Real-time Allocation of Firing Units To Hostile Targets

FREDRIK JOHANSSON
GÖRAN FALKMAN

The protection of defended assets such as military bases and population centers against hostile targets (e.g., aircrafts, missiles, and rockets) is a highly relevant problem in the military conflicts of today and tomorrow. In order to neutralize threats of this kind, they have to be detected and engaged before causing any damage to the defended assets. We review algorithms for solving the resource allocation problem in real-time, and empirically investigate their performance using the open source testbed SWARD. The results show that many of the tested algorithms produce high quality solutions for small-scale problems. A novel variant of particle swarm optimization seeded with an enhanced greedy algorithm is described and is shown to perform best for large instances of the real-time allocation problem.

Manuscript received October 14, 2010; revised February 21, July 12, and September 9, 2011; released for publication September 29, 2011.

Refereeing of this contribution was handled by Huimin Chen.

Authors' addresses: F. Johansson, Division of Information Systems, Swedish Defence Research Agency, SE-164 90, Stockholm, Sweden, E-mail: (fredrik.johansson@foi.se); G. Falkman, Informatics Research Centre, University of Skövde, PO Box 408, SE-541 28 Skövde, Sweden, E-mail: (goran.falkman@his.se).

1557-6418/11/$17.00 © 2011 JAIF

1. INTRODUCTION

A severe threat encountered in many international peacekeeping and peace forcing operations is that of rockets, artillery, and mortars (RAM) fired by insurgents towards military bases, troops, and other assets. Attacks like these have cost many human lives in places like Iraq and Afghanistan during recent years. Similar attacks are faced by civilians in some parts of Israel on a regular basis, where so called Katyusha and Qassam rockets are fired against Israeli population centers such as Sderot and Ashkelon. Asymmetrical threats like these have caused an increased interest in systems for detecting and tracking incoming RAM before they hit their intended targets. The detection and tracking of RAM makes it possible to estimate the point of impact, so that any troops or civilians in the impact area can be alerted. However, such a warning is not always enough, due to very quick course of events, and that buildings and infrastructure will be destroyed no matter how early warnings come, given that active countermeasures are not taken. Hence, one would like to destroy incoming RAM before they hit their intended targets (and before they risk causing collateral damage upon destruction). Systems for detecting, tracking, and engaging RAM are often referred to as Counter Rocket, Artillery, and Mortar (C-RAM) systems. An example of such a system is the recently deployed Israeli Iron Dome system. Another kind of air defense situation is that in which we would like to protect defended assets against maneuvering targets such as fighter aircrafts, attack helicopters, and non-ballistic missiles. For such kind of threats, we can in general not easily predict which defended asset (if any) is the intended target of an attack, making it necessary to estimate the level of threat posed by detected targets to the defended assets in a so-called threat evaluation process.

When faced with many simultaneous threats, it is unlikely that the defenders can take action against all incoming threats, since there often are fewer firing units available than there are threats. Even when this is not the case, a problem is to know which firing unit to use against which threat in order to maximize the survivability of the defended assets or minimize the total expected target value of surviving hostile targets. This can be described as a resource allocation problem, known as the weapon allocation problem [7] within the field of operations research. Unfortunately, the allocation of defensive firing units to targets has been shown to be **NP**-complete [23].

The time available for weapon allocation depends on many factors such as the type of RAM used, the range from which it is fired, type of detection radar and type of defensive weapons (rapid-fire guns, lasers, radar-guided missiles, etc.). However, taking into account that the incoming threats often have high speed and are fired from a range of only a few kilometers, very short time is available for detection, weapon allocation, and

interception. Hence, empirical results for how weapon allocation algorithms perform on problem instances of various size are needed.

We have reviewed the available literature in order to identify suitable weapon allocation algorithms, and we have implemented and systematically evaluated the real-time performance of a selection of the identified algorithms on static asset-based weapon allocation problems. The results show that especially particle swarm optimization algorithms produce high quality solutions for small-scale problems. In this article, we describe a novel variant of particle swarm optimization seeded with an enhanced greedy algorithm and show that the seeded version performs very well relative to previously tested algorithms also for large-scale instances of the real-time allocation problem.

The rest of this article is structured as follows. In Section 2, we present the static asset-based weapon allocation problem, which is a suitable optimization model when the impact area of a threat can be assumed to be known. We also present its target-based counterpart which is more suitable for air defense situations involving maneuvering targets. In Section 3, we present a literature survey of algorithms that have been suggested for static weapon allocation (both target-based and static-based). Based on this survey, we have implemented algorithms for static asset-based weapon allocation which are presented in Section 4. Experiments in which we compare the real-time performance of the implemented algorithms are presented in Section 5, and we conclude the article in Section 6.

2. WEAPON ALLOCATION

Informally, weapon allocation (often also referred to as weapon assignment or weapon-target allocation) can be defined as the reactive assignment of defensive weapon resources (firing units) to engage or counter identified threats (e.g., aircrafts, air-to-surface missiles, and rockets) [29]. More formally, the weapon allocation problem can be stated as a non-linear optimization problem in which we aim to allocate firing units so as to minimize the expected total value of the targets, or, alternatively, to maximize the expected survivability of the defended assets. These alternative views are referred to as target-based (weighted subtractive) defense and asset-based (preferential) defense, respectively. The asset-based formulation demands knowledge of which targets that are headed for which defended assets and thereby assumes a high level of situation awareness [24]. Therefore, the static asset-based weapon allocation problem formulation is suitable for problems involving defense against ballistic weapons, while the target-based formulation is more appropriate when the intended aims of the targets are not known [28]. In Section 2.1 we describe the static asset-based weapon allocation problem, and in Section 2.2 we give a similar description of the static target-based weapon allocation problem.

2.1. The Static Asset-Based Weapon Allocation Problem

When presenting the static asset-based weapon allocation problem, the following notation will be used:

- $|\mathbf{A}| \triangleq$ number of defended assets.
- $|\mathbf{W}| \triangleq$ number of firing units.
- $|\mathbf{T}| \triangleq$ number of targets.
- $\omega_j \triangleq$ protection value of defended asset A_j.
- $P_{ik} \triangleq$ probability that firing unit W_k destroys target T_i if assigned to it.
- $\pi_i \triangleq$ probability that target T_i destroys the asset it is aimed for.
- $\mathbf{G}_j \triangleq$ the set of targets aimed for defended asset A_j.
- $x_{ik} = \begin{cases} 1 & \text{if firing unit } W_k \text{ is assigned to target } T_i, \\ 0 & \text{otherwise.} \end{cases}$

In the static asset-based weapon allocation problem, each offensive target is assumed to be aimed at a defended asset, where each defended asset is associated with a protection value ω_j. Each target has an associated lethality probability π_i, indicating the probability that T_i destroys the defended asset it is aimed for, given that it is not successfully engaged. This probability depends on the accuracy of the targets as well as the nature of the defended assets [8]. As can be seen, we are assuming that such probabilities are target dependent only, i.e., we do not take the type of the defended asset into consideration. The defenders are equipped with firing units, where each pair of firing unit and target is assigned a kill probability P_{ik}. Now, the objective of the defense is to allocate the available firing units so as to maximize the total expected protection value of surviving defended assets [7]:

$$\max J = \sum_{j=1}^{|\mathbf{A}|} \omega_j \prod_{i \in \mathbf{G}_j} \left(1 - \pi_i \prod_{k=1}^{|\mathbf{W}|} (1 - P_{ik})^{x_{ik}} \right) \tag{1}$$

subject to:

$$\begin{aligned} \sum_{i=1}^{|\mathbf{T}|} x_{ik} &= 1, \qquad \forall k \\ x_{ik} &\in \{0,1\}, \qquad \forall i \forall k. \end{aligned} \tag{2}$$

In (1), the inner product $\prod_{k=1}^{|\mathbf{W}|}(1 - P_{ik})^{x_{ik}}$ should be interpreted as the probability that target T_i survives the countermeasures taken against it. Hence, the product $\prod_{i \in \mathbf{G}_j}(1 - \pi_i \prod_{k=1}^{|\mathbf{W}|}(1 - P_{ik})^{x_{ik}})$ is the probability that the defended asset A_j survives the attack of all targets aimed for it.

