Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Adversarial Task Assignment

Chen Hajaj and **Yevgeniy Vorobeychik**
Electrical Engineering and Computer Science
Vanderbilt University, Nashville, Tennses
{chen.hajaj, yevgeniy.vorobeychik}@vanderbilt.edu

Abstract

The problem of assigning tasks to workers is of long-standing fundamental importance. Examples of this include the classical problem of assigning computing tasks to nodes in a distributed computing environment, assigning jobs to robots, and crowdsourcing. Extensive research into this problem generally addresses important issues such as uncertainty and incentives. However, the problem of adversarial tampering with the task assignment process has not received as much attention.

We are concerned with a particular adversarial setting where an attacker may target a set of workers in order to prevent the tasks assigned to these workers from being completed. When all tasks are homogeneous, we provide an efficient algorithm for computing the optimal assignment. When tasks are heterogeneous, we show that the adversarial assignment problem is NP-Hard, and present an algorithm for solving it approximately. Our theoretical results are accompanied by extensive experiments showing the effectiveness of our algorithms.

1 Introduction

The problem of allocating a set of tasks among a collection of workers has been a fundamental research question in a broad array of domains, including distributed computing, robotics, and, recently, crowdsourcing [Alistarh *et al.*, 2012; Stone and Veloso, 1999; Liu and Chen, 2017]. Despite the extensive interest in the problem, however, there is little prior work on task assignment in settings where workers may be attacked. Such *adversarial task assignment* problems can arise, for example, when tasks are of high economic or political consequence, such as in robotic rescue missions following terror activities, or crowdsourcing to determine which executables are malicious or benign, or which news stories constitute fake news.

We investigate the adversarial task assignment problem in which a rational external attacker targets one or more workers after tasks have already been assigned. Equivalently, this can be viewed as a robust task assignment problem with unknown uncertainty about worker failures. We formalize the interaction between the attacker and requester (defender) as a Stackelberg game in which the defender first chooses an assignment, and the attacker subsequently attacks a set of workers so as to maximize the defender's losses from the attack. We seek a strong Stackelberg equilibrium (SSE) of this game and focus on computing an optimal robust assignment.

Our analysis begins with a setting in which tasks are homogeneous, that is, all tasks have the same utility for the defender (e.g., rescue soldiers from a battlefield, or label a large dataset of images). We characterize the optimal structure of a robust assignment, and use this insight to develop an algorithm that extracts this assignment in time linear in the number of tasks and targets, and quadratic in the number of workers. We show that this algorithm significantly outperforms several baselines, and obtains a good solution even when no adversary is present.

Next, we turn to heterogeneous task settings. This case, it turns out, is considerably more challenging. Specifically, we show that it may be beneficial to assign more than a single worker to a task. Moreover, even if we impose a restriction that only a single worker can be assigned to a task (optimal when tasks are homogeneous), extracting the optimal assignment is strongly NP-Hard. To overcome this issue, we propose an integer programming approach for solving the restricted problem, as well as an algorithm for finding an approximately optimal assignment in the general case. Again, our experiments show that our approach significantly outperforms several baselines.

Related Work The problem of task assignment in adversarial settings has been considered from several perspectives. One major stream of literature is about robots acting in adversarial environments. Alighanbari and How [2005] consider assigning weapons to targets, somewhat analogous to our problem, but do not model the decision of the adversary; their model also has rather different semantics than ours. Robotic soccer is another common adversarial planning problem, although the focus is typically on coordination among robots when two opposing teams are engaged in coordination and planning [Jones *et al.*, 2006].

Another major literature stream which considers adversarial issues is crowdsourcing. One class of problems is a number of workers to hire [Carvalho *et al.*, 2016], the issue of individual worker incentives in truthfully responding

to questions [Singla and Krause, 2013], or in the amount of effort they devote to the task [Tran-Thanh *et al.*, 2014; Elmalech *et al.*, 2016; Liu and Chen, 2017], rather than adversarial reasoning per se. Another, more directly adversarial setting, considers situations where some workers simply answer questions in an adversarial way [Ghosh *et al.*, 2011; Steinhardt *et al.*, 2016]. However, the primary interest in this work is robust estimation when tasks are assigned randomly or exogenously, rather than task assignment itself. Similarly, prior research on machine learning when a portion of data is adversarially poisoned [Chen *et al.*, 2011; Xu *et al.*, 2010; Feng *et al.*, 2014; Chen *et al.*, 2013; Liu *et al.*, 2017] focuses primarily on the robust estimation problem, and not task assignment; in addition, it does not take advantage of structure in the data acquisition process, where workers, rather than individual data points, are attacked. Other works [Gu *et al.*, 2005; Alon *et al.*, 2015] focus on the change of the system after the assignment process and the structure of the social network rather than the assignment process itself.

Our work has a strong connection to the literature on Stackelberg security games [Conitzer and Sandholm, 2006; Korzhyk *et al.*, 2010; Tambe, 2011]. However, the mathematical structure of our problem is quite different. For example, we have no protection resources to allocate, and instead, the defender's decision is about assigning tasks to potentially untrusted workers.

