Decentralized Weapon-Target Assignment under Asynchronous Communications

Katherine Hendrickson*
University of Florida, Gainesville, Florida 32611

Prashant Ganesh†
University of Florida, Shalimar, Florida 32579

Kyle Volle‡
University of Florida, Shalimar, Florida 32579

Paul Buzaud§
University of Florida, Shalimar, Florida 32579

Kevin Brink¶
Air Force Research Laboratory, Eglin Air Force Base, Florida 32542

Matthew Hale‖
University of Florida, Gainesville, Florida 32611

The weapon-target assignment problem is a classic task assignment problem in combinatorial optimization, and its goal is to assign some number of workers (the weapons) to some number of tasks (the targets). Classical approaches for this problem typically use a centralized planner leading to a single point of failure and often preventing real-time re-planning as conditions change. This paper introduces a new approach for distributed, autonomous assignment planning executed by the weapons where each weapon is responsible for optimizing over distinct subsets of the decision variables. A continuous, convex relaxation of the associated cost function and constraints is introduced, and a distributed primal-dual optimization algorithm is developed that will be shown to have guaranteed bounds on its convergence rate, even with asynchronous computations and communications. This approach has several advantages in practice due to its robustness to asynchrony and resilience to time-varying scenarios, and these advantages are exhibited in experiments with simulated and physical commercial off-the-shelf ground robots as weapon surrogates that are shown to successfully compute their assignments under intermittent communications and unexpected attrition (loss) of weapons.

*Graduate Research Assistant, Department of Mechanical and Aerospace Engineering.
†Research Assistant Engineer, Department of Mechanical and Aerospace Engineering.
‡Adjunct Research Scientist, Department of Mechanical and Aerospace Engineering.
§Research Engineer, Department of Mechanical and Aerospace Engineering.
¶Senior Research Engineer, Weapon Dynamics, Guidance, and Navigation Control Branch, Munitions Directorate. Associate Fellow AIAA.
‖Assistant Professor, Department of Mechanical and Aerospace Engineering. Member AIAA.

Nomenclature

B bounded delay parameter further defined in Assumption 1

C_j^i times at which primal updates are sent by worker i to worker j

D subset of natural numbers that determines the times $k = tB, t \in D$, when all workers perform dual updates

$\mathcal{I}$ set containing the indices of all workers

k iteration count used by all workers for their primal and dual updates

K^i set of times at which worker $i \in \mathcal{I}$ performs primal updates

l number of tasks/targets

m number of workers/weapons

$\mathcal{M}$ set defined by $\{\nu \in \mathbb{R}_+^m : \|\nu\|_1 \leq \sum_{j=1}^{\ell} V_j\}$

$\mathcal{M}_i$ set defined by $\{\nu \in \mathbb{R}_+ : \nu \leq \sum_{j=1}^{\ell} V_j\}$

$Pk_{[i],j}$ probability weapon i destroys target j

$\overline{Pk}_{[i],j}$ probability weapon i arrives to and destroys target j

t natural number from the set D that determines the time $k = tB$ when all workers compute their dual updates

V_j value of target j

$\mathbf{x}^i$ worker i's copy of the primal variable $\mathbf{x}$

$\mathbf{x}_{[j]}^i$ worker i's value for the primal block j

$\hat{\mathbf{x}}_{\boldsymbol{\kappa}}$ primal component of the saddle point $(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}, \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}})$ of $L_{\boldsymbol{\kappa}}$

X constraint set defined by $[0,1]^{m\ell}$

X_i constraint set defined by $[0,1]^{\ell}$

α primal regularization parameter in $L_{\boldsymbol{\kappa}}$

δ dual regularization parameter in $L_{\boldsymbol{\kappa}}$

γ primal stepsize

$\boldsymbol{\kappa}$ pair of regularization parameters (α, δ)

$\boldsymbol{\mu}^i$ worker i's copy of the dual variable $\boldsymbol{\mu}$

μ_i^i worker i's dual entry i, which it is responsible for updating

$\hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}$ dual component of the saddle point $(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}, \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}})$ of $L_{\boldsymbol{\kappa}}$

ρ dual stepsize

$\tau_j^i(k)$ time at which worker j originally computed the primal update received by worker i at time k

I. Introduction

Since it was first posed in 1958 [1] the weapon-target assignment (WTA) problem has been a well-studied research problem in the fields of combinatorial optimization and operations research more broadly [2–5]. Given a set of

weapons of known probabilistic effectiveness and a set of targets of known value, the WTA problem seeks to assign each weapon to a target in a way that minimizes the post-engagement expected value of all surviving targets. This problem has obvious applications for military planners, but has also been used for (and is similar to) numerous other resource allocation problems such as emergency management [6] and advertising [7]. The WTA problem has been known to be NP-complete since 1986 [8], making it fertile ground for research into heuristic optimization algorithms with lower complexity [9–11]. While the general structure of the WTA problem admits many variations, such as partial-information WTA [12], multi-layer problems with targets, weapons, and counter-weapons [13], multi-task implementations that incorporate target identification, verification, and engagement [14], or sequential engagement WTA [15, 16], this paper will focus on expanding the classic problem formulation. Readers are referred to [17] for an overview of exact and heuristic approaches to the WTA problem. The original form of WTA was quasi-static in the sense that the targets and their values do not change throughout the engagement, nor do the properties of the weapons. The original solution (and many solutions thereafter) was also centralized in the sense that a single planner computed all weapon assignments.

While centralized planning may be well-suited to some scenarios, e.g., non-autonomous air-to-ground munitions, modern and future munitions have the ability to plan and act autonomously, which means that centralized planning is not required. Indeed it may fail to fully exploit the decision-making capabilities of individual weapons. Moreover, centralized pre-planning may work well in static conditions, though unexpected changes such as the attrition (loss during flight) of a weapon would typically require re-planning, which can be time-consuming due to the computational burden it imposes. Re-planning is also difficult to coordinate in a centralized way because it can be difficult to communicate with and coordinate weapons when they are already deployed and spread far apart. Algorithmically, it has been shown that in some scenarios where centralized and decentralized approaches are both feasible, the decentralized approach performs measurably better [18]. And while the WTA problem is easily specified in a centralized way, modern autonomy applications increasingly take place in unknown, unstructured, and contested environments, all of which suggest that the use of a centralized planner is either infeasible or undesirable because it creates a single point of failure and does not rapidly react to changing conditions.

In particular, so-called “fire and forget” approaches do not enable dynamically changing assignments of weapons, which does not allow weapons to change course during deployment and does not allow weapons to react to the success or failure of other weapons in achieving their objectives. Given that WTA planning is probabilistic, this lack of reactivity may result in the use of more weapons than necessary. For example, suppose that several weapons are assigned to a target to achieve the desired probability of success at planning time. The first one or two weapons may destroy the target at runtime, but the lack of re-planning means that the other weapons will continue as planned and attack a target that has already been destroyed. Similarly, if weapon attrition is expected, then five weapons may be assigned to simultaneously arrive at a single target when only two are actually required to reach the target to meet mission objectives. If no weapons are attrited early in the run, then there may be benefit in reassigning one more of those weapons, but a “fire and forget”

approach does not provide this capability. The inability to re-plan on the fly also means that weapons cannot modify their assignments in response to attrition among weapons assigned to other, higher priority targets, e.g., due to accidents or adversarial countermeasures, all of which can also lead to poor outcomes. One way to avoid such inefficiencies is to craft an online algorithm that can react to changing conditions by re-assigning weapons, yet standard centralized approaches will be unable to do so under realistic conditions.

In recent years, there has emerged significant interest in decentralized decision systems in control theory [19], optimization [20], and other fields [21]. Decentralized systems have the advantage that they do not require a centralized coordinator in order for each agent to act. Instead, agents use point-to-point interactions for decision-making. A variety of approaches have fit the WTA problem into a decentralized framework, such as evolutionary algorithms [22], game-theoretic formulations [23], parallel simulated annealing [20], ant colony algorithms [24], heuristic methods [25], nested partitions [26], mixed-integer linear programming [27], and auction algorithms [28]. Additionally, decentralized approaches are used for other components of autonomous weapons systems such as simultaneous interception of a target [29] and collision avoidance [30], which indicates that decentralized tasking algorithms can be incorporated into a holistic control framework. While distributed algorithms for constrained optimization problems have been developed for particular classes of problems [38] or limited models of asynchrony [39], many asynchronous algorithms apply only to unconstrained problem formulations [40, 41]. This leads to a need for a WTA problem formulation that is easy to distribute and an algorithm that tolerates asynchrony and solves constrained optimization problems of a more general form.

Given the advantages of a decentralized approach, in this paper a decentralized solver for the WTA problem is developed that eliminates any dependence upon a central coordinator and provides the ability to re-plan on-the-fly in response to attrition among weapons. The contributions of this paper include: a) a continuous convex relaxation of the classic WTA problem; b) the development of a distributed algorithm that tolerates arbitrarily large bounded delays; c) the derivation of explicit convergence rates that bound the distance from the relaxed problem's solution; and d) the demonstration of results in both simulated and hardware environments. Specifically, this paper has the advantage of explicitly considering the practical challenges inherent in asynchronous communications and computations among agents. First, a continuous convex relaxation of the classic WTA problem is presented that enables the use of convex optimization techniques to solve it. Then a distributed algorithm is developed for the constrained convex optimization problem that results from the relaxation. This algorithm is a first-order decentralized primal-dual algorithm, and all primal communications and computations are allowed to be asynchronous while some occasional coordination is required with dual updates. In contrast to existing averaging-based approaches in which all agents update all decision variables [31–37], this algorithm solves problems with inequality constraints and does so using a *block-based* update law [31, 42–44] in which each primal variable and each dual variable is updated by only a single agent. To the best of our knowledge, the only existing method of this class that permits asynchrony is earlier work by two of the

authors [45, 46]. The current paper extends that work to eliminate persistent errors in convergence and accommodates problems whose objective function does not have a diagonally dominant Hessian matrix (such as the one derived in this paper). Convergence rates are derived to bound the distance of this algorithm to a saddle point of the Lagrangian of the relaxed problem, and this point furnishes a solution to the original problem. Experimental implementations use a combination of commercial off-the-shelf (COTS) ground robots and simulated ground robots as weapon surrogates. Results illustrate the success of this algorithm both under static conditions and in a scenario in which weapons re-plan on-the-fly in response to weapon attrition by updating optimal or sub-optimal assignments that were computed before attrition occurred.

The paper is organized as follows. Preliminaries on the classic WTA problem formulation are provided in Section II. Section III provides the derivation of the continuous convex relaxation of the classic WTA problem formulation. The decentralized algorithm is presented and convergence rates are derived in Section IV. All experimental results are included in Section V. Finally, Section VI concludes and presents possible avenues for future research.

II. Preliminaries and Problem Statement

This section briefly recounts the standard WTA problem formulation, and the next section will detail its convex relaxation. The objective of the WTA problem is to optimally assign some number of weapons to engage each of a finite set of targets (where each weapon can engage at most one target) in order to minimize the expected post-engagement survival value of the targets. This requires three pieces of information: (i) the value of each target, which is V_j for target j; (ii) the effectiveness of weapon i against target j for all i and j, which is formalized by the probability $Pk_{[i],j} \in [0, 1]$ that weapon i destroys target j; and (iii) an accurate model of how collaboration among weapons impacts their effectiveness. This work uses the standard assumption that the cumulative likelihood of a given target being destroyed can be calculated by treating each weapon's engagement as an independent event.

Consider assigning m weapons to ℓ targets. Given that the WTA problem can apply to broader worker/task assignment problems, the targets will be referred to as "tasks" and the weapons as "workers." With this terminology, "engaging" a target is broadened to "engaging in" a task, and "destroying" a target is replaced with "completing" a task. For worker i, the vector $\mathbf{x}_{[i]} \in \mathbb{R}^\ell$ gives its assignments. Specifically,

$$x_{[i],j} = \begin{cases} 1 & \text{worker } i \text{ engages in task } j \\ 0 & \text{otherwise} \end{cases}.$$

Each worker can engage in only one task and $\sum_{j=1}^{\ell} x_{[i],j} = 1$. Combining these features gives the following standard problem formulation for all workers and tasks.