A solution to a static weapon allocation problem can be represented as a matrix of decision variables

$$\mathbf{x} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1|\mathbf{W}|} \\ x_{21} & x_{22} & \cdots & x_{2|\mathbf{W}|} \\ \vdots & \vdots & x_{ik} & \vdots \\ x_{|\mathbf{T}|1} & x_{|\mathbf{T}|2} & \cdots & x_{|\mathbf{T}||\mathbf{W}|} \end{bmatrix}. \quad (3)$$

Such a solution is feasible if it fulfills the constraints given in (2), i.e., that the entries of each column in (3) sum to one. For a problem instance consisting of $|\mathbf{T}|$ targets and $|\mathbf{W}|$ firing units, there are $|\mathbf{T}|^{|\mathbf{W}|}$ feasible solutions.

From the solution of the static asset-based weapon allocation problem, we can discover which of the defended assets that should be protected, and in which way each of the defended assets should be protected (preferential defense).

2.2. The Static Target-Based Weapon Allocation Problem

Using the same notation as in Section 2.1, but with the additional definition:

- $V_i \triangleq$ target value of target T_i,

we can define the static target-based weapon allocation problem as:

$$\min F = \sum_{i=1}^{|\mathbf{T}|} V_i \prod_{k=1}^{|\mathbf{W}|} (1 - P_{ik})^{x_{ik}} \quad (4)$$

subject to the constraints given in (2). Since the product as before is the probability that target T_i survives the countermeasures taken against it, the objective function should be interpreted as the minimization of the total expected target value of surviving targets.

The estimation of target values is far from trivial, and can be seen as a very important high-level information fusion problem. A survey of how threat values V_{ij}s can be estimated (representing the threat posed by target T_i to defended asset A_j) is presented in [11, 12]. Once such threat values have been calculated, these can be aggregated into target values using weighted averages such as:

$$V_i = \frac{\sum_{j=1}^{|\mathbf{A}|} V_{ij}\omega_j}{\sum_{j=1}^{|\mathbf{A}|} \omega_j}. \quad (5)$$

Nevertheless, this is only one choice of how to aggregate threat values into target values. Furthermore, the original target values rely on coarse models of what is threatening behavior or not (typically parameters such as distance between the target and the defended asset, the speed and heading of the target, target type, etc). To complicate matters, expert air defense operators frequently disagree about the threat of individual aircraft [31]. Consequently, it should be remembered that target values will always be associated with uncertainty, and that they to a large degree are subjective.

2.3. Properties of Weapon Allocation Problems

A few assumptions are made in the static weapon allocation formulations. Firstly, all firing units have to be assigned to targets, as indicated in the constraint given in (2). Moreover, all the firing units have to be assigned simultaneously, i.e., we can not observe the outcome of some of the engagements before a remaining subset of firing units are allocated. This is what is meant by *static* weapon allocation, as opposed to *dynamic* weapon allocation. The static formulation makes sense for the problem domain studied here, since the high speed of short-range RAM does not allow for several engagement cycles. We also assume that an engagement will not affect other engagements (e.g., that a firing unit can destroy another target than it is allocated to, or that targets can destroy other assets than they are aimed for). Without the last assumption, the geometry of the problem must be taken into account, creating an extremely complex problem. We also ignore the risk of collateral damage to the protected area when intercepting the targets.

Despite the assumptions, there is a combination of factors that make the static weapon allocation problems hard to solve. Firstly, the objective functions given in (1) and (4) are *non-linear*, so that well-known linear programming techniques such as the simplex algorithm can not be used to solve the problems. Secondly, the problems are *discrete*, since they only allow for integer valued feasible solutions due to the second constraint in (2) (i.e., fractional allocations are not possible). In general, this kind of integer programming problems are hard to solve. Thirdly, the problems are *stochastic*, due to kill probabilities (and lethality probabilities) not equal to zero or one. This non-determinism further complicates the problems. Fourthly, it is not unusual with *large-scale* problem instances, i.e., problems consisting of a large number of firing units, defended assets, and/or targets. The asset-based formulation can be shown to be a generalization of the static target-based weapon allocation formulation [7] presented in Section 2.2. The **NP**-completeness of the static target-based weapon allocation problem was established in [23], and hence, we can conclude that the static asset-based weapon allocation problem is **NP**-complete as well [7]. These properties taken together show that finding good solutions in real-time to static weapon allocation problems is indeed a very hard problem, and according to [7], rule out any hope of obtaining efficient optimal algorithms.

3. A SURVEY OF ALGORITHMS FOR WEAPON ALLOCATION

Initial research on the static target-based weapon allocation problem dates back as far as the end of the 1950s (cf. [5, 25]). Much of the initial research on the

problem seems to have been motivated by the threat from intercontinental ballistic missiles during the Cold War era [24]. Despite the end of the Cold War, research on defensive weapon allocation still remains a very active area [24]. The static target-based weapon allocation problem has been quite well studied, especially within the field of operations research. Despite the extensive research, static weapon allocation is an example of a classical operations research problem that still remains unsolved [1], in the sense that effective methods for real-time allocation are lacking. Moreover, the asset-based version of the problem is much less studied than its target-based counterpart.

Much of the original work on weapon allocation focused on the allocation of missiles to defended assets, rather than the other way around. Hence, the problems were often modeled from an attacker's side, instead of from the defending side. A brief summary and review of unclassified literature from the first years of research on the problem is given in [26]. Some years later, a monograph describing many of the developed mathematical models for weapon allocation problems was published in [6]. Unlike Matlin's review, the monograph by Eckler and Burr takes on the weapon allocation problem from a defender's view. The authors present a number of useful techniques for weapon allocation, such as relaxing the integer constraint and then make use of linear programming to solve the resulting continuous problem. This is a technique that is still in use (cf. [17]). It should be noted however, that fractional assignments of firing units to targets does not make sense, and rounding off the optimal solution to the relaxation of an nonlinear integer programming problem can yield solutions that are infeasible or far from the optimal solution to the original nonlinear problem [36]. Other kinds of tools such as the use of Lagrange multipliers and dynamic programming are also described in [6]. As the authors make clear, their focus is on analytical approaches, since it is argued that what they refer to as computer-oriented solutions give less insight into the weapon allocation problem than analytical approaches. A somewhat more recent survey of work within weapon allocation is presented in [3]. As in the earlier mentioned surveys, its focus is on analytical approaches to weapon allocation. However, it is mentioned that a shift towards various techniques such as implicit enumeration algorithms and nonlinear programming algorithms had been started at that time, since mathematical formulations of the weapon allocation problem are not generally amenable to solution in closed form [3, p. 66]. In later years, advanced computer-based techniques have been developed which are better suited for real-time weapon allocation [9]. In the following, we will focus on modern heuristic/approximate approaches, but will first present enumerative techniques, since such approaches can be very useful for special cases of static weapon allocation problems.

