SWARD: System for Weapon Allocation Research & Development

Fredrik Johansson
Informatics Research Centre
University of Skövde, Sweden.
fredrik.johansson@his.se

Göran Falkman
Informatics Research Centre
University of Skövde, Sweden.
goran.falkman@his.se

Abstract – *The allocation of firing units to hostile targets is an important process within the air defense domain. Many algorithms have been proposed for solving various weapon allocation problems, but evaluation of the performance of such algorithms is problematic, since it does not exist any standard scenarios on which to test the algorithms. It is to a large extent unknown how weapon allocation algorithms compare to each other when it comes to solution quality. We have developed the testbed SWARD, making it possible to systematically compare algorithm performance, and to support the development of new weapon allocation algorithms.*

Keywords: Air defense, performance evaluation, testbed, weapon allocation.

1 Introduction

A well-known reference model within the information fusion community is the so-called Joint Directors of Laboratories' (JDL) data fusion model (see e.g., [10, 9, 19, 23]), which provides a useful distinction among information fusion processes by dividing them into a number of different levels. The object assessment level is the most mature area of information fusion [10], while higher fusion levels are more immature. One important reason for why the research on high-level information fusion still is quite immature is due to the lack of standard data sets on which developed algorithms can be tested. Since researchers evaluate their algorithms and methods in different ways, it becomes very hard to get a good understanding of which algorithms that are promising and which are not. This situation is summarized by the following quote, taken from [23]:

> "The lack of common engineering standards for data fusion systems has been a major impediment to integration and re-use of available technology. There is a general lack of standardized - or even well-documented - performance evaluation, system engineering methodologies, architecture paradigms, or multi-spectral models of targets and collection systems. In short, current developments do not lend themselves to objective evaluation, comparison or re-use."

We have in earlier papers ([13, 14]) discussed how threat evaluation algorithms for air defense can be compared. The output from the threat evaluation process is an important input to weapon allocation, which is a resource management/process refinement problem highly relevant for the information fusion community. Many different algorithms for weapon allocation have been developed, but existing comparisons of algorithms are based on randomly created problem instances, making it hard to compare performance results from different experiments. In this paper, we describe the open source testbed SWARD, which is intended for evaluating the performance of algorithms for static weapon allocation. By using SWARD, it becomes possible for different researchers to test their algorithms for weapon allocation on the same problem instances. This makes it much easier to systematically evaluate the performance of weapon allocation algorithms. The strength of these kinds of tools can be seen in the machine learning community, in which the Weka workbench together with the use of the famous UCI (University of California, Irvine) datasets (see [3]) have made it possible to empirically benchmark a large number of machine learning algorithms on various datasets. We hope that SWARD will be used by researchers interested in air defense and weapon allocation, so that various algorithms can be compared more systematically than is the case at present. We also argue that similar ideas can be used for evaluating other resource management problem related to information fusion, such as sensor management.

The rest of this paper is structured as follows. In Section 2, we describe important functionality within an air defense system, of which weapon allocation is one important function. In Section 3, the different parts of the SWARD testbed are described in detail, and an illustrative example of what kinds of experiments that SWARD can be used for is presented in Section 4. Section 5 is devoted for a discussion, in which we discuss the choices of releasing SWARD as an open source project and implementing it in Java. Finally, the paper is concluded in Section 6.

2 Air defense systems

There are many different functions that must be handled in military decision support systems for air defense (often referred to as TEWA systems). In [4], the functions of (i) target detection, (ii) target tracking, (iii) target identification, (iv) threat evaluation, and (v) weapons assignment (in this paper referred to as weapon allocation) are mentioned as important for such systems. Additional to these functions, an important requirement on TEWA systems is to have a good graphical user interface (GUI) that enhances the operators' situation awareness, since it in the end is the human rather than the machine that should make the decision of whether or not to engage a target.

Our view of how the various parts of a TEWA system are functionally related is shown in Figure 1. Short summarized descriptions of each function are provided in the following subsections.

