenarios 3 and 4 (city b) based on satellite imagery of Kubaysa, Iraq. The actual target buildings (squares) and road nodes (octagons) shown are used in scenario 4.

Static allocations were created for each of the scenarios using the three heuristics used in [MaM12] (minimum search heuristic, village search genetic algorithm, and village search variable beam heuristic) and a new two phase greedy heuristic (maximum-minimum search heuristic) that is a variant of the minimum search heuristic. The maximum-minimum search heuristic (*Max-Min Search*) finds the maximum completion time building/*ST* pair for each *ST* in the first phase of the heuristic and then the minimum pairing from the first phase during the second phase of the heuristic. The four static allocation heuristics can be grouped into the simple or complex heuristic category. The minimum search heuristic and the maximum-minimum search heuristic are members of the simple category due to their greedy approach. The village search genetic algorithm 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 scenarios had 100 trials run for each of the four static allocations for a total of 1600 trials per dynamic heuristic.

All the heuristics used an overhead cost factor of five 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 to 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. 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 (e.g., [MaY10], [Yan08]), 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.

The results shown in Figure 34 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 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. 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 maximum-minimum 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 maximum-minimum search heuristic. Despite this, some success was found when the static allocation heuristic was more complex (village search variable beam search and village search genetic algorithm). In these cases, the CTR had over a 14% improvement and the DGA had a 32% improvement. The performance of the dynamic allocation heuristics against only greedy/simple static heuristics is shown in Figure 35 for comparison. The explanation for this disparity is that the complex static heuristics (i.e., Village Search Variable Beam, Village Search Genetic Algorithm) consider the full pmfs when conducting the static allocation plan and develop much better solutions than the greedy heuristics.

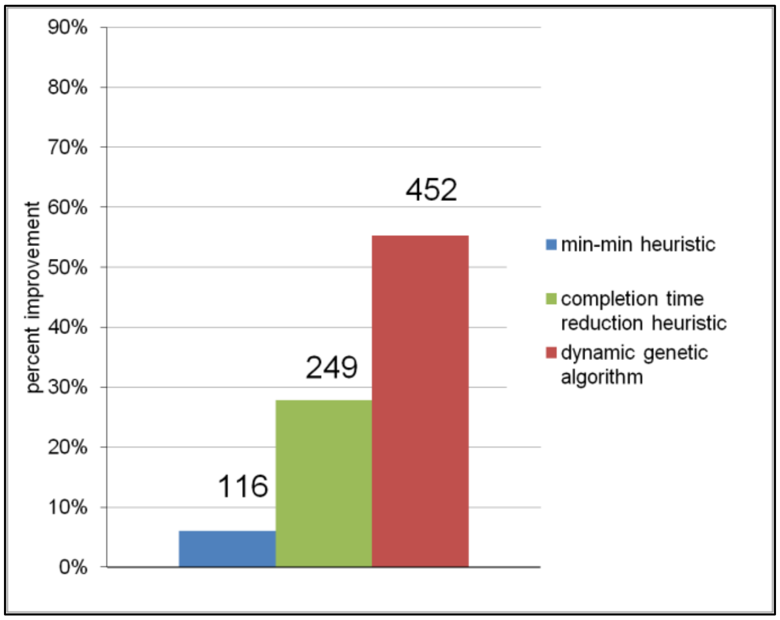


**Figure 34.** Percentage of trials improved (case 1) that meet the *MDT* using dynamically created team allocations given that the statically created team allocations (using all static heuristics) failed to meet the *MDT*. The actual number of trials with improvement out of 1600 is shown above each bar.

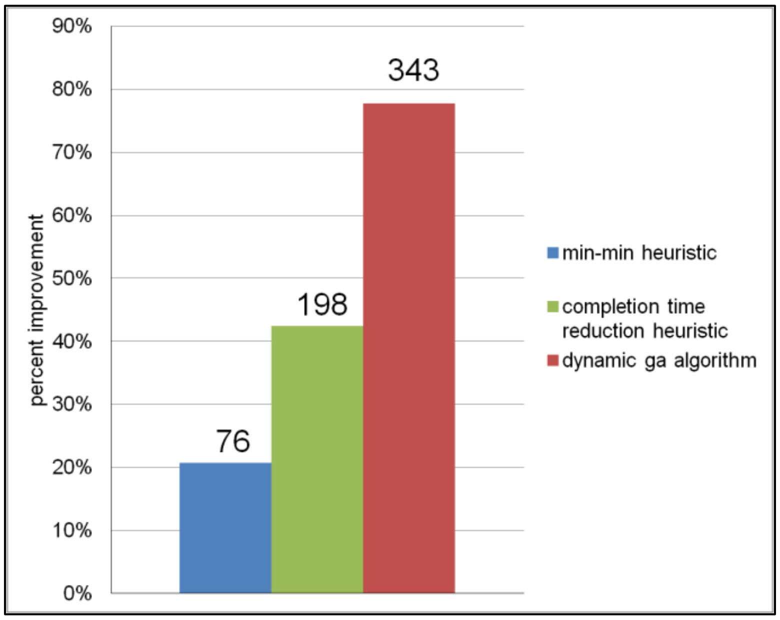


**Figure 35.** Percentage of trials improved (case 1) that meet the *MDT* using dynamically created team allocations given that the statically created team allocations (using greedy heuristics) failed to meet the *MDT*. The actual number of trials with improvement out of 800 is shown above each bar.

As mentioned previously, there are many types of events that can occur that may cause a static allocation to miss the *MDT*. I tested the dynamic heuristics against a second case (case 2) for additional insight on how they would perform under more severe circumstances. In this case, I use the same scenarios as before but I 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 sampling the building's search completion time cmf at 99.9 percent (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 36, 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 (e.g., see Figure 29). This is not a surprising result because 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 37, but still are fast enough for the problem sizes explored.