3.1. Exact Approaches

For small values of $|\mathbf{T}|$ and $|\mathbf{W}|$, the optimal solution to a static weapon allocation problem can easily be found by exhaustive search (also referred to as explicit enumeration), i.e., a brute-force enumeration where all feasible solutions are tested one after the other. However, as a static weapon allocation problem consists of $|\mathbf{T}|^{|\mathbf{W}|}$ feasible solutions, this is not a viable approach for air defense scenarios involving a large number of targets and firing units.

Exact polynomial time algorithms have been identified for the special case of the static target-based weapon allocation problem in which the kill probabilities of all firing units are assumed to be identical, i.e., $P_{ik} = P_i$. For this special case, the well known maximum marginal return (MMR) algorithm suggested in [5], and the local search algorithm suggested in [7] can be proven to be optimal. Some other special cases of the static target-based weapon allocation problem can be formulated as network flow optimization problems. If we assume the constraint that all firing units have kill probabilities $P_{ik} \in \{0, P_i\}$, i.e., that firing units either can or cannot reach a target, and in the former case, the kill probability only depend upon the target, the problem can be transformed into a minimum cost network flow problem with linear arc costs, for which several efficient algorithms exist [7]. A similar transformation can be done for the special case of the static target-based weapon allocation problem where we assume that $|\mathbf{T}| \leq |\mathbf{W}|$, and that at most one firing unit is to be allocated to each target. In this case, we can convert the problem into a so called transportation problem, for which efficient algorithms exist [7]. However, the general static target-based weapon allocation problem has been proved to be **NP**-complete [23], as have been discussed earlier. This also holds true for the asset-based version of the static weapon allocation problem, since this can be seen as a generalization of the static target-based version.

Another exact approach is to use branch-and-bound algorithms for finding the optimal solution. Branch-and-bound algorithms use tree representations of the solution space and are often able to prune away large subsets of feasible solutions through calculation of lower and upper bounds on different branches of the tree. In a recent article by [1], three branch-and-bound algorithms (using different lower-bound schemes) are investigated and are shown to give short computation times on average. The results are impressive, however, in theory the risk exists that the algorithm will require branching the full tree for some problem instances. This means that in worst-case, the performance of the branch-and-bound algorithm can be at least as bad as the performance of more naïve exhaustive search algorithms. Although it in practice is unlikely that this worst-case scenario will appear, it is unfortunately not possible to in advance compute an upper bound on the computational time it will

take to find the optimal solution to a problem instance when using a branch-and-bound algorithm. Hence, as can be seen in the results reported in [1], some problem instances of large size can be solved very quickly, while considerably smaller problem sizes can demand considerably more time for the optimal solution to be found. In other words, we have to rely on heuristic algorithms for large-scale problems when real-time guarantees are needed [1, 7].

3.2. Heuristic Approaches

A well-known heuristic approach for static target-based weapon allocation is the greedy maximum marginal return algorithm, originally suggested in [5]. A similar greedy algorithm is presented in [18]. Basically, the maximum marginal return algorithm works sequentially by greedily allocating firing units to the target maximizing the reduction of the expected value. It starts with allocating the first firing unit to the target for which the reduction in value is maximal, whereupon the value of the target is reduced to the new expected value. Once the first firing unit is allocated, the same procedure is repeated for the second firing unit, and so on, until all firing units have been allocated to targets. Pseudo code for the maximum marginal return algorithm is shown in Section 4.1. Obviously, the maximum marginal return algorithm is very simple and fast. This is a general advantage of greedy algorithms, but due to their greedy nature they are also very likely to end up with suboptimal solutions. Since the algorithm uses target values for choosing which target to be allocated next, it cannot be used as is for static asset-based weapon allocation. However, in [27] a number of greedy algorithms for asset-based weapon allocation are described. These algorithms basically work by approximating the asset-based problem with its target-based counterpart, by using the protection value of the defended asset to which the target is aimed for as the target value. When the problem has been approximated by a target-based problem, it is suggested that the maximum marginal return algorithm returns a solution that can be used as an approximative solution to the asset-based problem. Another suggested approach in [27] is to use the solution returned from the maximum marginal return algorithm and to apply local search on the solution so that the target allocated by one weapon can be swapped to the target allocated by another weapon, and vice versa.

Another kind of heuristic approach to a constrained version of the target-based weapon allocation problem has been suggested in [35], in which artificial neural networks are used. It is stated that solutions close to global optima are found by the algorithm, but results are only presented for a few small-scale problem instances, from which it in the authors' view is not possible to generalize. It is in [9] also argued that artificial neural network algorithms for weapon allocation sometimes are unsteady and non-convergent, leading to that obtained solutions may be both suboptimal and infeasible.

As an alternative, the use of genetic algorithms seems to be popular. Such an algorithm for static target-based weapon allocation is described in [16], while a genetic algorithm combined with local search is presented in [22] and [21]. The quality of the solutions returned by the greedy maximum marginal return algorithm presented in [18] is in [16] compared to the solutions returned by genetic algorithms. However, the algorithms are only evaluated on target-based weapon allocation problems. The standard genetic algorithm is outperformed on large-scale problem sizes, but only one problem instance is tested for each problem size, so the possibility to generalize the results can be questioned. Even though, the results seem to indicate that greedy search works better than standard genetic algorithms on large target-based problem instances. It is in [16] suggested that genetic algorithms can be seeded with the solution returned from a greedy algorithm, which seems to be a suitable approach to improve the quality of genetic algorithms on large problem sizes. In [2], a genetic algorithm combined with local search is suggested for a dynamic version of the asset-based weapon allocation problem. It is shown that local search improves the results, but that the computational time needed is increased. The effects of real-time requirements on the algorithms are not tested.

The use of ant colony optimization for target-based weapon allocation is suggested in [19, 20]. Reported results in [19] and [20] indicate that ant colony optimization algorithms perform better than standard genetic algorithms on large-scale problems, and that the algorithms can be improved upon by using local search. However, the algorithms were allowed to run for two hours, so it is unclear how this generalizes to settings with real-time requirements.

A simulated annealing algorithm for static asset-based weapon allocation is presented in [4]. Basically, simulated annealing is based on an analogy of thermodynamics with the way metals cool and anneal, in which a liquid that is cooled slowly is likely to form a pure crystal corresponding to a state of minimum energy for the metal, while a quick cooling phase is likely to result in states of higher energy levels [32]. By controlling an artificial "temperature" when making the optimization (corresponding to the minimization of energy levels), it becomes possible to escape from local minima in the hunt for the optimal solution (the purest crystal in the thermodynamics analogy). However, no evaluation of the quality of the solutions obtained by the algorithm is presented in [4], so it is unknown how good their implemented algorithm performs. Another implementation of a simulated annealing algorithm provides solutions of lower quality than ant colony optimization and genetic algorithms in a static target-based weapon allocation experiment described in [19]. The algorithms were, as describe above, allowed to run for two hours,

TABLE I
Algorithmic Approaches to Weapon Allocation

Algorithmic Approach	*References*
Branch-and-bound	[1]
Genetic algorithms	[16, 21]
Ant colony optimization	[19, 20]
Greedy algorithms	[5, 18]
VLSN	[1]
Neural networks	[35]
Particle swarm optimization	[33, 37]

so it is not known how the algorithms perform under more realistic time constraints.