2 Model

Consider an environment populated with a single requester (hereafter denoted by "*defender*"), a set of n workers, W, a set of m tasks, T, and an *adversary*. Furthermore, each worker $w \in W$ is characterized by a capacity constraint c_w, which is the maximum number of tasks it can be assigned, and an individual proficiency or the probability of successfully completing a task, denoted by p_w. Worker proficiencies are assumed to be common knowledge to both the defender and attacker. Such proficiencies can be learned from experience [Sheng *et al.*, 2008; Dai *et al.*, 2011; Manino *et al.*, 2016]; moreover, in many settings, these are provided by the task assignment (e.g., crowdsourcing) platform, in the form of a reputation system [Mason and Suri, 2012].

For exposition purposes, we index the workers by integers i in decreasing order of their proficiency, so that $P = (p_1, \ldots, p_n)$ s.t. $p_i \geq p_j \ \forall i < j$, and denote the set of k most proficient workers by W^k. Thus, the capacity of worker i would be denoted by c_i. Each task $t \in T$ is associated with a utility u_t that the defender obtains if this task is completed successfully. If the task is not completed successfully, the defender obtains zero utility from it.

We focus on the common case where the defender faces a budget constraint of making at most $B \leq m$ assignments; the setting with $B > m$ necessitates different algorithmic techniques, and is left for future work. The defender's fundamental decision is the *assignment* of tasks to workers. Formally, an assignment s specifies a subset of tasks $T'(s)$ and the set of workers, $W_t(s)$ assigned to each task $t \in T'(s)$.

Suppose that multiple workers are assigned to a task t, and let $L_t(s)$ denote the labels returned by workers in $W_t(s)$ for t (for example, these could simply indicate whether a worker successfully complete the task). Then the defender determines the final label to assign to t (e.g., whether or not the task has been successfully completed) according to some deterministic mapping $\delta : L_t(s) \to l$ (e.g., majority label), such that $L \in \{1, \ldots, j_t\}^{|W_t(s)|}$ and $l \in \{1, \ldots, j_t\}$. Naturally, whenever a single worker w is a assigned to a task and returns a label l_w, $\delta(l_w) = l_w$. Let ι_t be the (unknown) correct label corresponding to a task t; this could be an actual label, such as the actual object in the image, or simply a constant 1 if we are only interested in successful completion of the task. The defender's expected utility when assigning a set of tasks $T'(s)$ to workers and obtaining the labels is then

$$u_{def}(s) = \sum_{t \in T'(s)} u_t \Pr\{\delta(L_t(s)) = \iota_t\}, \qquad (1)$$

where the probability is with respect to worker proficiencies (and resulting stochastic realizations of their outcomes).

It is immediate that in our setting if there is no adversary and no capacity constraints for the workers, all tasks should be assigned to the worker with the highest p_w. Our focus, however, is how to optimally assign workers to tasks when there is an intelligent adversary who may subsequently (to the assignment) attack a set of workers. In particular, we assume that there is an adversary (attacker) with the goal of minimizing the defender's utility u_{def}; thus, the game is zero-sum. To this end, the attacker chooses a set of τ workers to attack, for example, by deploying a cyber attack against the corresponding computer nodes, physical attacks on search and rescue robots, or attacks against the devices on which the human workers performs their tasks. Alternatively, our goal is to be robust to τ-worker failures (e.g., N-τ-robustness [Chen *et al.*, 2014]). We encode the attacker's strategy by a vector α where $\alpha_w = 1$ iff a worker w is attacked (and $\sum_w \alpha_w = \tau$ since τ workers are attacked). The adversary's attack takes place *after* the tasks have already been assigned to workers, where the attacker knows the actual assignments of tasks to workers before deploying the attack, and the consequence of an attack on a worker w is that all tasks assigned to w fail to be successfully completed.

Clearly, when an attacker is present, the policy of assigning all tasks to the most competent worker (when there are no capacity constraints) will yield zero utility for the defender, as the attacker will simply attack the worker to whom all the tasks are assigned. The challenge of how to split the tasks up among workers, trading off quality with robustness to attacks, is the subject of our inqury. Formally, we aim to compute a strong Stackelberg equilibrium of the game between the defender (leader), who chooses a task-to-worker assignment policy, and the attacker (follower), who attacks a single worker [Stackelberg, 1952].

3 Homogeneous Tasks

We start by considering tasks which are *homogeneous*, that is, $u_t = u_{t'}$ for any two tasks t, t'. Without loss of generality, suppose that all $u_t = 1$. Note that since all tasks share the same utility, if $B < m$, the defender is indifferent regarding

the identity of tasks being assigned. Further, it is immediate that we never wish to waste budget, since assigning a worker always results in non-negative marginal utility. Consequently, we can simply randomly subsample B tasks from the set of all tasks, and consider the problem with $m = B$.