Problem 0 (Preliminary; no relaxation) *Consider m workers and ℓ tasks. Suppose $\mathbf{V} \in \mathbb{R}^\ell$ is given, where V_j is the*

value of task j, along with $Pk \in [0,1]^{m\times\ell}$, where $Pk_{[i],j}$ is the probability that worker i completes task j when it engages in task j. Then

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \sum_{j=1}^{\ell} V_j \prod_{i=1}^{m} \left(1 - Pk_{[i],j} x_{[i],j}\right) \\ \text{subject to} \quad & \mathbf{x} \in \{0,1\}^{m\ell}, \\ & \sum_{j=1}^{\ell} x_{[i],j} = 1 \text{ for all } i. \end{aligned} \qquad \lozenge$$

In this paper, each worker computes its own assignment. Mathematically, worker i computes the optimal value of the vector $\mathbf{x}_{[i]} \in \mathbb{R}^{\ell}$. Problem 0 is relaxed in the next section to enable the implementation and analysis of this decentralized approach.

III. Derivation of Convex Relaxation

The WTA formulation in Problem 0 is known to be NP-hard, and a convex relaxation will enable both efficient solutions and the development of a decentralized algorithm for solving it in Section IV. The goal of the relaxation is to develop an objective function that is both continuous and convex. The continuity requirement is easily satisfied by changing the constraints on the primal variables in Problem 0 from the binary set $\{0,1\}$ to the interval $[0,1]$. Then $x_{[i],j} \in [0,1]$ is regarded as the portion of "effort" expended by worker i on engaging in task j. This produces the relaxed cost function

$$C(\mathbf{x}) = \sum_{j=1}^{\ell} V_j \prod_{i=1}^{m} \left(1 - Pk_{[i],j} x_{[i],j}\right), \quad \text{where } \mathbf{x} = \begin{pmatrix} \mathbf{x}_{[1]} \\ \vdots \\ \mathbf{x}_{[m]} \end{pmatrix} \in [0,1]^{m\ell}, \quad \mathbf{x}_{[i]} \in [0,1]^{\ell}, \quad \text{and } x_{[i],j} \in [0,1]\,. \quad (1)$$

If the minimizer of the relaxed cost takes only values in $\{0,1\}$, then it produces the same value as the original un-relaxed cost.

Unfortunately, C in Eq. (1) is non-convex in $\mathbf{x}$, which will make it difficult to find a global minimum. To make it convex and preserve its equality to the un-relaxed cost on $\{0,1\}$, observe that $x_{[i],j} \in \{0,1\}$ gives

$$1 - Pk_{[i],j} x_{[i],j} = \left(1 - Pk_{[i],j}\right)^{x_{[i],j}}.$$

Then, replacing each term of the form $1 - Pk_{[i],j}x_{[i],j}$ with $\left(1 - Pk_{[i],j}\right)^{x_{[i],j}}$, the relaxed cost function can be written as

$$f(\mathbf{x}) = \sum_{j=1}^{\ell} V_j \prod_{i=1}^{m} \left(1 - Pk_{[i],j}\right)^{x_{[i],j}}. \tag{2}$$

Lemma 1 *The cost function f in Eq. (2) is convex in* $\mathbf{x}$.

Proof: Define the function $h_j(\mathbf{x}) = \prod_{i=1}^{m} \left(1 - Pk_{[i],j}\right)^{x_{[i],j}}$. Then $f(\mathbf{x}) = \sum_{j=1}^{\ell} V_j h_j(\mathbf{x})$, where $V_j \geq 0$ for all j. That is, f is a non-negatively weighted sum of the h_j's, and thus convexity of each h_j will imply convexity of f. Next,

$$\log h_j(\mathbf{x}) = \sum_{i=1}^{m} x_{[i],j} \log\left(1 - Pk_{[i],j}\right),$$

which is linear in $\mathbf{x}$. Then $\log h_j$ is convex in $\mathbf{x}$, and the function h_j is log-convex. Log-convexity implies convexity [47, Section 3.5], and thus each h_j is convex, which implies that f is. ■

Next constraints are relaxed. For all i the un-relaxed problem has $x_{[i],j} = 1$ for at most one j, and all other entries of $\mathbf{x}_{[i]}$ equal zero. This constraint on the un-relaxed problem can be restated as $\sum_{j=1}^{\ell} x_{[i],j} \leq 1$ for all i. This constraint is kept in the relaxed problem, which ensures that no worker can spend arbitrarily large effort on any given task. The set constraint $x_{[i],j} \in \{0, 1\}$ is then simply omitted. The full relaxed problem statement can now be given:

Problem 1 (Relaxed problem) *Given* $\mathbf{V} \in \mathbb{R}^{\ell}$, $Pk \in [0, 1]^{m \times \ell}$, *and* $X = [0, 1]^{m\ell}$,

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) = \sum_{j=1}^{\ell} V_j \prod_{i=1}^{m} \left(1 - Pk_{[i],j}\right)^{x_{[i],j}} \\ \text{subject to} \quad & \mathbf{x} \in X, \\ & g(\mathbf{x}) := \begin{pmatrix} \sum_{j=1}^{\ell} x_{[1],j} - 1 \\ \vdots \\ \sum_{j=1}^{\ell} x_{[m],j} - 1 \end{pmatrix} \leq 0, \end{aligned}$$

where the inequality holds component-wise.

The value of the relaxed cost function equals the un-relaxed cost function when $x_{[i],j} \in \{0, 1\}$. If the optimum of the relaxed problem consists of ones and zeros, then it is clearly also the optimal solution to the original problem. For relaxed solutions that are not in $\{0, 1\}$, Section V demonstrates in simulations and experiments that using the maximum effort as the task assignment often provides the optimum. When it does not provide the optimum, the results have appeared to be near-optimal and far better than fixed assignments, specifically in the case of attrition, which will be discussed in Section V. Additionally, it is emphasized that f is not inherently strongly convex and the Hessian matrix

of f is not diagonally dominant. Thus, previous work by two of the authors [45, 46] does not apply and a new algorithm must be developed.

A special case of interest is that of homogeneous workers or tasks. Homogeneous workers are those who have the same $Pk_{[i],j}$ values for all tasks j and homogeneous tasks have the same values V_j and $Pk_{[i],j}$ for all workers i. It is important to highlight that when homogeneous workers or tasks are considered, the relaxed cost function does not have optimums residing in $\{0, 1\}$. Instead the optima tend to be equally divided among the homogeneous options. For example, considering a simple case of one weapon against two identical targets, the weapon will have assignments of 0.5 for both targets, leading to a deadlock. As duplicated weapons/workers or targets/tasks are a common occurrence, especially in the context of WTA, one remedy is to use Pk values that include the probability of arrival, i.e., adjust Pk due to attrition risk or as small artificial hedge. These modifications to Pk may be distance-dependent, path dependent, or based on something else entirely, and can avoid homogeneity allowing full assignments to be achieved. For this paper, Pk values are adjusted to include the probability of arrival using a simple function of distance and are denoted $P_{[i],j}(\text{Arrival})$ for weapon i arriving at target j. This gives the probability of arrival and kill defined by

$$P_{[i],j}(\text{Arrival and Kill}) := \overline{Pk}_{[i],j} = Pk_{[i],j} P_{[i],j}(\text{Arrival}). \tag{3}$$

This implementation is presented in Section V and is shown to avoid deadlock due to partial assignments.

IV. Decentralized Optimization

This section presents the distributed algorithm used to solve Problem 1, and it proves convergence to an optimum under asynchronous primal communications and computations and intermittent dual computations. In particular, it is shown that Problem 1 is equivalent to a certain Lagrangian saddle point problem, and then a decentralized algorithm is developed to compute saddle points.

A. Lagrangian Saddle Point Formulation

Problem 1 has several properties that will be leveraged in the forthcoming analysis.

Remark 1 *Lemma 1 showed that the objective function f in Problem 1 is convex, and by inspection it is also twice continuously differentiable. Then its Hessian exists and is a continuous function of* $\mathbf{x}$*. The constraint set X is compact because it is a product of compact intervals. The continuity of the Hessian of f and compactness of X imply that the Hessian of f is bounded over X, and thus ∇f is Lipschitz over X. The constraints g satisfy Slater's condition [48, Assumption 6.4.2], i.e., there exists a* Slater point $\bar{\mathbf{x}}$ *such that $g(\bar{\mathbf{x}}) < 0$. In particular, $\bar{\mathbf{x}} = \mathbf{0}$ is a Slater point for g.*

These conditions are known [49] to imply that a solution to Problem 1 can be found by computing a saddle point of

its associated Lagrangian. For the Lagrangian $L(\mathbf{x}, \boldsymbol{\mu}) = f(\mathbf{x}) + \boldsymbol{\mu}^T g(\mathbf{x})$ the point

$$\left(\hat{\mathbf{x}}, \hat{\boldsymbol{\mu}}\right) := \arg\min_{\mathbf{x}\in X} \arg\max_{\boldsymbol{\mu}\in\mathbb{R}^m_+} L(\mathbf{x}, \boldsymbol{\mu})$$

produces $\hat{\boldsymbol{x}}$ as a solution to Problem 1. Here, $\mathbf{x}$ is referred to as the primal variables, $\boldsymbol{\mu}$ is referred to as the dual variables, and the set $\mathbb{R}^m_+$ is the non-negative orthant of $\mathbb{R}^m$. One well-known centralized algorithm to compute saddle points is the seminal Uzawa iteration [50], which simultaneously performs gradient descent in $\mathbf{x}$ and gradient ascent in $\boldsymbol{\mu}$.

To solve Problem 1 in a distributed way, a similar algorithm will be developed that distributes computations among workers and also provides robustness to asynchrony. It has been shown [42, 45, 51–53] that both strong convexity in $\mathbf{x}$ and strong concavity in $\boldsymbol{\mu}$ help provide robustness to asynchrony [31]. Although $L(\cdot, \boldsymbol{\mu})$ is not strongly convex (because neither f nor g is) and $L(\mathbf{x}, \cdot)$ not strongly concave (because it is affine in $\boldsymbol{\mu}$), these properties can be provided via regularization. A Tikhonov regularization [54] with parameters $\boldsymbol{\kappa} = (\alpha, \delta)$ gives the regularized Lagrangian

$$L_{\boldsymbol{\kappa}}(\mathbf{x}, \boldsymbol{\mu}) = f(\mathbf{x}) + \frac{\alpha}{2}\|\mathbf{x}\|^2 + \boldsymbol{\mu}^T g(\mathbf{x}) - \frac{\delta}{2}\|\boldsymbol{\mu}\|^2, \tag{4}$$

where $\alpha > 0$ and $\delta > 0$ are, respectively, the primal and dual regularization parameters. The function $L_{\boldsymbol{\kappa}}(\cdot, \boldsymbol{\mu})$ is α-strongly convex for all $\boldsymbol{\mu}$, and $L_{\boldsymbol{\kappa}}(\mathbf{x}, \cdot)$ is δ-strongly concave for all $\mathbf{x}$. These properties imply that $L_{\boldsymbol{\kappa}}$ has a unique saddle point, denoted $(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}, \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}})$. This saddle point is not equal to that of the unregularized L. However, there exist bounds on the distance between the saddle points of L and $L_{\boldsymbol{\kappa}}$ [54] that show that the error induced by regularizing is small as long as α and δ are small relative to other problem parameters. Therefore, small values of α and δ will be used to ensure that agents compute the saddle point of $L_{\boldsymbol{\kappa}}$ with negligible error.

As described above, all primal computations and communications can be asynchronous. The only assumption imposed on them is the following.

Assumption 1 (Bounded Delays) *Let K^i be the set of times at which worker i performs primal updates and $\tau^i_j(k)$ be the time at which worker j originally computed the primal update received by worker i at time k. There exists $B \geq 1$ such that*

- $\{k, k+1, \ldots, k+B-1\} \cap K^i \neq \emptyset$*, for all $k \geq 0$ and all i;*
- $0 \leq k - \tau^i_j(k) \leq B-1$*, for all $k \in K^i$, all j, and all i.* △

This ensures that no entry of a worker's primal block was computed more than B iterations prior.

To aid in the forthcoming analysis, a compact subset of $\mathbb{R}^m_+$ is computed that contains the dual component of the saddle point of $L_{\boldsymbol{\kappa}}$, which is $\hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}$.