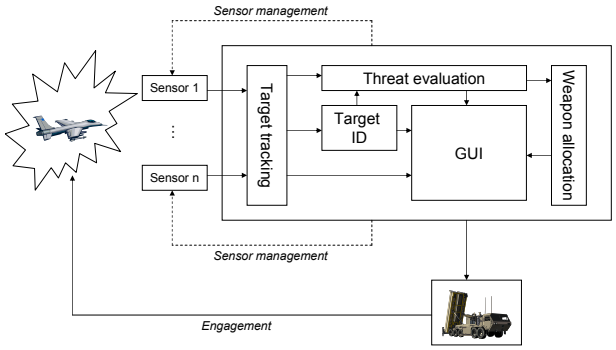

Figure 1: Functional overview of a TEWA system.

2.1 Target detection and tracking

Much research has been devoted to the low-level information fusion problems of target detection and target tracking. The target detection capabilities are obviously very dependent upon the choice of sensors and the quality of these, while target tracking basically is about estimating the current position and kinematics of a target, as well as predicting these into the future. Well-known examples of target tracking techniques are variations of Kalman filters and particle filters [2].

2.2 Target identification

Target identification concerns the transformation of a set of entity attributes such as speed, radar cross-section, shape, etc. into a label describing the identity of the target [10] (e.g., fighter, bomber, or helicopter). Algorithms for target identification are often based on pattern recognition techniques such as artificial neural networks [10].

2.3 Threat evaluation

Based upon information that can be derived from from the target tracking and target identification processes, the high-level information fusion problem of threat evaluation becomes to estimate the level of threat posed by detected air targets to defended assets [12, 22]. The calculated target values can be used to aid the human air defense decision makers on to which targets to focus their attention, but also to suggest which (if any) targets that should become subject for weapon allocation. Examples of algorithms that have been suggested for threat evaluation are Bayesian networks and fuzzy logic [12, 13].

2.4 Weapon allocation

Weapon allocation can be defined as the reactive assignment of defensive weapon resources to engage or counter identified threats [21]. Hence, the question to be answered by the weapon allocation process is which firing unit(s) to allocate to which target(s). Within *static* weapon allocation, which is the focus of this paper, it is assumed that all firing units are allocated in a single stage, while it in *dynamic* weapon allocation is allowed to allocate firing units to targets in a predefined number of discrete stages. A further division that can be done is the one between *target-based* and *asset-based* weapon allocation. In the former, calculated target values are used as a basis for which targets to engage, while information regarding the target aims and protection values of defended assets are used in the latter. The static target-based weapon allocation can be formalized as the non-linear optimization problem [11]:

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

subject to:

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

where $|\mathbf{T}|$ is the number of targets, $|\mathbf{W}|$ is the number of firing units, V_i is the target value of target T_i, P_{ik} is the kill probability for firing unit W_k on target T_i, and x_{ik} is a decision variable taking on value 1 if W_k is assigned to T_i and 0 otherwise. Similarly, the static asset-based weapon allocation problem can be defined as the non-linear optimization problem [11]:

$$F = \sum_{j=1}^{|\mathbf{A}|} \omega_j \prod_{i \in \mathbf{G}_j} (1 - \pi_i \prod_{k=1}^{|\mathbf{W}|} (1 - P_{ik})^{x_{ik}}), \tag{3}$$

subject to:

$$\begin{gathered} \sum_{i=1}^{|\mathbf{T}|} x_{ik} = 1, \ \ \forall k, \\ x_{ik} \in \{0, 1\}, \ \ \forall i \forall k, \end{gathered} \tag{4}$$

where $|\mathbf{A}|$ is the number of defended assets, ω_j is the protection value of defended asset A_j, $\mathbf{G}_j$ is the set of targets aimed for A_j, and π_i is the lethality value for target T_i.