We overload the notation and use $s = \{s_1, \ldots, s_n\}$ to denote the number of tasks allocated to each worker. Although the space of deterministic assignments is large, we now observe several properties of optimal assignments which allow us to devise an efficient algorithm for this problem.

Proposition 1. *Suppose that tasks are homogeneous. For any assignment s there is a weakly utility-improving assignment s' for the defender which assigns each task to a single worker.*

Proof. Consider an assignment s and the corresponding best response by the attacker, α, in which a worker $\bar{w}$ is attacked. Let a task $\bar{t}$ be assigned to a set of workers $W_{\bar{t}}$ with $|W_{\bar{t}}| = k > 2$. Then there must be another task t' which is unassigned. Now consider a worker $w \in W_{\bar{t}}$. Since utility is additive, we can consider just the marginal utility of any worker w' to the defender and attacker; denote this by $u_{w'}$. Let $T_{w'}$ be the set of tasks assigned to a worker w' under s. Let $u_w = \sum_{t \in T_w} u^M_{wt}$, where $u^M_{wt} = u_t \Pr\{\delta(L_t(s)) = \iota_t\} - u_t \Pr\{\delta(L_t(s) \setminus L^w_t) = \iota_t\}$ is the marginal utility of worker of w towards a task t. Clearly, $u_w \leq u_{\bar{w}}$, since the attacker is playing a best response.

Suppose that we reassign w from $\bar{t}$ to t'. If $w = \bar{w}$, the attacker will still attack w (since the utility of w to the attacker can only increase), and the defender is indifferent. If $w \neq \bar{w}$, there are two cases: (a) the attacker still attacks $\bar{w}$ after the change, and (b) the attacker now switches to attack w. Suppose the attacker still attacks $\bar{w}$. The defender's net gain is $p_w - u^M_{w\bar{t}} \geq 0$. If, instead, the attacker now attacks w, the defender's net gain is $u_{\bar{w}} - u_w \geq 0$. □

Consequently, we can restrict the set of assignments to those which assign a single worker per task; we denote this restricted set of assignments by S. Given a assignment $s \in S$ and the attack strategy α, the defender's expected utility is:

$$u_{def}(s, \alpha) = \sum_{w \in W} s_w p_w (1 - \alpha_w) \tag{2}$$

Next, we show that there is always an optimal assignment that assigns tasks to the k most proficient workers, for some k.

Proposition 2. *In an optimal assignment s, suppose that $s_i > 0$ for $i > 1$. Then there must be an optimal assignment in which $s_{i-1} > 0$.*

Proof. Consider an optimal assignment s and the attacker's best response α in which $\bar{W}$ is the set of workers being attacked. Now, consider moving 1 task from i to $i-1$. We denote the updated set of workers attacked (due to this change) as $\bar{W}'$. Suppose that $i \in \bar{W}$, that is, the worker i was initially attacked. If $i - 1 \in \bar{W}'$, there are two potions: 1) $i \in \bar{W}'$ (i.e., i is still being attacked) and hence the net gain to the defender does not change, and 2) $i \notin \bar{W}'$ and hence the net gain to the defender is $p_i(|T_i| - 1) \geq 0$. If $i - 1 \notin \bar{W}'$, the net gain is $p_{i-1} > 0$. Suppose that $i \notin \bar{W}$. If $i - 1$ is now attacked, the net gain is $p_w(|T_w| - 1) \geq 0$ (where $w \in \bar{W}$ and $w \notin \bar{W}'$). Otherwise (i.e., $i - 1 \notin \bar{W}'$), the net gain is $p_{i-1} - p_i \geq 0$. □

We can now present an assignment algorithm for optimal assignment (Algorithm 1) which has complexity $O(n^2 m\tau)$. The intuition behind the algorithm is to consider each worker i as a potential target of an attack, and then compute the best assignment subject to a constraint that i is attacked (i.e., that $p_i s_i \geq p_j s_j$ for all other workers $j \neq i$). Subject to this constraint, we consider all possible numbers of tasks that can be assigned to i, and then assign as many tasks as possible to the other workers in order of their proficiency (where the τ workers that contribute the most to the defender's utility are attacked). The only special case (Steps 7-10) is when assigning the last worker. In this case, it may be beneficial to alternate the last two workers' assignments to result in a more beneficial overall assignment. Optimality follows from the fact that we exhaustively search possible targets and allocation policies to these, and assign as many tasks as possible to the most effective workers.[1]

Algorithm 1 Homogeneous assignment

input: The set of workers W, and their proficiencies P
return: The optimal policy s^*