In [1], good performance results for an approach using a minimum cost flow formulation heuristic for generating a good starting feasible solution are presented. This feasible solution is then improved by a very large-scale neighborhood (VLSN) search algorithm that treats the problem as a partitioning problem, in which each partition contains the set of firing units assigned to target T_i. The very-large scale neighborhood search improves the original feasible solution by a sequence of cyclic multi-exchanges and multi-exchange paths among the partitions. As the name suggests, the size of the used neighborhoods are very large. To search such large neighborhoods typically takes considerably amounts of computations and demands implicit enumeration methods [10]. By using the concept of an improvement graph, it becomes possible to evaluate neighbors faster than other existing methods [1, 10].

Recently, the use of particle swarm optimization for static target-based weapon allocation has been suggested. In [33], a particle swarm optimization algorithm is implemented and compared to a genetic algorithm. The results indicate that the particle swarm optimization algorithm generates better solutions than the genetic algorithm, but the algorithms are only tested on a single problem instance consisting of five targets and ten firing units. For this reason, it is not possible to generalize the obtained results. Experiments presented in [37] also indicate that particle swarm optimization algorithms create better solutions than genetic algorithms for static target-based weapon allocation.

As evident from the literature survey presented above, a lot of different algorithmic approaches have been suggested for the static weapon allocation problem. A summary of some of the approaches presented above is presented in Table I.

4. THE IMPLEMENTED ALGORITHMS

Based on the results from the literature survey presented in Section 3, a number of heuristic algorithms have been implemented. Since the target-based weapon allocation seems more well-researched than the asset-based problem, the focus of the rest of this article will be on the latter.

The algorithms for static asset-based weapon allocation evaluated in this article share the same kind of representation, in which a solution is represented as a vector of length $|\mathbf{W}|$. Each element k in the vector points out the target T_i to which the weapon is allocated. As an example of this, the vector $[2,3,2,1]$ represents a solution in which W_1 and W_3 are allocated to T_2, W_2 is allocated to T_3, and W_4 is allocated to T_1.

4.1. A Maximum Marginal Return Algorithm for Static Weapon Allocation

A greedy algorithm for static target-based weapon allocation, known as the maximum marginal return (MMR) algorithm, was initially suggested in [5]. This algorithm (described with pseudo code in Algorithm 1) is very simple since it as already explained works greedily by assigning weapons sequentially to the target that maximizes the reduction of the expected target value. When the first weapon has been allocated to the target for which the reduction in value is maximal, the target value is reduced to the new expected value. After that, the same procedure is repeated for the second weapon, and so on, until all weapons have been allocated to targets, yielding a computational complexity of $O(|\mathbf{W}| \times |\mathbf{T}|)$.

ALGORITHM 1 *Maximum marginal return algorithm*
for all k *such that* $1 \leq k \leq |\mathbf{W}|$ **do**
  $highestValue \leftarrow -\infty$
  $allocatedTarget \leftarrow 0$
  for all i *such that* $1 \leq i \leq |\mathbf{T}|$ **do**
    $value \leftarrow V_i \times P_{ik}$
    if $value > highestValue$ **then**
      $highestValue \leftarrow value$
      $allocatedTarget \leftarrow i$
  assign W_k *to target* $T_{allocatedTarget}$
  $V_{allocatedTarget} \leftarrow V_{allocatedTarget} - highestValue$
return allocation

It is not obvious how to use the MMR algorithm for the static asset-based weapon allocation problem, since it in this version of the problem does not exist any target values. Instead, there are protection values associated with the defended assets, and lethality values associated with the targets. In [27], it is suggested that a defended asset's weight (protection value) is equally distributed over the targets aimed for it, so that a target's value is computed as $V_i = \omega_j / |\mathbf{G}_j|$ (where j is the index for the set $\mathbf{G}_j$ of which target T_i is a member), and that the asset-based problem is approximated with its target-based counterpart. Similar reasoning is presented in [7] where it is suggested that the value of a target is set to the expected destroyed protection value of the defended asset to which it is aimed, given that the target is not engaged and that all other targets aimed for the defended asset are destroyed.

We have here chosen to calculate the target value V_i for a target T_i as:

$$V_i = \omega_j \times \pi_i \tag{6}$$

where j is the index of the defended asset to which target T_i is aimed. Hence, the target value has been calculated as the product of the lethality probability π_i of the target and the protection value ω_j of the defended asset it is aimed at. In this way, we follow the approach suggested in [7] to use the protection value of the defended asset to impact on the target value, but we complement this with taking the lethality of the target into account, since this extra information otherwise is lost.

We have also included a variant of greedy search where we have taken the solution generated by the MMR algorithm and improved it with a simple local search (LS) that creates neighbor solutions by swapping two positions selected at random in the solution vector (this variant of the MMR algorithm, described with pseudo code in Algorithm 2, will in the following be referred to as MMR-LS). This algorithm is an implementation of the idea briefly discussed in [27].

ALGORITHM 2 *The MMR-LS algorithm*

```
bestSolution ← MMR()
J_best ← CalculateFitness(bestSolution)
while termination criteria not met do
  neighborSolution ← neighbor(bestSolution)
  J_new ← CalculateFitness(neighborSolution)
  if J_new > J_best then
    bestSolution ← neighborSolution
    J_best ← J_new
return bestSolution
```

Obviously, the quality of the solutions generated by the MMR-LS algorithm will always be at least as good as the quality of the solutions returned by the MMR algorithm.

4.2. An Enhanced Maximum Marginal Return Algorithm for Static Weapon Allocation

What here will be referred to as the enhanced maximum marginal return algorithm (the authors' terminology) is quite similar to the standard maximum marginal return algorithm. The difference is that in the enhanced maximum marginal return (EMMR) algorithm it is not predetermined which firing unit to allocate next. Instead, the choice of which firing unit to allocate next is based on which weapon-target pair that maximizes the marginal return. We have implemented this algorithm based on the description in [16], and the pseudo code for the algorithm is given in Algorithm 3. In the first iteration $it = 1$, $|\mathbf{W}| \times |\mathbf{T}|$ combinations are tested. The weapon-target pair with highest marginal return is selected, so that the firing unit is selected to the target, and the target value of the corresponding target is updated accordingly. After this, $|\mathbf{W}| - 1$ firing units are unallocated. In next iteration, the remaining $(|\mathbf{W}| - 1) \times |\mathbf{T}|$ weapon-target pairs are tested, and so on, until there does not remain any unallocated firing units. Hence, the time complexity of EMMR becomes $O(|\mathbf{W}|^2|\mathbf{T}|)$.

ALGORITHM 3 *Enhanced maximum marginal return algorithm (adapted from [16])*

```
for all it such that 1 ≤ it ≤ |W| do
  highestValue ← −∞
  allocatedTarget ← 0
  allocatedWeapon ← 0
  for all k such that 1 ≤ k ≤ |W| do
    for all i such that 1 ≤ i ≤ |T| do
      value ← V_i × P_ik
      if value > highestValue then
        highestValue ← value
        allocatedWeapon ← k
        allocatedTarget ← i
  assign W_allocatedWeapon to T_allocatedTarget
  V_allocatedTarget ← V_allocatedTarget − highestValue
return allocation
```

As the standard MMR algorithm, EMMR is relying on target values. Hence, we calculate target values according to (6), solve the approximated target-based problem using EMMR, and return the solution as the solution to the asset-based problem.

4.3. A Genetic Algorithm for Static Weapon Allocation

In [13], we presented a genetic algorithm (GA) designed for real-time allocation of defensive weapon resources to targets. The original version of the algorithm was intended for the static target-based problem, but we have now with some modifications adapted it to also suit the static asset-based formulation of the problem.

