Implementation and Analysis of Decentralized Methods for Weapon-Target Assignment

Implementation, demonstration, analysis, and recommended future work for the methods described in [1]

Landon Shumway
Department of Computer and Electrical Engineering
Brigham Young University
Provo, Utah
landon16@byu.edu

***Abstract*—Decentralized weapon-target assignment (WTA) is an optimization problem in which a combination of agent/target assignment pairs is selected to maximize the destruction of the target set. As a decentralized, cooperative method, it is an important problem in multi-agent autonomous control. This paper summarizes and explores previous implementations, specifically in the proposal of three novel cost function that incite differing behaviors in comparison to traditional cost functions. Example scenarios with low weapon counts are explored that highlight differences in behavior induced by these cost functions, which translate to behavior in higher-count scenarios. Future work is also outlined and recommended.**

***Index Terms*—multiagent systems, task allocation, decentralized optimization**

I. Introduction

A. Motivation of Problem

Weapon-target assignment (WTA) is an optimization problem in offensive combat scenarios involving swarms of autonomous agents. Given a set of offensive agents (weapons) and defensive targets, the optimal combination of weapon-target pairs is desired. What is "optimal", however, can differ greatly depending on the combat scenario and mission goals, and this definition varies even more as the weapon-to-target ratio changes, heterogeneous weapons are introduced, and different target priorities are defined. When posed as an optimization problem, what is "optimal" can be defined using a given cost function. In [1], multiple cost functions are given to demonstrate their influence on agent behavior and inter-agent cooperation.

This paper focuses on summarizing and implementing each cost function in [1]. In addition to defining weapon behavior, these cost functions are decentralized in the sense that each agent maintains an estimate of the current weapon-target assignment and selects a target that minimizes the given cost function using that estimate. Targets are selected using an auction algorithm in which each agent selects a target and broadcasts its selection to other agents within its communication range, which in turn rebroadcast the information along with their own selection in a daisy-chained communication network. This decentralized approach removes both the need for constant communication with a central entity and the dependence on any one agent to yield a solution, allowing the algorithm to function despite attrition to offensive agents.

B. Background Literature

Because the weapon-target assignement problem has existed in the literature for over 50 years, there are many unique approaches to solving for the most optimal solution in real time. The WTA problem was first explored in [2] in 1958. The proposed solution approximates the nonlinear aspects of the WTA problem in order to yield a solution using linear programming techniques. [3] poses the WTA problem as an integer programming problem and seeks to a find the maximum number of weapons/targets for which the optimal solution can be reasonably solved in real time. Both [4] and [5] leverage game theory to formulate a decentralized solution in which each weapon optimizes its own utility function.

Additional contributions to the literature have pulled from other generic optimization methods to solve the WTA problem. [6] proposes a problem-independent algorithm that leverages simulated annealing, which was first described in [7], to solve a generic task assignment optimization problem. This provides significant speed-up over serial algorithms. [8] embeds ant colony optimization into a genetic algorithm to mitigate the effects of local minima in WTA optimization.

Furthermore, several methods seek to tackle the the WTA problem using distributed or hybrid methods over wholly centralized solutions. [9] utilizes a centralized approach to broadcast initial assignments and leverages auction/negotiation methods to settle discrepancies. [10] divides the assignment problem into sub-problems by dividing weapons into coalitions and performing assignment algorithms within those coalitions.

Although [1] builds off of many of these methods, the research described therein, which is the focus of this paper, differs from other implementations in that each target is assigned a desired kill probability $Pk_{\text{des},j}$. When a given assignment of weapons to targets achieves that desired kill probability for a given target, there is no further incentive for additional agents to engage the target. This becomes especially beneficial in low weapon-to-target ratio scenarios because it prevents over-assignment to one particular target, even if the target is of high priority. Furthermore, the methods are implemented in such a way that dynamic changes to the scenario, such as the attrition of a weapon or the successful destruction of a target, allow for real-time adjustments to the current assignment. As such, it is assumed that each agent has a perfect knowledge of which weapons/targets have been destroyed and which are still active.

Since the publication of [1], many works have cited and built on its methods. [11], [12] focus on target tracking rather than the WTA problem, but they leverage a similar decentralized approach to facilitate coordination between multiple agents without a centralized infrastructure. [13], [14] build on the techniques in [1] while leveraging reinforcement learning to aid in decentralized assignment algorithms to solve the WTA problem. Although these methods move beyond the complexity described in [1], its methods are fundamental to their implementation and as such are explored in detail in this paper.

II. Methods

A. Model

For the simulation, targets are modeled as point masses on the ground at a given position (using a North-East coordinate system). Each weapon is modeled as a 2DOF system in which velocity and altitude are assumed to be constant, but position in the NE-plane, heading, and angular rate can change. Once a target is selected, the commanded heading for agent i is:

$$\psi_{i,com} = \arctan\left(\frac{e_j - e_i}{n_j - n_i}\right) \tag{1}$$

where (n_i, e_i) describes the current location of the weapon and (n_j, e_j) describes the current location of the target.

The control law for each agent is:

$$\dot{\psi} = k_\psi(\psi_{i,com} - \psi_i) \tag{2}$$

For simulation examples in this paper, $k_\psi = 3$, and $\dot{\psi}$ is saturated so that $|\dot{\psi}| < 1.265\, rad/s$.

Although altitude is constant, vertical dynamics are used to define when a target is in range of a weapon. It is assumed that weapons can vary their descent rate ($\dot{d}_i$) instantaneously as follows:

$$\dot{d}_i = \frac{v_h}{\sqrt{(n_i - n_j)^2 + (e_i - e_j)^2}} d_i \tag{3}$$

where v_h is the horizontal velocity of the agent in the North-East plane and d_i is the "down" coordinate (negative altitude) of the agent. For all simulations in this paper, $v_h = 50.0$ m/s.