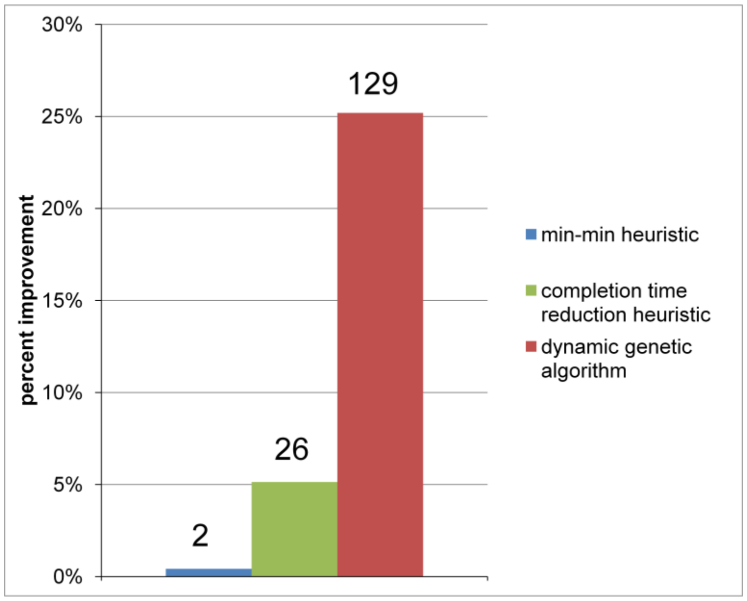


**Figure 36.** Percentage of trials improved (case 2) that meet the *MDT* using dynamically created team allocations given that the statically created team allocations (using all static heuristics) failed to meet the *MDT*. The actual number of trials with improvement out of 900 is shown above each bar.

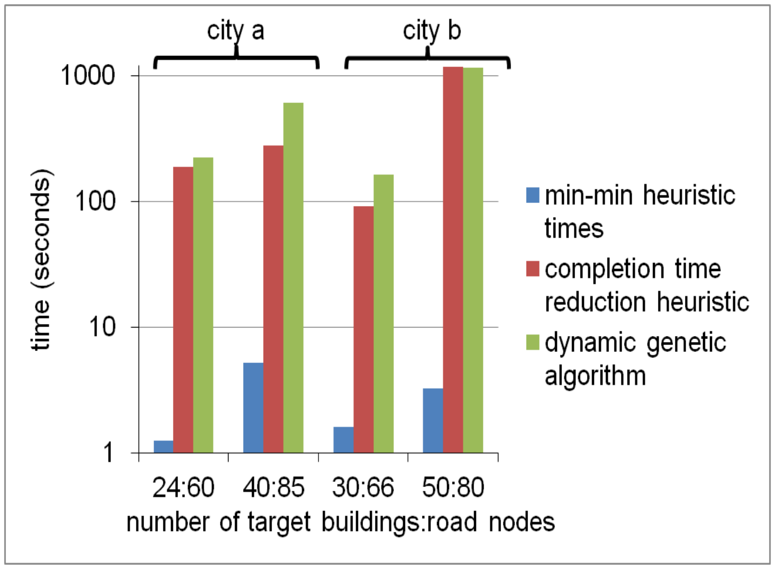


**Figure 37.** Average heuristic execution time (on a logarithmic scale) for an entire village search, averaged over 400 trials versus city complexity (number of target buildings: number of road nodes) for the min-min, completion time reduction, and dynamic genetic algorithm heuristics.

The dynamic heuristics would realistically be used at brigade-level units or lower and thus the number of *ST*s 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 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. Additionally, the CTR looping and DGA generational limits can be adjusted to conserve time. Finally, the CTR and DGA are “anytime algorithms” [Zil96] 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. To test the heuristics’ execution times it is necessary to run simulations, time the heuristic execution within the simulation, and then repeat 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. The main factor in evaluating the heuristic execution time is the overhead of the simulator and not the heuristic execution time. Given this is a stochastic problem the heuristic evaluation is embedded in the simulation of the village search and cannot be evaluated independently of the simulation. Therefore, for larger scenarios, it is not currently practical to evaluate 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 I explored are either based on properties of the static allocation or related to the *MDT*. 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 current completion time pmf's mean to the *MDT* and conducts a dynamic reallocation attempt if the current mean is greater than the *MDT*.

As Figures 38-41 show, there is a trade-off between the average 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. For the triggers tested, it is clear that triggering is not beneficial except for reducing the total system computational load. A better method of dealing with closely completing building searches may be to exclude the next building in the search queues for all search teams when considering the reallocation eligible buildings. 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 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.

