Evaluating the Performance of TEWA Systems

Örebro Studies in Technology 40

Fredrik Johansson

Evaluating the Performance of TEWA Systems

UNIVERSITY OF SKÖVDE

Title: Evaluating Evaluating the Performance of TEWA Systems

Publisher: Örebro University 2010
www.publications.oru.se
trycksaker@oru.se

Printer: Intellecta Infolog, Kållered 10/2010

ISSN 1650-8580
ISBN 978-91-7668-761-1

Abstract

It is in military engagements the task of the air defense to protect valuable assets such as air bases from being destroyed by hostile aircrafts and missiles. In order to fulfill this mission, the defenders are equipped with sensors and firing units. To infer whether a target is hostile and threatening or not is far from a trivial task. This is dealt with in a threat evaluation process, in which the targets are ranked based upon their estimated level of threat posed to the defended assets. Once the degree of threat has been estimated, the problem of weapon allocation comes into the picture. Given that a number of threatening targets have been identified, the defenders need to decide on whether any firing units shall be allocated to the targets, and if so, which firing unit to engage which target. To complicate matters, the outcomes of such engagements are usually stochastic. Moreover, there are often tight time constraints on how fast the threat evaluation and weapon allocation processes need to be executed. There are already today a large number of threat evaluation and weapon allocation (TEWA) systems in use, i.e. decision support systems aiding military decision makers with the threat evaluation and weapon allocation processes. However, despite the critical role of such systems, it is not clear how to evaluate the performance of the systems and their algorithms. Hence, the work in thesis is focused on the development and evaluation of TEWA systems, and the algorithms for threat evaluation and weapon allocation being part of such systems. A number of algorithms for threat evaluation and static weapon allocation are suggested and implemented, and testbeds for facilitating the evaluation of these are developed. Experimental results show that the use of particle swarm optimization is suitable for real-time target-based weapon allocation in situations involving up to approximately ten targets and ten firing units, while it for larger problem sizes gives better results to make use of an enhanced greedy maximum marginal return algorithm, or a genetic algorithm seeded with the solution returned by the greedy algorithm.

Keywords: air defense, information fusion, performance evaluation, threat evaluation, TEWA, weapon allocation

Acknowledgments

First of all, I would like to express my gratitude to Göran Falkman. You have been a great support during my years as a PhD-student, and I could not have wished for a better supervisor. Thanks also to my co-supervisors Lars Niklasson and Lars Karlsson.

I would also like to take the opportunity to thank for all the feedback and detailed comments on the draft of this thesis that I have received from Håkan Warston and my external reviewer Egils Sviestins. Your comments have been very appreciated and have helped me to improve the quality of the thesis significantly. I am also grateful to Martin Smedberg and Thomas Kronhamn. All of you have provided much knowledge to this project, not least from an industrial perspective. Thanks also to Saab AB for the support.

Many thanks to my colleagues at University of Skövde: Christoffer Brax and Richard Laxhammar for being my room mates in the "Aquarium", and Anders Dahlbom, Alexander Karlsson, Maria Nilsson, Maria Riveiro, Tove Helldin, Ronnie Johansson and Joeri van Laere for many fruitful and funny discussions over lunches and coffee breaks. I will certainly miss you guys when moving to Stockholm. A special thanks to all the members in the GSA group and the information fusion research program.

There are also many people outside the academic environment that have supported me in a number of ways, not least by providing an invaluable source of (positive) distraction from the doctoral studies. I would like to thank many of the members in Skövde Taekwon-do Club, you all know who you are. A special thanks to my friend Daniel Dongo with whom I have spent a lot of hours in the gym, the Do Jang, and everywhere else.

I would also like to show my gratitude to my family. My parents for always believing in and supporting me, and my sister for being my first teacher in mathematics (strict but loving) and participating in my (not so scientific) kitchen experiments when being younger. Thanks to my nieces Clara, Ella, and Maja for just being wonderful. Last, but certainly not least, I would like to thank my loved Marie for your patience during the writing of this thesis, and for always being there for me.

Contents

List of Publications

I. **Johansson, F.** and Falkman, G. (2010) Real-time allocation of defensive resources to rockets, artillery, and mortars. *Proceedings of the 13th International Conference on Information Fusion*, Edinburgh, United Kingdom, July 2010.

II. **Johansson, F.** and Falkman, G. (2010) SWARD: System for weapon allocation research & development. *Proceedings of the 13th International Conference on Information Fusion*, Edinburgh, United Kingdom, July 2010.

III. **Johansson F.** and Falkman G. (2010) A suite of metaheuristic algorithms for static weapon-target allocation. In Arabnia, H. R., Hashemi, R. R., and Solo, A. M. G. (Eds.): *Proceedings of the 2010 International Conference on Genetic and Evolutionary Methods*, pp. 132–138, CSREA Press.

IV. **Johansson F.** and Falkman G. (2009) An empirical investigation of the static weapon-target allocation problem. In Johansson, R., van Laere, J., and Mellin, J. (Eds.): Proceedings of the 3rd Skövde Workshop on Information Fusion Topics, *Skövde Studies in Informatics 2009:3*, pp. 63–67.

V. **Johansson F.** and Falkman, G. (2009) Performance evaluation of TEWA systems for improved decision support. In Torra, V., Narukawa, Y., and Inuiguchi, M. (Eds.): Proceedings of the 6th International Conference on Modeling Decisions for Artificial Intelligence, *Lecture Notes in Artificial Intelligence 5861*, pp. 205–216, Springer-Verlag, Berlin Heidelberg.

VI. **Johansson, F.** and Falkman, G. (2009) A testbed based on survivability for comparing threat evaluation algorithms. In Mott, S., Buford, J. F., Jakobson, G., and Mendenhall, M. J. (Eds.): *Proceedings of SPIE, Vol. 7352* (Intelligent Sensing, Situation Management, Impact Assessment, and Cyber-Sensing), Orlando, USA, April 2009.

VII. **Johansson, F.** and Falkman, G. (2008) A survivability-based testbed for comparing threat evaluation algorithms. In Boström, H., Johansson, R., and van Laere, J. (Eds.): Proceedings of the 2nd Skövde Workshop on

Information Fusion Topics, *Skövde Studies in Informatics 2008:1*, pp 22–24.

VIII. **Johansson, F.** and Falkman, G. (2008) A comparison between two approaches to threat evaluation in an air defense scenario. In Torra, V. and Narukawa, Y. (Eds.): Proceedings of the 5th International Conference on Modeling Decisions for Artificial Intelligence, *Lecture Notes in Artificial Intelligence 5285*, pp. 110–121, Springer-Verlag, Berlin Heidelberg.

IX. Niklasson, L., Riveiro, M., **Johansson, F.**, Dahlbom, A., Falkman, G., Ziemke, T., Brax, C., Kronhamn, T., Smedberg, M., Warston, H. and Gustavsson, P. (2008) Extending the scope of Situation Analysis. *Proceedings of the 11th International Conference on Information Fusion*, Cologne, Germany, July 2008.

X. **Johansson, F.** and Falkman G. (2008) A Bayesian network approach to threat evaluation with application to an air defense scenario. *Proceedings of the 11th International Conference on Information Fusion*, Cologne, Germany, July 2008.

XI. Riveiro, M., **Johansson, F.**, Falkman G. and Ziemke, T (2008) Supporting Maritime Situation Awareness Using Self Organizing Maps and Gaussian Mixture Models. In Holst, A., Kreuger, P., and Funk, P. (Eds.): Tenth Scandinavian Conference on Artificial Intelligence. Proceedings of SCAI 2008. *Frontiers in Artificial Intelligence and Applications 173*, pp. 84–91, IOS Press.

XII. **Johansson, F.** and Falkman, G. (2007) Detection of vessel anomalies — a Bayesian network approach. *Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, Melbourne, Australia, December 2007.

XIII. Niklasson, L., Riveiro, M., **Johansson, F.**, Dahlbom, A., Falkman, G., Ziemke, T., Brax, C., Kronhamn, T., Smedberg, M., Warston, H. and Gustavsson, P. (2007) A Unified Situation Analysis Model for Human and Machine Situation Awareness. In Koschke, R., Herzog, O., Rödiger, K.-H., and Ronthaler, M. (Eds.): Trends, Solutions, Applications. Proceedings of SDF 2007. *Lecture Notes in Informatics P-109*, pp. 105–110, Köllen Druck & Verlag.

XIV. **Johansson, F.** and Falkman, G. (2006) Implementation and integration of a Bayesian Network for prediction of tactical intention into a ground target simulator. *Proceedings of the 9th International Conference on Information Fusion*, Florence, Italy, July 2006.

List of Figures

List of Tables

List of Algorithms

Chapter 1
Introduction

On July 3 1988, in the Strait of Hormuz, the US Navy Cruiser USS Vincennes launches two missiles against what is supposed to be a hostile Iranian Air Force F-14 military aircraft in attack mode. USS Vincennes is at the same time engaged in a surface battle with an Iranian vessel (Smith et al., 2004). The aircraft is shot down, and later turns out to be the passenger airliner Iran Air Flight 655. As a result of the erroneous decision to open fire against the aircraft, 290 civilian people were killed (Fisher and Kingma, 2001). During the second Gulf war, nearly fifteen years later, US Army Patriot air defense units are involved in fratricide incidents where coalition aircrafts on two occasions are misclassified as hostile missiles. This results in the shoot down of a British Royal Air Force Tornado GR4 on March 22 2003, and a US Navy F/A-18 Hornet on April 2 the same year. These two fratricide incidents resulted in the killing of a total of three crew members (British Ministry of Defence, 2004; Defense Science Board Task Force, 2005).

The tragic incidents described above highlight the severe consequences that can follow from erroneous decision making in case of stressful air defense situations. Various investigations into these incidents have been undertaken by analysts and scholars, in which different causes to the incidents have been suggested. In the disaster with Iran Air Flight 655, factors such as an inexperienced crew with poor reaction to combat, lack of time, insufficient data quality, and failure of the battle management system (Fisher and Kingma, 2001) have been suggested as main factors contributing to the decision that brought so many people to their deaths, while e.g. classification criteria, rules of engagement, crew training, and malfunction of the system for identification were contributing factors in the fratricide incidents during the second Gulf war (British Ministry of Defence, 2004). Although part of the problem in the incidents mentioned above, the computerized support from air defense systems is invaluable for human air defense operators, due to the complex nature of the situations in which they have to take decisions. The decisions have to be taken under severe time pressure, based on imperfect data and information stemming from hetero-

geneous sensors (Paradis et al., 2005; Benaskeur et al., 2008). As stated in Kott et al. (1999):

> "Military decision makers face an immediate need for assistance in the job of transforming enormous amounts of low-level data, incomplete, uncoordinated and uncertain, into a few aggregated, understandable and actionable elements of information."

The tasks performed by air defense decision makers can be characterized as cognitively challenging already under normal conditions, and often become considerably harder as the tempo increases (Liebhaber and Feher, 2002a). For this reason, they are in large-scale, time-critical air defense situations in need of support with assessing whether there are any threatening hostile targets nearby (Tweedale and Nguyen, 2003), and if so, what actions or countermeasures that should be taken (Jorgensen and Strub, 1979; Brander and Bennet, 1991). Previous studies have shown that experienced operators are competent in responding quite efficiently to sequential threats, but that they in complex tactical situations involving multiple threats tend to come up with suboptimal solutions based solely on intuition and rules of thumb (Benaskeur et al., 2008). Such solutions are not guaranteed to give satisfactory results (Frini et al., 2008). Moreover, it is not unlikely that the decision makers will make fatal errors in such situations, due to the high level of stress and the complexity of the environment (Beaumont, 2004). Since air defense decision making has severe, and often catastrophic consequences if errors are made (Liebhaber and Feher, 2002a), the need for computerized support becomes crucial.

In this thesis, we deal with the development and evaluation of algorithms for so called threat evaluation and weapon allocation (TEWA) systems. These are computerized systems supporting human air defense decision makers with the processes of threat evaluation and weapon allocation. Informally, the purpose of threat evaluation is to determine the level of threat posed by detected air targets to defended assets such as air bases and power plants, while the purpose of weapon allocation is to (if necessary) allocate available firing units to threatening targets, in order to protect the defended assets. Military decision support systems such as TEWA systems are very much desired by decision makers having to make proper decision on the battlefield (Lee et al., 2002c; Beaumont, 2004). Most available research related to TEWA systems focuses solely on either the threat evaluation or weapon allocation process, with an overwhelming majority on the latter. Additionally, a majority of available research on threat evaluation is to be found in the research area of information fusion, while weapon allocation traditionally is studied within the field of operations research. An integrated view on threat evaluation and weapon allocation is rarely taken, despite that there are strong interdependencies between the two, in that the target values generated in the threat evaluation process have a large impact on the result of the weapon allocation process. As an example, a target obtaining a certain target value may have many firing units allocated to it, while it

may not be assigned any firing units at all in the case of a slightly lower target value. Moreover, the real-time requirements on the threat evaluation and weapon allocation algorithms that are part of a TEWA system are seldom taken into consideration, although the criticality of the real-time aspects is one of the major characteristics of real-world air defense situations (Allouche, 2006; Huaiping et al., 2006). As described in Joint Chiefs of Staff (2007), air defense operations require streamlined decision making processes, due to their time-sensitive nature. These requirements heavily influence what type of algorithms that are suitable for threat evaluation and weapon allocation. Nevertheless, the real-time aspects of this kind of decision problems have not traditionally been in focus of the operations research community (Séguin et al., 1997).

Due to the critical role of TEWA systems, the evaluation of such systems and their threat evaluation and weapon allocation algorithms becomes very important. Despite this, very little open research on performance evaluation of threat evaluation and weapon allocation algorithms exists, and the current level of evaluation provided in literature is too simplistic considering the consequences it can give to engage a non-hostile target, or to engage a hostile target too late. The performance of threat evaluation algorithms is rarely discussed at all, and if so, the target values (i.e. the estimated levels of threat posed by the detected air targets) produced by the suggested algorithms are only compared to expert knowledge on one or a few cases. The performance of weapon allocation algorithms is more often investigated, but as discussed above, the real-time aspects are often neglected. Thus, there is a need for research on evaluation of TEWA systems and the real-time performance of the threat evaluation and weapon allocation algorithms being part of such a system.

1.1 Aims and Objectives

In this dissertation, suitable algorithms for real-time threat evaluation and weapon allocation are investigated. Moreover, we also consider the problem of evaluation of such algorithms, as well as the TEWA systems to which the algorithms belong. Almost all existing research focus solely on either threat evaluation or weapon allocation, so one intended role of this thesis is also to provide an integrated view on threat evaluation and weapon allocation, filling the gap between the research fields of information fusion and operations research. The main research aim that will be addressed in this dissertation is:

Aim: *To investigate the performance of real-time threat evaluation and weapon allocation algorithms, and the TEWA systems of which such algorithms are central parts.*

In order to fulfill this aim, a number of objectives can be identified. These objectives are listed below, together with a short description of each objective:

O1. *Review and analyze existing algorithms and methods for threat evaluation and weapon allocation.*

A small number of algorithms for threat evaluation have been suggested in open literature, while considerably more algorithms have been developed for the problem of weapon allocation. By reviewing and analyzing existing literature on threat evaluation and weapon allocation algorithms, a more complete picture of the research problem can be created. Studying threat evaluation and weapon allocation algorithms together also leads to a more integrated view on threat evaluation and weapon allocation.

O2. *Suggest and implement algorithms for real-time threat evaluation and weapon allocation.*

Based on the outcome from the first objective, a smaller subset of promising algorithms for threat evaluation and weapon allocation, respectively, can be identified. By analyzing and adapting some of these, algorithms suitable for real-time use are developed.

O3. *Suggest methods for evaluating the performance of TEWA systems, and the threat evaluation and weapon allocation algorithms being part of such systems.*

The methodology (which at least in open literature) is used today for evaluating the performance of threat evaluation algorithms is rudimentary, if the algorithms are evaluated at all. In the case of evaluation of weapon allocation algorithms it is more straightforward how algorithms can be evaluated, since there exists an objective function value which can be used for comparison. However, as we will see, there are problems associated with this kind of evaluation as well. Therefore, methods for systematic evaluation of TEWA systems and their components have to be proposed.

O4. *Develop testbeds for performance evaluation.*

In order to demonstrate the usefulness of the suggested methods for performance evaluation of TEWA systems, and to facilitate the evaluation of threat evaluation and weapon allocation algorithms, testbeds need to be developed. Such testbeds make it easier to systematically compare the performance of different algorithms, leading to better and more robust evaluations.

O5. *Evaluate the performance of the developed algorithms.*

The purpose of this objective is to compare how the developed algorithms compare relative to other algorithms under various real-time constraints, and to provide concrete guidelines for what type of algorithms to use for different classes of situations.

The reviews of existing algorithms provide insights into which type of algorithms that can be suitable for threat evaluation and weapon allocation, and by implementing these, suggested methods for performance evaluation can be applied for comparing the (real-time) performance of the implemented algorithms. Hence, the fulfillment of all of the objectives makes it possible to reach the research aim put forward above.

1.2 Research Methodology

In order to fulfill the aim put forward in section 1.1, appropriate research methods need to be identified and used for the identified objectives.

The first objective, i.e. the review of existing algorithms and methods for threat evaluation and weapon allocation is addressed by performing two *literature surveys* (Dawson, 2000). In the first survey, the open literature on threat evaluation is reviewed, where most of the available literature is to be found in the research field of information fusion. The second survey summarizes the substantial work that has been published on static weapon allocation (of which most material can be found in literature related to military operations research).

For fulfilling the second objective, promising algorithms are identified and implemented, hence, the research methodology made use of is *implementation* (Berndtsson et al., 2002). This is also used for objective four, in which the suggested methods for evaluating TEWA systems and their threat evaluation and weapon allocation algorithms (the result from objective three) is realized in testbed implementations.

When evaluating and comparing the developed algorithms, i.e. objective five, a *quantitative method* is used. The performance of the developed weapon allocation algorithms is in a number of empirical *experiments* (Cohen, 1995) compared using a large number of problem instances of various size, on which the objective function values of the solutions produced by the different algorithms are evaluated. For small problem sizes, such comparisons are complemented with information regarding the produced solutions' deviation from the optimal solution. When evaluating the threat evaluation algorithms, the used quantitative measures are complemented with a more *qualitative analysis*, since there is a lack of realistic scenarios that can be used for providing robust quantitative numbers, and since there is no "physical" value the estimated threat values can be compared to.

The developed algorithms for threat evaluation have been presented and discussed with former air defense officers currently working within the defense industry (some of them working with TEWA systems). They have also been shown the suggested and developed methodology for evaluating TEWA systems and their embedded threat evaluation and weapon allocation algorithms using computer-based simulations, as well as provided valuable input to it. Additionally, a visit to the Swedish Armed Forces' Air Defence Regiment has been made, in which threat evaluation and weapon allocation in practice has been

discussed. Moreover, experts from Saab Electronic Defence Systems have been part of the author's research group and have regularly been giving feedback on new ideas, developed algorithms, evaluation methodology, etc.

1.3 Scientific Contribution

In this dissertation, a number of scientific contributions have been achieved. These contributions are outlined below, organized according to the areas of threat evaluation, weapon allocation, and TEWA systems. The first set of contributions is related to threat evaluation:

- Formalization of the threat evaluation process (section 3.1).

  As highlighted by Roux and van Vuuren (2007), threat evaluation is in the context of (ground-based) air defense a poorly defined process. The provided formalization of the threat evaluation process is an attempt to make this process more clearly defined, which is useful when discussing and developing algorithms for threat evaluation.

- A literature review of existing literature on parameters and algorithms suitable for threat evaluation (section 3.2).

  Various parameters have earlier been suggested for threat evaluation, but we are here summarizing these and classify them into the categories of capability, intent, and proximity parameters. Moreover, the algorithms that have been proposed for threat evaluation in open literature are reviewed. To the best of the authors' knowledge, no such review has earlier been undertaken. This review is mainly contributing to the research field of information fusion.

- Development of a Bayesian network for threat evaluation, and implementation of a fuzzy logic approach adapted from Liang (2006) (section 4.1).

  These implementations show how a subset of the reviewed parameters for threat evaluation can be used to estimate the level of threat posed by a target to a defended asset. This adds to research on high-level information fusion and military decision support.

The second set of contributions regards static weapon allocation:

- A literature review of existing algorithms for static weapon allocation (section 3.3).

  Various reviews have earlier been made on algorithms and methods for static weapon allocation, but most of these were made decades ago so that they focus on analytical approaches and do not cover more recent computer-based heuristic (i.e. approximate) techniques. Some more recent surveys do exist (see Malcolm (2004) and Huaiping et al. (2006)),

but these are far from exhaustive. Hence, this review complements earlier research on static weapon allocation and should be of value for the field of military operations research.

- Development of the open source testbed SWARD (System for Weapon Allocation Research and Development) for evaluation of static weapon allocation algorithms (section 6.2).

  SWARD makes it possible to in easily repeatable experiments benchmark static weapon allocation algorithms on a large number of problem instances. This allows for an increased understanding of which weapon allocation algorithms to be used for which kinds of air defense situations.

- Development of a particle swarm optimization algorithm and a genetic algorithm for static weapon allocation, and implementation of these as well as other weapon allocation algorithms into SWARD (section 4.2 and 6.2).

  Neither the use of genetic algorithms, particle swarm optimization, nor the other implemented algorithms for weapon allocation is completely novel, but the release of source code for the implementations allow for other researchers to test exactly the same algorithms. A typical problem is otherwise that a lot of details for weapon allocation algorithms are not revealed in published articles, making it hard for researchers to reimplement other researchers' algorithms.

- Evaluation of the real-time performance of all the implemented algorithms for static weapon allocation, where the results indicate what kind of algorithms that are suitable for air defense scenarios of various size (section 7.2).

  Effects of tight real-time constraints on tested algorithms for static weapon allocation are earlier not known, and the results from the experiments give new insights to which algorithms to use for particular types of air defense situations. These results can also be generalized to all kinds of resource allocation problems involving tight real-time constraints.

The third set of contributions concerns evaluation of TEWA systems. This kind of evaluation can also be used to evaluate the threat evaluation and weapon allocation algorithms being part of the TEWA system:

- Suggestion of a methodology for evaluating TEWA systems using simulations, in which a survivability metric is used (section 5.4).

  Although a simple idea, the measuring of survivability of defended assets and the weapon resource usage in computer-based simulations provides a way to systematically compare the performance of different TEWA system configurations. This kind of evaluation makes it possible for developers of TEWA systems to in an early stage evaluate the performance

of developed systems, and to find weaknesses on particular types of air defense scenarios.

- Development of the testbed STEWARD (System for Threat Evaluation and Weapon Allocation Research and Development), implementing the suggested methodology (section 6.1).

  STEWARD is a prototype testbed acting as a proof-of-concept of the proposed simulation-based methodology. STEWARD is connected to the simulation engine STAGE Scenario, in which air defense scenarios are created. The testbed demonstrates how the survivability and resource usage can be measured in dynamic scenarios.

Summarizing the contributions, literature surveys have been made, in which the open literature on threat evaluation and static weapon allocation are reviewed. By reviewing threat evaluation algorithms and weapon allocation algorithms together, a view on the interdependences between the threat evaluation and weapon allocation processes is provided. The review of threat evaluation algorithms has also resulted in a formalization of the threat evaluation process. Algorithms have been developed for threat evaluation as well as static weapon allocation. A number of weapon allocation algorithms (many of them metaheuristics inspired by biological phenomena) have been implemented into the developed open source testbed SWARD, in which it has been evaluated how they perform under real-time conditions. SWARD facilitates the use of standardized problem instances and repeatability, making it possible for various researchers to benchmark novel algorithms against reported experimental results for other algorithms. Evaluation of threat evaluation algorithms and TEWA systems is even harder than the evaluation of weapon allocation algorithms, since there are no objective function values that can be used for comparisons. In order to handle this problem, a survivability criterion is put forward, measuring the survivability of defended assets. By connecting a simulation engine to modules consisting of algorithms for threat evaluation and weapon allocation, it becomes possible to measure the effectiveness of the threat evaluation and weapon allocation algorithms. This opens up for more systematic comparisons of threat evaluation algorithms and TEWA systems, and this idea has been implemented into the testbed STEWARD.

1.4 Publications

The following publication list provides a short summary of the author's publications, and a description of how these contribute to the dissertation. The publications are divided into those of high relevance for the thesis, and those of lower relevance. Within these categories, the publications are listed in chronological ordering.

Publications of high relevance

1. **Johansson, F.** and Falkman G. (2008) A Bayesian network approach to threat evaluation with application to an air defense scenario. *Proceedings of the 11th International Conference on Information Fusion*, Cologne, Germany, July 2008.

   This paper gives a formal description of the threat evaluation process from an air defense perspective. Existing literature on threat evaluation is reviewed, resulting in a summary of parameters suggested for threat evaluation. We also review what kind of algorithms that have been suggested for threat evaluation in available literature. Based on the literature review, a Bayesian network for threat evaluation is presented. The suggested algorithm for threat evaluation is tested on a small scenario created in the scenario generator STAGE Scenario. The paper contributes to objective 1 and 2.

2. **Johansson, F.** and Falkman, G. (2008) A comparison between two approaches to threat evaluation in an air defense scenario. In Torra, V. and Narukawa, Y. (Eds.): Proceedings of the 5th International Conference on Modeling Decisions for Artificial Intelligence, *Lecture Notes in Artificial Intelligence 5285*, pp. 110–121, Springer-Verlag, Berlin Heidelberg.

   In this paper, the Bayesian network algorithm for threat evaluation presented earlier is compared to a fuzzy logic algorithm. The outputted target values from the algorithms on a basic air defense scenario are compared to each other, as well as to human expert knowledge. Moreover, more general characteristics of the algorithms are compared, such as smoothness of calculated target values and ability to handle uncertain data. It is also within this paper the problem of comparing threat evaluation algorithms is identified. The paper contributes to objective 2, and to some degree to objective 3 and 5.

3. **Johansson, F.** and Falkman, G. (2008) A survivability-based testbed for comparing threat evaluation algorithms. In Boström, H., Johansson, R., and van Laere, J. (Eds.): Proceedings of the 2nd Skövde Workshop on Information Fusion Topics, *Skövde Studies in Informatics 2008:1*, pp 22–24.

   The idea of using a survivability metric and computer-based simulations for evaluating threat evaluation algorithms is introduced in this work. Hence, this publication concerns objective 3.

4. **Johansson, F.** and Falkman, G. (2009) A testbed based on survivability for comparing threat evaluation algorithms. In S. Mott, J. F. Buford, G. Jakobson, M. J. Mendenhall (Eds.): *Proceedings of SPIE, Vol. 7352* (Intelligent Sensing, Situation Management, Impact Assessment, and Cyber-Sensing), Orlando, USA, April 2009.

Here, different ways to assess the performance of threat evaluation algorithms are discussed. In specific, an implemented testbed (a first version of STEWARD) is described, in which threat evaluation algorithms can be compared to each other, based on the survivability criterion introduced in the earlier publication. Survivability is measured by running the threat evaluation algorithms on simulated scenarios and using the resulting target values as input to a weapon allocation module. Depending on how well the threat evaluation is performed, the ability to eliminate the incoming targets will vary (and thereby also the survivability of the defended assets). Obtained results for two different threat evaluation algorithms (the Bayesian network and the fuzzy logic algorithm) are presented and analyzed. The paper contributes to objective 3, 4, and 5.

5. **Johansson F.** and Falkman, G. (2009) Performance evaluation of TEWA systems for improved decision support. In Torra, V., Narukawa, Y., and Inuiguchi, M. (Eds.): Proceedings of the 6th International Conference on Modeling Decisions for Artificial Intelligence, *Lecture Notes in Artificial Intelligence 5861*, pp. 205–216, Springer-Verlag, Berlin Heidelberg.

   The survivability criterion described above is in this publication extended upon to take resource usage into consideration. Experiments are run where we simulate a high-intensity scenario a large number of times to demonstrate the ability to compare the effectiveness of different TEWA system configurations. Simulations show that small changes in threshold settings can have a large effect on the resulting survivability. The paper contributes to objective 3, 4, and 5.

6. **Johansson F.** and Falkman G. (2009) An empirical investigation of the static weapon-target allocation problem. In Johansson, R., van Laere, J., and Mellin, J. (Eds.): Proceedings of the 3rd Skövde Workshop on Information Fusion Topics, *Skövde Studies in Informatics 2009:3*, pp. 63–67.

   Here, we introduce the real-time requirements on weapon allocation. We empirically investigate how large static weapon allocation problems can be before they become unsolvable in real-time using exhaustive search. A genetic algorithm to be used for problem sizes of larger size is developed. Moreover, the performance of the genetic algorithm is compared to that a of a simple random search algorithm, and it is shown that the genetic algorithm outperforms random search on the problem instances tested. The paper contributes to objective 2 and 5.

7. **Johansson F.** and Falkman G. (2010) A suite of metaheuristic algorithms for static weapon-target allocation. In Arabnia, H. R., Hashemi, R. R., and Solo, A. M. G. (Eds.): *Proceedings of the 2010 International Conference on Genetic and Evolutionary Methods*, pp. 132–138, CSREA Press.

Here, three metaheuristic algorithms for static target-based weapon allocation are presented: ant colony optimization, genetic algorithms, and particle swarm optimization. For the particle swarm optimization algorithm, problems such as how to construct discrete allocations from the continuous swarm updates, and how to handle premature convergence and particles flying outside the bounds of the search space are discussed. The algorithms are tested on problem sizes varying in size from ten targets and ten firing units up to thirty targets and thirty firing units. This paper contributes to objective 2 and 5.

8. **Johansson, F.** and Falkman, G. (2010) SWARD: System for weapon allocation research & development. *Proceedings of the 13th International Conference on Information Fusion*, Edinburgh, United Kingdom, July 2010.

   In this paper, the developed testbed SWARD is presented in detail. SWARD has been released under an open source (BSD) license and supports the use of standardized datasets, in order to allow for more systematic performance evaluation of static target-based and asset-based weapon allocation algorithms. The paper contributes to objective 4 and 5.

9. **Johansson, F.** and Falkman, G. (2010) Real-time allocation of defensive resources to rockets, artillery, and mortars. *Proceedings of the 13th International Conference on Information Fusion*, Edinburgh, United Kingdom, July 2010.

   This paper (receiving an honourable mention in the category of best student paper) presents results from experiments in which SWARD has been used to test the real-time performance of modified versions of the genetic algorithm and the particle swarm optimization algorithm for static asset-based weapon allocation. We also test to approximate the static asset-based problem with its target-based counterpart, and thereafter use the maximum marginal return algorithm on the resulting target-based optimization problem. The paper contributes to objective 4 and 5.

Publications of less relevance

1. **Johansson, F.** and Falkman, G. (2006) Implementation and integration of a Bayesian Network for prediction of tactical intention into a ground target simulator. *Proceedings of the 9th International Conference on Information Fusion*, Florence, Italy, July 2006.

   In this paper, a Bayesian network for prediction of enemy intent is developed, based on knowledge elicited from military experts at Swedish Army Combat School and the former Ericsson Microwave Systems (nowadays the business unit Electronic Defence Systems at Saab AB). The model is intended for ground combat, but is of interest here since it shares some of

the parameters used for threat evaluation in this thesis. Intent is together with capability the main factors for determining the level of threat posed by targets to defended assets. For this reason, intent recognition becomes important for the threat evaluation process.

2. Niklasson, L., Riveiro, M., **Johansson, F.**, Dahlbom, A., Falkman, G., Ziemke, T., Brax, C., Kronhamn, T., Smedberg, M., Warston, H. and Gustavsson, P. (2007) A Unified Situation Analysis Model for Human and Machine Situation Awareness. In Koschke, R., Herzog, O., Rödiger, K.-H., and Ronthaler, M. (Eds.): Trends, Solutions, Applications. Proceedings of SDF 2007. *Lecture Notes in Informatics P-109*, pp 105–110, Köllen Druck & Verlag.

   This is a joint publication, written by some of the members in our research group for getting a common view on information fusion, decision support, situation awareness, etc. We present $(SAM)^2$, a model for situation analysis unifying the technological view on information fusion given by the JDL model with the view given by Endsley on how human decision makers obtain situation awareness. The model is intended for highlighting important issues with semi-automated decision support systems, such as human-computer interaction and information exchange between different fusion levels. These problems are relevant for TEWA systems, in which there must be an interaction between human and machine. The paper contributes to the background of the thesis.

3. **Johansson, F.** and Falkman, G. (2007) Detection of vessel anomalies — a Bayesian network approach. *Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, Melbourne, Australia, December 2007.

   In this paper, a data mining approach based on Bayesian networks for detecting anomalous vessel behavior is presented. This approach is different from traditional statistical approaches for anomaly detection in that it allows for the incorporation of human expert knowledge and transparent models. Anomaly detection is relevant for threat assessment, since it can help focusing on entities deviating from normal behavior. Such techniques can therefore be used in a preprocessing step to a threat evaluation process, but also in military operations other than war.

4. Riveiro, M., **Johansson, F.**, Falkman G. and Ziemke, T (2008) Supporting Maritime Situation Awareness Using Self Organizing Maps and Gaussian Mixture Models. In Holst, A., Kreuger, P., and Funk, P. (Eds.): Tenth Scandinavian Conference on Artificial Intelligence. Proceedings of SCAI 2008. *Frontiers in Artificial Intelligence and Applications 173*, pp 84–91, IOS Press.

As the previous paper, this publication is concerned with the problem of identifying anomalous vessel behavior. The main difference between the two publications is in the choice of method for anomaly detection. In this paper, clustering of the training data is performed using a self organizing map. The weights of the individual model vectors in the self organizing map are together with the dispersion of training data around the model vectors used for creating a Gaussian mixture model, which in its turn is used for calculating the likelihood of new vessel observations.

5. Niklasson, L., Riveiro, M., **Johansson, F.**, Dahlbom, A., Falkman, G., Ziemke, T., Brax, C., Kronhamn, T., Smedberg, M., Warston, H. and Gustavsson, P. (2008) Extending the scope of Situation Analysis. *Proceedings of the 11th International Conference on Information Fusion*, Cologne, Germany, July 2008.

   This is another joint publication from the author's research group, and basically is an extended version of the $(SAM)^2$-paper described above.

1.5 Delimitations

It should be made clear that the optimization problem studied in this thesis is a *static* weapon allocation problem. Consequently, it is assumed that all weapon systems (firing units) should be allocated to targets simultaneously. In a real-world air defense situation, the decision of which weapon system to allocate to which target needs to be complemented with scheduling of *when* to engage a target. This is done in a separate scheduling process which is not part of this thesis. However, the fact that we here restrict our focus to static weapon allocation does not mean that the developed algorithms cannot be used for dynamic scenarios, since algorithms for static weapon allocation can be considered to be very important subroutines to solve the dynamic weapon allocation problem (Ahuja et al., 2007). Examples of how static weapon allocation algorithms can be used for dynamically evolving scenarios are shown in this thesis.

We are in the weapon allocation process only taking into account the allocation of so called hard-kill defensive resources (weapons used to intercept targets and actively destroy them through direct impact or explosive detonation) to targets. Hence, we are not dealing with soft-kill resources such as the use of chaff and radar jammers. Although very interesting and usable in their own right, allocation of such resources cannot easily be modeled using the formulations of static weapon allocation used in this thesis.

In the presented literature surveys, only the open (i.e. unclassified) literature is reviewed, due to obvious reasons. This means that better (classified) methodologies for evaluating TEWA systems and the threat evaluation and weapon algorithms being part of such systems may exist, but none that the author of this thesis is aware of.

1.6 Thesis Outline

This dissertation is organized as follows: after this introductory chapter, it continues with chapter 2, describing background information regarding air defense, information fusion, uncertainty management, and optimization that will be used throughout this thesis. In chapter 3, a formal presentation of the threat evaluation and weapon allocation processes is given. Moreover, a survey of the existing open literature on threat evaluation and weapon allocation is presented. Based on the findings, a number of algorithms for real-time threat evaluation and weapon allocation have been implemented. Detailed descriptions of these algorithms are presented in chapter 4. In chapter 5, the problem of performance evaluation of algorithms for threat evaluation and weapon allocation is discussed. Suitable methods for performance evaluation of threat evaluation and weapon allocation, as well as complete TEWA systems, are suggested, and these have been implemented into the testbeds STEWARD and SWARD, which are presented in chapter 6. The suggested algorithms have been implemented into the developed testbeds, and experiments in which the testbeds are used to compare the performance of the developed algorithms are presented in chapter 7. Finally, the thesis is concluded in chapter 8, together with a discussion on future work.

Chapter 2
Background

This chapter gives a background to the subject of this thesis and introduces fundamental concepts that are needed. In section 2.1, the information fusion domain is described, highlighting the need for supporting decision makers with the fusion of data and information from many heterogeneous sources in order to allow for good and timely decision making. Section 2.2 introduces the reader to the air defense domain, which is the core application for the work presented in this thesis. An important aspect of the air defense domain is that the sensor observations on which the decision making are based often are associated with a certain amount of uncertainty. For this reason, different approaches to uncertainty management that will be used throughout the thesis are introduced in section 2.3. A large portion of the work in the thesis concerns the allocation of firing units to threatening targets, which can be formalized as an optimization problem. Consequently, a brief introduction to optimization is given in in section 2.4, together with a presentation of different kinds of metaheuristic approaches to optimization.

2.1 Information Fusion

The technological development during the last decades has resulted in that our world is constantly flooded with data and information from various sensors. Information that can be highly relevant for decision makers are to be found in the bit streams of sensor observations, but it becomes increasingly difficult to find the pieces of useful information within the rest. This information age problem is illustrated in the following quote, due to Naisbitt (1982):

> "We are drowning in information but starved for knowledge."

In other words, instead of being able to make better and more informed decisions thanks to the extra information, it is often the case that decision makers become overloaded with information that potentially are of no use to the current decision to be made. Connected to this information overload problem is

the question of how to combine different pieces of information, stemming from different sources or points in time. Problems like these are dealt with in the research field of *information fusion*.

Information fusion, sometimes also referred to as data fusion[1], is often used to combine data from multiple sensors and related information in order to make inferences that may not be possible to do by using a single, independent sensor (Hall and Llinas, 2001). In most cases, these inferences are made in order to support decision making. The newest widely accepted definition of fusion, proposed in Steinberg et al. (1999) is as follows:

> "Data fusion is the process of combining data or information to estimate or predict entity states."

In many cases, the objective of information fusion is to estimate or predict physical states of entities over some past, current or future time period (Steinberg et al., 1999). Such traditional applications of information fusion are for example involving the tracking of the position of air targets (see e.g. Koch (2007)) and estimation of target identity (Schuck et al., 2009). The objective may however also be to estimate and predict more abstract states, such as relations among entities or the intention of entities (Johansson and Falkman, 2006; Bell et al., 2002). This is an example of what in thesis will be referred to as *high-level information fusion*. The above definition of data fusion is quite broad, in order to make it general and not restrict its application to the defense domain. However, a more defense-focused definition from the initial Joint Directors of Laboratories (JDL) data fusion lexicon will be used here, since it more clearly and explicitly specifies what is of interest within the work presented in this thesis, i.e. timely assessments of situations and threats. According to Liggins et al. (2009), data fusion is in that view:

> "a process dealing with the association, correlation, and combination of data and information from single and multiple sources to achieve refined position and identity estimates, and complete and timely assessments of situations and threats, and their significance. The process is characterized by continuous refinements of its estimates and assessments, and the evaluation of the need for additional sources, or modification of the process itself, to achieve improved results."

2.1.1 The JDL Model

By far the most used model for describing the fusion processes and functions is the model developed in 1988 by the data fusion group of the Joint Directors

[1]Traditionally, the name data fusion has been most used, but as the annual fusion conference as well as a number of well-known journals use the name information fusion, the latter has become more widely used in recent years. In this thesis, the terms are used interchangeably.

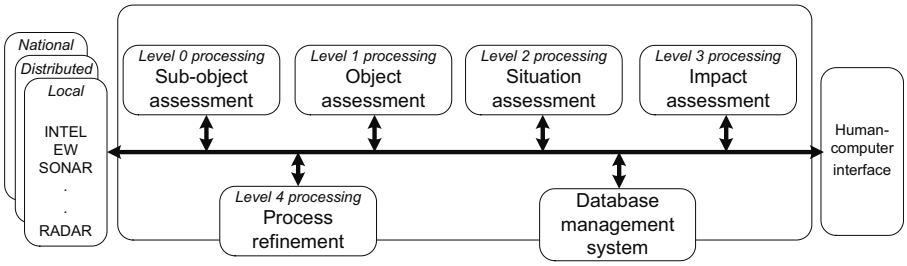


Figure 2.1: The JDL model (1998 year's revision, adapted from (Steinberg et al., 1999))

of Laboratories. This model is known as the JDL data fusion process model, or for short, the JDL model. The JDL model was developed in order to describe the different fusion processes that exist, and to assist in communication regarding fusion applications (Hall and McMullen, 2004). The model has been revised a number of times, and we will here use the terminology suggested in Steinberg et al. (1999), since it in the author's view makes most sense and is most commonly used.

The JDL model is divided into different levels, as can be seen in figure 2.1. It should be noted, however, that the creators of the model never intended to imply a process flow or a hierarchy between levels. A hierarchical sequence of the levels often take place, but there is nothing in the model as such that constrains the subprocesses to take place in sequential ordering (Liggins et al., 2009). A list of the fusion processes identified in the JDL model follows, together with a short description of each level:

- Level 0: Sub-Object Assessment

  At level 0, physical access is provided to the data obtained from sensors, either as raw bits or as a signal. The task on this level is mainly to preprocess data by correcting biases and standardizing the input before the data is fused with other data. The sub-object assessment processes are often performed within the individual sensors (Liggins et al., 2009). Examples of preprocessing that are dealt with on this level are signal processing, feature extraction, and filtering of data.

- Level 1: Object Assessment

  Level 1 fusion, referred to as object assessment (or sometimes entity assessment) is dealing with the combination of data from different sensor observations, in order to estimate and predict entity states (e.g. position, velocity, and other object attributes). Object assessment processing is often partitioned into three different problem categories: data correlation, state vector estimation (typically for target tracking), and estimation of a

target's identity (Hall and Steinberg, 2000; Hall and McMullen, 2004). Current approaches to object assessment are dominated by estimation techniques such as Kalman filters (Kalman, 1960), multiple-hypothesis tracking (Blackman, 2004), particle filters (Gustafsson et al., 2002; Arulampalam et al., 2001) and joint probabilistic data association filters (Vermaak et al., 2005) for target tracking problems, while the problem of identity estimation generally is performed using feature-based pattern recognition approaches, such as artificial neural networks (Hall and Steinberg, 2000). Identify estimation is further described in section 2.2.2.

- Level 2: Situation Assessment

  Situation assessment concerns the estimation and prediction of relations among entities and their relationships to the environment, in order to interpret the current situation. The process involves recognition of patterns, context-based reasoning, and understanding of temporal, spatial, causal, and functional relationships (Hall and Steinberg, 2000). Key functions in level 2 fusion are object aggregation, i.e. aggregation of entities into larger groups (Looney and Liang, 2003), force aggregation (Schubert, 2003), and multi-perspective assessment (i.e. to see a situation from both a neutral point of view, an own point of view, and from an adversarial point of view) (Hall and McMullen, 2004).

  Techniques for situation assessment are in general drawn from the field of pattern recognition (e.g. decision trees (Quinlan, 1986) and neural networks (Looney and Liang, 2003)) or the field of automated reasoning (e.g. blackboard systems, expert systems, and case-based reasoning) (Hall and McMullen, 2004).

- Level 3: Impact Assessment

  Level 3 fusion is according to Steinberg et al. (1999) concerned with the estimation and prediction of effects on situations of planned actions. Examples of this can be to draw inferences about potential threats (Roy et al., 2002), critical vulnerabilities (Falzon, 2006), and probable courses of action (Pawlowski et al., 2002; Bell et al., 2002). In fact, the original name for this level was threat assessment, showing the importance of the estimation of threat on this level. This name has however been changed to impact assessment in revisited versions of the JDL model, in order to broaden the concept to non-military domains (Liggins et al., 2009).