A maximum glide ratio γ_{max} is imposed on each agent such that:

$$\frac{v_h}{\dot{d}_i} \leq \gamma_{\text{max}} \tag{4}$$

If this condition is not met, the target is considered out of range of the agent and is not considered for assignment. For all simulations, $\gamma_{\text{max}} = 6.0$.

B. Kill Probability

One necessary assumption is that all weapons know from the start of the simulation how many targets there are and where they are located. Additionally, the desired kill probability of each target j, denoted as $Pk_{\text{des},j}$ (where $0 \leq Pk_{\text{des},j} \leq 1$), is globally known for each target. Given a set of N weapons pursuing target j, its kill probability can be calculated as:

$$Pk_{\Sigma,j} = 1 - \prod_{i=1,q_i=j}^{N} (1 - Pk_{i,j} + Pk_{i,j}Pa_{i,j}) \tag{5}$$

where q_i is the assignment of agent i and $Pa_{i,j}$ is the probability that agent i gets attrited on its path to target j. $Pk_{i,j}$ is the weapon effectiveness of the weapon/target pair (i, j), which is defined as the kill probability that target j will be eliminated if agent i collides with it. In the simulation examples, upon collision it is randomly determined if the target is destroyed based on the probability represented by its weapon effectiveness. Note that in all simulation examples in this paper, the weapon effectiveness of agent i is constant regardless of the target it pursues.

C. Engagement Simulation

At the start of the simulation, is is assumed that all weapons have knowledge of the weapon effectiveness for each agent, the desired kill probability and locations for each target, and the survival state of every agent and target at any given time. Furthermore, each agent maintains a (potentially imperfect) estimate of the current assignment Q of weapons to targets, the attrition probability for each weapon, and the kill probabilities for each target. This estimate is updated during communication, whose model is described in a following section.

When a weapon reaches its target within a defined buffer range, the weapon is destroyed, and the target is destroyed with a probability equal to the weapon effectiveness of the agent. For the simulation examples in this paper, the collision buffer range used is 5 meters. If the target is successfully destroyed, other agents that are pursuing that target will change their assignment to another target within range, since the desired kill probability of the destroyed target gets set to zero. In contrast, if a collision does not destroy a target, weapons pursuing lower priority targets may switch to that target since its current kill probability decreases.

D. Attrition

The following path-dependent model is used to calculate the probability that agent i is attrited along its path to collide with target j:

$$Pa_{i,j} = 1 - (1 - Pa)^{\ell_{int}} \tag{6}$$

where

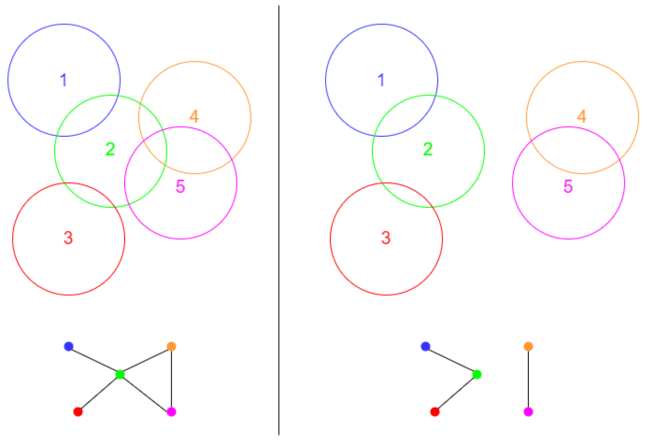


Fig. 1: Example of a connected network (left) and a disconnected network (right).

$$\ell_{int} = \ell_{ij}/\ell_a \tag{7}$$

and

$$\ell_{i,j} = \sqrt{(n_j - n_i)^2 + (e_j - e_i)^2} \tag{8}$$

The variable d_a is a user-defined number of subintervals to divide $d_{i,j}$, and it is set to 100 for the simulation examples to follow. Additionally, a value of $Pa = 0.004\%/\text{m}$ is used. Note that even with homogeneous agents, $Pa_{i,j}$ introduces heterogeneity into the agent set as a path-dependent value.

E. Communication

Agents select targets in a turn-based auction algorithm, with agents that have a higher kill probability going first. After an agent selects a target, it broadcasts its decision to other agents within its communication range, which in turn rebroadcast that information to agents in their respective communication range. It is assumed that the communication rate is much quicker than the rate needed for decision making, which in turn is assumed to be much faster than the timescale within which the weapons approach the target set.

Consequently, all agents in the daisy-chained network have an accurate, updated knowledge of the current assignment Q of connected agents and their targets. That estimate can be used by each agent to estimate the current attrition probability of each agent and the current kill probability of each target. However, if the graph becomes disconnected, agents that separate from the network will not receive updated assignment information and over time will not maintain an accurate estimate of the current assignment unless they return within range of an agent that is part of the connected graph. Figure 1 shows an example of this, where agents 4 and 5 become disconnected, resulting in outdated estimates of the assignments between agents that have no path to each other on the communication graph. As such, limited communication may significantly hinder performances when the communication graph becomes disconnected.

F. Optimization Algorithms

Although [1] implements two methods for solving optimization problems (greedy search and simulated annealing), for the scope of this project this paper focuses only on greedy search. During each decision round, each individual agent uses its current estimate of the current assignment Q:

$$Q = \{(i,j) | i \in I, j \in J\} \tag{9}$$

where $I = \{1, ..., N\}$ and $J \subseteq \{1, ..., M\}$. Note that under this notation, all N weapons are assigned to a target, while not all M targets may be pursued.