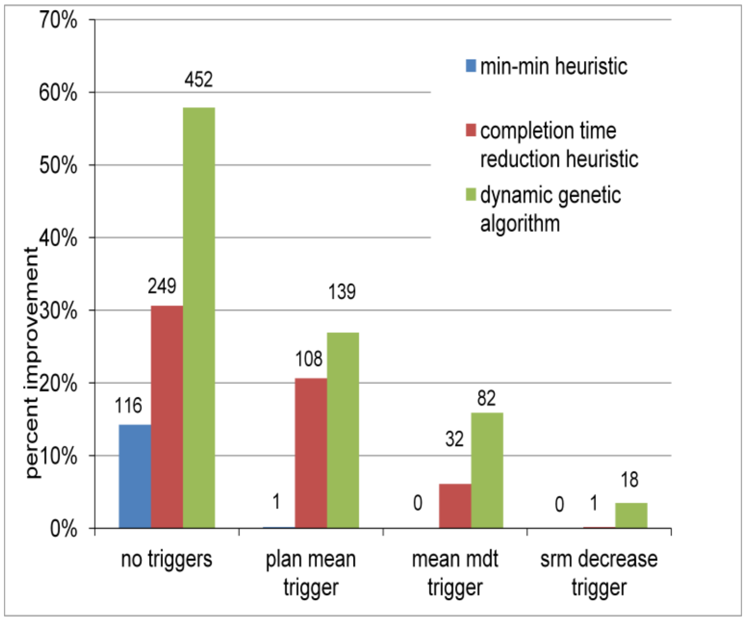


**Figure 38.** Comparison of dynamic allocation heuristics with no triggers and dynamic allocation heuristics with one of three trigger types: mean plus standard deviation; mean plus *MDT*; and *SRM* decrease. The actual number of trials with improvement out of 1600 is shown above each bar.

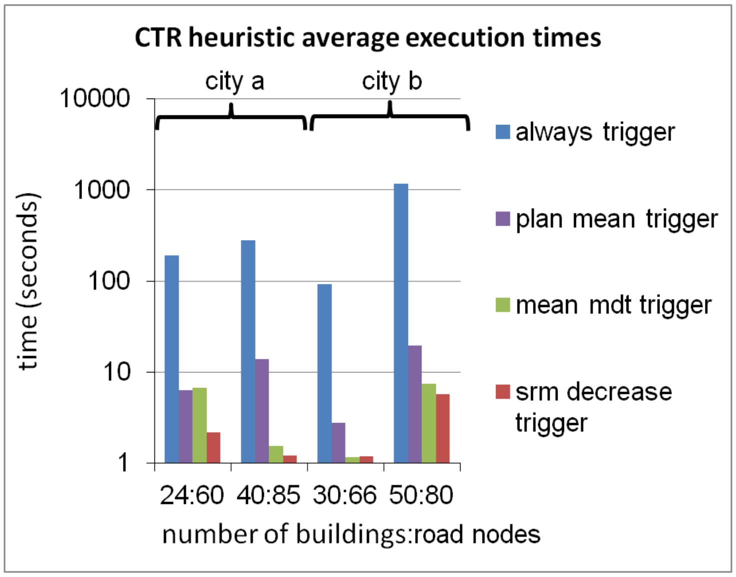


**Figure 39.** Execution time in seconds (on a logarithmic scale) for an entire village search, averaged over 400 trials versus city complexity (number of target buildings: number of road nodes) for the completion time reduction heuristic for four test conditions: no triggers, *SRM* decrease trigger, mean *MDT* trigger, and plan mean trigger on a logarithmic scale. The 95 percent confidence interval maximum values range from 3.9% of the mean for always trigger to 17.4% of the mean for *SRM* decrease trigger of the mean.

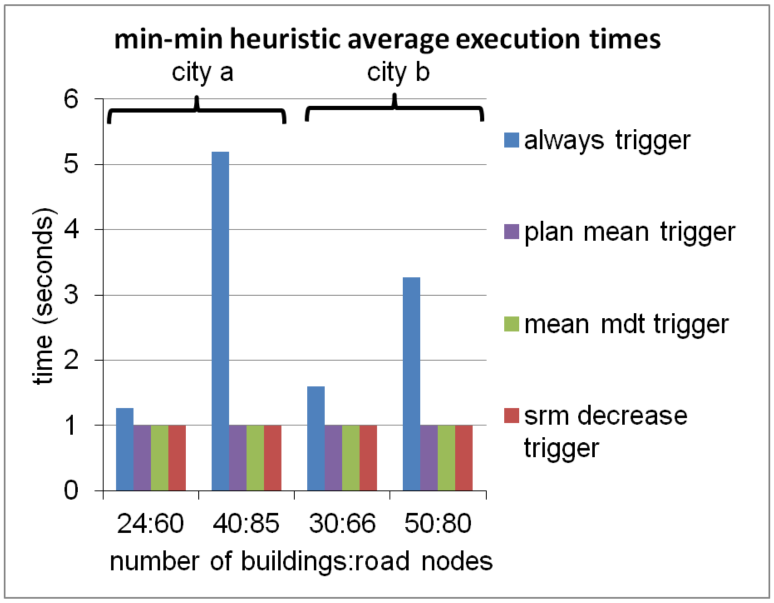


**Figure 40.** Execution time in seconds for an entire village search, averaged over 400 trials versus city complexity (number of target buildings: number of road nodes) for the min-min heuristic for four test conditions: no triggers, *SRM* decrease trigger, mean *MDT* trigger, and plan mean trigger on a logarithmic scale. The 95 percent confidence interval maximum values range from 2.0% for *SRM* decrease trigger to 9.9% of the mean for mean mdt trigger.