1: $u_{max} \leftarrow 0$
2: **for** $i \in \{1, \ldots, n\}$ **do**
3: **for** $s_i \in \{1, \ldots, c_i\}$ **do**
4: $\Upsilon_i \leftarrow s_i p_i, B \leftarrow m - s_i$
5: **for** $j \in \{1, \ldots, n\} \setminus i$ **do**
6: $s_j \leftarrow \min(\left\lfloor \frac{p_i}{p_j} s_i \right\rfloor, B, c_j), B \leftarrow B - s_j$
7: **if** $j < n \wedge B + 1 \leq \min(\left\lfloor \frac{p_i}{p_{j+1}} s_i \right\rfloor - 1, c_{j+1})$ **then**
8: $s' \leftarrow s, s'_j \leftarrow s_j - 1$
9: **if** $u_{def}(s, \alpha) \leq u_{def}(s', \alpha') + p_{j+1}$ **then**
10: $s_j \leftarrow s_j - 1, B \leftarrow B + 1$
11: $\Upsilon_j \leftarrow s_j p_j$
12: Sort Υ in ascending order
13: $util \leftarrow \sum_{k=1}^{n-\tau} \Upsilon_k$
14: **if** $util > u_{max}$ **then**
15: $u_{max} \leftarrow util, s^* \leftarrow s$
16: **return** s^*

4 Heterogeneous Tasks

It turns out that the more general problem in which utilities are heterogeneous is considerably more challenging than the case of homogeneous allocation. First, we show that even if the tasks' utilities are slightly different, it may be beneficial to assign the same task to multiple workers. Consider the case of an environment populated with 2 workers and 2 tasks. WLOG, we order the tasks by their utility, i.e., $u_{t_1} > u_{t_2}$. Regardless of the workers' proficiencies, assigning one worker per task will result in an expected utility of $\min(p_i u_{t_1}, p_j u_{t_2})$.

[1] A detailed proof of Algorithm 1's optimality is available at: https://arxiv.org/abs/1804.11221

On the other hand, assigning both workers to t_1 will result in an expected utility of $\min(p_i u_{t_1}, p_j u_{t_1})$ which is promised to be equal or higher. Aside from the considerably greater complexity challenge associated with solving problems with heterogeneous utilities suggested by this example, there is the additional challenge of incorporating (non-linear) decision rules into the optimization problem to resolving disagreement among workers, should it arise.

We begin by showing that if $B \leq m$, there is an optimal assignment in which only the B tasks associated with the highest utility are included.

Proposition 3. *Suppose that tasks are heterogeneous. For any assignment s there is a weakly utility-improving (i.e., results in the same or higher utility) assignment s' for the defender which only assigns tasks from the set of tasks with the B highest utilities.*

Proof. For readability, we assume that tasks are ordered based on their utility in decreasing order (i.e., $u_i \geq u_j, \forall i \leq j$), and that a single worker is assigned per task; generalization is straightforward. Consider an assignment s and the corresponding best response by the attacker, α, in which the set of workers $\bar{W}$ is attacked. Let a task t_i be s.t. $i > B$. Then there must be another task t_j, s.t. $j \leq B$, which is unassigned. Now consider a worker $w \in W_{t_i}$. Since utility is additive, we can consider just the marginal utility of any worker w' to the defender and attacker; denote this by $u_{w'}$. Let $T_{w'}$ be the set of tasks assigned to a worker w' under s. Let $u_w = \sum_{t \in T_w} u^M_{wt}$, where u^M_{wt} is the marginal utility of worker of w towards a task t.

Suppose that we reassign w from t_i to t_j. If $w \in \bar{W}$, the attacker will still attack w (since the utility of w to the attacker can only increase), and the defender is indifferent. If $w \notin \bar{w}$, there are two cases: (a) the attacker still attacks $\bar{W}$ after the change, and (b) the attacker now switches to attack w. Suppose the attacker still attacks $\bar{W}$. The defender's net gain is $p_w u_j - u^M_{w\bar{t}} \geq 0$. If, instead, the attacker now attacks w, the defender's net gain is $u_{w'} - u_w \geq 0$. Where w' is the worker that is not being attacked anymore. □

This allows us to restrict attention to the B highest-utility tasks, and assume that $m = B$.

We now show that the defender's assignment problem, denoted *Heterogeneous tasks assignment (HTA)*, is NP-Hard even if we restrict the strategies to assign only a single worker per task.

Proposition 4. *HTA is strongly NP-Hard even when we assign only one worker per task.*

Proof. We prove the proposition by reducing the decision version of the *Bin packing problem* (BP), which is a strongly NP-complete problem, to the decision version of the HTA problem. In the BP problem we are given a set $\{o_1, o_2, ..., o_m\}$ of m objects of sizes $\{v_1, v_2, ..., v_m\}$ and a set of n containers $\{C_1, C_2, ..., C_n\}$, each of size γ, and we need to decide if all the objects can be fitted into the given containers. Our transformation maps the set of m objects to a set of $m+1$ tasks $T = \{t_1, t_2, ..., t_{m+1}\}$ with utilities $\{v_1, v_2, ..., v_m, \gamma\}$ and the set of n containers to a set of $n+1$ workers $W = \{w_1, w_2, ..., w_{n+1}\}$. We consider the private case where all the workers have the same proficiency p (i.e. $p_w = p, \forall w \in W$). The decision version of the HTA problem asks if there exists an assignment of the $m+1$ tasks to the $n+1$ workers that achieves a utility of at least pV, where $V = \sum_{i=1}^{m} v_i$.