  A special challenge for level 3 processing is to determine enemy intent. This problem can be modeled using a variety of techniques when there exists a well-known enemy doctrine (see e.g. Suzić (2006) and Amori (1992)), but such doctrines are not often fully known (Fu et al., 2003). Another problem on this level is to model how the participants of a conflict situation interacts with each other, and how they adapt to each others actions. Research on game theory has been undertaken to address this

issue, but it still has shortcomings when it comes to applying it to realistic situation descriptions (Brynielsson, 2006; Hall and Steinberg, 2000). Other techniques for level 3 processing are drawn from the same field as for the level 2 processing, i.e. pattern recognition and automated reasoning (Hall and McMullen, 2004).

The threat evaluation and weapon allocation processes that are central for this thesis can according to Roux and van Vuuren (2007) both be classified as level 3 fusion processes.

- Level 4: Process Refinement

  Process refinement is a metaprocess monitoring the overall information fusion process to assess and improve the performance of the ongoing fusion via planning and control. That is, it seeks to optimize the ongoing fusion process in order to produce better fusion products, such as improved position, velocity, and identity estimates (Hall and McMullen, 2004). An important example of process refinement is sensor management (Xiong and Svensson, 2002), in which the use of sensors and sensor platforms is planned and managed, e.g. based on identified intelligence requirements.

According to Liggins et al. (2009), the object assessment level is the most mature area of information fusion. There exists a number of prototypes for the more immature levels of situation and impact assessment, however, only a few robust and operational systems (Liggins et al., 2009). It should however be mentioned that a lot of operational TEWA systems are deployed and operational already today. In the view of Steinberg (2009), level 3 fusion is a formally ill-defined and underdeveloped discipline of information fusion, but still very important for military and intelligence applications.

2.1.2 The OODA Loop

According to Nilsson (2008), the most accepted and most used decision making process model within the field of information fusion is the Observe-Orient-Decide-Act loop (often simply referred to as the OODA loop). This loop is illustrated in figure 2.2. Originally, the OODA loop was developed by the military strategist Colonel John Boyd in order to explain why American F-86 Sabre aviators were so successful in the Korean war, compared to their adversaries (Brehmer, 2005). In his analysis, Boyd found out that an explanation for the superior performance of the American aviators and their fighter aircrafts was that they were able to more quickly and accurately *Observe* their environment (detect enemy aircrafts), *Orient* themselves (by pointing their aircraft toward the enemy), *Decide* on what to do next, and to *Act* upon the taken decision (e.g. to press the trigger in order to shoot down their adversary). Hence, the proposed explanation for the superiority was that the American fighter pilots were able to get inside their opponent's decision cycle/OODA loop. Since its

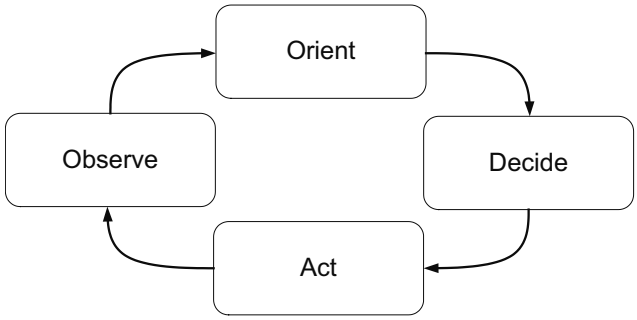


Figure 2.2: Illustration of Boyd's OODA loop.

original introduction, the simple OODA loop has become an important model in both business and military strategy (Richards, 2004). The importance of getting inside the adversary's OODA loop (i.e. to be able to go through the various stages in the loop more quickly and accurately than the opponent) is for example highlighted within maneuver warfare. In air defense situations, the speed and effectiveness of the execution of the defenders' OODA loop is of crucial importance, as discussed in section 2.2.

2.1.3 Decision Making and Situation Awareness

As identified earlier, it is often important for military decision makers to be able to make good decisions quickly. An important concept that has emerged in human decision making and which often is used within the field of information fusion is that of *situation awareness* (Endsley, 1988). According to Roux and van Vuuren (2007), situation awareness provides a primary basis for subsequent decision making in complex and dynamic environments. Endsley (1988) has defined situation awareness as:

> "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future."

This definition highlights that there are three levels of situation awareness, where the level of situation awareness depends on the level of information refinement (Brynielsson, 2006). Separating these levels, *perception* indicates a basic perception of important data, *comprehension* encompasses how people interpret the data by combining data and information and transferring it into knowledge, and *projection* denotes peoples' ability to predict future events and their implications (Endsley, 1995). Hence, the levels of situation awareness are closely related to levels 1–3 in the JDL model, as observed by e.g. Niklasson et al. (2008) and Lambert (2009).

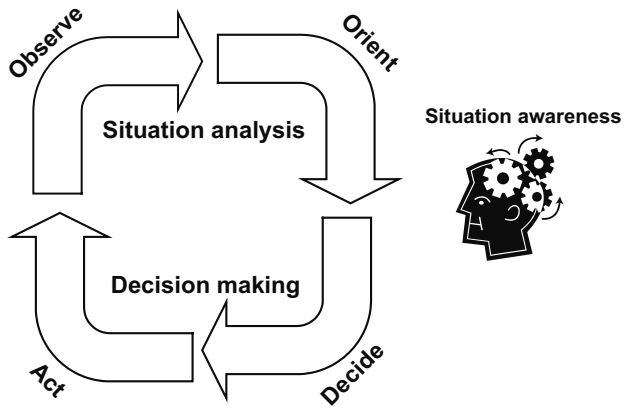


Figure 2.3: Illustration of the connection between situation analysis, situation awareness, and decision making (adapted from (Roy, 2001)).

Situation awareness is according to Roy (2001) the decision maker's mental model of the state of the environment and the main precursor to the decision making process. This is illustrated in figure 2.3, together with its relation to Boyd's OODA loop. Information fusion mainly concerns what in figure 2.3 is referred to as *situation analysis*, i.e. the Observe and Orient phases of the OODA loop. Hence, information fusion is fundamental for obtaining a high degree of situation awareness quickly and thereby make appropriate decisions. Information fusion is therefore considered to be the core technology underlying decision support systems used in applications in which large amounts of real-time information can be expected (Brynielsson, 2006). Nevertheless, to create a situation awareness is anything but easy. For the human decision maker it is challenging due to information overload and the uncertain and incomplete nature of the received data and information, but it is also challenging for computer-based systems due to the need of interpreting the information.

2.2 Air Defense

As should be clear from section 2.1, central themes for this thesis are information fusion and decision making. However, it is not decision making and information fusion in any domain that is of interest here, rather, the focus is solely on air defense applications. According to Joint Chiefs of Staff (2007), the concept of *air defense* includes:

> "defensive measures designed to destroy attacking aircraft or missiles in the atmosphere, or to nullify or reduce the effectiveness of such attack".

The importance of air defense is anything but new. Anti-balloon artillery was used already under the American Civil War, and the first aircraft ever downed in combat fell to ground fire as early as 1912[2] (Werrell, 2005). However, today's air defense weapon systems are far more technologically advanced than the modified artillery pieces used during the initial years. With the advent of radar came the possibility to detect aircrafts before they could be seen visually, allowing for much earlier warnings, especially by night. Later on, it also became possible to use radar for fire guidance. Moreover, old anti-aircraft guns have to a large extent been replaced by modern weapon systems such as surface-to-air missiles (SAMs), as well as high-speed Gatling guns[3] for terminal defense against fast-moving missiles. At the same time, the technological development on the attacker's side has been enormous too. Modern air defense units are not only facing threats such as aircrafts with high maneuverability in all three dimensions, but should also be able to defend against advanced weapons such as cruise missiles (CMs) and intercontinental ballistic missiles (ICBMs). Advances in technology have lead to increased speed, increased maneuverability, and decreasing radar cross section of threatening targets, which in their turn have lead to reductions in the available response time for the defense (Chalmers and da Ponte, 1995). In certain types of air defense, targets of interest may also be rockets, artillery, and mortars, commonly referred to as RAM. This type of air defense is often needed for protection of e.g. military bases against insurgent attacks in for example military international peace-keeping or peace-forcing operations. Targets of interest for more traditional air defense are comprised of two main elements: aircraft (manned and unmanned), and missiles (Joint Chiefs of Staff, 2007). As the technology has improved, and due to increases in the number of available missile systems, the number of countries with both ballistic missiles, cruise missiles, air-to-surface missiles (ASMs), and long-range missile capabilities has increased and is likely to increase even more in the near future (Joint Chiefs of Staff, 2007). Ballistic missiles are also often one of the weapons of choice for developing countries, due to their relatively cost-efficiency.

The most common task of the air defense is to protect so called *defended assets* against hostile *targets*. In order to protect the defended assets, *surveillance sensors* and *firing units*[4] are deployed. The defended assets are assets which have been identified as strategically important, and can for example be air bases, bridges, camps, command outposts, harbors, important supply lines, population centers, and power plants. It can also be the air defense units themselves, since these are highly valuable assets which often constitute the only protection of the other defended assets against air targets (Cheong, 1985). Hence,

[2]The aviator Constantin was during the Balkan War making an aerial study of the Turkish lines when killed by a rifle bullet fired from the ground (Roux and van Vuuren, 2008).

[3]This type of weapons are very common for the last layer of defense on naval crafts, on which they are referred to as CIWS (pronounced as sea-whiz), which is an abbreviation for close-in weapon system.

[4]The terms firing units and weapon systems are used interchangeably throughout the thesis.

the destruction of the defending air defense units make the remaining defended assets easy targets for attacking forces.

There are many different functions that must be handled in military decision support systems for air defense. The functions of (i) *target detection*, (ii) *target tracking*, (iii) *target identification*, (iv) *threat evaluation*, and (v) *weapons assignment* (here referred to as *weapon allocation*) are in Benaskeur et al. (2007) listed as being central for such systems. Additional to these functions, an important requirement on air defense systems is to have a good *man-machine interface* (MMI) that enhances the operators' situation awareness, since it in the end most often is the human rather than the machine that needs to make the decision of whether or not to engage a target. A view of how the various parts of an air defense system are functionally related to each other is shown in figure 2.4. Each function is described more thoroughly in section 2.2.1–2.2.5.

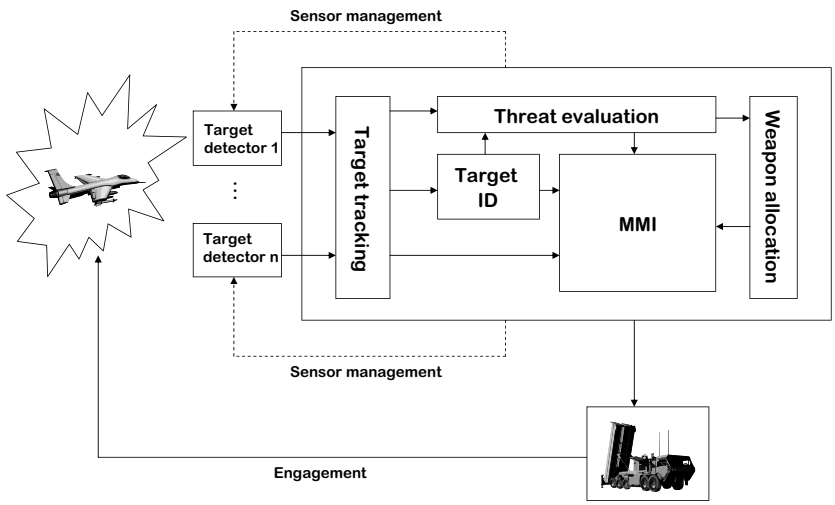


Figure 2.4: Functional overview of an air defense system.

2.2.1 Target Detection and Tracking

In order to be able to evaluate the threat posed by targets to defended assets and to identify suitable countermeasures, it is obvious that capabilities to detect and track targets are needed. The quality of the estimated target information

provided by available sensors play a crucial role for the outcome of the air defense processes.

When observing the environment in order to detect targets, there are many different kinds of sensors that can be used. The sensors that most often are used in air defense systems are surveillance radars (sensors sending out electromagnetic waves that are reflected by physical objects), but also thermal (e.g. infrared sensors) and optical sensors can be used (Roux and van Vuuren, 2008). Different sensors have different characteristics and there is typically a trade-off between the update rate, effective range, and range accuracy for a sensor. Typical specifications for 2D surveillance radars of various type are shown in table 2.1. As the name implies, a 2D radar works in two dimensions; *range* and *azimuth*. There are also 3D radars in use that in addition to range and azimuth information provide *elevation* information.

Table 2.1: Performance of a typical surveillance radar (adapted from Roux and van Vuuren (2008)).

	Update rate	Range	Range accuracy
Short range	$\leq$ 1 sec	0–50 km	5–20 m
Medium range	2–4 sec	50–200 km	20–100 m
Long range	10 sec	$\geq$ 200 km	100–500 m

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. Once a target has been detected, next step is to track it. Target tracking basically is about estimating the current position and kinematics of a target, as well as predicting these into the future. The target tracking is not dealt with in the work presented here, but it is important to understand that target estimates are imperfect, in that they are associated with uncertainty.

2.2.2 Target Identification

One important step in the target identification process 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 (Hall and McMullen, 2004) (e.g. fighter, bomber, or helicopter). Algorithms for accomplishing this are often based on pattern recognition techniques such as artificial neural networks (Hall and McMullen, 2004) or the use of techniques such as Bayesian networks and Dempster-Shafer theory (Schuck et al., 2009). Such an identification can also help in determining the *allegiance* of the target, i.e. a classification such as

Friend, *Neutral*, or *Hostile*[5]. To determine the friendly or hostile character of an unknown detected target is according to Joint Chiefs of Staff (2007) an essential air defense problem, since accurate and timely identification enhances the engagement of hostile aircrafts and missiles, as well as reduces the risk of fratricide.

Several different recognized methods can be used for identification. Examples of such methods are the use of so called *positive identification* and *procedural identification* (Joint Chiefs of Staff, 2007). What kind of method that is required for classification, among other factors, depends on the rules of engagement (ROE). The rules of engagement vary for different kinds of situations (e.g. whether the ongoing mission is part of a full-scale war or a peacekeeping operation). They may also vary between different kinds of air defense. As an example, the US Navy's rules of engagement used during hostilities dictate that aircrafts approaching aircraft carriers are assumed to be hostile, unless they are able to prove that they are friendly, or at least non-threatening (US Congress, Office of Technology Assessment, 1993). However, in the case of area air defense, the area defenders typically assume that an unknown aircraft might be friendly, unless there is positive evidence available showing that the target is indeed hostile (US Congress, Office of Technology Assessment, 1993). The concepts of positive identification and procedural identification are explained below:

- Positive Identification:

  The positive identification is an identity that is derived from visual observation and/or electronic systems, which possibly also are combined with other factors. What is meant by positive identification is that the identification is made based on the positive presence of evidence confirming the identity. As an example of this, a correct response to an identification friend or foe (IFF) interrogation system can be taken as positive evidence that the interrogated target is friendly. However, the lack of response to such an interrogation is not a positive evidence of the target being hostile. In many situations, a positive identification is required by the rules of engagements before an engagement decision is made (Joint Chiefs of Staff, 2007).

- Procedural Identification:

  A procedural identification separates air targets in the airspace by the use of factors such as geography, altitude, heading, and time (Joint Chiefs of Staff, 2007). Normally, a combination of positive and procedural identity (ID) is used to identify friendly and hostile targets. According to Liebhaber et al. (2002) and Smith et al. (2004), current air defense systems (such as the Aegis combat system) make use of kinematics such as course,

[5] A complete list of possible allegiances for use in NATO systems can be found in the NATO standard STANAG 1241.

speed, and altitude, as well as tactical data (e.g. type of radar) when determining a likely identity of aircraft contacts. The use of procedural ID can be advantageous for some missions and scenarios, but is generally not enough for engagement decisions (for this kind of decisions, a positive ID is normally required).

Target identification is not a central topic of this dissertation, but is very related to threat evaluation in that many of the parameters used for target identification also are used for threat evaluation. Furthermore, only target classified as $Hostile$ or $Unknown$ are usually becoming subject to threat evaluation (Roux and van Vuuren, 2007).

2.2.3 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* boils down to estimate the level of threat posed by detected air targets to defended assets (Roy et al., 2002), represented as threat values. Such threat values are aggregated into target values, specifying the total or average threat posed by a target (for a further discussion of threat and target values, see section 3.1). 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 decide or suggest which (if any) targets that should become subject for weapon allocation. Different target values will be calculated depending on which algorithm that is used for threat evaluation, and depending on the parameter settings. Parameters and algorithms that are suitable for threat evaluation are discussed in more depth in chapter 3. That chapter also provides a more formal description of the threat evaluation process.

2.2.4 Weapon Allocation

Weapon allocation can be defined as the reactive assignment of defensive weapon resources to engage or counter identified threats (Paradis et al., 2005). In order to protect their defended assets, the defense can use a number of different countermeasures. The two main classes of countermeasures that can be identified are so called *hard-kill* and *soft-kill* resources. A hard-kill weapon is used to intercept targets and actively destroy them through direct impact or explosive detonation, while soft-kill resources use different techniques to deceive or disorient the target so that it is not able to lock on the defended asset it is aimed for (Benaskeur et al., 2008). To exemplify these kind of defensive resources, there is on a typical frigate three kinds of hard-kill weapon systems (Beaumont, 2004): surface-to-air missiles, medium caliber guns, and close-in weapon systems. Typical characteristics for such weapon systems are shown in table 2.2. Examples of soft-kill resources are chaff decoy launchers (that are

Table 2.2: Performance of typical weapon systems (Benaskeur et al., 2007).

	Min range	Max range	Speed
Surface-to-air missiles	~ 1.5 km	~ 16 km	$\sim$ Mach 1
Medium caliber guns	~ 1 km	~ 5 km	~ 850 m/s
Close-in weapon systems	0 km	~ 2.5 km	~ 1200 m/s

used to seduce or distract targets) and radar jammers (which are used to perturb the enemy's radar) (Benaskeur et al., 2008). In this thesis, only hard-kill resources are taken into consideration. Hence, the question to be answered by the weapon allocation process studied here is which firing unit(s) to allocate to which target(s). The weapon allocation process is described in much more detail in chapter 3.

2.2.5 Man-Machine Interface

As stated in Hall and McMullen (2004):

> "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 in anything but rare circumstances (an exception being terminal defense against anti-ship missiles) 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, not least for legal issues. For this reason, it is important that the system can help the decision maker with achieving a situation awareness that can be used to make good and timely decisions. In order for the system to be useful for decision support, the operators' cognitive demands have to be taken into consideration. The use of air defense systems for decision support to human operators has in Morrison et al. (1997) been shown to increase the operators' situation awareness. However, human decision makers are unlikely to accept automated decisions without some explanation of the line of reasoning used for arriving at the decision, and air defense systems need to be designed with this in mind (Paradis et al., 2005).

In Liebhaber and Feher (2002a), guidelines for threat evaluation displays for threat evaluation and weapon allocation (TEWA) systems are discussed. According to their study, the threat evaluation part of a TEWA system should be able to:

- compute and display target values,
- support explanation-based reasoning (so that the decision maker understands the reason for the calculated target values),

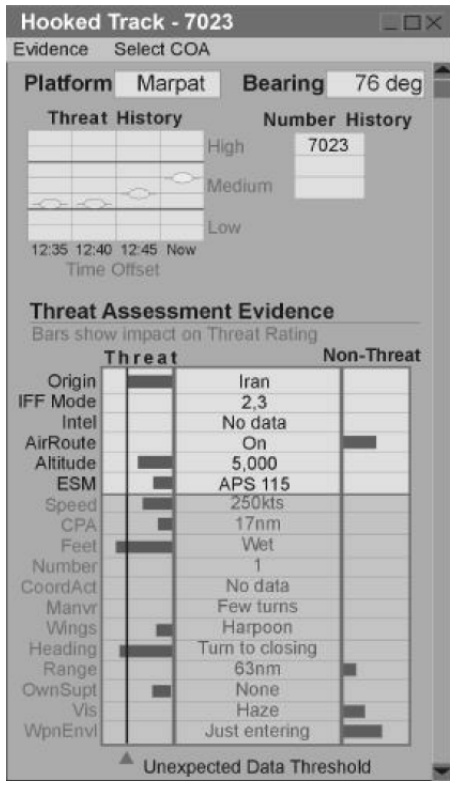


Figure 2.5: Graphical user interface displaying threat rating for a hooked track (from Liebhaber and Feher (2002a)).

- generate a target priority list that has a good match to human-generated lists.

It is also suggested that target values should be represented to the user using verbal descriptions, rather than through using numerical information, since numerical values imply a degree of precision that generally is not well anchored (Liebhaber and Feher, 2002a). It is according to their study also of value to be able to display a target value history, i.e. a view of how the target's estimated threat levels change over time. In figure 2.5 it is illustrated how many of these guidelines can be implemented in a suitable graphical user interface. Another important feature, suggested in Morrison et al. (1998) and Smith et al. (2004), is the ability to show a historical plot of the tracks. For cluttered airspaces, it is in St. John et al. (2004b,a) suggested that tactical displays should be decluttered through dimming the symbols of targets with low target values, in order to help the decision maker to obtain a good situation awareness.

The interaction between man and machine is of peripheral interest for this dissertation, but it should be noted that operators' trust in the TEWA system is crucial, and therefore the guidelines listed here need to be taken into account when designing such systems. Furthermore, these aspects make black-box techniques inappropriate for threat evaluation in most cases, since such techniques hinder the decision maker's understanding of the underlying reasoning mechanisms.

2.3 Uncertainty Management

Informally, *reasoning* is in this context a process in which information regarding some unobserved aspect of a situation is inferred from information regarding some observed parts of the situation (Bhatnagar and Kanal, 1992). A problem that is faced within many situations in which reasoning should be performed is that information is missing (i.e. that all relevant information is not available), and that all available information is not necessarily relevant, as discussed in section 2.1. Moreover, the available information is very often uncertain and imprecise (or even incorrect).

There are a number of different techniques available for reasoning and inference making in the presence of uncertain (and missing) information, e.g. certainty factors (Shortliffe and Buchanan, 1975), Dempster-Shafer theory (Shafer, 1976), Bayesian networks (Pearl, 1986), and fuzzy logic (Zadeh, 1983). In this section, the two latter techniques will be presented, since this is what have been used for making inferences in the work presented within this thesis. Section 2.3.1 introduces the concept of a Bayesian network, while the theory of fuzzy logic and fuzzy inference systems is introduced in section 2.3.2.

2.3.1 Bayesian Networks

A Bayesian network (sometimes also referred to as a belief network) is a probabilistic graphical model that characterizes a problem domain consisting of a set of random variables $\mathbf{U} = \{X_1,...,X_n\}$. These variables are in the Bayesian network represented as a set of corresponding nodes $\mathbf{V}$ in a directed acyclic graph $\mathcal{G} = (\mathbf{V},\mathbf{E})$, where the set of edges $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$ specifies (conditional) independence and dependence relations that hold between variables within the domain. Given the graph structure $\mathcal{G}$, a joint probability distribution P over $\mathbf{U}$ can be calculated from a set of local probability distributions associated with each node[6] X_i using the chain rule of Bayesian networks:

$$P(x_1,...,x_n) = \prod_{i=1}^{n} P(x_i|\mathbf{pa}_i), \tag{2.1}$$

[6]Since the nodes in $\mathcal{G}$ are in one-to-one correspondence with the random variables in $\mathbf{U}$, X_i is used to denote both variables and their corresponding nodes.

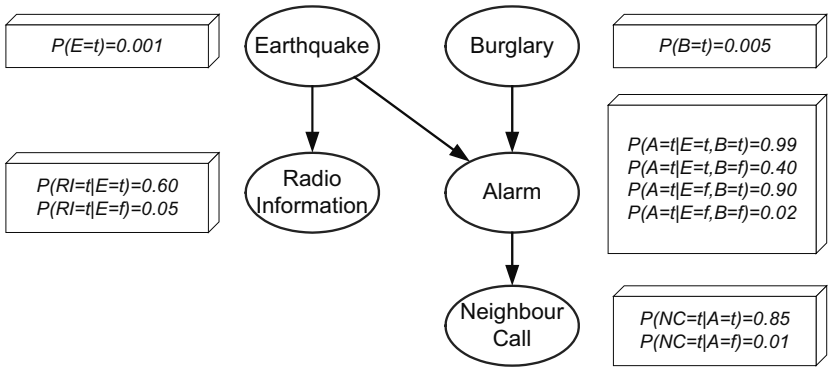


Figure 2.6: Example of a Bayesian network.

where the set of local probability distributions are the distributions in the product of equation 2.1 (with $\mathbf{pa}_i$ we refer to an assignment of values to the parent set $\mathbf{PA}_i$ of node X_i, while x_i is an assignment of a specific value to X_i). The joint probability distribution can be seen as a function assigning a number in the range $[0,1]$ to each possible combination of states of variables describing the domain. A strength of Bayesian networks is their ability to represent joint probability distributions in a compact manner, due to their encoding of conditional independences between different attributes in the domain.

To illustrate the use of Bayesian networks, consider a problem domain consisting of the random variables $Alarm$ (A), $Burglary$ (B), $Earthquake$ (E), $Neighbour\ Call$ (NC), and $Radio\ Information$ (RI). All of these variables can take on the values true (t) or false (f). In our example (which is a modified version of an example originally presented by Pearl (1988)), we have a burglar alarm installed in our house. The alarm is supposed to set off in case of a burglary, but there is a small probability that the alarm can malfunction. Moreover, there is a probability of false alarms, i.e. that the alarm sets off despite that there is no ongoing burglary. The main reason for such false alarms is in this example scenario that the alarm is set off due to minor earthquakes. Such earthquakes tend to be reported in the radio news, and if the alarm is sounding (due to an ongoing burglary, an earthquake, or any other reason) it is likely that we will receive a phone call from our neighbor, telling us about the sound of the alarm. This situation is illustrated in the Bayesian network shown in figure 2.6. Its graph structure shows how the random variables are related to each other (how to come up with an appropriate graph structure will be discussed later on), and the numbers shown are the ones needed for the local probability distributions, stored in so called conditional probability tables (CPTs). Note that some values are omitted, since these easily can be calculated by taking the complement to values found in the table (such as

$P(B = F) = 1 - P(B = T) = 0.995$). In the shown graph structure, the parent set of $Alarm$ is $\{Burglary, Earthquake\}$. Likewise, the parent set of $Burglary$ is $\emptyset$ since there are no incoming directed edges to $Burglary$. As an example of how joint probabilities can be obtained from the network, we can calculate the full joint probability $P(E = f,B = t,RI = f,A = t,NC = t)$. According to the chain rule of Bayesian networks, this can be calculated as:

$$P(E = f) \times \ldots \times P(NC = t|A = t) = 0.999 \times 0.005 \times 0.95 \times 0.9 \times 0.85 \approx 0.00363$$

Similarly, all other full joint probabilities for this domain can be obtained in the same manner from the Bayesian network. Hence, in this particular example, the Bayesian network encodes all $2^5 = 32$ full joint probabilities by storing only 10 values and the network structure. As we will see, all probabilities of interest for this domain can be calculated using this Bayesian network.

Conditional Independence and D-Separation

A central concept for the theory of Bayesian networks is that of conditional independence. For this reason, a formal definition of conditional independence is given in definition 2.1, analogous to the definition given in Jensen (2001).

Definition 2.1. *Let* $\mathbf{X} = \{X_1,...,X_n\}$, $\mathbf{Y} = \{Y_1,...,Y_m\}$ *and* $\mathbf{Z} = \{Z_1,...,Z_l\}$ *be three sets of variables. We say that* $\mathbf{X}$ *is conditionally independent of* $\mathbf{Y}$ *given* $\mathbf{Z}$ *(written as* $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$*) if* $P(\mathbf{X},\mathbf{Y}|\mathbf{Z}) = P(\mathbf{X}|\mathbf{Z})P(\mathbf{Y}|\mathbf{Z})$ *for every choice of values* $\mathbf{x}$, $\mathbf{y}$ *and* $\mathbf{z}$.

From equation 2.1 it follows that a random variable X_i is conditionally independent of the set of all its non-descendants, given the set of all its parents $\mathbf{PA}_i$. This is known as the *Markov condition*. Such a conditional independence that can be identified in our example scenario is that: $A \perp\!\!\!\perp RI|\{E,B\}$, i.e. given knowledge of the true state of the variables $Earthquake$ and $Burglary$, our belief in the alarm sounding will not change when obtaining information via the radio of whether there is an earthquake or not. By symmetry this also holds the other way around, i.e. information regarding the sound of the alarm will not change our belief in receiving radio information about earthquakes in our area, given that we know the state of $Earthquake$ and $Burglary$.

There are often more conditional independences encoded in a Bayesian network, since the Markov condition can entail other conditional independences. Which conditional independences that holds for a certain network structure $\mathcal{G}$ can be determined using the concept of *d-separation*. We will here explain the concept of d-separation using a definition given in Pearl (2000):

Definition 2.2. *A path* p *is d-separated (blocked) by a set of nodes* $\mathbf{Z}$ *if and only if*

1. p *contains a chain* $i \rightarrow m \rightarrow j$ *or a fork* $i \leftarrow m \rightarrow j$ *such that the middle node* m *is in* $\mathbf{Z}$*, or*

2. *p contains an inverted fork $i \rightarrow m \leftarrow j$ such that the middle node m is not in* **Z** *and such that no descendant of m is in* **Z**.

A set **Z** *d-separates a set* **X** *from a set* **Y** *if and only if* **Z** *blocks every path from a node in* **X** *to a node in* **Y**.

If two sets **X** and **Y** are d-separated by a third set **Z**, **X** and **Y** are conditionally independent given **Z**, i.e. $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{Z}$. In the definition above, the terms chain, fork, and inverted fork are used, but it is also common that these are referred to as serial connections (head-to-tail meetings), diverging connections (tail-to-tail meetings), and converging connections (head-to-head meetings), respectively. In our example Bayesian network, $B \rightarrow A \rightarrow NC$ is a chain, $RI \leftarrow E \rightarrow A$ is a fork, and $E \rightarrow A \leftarrow B$ is an inverted fork. Hence, some independences that can be identified using the concept of d-separation are that: $NC \perp\!\!\!\perp RI|E$ and $E \perp\!\!\!\perp B|\emptyset$, i.e. that our belief in receiving a call from our neighbor is not affected by radio information once we know the state of $Earthquake$, and that $Earthquake$ and $Burglary$ are independent when no other information is known.

Knowledge Elicitation

Creation of a Bayesian network involves three consecutive steps:

1. identification of important variables and their possible values,
2. identification of relationships between the identified variables,
3. elicitation of required quantitative conditional probability numbers.

The variables and their relationships correspond to the qualitative part of a Bayesian network and are expressed in a directed acyclic graph, as explained earlier. To identify the variables to be used is often quite straightforward. When deciding on which values a variable should be able to take on, it is important to remember that the states need to be *mutually exclusive* and *exhaustive*, i.e. that a variable cannot take on two or more states at the same time, and that all possible states are explicitly modeled. There are alternative ways to by hand create the graphical structure of a Bayesian network, however, the most convenient way is to draw an edge from a node X to a node Y if and only if X is a direct cause of Y, where the resulting graph is referred to as a *causal* directed acyclic graph (Neapolitan, 2003). If we mistakenly should draw an edge from a node X to a node Y when X in fact only has an indirect causal influence on Y through other random variables in the model, this unnecessarily increases the size of the Bayesian network (and thereby making it less informative), however, it is not too serious since the Markov condition still can be expected to hold (Neapolitan, 2003). It is worse if a causal edge is missed since it then might be conditional independence relations encoded in the Bayesian network that

does not hold in reality, violating the Markov condition. To elicit the qualitative part from domain experts is generally considered doable (nevertheless often an iterative labor intensive task) while it most often is the elicitation of the required (conditional) probabilities that is the main obstacle (Kjærulff and Madsen, 2007). A guide to much of the existing literature on knowledge elicitation for Bayesian networks is given in Druzdzel and van der Gaag (2000).

Probabilistic Inference in Bayesian Networks

Once a Bayesian network has been constructed, it can be used for probabilistic inference, i.e. to compute a posterior probability of interest conditional on available observations. This can mathematically be seen as given a set of observations (evidence) $\mathbf{z}$, a set of query variables $\mathbf{X}$, and a set $\mathbf{Y}$ including all variables except the variables in $\mathbf{X}$ and $\mathbf{Z}$ perform the computation of the posterior probability $P(\mathbf{X}|\mathbf{z})$. Given the full joint distribution encoded in the Bayesian network it is in theory easy to compute the answer by summing out the hidden (non-evidence) variables in $\mathbf{Y}$ as:

$$P(\mathbf{X}|\mathbf{z}) = \frac{\sum_{\mathbf{y}} P(\mathbf{X},\mathbf{Y},\mathbf{z})}{\sum_{\mathbf{x},\mathbf{y}} P(\mathbf{X},\mathbf{Y},\mathbf{z})}. \tag{2.2}$$

However, in this way the conditional independences encoded in the network are not made use of, and the calculations can become intractable for large networks. Since the graph structure $\mathcal{G}$ encodes conditional independences, these can be exploited to make probabilistic inference more efficient. Examples of such algorithms are the variable elimination algorithm (Zhang and Poole, 1996) and the junction tree algorithm (Lauritzen and Spiegelhalter, 1988). In the Bayesian network for threat evaluation presented in chapter 4.1.1, the junction tree algorithm (also known as the join tree algorithm) has been used. In this algorithm, the Bayesian network is converted into a secondary structure, known as a junction tree, on which probabilities of interest are computed (see Huang and Darwiche (1996) for a good presentation of the algorithm). Inference in general Bayesian networks has been shown to be **NP**-hard (Cooper, 1990). However, the computational complexity for the junction tree algorithm (as well as many other algorithms for probabilistic inference in Bayesian networks) can be estimated prior to the actual inference making. Moreover, in many practical applications a significant number of independences are present, making exact probabilistic inference tractable in most cases.

2.3.2 Fuzzy Logic

Fuzzy logic is a multi-valued logic developed by Zadeh in the 1960's with the objective of representing some types of approximate information that could not be represented by standard, crisp methods (Karray and de Silva, 2004). The use

of fuzzy logic builds upon the concept of *fuzzy sets*, where a fuzzy set can be seen as a generalization of a standard crisp set. In crisp set theory, members x of the universal set $\mathbf{X}$ are either members or nonmembers of a set $\mathbf{A} \subseteq \mathbf{X}$. That is, there is a discrimination function μ_A that for each $x \in \mathbf{X}$ assigns a value $\mu_A(x)$, such that:

$$\mu_A(x) = \begin{cases} 1 & \text{if and only if } x \in \mathbf{A}, \\ 0 & \text{if and only if } x \notin \mathbf{A}. \end{cases} \tag{2.3}$$

Within fuzzy set theory this is generalized, so that the values assigned to x fall within a specified range, indicating the element's membership grade in the (fuzzy) set in question (Klir and Folger, 1987). Larger values indicate a higher degree of membership, while lower values indicate a lower degree of membership. Hence, in fuzzy set theory there is a *membership function*

$$\mu_A : \mathbf{X} \rightarrow [0,1]. \tag{2.4}$$

It must be noted that membership grades are not to be confused with probabilities.

Operations on Fuzzy Sets

Classical crisp sets would not be very useful without operations such as intersection and union. The same holds true for fuzzy sets; we need the ability to combine different fuzzy sets by applying operations on them. Here, the basic operations of complement, union, and intersection on fuzzy sets are considered.

The complement $\mathbf{A}'$ to a fuzzy set $\mathbf{A} \in \mathbf{X}$ is in the work presented here a fuzzy set whose membership function is given by:

$$\mu_{A'}(x) = 1 - \mu_A(x), \forall x \in \mathbf{X}. \tag{2.5}$$

Other viable complement operators have been suggested in literature, but this is the most frequently used. In order for a function to be acceptable as an intersection operator, it must fulfill some basic requirements (cf. Kruse et al. (1994); Karray and de Silva (2004)). A function $\top$ fulfilling such requirements is known as a *t-norm*. A t-norm which is widely used is $\top_{min}$. Consider two fuzzy sets $\mathbf{A}$ and $\mathbf{B}$, belonging to the same universe $\mathbf{X}$. In this case, their intersection can be defined as:

$$\mu_{A \cap B}(x) = \top_{min}[\mu_A(x), \mu_B(x)], \forall x \in \mathbf{X}. \tag{2.6}$$

In the same manner, a *t-conorm* is a function $\perp$ that fulfills certain requirements for being a union operator. A widely used t-conorm is $\perp_{max}$, i.e.

$$\mu_{A \cup B}(x) = \perp_{max}[\mu_A(x), \mu_B(x)], \forall x \in \mathbf{X}. \tag{2.7}$$

This means that for intersection of two fuzzy sets, the minimum fuzzy set is taken as the result, while for union, the maximum fuzzy set is taken as the result. According to Kruse et al. (1994), these operators are very often used when working with fuzzy sets, and are comfortable to work with arithmetically.

Inference Making Using Fuzzy Inference Rules

Implications are in fuzzy knowledge-based systems (fuzzy inference systems) commonly represented by the use of fuzzy *if-then rules* (Karray and de Silva, 2004). Such rules are conditional statements as:

```
IF Speed == high AND Distance == close THEN Threat = high (1),
```

where the number within parentheses is the weight of the rule. A fuzzy inference system most often consists of several rules, and we will here describe the Mamdani fuzzy inference system (Mamdani, 1976) which is most commonly used. The Mamdani inference process consists of four steps:

1. Fuzzification
2. Rule evaluation
3. Aggregation
4. Defuzzification

As a first step in the Mamdani inference process, each crisp numerical input is *fuzzified* over all qualifying fuzzy sets required by the rules, i.e. the degree to which each part of the antecedent (the condition) of a rule is satisfied is determined using the specified membership functions. In next step in the inference process, the fuzzy intersection and/or union operators are applied to rules with more than one premise in their antecedent. In this way, it is made sure that all rules output a single value from their antecedent. As seen in the fuzzy inference rule above, the consequent of a rule is a fuzzy set. This fuzzy set is for each rule reshaped using the output from its antecedent on a specified implication method (e.g. the min-operator, truncating the fuzzy set specified by the consequent). Next, the output from all rules are combined into a single fuzzy set, using an aggregation operator. The aggregation operator that usually is used within fuzzy inference systems is the max-operator (Engelbrecht, 2002). Finally, the resulting aggregated fuzzy set is *defuzzified* into a single crisp (i.e. non-fuzzy) value, e.g. by calculating the centroid of the resulting fuzzy set. In this way, fuzzy inference rules can be used for making inferences.

To exemplify the use of fuzzy inference rules and the fuzzy inference process, consider a domain where we would like to determine how much tip to give, based on the quality of the service and the quality of the food. Assume that we come up with the following fuzzy inference rules:

```
IF Service == poor OR Food == rancid THEN Tip = low
IF Service == good THEN Tip = medium
IF Service == excellent OR Food == delicious THEN Tip = high.
```

In this example, no rule weights are assigned (i.e. they all have the default weight 1.0). Before these rules can be used, the membership functions for $Service$,

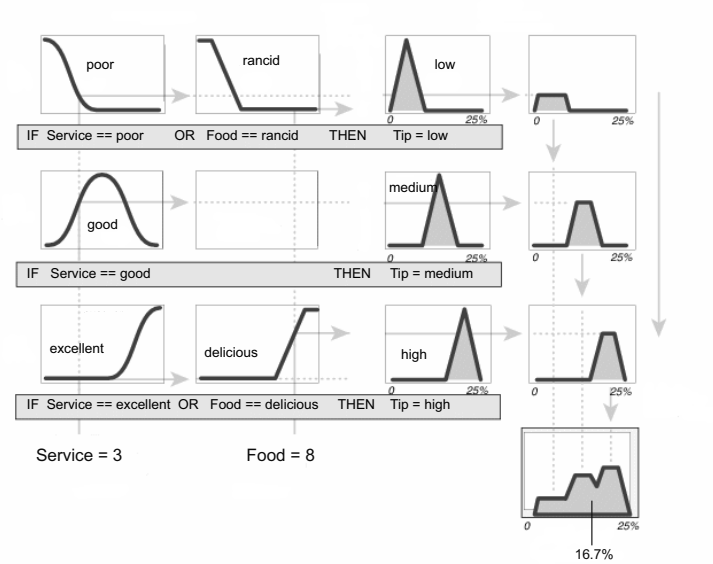


Figure 2.7: Example of inference in a fuzzy inference system (adapted from MathWorks (2010)).

$Food$, and Tip have to be specified (these can be seen in figure 2.7). The quality of the service and the quality of the food are both judged on a scale from 0 to 10, and based on these inputs, we would like our small fuzzy inference system to decide on the tip (in percentage of the cost of the meal).

Figure 2.7 illustrates the used membership functions and the four different steps in the fuzzy inference process, going from fuzzification of the crisp numerical input ($Service = 3$, $Food = 8$) to the defuzzified output $Tip = 16.7\%$ obtained by calculating the centroid from the resulting fuzzy set.

2.4 Optimization

An optimization problem can be described as the maximization or minimization of a function of a number of *decision variables* $x_1, x_2, \ldots, x_n$, where the function to be maximized or minimized is referred to as the *objective function*. The *objective function value* F is a scalar quantity derived from the decision variables as:

$$F = f(\mathbf{x}) = f(x_1, x_2, \ldots, x_n). \tag{2.8}$$

The decision variables are often subject to restrictions, referred to as *constraints*. An assignment of the decision variables fulfilling all the constraints is known as a *feasible solution*. In this sense, optimization can be seen as the

problem of choosing the best feasible solution from a set of available alternatives. A general constrained optimization problem can be written as (Antoniou and Lu, 2007):

$$\min F = f(\mathbf{x}) \tag{2.9}$$

subject to:

$$\begin{aligned} a_i(\mathbf{x}) &= 0, \text{ for } i = 1,2,\ldots,p \\ c_j(\mathbf{x}) &\geq 0, \text{ for } j = 1,2,\ldots,q \end{aligned} \tag{2.10}$$

In the case of the optimization problem being a maximization problem, such a problem can be transformed into a minimization problem as:

$$\max f(\mathbf{x}) = -\min[-f(\mathbf{x})]. \tag{2.11}$$

An important well-known class of optimization problems is one where we would like to minimize or maximize a linear function of decision variables, and where all constraints are linear equations or linear inequalities. Such a problem is known as a *linear programming (LP) problem*. Linear programming problems have the useful properties that the feasible region for such a problem is a convex set and that there must be an extreme point of the feasible region that is optimal, given that the linear programming problem has an optimal solution (Winston, 1997). Hence, we only need to search the finite set of extreme points of the feasible region when searching for an optimal solution to a linear programming problem. This is taken advantage of in the well-known simplex algorithm invented by Dantzig, which can be used to rather quickly solve linear programming problems of thousands of decision variables and constraints.

An optimization problem in which some or all of the decision variables are restricted to be integral is known as an *integer programming (IP) problem* (Winston, 1997). The two formulations of the static weapon allocation problem introduced in section 3.1 are examples of this class of problems. *Combinatorial optimization* concerns the search of finding solutions in discrete problem spaces (Papadimitriou and Steiglitz, 1998), and hence, integer programming problems are also combinatorial optimization problems. Most combinatorial optimization problems are hard to solve exactly, since all known exact approaches have worst-case computing times that grow exponentially with the size of the problem instance at hand (Maffioli and Galbiati, 2000). It is strongly believed that it is not possible to solve such **NP**-hard problems in polynomially bounded computation time (Dorigo and Stützle, 2004), even though this still has not been proved[7]. Due to the computational complexity, this kind of problems are often solved using heuristic algorithms, i.e. approximate algorithms that are not guaranteed to return the optimal solution. Many such algorithms

[7]The question of whether $\mathbf{P} = \mathbf{NP}$ or not has been called one of the seven most important open questions in mathematics and the Clay Math Institute has offered a million-dollar prize for a proof determining whether $\mathbf{P} = \mathbf{NP}$ or not (Fortnow, 2009).

are inspired by biological phenomena. Optimization problems occurring in nature (such as adaptation to changes in the environment) are in practice differing significantly from scientific optimization problems. Nevertheless, this does not mean that it is meaningless to apply some of the clever mechanisms that can be found in nature in various optimization algorithms. As stated in Wahde (2008):

> "[...] in view of the amazing complexity of biological organisms, and the perplexing ingenuity of some of the designs found by evolution, there is sufficient motivation for developing optimization algorithms based on natural evolution."

Within the work presented in this thesis, three metaheuristic optimization algorithms (i.e. high-level strategies which through adaptation can be applied to different optimization problems) have been used. For this reason, the following sections introduce the reader to genetic algorithms (section 2.4.1), ant colony optimization (section 2.4.2), and particle swarm optimization algorithms (section 2.4.3). More detailed descriptions of the algorithms are provided in section 4.2.