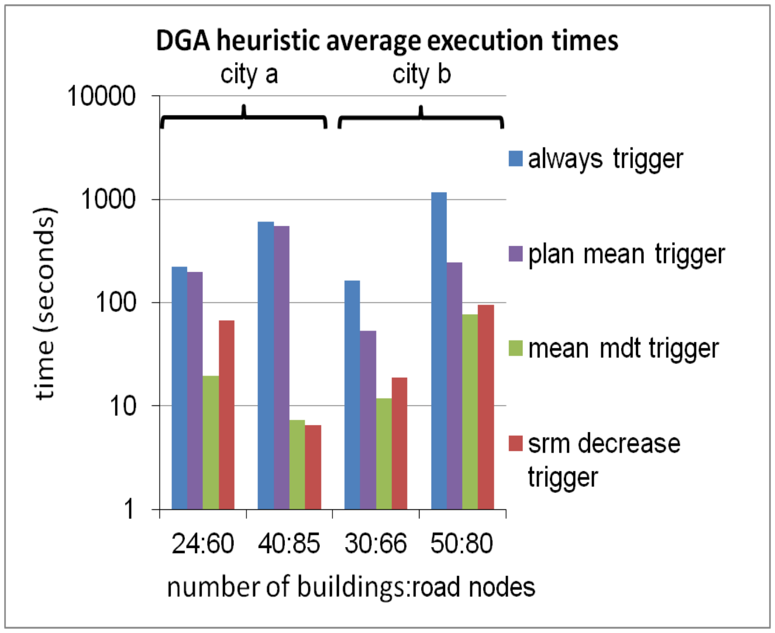


**Figure 41.** Execution time in seconds (on a logarithmic scale) for an entire village search, averaged over 400 trials versus city complexity (number of target buildings: number of road nodes) for the dynamic genetic algorithm for four test conditions: no triggers, *SRM* decrease trigger, mean *MDT* 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.

## 4.6 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). The results show that there are many times when these dynamic reallocations can produce new allocations that meet the mission constraints when the static plans fail. When the static resource allocation heuristic is relatively fast and simplistic (e.g., greedy heuristics) the dynamic heuristics perform better and produce allocations with higher stochastic robustness values. Additionally, the two main heuristics (completion time reduction and dynamic genetic algorithm) produce robust solutions that account for uncertainty in the environment while 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., addition of target buildings, loss of a search team), expanding the size of the test scenarios, and experimenting with other triggers.  Further work can be done on changes to the dynamic genetic algorithm 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.

# 5. CONCLUSIONS

Village search is an important and dangerous military operation. Using resources in an effective manner is essential. A tool with the power to automatically generate village models, develop resource allocation plans using a GUI and heuristics, process real-time mission feedback, and develop dynamic resource allocation plans to improve results is needed. This dissertation explored several automated tools such as the Robust People, Animals, and Robots (RoPARs) software to assist military leaders with making critical robust resource allocations decisions in static and dynamic domains.

The mathematical model presented is flexible and can be tailored to meet user requirements in terms of the modeled perturbation parameters and the selected performance feature. The static and dynamic heuristics created are implementable on current military computing systems (laptops) and thus immediately usable. The speed of the military decision cycle, which consists of detecting an enemy action, deciding on an appropriate reaction, and then performing an action, is a factor in determining the winner of an encounter. The results show that robust allocations can be created in a timely manner, thus freeing military planners to focus on other critical tasks and allowing a quicker decision cycle. This faster decision cycle will provide a tactical advantage on the battlefield as friendly units can then get inside the enemy's decision cycle. Lastly, the robustness metric allows comparison of resource allocation plans and, with the incorporation of perturbation parameters that include likelihood of attack, support the production of safer plans.

In [MaS09], it was demonstrated that the application framework of the village search model is not limited to military village search. The concepts can be adapted to domains such as, civilian search and rescue missions, community policing, and sensor placement. This flexibility allows for many new research areas.

In the long term, as the ability to model the subject matter improves, the village search model can evolve. Each of the different search team types can be modeled in the aggregate by collecting data for the uncertainties (e.g., search rates, movement rates) at combat training centers. As technology improves, each team and individual within the team will be modeled using embedded feedback sensors. This will allow the precision of the system to improve. As this happens, control methods will have to be developed that avoid unintended consequences such as search teams negatively influencing the dynamic algorithms by intentionally moving slower to avoid conducting building searches.

The power of Petri Nets to model uncertainty, concurrency, and causality make it an excellent fit for this environment and the simulation trials demonstrate its validity and power. However, it is difficult to input the physical model of the village into the Petri Net tools I explored and thus this resource allocation method was abandoned in favor of the RoPARs tool. Research into automatic Petri Net model generation from terrain model data could make this area more exploitable.

Future static domain research work areas include improving the RoPARS GUI to allow automatic village search model creation using larger cities. The current software only works reliably with relatively small cities. Testing the resource allocation heuristics on these larger problem sizes will further validate their design. Complementary work can be done to allow the software tools to determine the best locations for phase lines and boundary lines given a set of target buildings and search teams. This would further automate what is now a subjective, manual task. Additionally, research into how and where to reposition idle teams (i.e., teams waiting before moving across a phase line) to improve allocation *SRM*s can be conducted.

In the dynamic domain, research should be extended to model other dynamic events that may require reallocation (e.g., addition of target buildings, blockage of a road segment). Work can be done on how to handle the loss of a search team or other catastrophic events and to develop criteria that trigger mission termination. Experimenting with other dynamic triggers is another area for research. Finally, further work can be done on changes to the dynamic genetic algorithm 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.

# REFERENCES

[AlM04] S. Ali, A. A. Maciejewski, H. J. Siegel, and J. Kim, “Measuring the robustness of a resource allocation,” *IEEE Transactions on Parallel and Distributed Systems*, Vol. 15, No. 7, July 2004, pp. 630–641.

[AlM08] S. Ali, A. A. Maciejewski, and H. J. Siegel, “Perspectives on robust resource allocation for heterogeneous parallel systems,” in *Handbook of Parallel Computing: Models, Algorithms, and Applications*, S. Rajasekaran and J. Reif, eds., Chapman & Hall/CRC Press, Boca Raton, FL, 2008, pp. 41-1–41-30.

[Ayd04] M. Aydin, *An Exploratory Analysis of Village Search Operations*, Master’s Thesis, Naval Postgraduate School, June 2004, 110 pp.

[Bab05] M. Babilot, *Comparison of a Distributed Operations Force to a Traditional Force in Urban Combat*, Master’s Thesis, Naval Postgraduate School, Sep. 2005, 188 pp.