Lemma 2 *Consider the regularized Lagrangian $L_{\boldsymbol{\kappa}}$ in Eq.* (4) *and let $(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}, \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}})$ denote its unique saddle point. Then*

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}} \in \mathcal{M} := \left\{ \boldsymbol{\mu} \in \mathbb{R}_+^m : \|\boldsymbol{\mu}\|_1 \leq \sum_{j=1}^{\ell} V_j \right\}.$$

Proof: The discussion in [51, Section II-C] shows that for any Slater point $\bar{\mathbf{x}}$ of g

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}} \in \mathcal{M} = \left\{ \boldsymbol{\mu} \in \mathbb{R}_+^m : \|\boldsymbol{\mu}\|_1 \leq \frac{f(\bar{\mathbf{x}}) + \frac{\alpha}{2}\|\bar{\mathbf{x}}\|^2 - \min\limits_{\mathbf{x}\in X} f(\mathbf{x})}{\min\limits_{1\leq p\leq m} -g_p(\bar{\mathbf{x}})} \right\}. \tag{5}$$

The non-negativity of f in Problem 1 implies non-negativity of $\min_{\mathbf{x}\in X} f(\mathbf{x})$. Remark 1 gives $\bar{\mathbf{x}} = \mathbf{0}$. Noting that $g(\mathbf{0}) = (-1, \ldots, -1)^T$ and $\min\limits_{1\leq p\leq m} -g_p(\bar{\mathbf{x}}) = 1$, the upper bound in Eq. (5) can be bounded via

$$\frac{f(\bar{\mathbf{x}}) + \frac{\alpha}{2}\|\bar{\mathbf{x}}\|^2 - \min\limits_{\mathbf{x}\in X} f(x)}{\min\limits_{1\leq p\leq m} -g_p(\bar{\mathbf{x}})} \leq f(\mathbf{0}) - \min_{\mathbf{x}\in X} f(\mathbf{x}) \leq f(\mathbf{0}),$$

and the lemma follows. ■

Thus, Problem 1 is equivalent to the following saddle point problem over the compact domain $X \times \mathcal{M}$.

Problem 2 *Let Assumption 1 hold and fix $\alpha, \delta > 0$. For $L_{\boldsymbol{\kappa}}$ defined in Eq.* (4)*, asynchronously compute*

$$\left(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}, \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\right) := \arg\min_{\mathbf{x}\in X} \arg\max_{\boldsymbol{\mu}\in\mathcal{M}} L_{\boldsymbol{\kappa}}(\mathbf{x}, \boldsymbol{\mu}). \qquad \lozenge$$

B. Accuracy of Solutions

This section shows that the solutions to Problems 1 and 2 are close to those of the original WTA problem given in Problem 0. This is done in two ways: through comparison of all three solutions for small-scale engagements, and through a comparison of the solution to Problem 2 to Problem 0 for a large-scale engagement. First, the solution to Problem 0 is compared to those of Problems 1 and 2. For a five-weapon, four-target scenario Pk values were randomly generated from a uniform distribution on $[0.2, 0.9]$ and target values were randomly generated from a uniform distribution on $[2, 10]$. A brute-force method was used to calculate the solution to Problem 0 by iterating over all possible assignments. The solution to Problem 1 was calculated using SciPy's minimize() solver [55, 56]. Finally, the solution to Problem 2 was calculated using centralized gradient descent (which is essentially equivalent to a centralized, synchronous version of Algorithm 1) with regularization parameters set to $\alpha = \delta = 0.01$. For each engagement, the solutions to Problems 1 and 2 were compared to the solution to Problem 0. This comparison was repeated one thousand times with Pk and V randomly generated each time. Results are displayed in Fig. 1 in the form of overlaid histograms. The two histograms

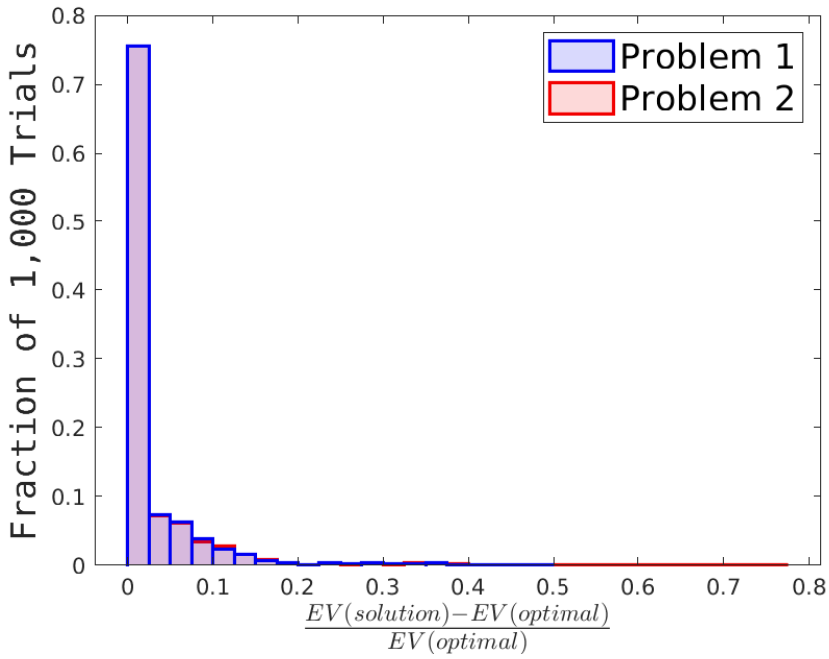


Fig. 1 Histogram showing the error ratio in the solutions to Problems 1 and 2 relative to Problem 0 across $1,000$ randomly generated scenarios. These errors are shown to typically be small, which indicates that Problems 1 and 2 well-approximate Problem 0.

show the fraction of total runs that achieved a given relative error in the expected values. A value of zero means that the solution to Problem 1 or 2 matches that of Problem 0 and a value of 0.5 means that the solution to Problem 1 or 2 gave a cost that was 50% higher than the solution to Problem 0 for that scenario. As evidenced by the figure, the histogram is clustered near zero which indicates that the solutions to both Problems 1 and 2 are close to the solution to Problem 0.

The solution to Problem 2 was also compared to the solution to Problem 0 for large-scale engagement scenarios. In particular, scenarios were considered with 100 weapons and 40 targets, with Pk values drawn uniformly from the interval $[0.2, 0.9]$ and target values drawn uniformly from $[2, 10]$. To solve Problem 0, two different solvers were used: an implementation of the graph-based optimization algorithm from [9] and Ipopt [57], which was called using the modeling library Pyomo [58, 59]. These two solutions to Problem 0 were compared to the solution to Problem 2 computed using centralized gradient descent. For Problem 0, the solution generated by the graph-based algorithm had an expected value of 1.35 and the solution generated by Ipopt had an expected value of 1.44. The gradient descent solution had an expected value of 1.87 for Problem 2. To put these numbers in context, the worst-case expected value is 224.4 (total value of all targets). One million random candidate assignments were generated to approximate the expected value of an arbitrary assignment and a mean of 56.2 was found. Thus, while the expected value associated with the solution to Problem 2 was slightly higher than that associated with the solutions for Problem 0, the difference is negligible when considered as a 97.6% (graph-based), 97.4% (Ipopt), and 96.7% (gradient descent) decrease in

expected value relative to the random assignment. Therefore, we proceed with solving Problem 2 with the understanding that it is an effective surrogate for Problem 0.

The algorithm for solving Problem 2 is presented next.

C. Asynchronous Primal-Dual Algorithm with Bounded Delays

Problem 2 is solved using a first-order gradient-based method because such methods naturally offer some robustness to asynchrony and are simple to implement. Computations are assigned to the workers as follows.

Worker i is responsible for updating the i^{th} block of the primal variable x and the i^{th} entry of the dual variable $\boldsymbol{\mu}$. This primal block is $\mathbf{x}_{[i]}$, which is simply worker i's own assignments to each task, and it must lie in the set $X_i := [0,1]^\ell$. The dual variable μ_i encodes the constraint $\sum_{j=1}^{\ell} x_{[i],j} \leq 1$, which bounds worker i's effort. There are ℓ tasks and efforts assigned to individual tasks take values in $[0,1]$, which gives $\mathbf{x}_{[i]} \in [0,1]^\ell$. The optimal value of the dual variable μ_i must lie in the set

$$\mathcal{M}_i := \left\{ \nu \in \mathbb{R}_+ : \nu \leq \sum_{j=1}^{\ell} V_j \right\}.$$

Worker i will project its primal updates onto X_i and its dual updates onto $\mathcal{M}_i$ in the forthcoming algorithm.

Operations in the primal and dual spaces are next separately examined. Then a formal algorithm statement is given.

(i) Computations and Communications of Updates to Primal Variables All workers index their computations with the same iteration counter $k \in \mathbb{N}$, but each one may or may not compute a new value of its decision variable at each tick of k. The set $K^i \subseteq \mathbb{N}$ denotes the subset of times at which worker $i \in \mathcal{I}$ computes an update. Assumption 1 enforces that there are at most B timesteps between consecutive primal updates, and there is not assumed to be any relationship between the set of times at which worker i performs updates and the set of times at which worker j performs updates, i.e., K^i need not have any relationship to K^j for $i \neq j$.

Workers also communicate their updated primal variables asynchronously. The set $C_j^i \subseteq \mathbb{N}$ denotes the infinite set of times at which worker i sends updated values of $\mathbf{x}_{[i]}$ to worker j (recall that they can be received later due to asynchrony). The times at which worker i's variables are communicated to other workers may differ from those of worker j, i.e., C_j^i need not have any relationship to C_r^i for $j \neq r$. These sets are also only used for analysis, and, in particular, they need not be known to the workers. The only assumption in force remains Assumption 1, which simply ensures that a worker's primal blocks were not computed more than B iterations prior.

(ii) Computations of Updates to Dual Variables To analytically establish dual convergence, all workers are required to periodically update their dual variable entries. Formally, all workers must perform a single update of their dual variable when $k = tB$ for all $t \in D$, where $D \subseteq \mathbb{N}$ is some infinite subset. Updates at the times in D are the only coordination required among workers. These updates do not require a communication event to also occur at the same

time. Thus, even if workers are in a communications-denied environment, the coordination is still able to proceed. Practical implementation details will be discussed in Section V.

We note that while worker i stores $\boldsymbol{\mu}^i$ onboard, only its i^{th} entry, μ_i^i, actually appears in its computations. Therefore, workers do not exchange values of their dual variables. In Algorithm 1, the notation indicates that worker i uses $\boldsymbol{\mu}^i$ in its computations, and this is done only to keep dimensions consistent. Only the value of μ_i^i actually affects its computations, and other entries of $\boldsymbol{\mu}^i$ can be fixed, e.g., held constant at zero, for the duration of the algorithm.

Statement of Algorithm A formal statement of the primal-dual algorithm for solving Problem 2 follows, where $\nabla_{\mathbf{x}_{[i]}}$ is the derivative with respect to the i-th block of $\mathbf{x}$ and $\Pi_S[\mathbf{y}]$ denotes the Euclidean projection of the point $\mathbf{y}$ onto the non-empty, compact, convex set S.

Algorithm 1 Distributed Gradient-Based Primal-Dual Optimization

Step 0: Initialize all workers with $\mathbf{x}(0) \in X$ and $\boldsymbol{\mu}(0) \in \mathcal{M}$. Set $k = 0$.

Step 1: For all $i, j \in \mathcal{I}$, if $k \in C_j^i$, then worker i sends $\mathbf{x}_{[i]}^i(k)$ to worker j.

Step 2: For all $i \in \mathcal{I}$, if $k \in K^i$, worker i executes the primal update

$$\mathbf{x}_{[i]}^i(k+1) = \Pi_{X_i}\left[\mathbf{x}_{[i]}^i(k) - \gamma \nabla_{\mathbf{x}_{[i]}} L_{\boldsymbol{\kappa}}\big(\mathbf{x}^i(k), \boldsymbol{\mu}^i(k)\big)\right].$$

If $k \notin K^i$, then $\mathbf{x}_{[i]}^i(k+1) = \mathbf{x}_{[i]}^i(k)$.

Step 3: For all $i, j \in \mathcal{I}$,

$$\mathbf{x}_{[j]}^i(k+1) = \begin{cases} \mathbf{x}_{[j]}^j(\tau_j^i(k+1)) & \text{worker } i \text{ receives a new value of } \mathbf{x}_{[j]}^j \text{ at time } k+1 \\ \mathbf{x}_{[j]}^i(k) & \text{otherwise} \end{cases}.$$

Step 4: Set $k := k + 1$.

Step 5: For all $i \in \mathcal{I}$, if $k = tB$ for $t \in D$, worker i executes the dual update

$$\mu_i^i(tB) = \Pi_{\mathcal{M}_i}\left[\mu_i^i(tB-1) + \rho \frac{\partial L_{\boldsymbol{\kappa}}}{\partial \mu_i}\Big(\mathbf{x}^i(tB), \boldsymbol{\mu}^i(tB-1)\Big)\right].$$

Step 6: Return to Step 1.