Small-scale problem instances can be solved optimally in a short amount of time, but for many realistic problem instances, heuristic algorithms have to be used. Examples of such algorithms are ant colony optimization algorithms [17] and greedy algorithms [15, 6].

2.5 GUI

As stated in [10]: "it is important to realize that ultimately, the output from a data fusion system is aimed at supporting a human decision process". In the context of air defense, it is a human decision-maker and not an automated system that takes the final decision of whether a target should be engaged by a firing unit or not. For this reason, it is very important that the system can help the decision-maker with achieving a good situation awareness.

According to [18], a TEWA system should display the calculated target values graphically rather than numerically, since this decreases risk-taking behavior and increases the air defense officers trust in the system (since numbers imply a degree of precision that often is not known in the case of threat evaluation). Moreover, it should provide explanation-based reasoning capabilities, so that decision-makers understand the reason for calculated target values [18]. To the best of our knowledge, such aspects are often neglected in real-world TEWA systems.

3 SWARD

Performance evaluation of weapon allocation algorithms has earlier been problematic. Examples of studies where weapon allocation algorithms have been compared exist (see e.g. [16, 17]), but only involving a small number of algorithms. To use results from such experiments to compare against own algorithms is not possible since the experiments are based on randomly generated problem instances which can not be recreated. Hence, the only alternative is to reimplement other researchers' algorithms and create new experiments, but the algorithms are not often described detailed enough to allow for such implementations. For handling this problem, we have created SWARD (System for Weapon Allocation Research and Development).

SWARD is implemented in Java, in order to make it platform independent. It has been released under the open source license BSD so that anyone can use the testbed and modify the source code freely. SWARD can be downloaded from http://sourceforge.net/projects/sward/.

The SWARD testbed consists of: a *GUI* package, an *algorithm* package, a *scenario* package, and an *experiment* package. Additionally, there is a *threat evaluation* package, supporting different kinds of threat evaluation. There is also functionality for calculating statistics, etc. in a *util* package that will not be described here. In Figure 2, it is described how the most important classes of SWARD are related to each other.

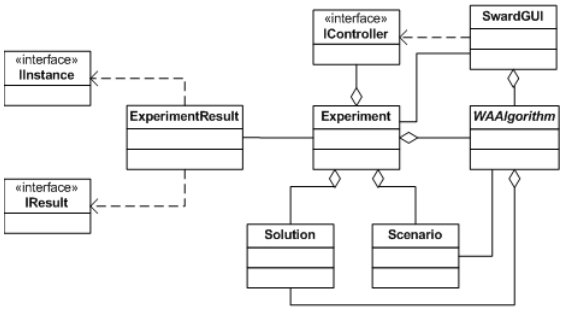

Figure 2: UML class diagram describing the relations between the most important classes in SWARD.

3.1 The scenario package

A central part of SWARD is the generation of scenarios (problem instances) on which to test different algorithms. There are two different kinds of scenarios that can be generated in SWARD; static target-based and static asset-based scenarios. A static target-based scenario consists of a matrix of kill probabilities (specifying the probability that a firing unit is able to destroy a target), and a vector of target values. Hence, given as input a parameter $|\mathbf{T}|$ (number of targets), a parameter $|\mathbf{W}|$ (number of firing units), and a seed s, a static target-based problem instance is generated, where kill probabilities are randomly generated in the interval $[0.6, 0.9]$ and target values are randomly generated in the interval $[25, 100]$. Obviously, these intervals can be changed, but these are the default settings that are used, for being in accordance with the experiments presented in [1]. By making use of the seed s, we can guarantee that problem instances easily can be recreated.