2.4.1 Genetic Algorithms

The concept of *genetic algorithms* was first introduced by Holland (1975). The technique is an example of a so called evolutionary algorithm, which is inspired by the processes of natural selection and evolution.

Initially, a genetic algorithm starts out with generating a (random) population, consisting of a fixed size of individuals. An individual is encoded using a chromosome (typically represented as a string) (Karray and de Silva, 2004), consisting of a number of variables (genes), where each individual (also referred to as phenotype) represents a candidate solution to the optimization problem at hand. The range of values a gene can take on is within the theory of genetic algorithms referred to as its alleles. A specific optimization problem can often be encoded using different chromosome representations, e.g. binary or real-number encoding schemes, and the choice of representation can have a large effect on the success of applying a genetic algorithm (Wahde, 2008).

In order to evaluate how good a specific solution is, a fitness function is used to measure its fitness. In the kind of optimization problems of interest here, the fitness function is typically designed to be the objective function of the optimization problem for maximization problems, or its inverse for minimization problems.

There are three important genetic operators that need to be designed when implementing a genetic algorithm: selection, crossover, and mutation. The selection operator is used to select the individuals that are to participate in the reproduction process which gives birth to the next generation of individuals. Generally, the used selection operator is designed to probabilistically select good solutions and remove bad solutions, based on their fitness function (Karray and

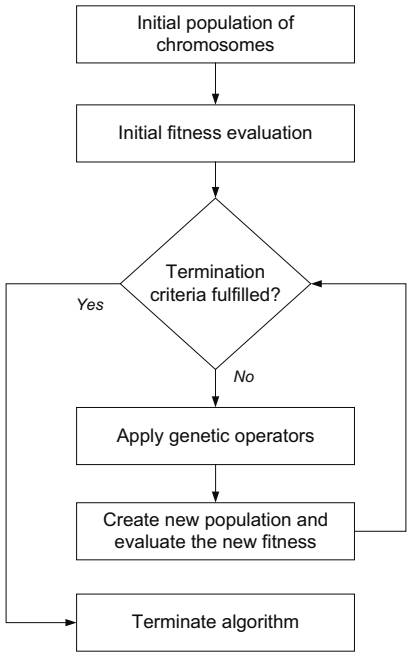


Figure 2.8: Illustration of the overall process flow of a standard genetic algorithm (adapted from Karray and de Silva (2004)).

de Silva, 2004). The two most common selection algorithms are roulette-wheel selection and tournament selection (Wahde, 2008). In roulette-wheel selection, the individuals are selected based on a fitness-proportional procedure, while tournament selection picks two (or more) individuals at random, selecting the best among them with a certain probability. The crossover operator is used to achieve genetic recombination, in which the genes of two parents are combined in order to form the chromosome of a new individual. The crossover can be designed in different ways, but only creates new orderings of already existing genes. On the other hand, the mutation operator can possibly change a gene completely, since it often is designed to randomly change one or more parts of the string representing the chromosome (Karray and de Silva, 2004). The process flow of a typical genetic algorithm is illustrated in figure 2.8. When the algorithm terminates, it returns the best solution found during the search.

Genetic algorithms have been used extensively throughout literature. To mention just a few examples, it has in the field of information fusion earlier been used for optimizing air combat tactics (Mulgund et al., 1998) and for generation and evaluation of potential enemy courses of actions for wargaming (Gonsalves et al., 2003).

2.4.2 Ant Colony Optimization

Ant colony optimization was originally proposed by Dorigo (1992). An inspiring source of the ant colony optimization metaheuristic is the behavior of real ants searching for food. In short, ants make use of a chemical substance called *pheromone*, which is deposited on the ground. Initially, ants tend to wander around randomly in their search for food. If an ant finds food, it starts laying down trails of pheromone along its path back to the colony. When an ant finds a path with pheromone, there is a high probability that it will stop wander randomly and instead follow the pheromone trail to the food source (Engelbrecht, 2002). Upon finding the food, the ants return to their colony, strengthening the pheromone trail. However, pheromone evaporates over time, so that unless the trail is reinforced by any ants, it will eventually disappear. The local communication mediated by the pheromone information is a very important component of ant behavior (Wahde, 2008). Using this kind of behavior, ants are successful in cooperatively finding short(est) paths between their colony and food sources.

In order to use ant colony optimization, the optimization problem has to be represented as a graph problem in which the optimal path is searched for in a graph $\mathcal{G}$ (Wahde, 2008). By releasing artificial ants onto the graph $\mathcal{G}$, and letting those select paths guided by the use of probabilistic rules, the full path traversed by an ant can represent a feasible solution to the original optimization problem. As the ants move around in the graph, pheromone is deposited on the graph edges of the used paths, so that the artificial ants implement an algorithmic counterpart to the local communication used by real ants. In this way, the artificial ants cooperatively search for good solutions to the problem via the exchange of pheromone information. The probabilistic rule is usually implemented so as to favor short edges (edges with low costs) with large amounts of pheromone (Dorigo and Gambardella, 1997) (i.e. exploitation) but also allow for biased exploration of other paths in the graph.

Ant colony optimization has in the information fusion domain earlier been applied to problems such as sensor scheduling (Schrage and Gonsalves, 2003) and determination of the enemy's possible avenues of approach (Svenson and Sidenbladh, 2003).

2.4.3 Particle Swarm Optimization

Particle swarm optimization (PSO) is a global optimization approach that was introduced by Kennedy and Eberhart (1995). Inspiration to the technique comes from the behavior of bird flocking (Kennedy and Eberhart, 1995), where so called *particles* form a *swarm*. Like birds, the particles move in a multidimensional search space, adjusting their associated position and velocity (Engelbrecht, 2002). The position of a particle corresponds directly to a candidate solution to the optimization problem being modeled.

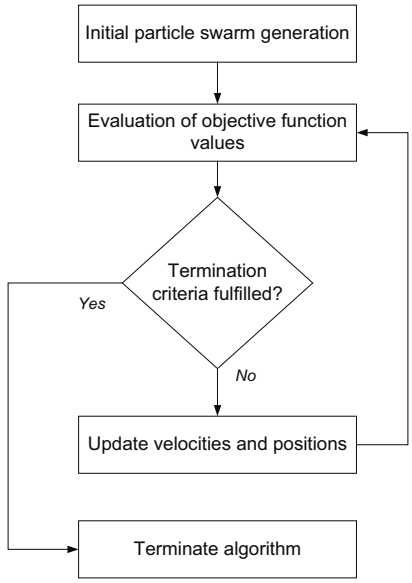


Figure 2.9: Illustration of the overall process flow of a standard particle swarm optimization algorithm.

An important property of many instances of swarm behavior that can be observed in nature is that there is no apparent leader that the other swarm members follow. Hence, it is believed that the swarm behavior is a result of local interactions only (Wahde, 2008). This property is also central in algorithmic implementations of particle swarm optimization. All particles in the swarm are initially given (random) positions and velocities. As described in figure 2.9, next step is to evaluate the objective function values of the feasible solutions corresponding to the particles' positions in the search space. Based on the objective function values, the velocities (and thereby also the positions) are updated. The mechanism for determining the changes in velocity is dependent on the performance of both the particle itself and the other particles in the swarm.

The original version of particle swarm optimization was intended for optimization problems with continuous decision variables, but there are also examples of where the use of particle swarm optimization has been applied to discrete optimization problems. Particle swarm optimization has earlier been used for e.g. sensor management in a biometric security system (Veeramachaneni et al., 2004) and assignment of unmanned aerial vehicles (UAVs) to tasks (Pan et al., 2009).

Chapter 3
Threat Evaluation and Weapon Allocation

In this chapter, we will provide formalizations of the threat evaluation and weapon allocation processes. The provided threat evaluation description is the author's attempt to formalize the previously quite poorly defined process of threat evaluation. There are surprisingly few articles that have been written on the topic of threat evaluation. An explanation to this is probably that the inner workings of threat evaluation systems typically are kept secret due to the high cost and extensive research required to develop them (Roux and van Vuuren, 2007), and to avoid that weaknesses in developed threat evaluation algorithms are exploited. Since there are no previous compilations of existing algorithms for threat evaluation, a review of the existing literature on threat evaluation is presented. As we will see, there exists considerably more work related to the problem of weapon allocation. However, a review of the existing literature on weapon allocation is presented as well, since many new algorithms have been proposed in later years, for which no previous surveys exist.

3.1 Formalization

The formalization of the threat evaluation process presented in section 3.1.1 is based upon work presented in Johansson and Falkman (2008a) and Johansson and Falkman (2009a). The static target-based weapon allocation problem described in section 3.1.2 was according to den Broeder et al. (1959) and Manne (1958) originally introduced by Flood. It was formally defined by Manne (1958). Its asset-based counterpart which also is presented in section 3.1.2 has earlier been described in e.g. Hosein (1990) and Hosein and Athans (1990b).

3.1.1 Threat Evaluation

The purpose of the threat evaluation process is to determine the level of threat posed by targets to defended assets (Paradis et al., 2005). Threat evaluation is a continuous process, since the sensed environment is constantly changing. According to Roux and van Vuuren (2007), this threat evaluation process is in the context of air defense poorly defined. In the following, this process will be described more formally, but first, we need to introduce some notation:

- $\mathbf{T} = \{T_1, \ldots, T_n\} \triangleq$ the set of targets detected by our sensors.
- $\mathbf{A} = \{A_1, \ldots, A_m\} \triangleq$ the set of defended assets we would like to protect.
- $V_{ij} \triangleq$ the threat value of the target-defended asset pair (T_i, A_j), $T_i \in \mathbf{T}, A_j \in \mathbf{A}$.
- $V_i \triangleq$ the aggregated target value for target $T_i \in \mathbf{T}$.

Moreover, $|\mathbf{T}|$ will in the following be used to denote the number of targets, while $|\mathbf{A}|$ will be used to denote the number of defended assets. Using this notation, we can formalize the threat evaluation problem as the creation of a function f that maps each target-defended asset pair (T_i, A_j) into a *threat value* V_{ij} (where $T_i \in \mathbf{T}$ and $A_j \in \mathbf{A}$). Such a threat value can be defined as the level of threat posed by target T_i to the defended asset A_j. For convenience, we determine the range of V_{ij} to be the interval [0,1], where 0 is the lowest possible threat value, and 1 is the highest possible threat value. Hence, we would like to design a function f, such that:

$$f : \mathbf{T} \times \mathbf{A} \rightarrow [0,1]. \tag{3.1}$$

The function f can for example be based upon information such as the distance between the target T_i and the defended asset A_j. A more thorough investigation of which parameters that are suitable for determining the threat value is presented in section 3.2.1. The information used for estimating threat values are usually acquired using various heterogeneous and imperfect sources, and are subject to uncertainty and deception (Benaskeur et al., 2008).

Threat evaluation processes can be divided into two separate classes, depending on the size $|\mathbf{A}|$ of the set of defended assets. The case where the task of the air defense is to protect a single defended asset, i.e. $|\mathbf{A}| = 1$, is known as *point defense* (Roux and van Vuuren, 2007), and is often used in air defense threat evaluation and weapon allocation systems on board naval crafts. In this thesis we also take into account the more complex case where a larger set of defended assets are to be defended, i.e. $|\mathbf{A}| > 1$. This is known as *area defense* (Roux and van Vuuren, 2007), and is often used in a ground based air defense context. Examples of defended assets in such a context are air bases, fuel depots, radars, and blue force units (Roux and van Vuuren, 2008).

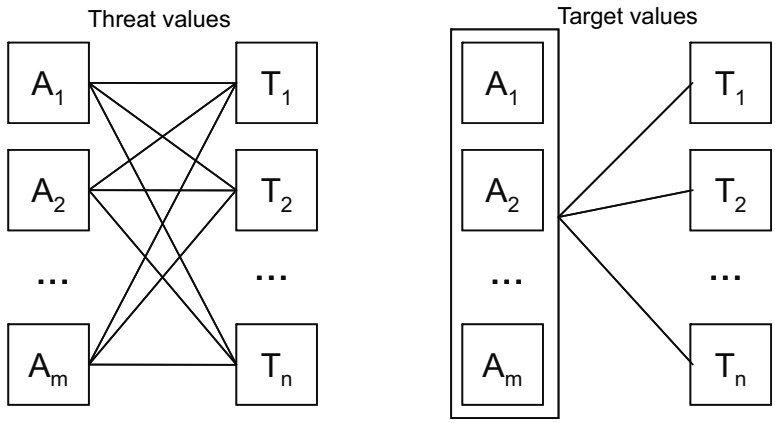


Figure 3.1: Illustration of the difference between threat values and target values.

Once threat values V_{ij} have been calculated for target-defended asset pairs using some specific algorithm, we often would like to process these into a single target value V_i for each target T_i (illustrated in figure 3.1). As we will see, this is required information for one type of weapon allocation (target-based weapon allocation). Moreover, to aggregate threat values into a combined target value is also very useful for human operators in stressful situations. This can be accomplished in different ways. A straightforward way is to calculate the target value as the average from target T_i's threat values. A problem with this approach is that it does not take differences among defended assets into account. In many scenarios, different defended assets will be of varying value to the defenders. Hence, different defended assets can be assigned different protection values (cf. the development of defended asset lists in the Armed Forces of the United States (Joint Chiefs of Staff, 2007)). We therefore introduce a user-defined parameter $\omega_j \in [0,1]$, denoting the protection value of a defended asset A_j. Such a value can be thought of as the weight of the defended asset, and is, as we soon will see, important input to one of the two static weapon allocation problems studied in this dissertation. The protection value can be used to calculate target value as the weighted average:

$$V_i = \frac{\sum_{j=1}^{|\mathbf{A}|} V_{ij}\omega_j}{\sum_{j=1}^{|\mathbf{A}|} \omega_j}, \tag{3.2}$$

where $|\mathbf{A}|$ is the total number of defended assets. This kind of weighted average can however result in inappropriate target values in some kind of situations, e.g. when there is a target representing a large threat to a single defended asset while there are a lot of defended assets to which the target does not constitute any

threat. Using such a weighted average can then result in a low (averaged) target value, which is not always wanted. An alternative, "safer" way to compute the resulting target value is to simply go through all threat values for a specific target T_i and pick the largest one. If we also take the protection value into account, we end up with:

$$V_i = \max\left(V_{ij}\omega_j\right), \quad j = 1, \ldots, |\mathbf{A}|. \tag{3.3}$$

Obviously, the aggregated target value is more convenient to work with than a set of threat values for each target. However, sometimes it can be very useful to have access to the "original" threat values (e.g. when an operator would like to know which defended asset that is threatened by a certain target). Therefore, the threat values often have to be stored as well (note that the target value calculation step is not necessary for the single platform perspective, i.e. when $|\mathbf{A}| = 1$, since it for that case only exists one threat value per target, and the defended asset always will be the same).

Once the target values for all targets have been calculated, a *prioritized threat list* can be created, i.e. a ranking from the target with the highest target value to the target with the lowest target value. This list can be used for giving a human operator feedback on which targets to give most attention first. Moreover, the calculated target values can in more automated systems be used to decide which targets that should be subject to weapon allocation, and also constitute an important parameter for which firing unit to allocate to which target.

3.1.2 Weapon Allocation

Informally, weapon allocation (often also referred to as weapon assignment or weapon-target allocation) can according to Paradis et al. (2005) be defined as:

> "the reactive assignment of weapon systems to engage or counter identified threats."

More formally, the weapon allocation problem can be stated as an optimization problem in which we aim to allocate firing units so as to minimize the expected total target value of the targets. Alternatively, the weapon allocation problem can be stated as an optimization problem where the objective becomes 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 (Hosein, 1990).

When presenting the two versions of the static weapon allocation problem that will be used throughout this thesis, the following notation will be used, in addition to the notation introduced in section 3.1.1:

- $\mathbf{W} = \{W_1, \ldots, W_p\} \triangleq$ the set of defensive firing units.

- $\mathbf{G}_j \triangleq$ the set of targets aimed for defended asset $A_j \in \mathbf{A}$.
- $P_{ik} \triangleq$ estimated kill probability, i.e. probability that weapon $W_k \in \mathbf{W}$ destroys target $T_i \in \mathbf{T}$, if assigned to it.
- $\pi_i \triangleq$ estimated probability that target $T_i \in \mathbf{T}$ destroys the asset it is aimed for.
- $\omega_j \triangleq$ protection value of defended asset $A_j \in \mathbf{A}$.
- $X_{ik} = \begin{cases} 1 & \text{if weapon } W_k \in \mathbf{W} \text{ is assigned to target } T_i \in \mathbf{T}, \\ 0 & \text{otherwise.} \end{cases}$

Furthermore, $|\mathbf{W}|$ will be used to denote the number of firing units.

The Static Target-Based Weapon Allocation Problem

Given an air defense situation consisting of $|\mathbf{W}|$ firing units and $|\mathbf{T}|$ targets, the static target-based weapon allocation problem can mathematically be formalized as the nonlinear integer programming problem:

$$\min F = \sum_{i=1}^{|\mathbf{T}|} V_i \prod_{k=1}^{|\mathbf{W}|} (1 - P_{ik})^{X_{ik}}, \tag{3.4}$$

subject to:

$$\begin{gathered} \sum_{i=1}^{|\mathbf{T}|} X_{ik} = 1, \quad \forall k, \\ X_{ik} \in \{0,1\}, \quad \forall i \forall k. \end{gathered} \tag{3.5}$$

The optimization of the objective function given in equation 3.4, subject to the constraints given in equation 3.5, can be interpreted as finding the allocation of firing units to targets that minimizes the expected total value of surviving targets (since the product $\prod (1 - P_{ik})^{X_{ik}}$ represents the expected probability of survival for target T_i). For this reason, this type of objective function can be referred to as weighted subtractive defense or target-based weapon allocation (Hosein, 1990). Since target values by definition never can be negative, the objective function value F is bounded below by 0. Moreover, it is bounded above by $\sum_{i=1}^{|\mathbf{T}|} V_i$ (since the survival probability of target T_i never can exceed the value 1.0).

A few assumptions are made in the static target-based weapon allocation formulation. First of all, all firing units have to be assigned to targets, as indicated in the first constraint given in equation 3.5. A firing unit therefore need to be allocated to exactly one target. It shall however be noted that many firing

units can be allocated to the same target in order to decrease the probability of the target surviving the attack. This type of allocations are known as Salvo attacks (Hosein and Athans, 1990a; Eckler and Burr, 1972). Moreover, all 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. 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). Without the last assumption, the geometry of the problem must be taken into account, creating an extremely complex problem. Despite these assumptions, the static target-based weapon allocation problem is anything but easy to solve. The **NP**-completeness of the problem was established by Lloyd and Witsenhausen (1986), indicating the hardness of the problem. This is problematic since we often have to deal with large-scale problems, i.e. air defense situations consisting of a large number of targets and/or firing units.

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

$$\mathbf{X} = \begin{bmatrix} X_{11} & X_{12} & \dots & X_{1|\mathbf{W}|} \\ X_{21} & X_{22} & \dots & X_{2|\mathbf{W}|} \\ \vdots & \vdots & X_{ik} & \vdots \\ X_{|\mathbf{T}|1} & X_{|\mathbf{T}|2} & \dots & X_{|\mathbf{T}||\mathbf{W}|} \end{bmatrix}. \tag{3.6}$$

Such a solution is feasible if it fulfills the constraints given in equation 3.5, i.e. that the entries of each column in equation 3.6 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.

To illustrate the problem of static target-based weapon allocation, we will here consider a specific problem instance, consisting of 5 targets and 4 firing units. Such a problem instance is defined by a target value vector $\mathbf{V}$ and a matrix $\mathbf{P}$ of kill probabilities, e.g.:

$$\mathbf{P} = \begin{bmatrix} 0.8 & 0.9 & 0.6 & 0.7 \\ 0.3 & 0.5 & 0.4 & 0.6 \\ 0.5 & 0.6 & 0.3 & 0.7 \\ 0.5 & 0.7 & 0.6 & 0.4 \\ 0.8 & 0.6 & 0.6 & 0.8 \end{bmatrix}.$$

$$\mathbf{V} = \begin{bmatrix} 0.7 \\ 0.6 \\ 0.8 \\ 0.4 \\ 0.6 \end{bmatrix}$$

In this case, there are $5^4 = 625$ different possible allocations, and it is far from trivial for a decision maker to find the best of the solutions in a short amount of time. To this particular problem instance, the allocation:

$$\mathbf{X}_{opt} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

is the optimal solution, resulting in the objective function value 1.19.

The Static Asset-Based Weapon Allocation Problem

The main difference between the static target-based weapon allocation problem and the static asset-based weapon allocation problem is that in the latter, the aims of the targets are assumed to be known. In other words, the asset-based formulation demands knowledge of which targets that are headed for which defended assets. Therefore, the static asset-based weapon allocation problem formulation is suitable for ballistic missile defense problems (such as base camp protection against artillery and mortars,) while the target-based formulation is more appropriate when the intended aim of the targets are not known for sure (Murphey, 2000). To determine which targets that are headed for which assets can be thought of as a data association problem (Malcolm, 2004).

In the static asset-based weapon allocation problem, each offensive target $T_i \in \mathbf{T}$ is assumed to be aimed at a defended asset, where each defended asset $A_j \in \mathbf{A}$ 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 (Hosein and Athans, 1990b). As in the static target-based weapon allocation problem, the defenders are equipped with firing units, where each pair of firing units and targets 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 expected total protection value of surviving defended assets, i.e.:

$$\max J = \sum_{j=1}^{|\mathbf{A}|} \omega_j \prod_{i \in \mathbf{G}_j} (1 - \pi_i \prod_{k=1}^{|\mathbf{W}|} (1 - P_{ik})^{X_{ik}}), \tag{3.7}$$

subject to:

$$\begin{aligned} \sum_{i=1}^{|\mathbf{T}|} X_{ik} = 1, \quad \forall k, \\ X_{ik} \in \{0,1\}, \quad \forall i \forall k. \end{aligned} \tag{3.8}$$

In equation 3.7, the inner product represents the expected survival probability of target T_i, while the outer product represents the expected survival probability of defended asset A_j. Consequently, the sum over the outer product gives us the total expected protection value of the defended assets, which we would like to maximize, as stated above.

From the solution of this optimization 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. It can be noted that if exactly one target is aimed for each of the defended assets, the protection value of the defended asset can be assigned as the target value of the attacking target, whereupon the expected total value of surviving targets can be minimized. Hence, for this special case, we end up with the static target-based weapon allocation problem, stated in equation 3.4. In this way, the asset-based formulation can be seen as a generalization of the target-based formulation. Based on this reasoning, it can be concluded that the static asset-based weapon allocation problem is **NP**-complete as well (Hosein, 1990).

To illustrate the problem of static asset-based weapon allocation, we will here consider a small problem instance consisting of 5 targets, 3 defended assets, and 3 firing units. Such a problem instance is defined by a vector of protection values, a vector of lethality probabilities, and a matrix of kill probabilities, e.g.:

$$\boldsymbol{\omega} = \begin{bmatrix} 0.7 \\ 0.4 \\ 0.6 \end{bmatrix}$$

$$\boldsymbol{\pi} = \begin{bmatrix} 0.5 \\ 0.4 \\ 0.6 \\ 0.8 \\ 0.4 \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} 0.8 & 0.9 & 0.7 \\ 0.3 & 0.5 & 0.6 \\ 0.5 & 0.6 & 0.7 \\ 0.5 & 0.7 & 0.4 \\ 0.8 & 0.6 & 0.5 \end{bmatrix}.$$

We also need information regarding which defended asset a particular target is headed for. In this example, we assume that $\mathbf{G}_1 = \{T_1\}$, $\mathbf{G}_2 = \{T_2,T_3\}$, and $\mathbf{G}_3 = \{T_4,T_5\}$, i.e. the first target is headed for the first defended asset, the second and third targets are headed for the second defended asset, and the fourth and fifth targets are headed for the third defended asset. For this particular problem instance, there are $5^3 = 125$ feasible solutions (note that the number of feasible solutions does not depend on the number of defended assets, but only on the number of targets and firing units). Despite the small

scale of this problem, it is nearly impossible for a human decision maker to very quickly come up with an optimal or near-optimal solution to the problem of which firing units to allocate to which targets. Hence, computer support is needed.

3.2 Parameters and Algorithms for Threat Evaluation

In this section, a review of existing literature on threat evaluation is presented. More specifically, parameters that have been suggested as being useful for threat evaluation are identified, analyzed, and discussed in section 3.2.1. Different kinds of algorithms that have been proposed for threat evaluation are presented in section 3.2.2. Most existing literature on parameters and algorithms for threat evaluation stems from the research area of information fusion, and has as a consequence of this been published in the annual international conferences on information fusion, or in information fusion journals. Moreover, a couple of interesting papers related to threat evaluation have been published in defense related conferences such as the international society for optics and photonics (SPIE) conferences and international command and control research and technology symposiums (ICCRTS). Lastly, some work can be traced back to the tactical decision making under stress (TADMUS) program, initiated by the US Navy for enhancing tactical decision making in air defense scenarios in littoral environments.

3.2.1 Parameters for Threat Evaluation

In order to be able to identify parameters suitable for threat evaluation, it is necessary to first define what is meant by a threat. A threat is in Roy et al. (2002) defined as:

> "an expression of intention to inflict evil, injury, or damage."

This definition is also in accordance with the definition of the word given by the Merriam-Webster dictionary. In the Oxford English dictionary, a threat is defined as a person or thing likely to cause damage or danger.

Threats can according to Steinberg (2005) be modeled in terms of relationships between threatening entities and threatened entities. In the terminology used throughout this thesis, we refer to the threatening entities as targets, while threatened entities are referred to as defended assets. Hence, as made clear in section 3.1.1, a threat is posed by a target to a defended asset. This also means that a target can constitute a threat to some defended assets, at the same time as it does not constitute a threat towards other defended assets. Naturally, the question then arises of how to judge whether a certain target constitutes a threat to a defended asset or not (or rather, to which extent it constitutes a threat)?

Looking at existing literature, a target's threat is often assessed as a combination of its *capability* and *intent* to inflict damage on a defended asset (cf. Nguyen (2002); Roy et al. (2002), and Waltz and Llinas (1990)). What is meant by a target's capability is its ability to inflict injury or damage to the defended asset, while its intent refers to its will or determination to inflict such damage (Paradis et al., 2005). Besides the threat components of capability and intent, a third component is described in Little and Rogova (2006): *opportunity*. An opportunity is in this context a spatio-temporal state of affair making it possible to carry out one's intent given sufficient capabilities.

Based on the findings from the review of existing literature on suitable parameters to use for threat evaluation, three main classes of parameters have been identified: *proximity*, *capability*, and *intent* parameters.

Proximity Parameters

An important class of parameters for assigning threat values to pairs of targets and defended assets are proximity parameters, i.e. parameters that in some sense are measuring the target's proximity to the defended asset. A key parameter that is used in many threat evaluation techniques (Roy et al., 2002; Allouche, 2005) is the distance from the defended asset to the *closest point of approach* (CPA). The CPA is the point where the target eventually will be the closest to the defended asset, given current track estimates. If we assume that the defended assets are stationary (i.e. non-moving), the CPA is the orthogonal projection of the position of the defended asset on the extension of the target's velocity vector (see figure 3.2). The distance between the position of the defended asset (often referred to as the threat reference point or point of interest) and the CPA can be used as one measure of the threat level. Targets that are far away from a defended asset can be assumed as not constituting a threat to the defended asset at present, while a shorter distance indicates a more potential viable threat. The distance from the defended asset to the CPA will in the following be referred to as *range at CPA*, while the distance from the target to the CPA will be referred to as *range from CPA*.

A number of parameters that are closely related to these distances are *time to CPA* (TCPA), *CPA in units of time* (CPA IUOT), and *time before hit* (TBH). If two targets of the same kind have the same CPA but different TCPA, the one with lowest TCPA will based on this reasoning constitute a larger threat to the defended asset. The TCPA parameter is calculated as the distance between the CPA and the target's current position, divided by the target's current speed, i.e.

$$TCPA = \frac{range\,from\,CPA}{speed}. \tag{3.9}$$

The parameter CPA IUOT is the time it would take the target to hit the defended asset after a 90° turn at its CPA, and is calculated as the distance be-

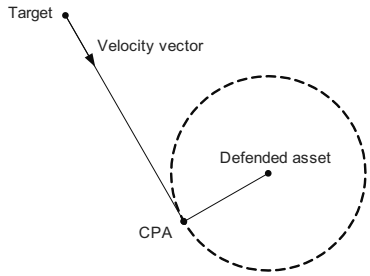


Figure 3.2: Closest point of approach between a target and a stationary defended asset.

tween the CPA and the position of the defended asset divided by the target's current speed:

$$CPA\,IUOT = \frac{range\,at\,CPA}{speed}. \tag{3.10}$$

The TBH parameter is an estimate of the time it would take the target to hit or reach the defended asset. This is directly applicable for targets such as missiles, but can also be used as a proximity measure for weapon platforms (helicopters, fighters, etc.). This is according to Roux and van Vuuren (2008) a commonly used parameter for threat evaluation, where a target with a long TBH to a defended asset may be assigned a lower threat value than a target with shorter TBH. The TBH is calculated as:

$$TBH = TCPA + CPA\,IUOT = \frac{range\,from\,CPA + range\,at\,CPA}{speed}. \tag{3.11}$$

In the above calculations, it is assumed that targets have constant velocities. This is an assumption that often is made when performing threat evaluation. For many platforms and conventional weapons this is a reasonable assumption, since they seldom make rapid maneuvers between two track updates (Oxenham, 2003). However, for highly maneuvering targets such as guided missiles, the assumption is often unrealistic (Allouche, 2005). In such cases, the quick maneuvers of the target may cause large changes in the values of the above parameters between two updates. This can in its turn cause rapid changes in the threat values assigned to a target-defended asset pair. Such changes are clearly unwanted since it often results in an unstable prioritized threat list. In order to handle the problem of highly maneuvering targets, there is sometimes a need for the use of smoothing techniques for the trajectory of targets in order to ensure only gradual changes in their threat values (Roy et al., 2002). For more on this issue, see Allouche (2005) and Allouche (2006).

Other proximity parameters that are suggested in the literature are the distance between the target and the defended asset and its rate of change. According to Okello and Thoms (2003), the threat posed by a very distant weapon

platform should be close to minimum and then increase gradually when the target approaches the defended asset. It should reach its maximum when the defended asset is within range of the target's weapon systems. The same kind of thinking can naturally be extended to targets that are weapons (e.g. missiles). Such targets are not at all constituting a threat to the defended asset when the distance between the target and the defended asset is larger than the weapon reach, but if the distance is shorter, the target becomes a more imminent threat the closer it comes.

Capability Parameters

The next class of parameters for threat evaluation that has been identified is the capability parameters. Such parameters reflect a target's capability to cause damage to a defended asset. A central parameter here is target type. The target type is often not known with certainty, but the target's estimated speed, radar cross-section, answer to IFF-interrogation, etc. can give a reasonable indication of what kind of target that has been detected. Once the target type has been estimated, parameters such as weapon type and weapon envelope can be inferred, given that the target is a weapon platform (otherwise the target type will also be the weapon type). These parameters are concerned with the lethality of the target and clearly are related to the proximity parameters, since a target is more threatening if it is able to overlay its weapon envelope over a defended asset than if it is outside the range of its weapon systems (Okello and Thoms, 2003). Basically, a hostile aircraft is of high importance when defended assets are within the firing range of its missiles (Virtanen et al., 2006). Fuel capacity is another capability parameter that is related to proximity. Given an identified target type it may be possible to have prior knowledge regarding the fuel capacity. Such information can be used to reason about the target's maximum radius of operation. According to Oxenham (2003), the most important parameters for threat evaluation are the lethality of the target, the imminence of the target to the defended asset, and the geometry of the target's weapon envelope relative to the defended asset.

Intent Parameters

The class of intent parameters is a broad category, containing parameters that can reveal something about the target's intent to cause damage to the defended asset. An example of this is the target's kinematics. According to Oxenham (2003), the target's velocity (i.e. its speed and heading) in combination with its altitude can be a good indicator of the target's intent to attack a defended asset. Another parameter based on kinematic data that can be used is the number of recent maneuvers (Liebhaber and Feher, 2002a). Here, it is assumed that a target that is maneuvering a lot is having a more threatening behavior than a non-maneuvering target. The typical behavior for commercial aircraft is to

fly with steady speed, at a constant altitude, and at a straight line. Moreover, they tend to use established airways and respond to verbal queries on special radio frequencies (Smith et al., 2004). In other words, aircrafts deviating from such behavior may be an indication of that the intent possibly is hostile. Other parameters that may provide clear indications of hostile intent are the use of radar jamming and other kinds of electronic warfare, and whether the target's fire-control radar is on or not (Oxenham, 2003; Benavoli et al., 2007).

Even though there are many parameters that can reveal something regarding the intent of a target, it is important to keep in mind that intent recognition is no exact science. According to St. John et al. (2004a), the intent of an aircraft can never be established with certainty. Still, there exists work on intent inference for air defense based on aircraft flight profile analysis, such as the fuzzy logic approach reported in Foo et al. (2009). The author of this thesis has earlier investigated the possibilities of using Bayesian networks for inferring the intent of ground targets, as reported in Johansson and Falkman (2006).

To summarize the findings from this survey over parameters used for threat evaluation, we can identify three main classes of parameters: proximity parameters, intent parameters, and capability parameters. Examples of these are illustrated in table 3.1. Proximity parameters are generally quite easy to use

Table 3.1: Classes of parameters for threat evaluation.

Parameter class	*Examples*
Proximity parameters	Euclidean distance, CPA
Capability parameters	Target type, weapon envelope
Intent parameters	Speed, heading

since the estimates needed for calculation of proximity usually can be obtained from sensors. Capability parameters often need more refined information since it can be hard to estimate parameters such as target type from sensor observations only. Lastly, knowing a target's intent would be extremely useful when assessing the level of threat posed by a target. However, even though it is possible to obtain parameters such as speed and heading, it is often hard to construct a good model that makes it possible to with a high precision infer a target's intent from such parameters.

Parameters Used by Air Defense Operators

In Liebhaber and Feher (2002a) and Liebhaber et al. (2002), a list of parameters used for threat evaluation by navy air defense operators are presented. Out of these, six parameters are referred to as being of critical importance when evaluating the threat. These are: the country of origin (which is based on the initial point of detection), IFF Mode, intelligence reports, altitude, proximity to a commercial airlane, and electronic support measures (i.e. the target's radar

signature, revealing information regarding the target type). Other parameters

Table 3.2: A list of parameters used by air defense operators in US Navy for threat evaluation (adapted from Liebhaber and Feher (2002a)). Critical parameters are marked with **bold**.

Parameter	*Description*
Altitude	The target's altitude (feet above ground).
Country of origin	Over which country was target first located?
Following airlane	Does target follow a commercial airlane or not?
IFF mode	Identified as friend, foe, or neutral?
Intel	Intelligence reports.
Radar/ESM	Kind of radar system used?
Coordinated activity	Is target nearby/communicating with other targets?
Course	The target's heading relative the defended asset.
Range at CPA	Distance from target's CPA to the defended asset.
Feet wet	Is the target flying over water or not?
Maneuvers	Number of recent maneuvers.
Number	Number of aircrafts in target's formation.
Range	Target's distance from defended asset.
Speed	An estimate of the target's speed.
Visibility	How far is the line of sight?
Weapon envelope	How large is target's estimated weapon envelope?
Wings clean	Is the target carrying weapons?

that are used by air defense operators are according to Liebhaber and Feher (2002a) and Liebhaber et al. (2002): the target's heading relative to the defended asset (in their study there is only one defended asset and that is the own naval craft), whether the target is involved in coordinated activity or not, range at CPA, whether the target is flying over water or land, the number of recent maneuvers, the number of aircraft flying in formation, the target's distance from the own ship, the target's speed, the target's position relative to its estimated weapon envelope, whether it carries weapons or not, and the type of weapons carried by the target. A summarizing list of these parameters is given in table 3.2.

3.2.2 Existing Algorithms for Threat Evaluation

To develop reliable automated threat evaluation algorithms for aiding air defense decision makers have for long been considered a "holy grail" (St. John et al., 2004a). According to Liebhaber and Smith (2000), it is not well understood and documented how air defense decision makers estimate target values. A review of early attempts in building automated systems for threat evaluation is presented in Waltz and Llinas (1990). The main problem with these early

attempts is according to Benavoli et al. (2007) the lack of means to deal with uncertain domain knowledge.

In a series of experiments with US Navy officers summarized in Liebhaber and Feher (2002a), it has been studied how human air defense operators are likely to perform manual threat evaluation. It should be noted that these studies only deal with point defense, i.e. protection of a single defended asset (the naval craft itself). Hence, it is not obvious how to extend this to area defense involving the protection of several defended assets. In the findings from their studies, it is suggested that human operators evaluate the threat posed by air targets through initially activating a template corresponding to the target's particular type. The activated template is used to set a baseline threat rating for the target (i.e. a target value). Parameters relevant to the active template are then assessed and compared to expected values for the activated template. Depending on the degree of match between the actual value of a parameter and its expected value, the current threat rating for the target is adjusted up or down (as an example given in Liebhaber and Feher (2002a), an aircraft in a littoral environment with a speed of 250 knots adds 0.2 to the current threat rating, while a speed of 500 knots adds 1.8[1]). In Liebhaber and Smith (2000), it is mentioned that the geopolitical situation influences the tolerance for deviations from expected values. In case of no unexpected data, the process is stopped after evaluation of the most important parameters. Otherwise the so called exception score becomes negative and the evaluation process will continue with more parameters (see the list given in table 3.2), until there are no more parameters to process, or until the fit of the expected values and the parameter values is good enough. From this naturalistic model a *rule-based algorithm* for threat evaluation was developed (Liebhaber and Feher, 2002a). The same algorithm outline is described for threat evaluation of surface targets in Liebhaber and Feher (2002b). A very high-level description of the rule-based algorithm is shown in algorithm 3.1[2].

In Runqvist (2004), the threat evaluation module of a commercial air defense system is described. For each defended asset, a defended area (represented as a circle) has to be defined. This is accomplished by an operator specifying the position of the defended asset, together with its priority and the radius of the defended area. The parameters that are used for threat evaluation in the system are time, relative bearing, altitude, identity, number of engagements, speed, size, and aircraft type. The parameter time is the minimum time it will take the target to reach the defended area border (assuming constant speed), while the identity parameter specifies whether the target is identified as hostile or unknown. Number of engagements refers to the number of friendly weapon systems currently engaged to the target (Runqvist, 2004). The operator spec-

[1] Obviously, the threat ratings used are not restricted to the interval [0,1].

[2] Whether air defense operators really perform threat evaluation in this way or not is not easy to tell, but the use of exception score as described in the algorithm does not seem very rational to the author of this thesis.

Algorithm 3.1 Rule-based algorithm for threat evaluation (Liebhaber and Feher, 2002b).

```
Exception Score ← 0
Threat ← ID Threat Rating + Platform Threat Rating
for all critical parameters do
    Get Value and Weight for current parameter
    Threat ← Threat ± Threat Change Rating
    if Value is an exception then
        Exception Score ← Exception Score - Weight
    end if
end for
while Exception Score < 0 and more parameters exist do
    Get Value and Weight for the current parameter
    Threat ← Threat ± Threat Change Rating
    if Value is an exception then
        Exception Score ← Exception Score - Weight
    else
        Exception Score ← Exception Score + Weight
    end if
end while
```

ifies rules for each parameter, assigning a number of *threat points* depending on the parameter's actual value. An example of a rule that can be specified is (Runqvist, 2004):

$$\text{IF } Time < 1 \text{ THEN } Points = Points + 15. \tag{3.12}$$

For each target-defended asset pair, the points from the individual parameters are summed and then multiplied with a weight that is dependent on the priority of the defended asset. Finally, this value is divided by the maximum number of points, resulting in a normalized threat value between 0 and 1 assigned to the target-defended asset pair.

Another kind of rule-based algorithm is suggested in Liang (2007), in which *fuzzy inference rules* are used to calculate the level of threat posed by air targets to naval crafts using altitude, speed, CPA, and range as input parameters. For each input parameter, three membership functions are defined. Using the membership functions, fuzzy inference rules are defined for how the input should affect the output parameter threat rating. Hence, the suggested approach is used for the protection of a single naval craft. A modified version of this algorithm is presented in section 4.1.2.

In Dall (1999), an approach to threat evaluation based on predicate logic is suggested. However, in the proposed approach it is not the case that an actual threat or target value is calculated. Instead, it is checked whether there is any

support for the hypothesis that the target T_i threatens the defended asset A_j. This is accomplished through the use of logical rules such as:

$$(\forall x)(\forall y)(Target(x) \wedge Asset(y) \wedge Capability(x,y) \\ \wedge Intent(x,y)) \rightarrow Threat(x,y), \tag{3.13}$$

where rules for $Intent(x,y)$ and $Capability(x,y)$ are further defined, and so on. Hence, for the target to be considered as being threatening to the defended asset, it is required that the target has the capability and the intent to cause damage to the defended asset.

In Dong and Qing (1999), a *neural network approach* to threat evaluation is suggested. A feature of artificial neural networks which is put forward as an advantage is that they are learned from training data instead of being created from explicit expert knowledge. The type of neural network used is a so-called multilayer feed-forward network, consisting of a layer of input nodes, a hidden layer, and an output layer in which the calculated threat is given as output. The input parameters used in the network are the distance between the target and the own ship (the defended asset), the target's speed, and the target's heading relative the defended asset. A similar architecture is used in Azak and Bayrak (2008) (shown in figure 3.3), in which a feed-forward network consisting of four input nodes (where two nodes correspond to various distance measures, and the two others to target speed and orientation of the target) is used. Moreover, there are two hidden nodes and four output nodes, where the latter correspond to different threat categories. The network has been trained using synthetic data and evaluated by measuring the classification error, but no discussion is provided on how such labeled data that is needed for training would be obtained in a real system. No such discussion is provided in Dong and Qing (1999) either, except for that it is mentioned that "expert human exemplary data" have been used in a simulation for which no details are given.

The last family of algorithms for threat evaluation that we have been able to identify in the literature is so called *graphical models*. Okello and Thoms (2003) present a Bayesian network based algorithm in which target state estimates are used for evaluating the threat posed by a target to a defended asset. The target's capability to threaten a defended asset is measured by comparing the maximum range of the target's weapon systems with the range between the target and the defended asset. In the same manner, the target's intent to threaten the defended asset is measured by looking at the rate of change of the range between the target and the defended asset, together with the angle between the target's velocity vector and the vector pointing from the target to the defended asset. The threat value is then calculated as the product of the capability and intent components. A prototype Bayesian network for threat evaluation is also described in Runqvist (2004), in which a simplified version of the network for target intent recognition suggested in Nguyen (2002) is used as a basis for the threat evaluation. Another example of the use of graphical models for threat

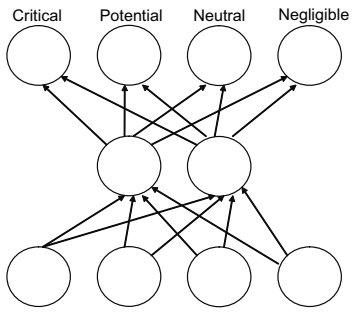


Figure 3.3: Feed-forward topology describing the artificial neural network used in Azak and Bayrak (2008).

evaluation is presented in Benavoli et al. (2007), where an evidential network (i.e. a valuation-based network using belief functions) is used to represent and reason about threats in the context of air defense. The evidence variables that are used in the network are: evasive maneuvers, fire control radar, countermeasures, political climate, IFF squawking, flight plan agreement, platform type, and imminence. Most of these variables are binary, while some can take on more states. Basically, the variables reveal whether the target is taking any evasive actions, turning on its fire control radar (i.e. is about to fire a weapon), employing electronic warfare countermeasures such as the use of jamming or chaff, what the political climate is like, if it answers correctly to IFF interrogation, and so on. Together with a few mediating variables, these evidence variables influences the intent and capability variables, which in turn have impact on the value of the threat variable.

Table 3.3: Algorithmic approaches to threat evaluation.