D. Primal Convergence

This section establishes primal convergence rates for a fixed dual variable. That is, convergence is analyzed for computations of workers that are asynchronously performing gradient descent to minimize $L_{\boldsymbol{\kappa}}\big(\,\cdot\,, \boldsymbol{\mu}(tB)\big)$ with $\boldsymbol{\mu}(tB)$

fixed. At times $k \in K^i$ after computing $\boldsymbol{\mu}(tB)$ and prior to the next dual update, worker i runs

$$\mathbf{x}^i_{[i]}(k+1) = \Pi_{X_i}\left[\mathbf{x}^i_{[i]}(k) - \gamma \nabla_{\mathbf{x}_{[i]}} L_{\boldsymbol{\kappa}}\big(\mathbf{x}^i(k), \boldsymbol{\mu}(tB)\big)\right]$$

to drive its iterates towards $\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(tB) := \arg\min_{\mathbf{x}\in X} L_{\boldsymbol{\kappa}}(\mathbf{x}, \boldsymbol{\mu}(tB))$. Workers can perform computations asynchronously as long as Assumption 1 is satisfied. Under this assumption, existing work in [60] can be leveraged to quantify convergence of the primal variable.

Lemma 3 *Let Assumption 1 hold, and consider using Algorithm 1 to solve Problem 2. Fix $t \in D$ and let t' be the smallest element of D that is greater than t. Fix $\boldsymbol{\mu}(tB) \in \mathcal{M}$ and define $\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(tB) = \arg\min_{\mathbf{x}\in X} L_{\boldsymbol{\kappa}}(\mathbf{x}, \boldsymbol{\mu}(tB))$. Then for workers executing Algorithm 1 for times $k \in \{tB, tB+1, \ldots, t'B\}$, there exists a scalar $\gamma_1 > 0$ such that for $0 < \gamma < \gamma_1$*

$$\|\mathbf{x}(t'B) - \hat{\mathbf{x}}_{\boldsymbol{\kappa}}(tB)\| \leq (1 - c\gamma)^{t'-t} \|\mathbf{x}(tB) - \hat{x}_{\boldsymbol{\kappa}}(tB)\|,$$

where c is a positive constant, $(1 - c\gamma) \in [0, 1)$.

Proof: See the appendix. ∎

E. Dual Convergence

Lemma 3 quantifies the behavior of the primal variable between dual updates, and this subsection quantifies the behavior of the dual variable when it is updated over time. The following lemma is first stated, and it will be used later in the proof of the main convergence theorem.

Lemma 4 *Let $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2 \in \mathbb{R}^m_+$. Then for any solutions $\mathbf{x}_1$ and $\mathbf{x}_2$ such that*

$$\mathbf{x}_1 = \arg\min_{\mathbf{x}\in X} L_{\boldsymbol{\kappa}}(\mathbf{x}, \boldsymbol{\mu}_1) \quad \text{and} \quad \mathbf{x}_2 = \arg\min_{\mathbf{x}\in X} L_{\boldsymbol{\kappa}}(\mathbf{x}, \boldsymbol{\mu}_2)$$

the pairs $(\mathbf{x}_1, \boldsymbol{\mu}_1)$ and $(\mathbf{x}_2, \boldsymbol{\mu}_2)$ satisfy

$$\big(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1\big)^T \big(-g(\mathbf{x}_2) + g(\mathbf{x}_1)\big) \geq \frac{\alpha}{M^2} \|g(\mathbf{x}_2) - g(\mathbf{x}_1)\|^2,$$

where $M := \max_{\mathbf{x}\in X} \|\nabla g(\mathbf{x})\|$.

Proof: See Lemma 4.1 in [44]. ∎

This lemma is used to upper bound the change in distance from the dual optimum after workers perform a dual update in Step 5 of Algorithm 1.

Theorem 1 *Let Assumption 1 hold and let the dual stepsize $\rho > 0$ in Algorithm 1 satisfy*

$$\rho < \min\left\{\frac{2\alpha}{m\ell + 2\alpha\delta}, \frac{2\delta}{1+\delta^2}\right\}.$$

Denote with t_1 and t_2 two consecutive times that dual updates have occurred, with $t_1 < t_2$ and $t_1, t_2 \in D$. When solving Problem 2, the sequence of dual variables generated by Algorithm 1 satisfies

$$\|\boldsymbol{\mu}(t_2 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 \leq q_d \|\boldsymbol{\mu}(t_1 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 + q_d q_p^{2(t_2 - t_1)} m^2 \ell^2 + 2\rho^2 q_p^{t_2 - t_1} m^2 \ell^2,$$

where $q_d := (1-\rho\delta)^2 + \rho^2 \in [0, 1)$, $q_p := (1 - c\gamma) \in [0, 1)$, m is the number of workers, and ℓ is the number of tasks.

Proof: See the appendix. ∎

The terms $q_d q_p^{2(t_2-t_1)} m^2 \ell^2$ and $2\rho^2 q_p^{t_2-t_1} m^2 \ell^2$ are dependent on input parameters and the number of primal iterations that occur between dual updates. These terms may be reduced by allowing more time to elapse between dual updates, which makes $t_2 - t_1$ larger and thus makes $q_p^{t_2-t_1}$ smaller. These terms may also be reduced through the choice of input parameters. For example, the last term may be reduced by simply choosing a small dual stepsize ρ. As intuition would expect, there is an associated cost to scaling the algorithm to larger numbers of workers and tasks, which is reflected by the $m\ell$ factor of both terms, and this term grows as the number of workers and/or number of tasks grows.

This theorem may be recursively applied to find the distance from the optimal dual variable at each k such that $k = tB$ for some $t \in D$. This is done next.

Theorem 2 *Let Assumption 1 hold and consider the use of Algorithm 1 to solve Problem 2. Let the dual stepsize $\rho > 0$ satisfy*

$$\rho < \min\left\{\frac{2\alpha}{m\ell + 2\alpha\delta}, \frac{2\delta}{1+\delta^2}\right\}.$$

Let t_n denote the n^{th} entry in D where $t_1 < t_2 < \cdots < t_n$. That is, $t_n B$ is the time at which the n^{th} dual update occurs across all workers. Then workers executing Algorithm 1 generate dual variables that satisfy

$$\|\boldsymbol{\mu}(t_n B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 \leq q_d^n \|\boldsymbol{\mu}(0) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 + \left(q_d q_p^2 m^2 \ell^2 + 2\rho^2 q_p m^2 \ell^2\right) \frac{1 - q_d^{n+1}}{1 - q_d},$$

where $q_d = (1-\rho\delta)^2 + \rho^2 \in [0, 1)$, $q_p = (1 - c\gamma) \in [0, 1)$, m is the number of workers, and ℓ is the number of tasks.

Proof: Recursively applying Theorem 1 gives

$$\begin{aligned}\|\boldsymbol{\mu}(t_nB) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 &\leq q_d\|\boldsymbol{\mu}(t_{n-1}B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 + q_d q_p^{2(t_n - t_{n-1})} m^2\ell^2 + 2\rho^2 q_p^{t_n - t_{n-1}} m^2\ell^2 \\ &\leq q_d^2\|\boldsymbol{\mu}(t_{n-2}B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 + q_d^2 q_p^2 m^2\ell^2 + 2\rho^2 q_d q_p m^2\ell^2 + q_d q_p^2 m^2\ell^2 + 2\rho^2 q_p m^2\ell^2 \\ &\leq q_d^n\|\boldsymbol{\mu}(0) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 + (q_d q_p^2 m^2\ell^2 + 2\rho^2 q_p m^2\ell^2)\sum_{i=0}^{n} q_d^i,\end{aligned}$$

where the second inequality follows because $t_n - t_{n-1} \geq 1$. Because $q_d \in [0, 1)$, summing the geometric series in the last term gives the final result. ∎

F. Overall Convergence

Now Lemma 3 and Theorem 2 can be combined to establish an upper bound on the distance from the primal optimum at each time.

Theorem 3 *Consider Problem 2 and let Assumption 1 hold. Let the dual stepsize $\rho > 0$ satisfy $\rho < \min\left\{\frac{2\alpha}{m\ell + 2\alpha\delta}, \frac{2\delta}{1+\delta^2}\right\}$. Let t_n denote the n-th entry in D where $t_1 < t_2 < \cdots < t_n$. That is, t_nB is the time at which the n^{th} dual update occurs across all workers. Then workers executing Algorithm 1 generate primal iterates that satisfy*

$$\|\mathbf{x}(t_nB) - \hat{\mathbf{x}}_{\boldsymbol{\kappa}}\| \leq q_p^{t_n - t_{n-1}}\sqrt{m\ell} + \frac{\sqrt{m\ell}}{\alpha}\sqrt{q_d^n\|\boldsymbol{\mu}(0) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 + \left(q_d q_p^2 m^2\ell^2 + 2\rho^2 q_p m^2\ell^2\right)\frac{1 - q_d^{n+1}}{1 - q_d}},$$

where $q_p = (1 - c\gamma) \in [0, 1)$, $q_d = (1-\rho\delta)^2 + \rho^2 \in [0, 1)$, m is the number of workers, and ℓ is the number of tasks.

Proof: Following the steps in the proof of Theorem 2 in [52] gives

$$\|\mathbf{x}(t_nB) - \hat{\mathbf{x}}_{\boldsymbol{\kappa}}\| \leq q_p^{t_n - t_{n-1}}\sqrt{m\ell} + \frac{\sqrt{m\ell}}{\alpha}\|\boldsymbol{\mu}(t_nB) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|.$$

Applying Theorem 2 (from the current paper) completes the proof. ∎

Remark 2 *The results in Theorem 3 illustrate a trade-off between the quality of final answer obtained and the rate of convergence to that solution, particularly as influenced by the regularization parameters α and δ. As stated previously, smaller regularizations lead to smaller regularization error [54]. In Theorem 3, this means that smaller values of α and δ provide a solution that is closer to the solution to the unregularized problem, and thus such a solution is of higher quality. However, the second term in Theorem 3 is pre-multiplied by $\frac{1}{\alpha}$, and small values of α make this term large and thus require more iterations to reach a solution. In addition, the dual contraction term q_d is a function of δ, with smaller δ making q_d larger. Thus, smaller values of α and δ can give a solution of higher quality, but this solution requires more iterations to reach, as quantified by Theorem 3. Similarly, from Theorem 3 we see that larger values of α and δ give a lower-quality solution, but it can be reached in fewer iterations.*

V. Experimental Results

This section presents experimental results that illustrate the performance of Algorithm 1*. While the results of this paper may be used for any worker/task assignments, this section considers a weapon-target assignment (WTA) problem specifically. Each weapon has a likelihood of killing a given target, which is the weapon's "probability of kill" for that target. The weapons work collectively to minimize the "total expected value" given by f in Problem 1, and Algorithm 1 is deployed on a combination of physical and simulated weapon surrogates, in this case physical and simulated TurtleBots. The experiments we present provide experimental testing and validation of the actual execution of the weapon-target assignments that are generated using Algorithm 1. Indeed, the outputs of Algorithm 1 (and other WTA algorithms) are simply assignments of weapons to targets, but these assignments must actually be executed by mobile autonomous agents with non-trivial dynamics. The robots we use provide a realistic representation of the types of vehicles and types of dynamics one could deploy in a WTA application. Moreover, to demonstrate that our algorithm can accommodate asynchrony among agents, running on robots gives a realistic test case in which asynchrony naturally arises and thus provides an opportunity to illustrate the success of our developments in an actual deployment of the kind for which they were designed.

The first scenario demonstrates a team of weapons cooperatively solving Problem 2 in real time in a physics simulation environment. In the second scenario, the performance of Algorithm 1 is evaluated when considering a larger number of weapons and targets. This scenario also accounts for homogeneous weapons and targets by adjusting the Pk values based on distance and associated attrition risk as discussed in Section III. The third scenario demonstrates the benefits of the decentralized setup by attriting a weapon midway through the optimization to show that the other weapons are able to react to and compensate for this loss.

A. Combined Simulation and Hardware Setup

Robot Operating System (ROS) [61] is used to enable message passing between weapon surrogates over a local network, and Gazebo [62], a physics-based simulation environment, is used to model the behavior of a physical robot. As described above, each weapon updates a subset of the primal variables and an entry of the dual variable that encodes the constraint on its own assignments. The dual variable enforces the constraint that the sum of a weapon's efforts should not exceed one. The primal variables are the portion of effort a weapon should expend on each target. The arg max over these efforts is then used to determine a target assignment for each weapon. Although this type of approach cannot be used for all relaxations of discrete optimization problems, it is shown to provide an appropriate level of performance in the WTA problems considered and does so while tolerating asynchronous primal communications and computations.