Using its estimate of Q, each agent selects the target j that minimizes some cost function the most:

$$q_i = \arg\min_{j \in J} C(Q_j) \tag{10}$$

where Q_j is the resulting global assignment set if weapon i selects target j, and $C(Q_j)$ is its associated cost. Although greedy search may not always find the optimal solution, it consistently finds a local minimum, which is sufficient for the scope of this project.

Note that since each agent uses its own estimate of the assignment and executes its own greedy search, this method represents a fully decentralized approach to the WTA problem.

G. Cost Functions

1) Sufficiency Threshold Cost Function: The main contribution of [1] is the proposal of novel cost functions to induce different behaviors among agents in assigning themselves to targets. Typical WTA algorithms use the following traditional cost function:

$$C_{\text{T}}(Q) = \sum_{j=1}^{M} (1 - Pk_{\Sigma,j}) Pk_{\text{des},j} \tag{11}$$

This equation represents the summation of the cost induced by each agent, and it starts high and decreases as the optimization continues. However, it does not address any change to the cost function when a target's desired kill probability has been exceeded by its current kill probability according to the agents assigned to it ($Pk_{\Sigma,j} > Pk_{\text{des},j}$). The following novel cost function, denoted in [1] as the "sufficiency threshold" cost function, disincentivizes an agent from engaging a target whose desired kill probability has already been reached by providing no reduction in cost:

$$C_{\text{ST}}(Q) = \sum_{j=1}^{M} \begin{cases} 0 & Pk_{\Sigma,j} > Pk_{\text{des},j} \\ \frac{Pk_{\text{des},j} - Pk_{\Sigma,j}}{(1 - Pk_{\text{des},j})^{\alpha}} & Pk_{\Sigma,j} \leq Pk_{\text{des},j} \end{cases} \tag{12}$$

Note that the term in the denominator essentially defines the value/priority of a target as a nonlinear function of $Pk_{\text{des},j}$, and α can be considered as a tuning parameter in that an increase in α puts more weight on the value of $Pk_{\text{des},j}$, prioritizing higher-value targets more. The examples to follow use $\alpha = 1$. Convergence properties and an in-depth theoretical comparison between the sufficiency threshold cost function and the traditional cost function can be viewed in [1].

2) Enforced Tiering Cost Function: Another mechanism to represent target priority is to sort targets into priority tiers and encourage agents to reach the desired kill probability for all targets in a higher priority tier before engaging targets in a lower priority tier. There are many ways to define these tiers, including desired kill probability, geographic region, defense capabilities, etc.

Let T be one of τ tiers ($T \in 1, ..., \tau$), where a lower value for T represents a higher priority tier. The "enforced tiering" cost function can be defined as:

$$C_{\text{ET}}(Q) = \sum_{T=1}^{\tau} \begin{cases} C_{\text{ST}}(Q_T) & (Pk_{\Sigma,j} \geq Pk_{\text{des},j} \forall j \in t, \forall t < T) \\ P_T & \text{else} \end{cases} \tag{13}$$

Note that within each tiered subset of targets, the sufficiency threshold cost function is utilized according to the subset $Q_T \subseteq Q$ that contains the targets in tier T. If targets in a higher tier than T have not all reached their desired kill probability values, a penalty P_T is applied for assigning an agent to a target in T, and [1] shows that the value of P_T must be greater than $C_{\text{ST,max}}$ in order for agents to properly assign the last target in tier $T-1$. It should also be noted that one trade-off of enforced tiering is that local minima are more prevalent.

3) Completion Cost Function: The last cost function proposed in [1] is especially tailored for a set of heterogeneous agents, particularly in low weapon-to-target ratio scenarios. In such cases, the desired behavior is often to meet the desired kill probability for as many targets as possible. To achieve this behavior, the following "completion" cost function is proposed:

$$C_{\text{C},i}(Q_j) = \frac{\ln(1 - Pk_{\text{des},j}) - \ln(1 - Pk_{\Sigma,j})}{\ln(1 - Pk_{i,j} + Pk_{i,j}Pa_{i,j})} \tag{14}$$

where Q_j denotes the assignment Q where agent i has selected target j. The full derivation of this cost function can be read in [1], but a simplified explanation is that agent i calculates how many additional weapons of its own effectiveness would be needed to reach $Pk_{\text{des},j}$ for target j. This calculation is done for all targets, and the target that requires the least amount of additional weapons is selected. It is important to note that this cost function differs from the previous cost functions in that the comparison between the cost for a given assignment Q by weapon i and the cost for the same Q by agent k is not possible if i and k have different $Pk_{i,j}$ and/or $Pa_{i,j}$ values. This makes the cost function unique for each agent when compared with other agents of differing effectiveness values.

It is also important to note that certain edge cases must be addressed for successful implementation of this cost function. If $Pk_{\Sigma,j} \geq Pk_{\text{des},j}$ for target j, the cost function should yield an arbitrarily large value so that weapon i does not select it, unless the weapon is already in pursuit of target j, in which case the cost function should return a value of zero so that the agent does not repeatedly switch to other targets.

For each of these cost functions, [1] provides an in-depth convergence analysis, and for the purposes of this paper, they will not be addressed here. It is sufficient to state that these cost functions yield either the globally optimal assignment or reasonable locally optimal assignments depending on a homogeneous or heterogeneous set of weapons.

H. Engagement Geometry

For each simulation example, unless otherwise specified, targets are spawned randomly at an altitude of 0 meters in a 750×750 meter area, as are all weapons at an altitude of 500 meters. Both spawning areas are located 2500 meters from each other. This geometry is shown in Figure 2, which is taken directly from [1].