The algorithm is described in pseudo code in Algorithm 4. First, an initial population consisting of *nrOfIndividuals* is created, through generation of a vector of length $|\mathbf{W}|$. In this vector each element W_k is assigned a random integer value in the interval $\{1,\ldots,|\mathbf{T}|\}$. In each generation we evaluate all individuals in the population and determine their objective function values in accordance with (1). Hence, each individual is assigned a fitness value that is used in the following phases of selection and recombination. After the evaluation phase, deterministic tournament selection is used as selection mechanism to determine which individuals in population *Pop* that should be used as parents for *Pop*′, i.e., we pick two individuals at random from *Pop* and select the one with best fitness value. When two parents have been selected from *Pop*, we apply one-point crossover at a randomly selected position $k \in \{1,\ldots,|\mathbf{W}|\}$, generating two individuals that become members of *Pop*′. This is repeated until there are *nrOfIndividuals* in *Pop*′. Thereafter, we apply mutation on a randomly selected position $k \in \{1,\ldots,|\mathbf{W}|\}$ in the first individual of *Pop*′, where the old value is changed into $i \in \{1,\ldots,|\mathbf{T}|\}$. Hence, there is a probability of $1/|\mathbf{T}|$ that the individual is unaffected of the mutation. The mutation operator is repeated on all individuals in *Pop*′ and the resulting individuals become members of the new population *Pop*. This loop is repeated until the termination criterion is fulfilled (the upper limit on

the computational time bound is reached). At this point, the individual with the best fitness found during all generations is returned as the allocation recommended by the algorithm.

ALGORITHM 4 *Pseudo code for our genetic algorithm*

```
fitness_best ← −∞
Pop ← GenerateInitialPopulation()
while termination criteria not met do
  for l ← 1 to nrOfIndividuals do
    J_l ← CalculateFitness(Pop(l))
    if J_l > fitness_best then
      g⃗ ← Pop(l)
      fitness_best ← J_l
  Pop′ ← Crossover(Pop)
  Pop ← Mutate(Pop′)
return g⃗
```

Furthermore, we have implemented a variant of the genetic algorithm that is seeded with well-performing individuals. Instead of creating all individuals in the initial population at random, κ individuals are created based on the solution returned by the EMMR algorithm (a random swap between the targets of two of the firing units is first made for each of the seeded individuals in order to create some diversity among them). The remaining individuals are created randomly just as before. This seeded version of the genetic algorithm will in the following be referred to as GA-S.

4.4. A Particle Swarm Optimization Algorithm for Static Weapon Allocation

In [14], we developed a particle swarm optimization (PSO) algorithm for the static target-based weapon allocation problem. We have modified this algorithm to also suit the static asset-based weapon allocation problem.

A particle swarm consists of *nrOfParticles* particles, in which each particle is associated with a position $\vec{x}_l^t$, a velocity $\vec{v}_l^t$, and a memory $\vec{b}_l^t$ storing the particle's personal best position. Moreover, we also store the swarm's global best position in a vector $\vec{g}^{\,t}$. Each particle corresponds to a solution, given by the particle's position.

ALGORITHM 5 *Pseudo code for our particle swarm optimization algorithm*

```
Initialization()
while termination criteria not met do
  for l ← 1 to nrOfParticles do
    J_l ← CalculateFitness(x⃗_l)
    if J_l = CalculateFitness(g⃗) then
      p_l ← Reinitialize()
    else
      if J_l > CalculateFitness(b⃗_l) then
        b⃗_l ← x⃗_l
        if J_l > CalculateFitness(g⃗) then
          g⃗ ← x⃗_l
  for l ← 1 to nrOfParticles do
    v⃗_l ← UpdateVelocity(p⃗_l)
    x⃗_l ← UpdatePosition(p⃗_l)
return g⃗
```

The algorithm is described in pseudo code in Algorithm 5. In an initialization phase, each particle is assigned an initial position $\vec{x}_l^0$ (where the elements in the initial position vectors are integers randomly distributed between 1 and $|\mathbf{T}|$), and an initial velocity $\vec{v}_l^0$ (a vector of real numbers randomly distributed from the uniform distribution $U[-0.5|\mathbf{T}|, 0.5|\mathbf{T}|]$). A fitness value is calculated for each particle, given by the objective function value J (see (1)) that is obtained for the solution corresponding to the particle's position. The new fitness is compared to the personal best and the global best to see whether these should be updated accordingly. After this, the velocity and position is updated for each particle, according to (7) and (8).

$$\vec{v}_l^{\,t+1} = \omega\vec{v}_l^{\,t} + c_1\vec{r}_1^{\,t} \circ (\vec{b}_l^{\,t} - \vec{x}_l^{\,t}) + c_2\vec{r}_2^{\,t} \circ (\vec{g}^{\,t} - \vec{x}_l^{\,t}) \quad (7)$$

$$\vec{x}_l^{\,t+1} = \vec{x}_l^{\,t} + \vec{v}_l^{\,t+1}. \quad (8)$$

In (7), ω is a parameter referred to as inertia or momentum weight, specifying the importance of the previous velocity vector, while c_1 and c_2 are positive constants specifying how much a particle should be affected by the personal best and global best positions (referred to as the cognitive and social components, respectively). $\vec{r}_1^{\,t}$ and $\vec{r}_2^{\,t}$ are vectors with random numbers drawn uniformly from the interval $[0, 1]$. Moreover, the $\circ$-operator denotes the Hadamard product, i.e., element-by-element multiplication of the vectors. In order to avoid that particles gain too much momentum, a $V_{\max}$ parameter that constrains the velocities to stay in the interval $[-V_{\max}, V_{\max}]$ has been introduced.

After the position update specified in (8), we round off the particles' positions to their closest integer counterpart. In next iteration we calculate the particles' new fitness values, whereupon the velocities and positions are updated, and so on. This is repeated until a termination criterion is met, i.e., that no more time remains. When this happens, the best solution obtained so far is returned as output from the algorithm.

A problem that must be handled is particles moving outside the bounds of the search space. When this happens, we reinitialize the position and velocity values of the coordinate for which the problem occurred. Moreover, in order to avoid premature convergence to local optima (stagnation), we reinitialize the velocity vector for particles rediscovering the current best solution. For a more thorough explanation of the problem of stagnation in particle swarm optimization, see [34].

In addition to the described particle swarm optimization algorithm, we have also included a variant in which we seed the starting position for κ particles in the initial population in the same way as with the GA-S algorithm (while their initial velocities are randomized in the same manner as for the remaining particles). This seeded particle swarm optimization algorithm will in the following be referred to as PSO-S. To the best of our knowledge, the use of seeded particles is novel for the weapon allocation problem.

5. EXPERIMENTS

In the experiments reported here, we have used the open source testbed SWARD[1] (System for Weapon Allocation Research and Development) which we have developed in order to allow for systematic comparison of various weapon allocation algorithms [11, 15]. The testbed is implemented in Java, and we have been running the experiments on a computer with a 2.67 GHz Intel Core i7 CPU and 8 GB RAM. By using SWARD, we make sure that the experiments presented here are easily reproducible, so that researchers can test other algorithms on the same problem instances.

In order to recreate the problem instances used in the experiment presented in Section 5.1.1, the following settings should be used in SWARD:

- $T_{start} = 5$, $W_{start} = 5$,
- $T_{end} = 9$, $W_{end} = 9$,
- $T_{step} = 1$, $W_{step} = 1$,
- $iterations = 10$, $DAs = 5$,
- $seed = 0$, $timeLimit = 1000$ ms.