In the same way, static asset-based scenarios are generated from the input parameters $|\mathbf{T}|$, $|\mathbf{W}|$, $|\mathbf{A}|$ (number of defended assets), and a seed s. The generated asset-based scenarios consist of kill probabilities in the same interval as the target-based scenarios, but the target values are replaced with a vector of lethality probabilities (probability that a target destroys the defended asset it is aimed for), a vector of protection values (weights for the defended assets), and a vector of target aims (the defended assets the targets are aimed for). The default settings are that lethality probabilities are generated randomly in the interval $[0.3, 0.8]$, protection values in $[25, 100]$, while the target aims are generated in the following way: if $|\mathbf{T}| \geq |\mathbf{A}|$, the aim of target T_i is set to the ith defended asset, and the aims of the remaining targets are randomly selected from the defended assets. If $|\mathbf{T}| < |\mathbf{A}|$, the aim of target T_i is set to the ith defended asset, and the rest of the defended assets are not attacked by any targets.

3.2 The algorithm package

Within the algorithm package, there is an abstract class *WAAlgorithm* with a method for calculating the objective function value F for a given allocation of firing units to targets. The objective function value is calculated in ac-

cordance with Equation 1 for target-based scenarios, and in accordance with Equation 3 for asset-based scenarios. The class is declared as abstract since no object instances are allowed to be created of the class, it should instead be inherited by classes that are implementations of weapon allocation algorithms. There are currently nine different weapon allocation algorithms implemented within the package. These are: two versions of genetic algorithms (with and without seed), an ant colony optimization algorithm, a random search algorithm, an exhaustive search algorithm, three variants of greedy search algorithms, and a particle swarm optimization algorithm. Other researchers are encouraged to implement their own algorithms for weapon allocation into the testbed.

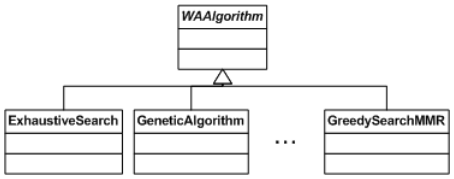

Figure 3: UML class diagram illustrating that specific weapon allocation algorithms such as a genetic algorithm are implemented as classes that inherits attributes and methods from the abstract class WAAlgorithm.

3.3 The experiment package

An experiment is set up by specifying parameters for which value of $|\mathbf{T}|$ to start with, to end with, and how much to increment this value in each step (from now on referred to as T_{start}, T_{end}, and T_{inc}. The same kind of parameters should be set for $|\mathbf{W}|$. These parameters are referred to as W_{start}, W_{end}, and W_{inc}. Additionally, the number of problem instances that should be tested for each problem size ($|\mathbf{T}|, |\mathbf{W}|$) should be specified using the parameter $nrOfIterations$, as well as whether the generated problem instances should be asset-based or target-based. Finally, the parameter $searchTime$, which is the amount of time that each algorithm is allowed to search for a solution, is needed. Together, these parameters define an experiment setup.

The results from an experiment are handled using the class *ExperimentResult*. This class uses internal helper classes implementing the interfaces *IInstance* and *IResult*, respectively, which also are defined within the same package. Results that can be stored from an experiment are the algorithms' calculated objective function values, and the algorithms' obtained ranks for each tested problem instance. Moreover, average objective function values and average ranks can be calculated, together with standard deviations. If all algorithms participating in an experiment obtain different objective function values on a problem instance, the best algorithm obtains rank 1, the second best rank 2 and so on, for that particular problem instance. However, if several algorithms obtain the same objective function value, the calculations of ranks become more complex. The rank for an algorithm is based on the number of algorithms that achieved better objective function values and the number of algorithms that achieved an equal objective function values, as can be seen in Algorithm 1 (this is valid for target-based scenarios, while the $<$ on row 4 should be changed to $>$ for asset-based scenarios).