The primal variables are initialized to a uniform engagement, equal to $\frac{1}{\ell}$ for each entry, and the dual variables are initialized to zero. To simulate asynchronous communications, each weapon generates a random time delay (upper

*Code for this section may be found at `https://github.com/uf-reef-avl/distributed_wta/`.

(a) (b)

Fig. 2 (a) Screenshot of the Gazebo Environment. (b) Photo of the ground robots used for the attrition scenario.

bounded by B) between communications of its primal variables. After its most recent update has been communicated, each weapon samples a uniform distribution with the primal communication delay bounds to generate a new time delay between its communications to other agents. The computer clock is used to coordinate dual updates periodically (Step 5 of Algorithm 1) where all weapons update their dual variables. As discussed in Section IV, these coordinated updates do not require coordinated communications and unless otherwise noted, this coordination occurs every B seconds. Although outside the scope of this effort, it could also be assumed that weapons use GPS for time, that communications (when they occur) include a time transfer capability that help all system clocks maintain an accurate time, or that over the duration of the mission the individual weapons have high enough quality clocks that clock drift is not a concern.

For a hardware implementation, each weapon is a ground robot that runs the optimization algorithm onboard an Intel NUC with an Intel i5 processors. The simulated weapons are also represented as ground robots in Gazebo, and the computations for all the simulated weapons run onboard a laptop computer with an Intel i9 processor. The weapons and targets are visualized using RViz. Figure 2 shows a screenshot of the Gazebo screen (in Fig. 2a) and a photo of the hardware weapons (Fig. 2b) interacting during a test run.

B. Simulated Baseline Scenario

The first scenario simulates a team of five weapons being assigned to four targets with parameters given in Table 1. The importance of target j (its value, V_j) and the likelihood of a target being killed by weapon i when engaging it (probability of kill, $Pk_{[i],j}$) were created to be heterogeneous to allow the use of a constant Pk throughout the scenario rather than adjusting it to include the probability of arrival.

Figure 3a shows the evolution in the assignments of each weapon, which is computed as $\arg\max_j x_{[i],j}$ for each weapon i at each iterate. Figure 3b shows the projected Pk for each of the four targets over the duration of the simulation. The reported Pk at a given time is a function of the weapons currently assigned to that particular target at that time and the weapon's associated Pk value for that particular target. The final assignments at the end of the simulation were 1-2-2-4-3, meaning Weapon 1 was assigned to Target 1, Weapon 2 was assigned to Target 2, and so on. The optimality

Table 1 Parameters used for the simulated baseline scenario

Parameter	Value
Pk	$\begin{bmatrix} 0.8 & 0.5 & 0.1 & 0.1 \\ 0.6 & 0.5 & 0.1 & 0.1 \\ 0.2 & 0.5 & 0.1 & 0.1 \\ 0.3 & 0.1 & 0.7 & 0.5 \\ 0.3 & 0.1 & 0.7 & 0.3 \end{bmatrix}$
$\mathbf{V}$	$\begin{bmatrix} 7 & 4 & 2 & 4 \end{bmatrix}$
Primal Communication Delay Lower Bound	0.1 seconds
Primal Communication Delay Upper Bound, B	1.0 seconds
δ	0.01
ρ	0.0009
γ	0.01
α	0.01

of the assignments was confirmed using a brute-force approach which calculated the expected value for every possible assignment. Figure 4 shows the evolution of the expected values of each target post-engagement and the total expected value, which is simply the sum of the expected values for each target.

All weapons initially choose to go after the high-valued Target 1 until they recognize that Weapon 1 is already assigned to it. Weapon 2's Pk values were intentionally designed to show how weapons adapt to others' assignments; when Weapon 2 realizes that Weapon 1 is assigned to the high-valued Target 1, it is then able to further reduce the total expected value by engaging Target 2 instead. Similarly, Weapon 5 engages Target 4 over Target 3.

The speed at which weapons are influenced by others' assignments and adjust their own is related to the delay bound B. Further experimentation confirmed that the larger the delay bound B, the longer it takes weapons to reach their optimal assignments. Additionally, since B is also related to dual updates which gradually enforce constraints, larger values of B also result in more indecisiveness at the beginning of a simulation. This can be seen in Weapons 2 and 5 around 25 seconds as this is the point that the constraints g become strictly enforced. This results in a shift of assignment and a temporary increase in the expected value. As weapons continue optimizing, they are able to reduce the expected value by shifting assignments once again. This shift takes time as weapons gradually change their primal variables (those responsible for assignments). This demonstrates the trade-off between frequent dual updates which enforce the constraints earlier (but require more coordination) and less frequent dual updates which may result in a shift in assignments later in the scenario (as seen here).

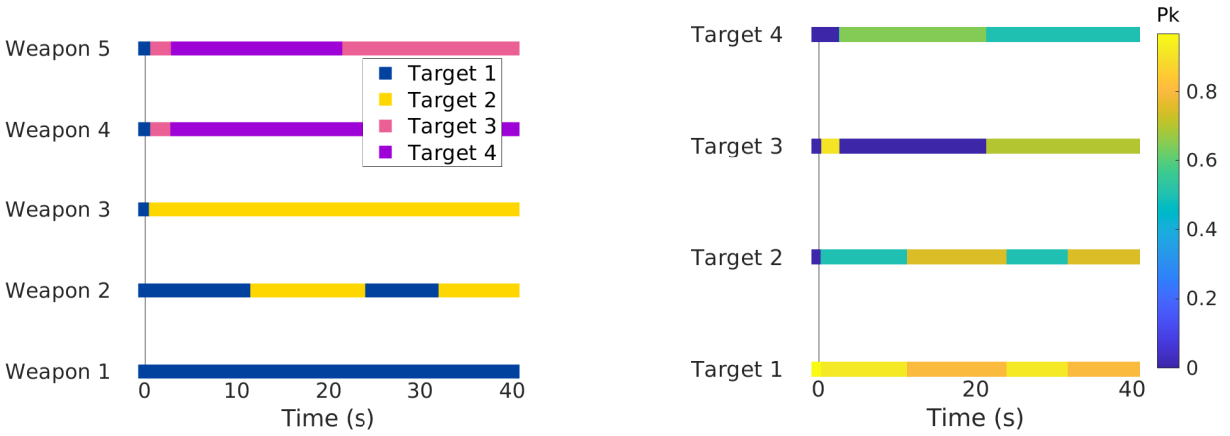


Fig. 3 Simulated baseline scenario: (a) The assignments for each of the five weapons over time. (b) The projected Pk for the targets based on the assignments over time.

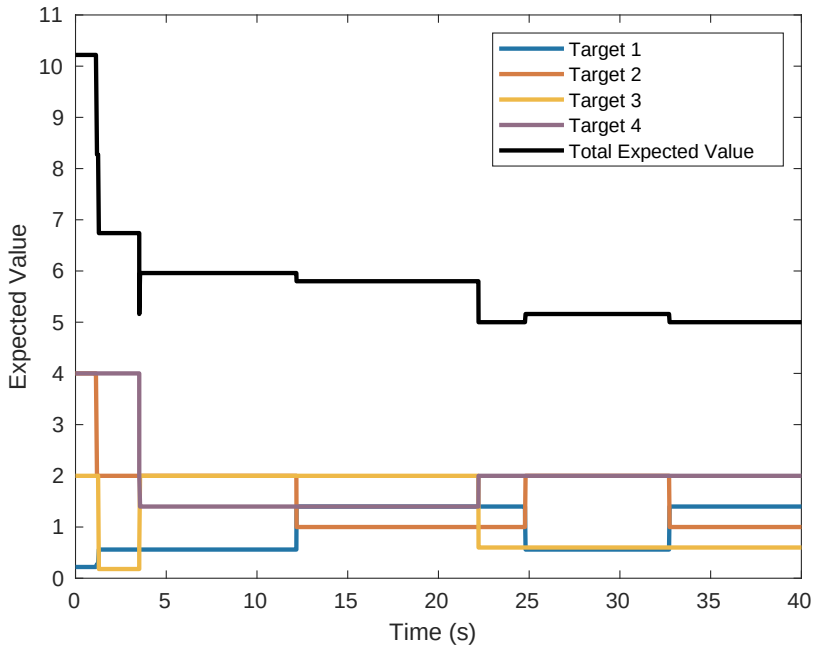


Fig. 4 Evolution of the expected values for each of the four targets and the total expected value (in black) for simulated baseline scenario with five weapons and four targets.

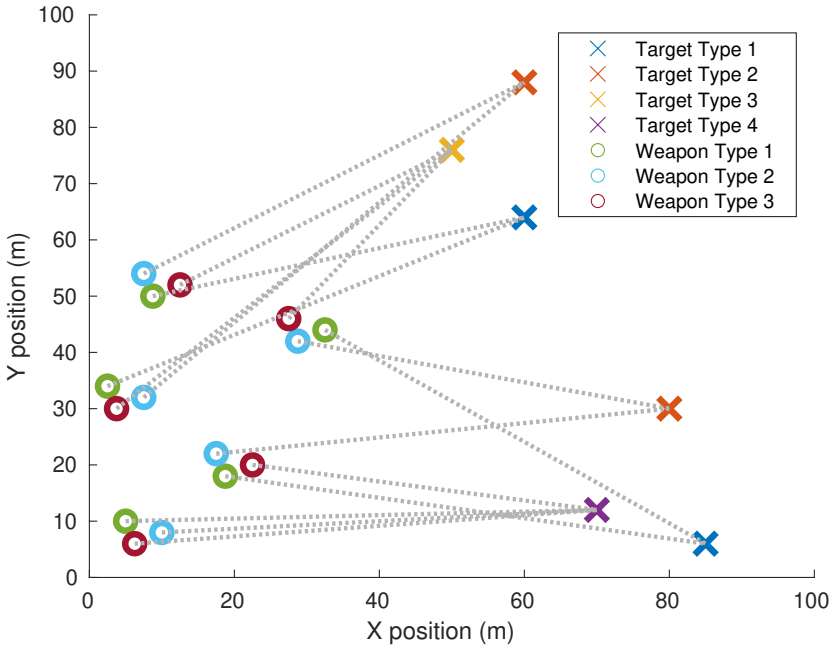


Fig. 5 Initial positions for weapons and targets in the homogeneous scenario. The dotted grey lines indicate the final weapon-target assignments.

C. Homogeneous Weapons and Targets Scenario

Algorithm 1 was also run for fifteen weapons versus six targets. Three types of weapons (with five of each type) were considered against four types of targets. Table 2 shows the parameters used. Each weapon type was strong against a particular target type and all were equally weak against Target 6. As discussed in Section III, considering the probability of arrival accommodates homogeneous weapons and targets. P(Arrival) was calculated in relation to distance from the targets, both at the beginning of the simulation and every B seconds afterwards. In particular, we defined a very simplistic P(Arrival) for weapon i against target j as

$$P_{[i],j}(\text{Arrival}) = 1 - 0.1(\text{Distance between weapon } i \text{ and target } j).$$

This is then used in Eq. (3) to determine the probability of arrival and kill. This is a very simplistic placeholder (one that works for the distances associated in the scenario presented here) for more realistic probability of arrival models but shows deadlock avoidance while allowing future work to progress to more complicated scenarios, e.g., interesting mixes of weapons/targets, flyable path constraints, path dependent attrition, etc.