RESULTS

In [1], many results are presented for all cost functions, including high agent/target-count demonstrations (i.e. 9 targets and 27 agents) and Monte Carlo simulation results. Under the scope of this project, the results will be limited to lower-count demonstrations, and further investigation of results are encouraged to be read in [1].

Each demonstration will explore weapon behavior under a different cost function for a scenario with 6 targets and 8 weapons. Each agent has a weapon effectiveness $Pk_{i,j} = 0.7$. This is a relatively low weapon effectiveness value, which will result in more cases where an agent collides with a target without destroying it. However, this decrease in weapon effectiveness is intentional in order to demonstrate assignment switching behavior when these unsuccessful collisions occur. Out of the six targets, Target 0 and Target 1 have $Pk_{\text{des},j} = 0.9$, Target 2 and Target 3 have $Pk_{\text{des},j} = 0.8$, and Target 4 and Target 5 have $Pk_{\text{des},j} = 0.7$.

It is important to note that with the singular application of each cost function to the same scenario, it is not an accurate comparison to evaluate success between cost functions by the number of targets that have been destroyed for two reasons. First, each cost function defines its own measure of success, which is why behavior changes. Second, the success of a collision is determined based on a random function with probability weighted according to weapon effectiveness. Even though the seed for each random function was the same between runs, since different agents are colliding with different targets (depending on the cost function), the success of collisions still varies. To address this, [1] performs full Monte Carlo simulations to track results over many simulation runs. Such analysis is out of scope for this paper, whose main purpose is to demonstrate varying behavior in target assignment with different cost functions. The reader is encouraged to refer to [1] for further comparison between cost functions.

The traditional cost function is the first to undergo examination. Figure 3 in Appendix A displays the achieved kill probability for each target j ($Pk_{\Sigma,j}$) over time, which targets are assigned to which agents over time, and descriptions for

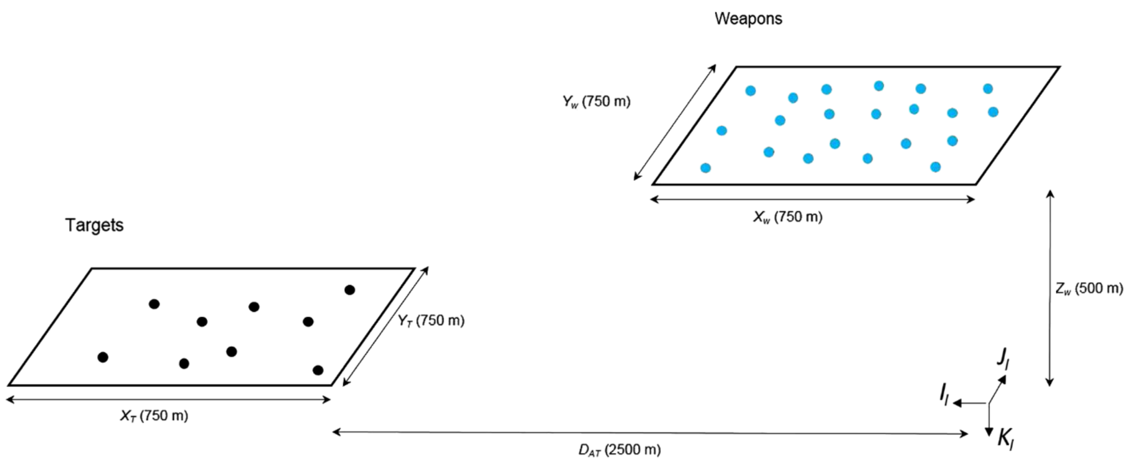


Fig. 2: Initial engagement geometry.

key events and their associated time stamps. The following summary identifies those key moments in the simulation and provides an example on how to read the graphs. Note that within the first second, assignments change as agents receive the target selections of other agents and settle into a solution. After this transient behavior, Agent 1 is pursuing Target 5. At $t = 7.1$ s, Agent 1 gets attrited. In response to the decrease in the current kill probability of Target 5, Agent 0 switches from pursuing Target 0 to pursuing Target 5. Agent 3 then switches from pursuing Target 1 to pursuing Target 0. Similar switches occur when Agent 3 gets attrited at $t = 17.0$ s and Agent 7 gets attrited at $t = 26.9$ s. At $t = 45.4$ s, Agent 5 collides with Target 5 and successfully destroys it. Similarly, at $t = 52.0$, Agent 0 collides with Target 1 and successfully destroys it. However, at $t = 56.6$ s, Agent 6 collides with Target 0 but does not destroy it. This causes Agent 2 to switch its pursuit from Target 3, which has a lower $Pk_{\text{des},j}$, to Target 0. At $t = 60.3$ s, Agent 4 collides with Target 2 without destroying it, and at $t = 64.6$ the same occurs between Agent 2 and Target 0. At this point, the simulation ends, since all agents have been destroyed.

Note that, in the case of the traditional cost function, a heavy emphasis is put on assigning agents to targets of higher priority. However, particularly before the attrition of agents, there is still a tendency to assign agents to those targets even when their $Pk_{\text{des},j}$ has already been met. In contrast to this behavior, the sufficiency threshold cost function's equivalent graphs and table are shown in Figure 4 of Appendix A. Note that behavior is similar, but weapons have less of a tendency to overload assignment to high-priority targets if those targets' $Pk_{\text{des},j}$ values have already been met, resulting in a more spread-out distribution of assignments among targets (a higher spread of assignments to Targets 2/3 and 4/5 than in the traditional cost function). This also induces more switching behavior as agents get attrited and unsuccessful collisions occur.