Similarly, for recreating the problem instances used in the experiment presented in Section 5.1.2, the following settings should be used:

- $T_{start} = 10$, $W_{start} = 10$,
- $T_{end} = 30$, $W_{end} = 30$,
- $T_{step} = 10$, $W_{step} = 10$,
- $iterations = 100$, $DAs = 5$,
- $seed = 0$, $timeLimit = 1000$ ms.

The time limits make sure that no algorithms are allowed to run for more than a total time of one second (including seeding). For the genetic algorithms we have used the parameter setting: $nrOfIndividuals = \max(|\mathbf{T}|, |\mathbf{W}|)$. Additionally, we have for the seeded version used $\kappa = 0.5 \times nrOfIndividuals$. For the particle swarm optimization algorithm we have used $nrOfParticles = 50$, $c_1 = 2.0$, $c_2 = 2.0$, $\omega = 0.8$, and $V_{\max} = 0.5 \times |\mathbf{T}|$. The same settings have been used for the seeded version, with the additional parameter setting $\kappa = 25$.

5.1. Heuristic Algorithm Performance

For scenarios that demand solving the static asset-based weapon allocation problem faster than is possible with optimal algorithms, we have to rely on heuristic algorithms. In Section 5.1.1, we present experimental results obtained with the suggested heuristic algorithms on small-scale problem instances, while we in Section 5.1.2 present results on large-scale problem instances.

5.1.1. A comparison against the optimal solution for small-scale problems

We have in order to investigate the quality of the solutions generated by the suggested algorithms compared their obtained objective function values to the optimal objective function values obtained by exhaustive search for relatively small-scale scenarios between ($|\mathbf{T}| = 5, |\mathbf{W}| = 5$) and ($|\mathbf{T}| = 9, |\mathbf{W}| = 9$).

TABLE II
Deviation from Optimal Solution (in %)
Averaged Over Ten Problem Instances

	5×5	6×6	7×7	8×8	9×9
GA	**0**	**0**	**0**	0.1	0.7
GA-S	**0**	**0**	**0**	0.2	0.5
PSO	**0**	**0**	**0**	**0**	**0.2**
PSO-S	**0**	0.1	0.1	0.2	**0.2**
MMR	2.9	3.7	4.8	6.4	6.6
EMMR	0.3	0.8	0.8	0.8	0.9
MMR-LS	0.6	1.1	0.8	1.3	1.7

The average percentage deviation from the optimal solution is a common metric to use for evaluating heuristic algorithms on small-scale optimization problems where the optimal solution can be calculated, and therefore it also has been used here. The percentage deviation Δ_{alg} for a specific algorithm on a specific problem instance has been calculated as:

$$\Delta_{alg} = \frac{|J_{alg} - J_{opt}|}{J_{opt}} \times 100 \tag{9}$$

where J_{alg} is the objective function value for the tested algorithm and J_{opt} is the optimal objective function value. In the tables, we use **bold** to show which obtained objective function value that is the best for each tested problem size.

Looking at Table II, the algorithms' percentage deviations from the optimal solution show that most of the algorithms are able to find optimal or very near-optimal solutions for the smallest tested problem sizes. The MMR algorithm is by far the worst of the algorithms on the tested small-scale scenarios, but when allowed to improve its initial solution by local search (i.e., the MMR-LS algorithm), the quality is improved. The EMMR algorithm produces solutions that are better than both the MMR and MMR-LS algorithms. However, as can be seen, all these greedy heuristics are outperformed by the nature-inspired metaheuristics. Of the nature-inspired metaheuristics, the PSO algorithm performs somewhat better than the others. In fact, it produces optimal solutions to all problem instances of size ($|\mathbf{T}| = 5, |\mathbf{W}| = 5$)–($|\mathbf{T}| = 8, |\mathbf{W}| = 8$), and for ($|\mathbf{T}| = 9, |\mathbf{W}| = 9$) it is in one second able to generate almost optimal solutions to problems consisting of $9^9 = 387{,}420{,}489$ feasible solutions.

It should be noted that the results obtained on small-scale problems do not necessarily extends to large-scale problems. For small instances of any combinatorial problem, it is likely that algorithms such as PSO algorithms and GAs are able to search a large fraction of the solution space in a short period of time, making it more probable to find a high quality solution, while

[1]The open source testbed SWARD can be downloaded from http://sourceforge.net/projects/sward/.

TABLE III
Average Objective Function Value for $|\mathbf{T}| = 10$
Averaged Over 100 Static Asset-Based Problem Instances
(higher objective function values are better)

	10×10	10×20	10×30
GA	266.7 (42.0)	301.9 (49.1)	302.4 (46.5)
GA-S	268.7 (42.3)	304.1 (49.5)	303.1 (46.6)
PSO	269.2 (42.1)	303.2 (49.4)	302.3 (46.5)
PSO-S	**270.0** (42.2)	**304.3** (49.5)	**303.2** (46.6)
MMR	251.5 (40.3)	296.8 (48.6)	301.2 (46.3)
EMMR	268.1 (42.2)	304.1 (49.5)	303.1 (46.6)
MMR-LS	267.5 (42.7)	303.6 (49.4)	302.9 (46.6)

TABLE IV
Average Objective Function Value for $|\mathbf{T}| = 20$
Averaged Over 100 Static Asset-Based Problem Instances
(higher objective function values are better)

	20×10	20×20	20×30
GA	153.7 (30.6)	219.3 (36.4)	263.5 (34.7)
GA-S	154.7 (30.9)	237.3 (37.1)	279.8 (36.0)
PSO	158.1 (30.6)	212.6 (37.2)	245.1 (34.1)
PSO-S	**160.0** (31.1)	**238.4** (37.5)	**280.0** (36.0)
MMR	117.6 (28.6)	210.2 (36.4)	261.2 (34.0)
EMMR	127.6 (29.8)	237.0 (37.0)	279.8 (36.0)
MMR-LS	128.9 (30.2)	234.9 (37.6)	277.2 (35.4)

one wrong decision by a constructive, one-pass heuristic may result in a solution differing dramatically from the optimum of a small case [30]. Therefore, the results should not without further tests be generalized to larger problem sizes. With this said, it is still very relevant to test the performance on small-scale problem instances, not at least since it in many real-world air defense scenarios is likely that the number of targets and available firing units will be close to the settings tested here.

5.1.2. A comparison between algorithms on larger-scale problems

In a second experiment with the heuristic algorithms, we have tested them on larger-scale problems ranging in between ($|\mathbf{T}| = 10, |\mathbf{W}| = 10$) and ($|\mathbf{T}| = 30, |\mathbf{W}| = 30$). The algorithms have also in this experiment been allowed to run for one second on each problem instance. The optimal solutions are hard to obtain for large-scale problem instances, so instead of calculating the deviation from the optimal solution, we have here simply plotted the objective function values obtained (averaged over 100 problem instances) in Tables III–V. We also show the associated standard deviations within parentheses. As before, **bold** is used to indicate the best obtained objective function value on each problem size.

A note to make is that the standard deviations shown in many cases are larger than the differences in mean values among the algorithms. However, this should not be interpreted as that there are no significant differences among the algorithms. Rather, the largest part of these standard deviations are due to the differences between various problem instances. In some problem instances the optimal objective function values are lower, while they in others are higher (as a natural result of the random fashion in which the problem instances are generated). As a consequence of this, also optimal algorithms would obtain large standard deviations.