Algorithmic approach	*References*
Rule-based algorithms	Runqvist (2004); Liebhaber and Feher (2002b)
Fuzzy logic	Liang (2006, 2007)
Neural networks	Dong and Qing (1999); Azak and Bayrak (2008)
Graphical models	Okello and Thoms (2003); Benavoli et al. (2007, 2009)

In table 3.3, we summarize the different algorithmic approaches to threat evaluation suggested within available literature.

3.3 Algorithms for Static Weapon Allocation

Initial research on the static target-based weapon allocation problem dates back as far as the end of the 1950s (cf. Manne (1958); den Broeder et al. (1959)). 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 (Malcolm, 2004). Despite the end of the Cold War, research on defensive weapon allocation still remains a very active area (Malcolm, 2004). A severe threat encountered in many military international operations of today and tomorrow is that of rockets, artillery, and mortars 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. This kind of attacks has actuated the need for quick and efficient air defense. Similarly, the development of the so called Iron Dome system, intended for providing a defensive countermeasure to rocket threats against Israel's civilian population centers is highly topical. In this kind of applications, the assumption of static weapon allocation that all firing units are to be allocated at once is realistic due to the short range of the attacks, leading to very short reaction times and few engagement opportunities.

The static weapon allocation problems of focus within this thesis have 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 (Ahuja et al., 2007), in the sense that effective methods for real-time allocation are lacking.

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 Matlin (1970). Some years later, a monograph describing many of the developed mathematical models for weapon allocation problems was published by Eckler and Burr (1972). Unlike Matlin's review, the monograph by Eckler and Burr takes on the weapon allocation problem from a defender's view. A quote from that monograph says that:

> "None of the references actually cited in this monograph is classified. Perhaps surprisingly, it has not been found essential to cite any. Mathematical models, by their nature, do not normally require classification, and it appears that the authors of classified studies have usually tried to publish the mathematical aspects of such studies in the unclassified literature."

This is interesting reading and suggests that research on weapon allocation is (or at least was) more open than the research on its threat evaluation counterpart seems to be nowadays. 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 still is in use (cf. Karasakal (2008)). 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 (Winston, 1997). Other kinds of tools such as the use of Lagrange multipliers and dynamic programming are also described in Eckler and Burr (1972). 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 Cheong (1985). 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 (Cheong, 1985, p. 66). In later years, advanced computer-based techniques have been developed which are better suited for real-time weapon allocation (Huaiping et al., 2006). In the following, we will not consider techniques that are relying on gradients (i.e. calculus-based techniques). We will instead focus on enumerative techniques, and heuristic/approximate techniques such as guided random search techniques.

Enumerative techniques are typically guaranteed to find an optimal solution in bounded time for every finite size problem instance, hence, they are referred to as *exact* algorithms. The problem with exact algorithms is that the computation times needed often are too high for practical applications. With *heuristic* algorithms (e.g. guided random search techniques), we cannot guarantee to find optimal solutions, but instead we can find good solutions in a significantly reduced amount of time (Blum and Roli, 2003). Hence, heuristic algorithms are used when we seek good feasible solutions to optimization problems in circumstances where the complexity of the problem or the limited time available for its solution do not allow using exact algorithms (Rardin and Uzsoy, 2001). According to Aarts and Lenstra (1997), this kind of algorithms is the method of choice for **NP**-hard problems, as it provides a robust approach for producing solutions of high quality in reasonable time for problems of realistic size. The real-time aspects is one of the major characteristics of real-world air defense situations (Allouche, 2006; Huaiping et al., 2006), but despite this, the real-time aspects have not traditionally been in focus of the operations research community (Séguin et al., 1997).

3.3.1 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. In such cases, it is not possible to in real-time (or even after several years of computation) obtain an optimal solution using an exhaustive search algorithm. The issue of how large problem sizes that can be handled in real-time with exhaustive search algorithms is discussed in section 7.2.1. 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 den Broeder et al. (1959), and the local search algorithm suggested in Hosein (1990) 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 (Hosein, 1990). 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 (Hosein, 1990). However, the general static target-based weapon allocation problem has been proved to be **NP**-complete (Lloyd and Witsenhausen, 1986), 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 Ahuja et al. (2007), 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 Ahuja et al. (2007), some problem instance 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 (Hosein, 1990; Ahuja et al., 2007).

3.3.2 Heuristic Approaches

A well-known heuristic approach for static target-based weapon allocation is the greedy maximum marginal return algorithm, originally suggested in den Broeder et al. (1959). A similar greedy algorithm is presented in Kolitz (1988). 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 4.2.3. 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 Metler and Preston (1990) 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 Metler and Preston (1990) 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 Wacholder (1989), 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 author's view is not possible to generalize. It is in Huaiping et al. (2006) 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 Julstrom (2009), while a genetic algorithm combined with local search is presented in Lee et al. (2002c) and Lee and Lee (2003). The quality of the solutions returned by the greedy maximum marginal return algorithm presented in Kolitz (1988) is in Julstrom (2009) compared to the solutions returned by genetic algorithms. 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 problem sizes. It is in Julstrom (2009) 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 Chen et al. (2009), 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 compared to use the genetic algorithm without local search, but that the computational time needed is increased. The effect of real-time requirements on the algorithms are not tested.

The use of ant colony optimization for weapon allocation is suggested in Lee et al. (2002a); Lee and Lee (2005). Reported results in Lee and Lee (2005) and Lee et al. (2002a) 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 Ciobanu and Marin (2001). 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 (Suman and Kumar, 2006). 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 Ciobanu and Marin (2001), 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 Lee and Lee (2005).

The algorithms were, as describe above, allowed to run for two hours, so it is not known how the algorithms perform under more realistic time constraints.

Ahuja et al. (2007) present good performance results for an approach using a minimum cost flow formulation heuristic for generating a good starting feasible solution. 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 weapons 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 neighborhoods used are very large. To search such large neighborhoods typically takes considerably amounts of computations and demands implicit enumeration methods (Jha, 2004). By using the concept of an improvement graph, it becomes possible to evaluate neighbors faster than other existing methods (Ahuja et al., 2007; Jha, 2004).

Recently, the use of particle swarm optimization for static target-based weapon allocation has been suggested. In Teng et al. (2008), 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 Zeng et al. (2006) 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 3.4.

Table 3.4: Algorithmic approaches to the static weapon allocation problem.

Algorithmic approach	*References*
Branch-and-bound	Ahuja et al. (2007)
Genetic algorithms	Julstrom (2009); Lee and Lee (2003)
Ant colony optimization	Lee et al. (2002a); Lee and Lee (2005)
Greedy algorithms	den Broeder et al. (1959); Kolitz (1988)
VLSN	Ahuja et al. (2007)
Neural networks	Wacholder (1989)
Particle swarm optimization	Zeng et al. (2006); Teng et al. (2008)

3.4 Discussion

As indicated in section 3.1.1, it is possible to calculate aggregated target values from individual threat values in different ways. The weighted average presented in equation 3.2 seems reasonable in most air defense situations, but can be problematic to use when there e.g. are very many defended assets, since there is a potential risk that the threat against one single defended asset "disappears" in the aggregation of threat values into a single target value due to the many defended assets to which the target does not constitute a threat. The alternative presented in equation 3.3 in general results in higher target values. Of course, there are also other ways in which target values can be calculated. The problem here is that there is no single aggregation operator that can be argued to be better than all others for all kinds of situations. This leads us to the problem of what threat values and target values really represent. As we have seen, the threat value is the level of threat posed by a target to a defended asset, but this threat value will vary depending on what parameters and algorithms that are used for estimating the threat value. Consequently, this will hold true also for target values. Furthermore, as will be discussed more in chapter 5, human air defense experts are often disagreeing on a target's level of threat (St. John et al., 2004a). Arguably, there are no such things as objective threat and target values. In the author's view, these values are subjective. These subjective values are nevertheless still useful, in that they can be used to judge how threatening a target is, given a specific threat model of what parameters that should decide the level of threat. However, that target values are subjective by nature is rarely mentioned in literature. On the contrary, target values are in studies of weapon allocation problems nearly always assumed to be known perfectly and are taken for granted. We are not providing a solution to how to deal with the subjectiveness of target values, but really would like to highlight that the allocation given as output from a weapon allocation algorithm is only (at the very best) optimal given a specific threat model.

In both the static target-based and asset-based weapon allocation problems presented in this chapter, it is assumed that all firing units should be allocated to targets. Various variants of especially the static target-based problem presented here have been suggested in literature. It is e.g. in Jha (2004) mentioned that constraints such as lower and upper bounds on the total number of firing units assigned to a target T_i can be added, as well as bounds on the survival value of a target. A modification of the general static weapon allocation problems that we think can be fruitful in order to make them more adapted to realistic settings is to add a cost to each use of firing units, since interceptors such as surface-to-air missiles are quite expensive, but also due to the fact that the savings of interceptors can be needed later on in a dynamically evolving scenario. Hence, we suggest that it may be interesting and worthwhile to develop modified versions of the problems where a resource usage cost C_k is added to each firing unit W_k, where the assumption that all firing units should be allocated to targets is re-

laxed. A modified version of the static target-based weapon allocation problem could then look like:

$$\min F = \sum_{i=1}^{|\mathbf{T}|} V_i \prod_{k=1}^{|\mathbf{W}|} (1 - P_{ik})^{X_{ik}} + \sum_{i=1}^{|\mathbf{T}|} \sum_{k=1}^{|\mathbf{W}|} X_{ik} C_k, \tag{3.14}$$

subject to:

$$\begin{gathered} \sum_{i=1}^{|\mathbf{T}|} X_{ik} \leq 1, \ \ \forall k, \\ X_{ik} \in \{0,1\}, \ \ \forall i \forall k. \end{gathered} \tag{3.15}$$

Similarly, the modified version of the static asset-based weapon allocation problem can be formulated as:

$$\max J = \sum_{j=1}^{|\mathbf{A}|} \omega_j \prod_{i \in \mathbf{G}_j} (1 - \pi_i \prod_{k=1}^{|\mathbf{W}|} (1 - P_{ik})^{X_{ik}}) - \sum_{i=1}^{|\mathbf{T}|} \sum_{k=1}^{|\mathbf{W}|} X_{ik} C_k, \tag{3.16}$$

subject to:

$$\begin{gathered} \sum_{i=1}^{|\mathbf{T}|} X_{ik} \leq 1, \ \ \forall k, \\ X_{ik} \in \{0,1\}, \ \ \forall i \forall k. \end{gathered} \tag{3.17}$$

Hence, if a cost associated with the use of a weapon system is higher than the estimated value of the engagement, it is not allocated to any target. A problem associated with such a formulation obviously is to find the balance of selecting appropriate costs for resource usage. Such a cost function is also used in the general optimization problem of allocating resources to tasks, as presented in Gelenbe et al. (2010).

The threat evaluation algorithms reviewed in this chapter are intended for classical air defense operations where there are a set of defended assets that should be protected against conventional air targets such as fighter planes and missiles. However, there are in today's world many cases of asymmetrical warfare in which the opponent does not follow classical doctrines that can be defined in a top-down manner. An example of this is the hijacking of commercial passenger jet airliners by al-Qaeda terrorists on September 11, 2001, which among other targets were used for crashing into the World Trade Center and Pentagon. In order to be able to detect and counter this kind of attacks, alternative methods are needed. Since this kind of unexpected situations are hard to define in advance, a promising idea is to use so called *anomaly detection methods* for this kind of situations. Initial work on using such an approach for

detection of vessel anomalies in the maritime domain has been presented in Johansson and Falkman (2007) and Riveiro et al. (2008), and the author of this thesis argues that the same kind of approach can be useful in the air defense domain.

3.5 Summary

Summarizing the findings in this chapter, a formal description of the threat evaluation problem has been given. We have also given mathematical formalizations of two variants of the static weapon allocation problem; static target-based weapon allocation and static asset-based weapon allocation. We have seen that open literature on threat evaluation is sparse, while considerably more research exists for static weapon allocation. Parameters that can be used for threat evaluation have been divided into three classes; proximity parameters (e.g. range to CPA), capability parameters (e.g. weapon envelope), and intent parameters (e.g. speed and heading). In the review of algorithms for threat evaluation, we have identified four main approaches. These are based on classical two-valued logic, fuzzy logic, artificial neural networks, and graphical models (e.g. Bayesian networks), respectively. A similar review of algorithms for static weapon allocation has revealed that much of the first decades of research mainly was devoted to analytical approaches. In the research during later years, a vast amount of heuristic approaches have been suggested, such as genetic algorithms, particle swarm optimization, ant colony optimization, and simulated annealing. However, despite some initial comparisons, it is not well-known how the algorithms perform, especially not under real-time constraints.

Chapter 4
Algorithms for Real-Time Threat Evaluation and Weapon Allocation

In this chapter, we present specific implementations of algorithms for real-time threat evaluation and weapon allocation. The purpose of these implementations is to allow for comparison of the various algorithms, as described further in chapter 7. The chosen implementations are based on the literature survey presented in chapter 3. Many of the algorithms presented here have earlier been published in conference proceedings. The Bayesian network for threat evaluation presented in section 4.1.1 was originally described in Johansson and Falkman (2008a), while the fuzzy logic algorithm for threat evaluation presented in section 4.1.2 earlier has been published in Johansson and Falkman (2008b). The implemented exhaustive search algorithm and the genetic algorithm for static target-based weapon allocation was originally published in Johansson and Falkman (2009b), while the particle swarm optimization and the ant colony optimization algorithms previously have been published in Johansson and Falkman (2010a).

4.1 Algorithms for Real-Time Threat Evaluation

As identified in section 3.2, suggested approaches for automated threat evaluation can be divided into classical two-valued logic, fuzzy logic, artificial neural networks, and graphical models. Since the data which is used as input to threat evaluation algorithms often are uncertain by nature (Paradis et al., 2005; Benaskeur et al., 2008), and since the domain knowledge itself often is incomplete, it makes sense to only use approaches that can handle uncertainties within this domain. For this reason, we have chosen not to implement any algorithm based on two-valued logic (since such logic only can handle values that are *true* or

$false$). Artificial neural networks have not been seen as suitable for our purposes, since they rely on large amounts of labeled training data which at the least are very hard to acquire for this domain. Moreover, they are opaque to the user, so that they provide no explanations to a decision maker on why a target is considered to be threatening or not. Instead, we have implemented an algorithm based on fuzzy logic, as well as a graphical model approach, in which Bayesian networks have been used. Both these approaches are able to handle uncertainty, and are potentially more transparent to the user of the algorithm. The Bayesian network algorithm is presented in section 4.1.1, while the fuzzy logic approach is presented in section 4.1.2.

4.1.1 A Bayesian Network Approach

When creating the Bayesian network for threat evaluation, we have used many of the parameters identified in section 3.2.1. First of all, the variables and states to include into the network needed to be decided on. The random variable (node) of main interest here is the query variable $Threat$, since this is the variable that is used to calculate a threat value. Since we want to calculate a threat value in the range [0,1], the posterior probability of $P(Threat = true|\mathbf{z})$ (where $\mathbf{z}$ is an instantiation of a set of information variables $\mathbf{Z}$) is used as our estimate of the threat value posed by the observed target T_i to a specific defended asset A_j. Hence, a specific Bayesian network corresponds to a single target-defended asset pair (T_i,A_j). This means that a Bayesian network instance should be created for each target-defended asset pair, and that the resulting target value needs to be aggregated from the individual threat values.

As noticed earlier, threat can be seen as a combination of capability and intent. Therefore, a $Capability$ variable and an $Intent$ variable has been identified as well. As discussed in section 3.2.1, many different variables can be used to reason about the intent and capability of a target to damage a defended asset, so we have here chosen a representative subset of these. Capability parameters that have been included into our Bayesian network for threat evaluation are:

- $Target\ type$ and $Within\ weapon\ envelope?$,

while included intent and proximity parameters are:

- $Speed$, $Range\ from\ CPA$,
- $Range\ at\ CPA$, $TCPA$,
- $CPA\ IUOT$, TBH, and $Distance$.

Most of the variables are continuous by nature, but have (which is very common when working with Bayesian networks) been discretized, so that the range is $\{low,medium,high\}$ for $Speed$, $Capability$, and $Intent$, while it for the variable $Weapon\ range$ is $\{none,short,medium,long\}$, etc. The current version of

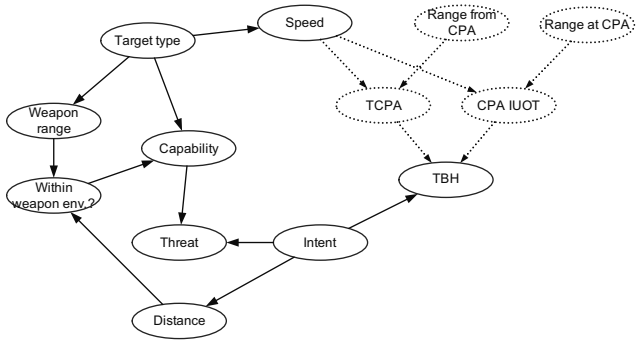


Figure 4.1: The variables used in the Bayesian network for threat evaluation, and the relationships between them.

the Bayesian network only considers the target types $F16$, $Mig21$, $B2$, and $B747$ (assuming a quite limited universe), but would obviously need to be extended with other types of aircrafts as well as different types of missiles, etc. in a fully operational system. However, the current network should be enough to demonstrate the usefulness of Bayesian networks for threat evaluation. To add more aircraft or missile types to the Bayesian network would not impact much on the complexity of the network, since it would not require adding extra nodes. The extra burden would rather only be noticeable when required to specify some extra numbers in the conditional probability table for $Target\,type$ and the nodes having $Target\,type$ as a parent.

In a second step, the structure of the Bayesian network needs to be identified, i.e. the directed edges between the nodes. Here, the technique of drawing directed edges from causes to their effects has been used. Starting with the variable $Threat$, we have earlier noticed that this is a combination of $Intent$ and $Capability$, hence, the $Intent$ and $Capability$ can be seen as causes of $Threat$. Consequently, a directed edge from $Intent$ to $Threat$ can be drawn, as from $Capability$ to $Threat$. Similarly, a specific $Target type$ has a causal effect on the estimated $Capability$. The target's $Intent$ can be seen as the cause of the Euclidean distance between the target and the defended asset ($Distance$), as well as of the TBH, and so on. All the used random variables and the relationships between them are shown in figure 4.1. As can be seen, some of the variables and the directed edges ($Range\ from\ CPA$, $Range\ at\ CPA$, $TCPA$, $CPA\ IUOT$, and their associated edges) are different from the others. These are not part of the final Bayesian network, but rather are used to (deterministically) calculate the value of TBH before any actual inference is made.

Even in this rather limited Bayesian network, knowledge elicitation of the probability distributions needed for the conditional probability tables (CPTs) of the network is the most cumbersome step. To illustrate what the conditional

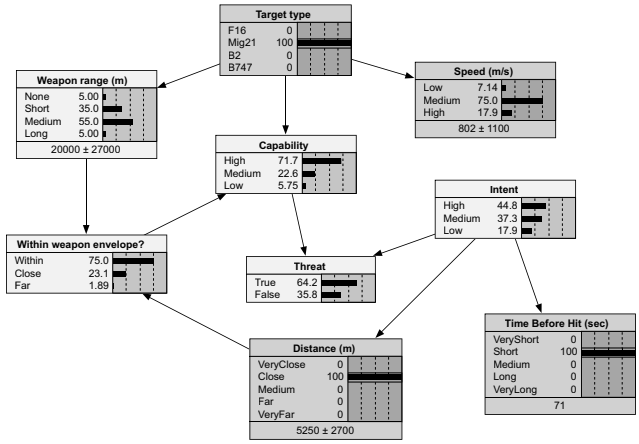


Figure 4.2: An example of an inference with the Bayesian network.

probability tables look like, the one used for the $Threat$ variable is shown in table 4.1. The remaining conditional probability tables can be found in Appendix A.

Table 4.1: Conditional probability table for $Threat$.

$Capability$	$Intent$	$true$	$false$
high	high	0.90	0.10
high	medium	0.70	0.30
high	low	0.20	0.80
medium	high	0.75	0.25
medium	medium	0.50	0.50
medium	low	0.10	0.90
low	high	0.50	0.50
low	medium	0.25	0.75
low	low	0.03	0.97

Since the $Threat$ variable has only two possible states ($true$ and $false$), and since its parents ($Capability$ and $Intent$) have three possible states each, a total of eighteen conditional probabilities are given in the conditional probability table.

In figure 4.2, it is shown how an instantiated Bayesian network is used to calculate the posterior probability of $Threat$, given certain information that the target is a $Mig21$ at a close distance and short TBH. Moreover, the speed of the target has been observed, but this observation is associated with uncer-

tainty, so that we are 75% certain that the discretized speed is medium. Given these observations, the posterior probability $P(Threat = true|\mathbf{z}) \approx 0.64$ has been calculated, which is taken as the estimate of the threat value the target is constituing to the defended asset.

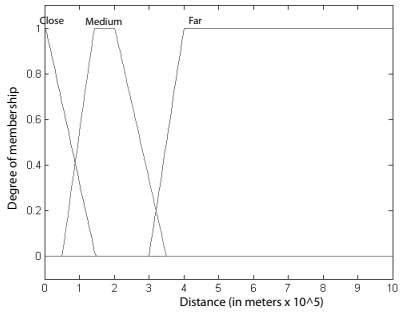
The developed Bayesian network for threat evaluation has been presented and discussed with experts working within the defense industry. Some of these experts have previously also worked as air defense officers. The appropriateness of the structure of the Bayesian network has in these discussions been confirmed, but the actual numbers used in the conditional probability tables have not been discussed and should only be seen as the author's opinion of suitable values.

4.1.2 A Fuzzy Logic Approach

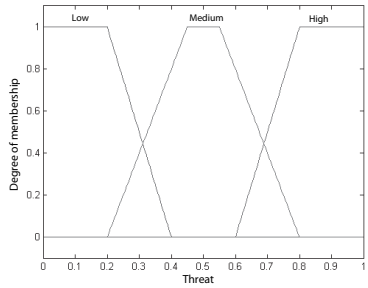
In the implemented fuzzy inference system (which is based on the work presented by Liang (2007)), we have used the same set of input variables as in the Bayesian network implementation, i.e. $Target\ type$, $Speed$, $Distance$, and $Time\ Before\ Hit$. As in the Bayesian network case, the purpose of the fuzzy inference system is to calculate a threat value for the corresponding target-defended asset pair, so that the output of the system is a fuzzy set $Threat$. The rules used in the implementation can be seen below:

```
TargetType == F16 --> Threat = high (1)
TargetType == Mig21 --> Threat = med (1)
TargetType == B2 --> Threat = high (1)
TargetType == B747 --> Threat = low (1)
TBH == short --> Threat = high (1)
TBH == med --> Threat = med (1)
TBH == long --> Threat = low (1)
Speed == low AND Distance == far --> Threat = low (1)
Speed == med AND Distance == med --> Threat = med (1)
Speed == high AND Distance == close --> Threat = high (1)
Distance == far --> Threat = low (0.25)
Distance == med --> Threat = med (0.25)
Distance == close --> Threat = high (0.25)
```

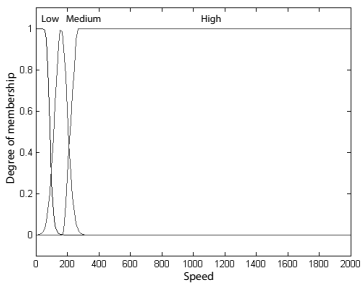
For each rule the number within parentheses denotes the weight of the rule. As can be seen, the $Capability$ and $Intent$ variables are not here used explicitly, instead, $Target\ type$ is modeled as having a direct impact on the threat value. The same holds true for TBH and $Distance$. The reason for this choice is to make it as similar to Liang's types of rules as possible, in which no intermediate variables are used. Moreover, the combination of $Speed$ and $Distance$ has been considered as having a mutual effect on the threat level, so that a high speed combined with a close distance are indications of a target being a very viable threat.



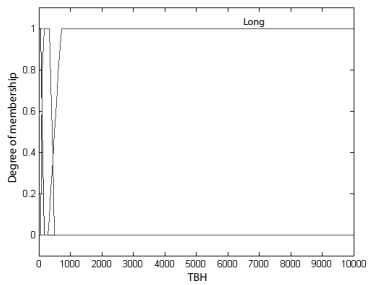
(a) Membership functions for $Distance$



(b) Membership functions for $Threat$



(c) Membership functions for $Speed$



(d) Membership functions for TBH

Figure 4.3: Illustration of some of the membership functions used in the fuzzy inference system for threat evaluation.

The set that has been used to model $Target\ type$ is actually a crisp set (i.e. a target is either a $F16$, $Mig21$, $B2$, or $B747$). The other variables have been modeled using fuzzy sets, and the membership functions for $Distance$, $Threat$, $Speed$, and TBH are shown in figure 4.3.

4.2 Algorithms for Real-Time Weapon Allocation

As discussed in section 3.1.2, a large number of different algorithms for static weapon allocation have been suggested in the open literature throughout the years. We will in this section present implementations of some of the heuristic algorithms that have been identified as promising for static real-time weapon allocation. Moreover, we will present an exact exhaustive-search algorithm, since such algorithms always return an optimal solution, given that there are no restrictions on the available search time. It is obvious that there in practice is not unlimited amount of time available for searching for an optimal

solution, but there are for small-scale problem instances no reasons not to use exact algorithms for real-time applications. The descriptions of the implemented algorithms are in most cases made on a quite high level, but for specific implementation details the Java source code can be downloaded from http://sourceforge.net/projects/sward/ as explained further in chapter 6.

Some of the presented algorithms (e.g. the maximum marginal return algorithm) make use of information regarding target values. These are therefore only directly applicable to static target-based weapon allocation. However, as was mentioned in section 3.1.2, the static asset-based weapon allocation problem can be approximated by its target-based counterpart. If information regarding which defended assets the targets are aimed for are combined with information regarding protection values of the defended assets, this can be used as a substitute for target values. Hence, by using this kind of approximation, the algorithms relying on target values can (with some minor modifications) be used for static asset-based weapon allocation as well.

4.2.1 Representation

The implementations of weapon allocation algorithms suggested in this thesis share the same kind of representation, in which a solution is represented as a vector of natural numbers of length $|\mathbf{W}|$. Each element k in the vector, (where $1 \leq k \leq |\mathbf{W}|$), points out the target T_i to which firing unit W_k is allocated. As an example of this, the vector $[2, 3, 2, 1]^\mathsf{T}$ represents a solution to a problem consisting of four firing units, in which firing units W_1 and W_3 are allocated to target T_2, W_2 is allocated to T_3, and W_4 is allocated to T_1. Hence, a mapping from the vector to a matrix of decision variables is provided, so that X_{14}, X_{21}, X_{23}, and X_{32} take on the value 1 for the above example, while the remaining decision variables take on the value 0.

4.2.2 An Exhaustive Search Algorithm

As the name implies, exhaustive search algorithms exhaustively generate and test all feasible solutions. When all feasible solutions have been tested, the algorithm returns the optimal solution, i.e. the solution that minimizes (for minimization problems, e.g. the static target-based weapon allocation problem as formulated in equation 3.4) or maximizes (for maximization problems, e.g. the static asset-based weapon allocation problem as formulated in equation 3.7) the objective function value. A pseudo code description of the implemented algorithm for static target-based weapon allocation is given in algorithm 4.1, while the corresponding pseudo code for static asset-based weapon allocation is shown in algorithm 4.2. Since there are $|\mathbf{T}|^{|\mathbf{W}|}$ feasible solutions to test, this is the number of times the method $createSolution()$ will be called. The method takes a natural number as input and creates a solution by converting the input data from the decimal number base (i.e. with radix 10) to radix $|\mathbf{T}|$. As

Algorithm 4.1 Exhaustive search algorithm for static target-based weapon allocation

Input: A vector $\mathbf{V}$ of target values and a matrix $\mathbf{P}$ of kill probabilities.
Output: The optimal solution sol_{best}.

1: $F_{best} \leftarrow \infty$
2: $count \leftarrow 0$
3: **while** $count < |\mathbf{T}|^{|\mathbf{W}|}$ **do**
4: $sol \leftarrow createSolution(count)$
5: $F_{sol} \leftarrow 0$
6: **for all** i such that $1 \leq i \leq |\mathbf{T}|$ **do**
7: $P_{survival} \leftarrow 1.0$
8: **for all** k such that $1 \leq k \leq |\mathbf{W}|$ **do**
9: $P_{survival} \leftarrow P_{survival} \times (1 - P_{ik})^{X_{ik}}$
10: **end for**
11: $F_{sol} \leftarrow F_{sol} + V_i \times P_{survival}$
12: **end for**
13: **if** $F_{sol} < F_{best}$ **then**
14: $F_{best} \leftarrow F_{sol}$
15: $sol_{best} \leftarrow sol$
16: **end if**
17: $count \leftarrow count + 1$
18: **end while**
19: **return** sol_{best}

an example, for a problem where $|\mathbf{T}| = 4$ and $|\mathbf{W}| = 3$, the decimal value 5 generates the string 11 when converted. This string is left-padded with 0's until the length of the generated string is equal to $|\mathbf{W}|$, so that the string in our case becomes 011. In order to match the representation described above, the value of each element in the generated string is increased with 1 so that the corresponding feasible solution in our case becomes $[1,2,2]^{\mathsf{T}}$, indicating that the first firing unit should be allocated to T_1, and the second and third firing units should be allocated to T_2. This is directly mapped to the corresponding matrix of decision variables, as explained in section 4.2.1. The objective function value is calculated as specified in equation 3.4 for target-based weapon allocation and as specified in equation 3.7 for asset-based weapon allocation. If the objective function value for the current solution is better than the current best, the solution is stored as the current best solution. This is repeated for all $|\mathbf{T}|^{|\mathbf{W}|}$ feasible solutions. Hence, since the algorithms explicitly enumerate all feasible solutions, the current best solution will in the end be the optimal solution to the problem. Due to the bad scaling of the exhaustive search algorithm, it is obvious that it cannot be used in real-time for arbitrarily large problem sizes (i.e. the number of targets, firing units, and defended assets). Nevertheless, it is easy to give good estimates of how long search time the algorithm needs. Hence, if the demanded search time is short enough, it is preferable to use exhaustive search since the algorithm is guaranteed to find a global optimal solution.

4.2.3 A Maximum Marginal Return Algorithm

As briefly discussed in chapter 3, the maximum marginal return algorithm is a simple greedy algorithm, suggested already in den Broeder et al. (1959), which for the special case of the static target-based weapon allocation problem where $P_{ik} = P_i$ can be shown to be optimal (see Hosein (1990)). For the more general version of the static target-based weapon allocation problem studied in this thesis, it is a greedy heuristic algorithm, i.e. it is not guaranteed to find the optimal solution. The main advantages of the algorithm's greedy nature are that it is very fast and simple to implement. Basically, the maximum marginal return algorithm works by assigning firing units sequentially to the target which has the maximum decrease on the objective cost, i.e. the total expected target value of surviving targets. When the first firing unit 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 firing unit, and so on, until all firing units have been allocated to targets. The algorithm is described with pseudo code in algorithm 4.3.

Looking at the complexity of the algorithm, $|\mathbf{T}|$ marginal returns are calculated for each iteration (i.e. each firing unit). Moreover, for each iteration, the target value will be updated for the target maximizing the marginal return. Since there are $|\mathbf{W}|$ firing units, this is the number of iterations that will be run. Hence, the computational complexity becomes $O(|\mathbf{W}| \times |\mathbf{T}|)$.

Algorithm 4.2 Exhaustive search algorithm for static asset-based weapon allocation

Input: A vector $\boldsymbol{\pi}$ of lethality probabilities, a vector $\boldsymbol{\omega}$ of protection values, sets of target aims, and a matrix $\mathbf{P}$ of kill probabilities.
Output: The optimal solution sol_{best}.

1: $J_{best} \leftarrow 0$
2: $count \leftarrow 0$
3: **while** $count < |\mathbf{T}|^{|\mathbf{W}|}$ **do**
4: $sol \leftarrow createSolution(count)$
5: $J_{sol} \leftarrow 0$
6: **for all** j such that $1 \leq j \leq |\mathbf{A}|$ **do**
7: $P_{survival}(A_j) \leftarrow 1.0$
8: **for all** i such that $1 \leq i \leq |\mathbf{T}|$ **do**
9: **if** $T_i \in \mathbf{G}_j$ **then**
10: $P_{survival}(T_i) \leftarrow 1.0$
11: **for all** k such that $1 \leq k \leq |\mathbf{W}|$ **do**
12: $P_{survival}(T_i) \leftarrow P_{survival}(T_i) \times (1 - P_{ik})^{X_{ik}}$
13: **end for**
14: $P_{survival}(A_j) \leftarrow P_{survival}(A_j) \times (1 - \pi_i \times P_{survival}(T_i))$
15: **end if**
16: **end for**
17: $J_{sol} \leftarrow J_{sol} + \omega_j \times P_{survival}(A_j)$
18: **end for**
19: **if** $J_{sol} > J_{best}$ **then**
20: $J_{best} \leftarrow J_{sol}$
21: $sol_{best} \leftarrow sol$
22: **end if**
23: $count \leftarrow count + 1$
24: **end while**
25: **return** sol_{best}

Algorithm 4.3 Maximum marginal return algorithm for static target-based weapon allocation

Input: A vector $\mathbf{V}$ of target values and a matrix $\mathbf{P}$ of kill probabilities.
Output: The greedily generated allocation.

```
1: for all k such that 1 ≤ k ≤ |W| do
2:    highestValue ← −∞
3:    allocatedTarget ← 0
4:    for all i such that 1 ≤ i ≤ |T| do
5:       value ← V_i × P_ik
6:       if value > highestValue then
7:          highestValue ← value
8:          allocatedTarget ← i
9:       end if
10:   end for
11:   allocate firing unit W_k to target T_allocatedTarget
12:   V_allocatedTarget ← V_allocatedTarget − highestValue
13: end for
14: return allocation
```

Since target values are central for the maximum marginal return algorithm, it is only directly applicable to target-based weapon allocation. However, as briefly described in Metler and Preston (1990), it can also be used for asset-based weapon allocation by approximating the static asset-based weapon allocation problem (equation 3.7) with its target-based counterpart (equation 3.4). By doing this, the obtained feasible solution for the target-based problem can be used as solution for the asset-based version. In order for this to be possible, it must be decided how to estimate the target values. Metler and Preston (1990) suggest to estimate target values by uniformly spreading out the protection value of a defended asset among the targets aimed for it. In Hosein (1990), it is instead suggested that a target is assigned the protection value of the target it is aimed for, i.e. that it does not matter how many targets that are aimed for the defended asset. It should be noted that the produced solutions not necessarily are good for static asset-based weapon allocation even though they might be near-optimal for static target-based weapon allocation. It is in Hosein (1990) argued that such approximations are suitable for situations with a so called *strong defense*, i.e. situations where the ratio between firing units and targets is high so that all defended assets can be protected, and where the expected number of targets surviving the weapon engagements is small (much less than one). On the contrary, they are not suitable for situations with a *weak defense*, since the approximation will work reasonably good for the defended assets whose

attackers are engaged, but bad for the remaining defended assets that cannot be protected due to the lack of firing units (Hosein, 1990).

4.2.4 An Enhanced Maximum Marginal Return Algorithm

What here will be referred to as the enhanced maximum marginal return algorithm (the author's terminology) is quite similar to the standard maximum marginal return algorithm. The difference between the two algorithms is that while it in the standard version is predetermined which firing unit that should be allocated next, it is in the enhanced maximum marginal return algorithm 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. This version of the maximum marginal return algorithm is briefly presented in Julstrom (2009). We have implemented this algorithm based on the description in Julstrom (2009), and the pseudo code for the algorithm is given in algorithm 4.4. In the first iteration $it = 1$, $|\mathbf{W}| \times |\mathbf{T}|$

Algorithm 4.4 Enhanced maximum marginal return algorithm for static target-based weapon allocation (adapted from (Julstrom, 2009))

Input: A vector $\mathbf{V}$ of target values and a matrix $\mathbf{P}$ of kill probabilities.
Output: The greedily generated allocation.

```
1: for all it such that 1 ≤ it ≤ |W| do
2:    highestValue ← −∞
3:    allocatedTarget ← 0
4:    allocatedWeapon ← 0
5:    for all k such that 1 ≤ k ≤ |W| do
6:       for all i such that 1 ≤ i ≤ |T| do
7:          value ← V_i × P_ik
8:          if value > highestValue then
9:             highestValue ← value
10:            allocatedWeapon ← k
11:            allocatedTarget ← i
12:         end if
13:      end for
14:   end for
15:   assign firing unit W_allocatedWeapon to target T_allocatedTarget
16:   V_allocatedTarget ← V_allocatedTarget − highestValue
17: end for
18: return allocation
```

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, in each of the $|\mathbf{W}|$ steps, the algorithm considers assigning each of the unallocated weapons to each of the $|\mathbf{T}|$ targets, so that the total number of considerations become

$$(|\mathbf{W}| + (|\mathbf{W}| - 1) + \ldots + 2 + 1) \times |\mathbf{T}| = \frac{|\mathbf{W}|(|\mathbf{W}| + 1)}{2} \times |\mathbf{T}|. \qquad (4.1)$$

Therefore, the time complexity of the enhanced maximum marginal return algorithm becomes $O(|\mathbf{W}|^2|\mathbf{T}|)$. Hence, the complexity is somewhat worse than that of the standard maximum marginal return algorithm, but it still scales very good to large problems, compared to e.g. enumeration algorithms such as exhaustive search.

4.2.5 A Naïve Random Search Algorithm

Algorithm 4.5 Random search algorithm for static target-based weapon allocation

Input: A vector $\mathbf{V}$ of target values, a matrix $\mathbf{P}$ of kill probabilities, and the time allowed for search.
Output: sol_{best}, which is the best random solution found during search.

```
1: F_best ← ∞
2: while termination criteria not met do
3:    sol ← ∅
4:    for all k such that 1 ≤ k ≤ |W| do
5:       randomly select an integer i in {1, ..., |T|}
6:       add allocation of firing unit W_k to target T_i to sol
7:    end for
8:    F_sol ← calculateFValue(sol)
9:    if F_sol < F_best then
10:      F_best ← F_sol
11:      sol_best ← sol
12:   end if
13: end while
14: return sol_best
```

The algorithm implementations presented in this chapter so far are deterministic in the sense that they always will produce the same solution when run on a specific problem instance (the different algorithms are not guaranteed to return the same solution, but the solution returned by e.g. the maximum

marginal return algorithm will always be the same for the specific problem instance). In some sense, the complete opposite to this kind of approaches is pure random search, i.e. to create (feasible) solutions randomly. Along these lines, a very simple random search algorithm has been implemented as a baseline to which more sophisticated algorithms can be compared. The algorithm generates random allocations as fast as possible, selecting the target to which a firing unit should be allocated from the set $\{1, \ldots, |\mathbf{T}|\}$ in a uniformly distributed manner. When a complete solution has been generated, its objective function value is calculated. If the objective function value of a newly generated solution is better than the objective function value of the best generated solution so far, the new solution is stored as the current best solution. This is repeated until no more time remains. When this happens, the current best solution is given as output from the algorithm. It shall however be noted that this somewhat naïve algorithm is only intended for being used for benchmarking purposes. Pseudo code for the static target-based version of this algorithm is given in algorithm 4.5, and the static asset-based formulation only requires minor changes of the algorithm.

4.2.6 A Greedy Local Search Algorithm

Algorithm 4.6 Maximum marginal return algorithm combined with local search for static asset-based weapon allocation

Input: A vector $\boldsymbol{\pi}$ of lethality probabilities, a vector $\boldsymbol{\omega}$ of protection values, sets of target aims, a matrix $\mathbf{P}$ of kill probabilities, and the time allowed for search.
Output: The heuristic solution sol_{best}.

```
1: sol_best ← MMR()
2: J_best ← CalculateJValue(sol_best)
3: while termination criteria not met do
4:    sol_neighbor ← SearchLocal(sol_best)
5:    J_neighbor ← CalculateJValue(sol_neighbor)
6:    if J_neighbor > J_best then
7:       sol_best ← sol_neighbor
8:       J_best ← J_neighbor
9:    end if
10: end while
11: return sol_best
```

Another implemented algorithm that makes use of random search takes the solution generated by the standard or advanced version of the maximum marginal return algorithm as input and improves it using local search. The implemented algorithm creates neighbor solutions by swapping two positions

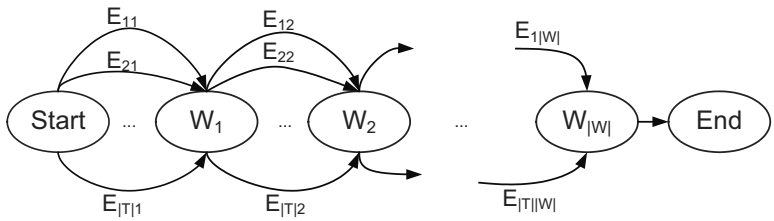


Figure 4.4: A directed graph representation of the static weapon allocation problem

selected at random in the solution vector, but other kinds of randomly selected neighborhood solutions can also be thought of. The algorithm described with pseudo code in algorithm 4.6 is intended for static asset-based weapon allocation, but can of course easily be adapted for static target-based weapon allocation.

Obviously, the quality of the solution generated by algorithm 4.6 will always be at least as good as the quality of the solution returned by the maximum marginal return algorithm used for obtaining the initial solution.

4.2.7 An Ant Colony Optimization Algorithm

Among the more complicated algorithms for static target-based weapon allocation, an ant colony optimization algorithm has been implemented. This algorithm makes use of the metaheuristic ant colony system proposed by Dorigo and Gambardella (1997). The implementation builds upon the algorithm described in Lee et al. (2002a) and Lee et al. (2002b). Central for the implemented algorithm is a directed graph representation of the static target-based weapon allocation problem, depicted in figure 4.4. The ants move stochastically through this graph, biased by a pheromone model which is modified at runtime by the ants.

The ants start out in the ant colony. From the colony, $|\mathbf{T}|$ edges lead to node W_1. From W_1, $|\mathbf{T}|$ edges lead to node W_2, etc., ending in the "food node" $W_{|\mathbf{W}|}$. Hence, the edge E_{ik} taken from the preceding node to node W_k represents the target T_i to which weapon W_k should be allocated, and so, the complete path taken by an ant corresponds to a feasible solution to the static target-based weapon allocation problem. The pseudo code used for determining which ways the ants take through the graph is described in algorithm 4.7.

In the $Initialization()$-method, heuristic information η_{ik} is assigned to each edge E_{ik} according to the product $V_i \times P_{ik}$, so that edges with higher products get a higher initial heuristic value (improving their chances of being selected). Initial pheromone τ_{ik} is also assigned uniformly to each edge.

Algorithm 4.7 Ant colony optimization algorithm for static target-based weapon allocation

Input: A vector $\mathbf{V}$ of target values, a matrix $\mathbf{P}$ of kill probabilities, and time allowed for search.
Output: sol_{best}, which is the best solution found during search.

```
1: Initialization()
2: while termination criteria not met do
3:    for ant ← 1 to nrOfAnts do
4:       for k ← 1 to |W| do
5:          q ← U[0,1]
6:          if q ≤ q0 then
7:             select i according to arg max_{i∈{1,...,|T|}} (τ_ik^α × η_ik^β)
8:          else
9:             select i according to roulette-wheel selection
10:         end if
11:         let current ant use edge E_ik
12:         UpdateHeuristicInfo()
13:      end for
14:      calculate fitness for chosen path and replace sol_best if better
15:      ResetHeuristicInfo()
16:      LocalPheromoneUpdate()
17:   end for
18:   GlobalPheromoneUpdate()
19: end while
20: return sol_best
```

When an ant decides which edge to take from the current node to node W_k, the pseudo random proportional rule given in equation 4.2 is used.

$$i = \begin{cases} \arg\max_{i \in \{1,\ldots,|\mathbf{T}|\}} (\tau_{ik}^{\alpha} \eta_{ik}^{\beta}) & q \leq q_0 \\ s & \text{otherwise.} \end{cases} \tag{4.2}$$

Here, α and β are parameters specifying the importance of pheromone and heuristic information, respectively. Moreover, q_0 is a threshold regulating the balance between so called *exploitation* and *biased exploration*. Furthermore, q is a random number drawn from the uniform distribution $U[0,1]$, while s is an index selected using roulette wheel selection, where the probability that s is selected becomes:

$$P_s = \frac{\tau_{sk}^{\alpha} \eta_{sk}^{\beta}}{\sum_{l \in \{1,\ldots,|\mathbf{T}|\}} \tau_{lk}^{\alpha} \eta_{lk}^{\beta}}. \tag{4.3}$$

That is, the target T_i to which weapon W_k should be allocated is greedily selected to be the target (edge) maximizing the product of the pheromone and heuristic information, given that $q \leq q_0$. Otherwise, the target is determined through roulette wheel selection, where the amount of pheromone and heuristic information determines the probability for a target (edge) to be selected.