Algorithm 1 Pseudo code for calculation of ranks.

```
1  for alg ← 1 to nrOfAlgorithms do
2    lesser ← 0, equal ← 0, rank ← 1
3    for alg2 ← 1 to nrOfAlgorithms do
4      if FValue(alg) < FValue(alg2) then
5        lesser ← lesser + 1
6      else
7        if FValue(alg) = FValue(alg2) then
8          equal ← equal + 1
9    rank ← rank + lesser
10   if equal > 1 then
11     extraRank ← 0
12     for cnt ← 1 to equal do
13       extraRank ← extraRank + cnt
14     rank ← rank + extraRank/equal
```

3.4 The GUI package

Experiments can be set up using code, but in most cases it is more convenient to use the GUI that has been developed for SWARD. Figure 4 illustrates what the graphical user interface looks like. The values that can be edited within the text boxes corresponds to the experiment parameters described in Section 3.3, so that all relevant experiment settings can be adjusted within the graphical user interface. The experiment settings chosen within the graphical user interface are accessible by the use of the interface *IController*, which also is part of the GUI package.

3.5 The threat evaluation package

As described in Section 2.4, a vital input to the (target-based) weapon allocation process is target values, as given by the threat evaluation process of a TEWA system. In order to support trusted explanation-based threat evaluation [18] (see Section 2.5), SWARD algorithms take as their input general threat evaluation objects. The generic threat evaluation framework used in SWARD supports both quantitative (numeric) and more qualitative (text-based) threat evaluation. Threat evaluation objects also include information about the quality (confidence and ambiguity) [20] and pedigree (origin and history) of target values [5].

The *IThreatEvaluation* interface declares methods for setting and getting target values, as well as their confidence and pedigree. The SWARD distribution includes two basic threat values: numeric (real-valued) and quantitative (string-valued).

The purpose of the *confidence* package is to support many different uncertainty management schemes (e.g., Bayesian probability or Dempster-Shafer theory). Inspired by the

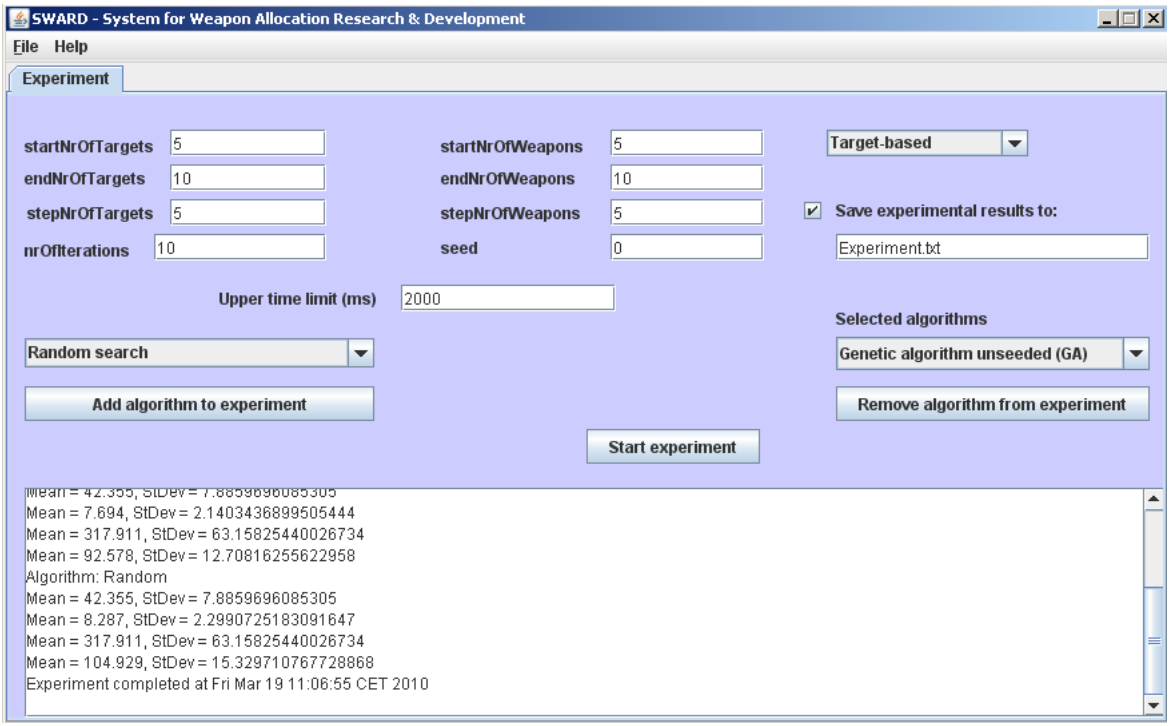

Figure 4: Screenshot of the SWARD graphical user interface.