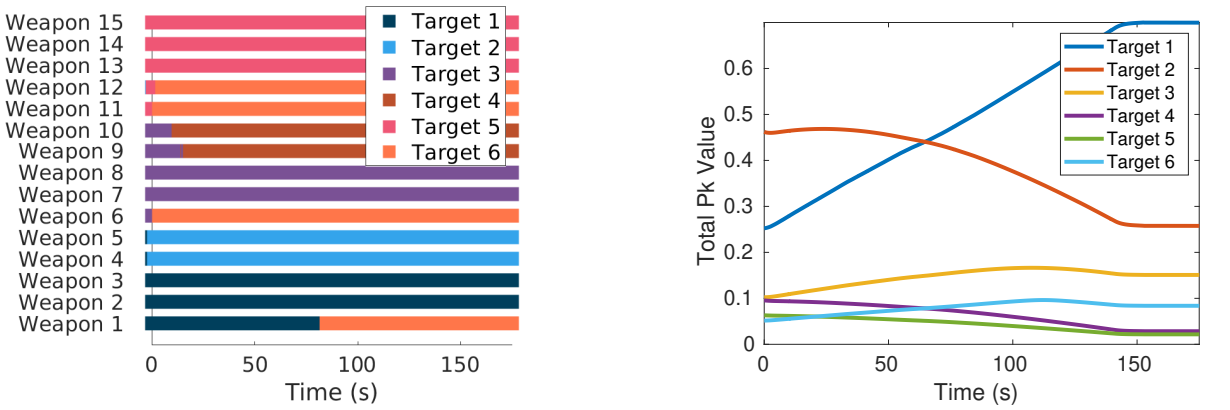
The initial positions of the weapons and targets are displayed in Figure 5 with the dotted grey lines indicating the final assignments. Intuitively, weapons tend to be assigned to targets that are closest to them. This also prevents the indecision that sometimes occurs towards the end of a simulation when a weapon is deciding between two options that

Table 2 Parameters used for the homogeneous scenario

Parameter	Value
Pk	$\begin{bmatrix} 0.7 & 0.7 & 0.2 & 0.2 & 0.1 & 0.1 \\ 0.7 & 0.7 & 0.2 & 0.2 & 0.1 & 0.1 \\ 0.7 & 0.7 & 0.2 & 0.2 & 0.1 & 0.1 \\ 0.7 & 0.7 & 0.2 & 0.2 & 0.1 & 0.1 \\ 0.7 & 0.7 & 0.2 & 0.2 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.6 & 0.6 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.6 & 0.6 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.6 & 0.6 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.6 & 0.6 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.6 & 0.6 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.5 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.5 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.5 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.5 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.5 & 0.1 \end{bmatrix}$
V	$\begin{bmatrix} 4 & 4 & 5 & 5 & 6 & 8 \end{bmatrix}$
Primal Communication Delay Lower Bound	0.1 seconds
Primal Communication Delay Upper Bound, B	0.5 seconds
δ	0.01
ρ	0.002
γ	0.01
α	0.1

lead to very similar expected values. Here, as a weapon gets close to a target it is discouraged from changing course to a similar target due to the distance-dependent $\overline{Pk}$ values.

Figure 6a shows the evolution of the assignments over the simulation. By considering the P(Arrival), homogeneous weapons are ultimately assigned to different targets including ones they are weaker against. Because weapons and targets are at various distances, this provides sufficient heterogeneity to avoid a deadlock of partial assignments as discussed in Section III. Rather than displaying the projected Pk for all targets, Figure 6b shows the changing $\overline{Pk}$ values for Weapon 3 over the course of the simulation. As Weapon 3 reaches Target 1, the corresponding $\overline{Pk}$ converges to the unaltered Pk in Table 2. Similarly, as it increases in distance from Target 2, the value of $\overline{Pk}$ decreases. Results also demonstrate the efficiency of the algorithm in real-time adjustment of the problem being considered, as each new $\overline{Pk}$ table equates to a new problem for the weapons to solve. It also better reflects real-world applications where mission planners may want to account for the difficulty of reaching particular targets. The expected value plot is not provided as



(a) **(b)**

Fig. 6 Results for the homogeneous scenario: (a) The assignments for each of the fifteen weapons over time. (b) The total $\overline{Pk}$ for Weapon 3 as a function of time.

the changing $\overline{Pk}$ values make it difficult to determine the effect of assignment versus the weapons' changing proximity to targets.

D. Attrition Scenario

To demonstrate the benefits of the decentralized setup, an attrition scenario is implemented in which a weapon is shut down while pursuing a target. The other weapons recognize that the weapon has been attrited and adapt the optimization algorithm to ensure the problem is still solved and the expected post-engagement value is still minimized by the remaining weapons. This scenario considers five weapons, three of which are physical robots and two of which are simulated robots, to demonstrate practical usefulness on hardware. The weapons are tasked with engaging four targets. Similar to the simulated scenarios, all weapons (both simulated and hardware) communicate through a local network facilitated through ROS.

Table 3 shows the parameters used during the attrition scenario. The dual update frequency was reduced compared to other experiments with dual updates occurring every $2B = 1.0$ seconds as it removed some oscillations at the beginning of the scenario. For demonstration purposes, the weapon to be attrited (Weapon 2) is very effective against the high-valued Target 2.

Weapon 2 is attrited approximately 15 seconds after initialization by manually stopping its optimization algorithm and halting any communication from it. The other weapons recognize that a weapon has been attrited when they have not received any communications from that weapon after a predetermined threshold, in this case 5 seconds. At that point, the remaining weapons run Algorithm 1 from their previous solutions to compute new assignments with the attrited weapon being "assigned" to a null target (whose values and associated Pk values are zero). Figure 7 shows the weapons' changing assignments and the four targets' projected Pk values. In Figure 7a, the gray region denotes

Table 3 Parameters used for the attrition scenario

Parameter	Value
Pk	$\begin{bmatrix} 0.5 & 0.3 & 0.4 & 0.3 \\ 0.2 & 0.6 & 0.5 & 0.4 \\ 0.6 & 0.3 & 0.4 & 0.1 \\ 0.3 & 0.2 & 0.6 & 0.7 \\ 0.4 & 0.3 & 0.7 & 0.5 \end{bmatrix}$
$\mathbf{V}$	$\begin{bmatrix} 5 & 6 & 4 & 5 \end{bmatrix}$
Primal Communication Delay Lower Bound	0.1 seconds
Primal Communication Delay Upper Bound, B	0.5 seconds
δ	0.01
ρ	0.01
γ	0.1
α	0.1

where Weapon 2 is attrited. It takes the other weapons less than 10 seconds to recognize that Weapon 2 is attrited and to reach a new optimal assignment with Weapon 1 being reassigned. Target 2's projected Pk value is zero in Figure 7b for the time between attrition and Weapon 1's reassignment. The computation of new assignments can also be seen in Figure 8, which shows the time evolution of expected values for the attrition case. This figure demonstrates the benefit of allowing weapons to re-optimize: the new, final assignment has a lower expected value than that after attrition and before re-assignment.

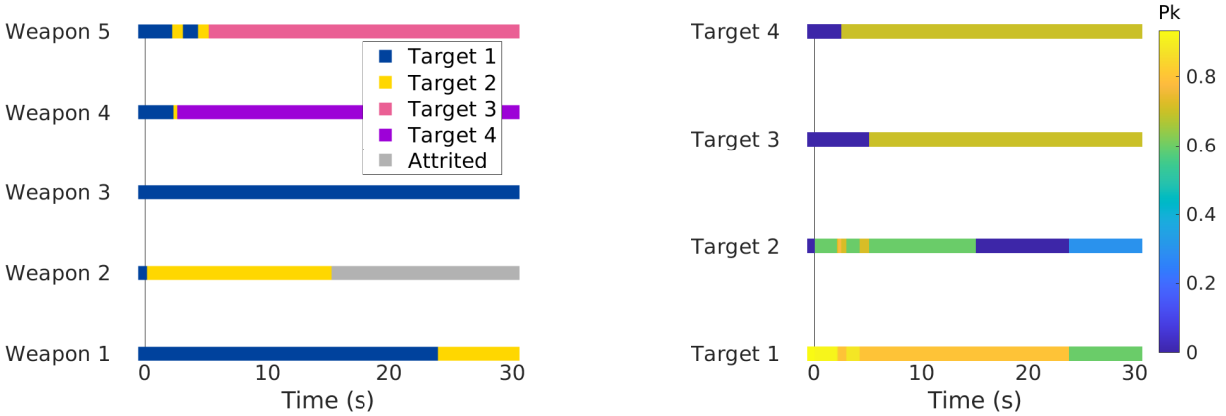


Fig. 7 Results for the attrition scenario: (a) The assignments for each of the five weapons over time. The gray region on Weapon 2's assignment indicates the period where the weapon was attrited. (b) The achieved Pk for the targets as a function of the weapons' assignments over time.

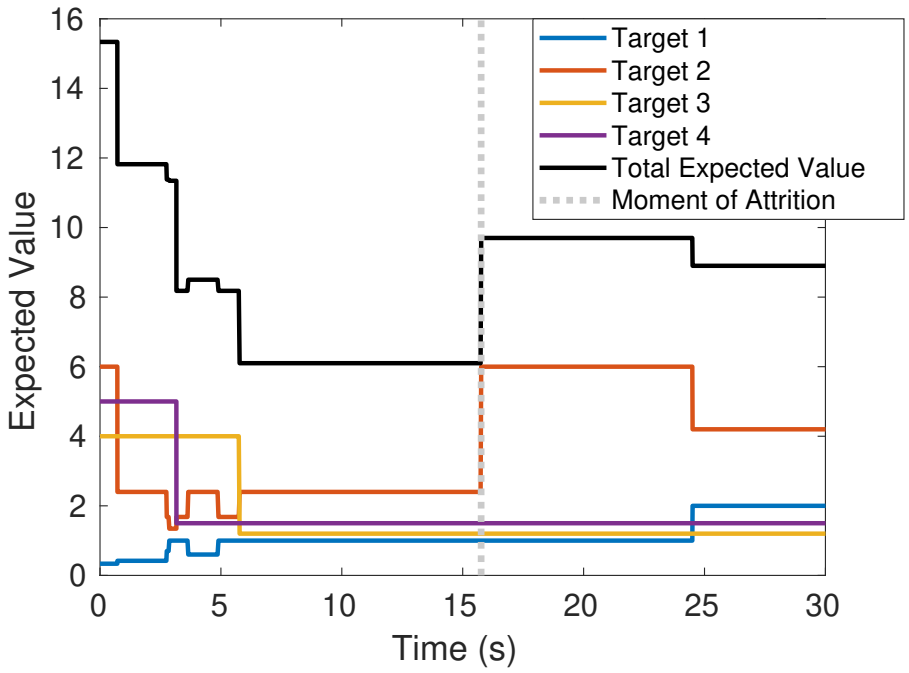


Fig. 8 The evolution of the expected value for each of the four targets in the attrition scenario, and the total expected value (black). The time of attrition is indicated by the dotted gray line.

VI. Conclusion

This paper presented a convex relaxation of the classical weapon-target assignment (WTA) problem, and provides a solution in the form of an efficient distributed approach which is robust to asynchronous computations and communications among workers. The theoretical results in this paper prove convergence of the proposed approach and formally guarantees its robustness to realistic delays in computing and communicating information, suggesting wide applicability for distributed and coordinated tasking efforts. Experimental results illustrated the performance and scalability of the proposed solution by considering three weapon-target assignment scenarios, including a larger number of workers and tasks as well as accounting for the loss of a worker during optimization. The experimental results show that this algorithm is robust to changing conditions in practice, namely attrition, which validates the ability of the proposed algorithm to leverage decentralization to adapt to changes and re-plan on the fly.

Appendix

Proof of Lemma 3: First the satisfaction of Assumptions A and B in [60] is shown, which enables the use of Proposition 2.2 from that same reference. Assumption A requires that (i) $L_{\boldsymbol{\kappa}}(\cdot, \boldsymbol{\mu}(tB))$ is bounded from below on X, (ii) the set X contains at least one point $\mathbf{y}$ such that $\mathbf{y} = \Pi_X[\mathbf{y} - \nabla_{\mathbf{x}} L_{\boldsymbol{\kappa}}(\mathbf{y}, \boldsymbol{\mu}(tB))]$, and (iii) $\nabla_{\mathbf{x}} L_{\boldsymbol{\kappa}}(\cdot, \boldsymbol{\mu}(tB))$ is Lipschitz on X. First, Remark 1 shows that ∇f is Lipschitz over X. Because g is affine, $L_{\boldsymbol{\kappa}}$ is affine in $\boldsymbol{\mu}$, and $\boldsymbol{\mu} \in \mathcal{M}$ which

is compact, $\nabla_{\mathbf{x}} L_{\boldsymbol{\kappa}}$ is also Lipschitz over X. Then Assumption A.iii is satisfied. For every choice of $\boldsymbol{\mu}(tB) \in \mathcal{M}$, the function $L_{\boldsymbol{\kappa}}(\cdot, \boldsymbol{\mu}(tB))$ is bounded from below because it is continuous on its compact domain X, and Assumption A.ii is satisfied. For every $t \in D$ and every fixed $\boldsymbol{\mu}(tB) \in \mathcal{M}$, the strong convexity of $L_{\boldsymbol{\kappa}}\big(\,\cdot\,, \boldsymbol{\mu}(tB)\big)$ implies that it has a unique minimum over X, denoted $\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(tB)$. This is the unique fixed point of the projected gradient descent mapping. Then Assumption A.i is satisfied, and hence all conditions of Assumption A in [60] are satisfied.