When an ant reaches a new node (i.e. a weapon has been allocated to a target T_i), it updates its local target values as:

$$V_i = V_i \times (1 - P_{ik})^{X_{ik}}, \tag{4.4}$$

(where $X_{ik} = 1$ if and only if edge E_{ik} has been used), affecting the ant's heuristic information regarding the remaining edges so that it becomes less likely that more weapons are assigned to the selected target in the ant's generated solution. When the ant reaches the food node $W_{|\mathbf{W}|}$, the heuristic information is in $ResetHeuristicInfo()$ reinitialized in the same way as in the $Initialization()$-method, and a local update of pheromone is in $LocalPheromoneUpdate()$ applied on the recently used edge as:

$$\tau_{ik} = (1 - \varphi)\tau_{ik} + \varphi\tau_0, \tag{4.5}$$

where $\varphi \in [0,1]$ is a constant regulating the pheromone evaporation. This local update is used for diversifying the paths taken by ants within the same iteration. After all ants within an iteration have reached $W_{|\mathbf{W}|}$, a global pheromone update is in $GlobalPheromoneUpdate()$ performed according to equation 4.6. In this update, $\rho \in [0,1]$, while $\Delta\tau_{ik}^{best}$ is assigned the value $1/F^{best}$ if edge E_{ik} is part of the iteration-best solution, and 0 otherwise.

$$\tau_{ik} = (1 - \rho)\tau_{ik} + \rho\Delta\tau_{ik}^{best}. \tag{4.6}$$

In this way, the level of pheromone is increased for the path taken by the iteration-best ant, increasing the likelihood for this path to be taken by ants in future iterations. The global best solution is all the time kept track of, so that when no time remains, the best solution found during the search time is returned by the algorithm.

4.2.8 A Genetic Algorithm

Another metaheuristic algorithm that has been implemented is a genetic algorithm. The implemented genetic algorithm for static asset-based weapon allocation is described with pseudo code in algorithm 4.8, and it should be straightforward how to modify it to be used for static target-based weapon allocation. In the first step, an initial population Pop consisting of $nrOfIndividuals$ is

Algorithm 4.8 Genetic algorithm for static asset-based weapon allocation

Input: A vector $\boldsymbol{\pi}$ of lethality probabilities, a vector $\boldsymbol{\omega}$ of protection values, sets of target aims, a matrix $\mathbf{P}$ of kill probabilities, and the time allowed for search.
Output: The heuristic solution sol_{best}.

```
1: fitness_best ← −∞
2: Pop ← GenerateInitialPopulation(nrOfIndividuals)
3: while termination criteria not met do
4:    for l ← 1 to nrOfIndividuals do
5:       J_l ← CalculateFitness(Pop(l))
6:       if J_l > fitness_best then
7:          sol_best ← Pop(l)
8:          fitness_best ← J_l
9:       end if
10:   end for
11:   Pop′ ← SelectionAndCrossover(Pop)
12:   Pop ← Mutate(Pop′)
13: end while
14: return sol_best
```

created. This is accomplished through the generation of a vector of length $|\mathbf{W}|$, where each element W_k randomly is assigned an integer value in $\{1,\ldots,|\mathbf{T}|\}$. In each generation, each individual in the current population is evaluated by determining its objective function value. Each individual is thus assigned a fitness value that can be used in the selection and recombination step. After this, deterministic tournament selection is used to determine which individuals in population Pop that should be used as parents for Pop', i.e. two individuals are picked at random from Pop and the one with best fitness value is selected. The reason for using this simple selection mechanism is that it is faster than more advanced selection mechanisms such as roulette-wheel selection. When two parents have been selected from Pop, one-point crossover (illustrated in figure 4.5) is applied at a randomly selected position $k \in \{1,\ldots,|\mathbf{W}|\}$. This crossover operator generates two individuals that become members of Pop'. This is repeated until there are $nrOfIndividuals$ in Pop'. Thereafter, a mutation operator is applied 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 $|\mathbf{T}|^{-1}$ 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 a termination condition is fulfilled, in our case that 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 solution of which weapons that should be allocated to which targets.

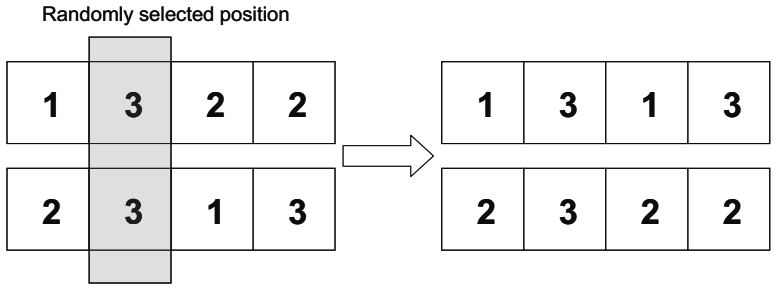


Figure 4.5: Illustration of the one-point crossover operator

A small variation of the genetic algorithm which also has been implemented is to seed one or more individuals in the initial population with a solution that is generated using the greedy maximum marginal return algorithm (presented in section 4.2.3) or its enhanced version (presented in section 4.2.4). Obviously, at least a part of the valuable search time available is used to generate the individual using greedy search instead of the even faster random generation. However, this sacrifice can be very valuable, since the seeded individual(s) most likely will have a better fitness and thereby can guide the evolution process, so that better individuals are created more quickly.

4.2.9 A Particle Swarm Optimization Algorithm

In the implemented particle swarm optimization algorithm, a swarm consisting of $nrOfParticles$ particles is created. Each particle p_j has in every time step t an associated position $\vec{x_j}^t$ and a velocity $\vec{v_j}^t$. The search space is $|\mathbf{W}|$-dimensional, so that each dimension represents a firing unit W_k. Each particle is also associated with a memory $\vec{b_j}^t$ storing the particle's personal best position. Moreover, the swarm's global best position is stored in the vector $\vec{g}^t$. In the initialization phase (see algorithm 4.9), each particle gets an initial position $\vec{x_j}^0$ and velocity $\vec{v_j}^0$. The elements in the initial position vectors are natural numbers randomly distributed between 1 and $|\mathbf{T}|$, while the elements in the initial velocity vectors are real numbers uniformly distributed in the interval $U[-|\mathbf{T}|,|\mathbf{T}|]$.

Algorithm 4.9 Particle swarm optimization algorithm for static asset-based weapon allocation

Input: A vector $\boldsymbol{\pi}$ of lethality probabilities, a vector $\boldsymbol{\omega}$ of protection values, sets of target aims, a matrix $\mathbf{P}$ of kill probabilities, and the time allowed for search.
Output: The heuristic solution sol_{best}.

```
1: Initialization()
2: while termination criteria not met do
3:    for l ← 1 to nrOfParticles do
4:       J_l ← CalculateFitness(x⃗_l)
5:       if J_l = CalculateFitness(g⃗) then
6:          p_l ← reinitialize()
7:       else
8:          if J_l > CalculateFitness(b⃗_l) then
9:             b⃗_l ← x⃗_l
10:            if J_l > CalculateFitness(g⃗) then
11:               g⃗ ← x⃗_l
12:            end if
13:         end if
14:      end if
15:   end for
16:   for l ← 1 to nrOfParticles do
17:      v⃗_l ← UpdateVelocity(p⃗_l)
18:      x⃗_l ← UpdatePosition(p⃗_l)
19:   end for
20: end while
21: return g⃗
```

The position and velocity for each particle is in the implemented algorithm updated according to equation 4.7 and 4.8.

$$\vec{v_j}^{t+1} = \omega \vec{v_j}^{t} + c_1 \vec{r_1}^{t} \circ (\vec{b_j}^{t} - \vec{x_j}^{t}) + c_2 \vec{r_2}^{t} \circ (\vec{g}^{t} - \vec{x_j}^{t}) \tag{4.7}$$

$$\vec{x_j}^{t+1} = \vec{x_j}^{t} + \vec{v_j}^{t+1} \tag{4.8}$$

In equation 4.7, ω is a parameter (not to be confused with the vector of protection values) 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. These constants are often referred to as the *cognitive component* and the *social component*, respectively. $\vec{r_1}^{t}$ and $\vec{r_2}^{t}$ are vectors with random numbers drawn uniformly from $[0,1]$. Moreover, the $\circ$-operator denotes the *Hadamard product*, i.e. element-by-element multiplication of the vectors.

In the standard version of particle swarm optimization, it is assumed that the particles take on values in a continuous search space. However, as we know, only integer solutions are feasible for the static weapon allocation problems studied here. The method that has been used for making it possible to apply particle swarm optimization to the integer problem of weapon allocation is to round off the obtained positions (dimension by dimension) to the nearest integer value after the position update specified in equation 4.8. This is in accordance to how standard, i.e. continuous, particle swarm optimization is adapted to integer programming problems in Laskari et al. (2002), and is referred to as *variable truncation* by Wahde (2008). Another problem that must be handled is what to do when particles fly outside the bounds of the search space. When this happens, the position and velocity values of the element for which the problem occurred are reinitialized. Moreover, to avoid problems with premature convergence to local optima, the position and velocity vectors are reinitialized for particles rediscovering the current best solution.

4.3 Discussion

An interesting observation worth noticing is that the maximum marginal return algorithm (and also the enhanced maximum marginal return algorithm) always will return the same solution when run on a specific problem instance, i.e. the algorithm is deterministic. This can be compared to most other algorithms for static weapon allocation algorithms presented in this chapter, which do not always return the same solution. Another note to make is that the firing units are assumed to be ordered in some way, since the first firing unit is allocated first, than the second, and so on. Different orderings of firing units may return different solutions.

Among the algorithms for threat evaluation and weapon allocation presented in this chapter, many of the algorithms are variations of ideas previously presented by other researchers (which is expected since the choice of algorithms is based on the literature surveys presented in chapter 3). In the following paragraphs, the implemented algorithms are compared to existing work, in order to highlight similarities and differences, and to more clearly state what is novel and what is not. The implemented Bayesian network is unique in the choice of random variables to use for threat evaluation, the structure of the network, and subsequently also the conditional probability tables used. Okello and Thoms (2003) have earlier used a Bayesian network for threat evaluation in the air defense domain, but they have used a smaller example Bayesian network with fewer nodes, and have instead of discretization used a linear Gaussian approximation of continuous variables. The developed Bayesian network is more similar to the discrete Bayesian network suggested in Runqvist (2004), but the parameters used for calculating capability and intent differ a lot. In the implemented fuzzy inference system, the input variables have been chosen to be as similar to the Bayesian network as possible. It has also been made as similar as possible to the descriptions provided in Liang (2006, 2007), so that the used membership functions and fuzzy inference rules have been selected in accordance to this. Among the implemented algorithms for weapon allocation, the described exhaustive search algorithm and the random search algorithm have to the best of the author's knowledge not been presented in previous literature. The maximum marginal return algorithm and the enhanced maximum marginal return algorithm have been implemented based on the descriptions given in den Broeder et al. (1959) and Julstrom (2009), respectively. The implemented greedy local search algorithm is loosely based upon an idea briefly discussed in Metler and Preston (1990). The idea of using ant colony optimization for weapon allocation is based on Lee and Lee (2005), but the main difference is that it here is used for global search instead of local search as originally proposed by Lee and Lee. Comparing the implemented genetic algorithm to previously existing genetic algorithms for static weapon allocation (e.g. Julstrom (2009) and Lee and Lee (2003)), the main differences are in the choice of selection and recombination operators. Those have here been selected to allow for real-time applications. For the same reason, we have also avoided to apply local search after recombination, which has been used in Lee and Lee (2003, 2005). Finally, the particle swarm optimization algorithm has been developed independently of any other research on weapon allocation. However, as discussed earlier, other descriptions of the use of particle swarm optimization algorithm for static target-based weapon allocation do exist. It is not entirely clear to the author how Zeng et al. (2006) have implemented their algorithm, but it seems clear that the same kind of representation has been used. However, they have applied local search in their implementation, which has not been used here. In Teng et al. (2008), the inertia weight is adaptively adjusted and

they apply another approach than the author to avoid that particles converge prematurely to local optima.

A promising algorithm for real-time weapon allocation which has not been implemented is the VLSN algorithm developed by Ahuja et al. (2007), briefly presented in section 4.2. The main reason for why it has not been implemented is that it to the best of the author's knowledge has not been described thoroughly enough to allow for reimplementation. The results presented in Ahuja et al. (2007) however look very promising, and we hope that the algorithm in the future will be implemented into the testbeds described in later chapters of this thesis.

4.4 Summary

In this chapter, implemented algorithms for real-time threat evaluation as well as static weapon allocation have been described. A Bayesian network has been developed for threat evaluation, and we have implemented an adapted version of the fuzzy logic approach presented by Liang (2007). For real-time weapon allocation, an exhaustive search algorithm has been presented which is guaranteed to return an optimal solution if run long enough, but the problem is that the computation time needed is too high for most problem sizes. For this reason, heuristic algorithms are needed, and pseudo code for a number of such algorithms have have been given in this chapter. Among these are the well-known maximum marginal return algorithm, and an enhanced version of this, as well as a genetic algorithm, a particle swarm optimization algorithm, and an ant colony optimization algorithm. Moreover, a simple random search algorithm has been implemented, which can be used a baseline to which other more sophisticated algorithms can be compared. The implemented algorithms have also been included in the testbeds which are presented in chapter 6.

Chapter 5
Performance Evaluation

In this chapter, the problem of evaluating the performance of TEWA systems and the components being part of such a system is studied. This problem is very relevant, due to the critical consequences the decisions made or recommended by a TEWA system can have. Despite this, research on performance evaluation of TEWA systems and their contained threat evaluation and weapon allocation algorithms is very sparse. In section 5.1 it is shown that performance evaluation is considered to be a very important topic within the information fusion community, but that the evaluation of high-level information fusion applications is considered as problematic. In section 5.2 it is discussed how threat evaluation algorithm can be evaluated, and a similar discussion is provided for weapon allocation in section 5.3. In section 5.4, the evaluation of TEWA systems is highlighted, and a novel performance evaluation metric is put forward.

5.1 Performance Evaluation and Information Fusion

As has been described earlier in this thesis, threat evaluation and weapon allocation can be seen as high-level information fusion problems. When it comes to high-level information fusion, it is not obvious how to evaluate the performance of developed algorithms and systems. This is illustrated by the following quote by Steinberg et al. (1999):

> "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 reuse."

More than ten years have passed since this statement was made. Many tracks have been devoted to performance evaluation on the annual international con-

ferences on information fusion, but most of the articles concern evaluation of low-level information fusion algorithms. Despite the seemingly large interest in the topic, it is in Liggins et al. (2009) argued that the field of information fusion has suffered from a lack of rigor with regard to the evaluation and testing of algorithms. This argument is strengthened by Llinas (2009), stating that although much research has been devoted to the development of new fusion algorithms, much less work has been put in to evaluation of how well developed algorithms work, and how they compare to alternative algorithms on common problems.

As made clear by Waltz and Llinas (1990), the formulation of standard measures of correctness for high-level information fusion processes is extremely important. However, as highlighted by the same authors:

> "Given these characteristics of situation and threat assessment, it is clear that there would be no single way to perform these processes. Unlike level 1 fusion processing, where correctness is easier to define and such a definition, in turn, provides a way to evaluate the quality of a given methodology (for position or identity estimation), correctness in assessing a situation or threat is a much less clear notion."

A problem related to the evaluation of information fusion processes is that such processes are either components of a system, or enhancements to a system, i.e. they rarely represent the whole system under test (Llinas, 2009), but rather are part of a larger system process. For this reason, if it is the the performance of the information fusion process as such that needs to be evaluated, other components in the system must be kept fixed during evaluation. According to Llinas (2009), criteria such as *computational complexity*, *time-critical performance*, and *adaptability* are applicable when evaluating all kinds of information fusion applications.

In a recent paper by van Laere (2009), it is observed that many researchers within the information fusion community have reflected upon that, generally, ground truth information is not available in practice. This makes performance evaluation harder, as most evaluation metrics within information fusion are relying of information regarding ground truth (van Laere, 2009). According to Yang et al. (2008), there are two types of performance evaluation; *comparative* and *absolute*. In comparative performance evaluation, a set of performance metrics are used to rank the algorithms or systems, while absolute performance evaluation is concerned with a particular algorithm or system (Yang et al., 2008). In the following section, evaluation of a more specific information fusion problem is studied, namely, the evaluation of threat evaluation algorithms.

5.2 Evaluating Threat Evaluation Algorithms

Since it exists many operational air defense systems with threat evaluation capabilities, and since calculated target values form an important foundation for

decisions on whether a target should be engaged or not, it becomes very relevant to raise the question of how algorithms for threat evaluation can be evaluated. According to Dall (1999), performance evaluation of threat evaluation algorithms is problematic, since suitable performance measures are hard to find. Actually, due to the immature level of research on threat evaluation, systematic comparisons of threat evaluation algorithms are lacking within open literature.

It is in Dall (1999) suggested that a realistic performance measure for automated threat evaluation is that it should perform with *accuracy* and *speed* comparable to a human expert. As argued by Dall (1999), automated threat evaluation algorithms that perform far worse than a human expert are not very useful. At the same time, they do not need to perform better than human experts. If the algorithm and the human perform on the same level and their judgments are correlated, the algorithm can be used for offloading, so that the expert can focus on the more difficult cases (Dall, 1999). On the other hand, if they perform on the same level and their judgments are uncorrelated, using them in combination will improve the overall performance (Dall, 1999) (this can be compared to the use of ensembles in the area of machine learning, where diversity among classifiers is aimed for, cf. Krogh and Vedelsby (1995)).

Unfortunately, the performance of human decision makers is not well characterized within the air defense domain (Dall, 1999). It is not trivial to measure the accuracy of a threat evaluation algorithm (or a decision maker) since this demands ground truth in some sense, i.e. information regarding how threatening a target really is. We will in section 5.4 discuss how such information can be obtained indirectly using the concepts of survivability and effectiveness, but in general, there is no such thing as a physical target value property that can be measured or obtained directly. However, it is (at least in theory) possible to measure how much target values calculated by threat evaluation algorithms deviate from target values estimated by human experts. In this way, the expert's estimates can be used as ground truth. Hence, one potential way to judge the accuracy of a threat evaluation algorithms is to measure the overall deviation from target values obtained from human experts on the same air defense scenarios. This kind of evaluation methodology is illustrated in figure 5.1.

There are a number of studies in which the output given by automated threat evaluation algorithms on a small amount of scenarios (typically only a single scenario) is analyzed and judged by, or compared to, human expert knowledge. In Liang (2006) and Liang (2007), the target values for three different air targets are displayed for a small number of discrete time steps, as calculated by a fuzzy logic algorithm. The resulting ranking are (very briefly) compared to the ranking given by experts for the same scenario. A similar evaluation approach is taken in Benavoli et al. (2007), where the calculated target values are shown for ten different time steps, during which incoming evidence streams are fed into the evidential-based threat evaluation algorithm. This idea is extended in Benavoli et al. (2009). There, the algorithm's calculated target values are shown for some different "extreme" cases (as judged by hu-

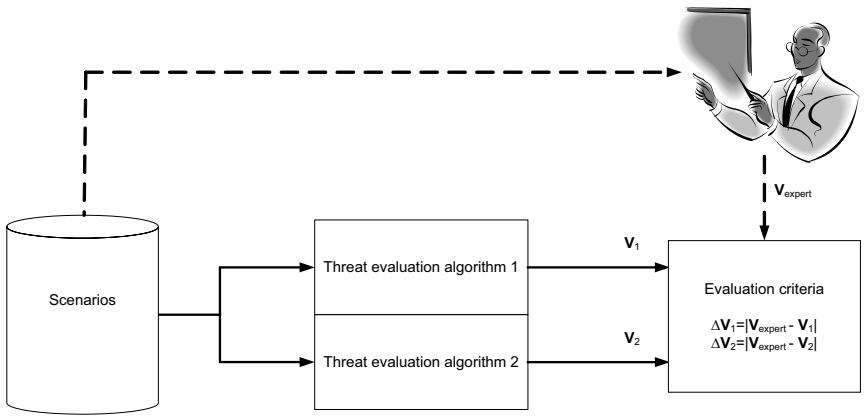


Figure 5.1: Illustration of how threat evaluation algorithms can be evaluated using expert knowledge.

man expert knowledge); a highly threatening situation, and a non-threatening situation. In this way, it is observed whether the computed target values are in accordance with expert judgments or not.

A problem with using this kind of evaluation is that threat ratings vary among experts. It has been shown that expert decision makers often disagree about the threat of individual aircraft (St. John et al., 2004b,a), making it hard to measure performance on air defense tasks involving substantial expert user judgment. This problem can partially be tackled by involving several experts in the studies, but the fundamental problem is that this kind of evaluation only can be based upon a fairly limited number of different scenarios, due to the need for expert judgments regarding the quality of the generated target values. Another problem with this kind of evaluation is that we then only strive for correlated judgments and thereby miss the opportunity of creating uncorrelated target value predictions. In other words, the developed threat evaluation algorithms can be used to offload the human operator, but the algorithms will most likely make the same kind of systematic errors as the operators make. Also, it is possible that developed algorithms can create target values that better reflect the threatening behavior of targets than human operators can. However, such algorithms will perform poorly according to the described performance metric, since the calculated target values will deviate from the operators' opinion.

For the above reasons, it is far from perfect to judge the accuracy of a threat evaluation algorithm only by using this kind of evaluation.

The time demanded by air defense decision makers to perform threat evaluation is according to an early study presented in Jorgensen and Strub (1979) dependent on the number of targets and on the rate at which the targets approach. In that study, the average time from target detection to manual weapon allocation was close to four minutes in simulated combat exercises for human air defense decision makers (Jorgensen and Strub, 1979). It should be noted that this is an old study, so it is not clear how the obtained results can be translated to more modern settings. In Khosla (2001), it is mentioned that a cycle of threat evaluation and weapon allocation is allowed to take a few seconds at most.

Except for the accuracy and time requirements on threat evaluation algorithms, other criteria of importance can be identified as well. Depending on in which kind of environment a threat evaluation algorithms should operate, there might be restrictions on the amount of memory capacity available. Hence, an algorithm's *memory complexity* is of importance. There is often a trade-off between time and memory complexity, in which the execution time can be decreased by increasing the memory consumption, and vice versa. Even though an algorithm is fast and demands only small amounts of memory for scenarios involving a few targets and defended assets, it may be the case that it requires much longer execution time and higher memory consumption as the number of targets and defended assets increases. Hence, the algorithm's *scalability* is of importance. Another criterion of interest is the algorithm's *sensitivity* to changes in the input parameters. If the calculated target values are drastically affected by small changes in the input parameters, there is a risk of very unstable target values. This is not wanted, since such oscillating behavior can have negative impact on the decision making process (Allouche, 2005).

There are also more soft properties of the used threat evaluation algorithms that can play an important role. The *adaptivity* of an algorithm has to do with how hard or easy it is to add new parameters, rules, etc. to the threat evaluation algorithm, while the algorithm's *transparency* is a (subjective) measure of how transparent it is to a human operator. As highlighted in e.g. Morrison et al. (1998); Liebhaber and Feher (2002a), it is important with explanation-based capabilities in air defense systems so that the decision maker can obtain an understanding of why a target is considered as threatening or not. According to Davis et al. (1977), ability to generate explanations is crucial for user acceptance of decision support systems.

5.3 Evaluating Weapon Allocation Algorithms

When it comes to performance evaluation of weapon allocation algorithms, such evaluation is very related to how algorithms for other kinds of optimization problems traditionally are evaluated.

For exact algorithms, i.e. optimization algorithms that always produce optimal solutions, the obvious criterion to use for performance evaluation is time-efficiency (Rardin and Uzsoy, 2001). Another criterion that can be of interest is the amount of memory needed to create the solution. In order to study the performance of heuristic algorithms developed for complex optimization problems, empirical analysis often needs to be used. Such an analysis involves experiments in which the heuristic methods are applied to a collection of problem instances. The performance measures for experiments on heuristic algorithms generally involve comparing the observed solution quality and computational time to exact solution methods (Rardin and Uzsoy, 2001). However, a general problem with heuristic algorithms is that the quality of their generated solutions is hard to evaluate, due to the absence of optimal algorithms for large problem sizes (Ahuja et al., 2007). For this reason, we often have to evaluate the quality of solutions to large-scale problems without access to the optimal solution (Rardin and Uzsoy, 2001). Figure 5.2 is an example of how two heuristic weapon allocation algorithms can be compared to each other, given that it is possible to calculate the optimal solution (obviously, this can easily be extended to allow for comparison of a larger number of algorithms). If this is not the case, the evaluation can instead be based upon which of the algorithms that has the lowest objective function value. A heuristic is said to dominate another if both the solution quality and the computational time are better, all other things being equal (Metler and Preston, 1990).

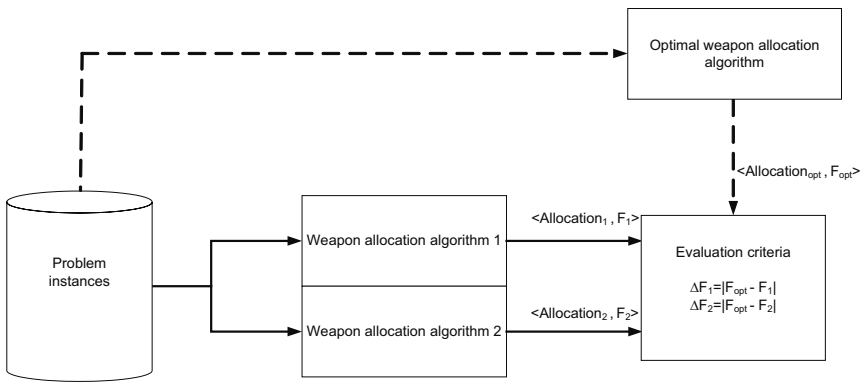


Figure 5.2: Illustration of the comparison of two weapon allocation algorithms.

The comparison of heuristic weapon allocation algorithms may seem trivial, but there are a number of aggravating circumstances. Firstly, there is a lack of unclassified real-world data sets on which to evaluate the algorithms (Julstrom, 2009). For this reason, it seems to be standard procedure to evaluate the algorithms on randomly generated synthetic data sets (see e.g. Lee et al. (2002b);

Lee and Lee (2005); Ahuja et al. (2007)). Secondly, most heuristic algorithms have a large degree of inbuilt randomness. It is therefore important to base the evaluation of such algorithms on a statistically sufficient number of runs, rather than a single one. Successive trials can produce quite different outcomes because of the impact of random choices along the way. Hence, several runs are generally preferable, in which different random number seeds controlling the evolution of computation are used, in order to get robust results (Rardin and Uzsoy, 2001). Thirdly, algorithms for static weapon-allocation only have a very limited time available for searching for good or optimal solutions, due to the real-time requirements. This is often overlooked when assessing the performance of weapon allocation algorithms, as discussed in section 3.3.2. Hence, the algorithms should be evaluated and tested on situations that are as realistic as possible, i.e. since the deadlines are very tight for the algorithms when applied in a real-world situation, they should not be allowed to run for longer time when being evaluated. The reason for this is that an algorithm that generates very good solutions after a long search time can return quite bad solutions when the available search time is shorter. For making comparisons fair, this also means that the algorithms should be allowed to run for equally long time. This may seem obvious, but there are many examples of where it only is ensured that algorithms are allowed to run for equally many iterations, despite that different algorithms may require different amounts of time for completing one iteration. Fourthly, there is a problem when trying to compare the performance of a weapon allocation algorithm to algorithms suggested by others. An important reason for this is that problem instances typically are randomly generated from (uniform) probability distributions. If an algorithm is tested on one or more problem instances generated at random from a uniform $[0,1]$-distribution in one experiment, and another algorithm is tested on equally many problem instances generated at random from a similar distribution, it is not obvious how to compare the performance of the algorithms. An alternative approach is to implement all algorithms on which the comparison should be based, but it is often the case that all the details of a weapon allocation algorithm are not published, making it hard to implement weapon allocation algorithms developed by other researchers (not to mention the amount of extra work needed). Lastly, as discussed earlier, it is in general hard to evaluate how accurate heuristic algorithms are, since the optimal solution can not be calculated for large-scale problems.

In order to address some of the problems related to performance evaluation of weapon allocation algorithms, success factors within the research area of machine learning can be studied. The so called UCI (an abbreviation for University of California, Irvine) machine learning repository (Asuncion and Newman, 2007) is and has been heavily used for empirical analysis, comparison, and benchmarking of different machine learning algorithms. As an indication of the impact of the repository, it is according to its web page (Asuncion and Newman, 2007) one of the most cited papers and resources within computer

science. Similar data sets exist for specific operations research problems such as the well-known traveling salesman problem (TSP), but no such benchmark data sets have earlier existed for weapon allocation research. Hence, it is here argued that a set of "standardized" data sets or problem instances for weapon allocation would benefit the comparison and benchmarking of static weapon allocation algorithms. Standardized challenge problems and associated data sets are according to Llinas (2009) a goal that the information fusion community should aim at.

Another successful concept within the machine learning community is the open source software workbench WEKA (Waikato Environment for Knowledge Analysis) (Hall et al., 2009). According to Hall et al. (2009), WEKA has been downloaded more than 1.4 million times since April 2000. WEKA clearly facilitates the performance evaluation and benchmarking of machine learning algorithms, since it includes open source implementations of a large set of different state of the art algorithms for machine learning. A workbench built on the same principles as WEKA would be beneficial for research on weapon allocation, as the possibility to benchmark weapon allocation algorithms against each other probably would lead to more open and active research, which in the long run can give new state of the art algorithms and may contribute to a better theoretical framework. As argued in Llinas (2009):

> "In an era of tight defense research budgets, algorithm-level sharable testbeds, is suspected and hoped, will become the norm for the DF community."

A testbed that incorporates the ideas of standard data sets and open source implementations for weapon allocation is presented in section 6.2.

5.4 Evaluating TEWA Systems

We have in section 5.2 and 5.3 discussed the problems of evaluating threat evaluation and weapon allocation algorithms, respectively. As could be seen, it is not obvious how to make such evaluations. Moreover, it has within the area of systems theory been an increasing understanding that the approach of analyzing complete systems by analyzing its subsystems in detail and in isolation does not hold in environments where systems become more and more interdependent and complex (van Laere, 2009). Since TEWA systems are complex, and since there are interdependencies between the threat evaluation and weapon allocation processes, it is not enough to in a reductionistic manner evaluate the different parts of a TEWA system independently of each other. In this section, metrics for evaluating the performance of complete TEWA systems is proposed. For being able to use these metrics properly, the use of computer-based simulations is called for. The original idea which this section is based on was briefly presented in Johansson and Falkman (2008c). That initial work has been extended upon in Johansson and Falkman (2009a,c). Except being used

for evaluation of TEWA systems, the suggested methodology can be used for evaluating individual parts of such systems (e.g. threat evaluation algorithms) as well.

5.4.1 A Performance Evaluation Metric for TEWA Systems

As the mission of air defense units most often is to ensure the survivability of defended assets, it makes sense to use this as an important criterion on which to evaluate the performance of a TEWA system. For this reason, we suggest the use of a quantitative *survivability* metric for evaluation of TEWA systems. The survivability of defended assets can be defined as:

$$S = \frac{\sum_{j=1}^{|\mathbf{A}|} \omega_j u_j}{\sum_{j=1}^{|\mathbf{A}|} \omega_j}, \tag{5.1}$$

where $|\mathbf{A}|$ as before is the number of defended assets, ω_j is the protection value of the defended asset A_j, and $\vec{u} \in \{0,1\}^{|\mathbf{A}|}$ is a binary vector defined as

$$u_j = \begin{cases} 1 & \text{if defended asset } A_j \text{ survives;} \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

Hence, the ratio between the protection value of surviving defended assets to the total protection value of all defended assets (surviving as well as destroyed) is calculated, where the protection values can be seen as weights that specify the relative importance of the defended assets. Obviously, a TEWA system has not been successful if all the defended assets which the air defense should protect are destroyed in an air defense scenario, while the system can be considered more successful if all the defended assets survive the attacks aimed at them.

Luckily enough, battles involving air attacks are not very frequent in most parts of the world. This is one reason for why air defense cannot be evaluated in real-world conflicts only. Another reason is of course that it is too late to only evaluate a system when it already is operational. For those reasons, it becomes natural to use computer-based simulations in which the survivability of different systems or system configurations can be tested. The use of simulations for performance evaluation of TEWA systems has earlier been described as a cost-effective approach by Chapman and Benke (2000).

By letting a complete TEWA system be situated within a simulated environment, the survivability of defended assets in an air defense scenario can be used as a relative metric for measuring the performance of the TEWA system, to which the performance of other TEWA system implementations can be compared. One possible advantage of this way of performance evaluation is that the need for elicitation of target values from human air defense experts is removed (although not completely, as discussed further in section 5.5), and that a more "holistic" evaluation is made possible (compared to the reductionistic approach

in which only the individual parts of the system are evaluated). Moreover, the use of computer-based simulations allow for more systematic comparisons on a larger set of scenarios, opening up for more robust performance evaluation.

A potential problem with using the survivability metric is that it does not explicitly take misguided allocations against targets with non-hostile intent into consideration. It is often assumed that identification friend or foe (IFF) transponders can be used to avoid blue force targets from becoming candidates for weapon allocation, but this does not always hold true. Cooperative IFF systems can at best be used to positively identify friends, but they do not positively identify enemies. If no reply to an IFF interrogation is received, it is likely that it is an enemy, but it is also possible that the target is a neutral or a friend without an operation IFF transponder (illustrated in some of the fratricide accidents during the second Gulf war (British Ministry of Defence, 2004)). This must be taken into consideration somehow when making the performance evaluation, in order to avoid fratricide and engagement of civilian targets. If the scenario is complex enough, e.g. the ratio of the number of threatening targets to the number of defensive firing units is high, the allocation of weapons to harmless targets will be reflected in the resulting survivability value, since valuable weapons will be "wasted" on non-hostile targets. However, if there are only a few targets and a lot of available firing units, the engagement of harmless targets will not decrease the survivability. This is a serious problem, since we cannot possibly judge an air defense system that engages friendly or civilian targets in non-threatening scenarios to be equally good as a system that does not engage any targets in the same scenario.

As a solution to this problem, we introduce a cost for each engagement, i.e. use of firing units. This does not only punish the engagement of non-hostile targets, but all kind of unnecessary engagements. The saving of surface-to-air missiles and other kinds of defensive resources is important, both for protection against possible future attacks and because of the high price of a single missile (Karasakal, 2008). Hence, we are in addition to the survivability metric also introducing a *resource usage cost* C_k for each firing unit W_k. This cost can either be assumed to be the same for all firing units, or vary between firing units (since certain types of firing units are more "expensive" to use than others). In the same way as we would like to maximize the survivability of the defended assets, we want to minimize the resource usage cost. The *effectiveness* E of a TEWA system on a specific air defense scenario can therefore be expressed as the difference among these quantities:

$$E = \alpha S - \beta C, \tag{5.3}$$

where C is the sum of all individual resource usage costs during the scenario, and α and β are adjustable weights (where $\alpha+\beta = 1$) that can be used to decide on the relative importance of survivability and resource usage. This effectiveness metric is therefore proposed as a performance measure for comparison of

different TEWA systems. The use of this effectiveness metric for evaluation of TEWA systems is illustrated in figure 5.3, and the choice of metric is discussed further in section 5.5.

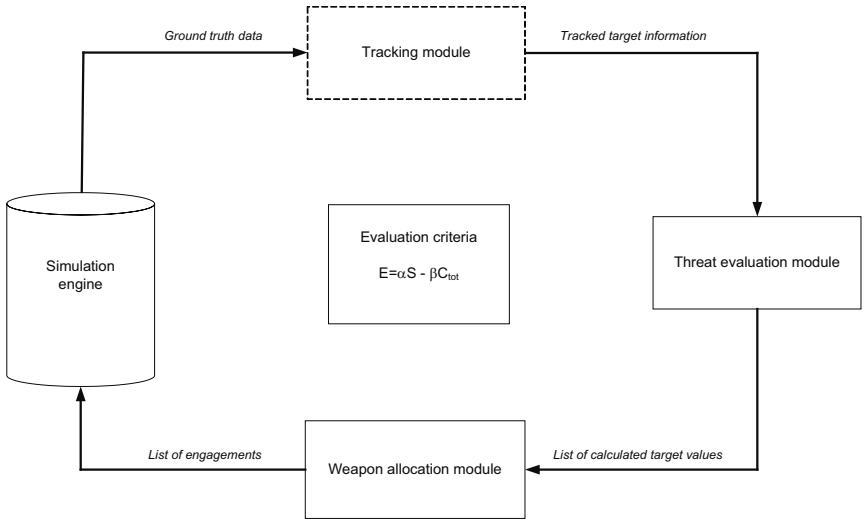


Figure 5.3: Illustration of the use of simulations and the suggested effectiveness metric to evaluate TEWA systems.

It is very important to have *robust* evaluations of the performance of TEWA systems, since the systems need to be able to operate under very varying operational conditions (e.g. scenarios with varying number of targets, firing units, political climates, etc.). Hence, in general, it is not enough to evaluate the effectiveness of a TEWA system on a very small restricted set of air defense scenarios. In order to be able to generalize the achieved results, the evaluation needs to be based on a large number of different air defense scenarios. Moreover, it also is necessary to run the same scenario more than once, since there are many non-deterministic outcomes of processes within a TEWA system, such as whether a target is detected or not, and whether a weapon-to-target engagement is successful or not. For this reason, the same scenario needs to be run multiple times in order to get statistically significant results. Hence, rather than just calculating a TEWA system's effectiveness on a scenario as shown in equation 5.3, we instead calculate it as an average of N scenario runs in accordance with equation 5.4:

$$E = \frac{\sum_{t=1}^{N} \alpha S_t - \beta C_t}{N}. \tag{5.4}$$

It should be noted that obtained effectiveness values are not very interesting in their own right, since what can considered to be a good effectiveness value

depends on the difficulty of the tested scenarios. Effectiveness should not be used as an absolute performance metric, but rather as a comparative metric suitable for comparing the performance of two or more TEWA systems. Except for comparing completely different TEWA systems, we can also use the effectiveness metric to compare different system configurations by keeping the remaining parts of the system fix. As an example, two threat evaluation algorithms can be compared by comparing the effectiveness of two systems where all parts are identical except for the used threat evaluation algorithms. Since everything else is identical, differences in effectiveness values are likely to be caused by the part which has been changed.

5.5 Discussion

As discussed in this chapter, it is not doable to compare the output from threat evaluation algorithms on more than a few data points, since the bottleneck of elicitation of target values from human experts soon becomes insurmountable. The use of the methodology of measuring the effectiveness in computer-based simulations of air defense scenarios is a first step for automating some parts of the evaluation, but it is important to realize that human experts still are needed for the evaluation, since such knowledge is needed for creating relevant air defense scenarios on which TEWA systems should be evaluated. Moreover, they are needed if the effectiveness of threat evaluation algorithms or TEWA systems are to be judged in an absolute manner (rather than only compared to other algorithms or TEWA systems). That human involvement always will and should be part of the evaluation process is clear, but as the situation looks today, there is still a bottleneck left in that many different air defense scenarios have to be created if the algorithms and systems are to be evaluated in a systematic manner. Even though there is good support for creating scenarios in commercial scenario generators such as STAGE Scenario (see chapter 6) and VR-Forces, it still takes quite some time to create realistic scenarios, making it hard to create as many scenarios as would be needed to produce robust and generalizable performance evaluations. We think that it would be possible to create templates of air defense scenarios on an even higher level of abstraction, which could be use to automatically (or at least semi-automatically) instantiate a large set of specific air defense scenarios of different sizes.

Looking at the proposed effectiveness metric defined in equation 5.3, we can see that S is bounded to be in the interval $[0,1]$, while C can grow with the size of the scenario (long scenarios involving a large number of firing units and targets can result in a larger value of C, since more engagements are likely to take place). To normalize the value of C does not seem meaningful, and hence, we must be aware of that the "optimal" value of E can be lower for long complex air defense scenarios than for short small-scale scenarios if the values of α and β are not adjusted accordingly. Another note to make is that we have

here chosen to use a simple linear model for calculating the effectiveness. Many other aggregations can be thought of, such as:

$$E = \frac{S^{\alpha}}{C^{\beta}}.$$

We are not arguing that the proposed effectiveness metric necessarily is the best possible way to aggregate the values of S and C, but rather that there is no point in replacing this simple model with a more complex one at this stage. The future may reveal that it is more fruitful to use more complex aggregation operators, but concrete experiments with the proposed simulation-based methodology have to be undertaken before such possible improvements can be tested.

To evaluate how well a TEWA system is performing based on the survivability of defended assets is clearly related to the objective function used to decide on how firing units should be allocated to hostile air targets in static asset-based weapon allocation. The main difference is that we here measure the total protection value of the surviving defended assets after a whole air defense scenario (instead of a single allocation phase), and that the actual scenarios are simulated. Moreover, the target values are not fixed but are continuously updated as the scenario evolves. Just as the maximization of the expected total protection value of surviving defended assets (static asset-based weapon allocation) has a counterpart in the minimization of the expected total target value of surviving targets (static target-based weapon allocation), it is for example possible to assign a value to each target type and introduce a metric that takes the total value of hostile targets shot down during a scenario into consideration (as well as the resource usage cost). The reason for why the survivability measure has been chosen is that it in the author's view is more close to the most common task of the air defense, i.e. to protect defended assets. Nevertheless, there are undoubtedly situations in which the destruction of high-valued targets can be of higher importance than the protection of defended assets with low protection values, so the value of destroyed target can certainly be a factor to weigh into the effectiveness of the TEWA system as well.

The approach presented by Chapman and Benke (2000) relies on the use of simulations to evaluate the survival probability of a single ship against one or more attacking cruise missiles. The main difference of our approaches therefore seems to be that we are interested in the protection of a set of defended assets and not only the survival of a single ship. Moreover, due to classified nature of their work, not many details are revealed on how the simulations are done.

5.6 Summary

In this chapter, evaluation of the performance of threat evaluation and weapon allocation algorithms has been discussed. As could be seen, neither of them are

straightforward to evaluate. Research on the topic of evaluation of threat evaluation algorithms is very sparse, as a natural consequence of the lack of research on threat evaluation in the first place. A problem related to performance evaluation of threat evaluation algorithms is that existing methodologies demand comparison to expert knowledge, which becomes a bottleneck that limits the robustness of obtained results since such knowledge elicitation is a hard task demanding a lot of time. A number of problems are associated with evaluation of weapon allocation algorithms as well, such as repeatability of experiments and the lack of unclassified problem instances on which to test developed algorithms. Another problem is that threat evaluation and weapon allocation algorithms only are evaluated independently of each other in a reductionistic manner, despite that air defense systems are very complex by nature. This has caused us to suggest a holistic methodology for comparing TEWA systems (or different configurations of TEWA systems) that make use of computer-based simulations. The proposed effectiveness metric is a combination of the survivability of defended assets, and the cost of the weapon resources used to defend the defended assets. This make it possible to evaluate algorithms without having to elicit target values from experts, to which calculated target values can be compared.

The idea of using simulations and the proposed effectiveness metric to evaluate TEWA systems has been implemented in the testbed STEWARD, which is described in section 6.1. Similarly, the ideas to overcome some of the identified problems with evaluating weapon allocation algorithms has resulted in the open source testbed SWARD, which is presented in section 6.2.

Chapter 6
Testbeds for Performance Evaluation of TEWA Systems

We will in this chapter describe two testbeds that have been developed for the purpose of performance evaluation. The testbed STEWARD presented in section 6.1 is a tool for demonstrating how different TEWA systems can be compared using the methodology based on effectiveness described in the last chapter. In order to overcome many of the current problems associated with the evaluation of weapon allocation algorithms, we are in section 6.2 presenting the open source testbed SWARD. Parts of STEWARD have earlier been described in Johansson and Falkman (2009a), while SWARD has been described in Johansson and Falkman (2010b).