CAED model [20], the *IConfidence* interface declares functionality for setting and getting the accuracy (e.g., probability or belief) and ambiguity (e.g., precision) of target values. Predefined types of accuracy are numeric (e.g., a real value representing the probability on the interval $[0, 1]$) and qualitative (e.g., a text string representing the judgement "unlikely"). The precision of a target value is measured relative the "bounds" of a given *IAmbiguity* object. As a subclass of numeric ambiguity, SWARD provides an *Ambiguity1D* class that models ambiguity as a 1-dimensional interval.

In order to increase operators' trust in a TEWA system, it is important to provide access to and support presentation of "data lineage", i.e., the history of how data is transformed by data processing methods (algorithms) [5]. In SWARD, the *pedigree* package addresses this issue by providing basic functionality for the tracing of target values in terms of generic data operations. An *IPedigree* object is essentially a list of processing events, where a processing event is a time-stamped and uniquely identified (through a universally unique identifier, UUID) data operation (operator plus operands).

To facilitate the use of different threat evaluation methods, as well as the construction of new methods, various object-oriented design patterns are used. The *confidence* package uses a bridge pattern [8, p. 151] to decouple the abstraction of confidence of target values from its implementation. By attaching different implementations of accuracy and ambiguity to confidence classes, the actual confidence method used can be altered and new confidence methods can be constructed. The top-level *threat evaluation* package makes use of a factory method pattern [8, p. 107] to defer the instantiation of threat evaluation components (target value, confidence, accuracy, ambiguity and pedigree objects) to implementations of the *IThreatEvaluationFactory* interface. By changing the factory, the actual threat evaluation method used can be altered. Two basic factories for numeric and qualitative threat evaluation are provided. SWARD's default threat evaluation factory uses real-valued target values with precise accuracy and null (empty) pedigree.

4 Example

In this section, we give an illustrative example of the kind of experiments SWARD can be used for. We have chosen to compare the performance of three different weapon allocation algorithms: a greedy algorithm known as the maximum marginal return (MMR) algorithm (originally presented in [6]), a simple random search algorithm, and a more complex genetic algorithm. To start with, the algorithms are added to the experiment. An algorithm is added by selecting it in the combo box to the left in Figure 4 and then clicking on the button "Add algorithm to experiment". Once all algorithms have been added to the experiment, we choose the type of scenarios to be "Target-based", so that this is the type of problem instances that will be generated. Next, we would like to determine how large problem sizes that should be tested. We want to generate problem instances of size ($|\mathbf{T}| = 10, |\mathbf{W}| = 10$), ($|\mathbf{T}| = 15, |\mathbf{W}| = 10$), and ($|\mathbf{T}| = 20, |\mathbf{W}| = 10$). Therefore, we set $startNrOfTargets$ to 10, $endNrOfTargets$ to 20, and $stepNrOfTargets$ to 5, while the corresponding numbers for the weapons all are set

to 10 (in fact, what $stepNrOfWeapons$ is set to does not matter in this case). The seed is for convenience set to 0. The higher value of $nrOfIterations$ we choose, the higher statistical confidence we can have in the results. However, the experiment will also take longer time with a high value, so there is a trade-off. We here choose to use the value 100, so that the algorithms will be evaluated at 100 problem instances for each of the three tested problem sizes. We set the time limit to 1000, so that each algorithm is allowed to search for solutions for one second on each problem instance. In total, this experiment will take at most 900 seconds to run (the MMR algorithms does not make use of all its search time). We decide to save the experimental results to disk by checking the appropriate check box. Finally, the experiment is started by clicking on "Start experiment".