In comparison with the previous two cost functions, the enforced tiering cost function exhibits very different behavior in target assignment, which can be observed in Figure 5 of Appendix A. The targets are divided into three tiers according to $Pk_{\text{des},j}$, with Target 0 and Target 1 in the highest-priority tier. As such, all agents focus on those targets to properly achieve the desired kill probability for each one. Once they are destroyed, remaining agents switch their assignments to the second tier (Target 2 and Target 3), and the same occurs for the third tier upon the destruction of the targets in the second tier.

While other cost functions cause high-priority targets to be engaged first, the completion cost function encourages agents to attack low-priority targets first in an effort to reach $Pk_{\text{des},j}$ for as many targets as possible. This behavior is reflected in Figure 6of Appendix A, where targets with lower $Pk_{\text{des},j}$ values are assigned over targets with higher values that require more weapons to reach their desired kill probability values. Note, however, that as these "easier" targets are destroyed, the higher-priority targets start to get more agents assigned to them.

Future Work

Due to time constraints for this project, this paper does not go beyond the methods implemented in [1]. However, extensions to these methods are under current implementation. First, these methods are currently being integrated into a 3D simulation environment with diverse agents that are represented with more complicated dynamic models, including a 3DOF boost-glide model and a thrusted 6DOF model. The simulation also has capabilities to implement more complicated guidance laws that more accurately reflect realistic midcourse and terminal guidance phases, such as altitude hold, dynamic pressure control, augmented proportional navigation, etc. With these more complicated models and guidance laws, more sophisticated, model-specific methods for determining if a target is in range will be implemented. Finally, although estimates of weapon effectiveness and attrition probability will still be needed to run the methods described in this paper, the random attrition of agents will be replaced with defensive agents in the 3D simulation that attempt to collide with the pursuers, and target destruction will therefore be defined as a successful collision of the offensive agent with the target. This will lead to future work of deriving accurate methods to estimate weapon effectiveness and attrition probability based on these more realistic conditions, rather than simply assuming they are known values.

Conclusion

The works in [1] have been summarized, presenting three novel cost functions that cause differing behavior among agents in a decentralized solution to the WTA problem. All cost functions take into account the desired kill probability ($Pk_{\text{des},j}$) of each target. The sufficiency threshold and enforced tiering cost functions focus agents on targets with high $Pk_{\text{des},j}$ values without over-assigning weapons, while the completion cost function focuses agents on targets with low $Pk_{\text{des},j}$ in an attempt to meet those values for as many targets as possible given the number of available weapons. Each of these cost functions can be used by any agent to optimize its own target assignment with the information it has on the assignment of other agents according to a limited-radius communication model. Overall, results in this paper show that behavior varies between cost functions as expected. Further analysis of convergence properties, Monte Carlo simulations, and examples of scenarios with a higher count of weapons and targets are detailed in [1]. Future work will include implementing these methods in a 3D simulation environment with more realistic models for dynamics, attrition, and collision as well as guidance laws that more accurately reflect typical missile guidance phases.

References

[1] K. Volle, J. Rogers, and K. Brink, “Decentralized Cooperative Control Methods for the Modified Weapon–Target Assignment Problem,” Journal of Guidance, Control, and Dynamics, Jul. 2016, doi: 10.2514/1.G001752.

[2] A. S. Manne, “A Target-Assignment Problem,” Operations Research, vol. 6, no. 3, pp. 346–351, Jun. 1958, doi: 10.1287/opre.6.3.346.

[3] R. K. Ahuja, A. Kumar, K. C. Jha, and J. B. Orlin, “Exact and Heuristic Algorithms for the Weapon-Target Assignment Problem,” Operations Research, vol. 55, no. 6, pp. 1136–1146, Dec. 2007, doi: 10.1287/opre.1070.0440.

[4] G. Arslan, J. R. Marden, and J. S. Shamma, “Autonomous Vehicle-Target Assignment: A Game-Theoretical Formulation,” Journal of Dynamic Systems, Measurement, and Control, vol. 129, no. 5, pp. 584–596, Apr. 2007, doi: 10.1115/1.2766722.

[5] A. Chapman, R. A. Micillo, R. Kota, and N. Jennings, “Decentralised Dynamic Task Allocation: A Practical Game–Theoretic Approach,” presented at the The Eighth International Conference on Autonomous Agents and Multiagent Systems (AAMAS ’09) (10/05/09 - 15/05/09), May 2009, pp. 915–922. Accessed: Apr. 10, 2024. [Online]. Available: https://eprints.soton.ac.uk/267066/

[6] E. E. Witte, R. D. Chamberlain, and M. A. Franklin, “Task assignment by parallel simulated annealing,” in Proceedings., 1990 IEEE International Conference on Computer Design: VLSI in Computers and Processors, Sep. 1990, pp. 74–77. doi: 10.1109/ICCD.1990.130165.

[7] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by Simulated Annealing,” Science, vol. 220, no. 4598, pp. 671–680, May 1983, doi: 10.1126/science.220.4598.671.

[8] Z.-J. Lee and W.-L. Lee, “A Hybrid Search Algorithm of Ant Colony Optimization and Genetic Algorithm Applied to Weapon-Target Assignment Problems,” in Intelligent Data Engineering and Automated Learning, J. Liu, Y. Cheung, and H. Yin, Eds., Berlin, Heidelberg: Springer, 2003, pp. 278–285. doi: 10.1007/978-3-540-45080-1-37.

[9] M. Alighanbari and J. P. How, “Decentralized Task Assignment for Unmanned Aerial Vehicles,” in Proceedings of the 44th IEEE Conference on Decision and Control, Dec. 2005, pp. 5668–5673. doi: 10.1109/CDC.2005.1583066.