If we started with a YES instance of the BP problem, then there exists an assignment $\mathcal{A}$ that fits all m objects into the n containers. Consider the following assignment of tasks to workers in the HTA problem. If $\mathcal{A}(o_i) = C_j$, we assign task t_i to worker w_j. Also, we assign task t_{m+1} (with utility γ) to worker w_{n+1}. Note that no worker can achieve an individual utility greater than $p\gamma$, which is achieved by worker w_{n+1}. Thus, the utility of the overall task assignment is $\sum_{i=1}^{m} pv_i + p\gamma - p\gamma = pV$, meaning that our transformation produced a YES instance of the HTA problem.

Now suppose that we ended up with a YES instance of the HTA problem. Then there exists a task assignment $\mathcal{B}$ such that the sum of utilities (V^*) minus the adversarial harm (γ^*) is at least pV (i.e. $V^* - \gamma^* \geq pV$). Note that $V^* = \sum i = 1^m pv_i + p\gamma = pV + p\gamma$ (each task is assigned to some worker). This implies $pV + p\gamma - \gamma^* \geq pV$ and $\gamma^*/p \leq \gamma$. Thus the utility sum (before performance p is applied) of the tasks assigned to any single worker cannot exceed γ. This could only happen if task t_{m+1} (with utility γ) was the only task assigned to the corresponding player. WLOG let that worker be w_{n+1}. All other tasks must have been assigned to workers $\{w_1, w_2, ..., w_n\}$. It is easy to see that this implies a feasible assignment of objects to containers in the BP problem - if $\mathcal{B}(t_j) = w_i$, for $1 \leq j \leq m$, then we place object o_j in container C_i. Thus the transformation must have started off with a YES instance of the BP problem. □

We now propose an algorithm which computes an approximately optimal assignment. We begin by supposing that only one worker can be assigned per task (we relax this shortly). In this case, the optimal attack can be computed using the following linear integer program:

$$\max_{\alpha} \sum_{w \in W} \alpha_w \sum_{t \in T} s_{wt} u_t p_w \tag{3a}$$

$$s.t. : \sum_{w \in W} \alpha_w = \tau \tag{3b}$$

$$\alpha_w \in \{0, 1\}. \tag{3c}$$

The objective (3a) aims to maximize the effect of the attack (i.e., the utility of the targets). Constraint (3b) ensures that the adversary attacks exactly τ workers. First, note that the extreme points of the constraint set are integral, which means we can relax the integrality constraint to $\alpha_w \in [0, 1]$. In order to plug this optimization into the defender's optimal assignment problem, we convert this relaxed program to its dual form:

$$\min_{\lambda, \beta} \lambda\tau + \sum_{w} \beta_w \tag{4a}$$

$$s.t. : \lambda + \beta_w \geq p_w \sum_{t \in T} s_{wt} u_t \; \forall \; w \tag{4b}$$

$$\beta \geq 0. \tag{4c}$$

Thus, the optimal assignment can be computed using the following linear integer program:

$$\max_{s,\gamma,\lambda,\beta} \sum_{w\in W} p_w \sum_{t\in T} s_{wt} u_t - \gamma \tag{5a}$$

$$s.t. : \gamma \geq \lambda\tau + \sum_{w} \beta_w \tag{5b}$$

$$\lambda + \beta_w \geq \sum_{t\in T} s_{wt} u_t p_w, \forall w \in W \tag{5c}$$

$$\sum_{w\in W} \sum_{t\in T} s_{wt} = m \tag{5d}$$

$$\sum_{w} s_{wt} = 1, \forall t \in T \tag{5e}$$

$$\sum_{t} s_{wt} \leq c_w, \forall w \in W \tag{5f}$$

$$s_{wt} \in \{0, 1\}. \tag{5g}$$

The objective (5a) aims to maximize the defender's expected utility given the adversary's attack (second term). Constraint (5b and 5c) validates that the adversary's targets are the workers who contribute the most to the defender's expected utility and Constraint (5d) ensures that each allocation assigns all the possible tasks among the different workers. Finally, Constraint (5e) ensures that only one worker is assigned for each task and Constraint (5f) ensures that no worker is assigned with more tasks than it can perform.

Next, we propose a greedy algorithm that attempts to incrementally improve utility by shifting workers among tasks, now allowing multiple workers to be assigned to a task. Whenever more than one worker is assigned to a given task, the defender has to choose a deterministic mapping δ to determine the outcome. We consider a very broad class of *weighted majority* functions for this purpose (natural if successful completion of a task means that the worker returned the correct label). In this mapping, each worker w is assigned a weight θ_w, and the final label is set according to the weighted majority rule, i.e., $\delta(L_t) = \text{sgn}(\sum_{w\in W_t(s)} \theta_w l_w)$.