When the experiment is finished, the algorithms' average objective function values are printed on screen, together with the calculated standard deviations (as shown in the bottom of Figure 4. More detailed statistics for each problem instance is written to file, as well as information regarding the algorithms' rank on each problem instance, if such statistics is wanted. Table 1 shows the resulting average objective function values obtained from the experiment described here.

Table 1: Average objective function values (with σ within parentheses). Averaged over 100 problem instances.

	10×10	15×10	20×10
MMR	127.0 (18.7)	332.7 (32.6)	591.2 (55.3)
Random	109.4 (13.2)	346.5 (36.4)	629.8 (58.5)
GA	93.0 (11.2)	311.6 (34.2)	579.3 (58.0)

As can be seen in Table 1, the genetic algorithm's (denoted as GA) average objective function values are the lowest for all the three tested problem sizes. Since we have formulated the static target-based weapon allocation problem as a minimization problem in Equation 1, this indicates that the genetic algorithm on average has found the solutions of highest quality among the three tested algorithms.

5 Discussion

A thing worth noticing is that the choice of implementing SWARD in Java means that the computer hardware is not necessarily made use of optimally due to the fact that Java is interpreted by a Java Virtual Machine. Much of the original slowness of Java was removed when features like Just-In-Time (JIT) compilation and adaptive optimization were introduced, but it is still likely that the SWARD code written in Java could have been made more efficient in e.g., C++. Since SWARD is a testbed rather than a weapon allocation module intended for being plugged into a real-world TEWA system, we think that the possibly unoptimized execution times are outweighed by the platform independence made possible by the choice of implementing the code in Java. However, Java will not be a suitable choice for a real-world TEWA system due to the reason mentioned above, and due to the automatic garbage collection in Java that can be a potential problem for critical real-time applications.

The choice to release SWARD as an open source project can be questioned, since there typically is a strong tradition to keep defense-related research non-public. A shift towards open source software can however be seen in many places, and according to a memorandum from U.S. Department of Defense [7], open source software is particularly suitable for experimentation and rapid prototyping. In this way, SWARD should be able to facilitate the development and testing of algorithms for weapon allocation. A problem might be that some researchers are not allowed to distribute their algorithms further for security reasons. However, even though this may be the case, we think that SWARD still can be used for benchmarking. Since it is possible to easily recreate experiments, it becomes possible to compare the objective function values between various algorithms without actually requiring to give away the algorithms themselves.

An information fusion resource management problem that is closely related to that of weapon allocation is the problem of allocating sensors to tasks, in [24] referred to as sensor resource allocation. Like the weapon allocation problem, this can be stated as an optimization problem, and we think that a testbed based on the same concepts as SWARD would be beneficial for research on sensor resource allocation.

6 Conclusions

We have in this paper presented the open source testbed SWARD (System for Weapon Allocation Research and Development), which has been developed for evaluating the performance of static weapon allocation algorithms. By using SWARD, it becomes easy to test and compare different weapon allocation algorithms on the same problem instances, allowing for more standardized and repeatable experiments. The testbed has also been made available for free download, so that any researcher can add more weapon allocation algorithms into the testbed, and run own experiments. An example experiment has been presented, in order to describe how easily experiments with static weapon allocation algorithms can be created using the graphical user interface provided as a part of SWARD. We encourage other researchers to implement their own weapon allocation algorithms into SWARD, to provide a better understanding of which kind of algorithms that are suitable for different conditions, such as various number of targets and firing units, and varying amounts of time available for searching for good solutions.

Acknowledgment

This work was supported by the Information Fusion Research Program (University of Skövde, Sweden) in partnership with Saab Electronic Defense Systems and the Swedish Knowledge Foundation under grant 2003/0104.