[10] O. Shehory and S. Kraus, “Methods for task allocation via agent coalition formation,” Artificial Intelligence, vol. 101, no. 1, pp. 165–200, May 1998, doi: 10.1016/S0004-3702(98)00045-9.

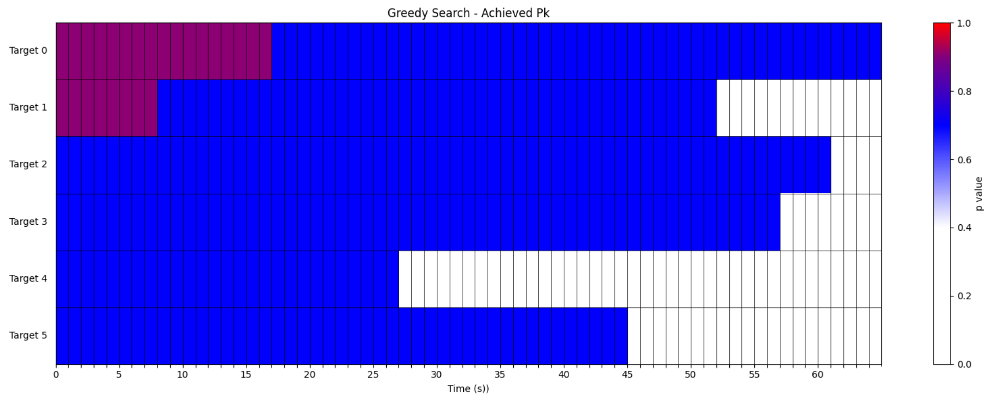
[11] J. P. Mathew and C. Nowzari, “Real-Time Distributed Multi-Robot Target Tracking via Virtual Pheromones,” in 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 2022, pp. 5833–5839. doi: 10.1109/IROS47612.2022.9981262.

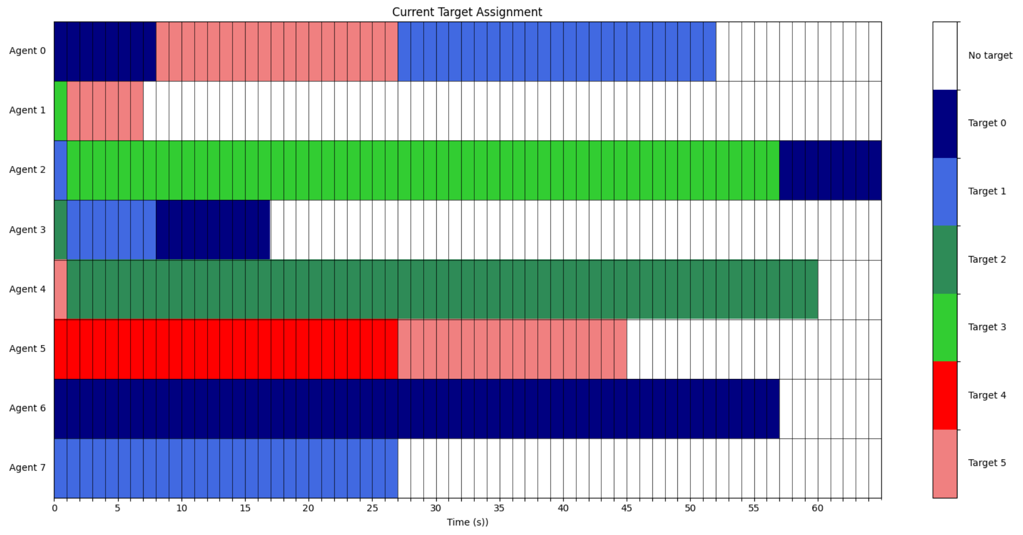
[12] J. P. Mathew and C. Nowzari, “Real-Time Distributed Infrastructure-free Searching and Target Tracking via Virtual Pheromones.” arXiv, Mar. 28, 2024. doi: 10.48550/arXiv.2311.13035.

[13] J. Guo, G. Hu, Z. Guo, and M. Zhou, “Evaluation Model, Intelligent Assignment, and Cooperative Interception in Multimissile and Multitarget Engagement,” IEEE Transactions on Aerospace and Electronic Systems, vol. 58, no. 4, pp. 3104–3115, Aug. 2022, doi: 10.1109/TAES.2022.3144111.

[14] B. Gaudet, K. Drozd, and R. Furfaro, “Deep Reinforcement Learning for Weapons to Targets Assignment in a Hypersonic strike.” arXiv, Oct. 27, 2023. doi: 10.48550/arXiv.2310.18509.