[Bek06] T. Bektas, “The multiple traveling salesman problem: An overview of formulations and solution procedures,” *Omega: The International Journal of Management Science,* Vol. 34, 2006, pp. 209–219.

[BiM02] J. Bilbao, A. H. Miguel, and H. D. Kambezidis, “Air temperature model evaluation in the north Mediterranean belt area,” *Journal of Applied Meteorology*, Vol. 41, Aug. 2002, pp. 872–884.

[BlT95] T. Blickle and L. Thiele, *A Comparison of Selection Schemes Used in Genetic Algorithms*, Technical Report TIK-Report No. 11, Ver. 2, CEN Lab, Swiss Federal Institute of Technology, Dec. 1995, 67 pp.

[BrF80] J. E. Bruhn, W. E. Fry, and G. W. Fick, “Simulation of daily weather data using theoretical probability distributions,” *Journal of Applied Meteorology,* Vol. 19, Sep. 1980, pp. 1029–1036.

[Cho83] J. S. Choe, *Some Stochastic-Duel Models of Combat,* Master’s Thesis, Naval Postgraduate School, Mar. 1983, 43 pp.

[Cof76] E. G. Coffman, Ed., *Computer and Job-Shop Scheduling Theory*. New York, NY: John Wiley & Sons, 1976.

[DeD92] M. Desrochers, J. Desrosiers, and M. Solomon, “A new optimization algorithm for the vehicle routing problem with time windows,” *Operations Research*, Vol. 40, No. 2 Mar. - Apr. 1992, pp. 342–354.

[Del09] Deloitte Development LLC, *ExSpecT software*, http://www.exspect.com/, accessed Oct 2009.

[FM01] *FM 3-31.1 Army and Marine Corps Integration in Joint Operations*, U. S. Army Training and Doctrine Command, Ft. Monroe, VA, Nov. 2001.

[FM98] *FM 34-8-2 Intelligence Officer’s Handbook*, U.S. Army Training and Doctrine Command, Ft. Monroe, VA, May 1998.

[FuL10] L. V. Fulton, L. S. Lasdon, R. R. McDaniel, Jr, and M. N. Coppola, “Two-stage stochastic optimization for the allocation of medical assets in steady-state combat operations,” *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, Vol. 7, No. 2, Apr. 2010, pp. 89–102.

[HiM08] R. D. Hill, C. P. Moate, and D. Blacknell, “Estimating building dimensions from synthetic aperture radar image sequences,” *IET Radar, Sonar, and Navigation*, Vol. 2, No. 3, 2008, pp. 189–199.

[IbK77] O. H. Ibarra and C. E. Kim. “Heuristic algorithms for scheduling independent tasks on non-identical processors,” *Journal of the ACM*, Apr. 1977, Vol. 24, No. 2, pp. 280–289.

[JaS95] N.K. Jaiswal, Y. Sangeeta, and S.C. Gaur, “Stochastic analysis of combat models under different termination decision rules,” *European Journal of Operational Research*, Vol. 83, 1995, pp. 530–546.

[JeK09] K. Jensen and L.M. Kristensen, *Coloured Petri Nets*, Springer-Verlag, Berlin, 2009.

[KaH08] H. Kanoh and K. Hara, “Hybrid genetic algorithm for dynamic multi-objective route planning with predicted traffic in a real-world road network,” *10th Annual Conference on Genetic and Evolutionary Computation*, Jul. 2008, pp. 657–664.

[KuG98] A. Kumar and L. S. Ganesh, “Use of Petri Nets for resource allocation in projects,” *IEEE Trans. on Engineering Management*, Vol. 45, No. 1, Feb. 1998, pp. 49 - 56.

[LaK99] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, “Genetic algorithms for the travelling salesman problem: A review of representations and operators,” *Artificial Intelligence Review*, Vol. 13, No. 2, 1999, pp. 129–170.

[Luc00] T. Lucas, "The stochastic versus deterministic argument for combat simulations: Tales of when the average won't do," *Military Operations Research: A Journal of the Military Operations Research Society*, Vol. 5, No. 3, 2000, pp. 9–28.

[MaF10] P. Maxwell, R. Friese, A. A. Maciejewski, H. J. Siegel, J. Potter, and J. Smith, "A demonstration of a simulation tool for planning robust military village searches," *Proceedings of the Huntsville Simulation Conference (HSC'10)*, Oct. 2010.

[MaM09] P. Maxwell, A. A. Maciejewski, H. J. Siegel, J. Potter, and J. Smith, "A mathematical model of robust military village searches for decision-making purposes," *2009 International Conference on Information and Knowledge Engineering*, July 2009, pp. 311–316.

[MaS09] P. Maxwell, H. J. Siegel, J. Potter, and A. A. Maciejewski, "The ISTeC People-Animals-Robots laboratory: Robust resource allocation," *2009 IEEE International Workshop on Safety, Security, and Rescue Robotics*, Nov. 2009.

[MaM12] P. Maxwell, A. A. Maciejewski, H. J. Siegel, J. Potter, G. Pfister, J. Smith, and R. Friese, "Robust static planning tool for military village search missions: model and heuristics," *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, to appear.

[MaY10] M. Mavrovouniotis and S. Yang, "Ant colony optimization with immigrants schemes in dynamic environments," *Lecture Notes in Computer Science*, Vol. 6239, 2011, pp. 371–380.

[MoR06] F. Momen and J. W. Rozenblit, "Dynamic decision support in the advanced tactical architecture for combat knowledge system," *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, Vol. 3, No. 1, Jan. 2006, pp. 11–26.

[NaL95] S. K. Nair, L. S. Thakur, and K. Wen, "Near optimal solutions for product line design and selection: Beam search heuristics," *Management Science*, Vol. 41, No. 5, May 1995, pp. 767–785.