TABLE V
Average Objective Function Value for $|\mathbf{T}| = 30$
Averaged Over 100 Static Asset-Based Problem Instances
(higher objective function values are better)

	30×10	30×20	30×30
GA	96.8 (20.4)	147.9 (30.4)	186.2 (33.7)
GA-S	97.2 (20.3)	151.8 (29.1)	**208.7** (36.9)
PSO	99.8 (21.4)	128.6 (29.2)	150.7 (29.2)
PSO-S	**103.0** (19.9)	**155.5** (31.3)	208.6 (36.8)
MMR	53.6 (20.4)	117.0 (27.0)	174.8 (32.2)
EMMR	59.7 (24.2)	135.7 (30.6)	208.1 (36.5)
MMR-LS	59.8 (23.9)	137.6 (31.1)	205.7 (37.0)

When analyzing the obtained results, it can be seen that the use of local search significantly improves the quality of the solutions found using MMR also on large problem sizes. A comparison of the solutions generated by MMR-LS with the ones returned by the EMMR algorithm shows that the performance of these are approximately equally good (although EMMR is significantly faster than MMR-LS). This indicates that it in the future may be worth studying if it would be beneficial to apply simple local search also to EMMR.

It can be seen that the seeded particle swarm optimization algorithm (i.e., PSO-S) is performing best relative to the other algorithms on all tested problem sizes except the largest, on which the seeded genetic algorithm (GA-S) performs slightly better. We have in earlier work [14] shown that PSO runs into some trouble when applied to large target-based problem instances under tight real-time constraints, and this trend can be seen also for the large asset-based problem instances tested here. However, when combined with the seeding mechanism, particle swarm optimization seems to work very well. It can be seen that the obtained objective function values for the greedy algorithms MMR-LS and EMMR are reasonably close to the best algorithms' objective function values for many of the tested problem sizes, while they for problem instances where $|\mathbf{T}| > |\mathbf{W}|$ are much worse. These results are in line with the analytical arguments in [7], predicting that it will work well to approximate the static asset-based weapon allocation problem with its target-based counterpart on problem instances involving a strong defense (a large number of firing units compared to the number of targets), while the approximation will work bad in cases of a weak defense (i.e., problem instances where there are more targets than firing units). Although the differences between e.g., EMMR and PSO-S or GA-S and PSO-S are not very large for problem instances involving a strong defense, the differences should not be ignored, since such small but significant differences can have severe

impact on the end result if such algorithms are applied in a real-world C-RAM system.

6. CONCLUSIONS AND FUTURE WORK

We have in this article presented the static asset-based weapon allocation problem, which is an optimization problem that needs to be solved in a short amount of time in air defense situations involving RAM threats such as rockets and mortars. We have also presented the static target-based weapon allocation problem, but the focus has been on the asset-based case. We have implemented two versions of a genetic algorithm, two versions of a particle swarm optimization algorithm, and various versions of the greedy maximum marginal return algorithm. Such algorithms have earlier been used for the static target-based weapon allocation problem, but as far as we know, it is previously unknown how they perform on the asset-based version of the problem. Our experiments have shown that optimal or very near-optimal solutions are obtained in real-time by the genetic algorithms and the particle swarm optimization algorithms on small-scale problems. The standard maximum marginal return algorithm yields worse solutions, but these can easily be improved upon by local search, or by using an enhanced version of the algorithm. However, the quality does not become as good as that of the genetic algorithms or the particle swarm optimization algorithms.

For larger problem instances the optimal solutions are not known, and can therefore not be used for comparison. Instead, the objective function values produced by the algorithms have been compared to each other. It has been shown that the greedy algorithms create solutions of good quality (compared to the other algorithms) for scenarios with a strong defense, but that they perform bad on scenarios involving a weak defense, i.e., where there is a larger number of targets than there are firing units.

The algorithm that has been performing the best on large-scale problem instances is a novel improvement on the particle swarm optimization algorithm where the initial population is seeded with individuals based on small variations of the solution returned by the enhanced maximum marginal return algorithm. For the problem instances where there is a strong defense, the algorithm is not able to improve very much on the solution returned by the enhanced maximum marginal return algorithm, but for the problem instances involving a weak defense, the difference is dramatic. For problems of quite small scale, the difference in solution quality is very small between the particle swarm optimization algorithm and its seeded version. However, as the problem size increases, the difference in solution quality becomes very evident.

6.1. Future Work

The obtained results can be used as benchmarks for other heuristic algorithms. Hence, it is our hope that the used data sets (problem instances) will be used by other researchers as well, so that a better understanding of which algorithms that work well for static asset-based weapon allocation is obtained. Moreover, in the current research on static asset-based weapon allocation, it is assumed that kill probabilities, lethality probabilities, and target aims are known with certainty. Obviously, these estimates will in real-world systems be associated with uncertainty, and it would therefore be interesting and useful to know how sensitive the solutions produced by the algorithms are to such uncertainties.

In the experiments presented in this article, we have been generating problem instances in which there are no dependences among the values of the parameters. As an example, there is no correlation between any of the kill probabilities involving a specific target (or rather, there might be such correlation, but if so, this is by pure chance). This is consistent with how weapon allocation algorithms have been evaluated earlier in reported literature, but it can be discussed whether this lack of structure really would be seen in estimated kill probabilities from real-world air defense scenarios. Thinking of such a scenario, two targets, T_1 and T_2, of the same type, approaching a firing unit W_1 from the same direction and on the same altitude, would most likely result in kill probabilities P_{11} and P_{12} being quite similar. Likewise, two firing units W_2 and W_3 would obtain kill probabilities of approximately same magnitude, given that the firing units were positioned close together and being of the same type. Hence, the random fashion in which problem instances have been generated here (and in previous reported experiments with weapon allocation algorithms) may not necessarily create the same kinds of search spaces that would be experienced in real-world air defense situations. An idea that could be of interest for the future is therefore to create problem instances with an inbound structure that better reflect reality.

ACKNOWLEDGMENT

We would like to express gratitude to the reviewers for their constructive comments and suggestions that have helped to improve the article. This work was supported by the Information Fusion Research Program (University of Skövde, Sweden) in partnership with Saab AB and the Swedish Knowledge Foundation under grant 2003/0104.

REFERENCES

[1] R. Ahuja, A. Kumar, K. Jha, and J. Orlin
Exact and heuristic methods for the weapon target assignment problem.
Operations Research, **55**, 6 (2007), 1136–1146.

[2] J. Chen, B. Xin, Z. Peng, L. Dou, and J. Zhang
Evolutionary decision-makings for the dynamic weapon-target assignment problem.
Science in China Series F: Information Sciences, **52**, 11 (2009), 2006–2018.

[3] C. K. Cheong
Survey of investigations into the missile allocation problem.
Master's thesis, Naval Postgraduate School, Monterey, CA. 1985.

[4] C. Ciobanu and G. Marin
On heuristic optimization.
An. Stiint. Univ. Ovidius Constanta, **9**, 2 (2001), 17–30.

[5] G. G. den Broeder, R. E. Ellison, and L. Emerling
On optimum target assignments.
Operations Research, **7**, 3 (1959), 322–326.

[6] A. R. Eckler and S. A. Burr
Mathematical Models of Target Coverage and Missile Allocation.
Technical Report DTIC: AD-A953517, Military Operations Research Society, Alexandria, VA, 1972.

[7] P. A. Hosein
A Class of Dynamic Nonlinear Resource Allocation Problems.
Ph.D. thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1990.

[8] P. A. Hosein and M. Athans
Preferential defense strategies: Part 1—the static case.
Technical report, Massachusetts Institute of Technology, 1990.