References

[1] Ravindra Ahuja, Arvind Kumar, Krishna Jha, and James Orlin. Exact and heuristic methods for the weapon target assignment problem. *Operations Research*, 55(6):1136–1146, 2007.

[2] Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2001.

[3] Arthur Asuncion and David Newman. UCI machine learning repository, 2007.

[4] Abder Rezak Benaskeur, Éloi Bossé, and Dale Blodgett. Combat resource allocation planning in naval engagements. Technical Report TR 2005-486, Defence R&D Canada - Valcartier, 2007.

[5] Marion G. Ceruti, Subrata Das, Adam Ashenfelter, Gary Raven, Richard Brooks, Moises Sudit, Genshe Chen, and Edward Wright. Pedigree information for enhanced situation and threat assessment. In *Proceedings of the 9th International Conference on Information Fusion*, 2006.

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

[7] DoD Chief Information Officer Memorandum. Clarifying guidance regarding open source software (OSS), October 2009.

[8] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley, 1995.

[9] David Hall and James Llinas. *Handbook of Multisensor Data Fusion*. CRC Press, Boca Raton, FL, USA, 2001.

[10] David Hall and Sonya McMullen. *Mathematical Techniques in Multisensor Data Fusion*. Artech House, 2004.

[11] Patrick A. Hosein. *A class of dynamic nonlinear resource allocation problems*. PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1990.

[12] Fredrik Johansson and Göran Falkman. A Bayesian network approach to threat evaluation with application to an air defense scenario. In *Proceedings of the 11th International Conference on Information Fusion*, 2008.

[13] Fredrik Johansson and Göran 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 (LNAI 5285)*, 2008.

[14] Fredrik Johansson and Göran Falkman. A testbed based on survivability for comparing threat evaluation algorithms. In *Proceedings of the SPIE Symposium on Defense, Security and Sensing*, 2009.

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

[16] Zne-Jung Lee and Chou-Yuan Lee. A hybrid search algorithm with heuristics for resource allocation problem. *Inf. Sci. Inf. Comput. Sci.*, 173(1-3):155–167, 2005.

[17] Zne-Jung Lee, Chou-Yuan Lee, and Shun Feng Su. An immunity-based ant colony optimization algorithm for solving weapon-target assignment problem. *Applied Soft Computing*, 2:39–47, 2002.

[18] Michael J. Liebhaber and Bela Feher. Air threat assessment: Research, model, and display guidelines. In *Proceedings of the 2002 Command and Control Research and Technology Symposium*, 2002.

[19] James Llinas, Christopher L. Bowman, Galina L. Rogova, Alan N. Steinberg, Edward L. Waltz, and Frank E. White. Revisions and extensions to the JDL data fusion model II. In *Proceedings of the Seventh International Conference on Information Fusion*, 2004.

[20] Aaron R. Newman. Confidence, pedigree, and security classification for improved data fusion. In *Proceedings of the 5th International Conference on Information Fusion*, 2002.

[21] Stéphane Paradis, Abder Rezak Benaskeur, Martin Oxenham, and Philip Cutler. Threat evaluation and weapons allocation in network-centric warfare. In *Proceedings of the 8th International Conference on Information Fusion*, 2005.

[22] Jean Roy, Stéphane Paradis, and Mohamad K. Allouche. Threat evaluation for impact assessment in situation analysis systems. In I. Kadar, editor, *Proceedings of SPIE: Signal Processing, Sensor Fusion, and Target Recognition XI*, volume 4729, pages 329–341, July 2002.

[23] Alan Steinberg, Christopher Bowman, and Frank White. Revisions to the JDL data fusion model. In *Proceedings of the SPIE Sensor Fusion: Architectures, Algorithms, and Applications III*, pages 430–441, 1999.

[24] Ning Xiong and Per Svensson. Multi-sensor management for information fusion: issues and approaches. *Information Fusion*, 3:163–186, 2002.