6.1 STEWARD

In order to demonstrate the idea with using computer-based simulations in which survivability of defended assets and resource usage are measured for comparing the performance of TEWA systems, the testbed STEWARD (System for Threat Evaluation and Weapon Allocation Research and Development) has been developed. STEWARD consists of two main modules. The first of the two modules is the commercial simulation engine STAGE Scenario[1], in which it is possible to create and run dynamic and interactive military scenarios, involving individual platforms such as military aircrafts that can be equipped with missiles. In order to build scenarios, there is a scenario editor within STAGE which can be used to position the platforms, assign waypoints, etc. Moreover, scripts and missions can be assigned to the individual platforms. Examples of useful scripts that can be written are to make a platform change course upon reaching a pre-defined waypoint, or to lock on a specific target when getting in range of it. Missions are quite similar to the scripts, but are constructed on a higher

[1] http://www.presagis.com/products_services/products/ms/simulation/stage/

level, allowing for making platforms respond to events according to pre-defined rules or doctrines. A very simplified example of a mission in STAGE Scenario is shown in table 6.1. Basically, what this simple mission does is that it activates the entity's radar when it is created and launches surface-to-air missiles against hostile targets that comes into the weapon range of the entity, as long as there are any defensive weapons left.

Table 6.1: Simplified example of a mission in STAGE.

INIT		Activate Radar
IF	Weapon Count = 0	Stop Cycle On
IF	TrackHostile = true AND TrackRange $\leq$ 10000m	Launch SAM
END		

The sensor observations from STAGE Scenario regarding entities in the scenarios are communicated in real-time to the second module, which is our TEWA module, into which the algorithms for threat evaluation and weapon allocation algorithms described in chapter 4 have been integrated. Once firing units have been allocated to targets that are ranked as threatening, the proposed allocations are communicated back to STAGE Scenario, in which they are realized into engagements within the simulations. Hence, fired weapons become part of the simulation. For sending messages between the simulation engine and the TEWA module we make use of socket communication (the main reason for this design choice being that support for the reception and transmission of messages through sockets is built into STAGE, but also allows for changing the simulation engine in the future if needed). An alternative, more effective, design choice for communication could have been to use STAGE's shared memory, but this would have made STEWARD more dependent on STAGE. The socket is used for sending XML-observations regarding targets, defended assets, and firing units from STAGE to the TEWA module, as well as sending information back from the TEWA module to STAGE. This is illustrated in figure 6.1.

In order to allow for the integration of STAGE Scenario and the TEWA module, the functionality of STAGE Scenario has been modified and extended using user plugins which have been written in (unmanaged) C++. The created code has been linked into dynamic libraries (.dll-files), which are hooked into the the simulation engine used in STAGE Scenario at runtime. Except for sending the the sensed target information to the implemented TEWA module, the extended functionality also allow for execution of the firing orders received from the TEWA module. Moreover, it keeps track of the survivability of defended assets, and the weapon resource usage costs.

The two threat evaluation algorithms described in section 4.1 have been implemented into the TEWA module. The user can in a graphical user interface choose which threat evaluation algorithm that should be used (see figure 6.2).

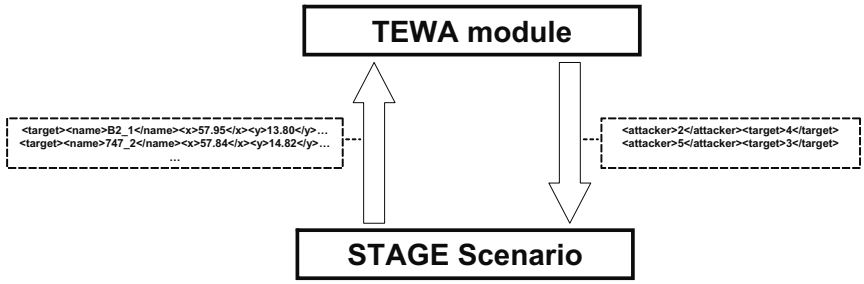


Figure 6.1: Illustration of the connection between the scenario engine and the TEWA module, together with examples of XML-messages sent between them.

The chosen threat evaluation algorithm is used to constantly calculate the threat value V_{ij} for each pair of targets and defended assets (using estimates of target speed, target type, and calculated distances between the target and the defended asset based on the communicated sensor observations). The estimated threat values are in their turn used to calculate the target value V_i for a target T_i. The calculated target values are presented in the graphical user interface, together with symbols showing the observed positions of defended assets and detected targets. Figure 6.2 illustrates an air defense situation consisting of four detected targets, in which T_1 and T_2 have been classified as neutral air targets, while T_3 and T_4 have been classified as hostile air targets. The lines shown are the targets' velocity vectors, showing their estimated speed and heading. Moreover, the values shown underneath the targets are their estimated target values. The graphical user interface has been designed coarsely, since the focus has been on the algorithmic aspects, so if a real air defense operator should use STEWARD, the interface has to be improved. The calculated target values are compared to an adjustable user-defined threshold τ. In the case of $V_i \geq \tau$, the target T_i is added to the set of targets $\mathbf{T}_{wa}$, which is the set of targets that become subject to weapon allocation. Hence, depending on the choice of the threshold setting, it is decided how high the target value V_i has to be in order for the target T_i to be considered as threatening enough for being a potential candidate for weapon allocation.

When the threat evaluation process has been performed and there are targets within $\mathbf{T}_{wa}$, a weapon allocation process is started in which firing units are allocated to the targets in $\mathbf{T}_{wa}$. The weapon allocation process is performed using the selected weapon allocation algorithm, which is chosen in the same manner as the threat evaluation algorithm (i.e. in the graphical user interface). The weapon allocation algorithms that can be chosen among are the static target-based weapon allocation algorithms described in section 4.2 (the asset-based versions have not been implemented here, since the testbed was not developed with this in mind, although it could be useful for evaluation also of

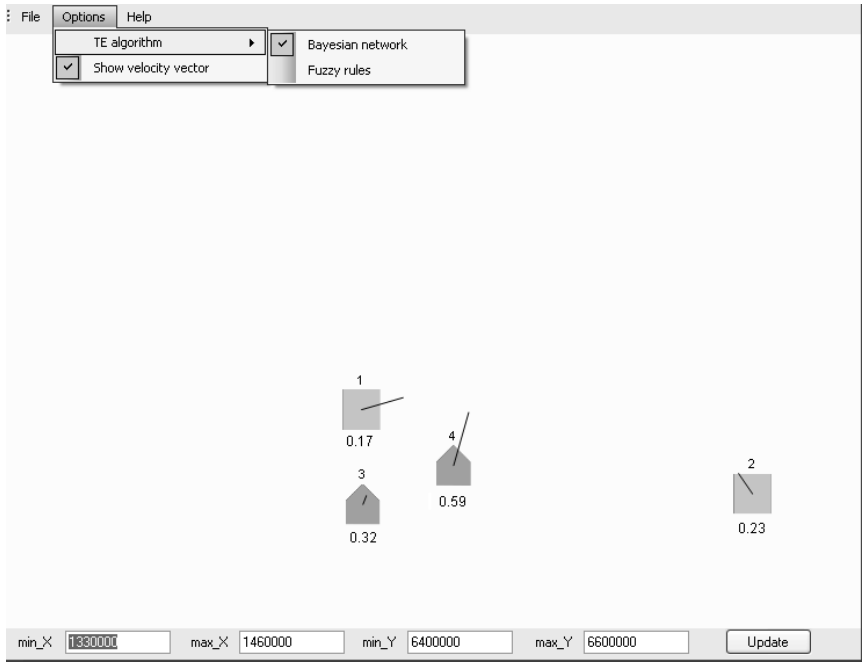


Figure 6.2: Illustration of STEWARD's graphical user interface.

such kind of air defense systems). Hence, the chosen weapon allocation algorithm aims at minimizing the expected target value of the targets in $\mathbf{T}_{wa}$ by allocating the set $\mathbf{W}_{wa}$ of available firing units to them. The use of firing units has been constrained so that a firing unit cannot make a new allocation until its last engagement is finished (i.e. until the engaging surface-to-air missile destroys or miss its intended target). This make perfect sense for missiles guided by the firing unit's radar, while it is less realistic for fire-and-forget missiles, i.e. missiles that do not require further guidance after launch. This constraint has been implemented into the testbed by the creation of a set $\mathbf{W}_{eng}$ of current engagements. If a firing unit is allocated to a target, the firing unit is added to this set, and information regarding the unique object ID of the weapon implementing the allocation is stored. The firing unit is not removed from this set until the testbed gets information from the simulation environment that the weapon with a matching object ID has been destroyed. Hence, all members of $\mathbf{W}_{eng}$ are excluded from the set $\mathbf{W}_{wa}$ of potential firing unit candidates for weapon allocation, i.e. $\mathbf{W}_{wa} = \mathbf{W} - \mathbf{W}_{eng}$.

Since the weapon allocation algorithms use information regarding kill probabilities when deciding on to which target a firing unit should be allocated, es-

timation of such kill probabilities are needed. In the current simulations, only the distance between a target T_i and a firing unit W_k is used to calculate the associated kill probability $P_{ik} \in [0,1]$. This is obviously a simplification since kill probabilities in real world depend on a lot of different factors such as distance, target type, weather conditions, target size, maneuverability of the target, etc., but for demonstrating the concept the simplified kill probability estimations used here should be adequate. Hence, it is here assumed that $\widehat{P}_{ik}$ is linearly decreasing with an increasing distance between T_i and W_k, except for very small and large distances, where $\widehat{P}_{ik}$ is assumed to be zero.

When the used weapon allocation algorithm has returned a feasible solution, this is communicated back to the simulation engine, in which weapons are fired against targets in accordance to the suggested allocation. The sensor observations are regularly sent to the TEWA module, so that target values are calculated, used as input to weapon allocation, and so on, until the scenario is over. Once the scenario is completed, the testbed calculates the survivability of the defended assets and the total cost for the used firing units. Since the outcomes of engagements are not deterministic, different runs of a scenario with a specific selection of threat evaluation and weapon allocation algorithms can result in different outcomes when it comes to survivability and usage of firing units. Hence, we have implemented the possibility to design experiments in which a scenario automatically is repeated a large number of times, in order to get more statistically significant results. For each simulation run, the survivability and the number of firing units used are logged.

6.2 SWARD

As identified in section 5.3, there is today a large problem when trying to compare and benchmark new weapon allocation algorithms to already existing ones. In order to overcome this problem, the open source software SWARD[2] (System for Weapon Allocation Research and Development) has been developed. The name is intended to contrast it to the testbed STEWARD, and to show that it focuses solely on the weapon allocation part of TEWA.

SWARD has been implemented in Java, in order to ensure platform independence and to make it available for as many researchers as possible. 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++. However, since SWARD is a testbed rather than a weapon allocation module intended for being plugged

[2]The current version of the open source testbed SWARD can be downloaded freely from http://sourceforge.net/projects/sward/

into a real-world TEWA system, it is the author's hope and belief that the possibly unoptimized execution times are outweighed by the platform independence made possible by the choice of implementing the code in Java.

The choice to release SWARD as an open source project needs to be commented upon, 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 (DoD Chief Information Officer Memorandum, 2009), 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. By releasing implementations of weapon allocation algorithms as open source code, the hope is that a larger community will be able to find disadvantages and advantages with existing algorithms and in the long run develop novel state of the art algorithms. This is also an answer to the wish for sharable testbeds within the information fusion community put forward in Llinas (2009).

A problem identified in section 5.3 was that unclassified real-world air defense scenarios on which weapon allocation algorithms can be tested is lacking. For this reason, researchers are most often testing developed algorithms for weapon allocation on randomly created problem instances. A key issue here is that details regarding which kind of probability distributions that are used for generating the problem instances, in which intervals the random numbers are drawn, etc. often are not published. Even in cases where most such details are given, it is not possible to recreate exactly the same problem instances as used in the experiments due to the random fashion in which problem instances are generated. This should not necessarily be a large problem if algorithms were tested on very large numbers of problem instances for each problem size, but since such testing takes time, algorithms are rarely tested on more than ten problem instances for each tested problem size. To make comparisons based on a small number of problem instances, where the problem instances vary between the different algorithms, can give very misleading results. For this reason, it has been made sure that SWARD guarantees that problem instances easily can be recreated on different occasions, demanding only a few parameter settings to be specified (these settings are described in section 6.2.1).

Yet another problem identified in section 5.3 was that weapon allocation algorithms rarely are described well enough to allow for implementation by other researchers. This might in some cases be due to a conscious choice not to give away details enough for others to implement the suggested algorithm, but in most cases, this probably is not the case. In SWARD, it is easy to implement and add new algorithms for weapon allocation to the existing ones. In the same manner as WEKA with its open source implementations facilitates the benchmarking and development of machine learning algorithms within the machine learning community, SWARD is intended to facilitate the benchmarking and development of new algorithms for static weapon allocation.

The SWARD testbed consists of four main packages: a `GUI` package, an `algorithm` package, a `scenario` package, and an `experiment` package. Additionally, there is functionality for calculating statistics, etc. in the `util` package.

In figure 6.3, it is illustrated how the most important classes of SWARD relate to each other. As can be seen, a very central class to SWARD is the class `Experiment`, which has relations to all the other important classes. Instantiations of this class control the generation of scenarios on which to test static weapon allocation algorithms, and solutions produced by the individual weapon allocation algorithms on the different scenarios tested are stored by the help of the `ExperimentResult` class. In order to set up experiments, the `SwardGUI` can be used. The individual packages and the classes being part of these are more thoroughly described in section 6.2.1–6.2.4.

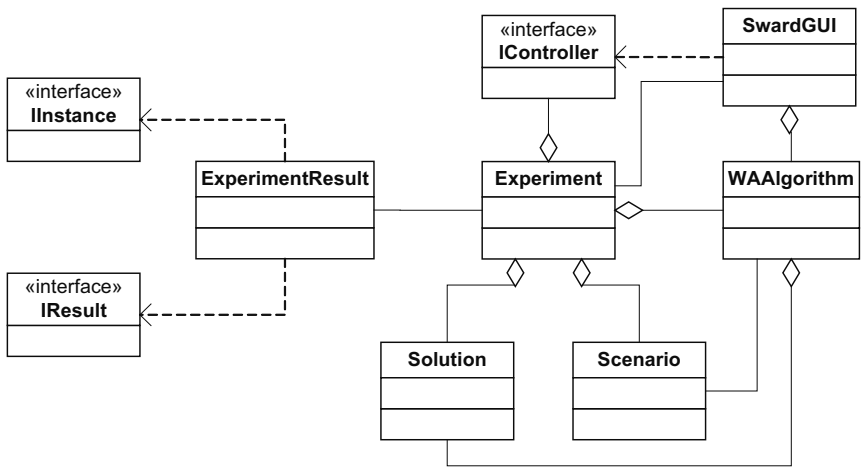


Figure 6.3: UML class diagram describing the relations between the main classes within SWARD.

6.2.1 The Scenario Package

A central part of SWARD is the generation of scenarios on which to test various algorithms for static weapon allocation. Note that these are not dynamically evolving air defense scenarios as in STEWARD, but rather static information representing the situation in a single point in time of a scenario. These snapshots of scenarios will in the following also be referred to as problem instances. In SWARD, both static target-based and static asset-based scenarios can be generated. A static target-based scenario consists of a matrix of kill probabilities and a vector of target values. Given as input a parameter $|\mathbf{T}|$, a parameter

$|\mathbf{W}|$, and a seed s, a static target-based problem instance is generated, where kill probabilities as default are randomly generated in the interval [0.6,0.9] and target values as default are randomly generated in the interval [25,100]. Hence, the default target values are not in the range [0,1] that we worked with in chapter 3, but can be seen as scaled target values between 0.25 and 1, where the threshold has been set so that targets with lower target values are not subject to weapon allocation. Obviously, these intervals can be changed, but these are used as default settings, in order for the created problem instances to be in accordance with the experiments presented in Ahuja et al. (2007). By making use of the seed s, it is guaranteed that problem instances easily can be recreated. This is ensured since the seed is used to control Java's pseudo-random number generator, so that experiments created with the same seed and parameter settings will generate identical sequences of problem instances independently of on when and on which machine SWARD is used.

In a similar way as the target-based scenarios, static asset-based scenarios are generated from the input parameters $|\mathbf{T}|$, $|\mathbf{W}|$, $|\mathbf{A}|$, 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, a vector of protection values, and a vector of target aims, as explained in section 3.1.2. The default settings are that lethality probabilities are generated randomly in the interval $[0.3, 0.8]$ and protection values in $[25, 100]$, where the latter have been chosen to be on the same scale as the target values used for target-based problem instances. 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 (if such targets exist) 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.

This distribution of target aims ensures that each defended asset is threatened by at least one target when $|\mathbf{T}| \geq |\mathbf{A}|$. An alternative way to generate the target aims can be to do this entirely random, in order to allow for situations where a majority of all targets are aimed for the same defended asset.

6.2.2 The Algorithm Package

Within the algorithm package, there is an abstract class `WAAlgorithm` with a method for calculating the objective function value for a given allocation of firing units to targets. The objective function value is calculated in accordance with equation 3.4 for target-based scenarios, and in accordance with equation 3.7 for asset-based scenarios. The class is declared as abstract since no object instances are allowed to be created of the `WAAlgorithm` class. Instead,

the class should 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. Most of the algorithms are suitable for both target-based and asset-based weapon allocation, as described in chapter 4. Other researchers are encouraged to implement their own algorithms for weapon allocation into the testbed.

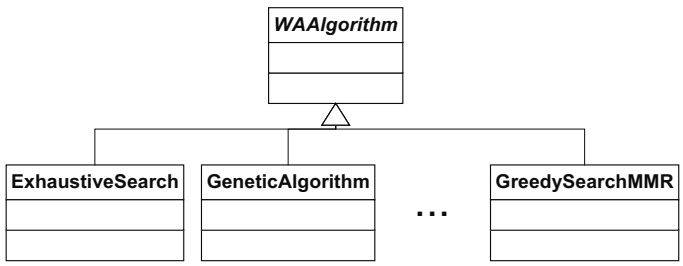


Figure 6.4: UML class diagram illustrating that specific weapon allocation algorithms such as a genetic algorithm are implemented as classes that inherits attributes and methods from the abstract class `WAAlgorithm`.

In order to demonstrate how it is possible to extend SWARD with new algorithms for static weapon allocation, it is now shown how `WAAlgorithm` has been implemented, and how new algorithms are supposed to inherit from and override necessary methods within `WAAlgorithm`.

```
public abstract class WAAlgorithm {
...
  // The solution to be returned by the algorithm
  Solution bestSolution;

    /* The empty base constructor for the algorithm */
    public WAAlgorithm() {}

    /* The method to override*/
    public abstract Solution calculateSolution(long
     timeInMilliSeconds);

    /* Calculates objective function value for a solution */
    public double calculateFValue(
     Vector<Integer> allocation) {...}
}
```

What is shown of `WAAlgorithm` are the fundamental parts of the class that have to be understood in order to implement a weapon allocation algorithm

into SWARD. As shown in the comments, it is essentially just to override the `calculateSolution()`-method in order to be able to implement a new algorithm. How to override this method is shown in the parts of the code that follows:

```
package sward.algorithm;
...
public class NewAlgorithm extends WAAlgorithm {
...
    public Solution calculateSolution(long timeInMS) {
        long start = System.currentTimeMillis();
        long stop = start + timeInMS;
        ...

        while (System.currentTimeMillis() < stop) {
            // Generate bestSolution here...
        }
        return bestSolution;
    }
}
```

It can be seen that we in the override of `calculateSolution()` simply keep track of the time allowed for search, and when there is no time left return the best solution found so far. This is essentially what all implemented algorithms should do, and what differ among algorithms are the techniques used for searching for good solutions.

6.2.3 The Experiment Package

An experiment is in SWARD 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_{step}). The same kind of parameters should be set for $|\mathbf{W}|$. These parameters are referred to as W_{start}, W_{end}, and W_{step}. 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 $timeLimit$, 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. Using the selected experiment settings, algorithms that have been selected to participate in the experiment are tested on the generated problem instances.

The class `ExperimentResult` is used to handle the results from an experiment. 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 their associated standard deviations. A number of experiments which have been constructed in SWARD are shown and explained in chapter 7.

6.2.4 The GUI Package

Experiments can be set up using code, but in most cases it is more convenient to use the graphical user interface that has been developed for SWARD. Figure 6.5 illustrates what the graphical user interface looks like. The values that can be edited within the text boxes correspond to the experiment parameters described in section 6.2.3, so that all relevant experiment settings can be adjusted within the graphical user interface.

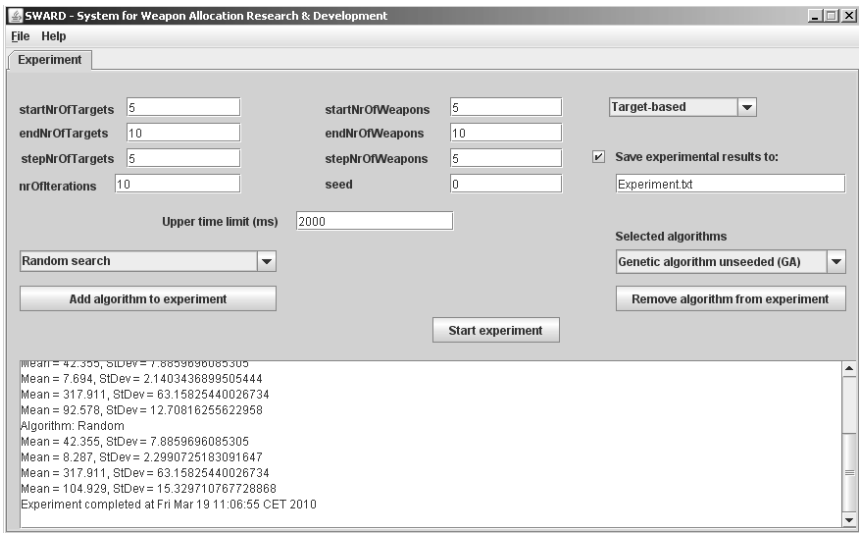


Figure 6.5: Screenshot of the SWARD graphical user interface.

In the experiment shown in figure 6.5, the algorithms have been constrained to search for two seconds on each problem instance. The algorithms are on each problem size tested on ten problem instances, as specified by the value of $nrOfIterations$. The other selected experiment settings show that a total of four problem sizes are tested: ($|\mathbf{T}| = 5$,$|\mathbf{W}| = 5$), ($|\mathbf{T}| = 5$,$|\mathbf{W}| = 10$), ($|\mathbf{T}| = 10$,$|\mathbf{W}| = 5$), and ($|\mathbf{T}| = 10$,$|\mathbf{W}| = 10$). In the bottom of figure 6.5, the average objective function values and the standard deviations obtained by the random search algorithm are shown for the four tested problem sizes. It can also be seen that the generated problem instances are target-based, as shown by the selection in the drop-down list to the right. The experiment settings chosen

within the graphical user interface can be accessed in the `Experiment` class by the use of the interface `IController`, which also is part of the `GUI` package.

6.3 Discussion

As has been mentioned, the current use of STEWARD is to demonstrate the idea of using computer-based simulations for evaluating the effectiveness of various TEWA systems. Hence, the current version of STEWARD is a prototype testbed, and more work needs to be done before it becomes really useful for benchmarking purposes. At the moment, "ground truth" position data are used as input to the TEWA module, i.e. no sensor models are used to simulate the imperfect nature of the target tracks produced in a real-world TEWA system. Such sensor models can quite easily be added to the user modules used in STAGE Scenario. The status of SWARD is more mature, and can already today be used for systematic performance evaluation of static weapon allocation algorithms, as shown in chapter 7. The latest stable release of SWARD (version 1.2) has already been downloaded from various countries (statistics available from `http://sourceforge.net/downloads/sward/stats_map`).

An advantage with SWARD compared to STEWARD is its open source nature, making it possible for anyone to freely use it. STEWARD is, as has been described in this chapter, relying on the commercial simulation engine STAGE Scenario. It is nothing in the used communication protocol or the TEWA module as such that constrains them to only being used with the current simulator, hence, it is possible that it in the future can be replaced with an open source simulation engine. However, there is currently a lot of functionality built into STAGE Scenario that cannot easily be implemented into any open source simulation engine that the author is aware of.

An interesting possible future use of the closed-loop simulations provided by STEWARD is to not only use it for measuring survivability and resource usage for specific scenarios, but also for automated reinforcement learning in which threat evaluation algorithms can be optimized so as to give as good results as possible on a set of air defense scenarios. As an example, this could be used for finding good conditional probability distributions for the Bayesian network algorithm, given a fixed structure, by letting the obtained survivability and resource usage cost values be used as the fitness of a specific parameter setting. The structure could still be identified by human experts, in order to allow for a well-understood Bayesian network that can be trusted by human decision makers, but the conditional probability distributions that are harder to elicit could instead be learned by running the scenarios multiple times and adjust the values of the conditional probability tables in the direction suggested by the obtained fitness scores. Although an interesting opportunity, it must first be made sure that the used air defense scenarios really are appropriate, since the learned threat evaluation algorithms otherwise can show unwanted behavior. The usual problems with overfitting must be looked after, and it must be carefully checked

that learned models look reasonable to human experts, since relying on automated algorithms that have been created erroneously would have much larger negative effects than not using them at all. In the author's view, very careful consideration must be given before a threat evaluation algorithm learned from either simulated or real data is used in a real-world system, but it is undoubtedly an interesting area of future research.

In SWARD, we are 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 are generated in SWARD (and in previous reported experiments with static weapon allocation algorithms) may not necessarily create the same kinds of search spaces that would be experienced in real-world air defense situations. For some of the algorithms implemented into SWARD, such as the maximum marginal return algorithms and the random search algorithm, this would not have any impact, but it is not unlikely that this could have at least a slight impact on the performance of algorithms such as particle swarm optimization algorithm, since the objective function values of nearby solutions then would be correlated, given that the firing units and targets were sorted after similarity in e.g. position or type. An idea which has not been made use of in the work presented here, but could be of interest for the future, is therefore to create problem instances with some inbound structure, e.g. by using STAGE Scenario or some other simulator to create scenarios, and to take snapshots from the scenarios as the problem instances used when evaluating the weapon allocation algorithms in SWARD. This is not very different from how we now evaluate TEWA systems using STEWARD, but the fundamental difference is that we then would not let the proposed allocations have any effect on the simulations, but only evaluate the weapon allocation algorithm on how good solutions they produce, as measured by the obtained objective function values.

6.4 Summary

In this chapter, the testbeds STEWARD and SWARD have been presented. STEWARD is intended for demonstration of how the performance of TEWA systems can be compared using computer-based simulations by measuring the

survivability of defended assets, and resource usage of firing units. The simulation engine used for controlling the simulations is the commercial software STAGE. The simulation engine communicates with a TEWA module using a socket connection, so that sensor observations are used as input to the selected threat evaluation algorithm. Targets that are assigned target values over a specified threshold value become subject to weapon allocation. Once the selected weapon allocation algorithm has allocated firing units to threatening targets, these allocations are communicated back to STAGE, in which the suggested allocations are implemented into actual engagements.

In the open source testbed SWARD, the performance of various algorithms for static weapon allocation can be compared. Moreover, new algorithms can easily be added to the testbed. By making experiments related to performance evaluation in SWARD, it is ensured that the experiments easily can be set up, and that the used problem instances can be recreated at later occasions on other systems, in order to allow for fair benchmarks of various algorithms for static weapon allocation.

Chapter 7
Experiments

In previous chapters, a number of algorithms for real-time threat evaluation and weapon allocation have been suggested, and different ways for evaluating the suggested algorithms have been presented. Moreover, detailed presentations of developed testbeds for facilitating the comparison of performance among different algorithms have been given. In this chapter, initial experiments in which threat evaluation algorithms are compared using STEWARD are presented in section 7.1. Thereafter, more rigorous experiments with the SWARD testbed in which the performance of the developed weapon allocation algorithms have been evaluated are presented in section 7.2. The purpose of these experiments is twofold in that they show which of the implemented algorithms that have the best real-time performance on various problem instances, and also serves as a demonstration of the suggested evaluation methodology and the developed testbeds.

7.1 Comparison of Threat Evaluation Algorithms

7.1.1 A First Scenario

As discussed earlier, scenarios to use for evaluation of TEWA systems with STEWARD can be generated in the STAGE Scenario tool. These scenarios are during runtime continuously communicated to the TEWA module, so that threat values can be calculated in real-time for target-defended asset pairs using the selected threat evaluation method. In order to demonstrate this approach, a test scenario consisting of a single defended asset and four air targets (one F-16, one B-2 bomber, and two Boeing 747) has been constructed. Since there is only a single defended asset in this scenario, the calculated threat values also correspond directly to target values for the individual targets. Speeds of the targets are close to constant, except for the F-16, which accelerates at the point of its turn against the defended asset. Figure 7.1 illustrates the scenario by showing the way points used for determining the trajectories of the targets and the ap-

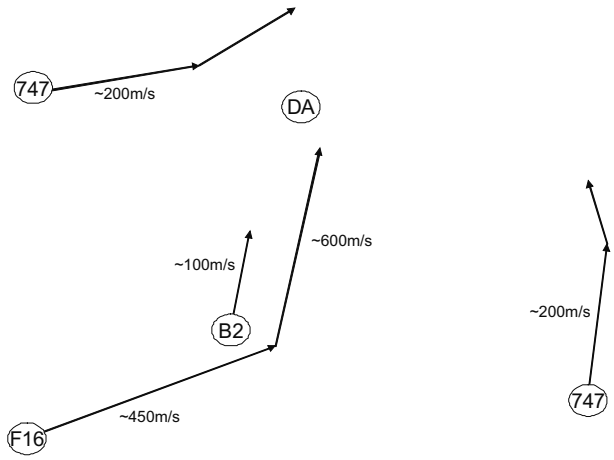


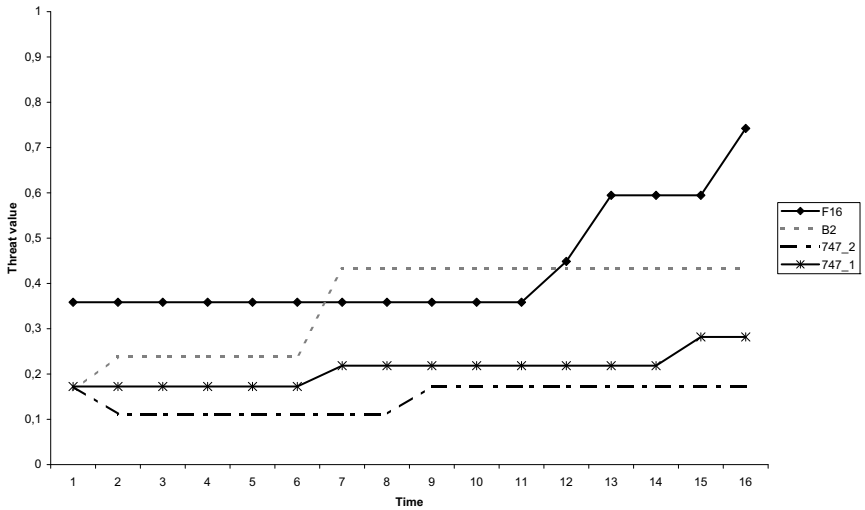
Figure 7.1: Illustration of the test scenario.

proximate speeds of the targets. The reason for why this particular scenario has been chosen is that it shows extremes with different kinds of air targets (fighters, bombers, and civilian aircrafts) with varying velocities. In figure 7.2(a), the threat values inferred by the Bayesian network presented in section 4.1.1 are shown, while the corresponding values inferred by the fuzzy logic algorithm presented in section 4.1.2 are shown in figure 7.2(b). In the figures, one unit of time corresponds to 50 time updates, which is approximately ten seconds.

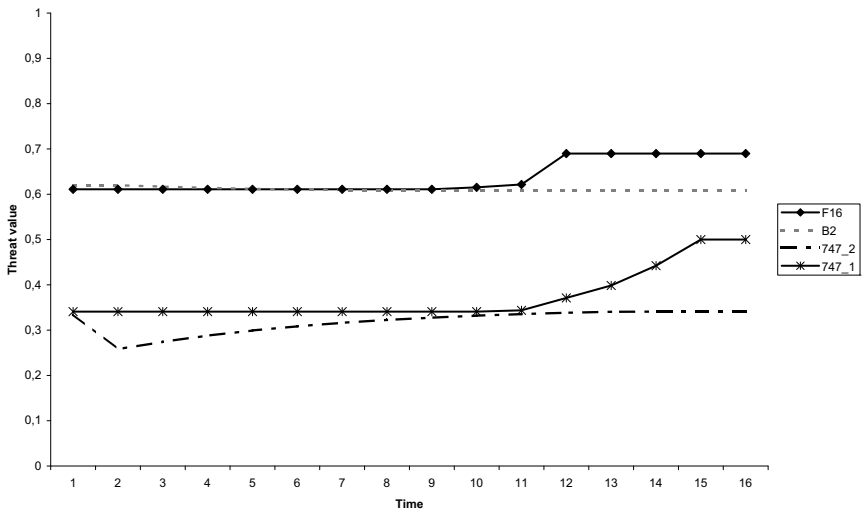
Analysis of Obtained Results

As can be seen, the resulting ranking of the threats are quite similar, even though the threat values in general are higher in figure 7.2(b). Another difference is that the threat values are more clearly separated in figure 7.2(a). The output from the Bayesian network better reflects the author's opinion regarding the threat values in the scenario, but it should also be noted that the time spent to create the fuzzy inference rules was shorter than for the development of the Bayesian network, since the used membership functions were chosen to be similar to the ones presented in Liang (2007).

Looking at the sensitivity of the algorithms, we can in figure 7.2 see that the changes in threat values are smoother when using the fuzzy inference sys-



(a) Bayesian network (see section 4.1.1)



(b) Fuzzy inference rules (see section 4.1.2)

Figure 7.2: Calculated threat value for different targets as a function of time.

tem than when the Bayesian network is used. Of course, the output from the Bayesian network becomes smoother the more states that are added. However, this comes with the cost of an increased burden in specifying more conditional probabilities. Also, the computational complexity increases with an increased number of states. This smoothness is an interesting property of fuzzy logic, compared to the Bayesian network's less soft boundaries between states.

Another difference between the two approaches to threat evaluation is that the inference goes in only one direction when using fuzzy inference rules (i.e. evidence regarding the input variables is entered, whereupon a crisp value for the output variable is calculated), while the Bayesian network can be used to compute an arbitrary posterior probability of interest, given some evidence. As an example, the Bayesian network can be used to calculate $P(Speed = low|Distance = far)$ as well as $P(Threat = true|Distance = close, TBH = short, Speed = low)$. An even more important property of Bayesian networks is their ability to handle missing information. A fuzzy inference system is often dependent upon that the values of all its input variables are known, while this is not the case of the Bayesian network. In a hostile situation this can be very important, since complete sensor coverage seldom is the case, sensors can be disturbed by countermeasures, etc.

In the implementation of the fuzzy logic algorithm, min- and max-operators have been used as t-norm and t-conorm. However, many other operators have been proposed within literature, and there is no common agreement on which that is the best. In fact, different t-norms and t-conorms seems to be appropriate for different types of problems (Kreinovich and Nguyen, 2006). The choice of t-norm and t-conorm will often influence the result of the fuzzy inference, hence, the choice of the "right" t-norm and t-conorm can be crucial, and can be seen as a problem with the fuzzy logic approach. Bayesian networks does not have this problem of ad hoc solutions, since they have a sound mathematical foundation within probability theory. Nevertheless, fuzzy inference rules are very appealing in that they are easy to work with. On the opposite, to obtain the probabilities that are required for the conditional probability tables in a Bayesian network can often be a daunting task. This problem can be solved by learning the probabilities from data (as described in section 2.3.1), however, this demands large amounts of data that seldom is available in this domain.

7.1.2 A Second Scenario

In order to demonstrate how the STEWARD testbed and the suggested performance evaluation metrics work, we have in STAGE also created a second air defense scenario consisting of two firing units (which also are the defended assets within the scenario), three hostile fighters, and a neutral civilian aircraft. Both firing units have been assigned a protection value of 1.0. The scenario is of quite high intensity and is approximately 8 minutes long. The blue force firing units are equipped with three surface-to-air missiles each. The hostile fighters

have one missile each, which they fire according to preassigned missions. Each simulation has been run hundred times for each TEWA system configuration, in order to get more reliable results. Hence, a TEWA system's effectiveness has been calculated as:

$$E = \frac{\sum_{t=1}^{100} \alpha S_t - \beta C_t}{100} \tag{7.1}$$

In the experiments, a constant kill probability of 0.7 within the range 0-500 kilometers has been used. If the range is larger, the kill probability becomes 0. For convenience, this holds true for both red force and blue force weapons in the simulation.

In the first experimental setup, we have used two identical TEWA system configurations, except for the used threat evaluation algorithms. The threat evaluation algorithm used in the first TEWA configuration is the implemented Bayesian network, while the second configuration uses the threat evaluation algorithm based on fuzzy logic. For both configurations we have used the exhaustive search algorithm for weapon allocation, since the number of firing units and targets has been small enough to allow for calculation of the optimal solution in real-time. The weapon allocation algorithm has been run with regular intervals, and has taken targets with target values higher than 0.5 into consideration. The results obtained for the two configurations are shown in the two first rows of table 7.1 (a uniform cost of 1 has been used for each usage of a firing unit).

In order to investigate what effect small changes of the used configuration can have on the overall performance of the TEWA system, the survivability and firing unit cost has also been measured when the threshold τ for how high the target value needs to be for a target to become subject to weapon allocation has been changed from 0.5 to 0.7. The results from these runs can be seen in the two last rows of table 7.1.

Table 7.1: Results for the tested TEWA configurations

Configuration	μ_S	μ_C	$E(\alpha = 0.99, \beta = 0.01)$
BN ($\tau = 0.5$)	0.255	4.83	0.204
FL ($\tau = 0.5$)	0.170	6.00	0.108
BN ($\tau = 0.7$)	0.480	3.76	0.438
FL ($\tau = 0.7$)	0.465	3.58	0.425

Analysis of Obtained Results

Looking at the survivability, we can see that the average is 0.255 for configuration 1, and 0.170 for configuration 2. Hence, if we only consider survivability, the first configuration is the best of the two for this particular scenario. If we

instead look at the weapon usage cost, the average for configuration 1 is 4.83, while it for configuration 2 becomes 6.00 (i.e. it uses all defensive resources in each run of the scenario). From this we can see that the first configuration also is more selective with its resources than the second configuration. Assuming use of the user-defined parameters $\alpha = 0.99$ and $\beta = 0.01$ (giving a much larger weight to survivability of defended assets than the saving of resources), we can calculate the effectiveness of each TEWA configuration for the specific scenario. Doing this, we end up with an effectiveness of 0.204 for configuration 1, while the corresponding number for configuration 2 becomes 0.108. The actual effectiveness number does not say anything on its own, but if we compare the numbers to each other, it can be concluded that the first TEWA configuration performs better than the second on the described scenario, given the choice of $\alpha = 0.99$ and $\beta = 0.01$.

As can be seen in the two last rows of table 7.1, the average survivability of the third configuration (i.e. the same configuration as configuration 1, except for the used threshold) has increased to 0.480, while it for the fourth configuration (the same as configuration 2 except for the used threshold) has increased to 0.465. The number of used weapons is noticeable lower in this second experiment, compared to the first one. This is expected, since we demand higher target values before targets are considered as potential candidates for weapon allocation. Using the same user-defined parameters as in experiment 1, i.e., $\alpha = 0.99$, $\beta = 0.01$, the resulting effectiveness becomes 0.438 for configuration 3 and 0.425 for configuration 4.

From the above results, it is obvious that rather small changes of a TEWA system configuration (in this case the threshold settings) can have major impact upon the resulting weapon allocation, which in its turn affects survivability and resource usage, and thereby also the effectiveness. The results should not be interpreted as any of the tested threat evaluation algorithms being superior to the other, since the TEWA configurations only have been tested on a single air defense scenario, on a specific choice of α and β. Hence, the provided experiment should only be taken as an example of how various TEWA configurations can be compared to each other, using the suggested performance evaluation methodology and the functionality provided with STEWARD.

7.2 Comparison of Weapon Allocation Algorithms

The algorithms for static weapon allocation implemented into the open source testbed SWARD have been used for experiments on problem sizes of various size, and these experiments and their results are presented here. In section 7.2.1, it is investigated how much computation time that is needed to solve small-scale problem instances optimally using exhaustive search, while the real-time performance of the implemented heuristic algorithms is tested on small-scale problem instances in section 7.2.2, and larger-scale problem instances in section

7.2.3. Finally, the algorithms' sensitivity to the choice of time available for search is investigated in section 7.2.4.

The experiments presented here have all been run on a computer with the following specifications:

- CPU: Intel Core 2 Quad Q9450, 2.66GHz (12MB L2 cache)
- Memory: 4GB RAM
- Operating system: Microsoft XP (Service Pack 3)

Moreover, the default settings in SWARD have been used for the generation of all problem instances. Hence, the static target-based problem instances have been generated randomly from uniform probability distributions so that target values $V_i \in U[25{,}100]$ and kill probabilities $P_{ik} \in U[0.6{,}0.9]$. Similarly, the static asset-based problem instances have been generated so that lethality probabilities have been generated randomly as $\pi_i \in U[0.3, 0.8]$ and protection values as $\omega_j \in U[25, 100]$. For those instances, the target aims have also been generated as described in section 6.2.1.

In the experiments with heuristic algorithms, we have on the target-based problem instances tested a total of eight different algorithms: the genetic algorithm (GA), the genetic algorithm seeded with an individual obtained through the use of the enhanced maximum marginal return algorithm (GA-S), the ant colony optimization algorithm (ACO), the particle swarm optimization algorithm (PSO), the maximum marginal return algorithm (MMR), the enhanced maximum marginal return algorithm (EMMR), the algorithm which improves the solution returned by the maximum marginal return algorithm using local search (MMR-LS), and the random search algorithm (RS). The same algorithms have been used for the asset-based problem instances, except for the ant colony optimization algorithm, since it is designed only for target-based problems. The MMR and EMMR algorithms have in the asset-based case worked on target-based approximations of the asset-based problem instances, as explained in section 4.2.3. For the approximation of target values, we have calculated the target value V_i for a target T_i as:

$$V_i = \omega_j \times \pi_i, \tag{7.2}$$

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 Hosein (1990) 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.

For the implemented weapon allocation algorithms, the following parameter settings have been used throughout all the experiments:

- Ant colony optimization algorithm
  - $nrOfAnts = \max(|\mathbf{T}|,|\mathbf{W}|)$, $q_0 = 0.5$, $\alpha = 1$, $\beta = 1$, $\varphi = 0.1$, $\rho = 0.1$.
- Genetic algorithm
  - $nrOfIndividuals = \max(|\mathbf{T}|,|\mathbf{W}|)$
- Particle swarm optimization algorithm
  - $nrOfParticles = 50$, $c_1 = 2.0$, $c_2 = 2.0$, $\omega = 0.8$.

The settings for the ant colony optimization algorithm make sure that there is a balance between the (greedy) exploitation and biased exploration of the search space, and that pheromone and heuristic information are assigned equal weight. Likewise, the settings for the particle swarm optimization algorithm dictate that the social and cognitive components of the swarm should be of equal importance when updating the velocity vectors of the particles. These suitable parameter settings for the various algorithms have been established based on experimentation in which the obtained objective function values have been used as selection criterion.

7.2.1 The Limit for Exhaustive Search

In a first experiment with the developed weapon allocation algorithms, the limit for how large problem sizes that can be solved reasonably fast using exhaustive search has been investigated. By varying $|\mathbf{T}|$ and $|\mathbf{W}|$ between 1 and 9, a total of 81 problem sizes have been tested for static target-based weapon allocation, and as many have been tested for static asset-based weapon allocation. In the latter case, the number of defended assets was fixed to five. For each tested problem size, ten problem instances were randomly created. The total time required for the ten runs of the specific problem size was measured, and this time was used to calculate an average computation time needed for each problem size. To average over ten problem instances should not really be necessary since all problem instances of a specific problem size demand equally many computations when applying the exhaustive search algorithm. Nevertheless, this approach was used to ensure that the granularity of which the operating system measure time would not affect the results, and to reduce the effects of e.g. garbage collection.

In order to recreate the experiment in SWARD, the following parameter settings should be used (note that the DAs parameter only is used for the asset-based problem instances):

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

The obtained results for the average computation times are presented in table 7.2 for the static target-based case, and in table 7.3 for the static asset-based case. For the problem sizes where no values are shown, the average computation time was less than or equal to 1 millisecond.