Assumption B in [60] is more technical and is not restated here. It requires that (i) the isocost curves of $L_{\boldsymbol{\kappa}}(\cdot, \boldsymbol{\mu}(tB))$ are separated, and (ii) the map $R(\mathbf{y}) = \mathbf{y} - \Pi_X[\mathbf{y} - \nabla_{\mathbf{x}} L_{\boldsymbol{\kappa}}(\mathbf{y}, \boldsymbol{\mu}(tB))]$ is Lipschitz at the origin. Both criteria are satisfied by functions that are strongly convex with Lipschitz gradients, and $L_{\boldsymbol{\kappa}}(\cdot, \boldsymbol{\mu}(tB))$ has both of these properties. Then Assumption B is satisfied as well. An application Proposition 2.2 in [60] completes the proof. ∎

Proof of Theorem 1: First note that the Lipschitz constant M of g may be upper-bounded by

$$M = \max_{\mathbf{x} \in X} \|\nabla g(\mathbf{x})\|_2 \leq \max_{\mathbf{x} \in X} \|\nabla g(\mathbf{x})\|_F = \max_{\mathbf{x} \in X} \sqrt{\sum_{i=1}^{m\ell} \sum_{j=1}^{m} |\nabla_i g_j(\mathbf{x})|^2} = \sqrt{m\ell}, \tag{6}$$

where the $\nabla g(\mathbf{x})$ matrix has m rows and $m\ell$ columns where each row is comprised of exactly ℓ entries that have the value 1 with the other entries being 0. Summing the squared absolute value of these entries gives the answer shown above.

Define $\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B) := \arg\min_{\mathbf{x} \in X} L_{\boldsymbol{\kappa}}(\mathbf{x}, \boldsymbol{\mu}(t_1 B))$ and recall $\hat{\mathbf{x}}_{\boldsymbol{\kappa}} = \arg\min_{\mathbf{x} \in X} L_{\boldsymbol{\kappa}}(\mathbf{x}, \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}})$. Expanding the dual update law and using the non-expansiveness of $\Pi_{\mathcal{M}}$ gives

$$\begin{aligned}
\|\boldsymbol{\mu}(t_2 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 &= \|\Pi_{\mathcal{M}}[\boldsymbol{\mu}(t_1 B) + \rho(g(\mathbf{x}(t_2 B)) - \delta \boldsymbol{\mu}(t_1 B))] - \Pi_{\mathcal{M}}[\hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}} + \rho(g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}) - \delta \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}})]\|^2 \\
&\leq \|\boldsymbol{\mu}(t_1 B) + \rho(g(\mathbf{x}(t_2 B)) - \delta \boldsymbol{\mu}(t_1 B)) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}} - \rho(g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}) - \delta \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}})\|^2 \\
&= \|(1 - \rho\delta)(\boldsymbol{\mu}(t_1 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}) - \rho(g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}) - g(\mathbf{x}(t_2 B)))\|^2 \\
&\leq (1 - \rho\delta)^2 \|\boldsymbol{\mu}(t_1 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 + \rho^2 \|g(\mathbf{x}(t_2 B)) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}})\|^2 \\
&\quad -2\rho(1 - \rho\delta)(\boldsymbol{\mu}(t_1 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}})^T (g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}) - g(\mathbf{x}(t_2 B))).
\end{aligned}$$

Adding $g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B)) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B))$ inside the last set of parentheses gives

$$\begin{aligned}
\|\boldsymbol{\mu}(t_2 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 \leq (1 - \rho\delta)^2 \|\boldsymbol{\mu}(t_1 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 + \rho^2 \|g(\mathbf{x}(t_2 B)) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}})\|^2 \\
-2\rho(1 - \rho\delta)(\boldsymbol{\mu}(t_1 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}})^T (g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B))) \\
-2\rho(1 - \rho\delta)(\boldsymbol{\mu}(t_1 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}})^T (g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B)) - g(\mathbf{x}(t_2 B))).
\end{aligned} \tag{7}$$

The application of Lemma 4 to the pairs $(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}, \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}})$ and $(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B), \boldsymbol{\mu}(t_1 B))$ gives

$$\begin{aligned}(\boldsymbol{\mu}(t_1 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}})^T (g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B))) &\geq \frac{\alpha}{M^2} \|g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B)) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}})\|^2 \\ &\geq \frac{\alpha}{m\ell} \|g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B)) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}})\|^2, \end{aligned} \tag{8}$$

where the last inequality applies Eq. (6) to bound M.

Applying Eq. (8) to the third term on the right-hand side of Eq. (7) gives

$$\begin{aligned}\|\boldsymbol{\mu}(t_2 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 \leq (1-\rho\delta)^2 \|\boldsymbol{\mu}(t_1 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 + \rho^2 \|g(\mathbf{x}(t_2 B)) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}})\|^2 \\ -2\rho(1-\rho\delta)\frac{\alpha}{m\ell} \|g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B)) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}})\|^2 \\ -2\rho(1-\rho\delta)(\boldsymbol{\mu}(t_1 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}})^T (g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B)) - g(\mathbf{x}(t_2 B))). \end{aligned} \tag{9}$$

To bound the last term, observe that $0 \leq \|(1-\rho\delta)(g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B)) - g(\mathbf{x}(t_2 B))) + \rho(\boldsymbol{\mu}(t_1 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}})\|^2$, which can be expanded and rearranged to give

$$\begin{aligned}-2\rho(1-\rho\delta)(\boldsymbol{\mu}(t_1 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}})^T (g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B)) - g(\mathbf{x}(t_2 B))) \\ \leq (1-\rho\delta)^2 \|g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B)) - g(\mathbf{x}(t_2 B))\|^2 + \rho^2 \|\boldsymbol{\mu}(t_1 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2. \end{aligned} \tag{10}$$

Applying the inequality in Eq. (10) to the last term in Eq. (9), one finds

$$\begin{aligned}\|\boldsymbol{\mu}(t_2 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 \leq (1-\rho\delta)^2 \|\boldsymbol{\mu}(t_1 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 + \rho^2 \|g(\mathbf{x}(t_2 B)) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}})\|^2 \\ - 2\rho(1-\rho\delta)\frac{\alpha}{m\ell} \|g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B)) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}})\|^2 \\ + (1-\rho\delta)^2 \|g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B)) - g(\mathbf{x}(t_2 B))\|^2 + \rho^2 \|\boldsymbol{\mu}(t_1 B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2. \end{aligned} \tag{11}$$

Next, add and subtract $g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B))$ inside the norm in the second term of Eq. (11) to rewrite it as

$$\begin{aligned}\rho^2 \|g(\mathbf{x}(t_2 B)) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}})\|^2 \leq \rho^2 \|g(\mathbf{x}(t_2 B)) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B))\|^2 + \rho^2 \|g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B)) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}})\|^2 \\ + 2\rho^2 \|g(\mathbf{x}(t_2 B)) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B))\| \|g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1 B)) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}})\|. \end{aligned}$$

Applying this to the second term in Eq. (11) and grouping terms gives

$$\begin{aligned}\|\boldsymbol{\mu}(t_2B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 \leq ((1-\rho\delta)^2+\rho^2)\|\boldsymbol{\mu}(t_1B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 + \left(\rho^2 - 2\rho(1-\rho\delta)\frac{\alpha}{m\ell}\right)\|g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1B)) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}})\|^2 \\ + ((1-\rho\delta)^2+\rho^2)\|g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1B)) - g(\mathbf{x}(t_2B))\|^2 \\ + 2\rho^2\|g(\mathbf{x}(t_2B)) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1B))\|\|g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1B)) - g(\hat{\mathbf{x}}_{\boldsymbol{\kappa}})\|.\end{aligned}$$

By hypothesis $\rho < \frac{2\alpha}{m\ell+2\alpha\delta}$, which makes the second term negative. Dropping this negative term and applying the Lipschitz property of g gives the upper bound

$$\begin{aligned}\|\boldsymbol{\mu}(t_2B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 \leq ((1-\rho\delta)^2+\rho^2)\|\boldsymbol{\mu}(t_1B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 + ((1-\rho\delta)^2+\rho^2)m\ell\|\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1B) - \mathbf{x}(t_2B)\|^2 \\ + 2\rho^2 m\ell\|\mathbf{x}(t_2B) - \hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1B)\|\|\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1B) - \hat{\mathbf{x}}_{\boldsymbol{\kappa}}\|.\end{aligned}$$

Lemma 3 may be applied to $\|\hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1B) - \mathbf{x}(t_2B)\|^2$ to give

$$\|\mathbf{x}(t_2B) - \hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1B)\|^2 \leq q_p^{2(t_2-t_1)}\|\mathbf{x}(t_1B) - \hat{\mathbf{x}}_{\boldsymbol{\kappa}}(t_1B)\|^2,$$

where $q_p = (1-c\gamma) \in [0,1)$. Next, the maximum distance between any two primal variables is given by $\max_{\mathbf{x},\mathbf{y}\in X}\|\mathbf{x}-\mathbf{y}\| = \sqrt{m\ell}$. Using this maximum distance and setting $q_d = ((1-\rho\delta)^2+\rho^2)$, it follows that $q_d \in [0,1)$ because $\rho < \frac{2\delta}{1+\delta^2}$. This gives the final result

$$\|\boldsymbol{\mu}(t_2B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 \leq q_d\|\boldsymbol{\mu}(t_1B) - \hat{\boldsymbol{\mu}}_{\boldsymbol{\kappa}}\|^2 + q_d q_p^{2(t_2-t_1)}m^2\ell^2 + 2\rho^2 q_p^{t_2-t_1}m^2\ell^2. \qquad \blacksquare$$

Funding Sources

This research was supported by the Air Force Research Laboratory Munitions Directorate under Task Orders FA8651-08-D-0108/048 and FA8651-20-F-1025, by the Air Force Office of Scientific Research (AFOSR) under grant no. FA9550-19-1-0169, and by the Office of Naval Research (ONR) under grants N00014-19-1-2543 and N00014-21-1-2495.

References

[1] Manne, A. S., "A Target-Assignment Problem," *Operations Research*, Vol. 6, No. 3, 1958, pp. 346–351. URL `http://www.jstor.org/stable/167023`.

[2] Matlin, S., "A Review of the Literature on the Missile-Allocation Problem," *Operations Research*, Vol. 18, No. 2, 1970, pp. 334–373. URL `http://www.jstor.org/stable/168691`.

[3] Lu, Y., and Chen, D. Z., "A new exact algorithm for the Weapon-Target Assignment problem," *Omega*, Vol. 98, 2021, p. 102138. https://doi.org/10.1016/j.omega.2019.102138.

[4] Tokgöz, A., and Bulkan, S., "Weapon Target Assignment with Combinatorial Optimization Techniques," *International Journal of Advanced Research in Artificial Intelligence*, Vol. 2, No. 7, 2013. https://doi.org/10.14569/IJARAI.2013.020707, URL http://dx.doi.org/10.14569/IJARAI.2013.020707.

[5] Huaiping, C., Jingxu, L., Yingwu, C., and Hao, W., "Survey of the research on dynamic weapon-target assignment problem," *Journal of Systems Engineering and Electronics*, Vol. 17, No. 3, 2006, pp. 559–565. https://doi.org/10.1016/S1004-4132(06)60097-2.

[6] Naidoo, S., "Applying Military Techniques used in Threat Evaluation and Weapon Assignment to Resource Allocation for Emergency Response-A Literature Survey," *Presented as seminar for Operations Research Society of South Africa*, 2008. URL http://www.orssa.org.za/wiki/uploads/Johannesburg/Naidoo2008.pdf.

[7] Cetin, E., and Esen, S. T., "A Weapon–Target Assignment Approach to Media Allocation," *Applied Mathematics and Computation*, Vol. 175, No. 2, 2006, pp. 1266–1275. https://doi.org/https://doi.org/10.1016/j.amc.2005.08.041.

[8] Lloyd, S. P., and Witsenhausen, H. S., "Weapons allocation is NP-complete," *1986 Summer Computer Simulation Conference*, 1986, pp. 1054–1058.

[9] Ahuja, R. K., Kumar, A., Jha, K. C., and Orlin, J. B., "Exact and heuristic algorithms for the weapon-target assignment problem," *Operations Research*, Vol. 55, No. 6, 2007, pp. 1136–1146. https://doi.org/10.1287/opre.1070.0440.

[10] Xin, B., Chen, J., Peng, Z., Dou, L., and Zhang, J., "An efficient rule-based constructive heuristic to solve dynamic weapon-target assignment problem," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, Vol. 41, No. 3, 2010, pp. 598–606. https://doi.org/10.1109/TSMCA.2010.2089511.