[Pap77] G. Papadimitriou, "The Euclidean travelling salesman problem is NP-complete," *Theoretical Computer Science*, Vol. 4, No. 3, Jun. 1977, pp. 237–244.

[PoB11] W. Powell, B. Bouzaiene-Ayari, J. Berger, A. Boukhtouta, and A. George, "The effect of robust decisions on the cost of uncertainty in military airlift operations," *ACM Transactions on Modeling and Computer Simulation*, Vol. 9, No. 4, Article 39, Mar. 2011, pp. 1–19.

[PoC04] D. Popken and L. Cox, "A Simulation-optimization approach to air warfare planning," *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, Vol. 1, No. 3, Aug. 2004, pp. 127–140.

[Pot96] J. Potvin, "Genetic algorithms for the traveling salesman problem," *Annals of Operations Research*, Vol. 63, 1996, pp. 339–370.

[Qui04] M. Quinn, *Parallel Programming in C with MPI and OpenMP*, McGraw-Hill, New York, NY, 2004.

[RiN05]D. Riera, M. Narciso, and C. Benqlilou, "A Petri Nets-based scheduling methodology for multipurpose batch plants," *Simulation*, Vol. 81, No. 9, Sep. 2005, pp. 613 - 623.

[Ris09] RiskAMP.com, http://www.riskamp.com/library/ pertdistribution.php, accessed Nov. 09.

[SaB99] I. Sabuncuoglu and M. Bayiz, "Job shop scheduling with beam search," *European Journal of Operational Research*, Vol. 118, 1999, pp. 390 – 412.

[SaC04] H. Saleh and R. Chelouah, "The design of the global navigation satellite system surveying networks using genetic algorithms," *Engineering Applications of Artificial Intelligence*, Vol. 17, 2004, pp. 111–122.

[Saw97] A. Sawhney, "Petri Net based simulation of construction schedules," *The 1997 Winter Simulation Conf.*, Dec. 1997, pp. 1111 - 1118.

[ShS08] V. Shestak, J. Smith, A. A. Maciejewski, and H. J. Siegel, "Stochastic robustness metric and its use for static resource allocations," *Journal of Parallel and Distributed Computing*, Vol. 68, No. 8, Aug. 2008, pp. 1157–1173.

[ShZ06]Z. Shen and S. Zhou, "Behavior representation and simulation for military operations on urbanized terrain," *Simulation*, Vol. 82, No. 9, Sep. 2006, pp. 593–607.

[SmB07] J. Smith, L. D. Briceño, A. A. Maciejewski, H. J. Siegel, T. Renner, V. Shestak, J. Ladd, A. Sutton, D. Janovy, S. Govindasamy, A. Alqudah, R. Dewri, and P. Prakash, "Measuring the robustness of resource allocations in a stochastic dynamic environment," *21st International Parallel and Distributed Processing Symposium (IPDPS 2007)*, Mar. 2007.

[SmC09] J. Smith, E. K. P. Chong, A. A. Maciejewski, and H. J. Siegel, "Stochastic-based robust dynamic resource allocation in a heterogeneous computing system," *2009 International Conference on Parallel Processing (ICPP 2009)*, Sep. 2009.

[SmS08] J. Smith, H. J. Siegel, and A. A. Maciejewski, “Robust resource allocation in heterogeneous parallel and distributed computing systems,” in *Wiley Encyclopedia of Computing*, B. Wah, ed., John Wiley & Sons, New York, NY, Vol. 4, pp. 2461–2470, 2008.

[SmS09] J. Smith, V. Shestak, H. J. Siegel, S. Price, L. Teklits, and P. Sugavanam, “Robust resource allocation in a cluster based imaging system,” *Parallel Computing*, Vol. 35, No. 7, July 2009, pp. 389–400.

[TaL00] L. Tang, J. Liu, A. Rong, and Z. Yang, “A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex,” *European Journal of Operations Research*, Vol. 124, 2000, pp. 267–282.

[WaM97] L. Wang, A. A. Maciejewski, H. J. Siegel, V. P. Roychowdhury, and B. D. Eldrige, “Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach,” *Journal of Parallel and Distributed Computing*, Vol. 47, No. 1, Nov. 1997, pp. 8–22.

[WeP05] L. Wei and V. Prinet, “Building detection from high-resolution satellite image using probability model,” *Geoscience and Remote Sensing Symposium (IGARSS '05)*, Vol. 6, July 2005, pp. 3888–3891.

[Yan08] S. Yang, “Genetic algorithms with memory- and elitism-based immigrants in dynamic environments,” *Evolutionary Computation*, Vol. 16, No. 3, Sep. 2008, pp. 385–416.

[YuJ02] Z. Yu, L. Jinhai, G. Gouchang, Z. Rubo, and Y. Haiyan, “An implementation of evolutionary computation for path planning of cooperative mobile robots,” *4th World Congress on Intelligent Control and Automation*, June 2002, pp. 1798–1802.

[ZhK02] L. Zhang, L. M. Kristensen, C. Janczura, G. Gallasch, and J. Billington, “A coloured Petri Net based tool for course of action development and analysis,” *Conf. on Application and Theory of Petri Nets: Formal Methods in Software Engineering and Defence Systems*, Vol. 1, 2002, pp. 125-134.

[Zil96] S. Zilberstein, “Using anytime algorithms in intelligent systems,” *AI Magazine*, Vol. 17, No. 3, 1996, pp. 73–83.