Table 7.2: Average computation time (in ms) for the exhaustive search algorithm on static target-based problems.

	|W|					
|T|	...	4	5	6	7	8	9
...							
3					4	18	56
4				9	43	197	558
5			3	23	131	711	3845
6			11	73	490	3201	20924
7		3	25	195	1502	11480	87422
8		6	50	448	3955	34708	302758
9		9	95	953	9464	93365	917651

Table 7.3: Average computation time (in ms) for the exhaustive search algorithm on static asset-based problems.

	|W|					
|T|	...	4	5	6	7	8	9
...							
3					7	23	73
4			3	14	62	267	773
5			6	31	168	901	4793
6			15	101	686	4221	27097
7		4	37	271	2098	15231	115238
8		7	73	643	5511	47218	401687
9		14	142	1368	13397	128034	1224496

Analysis of Obtained Results

As can be seen in table 7.2, all the tested target-based problem instances involving six firing units or less took less than one second to solve optimally using exhaustive search. Hence, for general static target-based problem instances where the number of threatening targets are nine or less, we can find the optimal solution in less than one second (using a computer equivalent in performance to what has been used here), as far as the number of available firing units is six

or less. Looking only at the problem sizes where $|\mathbf{T}| = 9$, i.e. the last row of table 7.2, we can see that computation time needed for solving a problem instance optimally is approximately increased by a factor of ten when an extra firing unit is added. As also can be seen, increasing the number of firing units has a much larger effect on the computation time needed than if the number of targets is increased. This is quite natural, given that there are $|\mathbf{T}|^{|\mathbf{W}|}$ feasible solutions to test, i.e. an increase of $|\mathbf{W}|$ in general gives raise to much more extra feasible solutions to check than a corresponding increase of $|\mathbf{T}|$.

In table 7.3, it can be seen that the results from the experiment with the asset-based problem instances follow the same pattern as the target-based problem instances. Generally, the computation times are higher in the asset-based cases compared to their target-based equivalents, which is expected since it is somewhat more complicated to calculate a solution's objective function value in an asset-based setting than in its target-based counterpart (see algorithm 4.2 and algorithm 4.1, respectively).

The presented results give a clear indication of how long it takes to solve problem instances of different size using exhaustive search. However, to give an answer to how large problem instances that can be solved in real-time demands more information, namely, exactly what is meant by real-time in this context. Unfortunately, there is no clear and definite answer to this, since this can vary with the type of targets, type of firing units, and what kind of air defense situation it is. However, looking at the results in table 7.2 and 7.3, we can clearly see that the largest problems tested take too long time to be solved in real-time air defense situations, since they take more than 15 minutes to solve. In this amount of time, an airplane flying with Mach 1 is able to travel further than 250 kilometers, which is a longer distance than the range of most air surveillance radars (see table 2.1). Similarly, the time it takes from the firing of nine short-range missiles until they hit their intended targets is much shorter than the more than twenty minutes it takes to solve a $(|\mathbf{T}| = 9, |\mathbf{W}| = 9)$-problem instance optimally for the asset-based case. Clearly, the time horizon for deciding which firing units to allocate to which targets is a matter of seconds at most (Khosla, 2001), not minutes. This has also been confirmed in the author's discussions with air defense experts. The upper time limit for how long we can let an algorithm search for a good solution is rather approximately 1–3 seconds, setting the limit for how large problem sizes that can be solved in real-time using exhaustive search on the used computer to around $(|\mathbf{T}| = 7, |\mathbf{W}| = 7)$ for both target-based and asset-based weapon allocation.

7.2.2 Performance of Heuristic Algorithms on Small-Scale Problems

In a second experiment, the real-time performance of the implemented heuristic algorithms for weapon allocation has been measured on a set of small-scale problems. An advantage with such small-scale problems is that it becomes pos-

sible to compute the optimal solution using exact methods, which can be used for comparison. In order to recreate the experiment that is presented here, the following parameter 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.

As seen in the parameter settings, the search time has been constrained so that a weapon allocation algorithm only is allowed to run for one second on each problem instance. The shown parameter settings are what have been used when evaluating the performance of the static asset-based weapon allocation algorithms. The same settings have been used when the static target-based weapon allocation algorithms have been tested, with the difference that the number of defended assets has not been specified or used in these experiments. Note that all the created problem instances are symmetric, i.e. there are as many firing units as there are targets. The impact of asymmetry in the number of targets and the number of firing units on the algorithms' solution quality is further investigated in section 7.2.3.

Table 7.4 shows the algorithms' average percentage deviations from the optimal solution for the tested target-based problem sizes, where the optimal solutions have been calculated using exhaustive search. The corresponding results for asset-based problem sizes are shown in table 7.5. 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:

$$\Delta_{alg} = \frac{|F_{alg} - F_{opt}|}{F_{opt}} \times 100, \tag{7.3}$$

where F_{alg} is the objective function value for the tested algorithm and F_{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.

Analysis of Obtained Results

A discovery that can be made when studying the results in table 7.4 and 7.5 is that the average percentage deviations from the optimal solution are higher in the target-based case than in the asset-based case. This should not necessarily be interpreted as that the algorithms are performing worse on target-based problem instances than asset-based problem instances (rather, the opposite is more likely true, since it as shown in section 7.2.1 takes longer time to evaluate a solution in the asset-based case, while the feasible solutions are equally many for an asset-based and a target-based problem instance of equal size).

Table 7.4: Deviation from optimal solution (in %) on target-based problem instances. Averaged over ten problem instances.

	5×5	6×6	7×7	8×8	9×9
GA	**0**	**0**	**0**	0.8	5.5
GA-S	**0**	**0**	**0**	2.4	5.8
ACO	**0**	**0**	**0**	0.9	1.7
PSO	**0**	**0**	**0**	**0**	**0.9**
MMR	39.8	27.2	44.7	50.3	44.5
EMMR	3.4	3.6	7.4	7.4	10.0
MMR-LS	**0**	**0**	5.4	7.7	5.6
RS	**0**	**0**	0.8	7.7	19.1

Table 7.5: Deviation from optimal solution (in %) on asset-based problem instances. 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**	0.1	0.5
PSO	**0**	**0**	**0**	**0**	**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	0.3	1.0	0.6	1.0	1.5
RS	**0**	**0**	0.4	1.2	2.7

Instead, this can be explained with that the objective function values obtained by the algorithms are much lower in the target-based cases than the asset-based cases, since these are minimization and maximization problems, respectively. This means that the obtained percentage deviation becomes higher in the minimization case, since the objective function value of the optimal solution generally is higher (see equation 7.3). Hence, the results shown in table 7.4 and 7.5 should not be compared to each other, since they stems from different "scales".

With this said, the results shown in table 7.4 can be studied. As can be seen, most of the algorithms are able to find the optimal solution to all problem instances of size ($|\mathbf{T}| = 5, |\mathbf{W}| = 5$) and ($|\mathbf{T}| = 6, |\mathbf{W}| = 6$). This is expected since it is possible to exhaustively search the whole search space in less than 100 milliseconds for those sizes, as was shown in table 7.2. Starting with the quite naïve random search strategy, it can be seen that the deviation from the optimal solution grows larger as the problem size increases, which is not very surprising. The well-known greedy maximum marginal return algorithm shows perhaps more surprisingly bad performance results on even the smallest tested problem sizes. This performance is improved upon by the enhanced version

of the maximum marginal return algorithm, as well as the algorithm in which the standard greedy algorithm is combined with local search, but as can be seen, all these greedy heuristics are outperformed by all the nature-inspired metaheuristics on all but the $(|\mathbf{T}| = 9, |\mathbf{W}| = 9)$-problem size. Of the nature-inspired metaheuristics, the ant colony optimization algorithm and the particle swarm optimization algorithm perform better than the others. Especially, the particle swarm optimization algorithm shows very good performance, given that it in one second is able to generate almost optimal solutions to problems consisting of $9^9 = 387{,}420{,}489$ feasible solutions.

Looking at the results from the tested small-scale asset-based problem instances shown in table 7.5, we can see that the trend is similar to the target-based results, in that all algorithms except for the greedy heuristics are able to find the optimal solutions to the small problem instances of size $(|\mathbf{T}| = 5, |\mathbf{W}| = 5)$ and $(|\mathbf{T}| = 6, |\mathbf{W}| = 6)$. The nature-inspired algorithms are performing best also in this case, and the particle swarm optimization algorithm is producing optimal solutions to all problem sizes except $(|\mathbf{T}| = 9, |\mathbf{W}| = 9)$, for which it produces near-optimal 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 particle swarm optimization algorithms and genetic algorithms 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 one wrong decision by a constructive, one-pass heuristic may result in a solution differing dramatically from the optimum of a small case (Rardin and Uzsoy, 2001). 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.

7.2.3 Performance of Heuristic Algorithms on Large-Scale Problems

In experiments with larger-scale problems, problem instances of sizes between $(|\mathbf{T}| = 10, |\mathbf{W}| = 10)$ and $(|\mathbf{T}| = 30, |\mathbf{W}| = 30)$ have been generated. For each problem size, hundred problem instances have been created in order to allow for more robust evaluation.

In order to recreate an experiment with the same problem instances in SWARD, the following parameters 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.

In table 7.6–7.8, we show the mean objective function values returned by the static target-based weapon allocation algorithms after one second, averaged over the hundred tested problem instances. 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. For getting a clearer picture of how the implemented algorithms perform in relation to each other, an average rank has also been calculated for the algorithms. This average rank information is presented in figure 7.3.

Table 7.6: Average objective function value for $|\mathbf{T}| = 10$. Averaged over 100 static target-based problem instances. (Lower objective function values are better).

	10×10	10×20	10×30
GA	93.4 (11.2)	16.2 (2.6)	3.0 (0.5)
GA-S	89.1 (10.6)	**12.0** (2.0)	**1.6** (0.3)
ACO	87.6 (10.3)	15.9 (2.2)	3.1 (0.5)
PSO	**86.4** (10.8)	15.4 (3.2)	4.1 (0.8)
MMR	127.0 (18.7)	25.7 (4.7)	5.2 (1.0)
EMMR	90.3 (11.4)	**12.0** (2.0)	**1.6** (0.3)
MMR-LS	90.0 (11.0)	18.5 (2.8)	4.2 (0.7)
RS	108.0 (13.5)	25.8 (3.9)	6.2 (0.9)

Table 7.7: Average objective function value for $|\mathbf{T}| = 20$. Averaged over 100 static target-based problem instances. (Lower objective function values are better).

	20×10	20×20	20×30
GA	576.6 (62.3)	226.6 (19.4)	99.8 (9.6)
GA-S	566.6 (61.8)	**162.7** (14.9)	**66.5** (7.3)
ACO	**564.7** (61.1)	199.9 (16.2)	93.2 (9.1)
PSO	570.8 (61.8)	283.0 (31.3)	148.3 (18.6)
MMR	588.2 (60.4)	230.1 (23.0)	104.1 (11.8)
EMMR	567.0 (61.9)	**162.7** (14.9)	**66.5** (7.3)
MMR-LS	565.7 (62.5)	210.8 (17.5)	102.4 (11.6)
RS	625.6 (62.4)	306.2 (28.6)	156.0 (16.0)

In a similar manner, the objective function values obtained when the algorithms have been applied to the generated asset-based problem instances are shown in table 7.9–7.11. The average ranks obtained for the algorithms when tested on the asset-based problem instances are shown in figure 7.4.

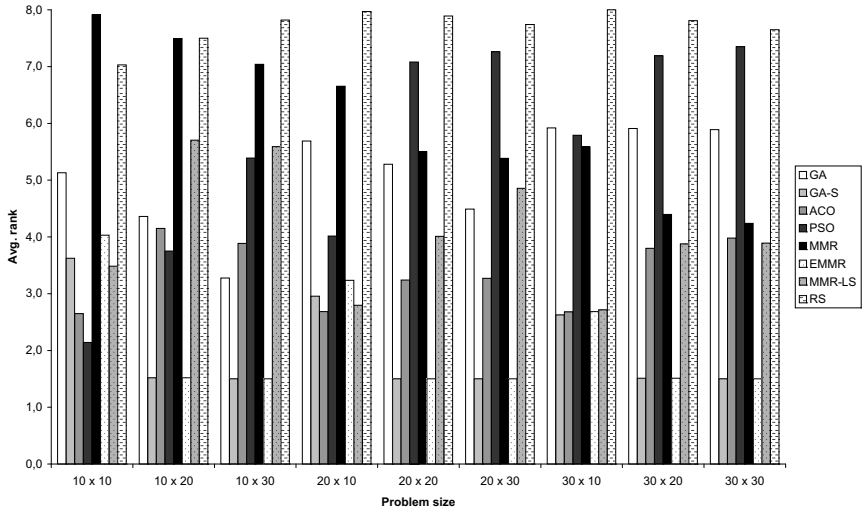


Figure 7.3: Average rank on target-based problem instances (where 1.0 is the best possible and 8.0 is worst possible rank).

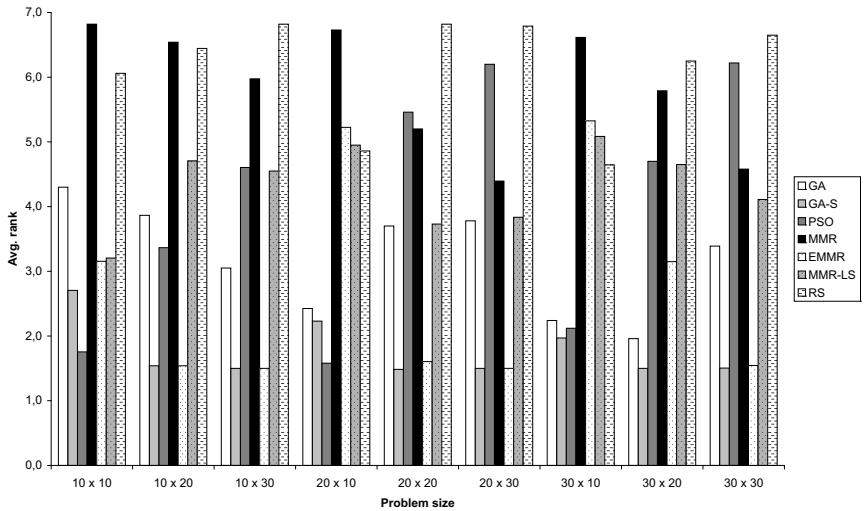


Figure 7.4: Average rank on asset-based problem instances (where 1.0 is the best possible and 7.0 is worst possible rank).

Table 7.8: Average objective function value for $|\mathbf{T}| = 30$. Averaged over 100 static target-based problem instances. (Lower objective function values are better).

	30×10	30×20	30×30
GA	1142.9 (81.8)	667.2 (58.0)	363.7 (25.6)
GA-S	1127.0 (80.0)	**584.5** (53.2)	**231.1** (14.6)
ACO	**1126.5** (80.1)	630.9 (52.4)	319.9 (18.0)
PSO	1147.4 (80.4)	785.1 (68.3)	527.1 (46.7)
MMR	1142.2 (78.0)	635.2 (51.0)	324.9 (23.6)
EMMR	1127.2 (80.1)	**584.5** (53.2)	**231.1** (14.6)
MMR-LS	1127.5 (81.7)	631.9 (52.0)	323.3 (23.4)
RS	1214.0 (79.5)	820.6 (66.9)	540.7 (36.9)

Table 7.9: 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	265.9 (41.4)	301.8 (49.1)	302.4 (46.5)
GA-S	268.6 (42.3)	**304.1** (49.5)	**303.1** (46.6)
RS	258.7 (40.4)	297.1 (48.6)	300.7 (46.3)
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	267.9 (42.7)	300.8 (49.0)	301.8 (46.4)
PSO	**270.0** (42.1)	302.3 (49.2)	301.8 (46.4)

Table 7.10: 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 (30.7)	219.2 (36.2)	262.8 (34.3)
GA-S	153.9 (30.7)	**237.2** (37.0)	**279.8** (36.0)
RS	135.9 (28.7)	195.1 (33.6)	238.3 (32.7)
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	128.7 (30.2)	218.1 (35.6)	262.2 (34.3)
PSO	**156.5** (30.7)	206.8 (36.1)	242.8 (34.2)

Analysis of Obtained Results

In our analysis of the results obtained when applying the algorithms to the target-based problem instances, it should first of all be noted that algorithms with low objective function values perform better than algorithms with high

Table 7.11: 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	96.1 (20.6)	147.2 (30.6)	186.3 (34.6)
GA-S	96.0 (20.8)	**150.2** (29.4)	**208.4** (36.7)
RS	74.9 (18.4)	110.5 (24.7)	143.4 (27.3)
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	59.7 (23.9)	123.1 (28.9)	176.1 (31.7)
PSO	**97.1** (21.2)	123.5 (28.0)	150.3 (29.4)

objective function values, since the static target-based weapon allocation problem has been formulated as a minimization problem (see equation 3.4). Another 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 an optimal algorithm would obtain a quite large standard deviation. This argument is strengthened by the presented averaged ranks, where it is obvious that there is a clear separation in performance among different algorithms.

Starting out the analysis with the random search algorithm, it can be seen that all other algorithms on average perform better on all problem sizes, except for the smallest problem size ($|\mathbf{T}| = 10, |\mathbf{W}| = 10$) tested, on which the maximum marginal return algorithm performs worse. That is also the algorithm that (except for the random search algorithm) produces the worst solutions for $|\mathbf{T}| = 10$. The quality of the solutions produced by the maximum marginal return relative the other algorithms become better as the problem size increases, but it never is the algorithm producing the best solutions, as confirmed in figure 7.3. Nevertheless, the enhanced version of the maximum marginal return algorithm produces solutions of good quality (relative the others) on all the tested problem sizes. Among the three nature-inspired metaheuristic algorithms, the genetic algorithm produces solutions that are very close in quality to the ones produced by the enhanced maximum marginal return algorithm. This is not very surprising since the genetic algorithm is seeded with the solution obtained by the enhanced maximum marginal return algorithm, but it is interesting to see that the quality of the solution returned by the genetic algorithm only is refined in the cases where $|\mathbf{W}| = 10$, while it for larger problem sizes is not

able to improve upon the the solution obtained from the enhanced greedy algorithm. This can be taken as a clear indication of that the search space becomes too large to be searched in such a small amount of time as one second. An algorithm which obviously also is cursed by very large search spaces is the particle swarm optimization algorithm. As can be see in figure 7.3, it is in top on the $(|\mathbf{T}| = 10, |\mathbf{W}| = 10)$ problem size, but gets a much worse rank as the problem size increases. In fact, on the largest tested problem size $(|\mathbf{T}| = 30, |\mathbf{W}| = 30)$, the average objective function value is not much better than that obtained by random search, as seen in table 7.8. Finally, the ant colony optimization algorithm shows a stable behavior as its produced solutions on average are closer to the average solution quality of the best algorithm than the worst, on all tested problem sizes. However, the ant colony optimization algorithm also seems to be better suited for smaller problem sizes, since it for all problem sizes where $|\mathbf{W}| = 10$ is among the top-two algorithms, while it is not for any other of the tested problem sizes (where it always is beaten by at least the seeded genetic algorithm and the enhanced maximum marginal return algorithm). Hence, the utility of using the nature-inspired metaheuristic algorithms is clearly decreasing with an increasing solution space, which is not surprising due to the tight real-time requirements.

In the results from the corresponding experiment with asset-based problem instances, it should be noted that a higher objective function value is better than a low, since it has been formulated as a maximization problem (see equation 3.7) instead of a minimization problem. As explained for the target-based results, the shown standard deviations are a natural consequence of the random generation of problem instances. The random search algorithm is, as in the target-based case, producing solutions of lower quality than the genetic algorithm and the particle swarm optimization algorithm on all tested problem sizes, but here the unguided random search is able to generate better solutions than the two versions of the greedy maximum marginal return heuristic on the tested problem sizes where $|\mathbf{W}| = 10$. This may seem surprising at first glance, but is an indication of that the approximation does not work properly in those cases. These empirical results fit well with the analytical arguments by Hosein (1990), presented in section 4.2.3, who argues that the method of approximating a static asset-based problem with its target-based counterpart should give good results for problem instances with a strong defense (i.e. where the ratio between the number of firing units and the number of targets is high) but that it may give solutions of lower quality for problem instances with a weak defense (i.e. where the ratio between the number of firing units and the number of target is low). For those cases, the genetic algorithm has been able to improve significantly upon the quality of the solutions generated by the enhanced maximum marginal return algorithm. However, as in the target-based case, it has not been able to improve much upon the quality for the problem sizes where the quality of the solutions produced by the enhanced maximum marginal return algorithm is high already. The particle swarm optimization algorithm also shows

consistent behavior with the results obtained from the target-based version of the experiment, in that it performs well in comparison to the other algorithms on the smaller problem instances, but experiences more troubles with larger search spaces.

7.2.4 Effect of Amount of Search Time on the Quality of Obtained Solutions

In order to investigate how much the performance of the algorithms are affected by the tight real-time requirements, we have let the algorithms run for 60 seconds each, on hundred target-based problem instances and hundred asset-based problem instances of size ($|\mathbf{T}| = 20, |\mathbf{W}| = 20$). For this experiment, the following settings have been used in SWARD:

- $T_{start} = 20$, $W_{start} = 20$, $T_{end} = 20$, $W_{end} = 20$, $T_{step} = 1$, $W_{step} = 1$,
- $iterations = 100$, $DAs = 5$, $seed = 0$, $timeLimit = 60000$ms.

(As in the former experiments, the parameter specifying the number of defended assets is only used for creation of the asset-based problem instances.) The average objective function values (and the associated standard deviations) obtained after one second and after one minute are shown in table 7.12 for the target-based instances, and in table 7.13 for the asset-based instances.

In figure 7.5, we have plotted the quality of the best solution found by each algorithm with five second intervals for one of the tested target-based problem instances. Similarly, figure 7.6 shows the corresponding results for one of the tested asset-based problem instances. The values plotted at time 0 are the objective function values of the solutions created in the first iteration. Note that these are not shown for some of the algorithms in figure 7.5, since e.g. the random search and the genetic algorithm in their first iterations have objective function values of higher value than the used maximum vale of the y-axis. The slope of the lines indicate how large the initial objective function values really are for these algorithms.

Analysis of Obtained Results

In figure 7.5, we can see that the highest increase in quality of the generated solutions occur during the first update (i.e. the time between the first iteration and the following five seconds), which is not very surprising given that the values at time 0 originates from one or a couple of solutions created at random for most of the algorithms. Since the maximum return algorithm and the enhanced maximum marginal return algorithm only generate one solution each, the objective function values never change for those. The greedy algorithm that applies local search to the solution returned by the maximum marginal return algorithm is during the first twenty seconds able to improve upon the best solution found,

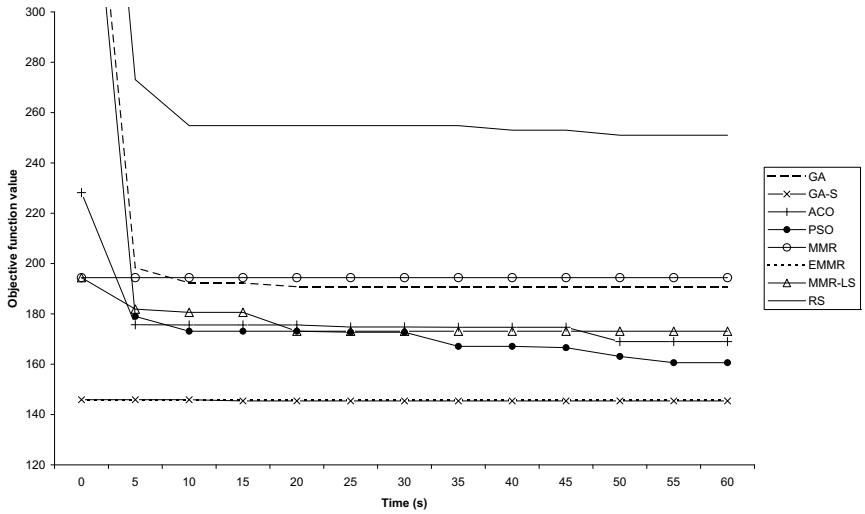


Figure 7.5: Objective function value as a function of time when algorithms are run for 60 seconds on a target-based problem instance of size ($|\mathbf{T}| = 20, |\mathbf{W}| = 20$).

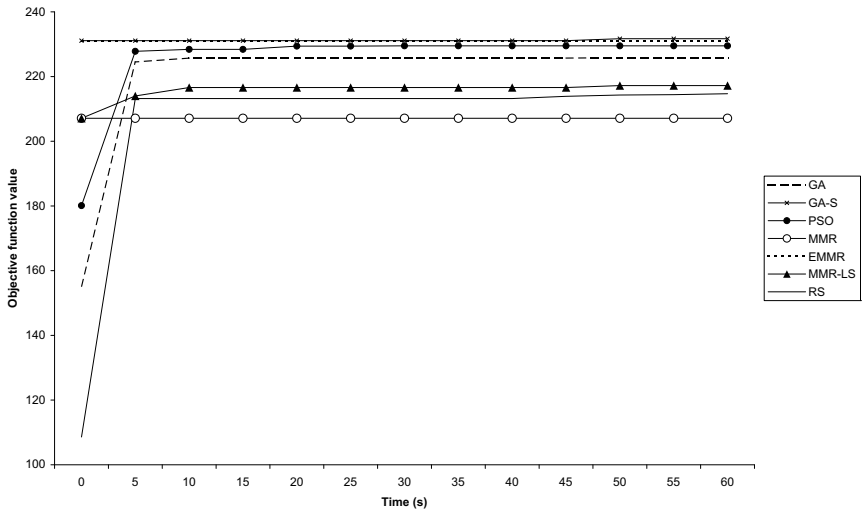


Figure 7.6: Objective function value as a function of time when algorithms are run for 60 seconds on an asset-based problem instance of size ($|\mathbf{T}| = 20, |\mathbf{W}| = 20$).

Table 7.12: Average objective function values for target-based problem instances of size ($|\mathbf{T}| = 20$,$|\mathbf{W}| = 20$), with standard deviations within parentheses.

	1 s	60 s	Improvement
GA	230.8 (21.8)	209.0 (17.9)	9.4%
GA-S	**164.4** (14.8)	**164.4** (14.8)	0%
ACO	203.0 (16.8)	187.4 (15.4)	7.7%
PSO	284.5 (30.2)	173.6 (17.1)	39.0%
MMR	231.4 (21.3)	231.4 (21.3)	-
EMMR	**164.4** (14.8)	**164.4** (14.8)	-
MMR-LS	213.0 (18.9)	197.8 (17.4)	7.1%
RS	312.2 (32.0)	276.9 (26.7)	11.3%

Table 7.13: Average objective function values for asset-based problem instances of size ($|\mathbf{T}| = 20$,$|\mathbf{W}| = 20$), with standard deviations within parentheses.

	1 s	60 s	Improvement
GA	218.3 (33.5)	223.7 (33.8)	2.5%
GA-S	**234.6** (35.5)	**234.9** (35.5)	0.1%
PSO	205.2 (32.4)	234.1 (35.4)	14.1%
MMR	206.4 (33.9)	206.4 (33.9)	-
EMMR	234.4 (35.6)	234.4 (35.6)	-
MMR-LS	215.1 (33.1)	220.7 (34.1)	2.6%
RS	193.4 (30.8)	204.3 (32.1)	5.6%

but after that, the quality is unchanged for the specific example shown in figure 7.5. In a similar way, the genetic algorithm seems to converge to a local optimum after twenty seconds. Still, some of the algorithms are able to eventually escape from such local optima, as can be seen for the ant colony optimization algorithm. Despite that there is a long period of time for which the ant colony optimization algorithm is not able to improve on the solution quality, it is later able to escape from the local optima. The algorithm that most steadily is able to improve upon the solution quality over time is the particle swarm optimization algorithm, and this behavior seems to hold in general for most problem instances and not only the specific run shown here. Similar observations that have been made for the target-based case can also be found in figure 7.6 for the asset-based problem instance. What is most noticeable from the figures is that no algorithm which does not make use of the solution generated by the enhanced maximum marginal return algorithm is able to find solutions with better quality, even though that solution is generated in much less than a second, compared to the 60 seconds used by most other algorithms. The only algorithm that is able to create a somewhat better solution is the seeded genetic

algorithm, indicating that the other algorithms will not be able to outperform the enhanced maximum marginal return algorithm on real-time weapon allocation problems, even if better computer hardware is used than have been used in the experiments presented in section 7.2.2 and section 7.2.3.

The graphs analyzed above give detailed information regarding the algorithms' behavior on a single problem instance, but no information on how the performance of the algorithms is increased in general with an increased search time. Studying the numbers in table 7.12 and 7.13, it is striking that the particle swarm optimization algorithm on average is able to increase its performance by far most for both the target-based and asset-based problem instances. This can partly be explained with that the solutions generated by the particle swarm optimization algorithm after one second were quite bad, but the algorithm is after one minute outperforming the other algorithms, except for the enhanced maximum marginal return algorithm and the seeded genetic algorithm. It is noticeable that the seeded genetic algorithm is not able to improve on the average objective function value at all in the target-based case, and very marginally in the asset-based case, despite the extra search time. The average objective function values for the remaining algorithms are quite far from the ones obtained with the enhanced maximum marginal return algorithm, the seeded genetic algorithm, and the particle swarm optimization algorithm.

7.3 Discussion

Although we have made clear that it almost always is a human decision maker that takes the decision of whether a target should be engaged or not (an exception being the firing of CIWS for terminal defense self-protection against antiship missiles (Neri, 1991)), the experiments with STEWARD have been fully automated, in that no human operator has been involved in the firing decisions sent back to STAGE Scenario. One reason for this is that the human-computer interaction has not been in focus of the PhD project of which this thesis is a result. However, it is nothing in the proposed methodology as such that prevents the involvement of a human decision maker when evaluating TEWA systems by measuring survivability and weapon resource usage in computer-based simulations. On the contrary, this kind of involvement would be needed for evaluation of all aspects of the air defense system of which the operator usually is an important component. However, experiments such as the one presented here are also needed, since the decisions recommended by the automated system otherwise can be overridden by the operator, potentially hiding weaknesses of the tested algorithms.

In the experiments with algorithms for static weapon allocation presented in this chapter, computer hardware of relatively good standard has been used. However, better hardware will be available in the near future, and exists already today. The obtained limits for how large problem sizes that can be solved in real-time using exhaustive search are, hence, not set in stone. Moreover, it is

also easy to distribute the computations over several processors or computers in parallel, in order to speed up the process. Despite this, the exponential increase of the solution space makes sure that heuristic algorithms will be needed also in the future.

In the experiments involving asset-based problem instances, we have in addition to the reported results also tested the approach suggested in Metler and Preston (1990), i.e. to spread out the protection value of a defended asset among all targets aimed for it. This gave somewhat worse performance, and hence, only the results from the approach described in equation 7.2 have been reported. However, it has in the experiments been ensured that the targets are spread out quite evenly over the defended assets, as a result of how the target aims are generated in SWARD (as explained in section 6.2). If target aims instead should have been selected in another way, it could have been more common with problem instances in which a lot of targets were aimed for one and the same target. Such situations are not unrealistic, since e.g. an attack against an aircraft carrier most likely would require a lot of attacking missiles targeted against the carrier in order to be successful, due to the heavy air defense protecting such an important defended asset. Hence, it could be of interest to see whether the used approach of approximating the target value with the product of the lethality probability of the target and the protection value of the target aim would be beneficial also for such problem instances, or if it for such cases is better to spread out the protection value of a defended asset over all targets aimed for it.

It should be noted that the presented results for the various tested nature-inspired metaheuristic algorithms are dependent on the used parameter settings, i.e. we can not infer that the obtained ranking of the algorithms will be the same for all possible selections of crossover operators, mutation rates, population sizes, etc. However, when deciding on which parameter settings to use, different selections have been tested and it seems like the obtained results are robust to at least small changes.

Comparing the obtained results to previously reported experiments, we can confirm that greedy search is indeed superior to (unseeded) genetic algorithms on large-scale target-based weapon allocation problem instances, as suggested by the experiment reported in Julstrom (2009). Likewise, the results obtained by Lee and Lee (2005) indicating that ant colony optimization is better than (unseeded) genetic algorithms for large-scale problems have been shown to hold also when subject to tight real-time constraints. However, it has been shown that the results that particle swarm optimization is better than genetic algorithms for target-based weapon allocation (reported in Teng et al. (2008)) cannot be generalized to hold also for large-scale problems.

7.4 Summary

The experiments in which STEWARD has been used to demonstrate the method for evaluating the performance of TEWA systems has involved four different TEWA configurations. The tested air defense scenario has been small enough to allow for exhaustive search to be used as weapon allocation algorithm, so what have been changed among the configurations are what threat evaluation algorithm to use (Bayesian network or fuzzy logic) and what threshold τ to use for how high target values need to be before the target is considered for weapon allocation (0.5 and 0.7, respectively). For the specific air defense scenario tested, the resource usage cost as expected became lower with the higher threshold, but it was also shown that the survivability became larger when using this higher threshold.

In the experiments with SWARD for evaluation of the real-time performance of the implemented algorithms for static weapon allocation, a number of results have been seen. Firstly, the limit for how large problem sizes that can be solved optimally using exhaustive search has been investigated. This limit in its turn depends on where the limit for real-time goes, but setting the latter limit to somewhere around 1–3 seconds, the upper limit on problem size becomes approximately ($|\mathbf{T}| = 7, |\mathbf{W}| = 7$) on the computer used in the experiments, corresponding to a little bit more than $820{,}000$ feasible solutions. This approximate limit is the same for both the target-based and asset-based formulations, even though the computational cost is somewhat higher for asset-based problem instances of a specific size, compared to their target-based counterparts. Since realistic air defense scenarios can consist of more firing units and targets than can be solved optimally using exhaustive search, we have also investigated the performance of the implemented heuristic algorithms in a number of experiments. In the first of these, the average percentage deviations from the optimal solutions have been tested for the different heuristic algorithms on small-scale problem instances. We could see that the standard maximum marginal return algorithm performed quite bad compared to the other algorithms, and that also the other greedy variants on average performed less good that the nature-inspired metaheuristics. Of these, the particle swarm optimization algorithm and the ant colony optimization algorithm performed best on the target-based weapon allocation problem, while the former was the best also for the asset-based problem. However, as shown in the experiment on larger-scale problems, the particle swarm optimization algorithm encountered difficulties with large search spaces, degenerating its real-time performance quite drastically. Instead, the algorithms producing the best solutions for the large-scale target-based problems were the enhanced maximum marginal return algorithm, and the genetic algorithm seeded with the solution generated from the former. In the case of large-scale asset-based problem sizes, the results are more complex. For problem instances involving a strong defense, the use of the enhanced maximum marginal return algorithm (and thereby also the seeded genetic algo-

rithm) seems appropriate, since this gives solutions of high quality compared to the other algorithms, but for problem instances involving a weak defense, the situation becomes different. In those cases, none of the greedy algorithms give good results. Rather, it is for those cases the nature-based algorithms that produce the best solutions. For the instances where the search space is of reasonable size, the particle swarm optimization algorithm performs best, while it in situations involving both a weak defense and a large search space is the genetic algorithm that performs the best.

Chapter 8
Conclusions and Future Work

The work which has been reported within this thesis is focused on the development and evaluation of TEWA systems, and their underlying algorithms for threat evaluation and weapon allocation. As highlighted, a fundamental problem is that the threat evaluation and weapon allocation processes need to be executed in real-time, a fact that is disregarded in most earlier research on weapon allocation algorithms.

In section 8.1, we summarize the contributions which have been made in the work presented here. An outline of future work is presented in section 8.2. Finally, a discussion regarding how the work and the achieved results presented here potentially can be generalized to other research domains or applications is given in section 8.3.

8.1 Contributions

In this doctoral thesis, a number of contributions have been made. We are here describing the contributions made in context to the objectives that were presented in section 1.1:

O1. *Review and analyze existing algorithms and methods for threat evaluation and weapon allocation.*

O2. *Suggest and implement algorithms for real-time threat evaluation and weapon allocation.*

O3. *Suggest methods for evaluating the performance of TEWA systems, and the threat evaluation and weapon allocation algorithms being part of such systems.*

O4. *Develop testbeds for performance evaluation.*

O5. *Evaluate the performance of the developed algorithms.*

Review and analyze existing algorithms and methods for threat evaluation and weapon allocation

Two literature surveys have been undertaken, in which existing algorithms for threat evaluation and weapon allocation, respectively, have been reviewed. The reviews are important, both in their own right and in combination, since this thesis to the best of the author's knowledge is the first work that thoroughly treats both threat evaluation and weapon allocation, and thereby provides a rare unified view on TEWA, i.e. threat evaluation and weapon allocation. Traditionally, threat evaluation is studied separately within the information fusion domain, while weapon allocation is studied in isolation within the field of operations research. However, both threat evaluation and weapon allocation functionality are needed within air defense systems, and hence, it makes very good sense to study them together. Moreover, there are interdependences between the processes of threat evaluation and weapon allocation, since estimated target values play an important role when it is decided to which target a specific firing unit should be allocated. When deciding on how high target values have to be for targets to become subject for weapon allocation, a comprehension of the underlying threat evaluation algorithms is necessary.

In the reviews it was found out that the amount of open literature on the research topic of threat evaluation is very sparse. In the previous existing work, four main approaches to threat evaluation could be identified. These are:

- classical (two-valued) logic
- fuzzy logic
- artificial neural networks
- graphical models

Of the above, the three last seem more promising since it is not possible to handle uncertainty using classical logic, which is a strong disadvantage since data used as input to threat evaluation typically originate from sensor observations which are imperfect by nature. Artificial neural networks are opaque models which in general are not transparent to the human user. Moreover, they are dependent on large amounts of labeled data which is hard to obtain for air defense scenarios. For these reasons, the fuzzy logic approach and the use of graphical models have been identified as being most promising for use in TEWA systems.

The review of literature on weapon allocation revealed that a lot more research has been published on the topic of weapon allocation than on threat evaluation. Many of the early publications on weapon allocation, of which some dates back as far as to the late 1950's, mainly focus on analytical approaches to special cases of the more general static target-based weapon allocation problem (e.g. problems including additional constraints such as at most

one firing unit can be allocated to each target). In more recent years, the focus has shifted towards more advanced computer-based heuristic algorithms that are better suited for real-world air defense problems. This latter kind of algorithms are also what the review of static weapon allocation algorithms presented in this thesis focuses on, due to its higher relevance for TEWA systems, and since there already are many existing surveys covering different analytical approaches.

Despite the vast amount of algorithms for static weapon allocation suggested within existing literature, it is not earlier well enough researched how the suggested algorithms are affected by the real-time requirements that are fundamental in real-world air defense scenarios. Of the existing heuristic algorithms for static weapon allocation, a number of techniques have been identified as promising (before any actual experiments were performed). These are e.g. genetic algorithms, ant colony optimization algorithms, particle swarm optimization algorithms, and more greedy algorithms such as the maximum marginal return algorithm.

Suggest and implement algorithms for real-time threat evaluation and weapon allocation

In the second objective, a number of the identified algorithms for threat evaluation and static weapon allocation have been implemented. Since most algorithms that have been presented for static weapon allocation have been described quite briefly in existing literature, this objective was far from straightforward to carry out. For small problem instances, an optimal (in the sense that the optimal solution always is found) exhaustive search algorithm has been implemented. Moreover, a number of heuristic algorithms have been implemented, as follows. A genetic algorithm has been developed and implemented that probably is quite similar to existing genetic algorithms for weapon allocation, but which has been tailored for real-time requirements when it comes to the choice of selection mechanism, mutation and crossover operators, etc. Similarly, a particle swarm optimization algorithm has been developed, which has many features in common with Zeng et al. (2006) and Teng et al. (2008). A third nature-based optimization algorithm which has been implemented for static target-based weapon allocation is an ant colony optimization algorithm, based on a description presented in Lee et al. (2002a). We have also implemented a simple random search algorithm which can be used as a baseline for comparisons, as well as a number of greedy algorithms. Of the latter, the well-known maximum marginal return algorithm presented in den Broeder et al. (1959) and the enhanced version of the previous algorithm, presented in Julstrom (2009), have been implemented. Moreover, we have also implemented a local search algorithm which take the solution generated by the maximum marginal return algorithm as input, and refines this using local search. Lastly, a version of the genetic algorithm discussed above, which is seeded by the solu-

tion returned by the greedy enhanced maximum marginal return algorithm has been implemented. All of the implementations have been made public available, as part of the SWARD testbed.

On the threat evaluation side, two different kinds of algorithms have been suggested and implemented. These are a developed Bayesian network, in which threat values are inferred from observations regarding target type, speed, distance, etc., and a fuzzy logic approach which is based on descriptions given in Liang (2006) and Liang (2007). In the latter, the used input variables and rules have been changed in order to match the variables used in the developed Bayesian network algorithm.

Suggest methods for evaluating the performance of TEWA systems, and the threat evaluation and weapon allocation algorithms being part of such systems

In order to be able to evaluate the real-time performance of the implemented algorithms, systematic benchmarking is needed. Some comparisons of algorithms for weapon allocation can be found in existing literature, however, these comparisons very rarely take real-time requirements into consideration. Moreover, obtained results cannot be used for benchmarking against other algorithms since not enough details are given on how the problem instances used for performance evaluation have been generated (i.e. the problem instances cannot be recreated). The situation is even more troublesome for evaluation of threat evaluation algorithms. At present, algorithms for threat evaluation are (in best case) only evaluated very briefly. When developed algorithms are evaluated, they are only tested on a very small number of simple air defense scenarios, in which the target values produced by the algorithms are compared to those estimated by some human expert on the same scenarios. For those deficiencies to be handled, a methodology for evaluating TEWA systems and their components (e.g. threat evaluation and weapon allocation algorithms) has been suggested. This methodology relies on the use of computer-based simulations, in which the survivability of defended assets and the usage of defensive weapon resources is measured. The developed methodology is also a contribution to the information fusion community, since the importance of more systematic evaluation of proposed high-level information fusion algorithms and applications often is highlighted, but that concrete examples of evaluation methodologies earlier have been lacking.

Develop testbeds for performance evaluation

The suggested methodology described above has been implemented into the prototype testbed STEWARD (System for Threat Evaluation and Weapon Allocation Research and Development), consisting of a TEWA module in which threat evaluation and static weapon allocation algorithms have been implemented, and the commercial simulation engine STAGE Scenario, in which it is

possible to create and run dynamic and interactive air defense scenarios. These modules have been integrated with each other so that sensed targets within the scenario continuously are communicated to the TEWA module for threat evaluation, and if necessary, weapon allocation. Suggested allocations are communicated back to STAGE Scenario, in which the firing units are fired in accordance to the received firing orders.

Additionally, an open source testbed for systematic comparisons of the performance of static weapon allocation algorithms has been developed. The name of this second testbed is SWARD (System for Weapon Allocation Research and Development), which has been chosen to contrast it to STEWARD. The developed algorithms for static weapon allocation presented in this thesis have been implemented into SWARD, and the testbed has been made publicly available so that it can be downloaded from the web page:

`http://sourceforge.net/projects/sward/`

The testbed indirectly gives access to standardized data sets (through the use of pseudo-randomly generated problem instances), so that researchers on different occasions (independently of where they are and which machine they use) easily can run newly developed algorithms on problem instances which already have been used by other researchers when testing other algorithms, so that a better understanding of how different algorithms perform in relation to each other under various conditions (e.g. real-time requirements) can be created.

Evaluate the performance of the developed algorithms

The prototype testbed STEWARD has been used to demonstrate the idea of using computer-based simulations for evaluating the effectiveness of different TEWA system configurations. An air defense scenario created in STAGE Scenario has been used to evaluate a total of four different configurations, in which exhaustive search has been used for weapon allocation, while both the Bayesian network approach and the fuzzy logic approach have been tested for threat evaluation. Moreover, two different thresholds for how high a target value must be for a target to become subject for weapon allocation have been tested.