In order to approximate the defender's expected utility, we use the sample average approximation (SAA) [Kleywegt *et al.*, 2002] for solving stochastic optimization problems by using Monte-Carlo simulation. Using this approach, the defender's utility can be approximated by:

$$u_{def}(C_K, W') = \sum_{t\in T} u_t \left(\sum_{k=1}^{K} \frac{\mathbb{I}\{\text{sgn}\sum_{w\in W'} s_{wt}\theta_w C_{wtk}\}}{K} \right) \tag{6}$$

where C_K is a set of K matrices, each of size n over m. Each cell C_{wtk} is a randomly sample based on p_w represents whether or not the worker w successfully completed the task. That is, $C_{wtk} = 1$ if worker w successfully completed task t, and $C_{wtk} = 0$ otherwise. In a similar manner, $s_{wt} = 1$ if worker w is assigned to task t, and $s_{wt} = 0$ otherwise.

Algorithm 2 formally describes the computation of this assignment. Given an optimal assignment extracted using the mixed-integer linear program in Equation (5), we iteratively alternate over all tasks in ascending order based on their utility. For each task, we reassign the worker associated with this task to the most beneficial task. If this reassignment improves the defender's utility, we label it as beneficial (Steps 9 and 10). Finally, we commit to the reassignment that will maximize the defender's utility (Step 12).

Algorithm 2 Heterogeneous assignment

input: The set of workers W, and their proficiencies P
return: The heuristic deterministic allocation

1: Extract the optimal 1-worker allocation using Equation 5
2: $util \leftarrow u_{def}(C_K, \alpha)$
3: **for** $t \in \{1, \ldots, m\}$ **do**
4: **for** $w \in \{1, \ldots, n\}$ **do**
5: $\bar{t} = t$
6: **if** $s_{wt} = 1$ **then**
7: **for** $t' \in \{m, \ldots, t+1\}$ **do**
8: $s_{wt'} = 1$, $s_{wt} = 0$, Update α
9: **if** $u_{def}(C_K, \alpha) > util$ **then**
10: $\bar{t} = t'$, $util \leftarrow u_{def}(C_K, \alpha)$
11: $s_{wt'} = 0$, $s_{wt} = 1$
12: $s_{wt} = 0$, $s_{w\bar{t}} = 1$
13: **return** s

5 Experiments

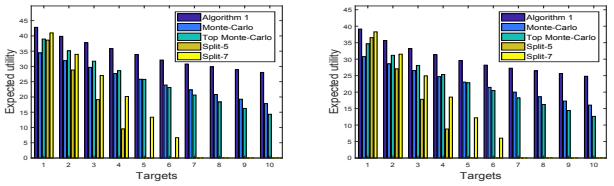
We now experimentally demonstrate the effectiveness of our proposed approaches. Workers' proficiencies are sampled using two distributions: a uniform distribution over the $[0.5, 1]$ interval and an exponential distribution with $\mu = 0.25$ where proficiencies are truncated to be in this interval for the latter. We compare our adversarial assignment algorithms to three natural baselines: *Split-k* and two versions of *Monte-Carlo* (involving random assignment of tasks to workers). Specifically, for the Split-k method, we divide tasks equally among the top k workers.[2] For the Monte-Carlo approach, we consider a simple variant which randomly distributes tasks among all the workers, denoted by *Monte-Carlo*, and a variant of this which randomly distributes the tasks among the top $\lceil \frac{n}{2} \rceil$ workers, denoted by *Top Monte-Carlo*. In both cases, the assigned worker for each task is picked uniformly at random.

Homogeneous Tasks We begin by considering homogeneous tasks. For each experiment, we take an average of 5,000 sample runs.

Figure 1 presents the results comparing our algorithm to baselines for 50 workers and tasks. As the figure shows, our algorithm outperforms the baselines, and the gap becomes particularly pronounced as the number of targets increases. Moreover, there doesn't appear to be a qualitative difference between uniform and exponential distribution in this regard.

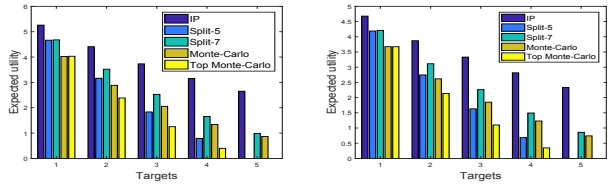
It is natural that we must trade off robustness with performance of robust algorithms in non-adversarial settings. We therefore conclude the homogeneous analysis by analyzing the loss incurred by allowing for robustness, compared to a solution which is optimal in non-adversarial settings. We vary

[2]The remainder is assigned in an iterative way from the least proficient worker to the most proficient one.



(a) Uniform distribution (b) Exponential distribution

Figure 1: Homogeneous tasks: comparison to baseline methods.



(a) Uniform distribution (b) Exponential distribution

Figure 2: Heterogeneous tasks: comparison to baseline methods.

the number of workers from 2 to 50, and fix the number of tasks at 100 and the number of targets optimized against at $t = 1$.