## Appendix A

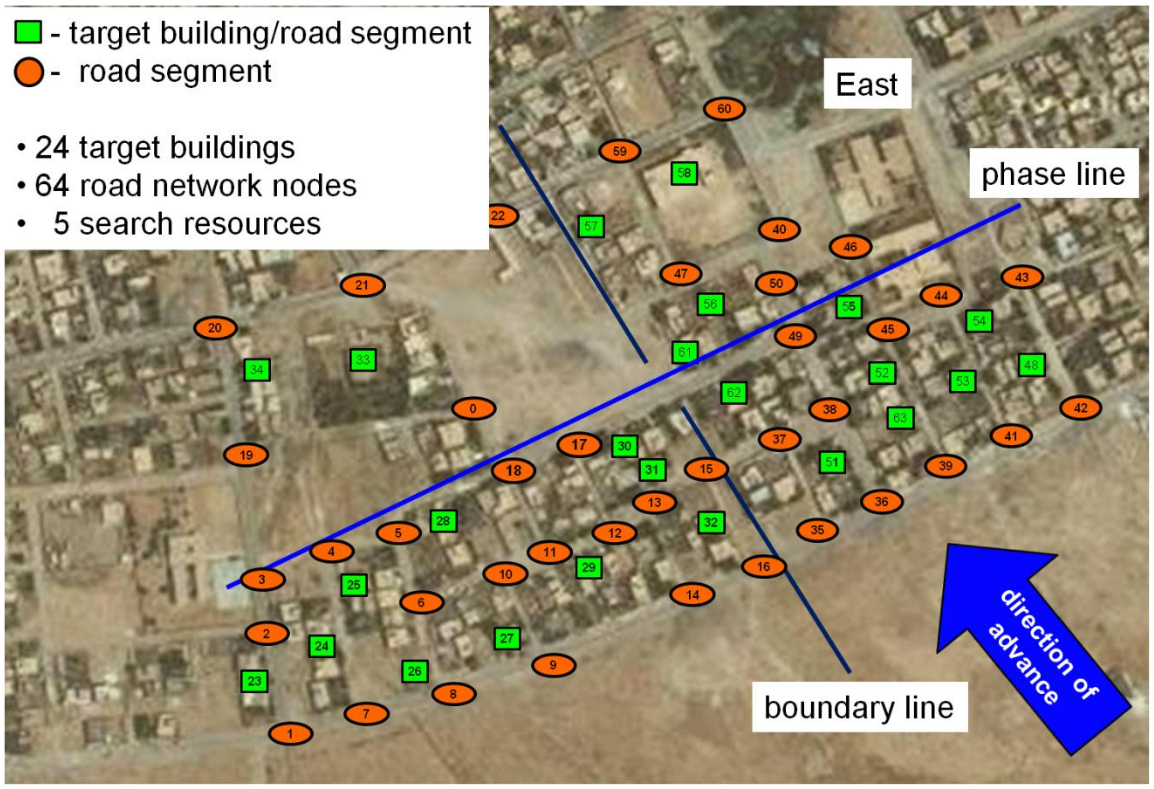


**Figure 42.** Test village used for static heuristic analysis.

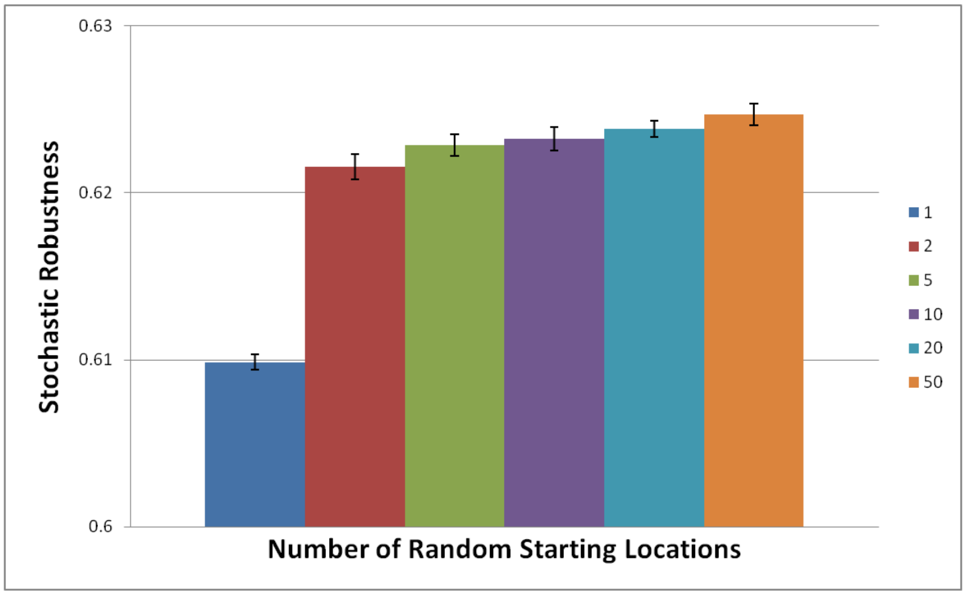


**Figure 43.** Minimum search heuristic stochastic robustness for varying number of random starting locations.

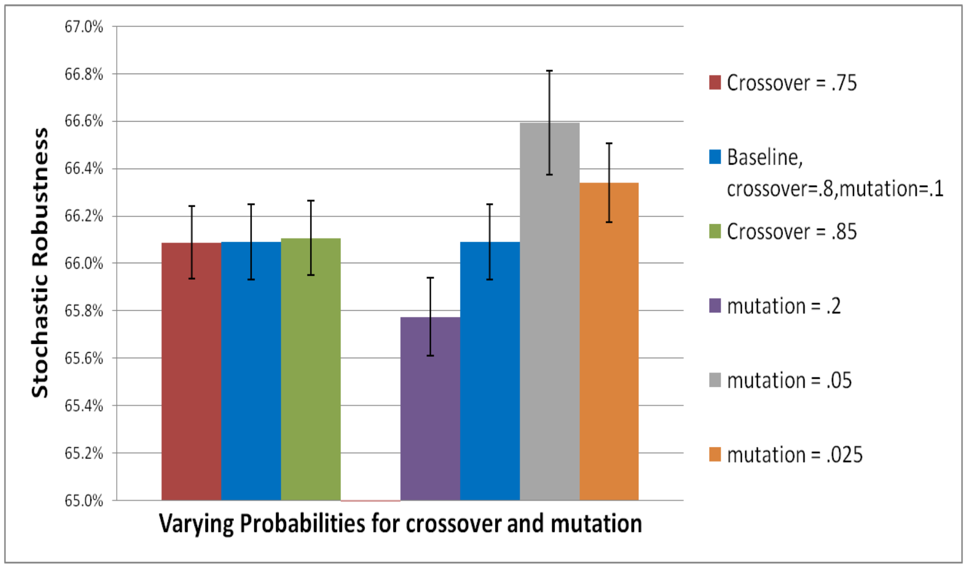


**Figure 44.** Experimental Results for VSGA heuristic varying probability of crossover and mutation.

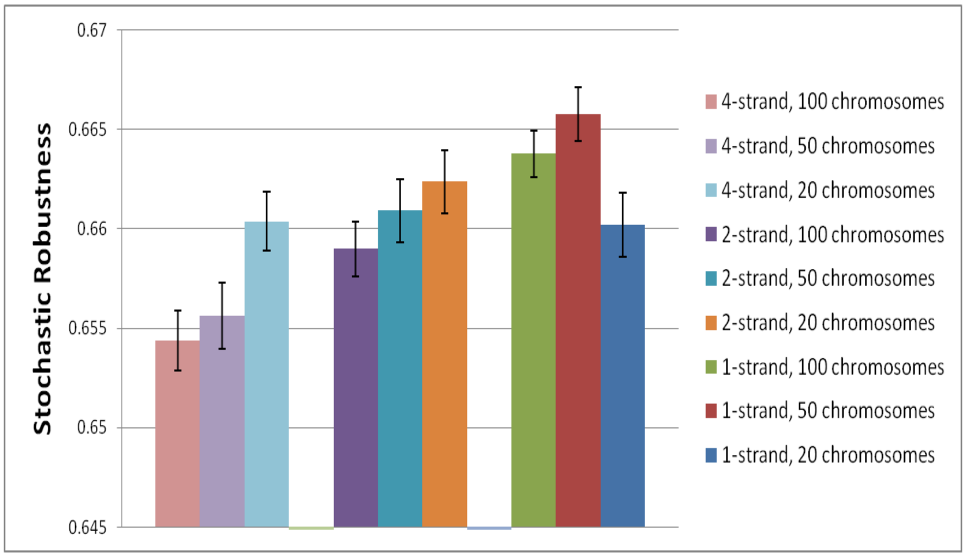