[11] Madni, A. M., and Andrecut, M., "Efficient heuristic approach to the weapon-target assignment problem," *Journal of Aerospace Computing, Information, and Communication*, Vol. 6, No. 6, 2009, pp. 405–414. https://doi.org/10.2514/1.34254.

[12] Murphey, R. A., "An approximate algorithm for a weapon target assignment stochastic program," *Approximation and Complexity in Numerical Optimization*, Springer, 2000, pp. 406–421. https://doi.org/10.1007/978-1-4757-3145-3_24.

[13] Shalumov, V., and Shima, T., "Weapon–target-allocation strategies in multiagent target–missile–defender engagement," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 10, 2017, pp. 2452–2464. https://doi.org/10.2514/1.G002598.

[14] Chandler, P., Pachter, M., Rasmussen, S., and Schumacher, C., "Multiple task assignment for a UAV team," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2002, p. 4587. https://doi.org/10.2514/6.2002-4587.

[15] Kalyanam, K., Casbeer, D., and Pachter, M., "Monotone optimal threshold feedback policy for sequential weapon target assignment," *Journal of Aerospace Information Systems*, Vol. 14, No. 1, 2017, pp. 68–72. https://doi.org/10.2514/1.I010501.

[16] Volle, K., and Rogers, J., "Weapon–target assignment algorithm for simultaneous and sequenced arrival," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 11, 2018, pp. 2361–2373. https://doi.org/10.2514/1.G003515.

[17] Kline, A., Ahner, D., and Hill, R., "The Weapon-Target Assignment Problem," *Computers and Operations Research*, Vol. 105, 2019, pp. 226–236. https://doi.org/10.1016/j.cor.2018.10.015.

[18] Kim, J., Lee, W.-C., Cho, D.-H., Song, J., and Choi, H.-L., "Decentralized weapon target assignment against high-speed enemy missiles," *AIAA Scitech 2020 Forum*, 2020, p. 0657. https://doi.org/10.2514/6.2020-0657.

[19] Alighanbari, M., and How, J. P., "Decentralized task assignment for unmanned aerial vehicles," *Proceedings of the 44th IEEE Conference on Decision and Control*, IEEE, 2005, pp. 5668–5673. https://doi.org/10.1109/CDC.2005.1583066.

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

[21] Chapman, A. C., Micillo, R. A., Kota, R., and Jennings, N. R., "Decentralised dynamic task allocation: a practical game: theoretic approach," *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 2009, pp. 915–922. URL 10.5555/1558109.1558139.

[22] Zhang, K., Zhou, D., Yang, Z., Pan, Q., and Kong, W., "Constrained multi-objective weapon target assignment for area targets by efficient evolutionary algorithm," *IEEE Access*, Vol. 7, 2019, pp. 176339–176360. https://doi.org/10.1109/ACCESS.2019.2955482.

[23] Arslan, G., Marden, J. R., and Shamma, J. S., "Autonomous vehicle-target assignment: A game-theoretical formulation," *American Society of Mechanical Engineers Journal of Dynamic Systems, Measurement, and Control*, Vol. 129, No. 5, 2007, pp. 584–596. https://doi.org/10.1115/1.2766722.

[24] Yanxia, W., Longjun, Q., Zhi, G., and Lifeng, M., "Weapon target assignment problem satisfying expected damage probabilities based on ant colony algorithm," *Journal of Systems Engineering and Electronics*, Vol. 19, No. 5, 2008, pp. 939–944. https://doi.org/10.1016/S1004-4132(08)60179-6.

[25] Kline, A. G., Ahner, D. K., and Lunday, B. J., "Real-time heuristic algorithms for the static weapon target assignment problem," *Journal of Heuristics*, Vol. 25, No. 3, 2019, pp. 377–397. https://doi.org/10.1007/s10732-018-9401-1.

[26] Cho, G., "Hybrid Nested Partitions Method with Intelligent Greedy Search for Solving Weapon-Target Assignment Problem," Master's thesis, Iowa State University, 2009. https://doi.org/10.31274/etd-180810-1053.

[27] Lee, D., Shin, M. K., and Choi, H.-L., "Weapon target assignment problem with interference constraints," *AIAA Scitech 2020 Forum*, 2020, p. 0388. https://doi.org/10.2514/6.2020-0388.

[28] Bogdanowicz, Z., Coleman, N., et al., "Sensor-target and weapon-target pairings based on auction algorithm," *Proceedings of the 11th WSEAS International Conference on Applied Mathematics*, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, WI, 2007, pp. 92–96. https://doi.org/10.1.1.527.5082.

[29] Sinha, A., Kumar, S. R., and Mukherjee, D., "Three-dimensional nonlinear cooperative salvo using event-triggered strategy," *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 2, 2021, pp. 328–342. https://doi.org/10.2514/1.G005367.

[30] Rodríguez-Seda, E. J., and Dawkins, J. J., "Decentralized cooperative collision avoidance control for unmanned rotorcraft with bounded acceleration," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 11, 2018, pp. 2445–2454. https://doi.org/10.2514/1.G003430.

[31] Bertsekas, D. P., and Tsitsiklis, J. N., *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, Inc., USA, 1989, pp. 425–569.

[32] Bertsekas, D. P., and Tsitsiklis, J. N., "Some Aspects of the Parallel and Distributed Iterative Algorithms — a Survey," *Automatica*, Vol. 27, No. 1, 1991, pp. 3–21. https://doi.org/10.1016/0005-1098(91)90003-K.

[33] Tsitsiklis, J., Bertsekas, D., and Athans, M., "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, Vol. 31, No. 9, 1986, pp. 803–812. https://doi.org/10.1109/TAC.1986.1104412.

[34] Nedic, A., and Ozdaglar, A., "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, Vol. 54, No. 1, 2009, pp. 48–61. https://doi.org/10.1109/TAC.2008.2009515.

[35] Nedic, A., Ozdaglar, A., and Parrilo, P. A., "Constrained Consensus and Optimization in Multi-Agent Networks," *IEEE Transactions on Automatic Control*, Vol. 55, No. 4, 2010, pp. 922–938. https://doi.org/10.1109/TAC.2010.2041686.

[36] Nedić, A., and Olshevsky, A., "Distributed Optimization Over Time-Varying Directed Graphs," *IEEE Transactions on Automatic Control*, Vol. 60, No. 3, 2015, pp. 601–615. https://doi.org/10.1109/TAC.2014.2364096.

[37] Zhu, M., and Martínez, S., "On distributed convex optimization under inequality and equality constraints," *IEEE Transactions on Automatic Control*, Vol. 57, No. 1, 2011, pp. 151–164. https://doi.org/10.1109/TAC.2011.2167817.

[38] Modi, P. J., Shen, W.-M., Tambe, M., and Yokoo, M., "Adopt: asynchronous distributed constraint optimization with quality guarantees," *Artificial Intelligence*, Vol. 161, No. 1, 2005, pp. 149–180. https://doi.org/10.1016/j.artint.2004.09.003.

[39] Notarnicola, I., and Notarstefano, G., "Asynchronous Distributed Optimization Via Randomized Dual Proximal Gradient," *IEEE Transactions on Automatic Control*, Vol. 62, No. 5, 2017, pp. 2095–2106. https://doi.org/10.1109/TAC.2016.2607023.

[40] Zhong, M., and Cassandras, C. G., "Asynchronous Distributed Optimization With Event-Driven Communication," *IEEE Transactions on Automatic Control*, Vol. 55, No. 12, 2010, pp. 2735–2750. https://doi.org/10.1109/TAC.2010.2049518.

[41] Iutzeler, F., Bianchi, P., Ciblat, P., and Hachem, W., "Asynchronous distributed optimization using a randomized alternating direction method of multipliers," *52nd IEEE Conference on Decision and Control*, 2013, pp. 3671–3676. https://doi.org/10.1109/CDC.2013.6760448.

[42] Hochhaus, S., and Hale, M. T., "Asynchronous Distributed Optimization with Heterogeneous Regularizations and Normalizations," *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 4232–4237. https://doi.org/10.1109/CDC.2018.8619015.

[43] Ubl, M., and Hale, M. T., "Totally Asynchronous Distributed Quadratic Programming with Independent Stepsizes and Regularizations," *58th Conference on Decision and Control (CDC)*, 2019, pp. 7423–7428. https://doi.org/10.1109/CDC40024.2019.9030251.

[44] Koshal, J., Nedić, A., and Shanbhag, U. V., "Multiuser optimization: Distributed algorithms and error analysis," *SIAM Journal on Optimization*, Vol. 21, No. 3, 2011, pp. 1046–1081. https://doi.org/10.1137/090770102.

[45] Hendrickson, K., and Hale, M., "Towards Totally Asynchronous Primal-Dual Convex Optimization in Blocks," *2020 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 3663–3668. https://doi.org/10.1109/CDC42340.2020.9304003.

[46] Hendrickson, K., and Hale, M., "Totally Asynchronous Primal-Dual Convex Optimization in Blocks," *arXiv preprint arXiv:2107.10338*, 2021. URL https://arxiv.org/abs/2107.10338.

[47] Boyd, S., and Vandenberghe, L., *Convex optimization*, Cambridge University Press, New York, 2004, pp. 104–112.

[48] Bertsekas, D., Nedic, A., and Ozdaglar, A., *Convex analysis and optimization*, Athena Scientific, Belmont, Massachusetts, 2003, Vol. 1, pp. 371–372.

[49] Kuhn, H. W., and Tucker, A. W., "Nonlinear programming," *Traces and emergence of nonlinear programming*, Springer, 2014, pp. 247–258.

[50] Arrow, K. J., Azawa, H., Hurwicz, L., and Uzawa, H., *Studies in linear and non-linear programming*, Stanford University Press, 1958, Vol. 2, Chap. 9.

[51] Hale, M. T., Nedić, A., and Egerstedt, M., "Cloud-based centralized/decentralized multi-agent optimization with communication delays," *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 700–705. https://doi.org/10.1109/CDC.2015.7402311.

[52] Hale, M. T., Nedić, A., and Egerstedt, M., "Asynchronous Multiagent Primal-Dual Optimization," *IEEE Transactions on Automatic Control*, Vol. 62, No. 9, 2017, pp. 4421–4435. https://doi.org/10.1109/TAC.2017.2662019.

[53] Ubl, M., and Hale, M., "Totally Asynchronous Large-Scale Quadratic Programming: Regularization, Convergence Rates, and Parameter Selection," *IEEE Transactions on Control of Network Systems*, Vol. 8, 2021, pp. 1465–1476. https://doi.org/10.1109/TCNS.2021.3068372.

[54] Facchinei, F., and Pang, J.-S., *Finite-dimensional variational inequalities and complementarity problems*, Springer Science & Business Media, New York, NY, 2007, pp. 1125–1135. https://doi.org/10.1007/b97543.

[55] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, Vol. 17, 2020, pp. 261–272. https://doi.org/10.1038/s41592-019-0686-2.

[56] Kraft, D., *A software package for sequential quadratic programming*, Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht, Wiss. Berichtswesen d. DFVLR, 1988. URL https://books.google.com/books?id=4rKaGwAACAAJ.

[57] Wächter, A., and Biegler, L. T., "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, Vol. 106, No. 1, 2006, pp. 25–57. https://doi.org/10.1007/s10107-004-0559-y.

[58] Hart, W. E., Watson, J.-P., and Woodruff, D. L., "Pyomo: modeling and solving mathematical programs in Python," *Mathematical Programming Computation*, Vol. 3, No. 3, 2011, pp. 219–260.

[59] Bynum, M. L., Hackebeil, G. A., Hart, W. E., Laird, C. D., Nicholson, B. L., Siirola, J. D., Watson, J.-P., and Woodruff, D. L., *Pyomo–optimization modeling in python*, 3rd ed., Vol. 67, Springer Science & Business Media, 2021.

[60] Tseng, P., "On the Rate of Convergence of a Partially Asynchronous Gradient Projection Algorithm," *SIAM Journal on Optimization*, Vol. 1, No. 4, 1991, pp. 603–619. https://doi.org/10.1137/0801036.

[61] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y., "ROS: an Open-Source Robot Operating System," *ICRA Workshop on Open Source Software*, Vol. 3.2, Kobe, Japan, 2009, p. 5.

[62] Koenig, N., and Howard, A., "Design and use paradigms for gazebo, an open-source multi-robot simulator," *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, Vol. 3, IEEE, 2004, pp. 2149–2154. https://doi.org/10.1109/IROS.2004.1389727.