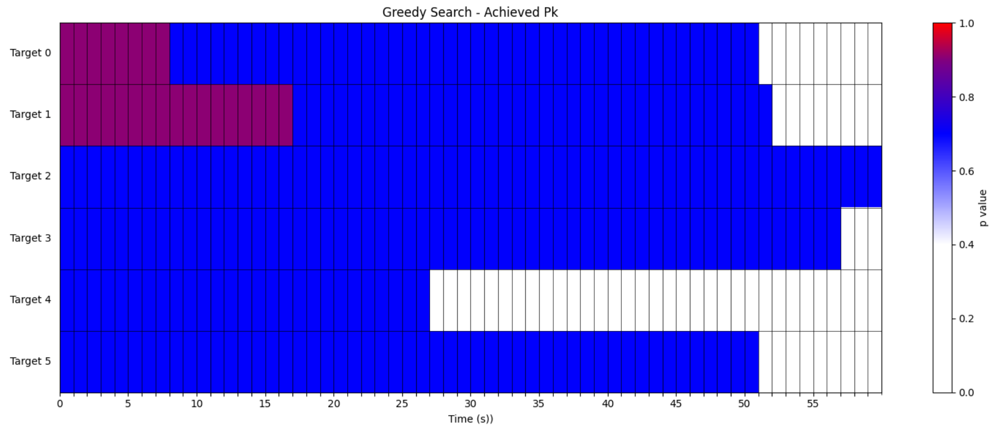
Appendix A
Simulation Tables

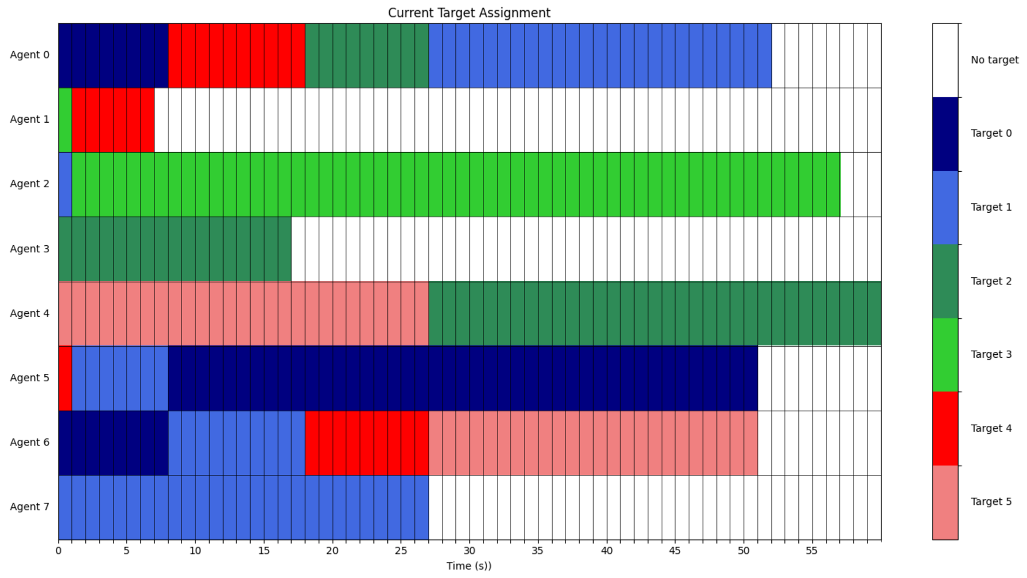




Time (s)	Event
7.1	Agent 1 gets attrited
17.1	Agent 3 gets attrited
26.9	Agent 7 gets attrited
45.4	Agent 5 collides with Target 5; Target 5 is destroyed
52.0	Agent 0 collides with Target 1; Target 1 is destroyed
56.6	Agent 6 collides with Target 0; Target 0 survives
60.3	Agent 4 collides with Target 2; Target 2 survives
64.6	Agent 2 collides with Target 0; Target 0 survives

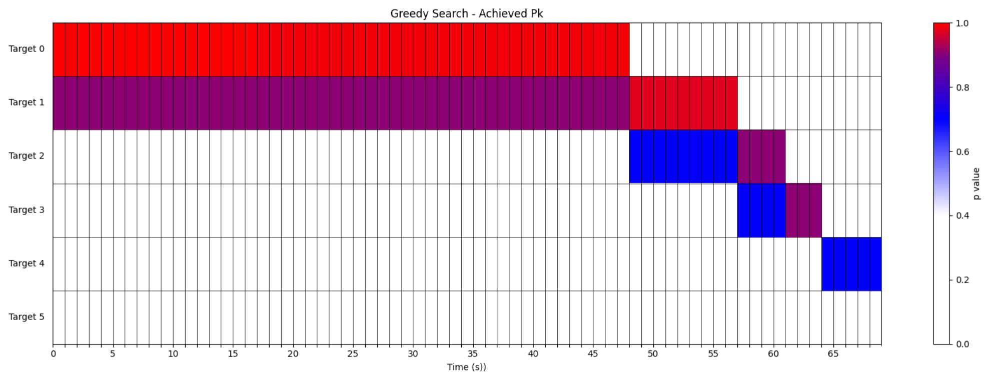
Fig. 3: Example simulation using the traditional cost function showing (top) target $Pk_{\Sigma,j}$ over time, (middle) agent assignment over time, and (bottom) key events.

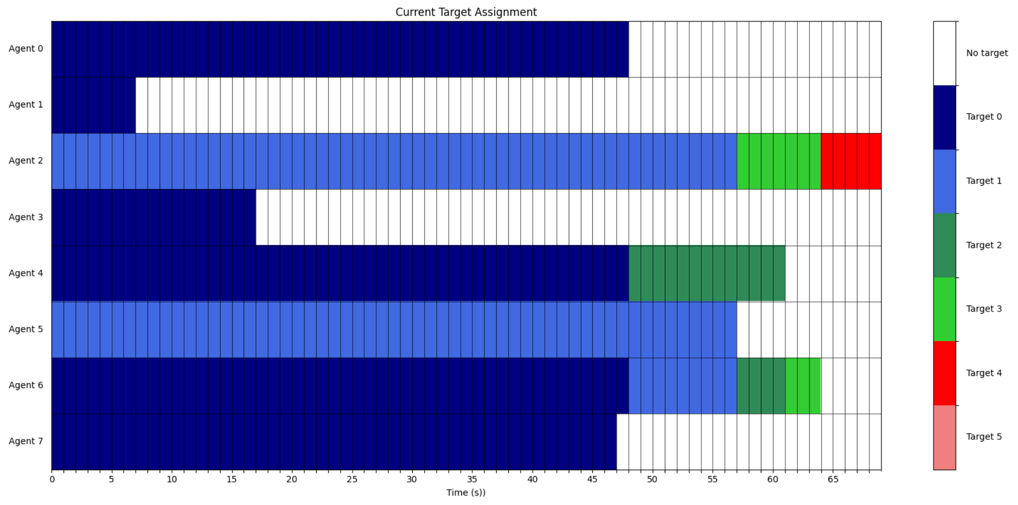




Time (s)	Event
7.1	Agent 1 gets attrited
17.1	Agent 3 gets attrited
26.9	Agent 7 gets attrited
50.8	Agent 6 collides with Target 5; Target 5 survives
51.1	Agent 5 collides with Target 0; Target 0 is destroyed
51.6	Agent 0 collides with Target 1; Target 1 is destroyed
56.9	Agent 2 collides with Target 3; Target 3 survives
63.0	Agent 4 collides with Target 2; Target 2 survives