**Figure 45.** Experiments for the VSGA heuristic varying the number of chromosomes in the population and the number of strands upon which crossover and mutation operations are performed per generation.

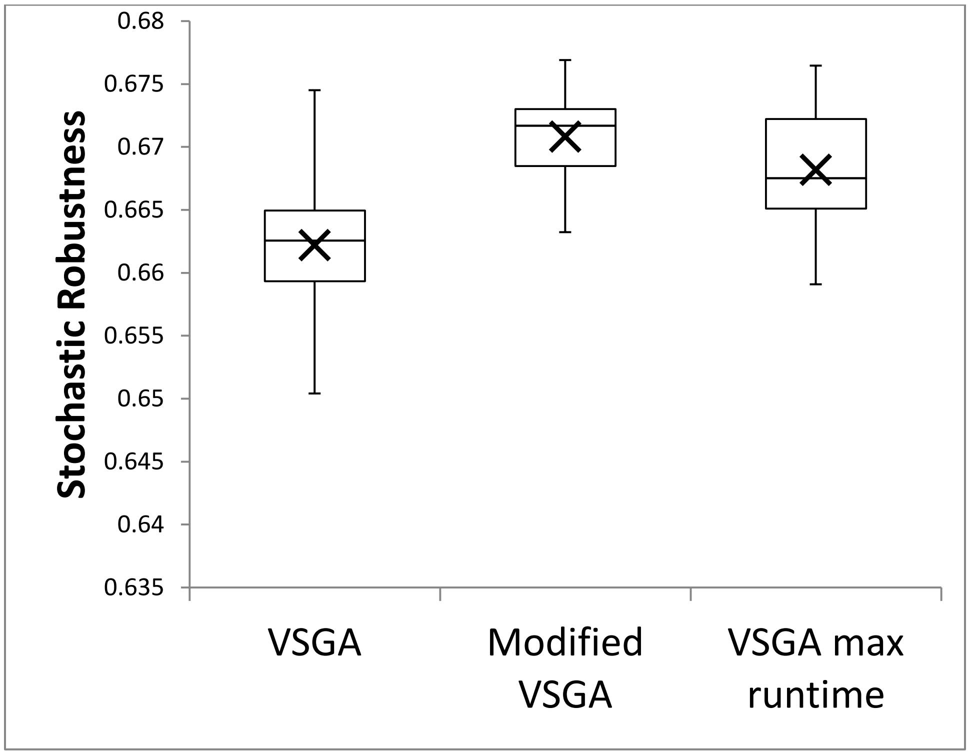


**Figure 46.** VSGA heuristic compared to modified VSGA and VSGA with maximum runtime where maximum runtime is equivalent to the sum of the runtimes for each combination in the modified VSGA.