The open source testbed SWARD has been used to systematically compare the real-time performance of the implemented algorithms for static weapon allocation on a large number of problem instances (scenarios) of various size. It has in the experiments been shown that exhaustive search can be usable for target-based as well as asset-based real-time allocation of firing units to targets in situations where the number of targets and firing units are (approximately) seven or less, but that other kinds of approaches are needed for problem instances of larger size. The performance of the implemented heuristic algorithms have been tested on both small-scale and large-scale problems. On the small-scale problems, the nature-inspired metaheuristics performed best, and of these, the particle swarm optimization algorithm was the winner for both

target-based and asset-based weapon allocation, since it produced optimal or very near-optimal solution for all the small-scale problem instances tested, i.e. problem sizes up to ($|\mathbf{T}| = 9, |\mathbf{W}| = 9$). The worst performing algorithm on the small-scale problem instances was the greedy maximum marginal return algorithm. However, on the tested problems of larger scale, the results looked different. It has been shown that the utility of using the implemented nature-based optimization algorithms quickly decreases when the problem size is increased, given that the solutions must be produced in real-time. For larger-scale target-based problem instances, the enhanced maximum marginal return algorithm (and thereby also the seeded genetic algorithm) generated the solutions of best quality, while the results were different for the asset-based case. In situations with a strong defense, i.e. a large number of firing units compared to the number of targets, the enhanced maximum marginal return algorithm provides solutions of good quality compared to the others, but for situations involving a weak defense, its produced results become poor. Overall, the seeded greedy algorithm produced the best solutions here, but there are indications that none of the algorithms produce solutions of especially good quality on these problem instances. Lastly, experiments have been performed in which it has been investigated whether the quality of the solutions improve significantly if the tight real-time requirements are relaxed (i.e. when the algorithms are allowed to search a longer time for good solutions). It has been shown that most of the algorithms were able to improve quite much on the average solution quality, but still could not produce better solutions than the seeded genetic algorithm, despite that the latter could not improve at all during the extra search time for the target-based instances, and only very marginally on the asset-based instances. The algorithm improving the quality of its generated solutions by far most was the particle swarm optimization algorithm, producing solutions that were almost as good as the seeded genetic algorithm when run for one minute.

8.1.1 Summary of Contributions

In this thesis, we formally describe the processes of threat evaluation and weapon allocation. The threat evaluation process is previously quite poorly defined, and a better understanding of this process is necessary for better threat evaluation algorithms to be constructed. Reviews of the available literature on threat evaluation and weapon allocation have been done, providing knowledge of different parameters that can be used for threat evaluation, and identification of classes of algorithms that can be suitable for threat evaluation and weapon allocation, respectively. Based on the results from these literature reviews, algorithms for threat evaluation and static weapon allocation have been implemented. The implementation details have been made public in order to make it possible for other researchers to use the same algorithms.

An important question raised in this doctoral thesis is how to evaluate the performance of TEWA systems and the threat evaluation and weapon allo-

cation algorithms being part of such systems? The testbed SWARD has been developed for making it possible to compare the performance of various algorithms for static weapon allocation, and its source code has been released so that other researchers can add more algorithms to the testbed, improve it, and make their own experiments using the testbed. For the comparison of threat evaluation algorithms as well as complete TEWA systems, an evaluation methodology involving the use of computer-based simulation has been proposed. By calculating the survivability of defended assets and the resource usage cost, an effectiveness metric can be used for comparing different TEWA system configurations. The suggested methodology has resulted in the prototype testbed STEWARD. It has been demonstrated how the developed testbeds can be used, and we have provided experimental results showing that nature-based metaheuristic algorithms outperform greedy search on small-scale static weapon allocation problem instances, but that the increase of the search space quickly decreases the real-time performance of the more advanced nature-based algorithms when the complexity of the problem instances is increased.

8.2 Future Work

As have been made clear, the scope of the weapon allocation part of this thesis has been restricted to static weapon allocation. However, it is in real-world air defense situations only specific circumstances in which all firing units will be allocated simultaneously. Examples of such situations are air defense situations involving rockets, artillery, or mortars fired from a short distance. For many other situations there are multiple engagement opportunities. The approach that has been used in this work for handling long dynamic scenarios with multiple engagement opportunities has been to iteratively sense the environment, estimate the threat posed by the detected targets to the defended assets, and to allocate available firing units only to targets that achieves a target value higher than a predefined threshold. A problem with this approach is that even if each individual static allocation is made optimal, the combination of all individual static weapon allocation decisions may not be optimal for the whole chain of defense processes (Chen et al., 2009). Another approach to handle dynamic scenarios is to treat the allocation problem as a dynamic optimization problem (cf. Hosein (1990)), in which a so-called shoot-look-shoot (SLS) strategy is used, i.e. the defense is divided into a number of phases, in which the defenders can engage the targets, observe the outcome of the previous engagements, engage again, and so on. As for the static case, it is not well-known how the algorithms suggested within existing literature perform in comparison to each other. It is therefore of large interest to conduct a study for dynamic weapon allocation similar to the one presented for the static version of the problem in this thesis. Likewise, for making such comparisons, a testbed similar to SWARD is needed, in which it should be possible to generate problem instances for dynamic weapon allocation.

The suggestion of the methodology based on measuring survivability and resource usage costs, and the development of the testbed STEWARD are important steps for being able to evaluate TEWA systems in a more systematic manner, which is necessary due to consequences it may have to take the wrong decision in this kind of systems. However, before this kind of methodology can be used in larger scale, it is necessary that a large amount of representative air defense scenarios can be created. As discussed in section 5.5, this might be achieved by the creation of air defense scenarios templates on a high level of abstraction, but whether this really is a suitable way, and if so, how such templates should be constructed, are research questions that need to be answered. Another improvement that is needed if STEWARD should become more than a proof-of-concept of the idea to measure the effectiveness of TEWA systems through the use of computer-based simulations is that sensor models that simulate the performance of real sensors should be added to the simulation engine, and replace the "ground truth" data that currently is sent from the simulator to the TEWA module. Another possible improvement of STEWARD for the future is to replace the used simulation engine with one based on open source code, so that STEWARD just as SWARD gets the possibility to be used by a larger group of researchers and developers.

Another direction for future research that has been briefly discussed is to use the closed-loop simulations in STEWARD to allow for automated reinforcement learning, which can be used to fine-tune the actual parameter settings of threat evaluation and weapon allocation algorithms. For such an approach to be of any real value, it is necessary that the air defense scenarios can be made very realistic, but it is definitely an unexplored research subject that can be of interest for the future.

Potential improvements of SWARD that have been discussed are to add other weapon allocation algorithms into the testbed. An important example of such an algorithm is the VLSN algorithm suggested in Ahuja et al. (2007). This algorithm seems to have great potential based on the computational results reported in Ahuja et al. (2007), but it has not been implemented in the work presented here since the algorithmic details have not been described thoroughly enough to allow for reimplementation by the author at this point. Other algorithms that, based on the results presented in this dissertation can be fruitful for future research, are so called greedy randomized adaptive search procedure (GRASP) metaheuristic algorithms. Algorithms that implement the GRASP metaheuristic typically construct the solution by iterative generation of greedy randomized partial solutions which are improved iteratively using local search. In this sense it is related to how we have generated solutions using a maximum marginal return algorithm and improved it using a genetic algorithm. Hence, the greedy part of a GRASP algorithm could rely on a maximum marginal return algorithm, since this kind of algorithms have shown good results for large-scale problem instances in our experiments. The main difference would be that the random component in a GRASP approach possibly allows

for getting out of local optima faster than a genetic algorithm. To the best of the author's knowledge, GRASP approaches have not earlier been suggested for weapon allocation.

8.3 Generalization to Other Research Areas

The work presented in this thesis has been constrained to threat evaluation and weapon allocation in the air defense domain. The thesis has deliberately been quite focused, but it is now time for elaborating on how the results obtained here potentially can be generalized to other research domains or applications.

A research problem adjacent to weapon allocation is the problem of allocating unmanned aerial vehicles (UAVs) and other kinds of sensors and sensor platforms to different tasks or entities. This problem is very similar to the weapon allocation problem since it deals with integer optimization problems that need to be solved in real-time within the defense domain.

The algorithms developed for weapon allocation in this thesis can also be of interest for domains that are very different from the military domain. As an example of this, the static target-based weapon allocation problem has in Cetin and Esen (2006) been used as a basis to allocate media vehicles (such as TV, radio, Internet, newspaper, billboards, etc.) to audience segments in advertising campaigns. Obviously, the real-time requirements are far from being as important in the advertising domain as in the air defense domain. However, this is an illustrative example of that resource allocation problems can be found in various domains. Another example of a related resource allocation problem, given in Gelenbe et al. (2010), is the optimization problem of allocating a set of ambulances or emergency personnel to a set of emergencies, where the goal is to maximize the number of collected injured individuals while the response time is minimized, and where it is uncertainty involved in whether an emergency unit will be able to reach a targeted emergency or not. Obviously, the real-time performance can be of uttermost importance in such problems.

When it comes to the developed algorithms for threat evaluation, the possible generalizations are perhaps not as obvious as in the weapon allocation case. However, even there it is possible to find related problems of high relevance. An example of this is threat detection algorithms to be used at airports for flagging of individuals for additional screening of baggage, based on e.g. flight records. An example of such a system being in use already today is the so called Computer Assisted Passenger Pre-Screening System (CAPPS), developed by the US Federal Aviation Administration (Witten and Frank, 2005). Moreover, we have here only discussed threat evaluation in an air defense context, and mostly from a ground-based perspective. However, threat evaluation can be very relevant for other military contexts as well, e.g. as decision support for fighter pilots, as well as on naval vessels (to which some extent has been discussed) and for active protection systems on e.g. tanks.

The results obtained in this thesis show that nature-inspired optimization algorithms such as particle swarm optimization and ant colony optimization are not very useful for large-scale weapon allocation problem instances, when the time allowed for search is very constrained. For this kind of large search spaces, it is more useful to use greedy heuristics when subject to tight real-time constraints. These results are likely to be generalizable to other kinds of real-time optimization problems as well. Although not many other applications will require as tight search time constraints as present in real-time air defense, it seems (based on the obtained results) likely that greedy algorithms are to be preferred to nature-inspired ones also for other large-scale optimization problems in which time constraints are of crucial importance.

Appendix A
Bayesian Network for Threat Evaluation

In this appendix, implementation details for the developed Bayesian network for threat evaluation are given, so that other researchers can implement the same network for comparison purposes. In tables A.1–A.8, the conditional probability tables that have been used in the Bayesian network are shown.

Table A.1: Conditional probability table for $Target\ type$.

$F16$	$Mig21$	$B2$	$B747$
0.30	0.20	0.05	0.45

Table A.2: Conditional probability table for $Weapon\ range$.

$Target\ type$	$none$	$short$	$medium$	$long$
F16	0.05	0.20	0.30	0.45
Mig21	0.05	0.35	0.55	0.05
B2	0.05	0.80	0.15	0.00
B747	0.99	0.01	0.00	0.00

The conditional probability table for $Within\ weapon\ envelope?$ has been constructed in Netica using the following expression:

$$WeapEnv(WeapRange, Dist) = \begin{array}{l} Dist - WeapRange \leq 0?Within : \\ Dist - WeapRange \leq 20000?Close : \\ Far \end{array}$$

Table A.3: Conditional probability table for *Speed*.

Target type	*low*	*medium*	*high*
F16	0.15	0.45	0.40
Mig21	0.20	0.70	0.10
B2	0.95	0.05	0.00
B747	0.98	0.02	0.00

Table A.4: Conditional probability table for *Capability*.

Weapon envelope	*Target type*	*high*	*medium*	*low*
within	F16	0.90	0.10	0.00
within	Mig21	0.80	0.20	0.00
within	B2	0.99	0.01	0.00
within	B747	0.05	0.10	0.85
close	F16	0.70	0.20	0.10
close	Mig21	0.50	0.30	0.20
close	B2	0.65	0.20	0.15
close	B747	0.00	0.05	0.95
far	F16	0.10	0.40	0.50
far	Mig21	0.05	0.35	0.60
far	B2	0.03	0.30	0.67
far	B747	0.00	0.02	0.98

Table A.5: Conditional probability table for *Intent*.

high	*medium*	*low*
0.25	0.30	0.45

Table A.6: Conditional probability table for *Threat*.

Capability	*Intent*	*true*	*false*
high	high	0.90	0.10
high	medium	0.70	0.30
high	low	0.20	0.80
medium	high	0.75	0.25
medium	medium	0.50	0.50
medium	low	0.10	0.90
low	high	0.50	0.50
low	medium	0.25	0.75
low	low	0.03	0.97

Table A.7: Conditional probability table for $Distance$.

$Intent$	$very\ close$	$close$	$medium$	far	$very\ far$
high	0.35	0.30	0.20	0.10	0.05
medium	0.15	0.25	0.35	0.20	0.05
low	0.05	0.20	0.25	0.25	0.25

Table A.8: Conditional probability table for TBH.

$Intent$	$very\ short$	$short$	$medium$	$long$	$very\ long$
high	0.35	0.30	0.20	0.10	0.05
medium	0.15	0.25	0.30	0.25	0.05
low	0.05	0.10	0.20	0.35	0.30

References

Emile Aarts and Jan Karel Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley & Sons Ltd., 1997.

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.

Mohamad Khaled Allouche. Real-time use of Kohonen's self-organizing maps for threat stabilization. *Information Fusion*, 6:153–163, 2005.

Mohamad Khaled Allouche. A pattern recognition approach to threat stabilization. Technical report, DRDC Valcartier, June 2006.

Richard D. Amori. An adversarial plan recognition system for multi-agent airborne threats. In *SAC '92: Proceedings of the 1992 ACM/SIGAPP Symposium on Applied computing*, pages 497–504. ACM Press, 1992.

Andreas Antoniou and Wu-Sheng Lu. *Practical Optimization: Algorithms and Engineering Applications*. Springer, 2007.

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

Arthur Asuncion and David Newman. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/ml/`.

Mustafa Azak and Ahmet Engin Bayrak. A new approach for threat evaluation and weapon assignment problem, hybrid learning with multi-agent coordination. In *Proceedings of the 23rd International Symposium on Computer and Information Sciences*, 2008.

Patrick Beaumont. Multi-platform coordination and resource management in command and control. Master's thesis, Faculté des sciences et de génie Université Laval, Québec, 2004.

Benjamin Bell, Eugene Santos Jr, and Scott M. Brown. Making adversary decision modeling tractable with intent inference and information fusion. In *Proceedings of the 11th Conference on Computer Generated Forces and Behavioral Representation*, pages 535–542, Orlando, FL, 2002.

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.

Abder Rezak Benaskeur, Froduald Kabanza, Eric Beaudry, and Mathieu Beaudoin. A probabilistic planner for the combat power management problem. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling*, 2008.

Alessio Benavoli, Branko Ristic, Alfonso Farina, Martin Oxenham, and Luigi Chisci. An approach to threat assessment based on evidential networks. In *Proceedings of the 10th International Conference on Information Fusion*, 2007.

Alessio Benavoli, Branko Ristic, Alfonso Farina, Martin Oxenham, and Luigi Chisci. An application of evidential networks to threat assessment. *IEEE Transactions on Aerospace and Electronic Systems*, 45:620–639, 2009.

Mikael Berndtsson, Jörgen Hansson, Björn Olsson, and Björn Lundell. *Planning and Implementing your Final Year Project - with Success!: A Guide for Students in Computer Science and Information Systems*. Springer, 2002.

Raj Bhatnagar and Laveen N. Kanal. *Fuzzy logic for the management of uncertainty*, chapter Models of enquiry and formalisms for approximate reasoning, pages 29–54. John Wiley & Sons, Inc., 1992.

Samuel S. Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 19(1):5–18, 2004.

Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization. *ACM Computing Surveys*, 35(3):268–308, 2003. ISSN 0360-0300.

George N. Brander and E. J. Bennet. Real-time rule based resource allocation in naval command and control. In *IEE Colloquium on Rule-Based Systems for Real-Time Planning and Control*, 1991.

Berndt Brehmer. The dynamic OODA loop: Amalgamating Boyd's OODA loop and the cybernetic approach to command and control. In *Proceedings of the Tenth International Command and Control Research and Technology Symposium (ICCRTS)*, McLean, Virginia, June 2005.

British Ministry of Defence. Aircraft accident to Royal Air Force Tornado GR MK4A ZG710, May 2004.

Joel Brynielsson. *A Gaming Perspective on Command and Control*. PhD thesis, School of Computer Science and Communication, Royal Institute of Technology, Stockholm, Sweden, June 2006.

Eyüp Cetin and Seda Tolun Esen. A weapon-target assignment approach to media allocation. *Applied Mathematics and Computation*, 175:1266–1275, 2006.

Bruce A. Chalmers and P. da Ponte. MIMD algorithms for naval resource planning: overview and preliminary assessment. Technical report, Defense Research Establishment Valcartier, 1995.

Stephen J. Chapman and Kurt K. Benke. Assessment of ship air defence performance by modelling and simulation. In *Proceedings of the Simulation Technology and Training Conference (SimTecT 2000)*, 2000.

Jie Chen, Bin Xin, ZhiHong Peng, LiHua Dou, and Juan Zhang. Evolutionary decision-makings for the dynamic weapon-target assignment problem. *Science in China Series F: Information Sciences*, 52(11):2006–2018, 2009.

Chow Kay Cheong. Survey of investigations into the missile allocation problem. Master's thesis, Naval Postgraduate School, Monterey, CA, 1985.

Camelia Ciobanu and Gheorghe Marin. On heuristic optimization. *An. Stiint. Univ. Ovidius Constanta*, 9(2):17–30, 2001.

Paul R. Cohen. *Empirical Methods for Artificial Intelligence*. The MIT Press, 1995.

Gregory F. Cooper. Probabilistic inference using belief networks is NP-hard. *Artificial Intelligence*, (42):393–405, 1990.

Ian W. Dall. Threat assessment without situation assessment. In Robin Evans, Lang White, Daniel McMichael, and Len Sciacca, editors, *Proceedings of Information Decision and Control 99*, pages 365–370, Adelaide, Australia, February 1999. Institute of Electrical and Electronic Engineers, Inc.

Randall Davis, Bruce Buchanan, and Edward Shortliffe. Production rules as a representation for a knowledge-based consultation program. *Artificial Intelligence*, 8(1):15–45, 1977.

Christian W. Dawson. *The Essence of Computing Projects: A Student's Guide*. Prentice Hall, 2000.

Defense Science Board Task Force. *Report of the Defense Science Board Task Force on Patriot System Performance : Report Summary*. United States Department of Defense, 2005.

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

DoD Chief Information Officer Memorandum. Clarifying guidance regarding open source software (OSS), October 2009. URL `http://cio-nii.defense.gov/docs/OpenSourceInDoD.pdf`.

An Xiao Dong and Liu Gou Qing. Application of neural network in the field of target threat evaluation. In *International Joint Conference on Neural Networks*, 1999.

Marco Dorigo. *Optimization, learning and natural algorithms (in Italian)*. PhD thesis, Politecnico di Milano, 1992.

Marco Dorigo and Luca Maria Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. The MIT Press, 2004.

Marek J. Druzdzel and Linda C. van der Gaag. Building probabilistic networks: "where do the numbers come from?". *IEEE Transactions on Knowledge and Data Engineering*, 12(4):481–486, 2000.

A. Ross Eckler and Stefan A. Burr. Mathematical models of target coverage and missile allocation. Technical Report DTIC: AD-A953517, Military Operations Research Society, Alexandria, VA, 1972.

Mica R. Endsley. Design and evaluation for situation awareness enhancement. In *Proceedings of the Human Factors Society 32nd Annual Meeting*, pages 97–101, Santa Monica, California, 1988.

Mica R. Endsley. Toward a theory of situation awareness in dynamic systems. *Human Factors*, 37(1):32–64, 1995.

Andries P. Engelbrecht. *Computational Intelligence: An Introduction*. John Wiley & Sons, Ltd., 2002.

Lucia Falzon. Using Bayesian network analysis to support centre of gravity analysis in military planning. *European Journal of Operational Research*, 170(2):629–643, 2006.

Craig W. Fisher and Bruce R. Kingma. Criticality of data quality as exemplified in two disasters. *Information & Management*, 39:109–116, 2001.

Pek Hui Foo, Gee Wah Ng, Khin Huang, and Rong Yang. Application of intent inference for air defense and conformance monitoring. *Journal of Advances in Information Fusion*, 4(1):3–26, 2009.

Lance Fortnow. The status of the P versus NP problem. *Communications of the ACM*, 52(9):78–86, 2009.

Anissa Frini, Adel Guitouni, Abder Rezak Benaskeur, and Éloi Bossé. Single ship resource allocation in above water warfare. Technical Report TR 2006-766, DRDC Valcartier, 2008.

Daniel Fu, Emilio Remolina, Jim Eilbert, and Stottler Henke. A CBR approach to asymmetric plan detection. In *KDD Workshop on Link Analysis for Detecting Complex Behavior*, 2003.

Erol Gelenbe, Stelios Timotheou, and David Nicholson. Fast distributed near-optimum assignment of assets to tasks. *The Computer Journal*, 2010. doi: 10.1093/comjnl/bxh000.

Paul G. Gonsalves, Janet E. Burge, and Karen A. Harper. Architecture for genetic algorithm-based threat assessment. In *Proceedings of the Sixth International Conference on Information Fusion*, 2003.

Fredrik Gustafsson, Fredrik Gunnarsson, Niclas Bergman, Urban Forssell, Jonas Jansson, Rickard Karlsson, and Per-Johan Nordlund. Particle filters for positioning, navigation and tracking. *IEEE Transactions on Signal Processing*, 50(2):425–437, 2002.

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

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

David Hall and Alan Steinberg. Dirty secrets in multisensor data fusion. In *Proceedings of the National Symposium on Sensor Data Fusion (NSSDF)*, San Antonio, TX, USA, June 2000.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.

John H. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, 1975.

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.

Patrick A. Hosein and Michael Athans. Some analytical results for the dynamic weapon-target allocation problem. Technical Report AD-A219281, MIT, 1990a.

Patrick A. Hosein and Michael Athans. Preferential defense strategies: Part 1 - the static case. Technical report, Massachusetts Institute of Technology, Laboratory for Information and Decision Systems, 1990b.

Chi Huaiping, Liu Jingxu, Chen Yingwu, and Wang Hao. Survey of the research on dynamic weapon-target assignment problem. *Journal of Systems Engineering and Electronics*, 17(3):559–565, 2006.

Cecil Huang and Adnan Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, 1996.

Finn V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001. ISBN 0387952594.

Krishna C. Jha. *Very large-scale neighborhood search heuristics for combinatorial optimization problems*. PhD thesis, University of Florida, 2004.

Fredrik Johansson and Göran Falkman. Implementation and integration of a Bayesian network for prediction of tactical intention into a ground target simulator. In *Proceedings of the 9th International Conference on Information Fusion*, Florence, Italy, July 2006.

Fredrik Johansson and Göran Falkman. Detection of vessel anomlies - a Bayesian network approach. In *Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 2007.

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*, 2008a.

Fredrik Johansson and Göran Falkman. A comparison between two approaches to threat evaluation in an air defense scenario. In Vicenc Torra and Yasuo Narukawa, editors, *Proceedings of the 5th International Conference on Modeling Decisions for Artificial Intelligence*, volume 5285 of *Lecture Notes in Artificial Intelligence*, pages 110–121, 2008b.

Fredrik Johansson and Göran Falkman. A survivability-based testbed for comparing threat evaluation algorithms. In Henrik Boström, Ronnie Johansson, and Joeri van Laere, editors, *Proceedings of the 2nd Skövde Workshop on Information Fusion Topics*, 2008c.

Fredrik Johansson and Göran Falkman. A testbed based on survivability for comparing threat evaluation algorithms. In S. Mott, J. F. Buford, G. Jakobson, and M. J. Mendenhall, editors, *Proceedings of the SPIE Symposium on Defense, Security and Sensing*, volume 7352 (Intelligent Sensing, Situation Management, Impact Assessment, and Cyber-Sensing), 2009a.

Fredrik Johansson and Göran Falkman. An empirical investigation of the static weapon-target allocation problem. In *Proceedings of the 3rd Skövde Workshop on Information Fusion Topics*, 2009b.

Fredrik Johansson and Göran Falkman. Performance evaluation of TEWA systems for improved decision support. In Vicenc Torra, Yasuo Narukawa, and Masahiro Inuiguchi, editors, *Proceedings of the 6th International Conference on Modeling Decisions for Artificial Intelligence*, volume 5861 of *Lecture Notes in Artificial Intelligence*, pages 205–216, 2009c.

Fredrik Johansson and Göran Falkman. A suite of metaheuristic algorithms for static weapon-target allocation. In *Proceedings of the 2010 International Conference on Genetic and Evolutionary Methods*, 2010a.

Fredrik Johansson and Göran Falkman. SWARD: System for weapon allocation research & development. In *Proceedings of the 13th International Conference on Information Fusion*, 2010b.

Joint Chiefs of Staff. Joint publication 3-01: Countering air and missile threats, Feb 2007.

Charles C. Jorgensen and Michael H. Strub. Analysis of manual threat evaluation and weapons assignment (TEWA), in the AN/TSQ-73 air defense system. Technical Report TR 419, U.S. Army, Research Institute for the Behavioral and Social Sciences, October 1979.

Bryant A. Julstrom. String- and permutation-coded genetic algorithms for the static weapon-target assignment problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2009)*, 2009.

Rudolf E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

Orhan Karasakal. Air defense missile-target allocation models for a naval task group. *Computers and Operations Research*, 35(6):1759–1770, 2008. ISSN 0305-0548.

Fakhreddine O. Karray and Clarence de Silva. *Soft computing and intelligent systems design*. Addison-Wesley, 2004.

James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

Deepak Khosla. Hybrid genetic approach for the dynamic weapon-target allocation problem. In *Proceedings of SPIE*, volume 4396, 2001.

Uffe B. Kjærulff and Anders L. Madsen. *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis*. Springer, 2007.

George J. Klir and Tina A. Folger. *Fuzzy sets, uncertainty, and information*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1987.

Wolfgang Koch. On exploiting negative sensor evidence for target tracking and sensor data fusion. *Information Fusion*, 8(1):28–39, 2007.

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

Alexander Kott, Martha Pollack, and Bruce Krogh. The situation assessment problem: Toward a research agenda. In *Proceedings of the DARPA-JFACC Symposium on Advances in Enterprise Control*, 1999.

Vladik Kreinovich and Hung T. Nguyen. Which fuzzy logic is the best: Pragmatic approach (and its theoretical analysis). *Fuzzy Sets and Systems*, 157 (5):611–614, March 2006.

Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 231–238. MIT Press, 1995.

Rudolf Kruse, Joan E. Gebhardt, and F. Klowon. *Foundations of Fuzzy Systems*. John Wiley & Sons, Inc., New York, NY, USA, 1994. ISBN 047194243X.

Dale A. Lambert. A blueprint for higher-level fusion systems. *Information Fusion*, 10(1):6–24, 2009.

Eleni C. Laskari, Konstantinos E. Parsopoulos, and Michael N. Vrahatis. Particle swarm optimization for integer programming. In *Proceedings of the 2002 Congress on Evolutionary Computation*, 2002.

Steffen L. Lauritzen and David J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, B50:157–224, 1988.

Zne-Jung Lee and Chou-Yuan Lee. A hybrid search algorithm with heuristics for resource allocation problem. *Information Sciences*, 173(1-3):155–167, 2005.

Zne Jung Lee and Wen Li Lee. A hybrid search algorithm of ant colony optimization and genetic algorithm applied to weapon-target assignment problems. In Jiming Liu, Yiu-Ming Cheung, and Hujun Yin, editors, *Proceedings of the 4th International Conference on Intelligent Data Engineering and*

Automated Learning, volume 2690 of *Lecture Notes in Computer Science*, pages 278–285. Springer, 2003.

Zne-Jung Lee, Chou-Yuan Lee, and Shun-Feng Su. Parallel ant colonies with heuristics applied to weapon-target assignment problems. In *Proceedings of the 7th Conference on Artificial Intelligence and Applications*, 2002a.

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, 2002b.

Zne Jung Lee, Shun Feng Su, and Chou Yuan Lee. A genetic algorithm with domain knowledge for weapon-target assignment problems. *Journal of the Chinese Institute of Engineers*, 25(3):287–295, 2002c.

Yawei Liang. A fuzzy knowledge based system in situation and threat assessment. *Journal of Systems Science & Information*, 4(4):791–802, Dec 2006.

Yawei Liang. An approximate reasoning model for situation and threat assessment. In *Proceedings of the 4th International Conference on Fuzzy Systems and Knowledge Discovery*, 2007.

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*, 2002a.

Michael J. Liebhaber and Bela Feher. Surface warfare threat assessment: Requirements definition. Technical report, SSC San Diego, 2002b.

Michael J. Liebhaber and C. A. P. Smith. Naval air defense threat assessment: Cognitive factors and model. In *Command and Control Research and Technology Symposium*, 2000.

Michael J. Liebhaber, D. A. Kobus, and Bela Feher. Studies of U.S. navy air defense threat assessment: Cues, information order, and impact of conflicting data. Technical report, SSC San Diego, 2002.

Martin E. Liggins, David L. Hall, and James Llinas, editors. *Handbook of multisensor data fusion*. CRC Press, 2nd edition, 2009.

Eric G. Little and Galina L. Rogova. An ontological analysis of threat and vulnerability. In *Proceedings of the 9th International Conference on Information Fusion*, 2006.

James Llinas. *Handbook of multisensor data fusion*, chapter Assessing the performance of multisensor fusion processes, pages 655–675. CRC Press, 2nd edition, 2009.

Stuart P. Lloyd and Hans S. Witsenhausen. Weapon allocation is NP-complete. In *Proceedings of the 1986 Summer Conference on Simulation*, 1986.

Carl G. Looney and Lily R. Liang. Cognitive situation and threat assessments of ground battlespaces. *Information Fusion*, 4:297–308, 2003.

Francesco Maffioli and Giulia Galbiati. Approximability of hard combinatorial optimization problems: an introduction. *Annals of Operations Research*, 96: 221–236, 2000.

Wiliiam P. Malcolm. On the character and complexity of certain defensive resource allocation problems. Technical Report DSTO-TR-1570, DSTO - Weapons Systems Division, 2004.

Ebrahim H. Mamdani. Application of fuzzy logic to approximate reasoning using linguistic synthesis. In *Proceedings of the Sixth International Symposium on Multiple-valued Logic*, pages 196–202, Los Alamitos, CA, USA, 1976. IEEE Computer Society Press.

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

MathWorks. *Fuzzy Logic Toolbox 2: User's Guide*, 2010.

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

William A. Metler and Fred L. Preston. A suite of weapon assignment algorithms for a SDI mid-course battle manager. Technical report, Naval Research Laboratory, 1990.

Jeffrey G. Morrison, Richard T. Kelly, Ronald A. Moore, and Susan G. Hutchins. Tactical decision making under stress (TADMUS) - decision support system. In *Proceedings of the 1997 IRIS National Symposium on Sensor and Data Fusion*, 1997.

Jeffrey G. Morrison, Richard T. Kelly, Ronald A. Moore, and Susan G. Hutchins. Implications of decision making research for decision support and displays. In J. A. Cannon-Bowers and E. Salas, editors, *Decision Making Under Stress: Implications for Training and Simulation*. American Psychological Association, 1998.

Sandeep Mulgund, Karen Harper, and Kalmanje Krishnakumar. Air combat tactics optimization using stochastic genetic algorithms. In *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, 1998.

Robert A. Murphey. Target-based weapon target assigment problems. In Panos M. Pardalos and Leonidas S. Pitsoulis, editors, *Nonlinear assignment problems: algorithms and applications*, pages 39–53. Kluwer Academic Publishers, 2000.

John Naisbitt. *Megatrends: Ten New Directions Transforming Our Lives*. Warner Books, 1982.

Richard E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, 2003.

Filippo Neri. *Introduction to Electronic Defense Systems*. Artech House, 1991.

X.Thong Nguyen. Threat assessment in tactical airborne environments. In *Proceedings of the Fifth International Conference on Information Fusion*, 2002.

Lars Niklasson, Maria Riveiro, Fredrik Johansson, Anders Dahlbom, Göran Falkman, Tom Ziemke, Christoffer Brax, Tomas Kronhamn, Martin Smedberg, Håkan Warston, and Per Gustavsson. Extending the scope of situation analysis. In *Proceedings of the 11th International Conference on Information Fusion*, 2008.

Maria Nilsson. *Mind the Gap: Human Decision Making and Information Fusion*. Licentiate thesis, Örebro University, Örebro, Sweden, 2008.

Nickens Okello and Gavin Thoms. Threat assessment using Bayesian networks. In *Proceedings of the Sixth International Conference on Information Fusion*, 2003.

Martin Oxenham. Enhancing situation awareness for air defence via automated threat analysis. In *Proceedings of the Sixth International Conference on Information Fusion*, volume 2, pages 1086–1093, 2003.

Feng Pan, Guanghui Wang, and Yang Liu. A multi-objective-based non-stationary UAV assignment model for constraints handling using PSO. In *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pages 459–466, 2009.

Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 1998.

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.

Angela M. Pawlowski, Sergio Gigli, and Frank J. Vetesi. Situation and threat refinement approach for combating the asymmetric threat. In *MSS NSSDF Conference*, San Diego, CA, 2002.

Judea Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29:241–288, 1986.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

John Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

Ronald L. Rardin and Reha Uzsoy. Experimental evaluation of heuristic optimization algorithms: A tutorial. *Journal of Heuristics*, 7:261–304, 2001.

Chet Richards. *Certain to win: The strategy of John Boyd applied to business*. Xlibris Corporation, 2004.

Maria Riveiro, Fredrik Johansson, Göran Falkman, and Tom Ziemke. *Tenth Scandinavian Conference on Artificial Intelligence*, chapter Supporting Maritime Situation Awareness Using Self Organizing Maps and Gaussian Mixture Models, pages 84–91. IOS Press, 2008.

Jaco N. Roux and Jan H. van Vuuren. Threat evaluation and weapon assignment decision support: A review of the state of the art. *ORiON*, 23:151–186, 2007.

Jaco N. Roux and Jan H. van Vuuren. Real-time threat evaluation in a ground based air defence environment. *ORiON*, 24(1):75–101, 2008.

Jean Roy. From data fusion to situation analysis. In *Proceedings of the Fourth International Conference on Information Fusion*, 2001.

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.

Agnes Runqvist. Threat evaluation. An application for air surveillance systems. Master's thesis, Uppsala University, 2004.

Dan Schrage and Paul G. Gonsalves. Sensor scheduling using ant colony optimization. In *Proceedings of the 6th International Conference on Information Fusion*, 2003.

Johan Schubert. Evidential force aggregation. In *Proceedings of the 6th International Conference on Information Fusion*, 2003.

Tod M. Schuck, J. Bockett Hunter, and Daniel D. Wilson. *Handbook of multisensor data fusion*, chapter Developing Information Fusion Methods for Combat Identification, pages 773–812. CRC Press, 2009.

René Séguin, Jean-Yves Potvin, Michel Gendreau, Teodor G. Crainic, and Patrice Marcotte. Real-time decision problems: An operational research perspective. *The Journal of the Operational Research Society*, 48(2):162–174, 1997.

Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

Edward H. Shortliffe and Bruce G. Buchanan. A model of inexact reasoning in medicine. *Mathematical Biosciences*, 23:351–379, 1975.

C.A.P. Smith, Joan Johnston, and Carol Paris. Decision support for air warfare: Detection of deceptive threats. *Group Decision and Negotiation*, 13(2):129–148, 2004.

Mark St. John, D. I. Manes, H. S. Smallman, Bela Feher, and J. G. Morrison. An intelligent threat assessment tool for decluttering naval air defense displays. Technical report, SSC San Diego, 2004a.

Mark St. John, D. I Manes, H. S. Smallman, Bela A. Feher, and J. G. Morrison. Heuristic automation for decluttering tactical displays. In *Proceedings of the Human Factors and Ergonomics Society 48th Annual Meeting*, 2004b.

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.

Alan N. Steinberg. An approach to threat assessment. In *Proceedings of the 8th International Conference on Information Fusion*, 2005.

Alan N. Steinberg. *Handbook of multisensor data fusion*, chapter Foundations of Situation and Threat Assessment, pages 437–501. CRC Press, 2009.

Balram Suman and Prabhat Kumar. A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the Operational Research Society*, 57:1143–1160, 2006.

Robert Suzić. *Stochastic Multi-Agent Plan Recognition, Knowledge Representation and Simulations for Efficient Decision Making*. PhD thesis, Royal Institute of Technology, 2006.

Pontus Svenson and Hedvig Sidenbladh. Determining possible avenues of approach using ants. In *Proceedings of the 6th International Conference on Information Fusion*, pages 1110–1117, 2003.

Peng Teng, Huigang Lv, Jun Huang, and Liang 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.

Jeffrey Tweedale and X. Thong Nguyen. An architecture for modelling situation and threat assessment. In *Proceedings of SimTecT 2003*, 2003.

US Congress, Office of Technology Assessment. *Who Goes There: Friend or Foe?* U.S. Government Printing Office, 1993.

Joeri van Laere. Challenges for IF performance evaluation in practice. In *Proceeedings of the 12th International Conference on Information Fusion*, 2009.

Kalyan Veeramachaneni, Lisa Osadciw, and Pramod Varshney. An evolutionary algorithm based approach for dynamic thresholding in multimodal biometrics. In *Proceedings of the 2004 International Conference on Biometric Authentication*, 2004.

Jaco Vermaak, Simon J. Godsill, and Patrick Perez. Monte Carlo filtering for multi target tracking and data association. *IEEE Transactions on Aerospace and Electronic Systems*, 41(1):309–332, 2005.

Kai Virtanen, Raimo P. Hämäläinen, and Ville Mattila. Team optimal signaling strategies in air combat. *IEEE Transactions on Systems, Man and Cybernetics*, 36(4):643–660, 2006.

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

Mattias Wahde. *Biologically inspired optimization methods*. WIT Press, 2008.

Edward L. Waltz and James Llinas. *Multisensor Data Fusion*. Artech House, 1990.

Kenneth P. Werrell. *Archie to SAM: A Short Operational History of Ground-Based Air Defense*. Air University Press, Maxwell Air Force Base, Alabama, USA, 2005.

Wayne L. Winston. *Operations Research: Applications and Algorithms*. Wadsworth Publishing Company, 1997.

Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, CA, 2nd edition, 2005.

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

Chun Yang, Erik Blasch, and Ivan Kadar. Optimality self online monitoring (OSOM) for performance evaluation and adaptive sensor fusion. In *Proceedings of the 11th International Conference on Information Fusion*, 2008.

Lotfi A. Zadeh. The role of fuzzy logic in the management of uncertainty in expert systems. *Fuzzy Sets and Systems*, 11:197–198, 1983.

Xiangping Zeng, Yunlong Zhu, Lin Nan, Kunyuan Hu, Ben Niu, and Xiaoxian He. Solving weapon-target assignment problem using discrete particle swarm optimization. In *Proceedings of the 6th World Congress on Intelligent Control and Automation*, 2006.

Nevin Lianwen Zhang and David Poole. Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, 5: 301–328, 1996.

Publications *in the series*
Örebro Studies in Technology

1. Bergsten, Pontus (2001) *Observers and Controllers for Takagi – Sugeno Fuzzy Systems*. Doctoral Dissertation.
2. Iliev, Boyko (2002) *Minimum-time Sliding Mode Control of Robot Manipulators*. Licentiate Thesis.
3. Spännar, Jan (2002) *Grey box modelling for temperature estimation*. Licentiate Thesis.
4. Persson, Martin (2002) *A simulation environment for visual servoing*. Licentiate Thesis.
5. Boustedt, Katarina (2002) *Flip Chip for High Volume and Low Cost – Materials and Production Technology*. Licentiate Thesis.
6. Biel, Lena (2002) *Modeling of Perceptual Systems – A Sensor Fusion Model with Active Perception*. Licentiate Thesis.
7. Otterskog, Magnus (2002) *Produktionstest av mobiltelefonantenner i mod-växlande kammare*. Licentiate Thesis.
8. Tolt, Gustav (2003) *Fuzzy-Similarity-Based Low-level Image Processing*. Licentiate Thesis.
9. Loutfi, Amy (2003) *Communicating Perceptions: Grounding Symbols to Artificial Olfactory Signals*. Licentiate Thesis.
10. Iliev, Boyko (2004) *Minimum-time Sliding Mode Control of Robot Manipulators*. Doctoral Dissertation.
11. Pettersson, Ola (2004) *Model-Free Execution Monitoring in Behavior-Based Mobile Robotics*. Doctoral Dissertation.
12. Överstam, Henrik (2004) *The Interdependence of Plastic Behaviour and Final Properties of Steel Wire, Analysed by the Finite Element Metod*. Doctoral Dissertation.
13. Jennergren, Lars (2004) *Flexible Assembly of Ready-to-eat Meals*. Licentiate Thesis.
14. Jun, Li (2004) *Towards Online Learning of Reactive Behaviors in Mobile Robotics*. Licentiate Thesis.
15. Lindquist, Malin (2004) *Electronic Tongue for Water Quality Assessment*. Licentiate Thesis.
16. Wasik, Zbigniew (2005) *A Behavior-Based Control System for Mobile Manipulation*. Doctoral Dissertation.

17. Berntsson, Tomas (2005) *Replacement of Lead Baths with Environment Friendly Alternative Heat Treatment Processes in Steel Wire Production*. Licentiate Thesis.

18. Tolt, Gustav (2005) *Fuzzy Similarity-based Image Processing*. Doctoral Dissertation.

19. Munkevik, Per (2005) *"Artificial sensory evaluation – appearance-based analysis of ready meals"*. Licentiate Thesis.

20. Buschka, Pär (2005) *An Investigation of Hybrid Maps for Mobile Robots*. Doctoral Dissertation.

21. Loutfi, Amy (2006) *Odour Recognition using Electronic Noses in Robotic and Intelligent Systems*. Doctoral Dissertation.

22. Gillström, Peter (2006) *Alternatives to Pickling; Preparation of Carbon and Low Alloyed Steel Wire Rod*. Doctoral Dissertation.

23. Li, Jun (2006) *Learning Reactive Behaviors with Constructive Neural Networks in Mobile Robotics*. Doctoral Dissertation.

24. Otterskog, Magnus (2006) *Propagation Environment Modeling Using Scattered Field Chamber*. Doctoral Dissertation.

25. Lindquist, Malin (2007) *Electronic Tongue for Water Quality Assessment*. Doctoral Dissertation.

26. Cielniak, Grzegorz (2007) *People Tracking by Mobile Robots using Thermal and Colour Vision*. Doctoral Dissertation.

27. Boustedt, Katarina (2007) *Flip Chip for High Frequency Applications – Materials Aspects*. Doctoral Dissertation.

28. Soron, Mikael (2007) *Robot System for Flexible 3D Friction Stir Welding*. Doctoral Dissertation.

29. Larsson, Sören (2008) *An industrial robot as carrier of a laser profile scanner. – Motion control, data capturing and path planning*. Doctoral Dissertation.

30. Persson, Martin (2008) *Semantic Mapping Using Virtual Sensors and Fusion of Aerial Images with Sensor Data from a Ground Vehicle*. Doctoral Dissertation.

31. Andreasson, Henrik (2008) *Local Visual Feature based Localisation and Mapping by Mobile Robots*. Doctoral Dissertation.

32. Bouguerra, Abdelbaki (2008) *Robust Execution of Robot Task-Plans: A Knowledge-based Approach*. Doctoral Dissertation.

33. Lundh, Robert (2009) *Robots that Help Each Other: Self-Configuration of Distributed Robot Systems*. Doctoral Dissertation.

34. Skoglund, Alexander (2009) *Programming by Demonstration of Robot Manipulators*. Doctoral Dissertation.

35. Ranjbar, Parivash (2009) *Sensing the Environment: Development of Monitoring Aids for Persons with Profound Deafness or Deafblindness*. Doctoral Dissertation.

36. Magnusson, Martin (2009) *The Three-Dimensional Normal-Distributions Transform – an Efficient Representation for Registration, Surface Analysis, and Loop Detection.* Doctoral Dissertation.

37. Rahayem, Mohamed (2010) *Segmentation and fitting for Geometric Reverse Engineering. Processing data captured by a laser profile scanner mounted on an industrial robot.* Doctoral Dissertation.

38. Karlsson, Alexander (2010) *Evaluating Credal Set Theory as a Belief Framework in High-Level Information Fusion for Automated Decision-Making*. Doctoral Dissertation.

39. LeBlanc, Kevin (2010) *Cooperative Anchoring – Sharing Information About Objects in Multi-Robot Systems*. Doctoral Dissertation.

40. Johansson, Fredrik (2010) *Evaluating the Performance of TEWA Systems*. Doctoral Dissertation.