Workers	5	10	15	20	25	30	35	40	45	50
Exp. loss	24.9%	17.4%	15.27%	13.2%	11.6%	8.6%	5.8%	5.8%	6.5%	4.6%

Table 1: Expected loss of using adversarial assignment in non-adversarial settings.

Table 1 shows the expected loss of using adversarial task assignment in a non-adversarial settings. With only 5 workers, we pay a steep price (just under 25%), but as the number of workers increases, the loss shrinks; with 50 workers, we only lose 4.6% compared to optimal non-robust assignment.

Heterogeneous Tasks We used CPLEX version 12.51 to solve the integer linear program above.

First, we analyze how the heterogeneous assignment given in mixed-integer linear program (MILP) (5) performs compared to the baselines when task utilities are sampled from $U[0, 1]$ and worker proficiencies are samples from $U[0.5, 1]$. We use similar baseline methods to the ones used in studying homogeneous task assignment.

Figure 2 depicts the expected utility for the defender when using each of the methods in an environment populated with 15 tasks and 10 workers where the number of targets the adversary attacks varies between 1 and 5 over $3,000$ runs. As is evident from the figure, even the baseline mixed-integer linear program (which assumes a single worker is assigned per task) significantly outperforms the baselines, with the difference growing as we increase the number of workers attacked.

Next, we evaluate how much more we gain by using Algorithm 2 after computing an initial assignment using MILP (5). In these experimets we use a natural weighted majority decision rule with $\theta_w = p_w$ (i.e., workers' proficiencies), and set $K = 2500$. We consider two uniform distributions for this study: $U[0, 1]$ and $U[0, 100]$. Each marginal improvement is averaged over 3,000 runs.

Dist.	Tasks	n=2	n=3	n=4	n=5	n=6
U[0,1]	3	57.08%	37.10%			
U[0,1]	4	26.47%	9.88%	9.17%		
U[0,1]	5	22.03%	3.83%	3.39%	3.46%	
U[0,1]	6	19.98%	2%	1.79%	1.93%	1.66%
U[0,100]	3	56.9%	37.92%			
U[0,100]	4	28.69%	9.59%	8.86%		
U[0,100]	5	20.02%	3.59%	3.51%	3.49%	
U[0,100]	6	17.41%	1.59%	1.71%	1.64%	1.77%

Table 2: Average improvement using Algorithm 2; $\tau = 1$.

Dist.	Tasks	n=3	n=4	n=5	n=6
U[0,1]	3	1115.41%			
U[0,1]	4	46.27%	49.75%		
U[0,1]	5	19.52%	16.01%	21.68%	
U[0,1]	6	9.88%	7.49%	10.9%	12.18%
U[0,100]	3	1130.13%			
U[0,100]	4	58.23%	64.45%		
U[0,100]	5	17.97%	14.62%	21.21%	
U[0,100]	6	8.62%	7.05%	9.83%	11.51%

Table 3: Average improvement using Algorithm 2; $\tau = 2$.

The results are shown in Tables 2 and 3. We can see that there are cases where assigning multiple workers per task can offer a significant benefit. However, as the problem size increases, this benefit significantly attenuates, and it may suffice to just rely on the assignment obtained from the MILP.

6 Conclusion

We consider the problem of assigning tasks to workers when workers can be attacked, and their ability to successfully complete assigned tasks compromised. We show that the optimal assignment problem (in the sense of Stackelberg equilibrium commitment), when the attack takes place after the tasks have been assigned to workers, can be found in pseudo-polynomial time. Furthermore, when tasks are heterogeneous, we show that the problem is more challenging, as it could be optimal to assign multiple workers to the same task. Even if we constrain the assignment such that only one worker is assigned per task, extracting the optimal assignment becomes strongly NP-Hard (we exhibit an integer linear program for the latter problem). Finally, we provide with an algorithm of converting this constraint assignment to one that allows multiple workers per task (and hence approximate optimal allocation).

Acknowledgments

This research was partially supported by the National Science Foundation (CNS-1640624, IIS-1526860, IIS-1649972), Office of Naval Research (N00014-15-1-2621), Army Research Office (W911NF-16-1-0069), and National Institutes of Health (UH2 CA203708-01, R01HG006844-05).

References

[Alighanbari and How, 2005] Mehdi Alighanbari and Jonathan P How. Cooperative task assignment of unmanned aerial vehicles in adversarial environments. In *American Control Conference, 2005.*, pages 4661–4666. IEEE, 2005.

[Alistarh *et al.*, 2012] Dan Alistarh, Michael A Bender, Seth Gilbert, and Rachid Guerraoui. How to allocate tasks asynchronously. In *IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 331–340. IEEE, 2012.

[Alon *et al.*, 2015] Noga Alon, Michal Feldman, Omer Lev, and Moshe Tennenholtz. How robust is the wisdom of the crowds? In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pages 2055–2061, 2015.