[9] C. Huaiping, L. Jingxu, C. Yingwu, and W. Hao
Survey of the research on dynamic weapon-target assignment problem.
Journal of Systems Engineering and Electronics, **17**, 3 (2006), 559–565.

[10] K. C. Jha
Very large-scale neighborhood search heuristics for combinatorial optimization problems.
Ph.D. thesis, University of Florida, 2004.

[11] F. Johansson
Evaluating the performance of TEWA systems.
Ph.D. thesis, Örebro University, 2010.

[12] F. Johansson and G. Falkman
A comparison between two approaches to threat evaluation in an air defense scenario.
In *Proceedings of the 5th International Conference on Modeling Decisions for Artificial Intelligence*, 2008, 110–121.

[13] F. Johansson and G. Falkman
An empirical investigation of the static weapon-target allocation problem.
In *Proceedings of the 3rd Skövde Workshop on Information Fusion Topics*, 2009.

[14] F. Johansson and G. Falkman
A suite of metaheuristic algorithms for static weapon-target allocation.
In *Proceedings of the 2010 International Conference on Genetic and Evolutionary Methods*, 2010.

[15] F. Johansson and G. Falkman
SWARD: System for weapon allocation research & development.
In *Proceedings of the 13th International Conference on Information Fusion*, 2010.

[16] B. A. Julstrom
String- and permutation-coded genetic algorithms for the static weapon-target assignment problem.
In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2009.

[17] O. Karasakal
Air defense missile-target allocation models for a naval task group.
Computers and Operations Research, **35**, 6 (2008), 1759–1770.

[18] S. E. Kolitz
Analysis of a maximum marginal return assignment algorithm.
In *Proceedings of the 27th Conference on Decision and Control*, 1988.

[19] Z-J. Lee and C-Y. Lee
A hybrid search algorithm with heuristics for resource allocation problem.
Information Sciences, **173**, 1–3 (2005), 155–167.

[20] Z-J. Lee, C-Y. Lee, and S-F. Su
Parallel ant colonies with heuristics applied to weapon-target assignment problems.
In *Proceedings of the 7th Conference on Artificial Intelligence and Applications*, 2002.

[21] Z. J. Lee and W. L. Lee
A hybrid search algorithm of ant colony optimization and genetic algorithm applied to weapon-target assignment problems.
In *Proceedings of the 4th International Conference on Intelligent Data Engineering and Automated Learning*, 2003, 278–285.

[22] Z. J. Lee, S. F. Su, and C. Y. Lee
A genetic algorithm with domain knowledge for weapon-target assignment problems.
Journal of the Chinese Institute of Engineers, **25**, 3 (2002), 287–295.

[23] S. P. Lloyd and H. S. Witsenhausen
Weapon allocation is NP-complete.
In *Proceedings of the 1986 Summer Conference on Simulation*, 1986.

[24] W. P. Malcolm
On the character and complexity of certain defensive resource allocation problems.
Technical Report DSTO-TR-1570, DSTO, 2004.

[25] A. S. Manne
A target-assignment problem.
Operations Research, **6**, 3 (May–June 1958), 346–351.

[26] S. Matlin
A review of the literature on the missile-allocation problem.
Operations Research, **18**, 2 (1970), 334–373.

[27] W. A. Metler and F. L. Preston
A suite of weapon assignment algorithms for a SDI midcourse battle manager.
Technical report, Naval Research Laboratory, 1990.

[28] R. A. Murphey
Target-based weapon target assigment problems.
In P. M. Pardalos and L. S. Pitsoulis (Eds.), *Nonlinear assignment problems: algorithms and applications*, 2000, 39–53.

[29] S. Paradis, A. R. Benaskeur, M. Oxenham, and P. Cutler
Threat evaluation and weapons allocation in network-centric warfare.
In *Proceedings of the 8th International Conference on Information Fusion*, 2005.

[30] R. L. Rardin and R. Uzsoy
Experimental evaluation of heuristic optimization algorithms: A tutorial.
Journal of Heuristics, 7 (2001), 261–304.

[31] M. St. John, D. I. Manes, H. S. Smallman, B. Feher, and J. G. Morrison
An intelligent threat assessment tool for decluttering naval air defense displays.
Technical report, SSC San Diego, CA, 2004.

[32] B. Suman and P. Kumar
A survey of simulated annealing as a tool for single and multiobjective optimization.
Journal of the Operational Research Society, 57 (2006), 1143–1160.

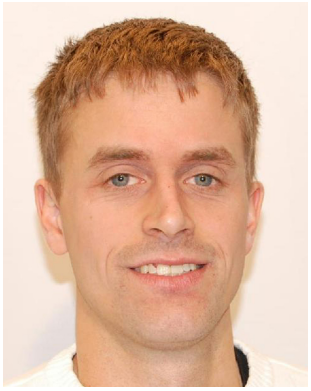
[33] P. Teng, H. Lv, J. Huang, and L. Sun
Improved particle swarm optimization algorithm and its application in coordinated air combat missile-target assignment.
In *Proceedings of the 7th World Congress on Intelligent Control and Automation*, 2008.

[34] F. van den Bergh and A. P. Engelbrecht
A new locally convergent particle swarm optimiser.
In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 2002.

[35] E. Wacholder
A neural network-based optimization algorithm for the static weapon-target assignment problem.
ORSA Journal on Computing, **1**, 4 (1989), 232–246.

[36] W. L. Winston
Operations Research: Applications and Algorithms.
Wadsworth Publishing Company. 1997.

[37] X. Zeng, Y. Zhu, L. Nan, K. Hu, B. Niu, and X. He
Solving weapon-target assignment problem using discrete particle swarm optimization.
In *Proceedings of the 6th World Congress on Intelligent Control and Automation*, 2006.



Fredrik Johansson obtained his M.Sc. in computer science from University of Skövde, Sweden, in 2005 and his Ph.D. in computer science from Örebro University, Sweden, in 2010.

During his Ph.D. studies he was a member of the Skövde Artificial Intelligence Lab (SAIL) and the Information Fusion Research Program at the Informatics Research Centre (IRC) in Skövde. Currently, he works as a scientist at the Swedish Defence Research Agency (FOI) in Kista.

His research interests are applied artificial intelligence and high-level information fusion, and the application of probabilistic techniques such as Bayesian networks for decision support. He is also interested in the use of techniques such as social network analysis, natural language processing, and web harvesting for supporting the work of intelligence analysts.

Göran Falkman obtained his Ph.D. in computing science from Chalmers University of Technology, Sweden, in 2003.

He holds a position as an Associate Professor of Computer Science, with a specialty in Interactive Knowledge Systems, at University of Skövde, Sweden, where he works as a researcher and senior lecturer within the Skövde Artificial Intelligence Lab (SAIL) at the Informatics Research Centre (IRC). He has been a project leader for three applied research projects within the area of information fusion, focusing on algorithms for threat evaluation and weapon allocation, visual analytics and maritime domain awareness, and anomaly detection for surveillance applications, respectively. He has also been the leader for the Situation Awareness scenario within the Infofusion research program at University of Skövde. Currently, he is one of the principal investigators of the Uncertainty Management in High-Level Information Fusion (UMIF) research project. Since 2009, he is an elected member of the Executive Board of the Swedish Artificial Intelligence Society (SAIS).

The research interests lie in the intersection of applied artificial intelligence, knowledge systems, interaction design, and information fusion. This includes work on the design, implementation and use of formal knowledge representation and knowledge-based systems (especially, case-based reasoning, ontology engineering, and the Semantic Web), as well as the use of interactive visualization for supporting knowledge-based reasoning processes (especially, situation analysis and decision-making).