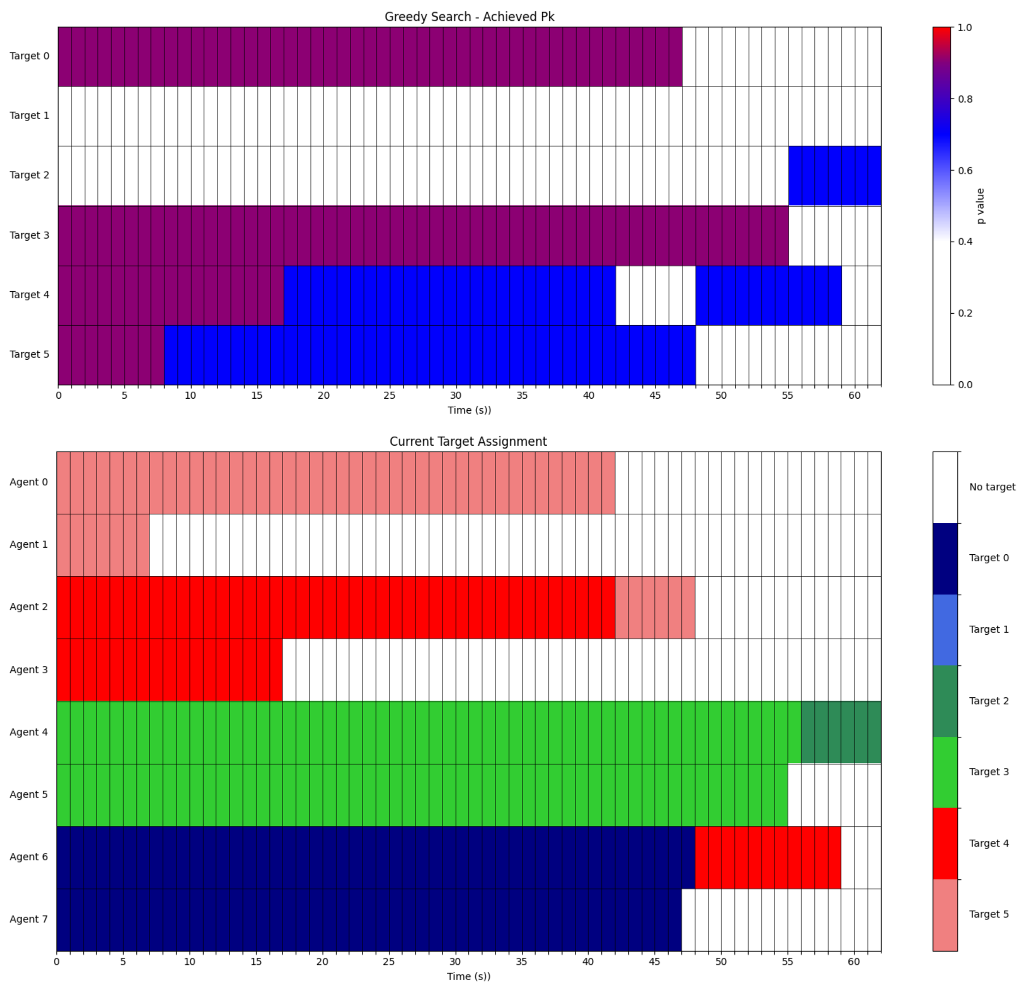
Fig. 4: Example simulation using the sufficiency threshold cost function showing (top) target $Pk_{\Sigma,j}$ over time and (bottom) agent assignment over time.





Time (s)	Event
7.1	Agent 1 gets attrited
17.0	Agent 3 gets attrited
47.4	Agent 7 collides with Target 0; Target 0 survives
47.8	Agent 0 collides with Target 0; Target 0 is destroyed
56.9	Agent 5 collides with Target 1; Target 1 is destroyed
60.6	Agent 4 collides with Target 2; Target 2 is destroyed
63.9	Agent 6 collides with Target 3; Target 3 is destroyed
69.0	Agent 2 collides with Target 4; Target 4 survives

Fig. 5: Example simulation using the enforced tiering cost function showing (top) target $Pk_{\Sigma,j}$ over time and (bottom) agent assignment over time.



Time (s)	Event
7.1	Agent 1 gets attrited
17.0	Agent 3 gets attrited
41.7	Agent 0 collides with Target 5; Target 5 survives
47.4	Agent 0 collides with Target 7; Target 0 is destroyed
47.8	Agent 2 collides with Target 5; Target 5 is destroyed
55.0	Agent 5 collides with Target 3; Target 3 is destroyed
58.9	Agent 6 collides with Target 4; Target 4 is destroyed
61.6	Agent 4 collides with Target 2; Target 2 survives

Fig. 6: Example simulation using the completion cost function showing (top) target $Pk_{\Sigma,j}$ over time and (bottom) agent assignment over time.