[Carvalho *et al.*, 2016] Arthur Carvalho, Stanko Dimitrov, and Kate Larson. How many crowdsourced workers should a requester hire? *Annals of Mathematics and Artificial Intelligence*, 78(1):45–72, 2016.

[Chen *et al.*, 2011] Yudong Chen, Huan Xu, Constantine Caramanis, and Sujay Sanghavi. Robust matrix completion and corrupted columns. In *International Conference on Machine Learning (ICML)*, pages 873–880, 2011.

[Chen *et al.*, 2013] Yudong Chen, Constantine Caramanis, and Shie Mannor. Robust sparse regression under adversarial corruption. In *International Conference on Machine Learning (ICML)*, pages 774–782, 2013.

[Chen *et al.*, 2014] Richard Li-Yang Chen, Amy Cohn, Neng Fan, and Ali Pinar. Contingency-risk informed power system design. *IEEE Transactions on Power Systems*, 29(5):2087 – 2096, 2014.

[Conitzer and Sandholm, 2006] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, pages 82–90, 2006.

[Dai *et al.*, 2011] Peng Dai, Daniel Sabby Weld, et al. Artificial intelligence for artificial artificial intelligence. In *Proceedings of AAAI*, pages 1153–1159, 2011.

[Elmalech *et al.*, 2016] Avshalom Elmalech, David Sarne, Esther David, and Chen Hajaj. Extending workers' attention span through dummy events. In *Proceedings of HCOMP*, 2016.

[Feng *et al.*, 2014] Jiashi Feng, Huan Xu, Shie Mannor, and Shuicheng Yan. Robust logistic regression and classification. In *NIPS*, pages 253–261, 2014.

[Ghosh *et al.*, 2011] Arpita Ghosh, Satyen Kale, and Preston McAfee. Who moderates the moderators? crowdsourcing abuse detection in user-generated content. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, pages 167–176, 2011.

[Gu *et al.*, 2005] Dazhang Gu, Frank Drews, and Lonnie Welch. Robust task allocation for dynamic distributed real-time systems subject to multiple environmental parameters. In *Distributed Computing Systems, 2005. ICDCS 2005. Proceedings. 25th IEEE International Conference on*, pages 675–684. IEEE, 2005.

[Jones *et al.*, 2006] Edward Gil Jones, Brett Browning, M Bernardine Dias, Brenna Argall, Manuela Veloso, and Anthony Stentz. Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 570–575. IEEE, 2006.

[Kleywegt *et al.*, 2002] Anton J Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

[Korzhyk *et al.*, 2010] Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Complexity of computing optimal stackelberg strategies in security resource allocation games. In *Proceedings of AAAI*, pages 805–810, 2010.

[Liu and Chen, 2017] Yang Liu and Yiling Chen. Sequential peer prediction: Learning to elicit effort using posted prices. In *Proceedings of AAAI*, pages 607–613, 2017.

[Liu *et al.*, 2017] Chang Liu, Bo Li, Yevgeniy Vorobeychik, and Alina Oprea. Robust linear regression against training data poisoning. In *ACM Workshop on Artificial Intelligence and Security*, 2017.

[Manino *et al.*, 2016] Edoardo Manino, Long Tran-Thanh, and Nicholas R Jennings. Efficiency of active learning for the allocation of workers on crowdsourced classification tasks. *arXiv preprint arXiv:1610.06106*, 2016.

[Mason and Suri, 2012] Winter Mason and Siddharth Suri. Conducting behavioral research on amazon's mechanical turk. *Behavior research methods*, 44(1):1–23, 2012.

[Sheng *et al.*, 2008] Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of ACM SIGKDD*, pages 614–622, 2008.

[Singla and Krause, 2013] Adish Singla and Andreas Krause. Truthful incentives in crowdsourcing tasks using regret minimization mechanisms. In *Proceedings of the international conference on World Wide Web*, pages 1167–1178. ACM, 2013.

[Stackelberg, 1952] Heinrich von Stackelberg. Theory of the market economy. 1952.

[Steinhardt *et al.*, 2016] Jacob Steinhardt, Gregory Valiant, and Moses Charikar. Avoiding imposters and delinquents: Adversarial crowdsourcing and peer prediction. In *NIPS*, pages 4439–4447, 2016.

[Stone and Veloso, 1999] Peter Stone and Manuela Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, 1999.

[Tambe, 2011] Milind Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.

[Tran-Thanh *et al.*, 2014] Long Tran-Thanh, Trung Dong Huynh, Avi Rosenfeld, Sarvapali D Ramchurn, and Nicholas R Jennings. Budgetfix: budget limited crowdsourcing for interdependent task allocation with quality guarantees. In *Proceedings of AAMAS*, pages 477–484, 2014.

[Xu *et al.*, 2010] Huan Xu, Constantine Caramanis, and Sujay Sanghavi. Robust PCA via outlier pursuit. In *NIPS*, pages 2496–2504, 2010.