Robust Multi-Agent Task Assignment in Failure-Prone and Adversarial Environments

Russell Schwartz   Pratap Tokekar

Robotics Algorithms & Autonomous Systems Lab
University of Maryland, College Park

Abstract—The problem of assigning agents to tasks is a central computational challenge in many multi-agent autonomous systems. However, in the real world, agents are not always perfect and may fail due to a number of reasons. A motivating application is where the agents are robots that operate in the physical world and are susceptible to failures. This paper studies the problem of Robust Multi-Agent Task Assignment, which seeks to find an assignment that maximizes overall system performance while accounting for potential failures of the agents. We investigate both, stochastic and adversarial failures under this framework. For both cases, we present efficient algorithms that yield optimal or near-optimal results.

I. INTRODUCTION

Task assignment is a central issue in most multi-agent systems, as it determines which agents should be assigned to which tasks in order to optimize the system performance [1], [2]. This topic has received much attention as the interest in multi-agent systems has steadily increased. Numerous formulations of the problem have been discussed with varying constraints and objectives [3]–[5]. In the simplest form, the problem of assigning agents to tasks can be formulated as that of one-to-one bipartite graph matching which can be solved efficiently in polynomial time. When additional constraints are introduced, finding the optimal assignment often proves to be computationally intractable due to the combinatorial nature of the problem. Nonetheless, algorithms for task assignment have been designed which ensure near-optimal performance with suitable runtimes. Such algorithms have found a variety of applications including vehicle-target assignment [6], sensor coverage problems [7], and discrete resource allocation [8]. Many of these approaches are predicated on the assumption that agents will always complete their assigned tasks, which limits their applicability to settings where agent failures are rare or impossible. To account for uncertainty in the agents' performance, more complex models are required such as the cost-based system presented in [9].

In this paper, we focus on scenarios where the agents operate in failure-prone or adversarial environments. In such environments, an agent is susceptible to internal or external factors which can impede its performance. In particular,
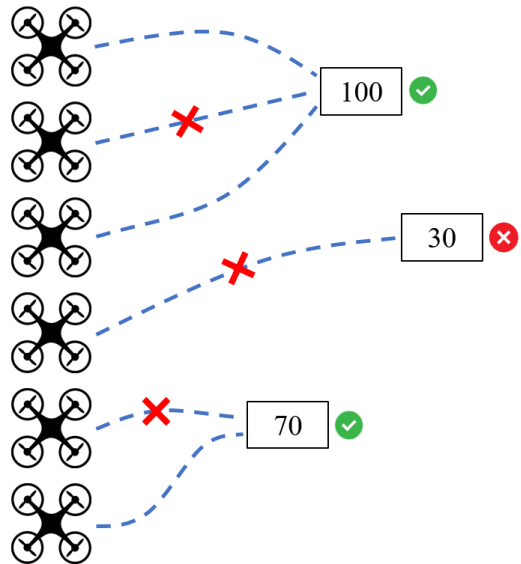
Fig. 1: An example of our agent/task assignment framework. Six agents are assigned to three tasks. Three of the agents fail and one of the tasks is not completed, resulting in a total profit of 170.

we consider failures and attacks which lead to complete malfunction (i.e., total failure in completing the assigned task) of individual agents. Our goal is to provide assignment algorithms that are robust to such failures.

We introduce the problem of robust multi-agent task assignment in both stochastic and adversarial environments. By stochastic, we refer to scenarios in which agents fail independently with a known probability p_i (possibly dependent on the task) as the result of some random process. By adversarial, we refer to scenarios in which at most α agents fail (where α is known) as the result of an attack by an intelligent adversary. The latter is, in essence, a Stackelberg game in which the assignment represents the leader's strategy, and the attack represents the follower's strategy [10]. For both environments, we provide exact and approximate solutions, with a particular effort placed on achieving runtimes that are independent of the number of agents. We conclude with empirical results of our algorithms' performance and a brief discussion of the possible applications of these concepts both within the field of robotics and elsewhere.

A. Related Work

There has been work on robust task assignment in the face of uncertainty in task profits and costs [11]. The ALLIANCE framework was one of the first works to designed to address multi-robot task assignment and coordination in failure-prone environments [12]. Since then a number of frameworks have been proposed to achieve fault-tolerant multi-agent systems [13]–[17]. Stochastic environments are most commonly addressed, since they appear often in the real world. Heuristics have been developed for various applications that help to account for random failures, but do not necessarily achieve optimality [9], [18]. Our work is concerned with a particular framework in which optimality (in terms of expected value) can be achieved. Specifically, we give exact expression for computing the optimal solution in case of stochastic failures for single-task, single-robot, instantaneous assignment (ST-SR-IA) as per the taxonomy by Gerkey and Mataric [2].

Assignment in adversarial environments, on the other hand, have not yet been as well explored. Aside from the obvious military applications, adversarial environments effectively models a variety of scenarios in which worst-case behavior is of key concern [19], [20]. Most closely related to our work, the notion of adversarial failure has been successfully used to model agent failures in the field of active target tracking [21], [22]. We use a similar model — an adversary can choose α robots to attack — but give an exact solution for the ST-SR-IA problem, unlike prior work which gave an approximate solution for the MT-SR-IA problem. We also present an approximate algorithm but show it to be near-optimal in practice.

Similar problems have been addressed in the combinatorial optimization community, specifically, the min-max knapsack, in which the values of the items in the knapsack are uncertain [23]. Our model corresponds to knapsack optimization where the *weights* are uncertain. As far as we are aware, the particular combinatorial optimization problems involved in our models are unique to this work.

B. Framework and Notation

For the purpose of this paper, the Robust Multi-Agent Assignment Problem (RMAAP) is concerned with the following natural framework. A set of tasks is given, each of which has an associated profit. A number of agents is also given, each of which can be assigned to at most one task. A task is completed successfully if it has at least one successful agent assigned to it. The objective is the sum of the profits of the completed tasks.

- Let k denote the number of tasks and N the number of agents.
- Let $T = [t_1, t_2, \cdots, t_k] \in \mathbb{R}^k$ denote the profits of the tasks. For convenience, we assume that T is given in *decreasing* order ($t_i \geq t_j$ for all $1 \leq i < j \leq k$).

- Let $x = [x_1, x_2, \cdots, x_k] \in \mathbb{Z}_{\geq 0}^k$ denote the assignment that assigns x_i agents to task i. An assignment x is valid iff $\sum_{i=1}^k x_i \leq N$.
- Let $H(n) = \min(1, n)$ denote the shifted heaviside function. This represents the completion indicator variable for a task with n successful agents assigned to it.

When no failures may occur, the total profit under assignment x is given by $\sum_{i=1}^k t_i \cdot H(x_i)$. In this case, the task assignment problem is trivial: total profit is clearly maximized by assigning one agent to each of the $\min(k, N)$ most valuable tasks. The problem becomes much more complex when failures are possible.

C. Organization and Contributions

The organization of the remainder of the paper is as follows. In Section 2, we formulate RMAAP for stochastic environments and then provide two different exact solutions (one pseudo-polynomial that runs in $O(k + N \log k)$ time and one strongly polynomial that runs in $O(k^2)$ time). In Section 3, we re-formulate RMAAP for adversarial environments and provide a justification for the attack model. We then give an exponential-time exact solution that runs in $O\left(\frac{N^k}{(k-1)!^2}\right)$ time along with a proof of NP-Hardness. This is followed by a polynomial-time approximate solution that runs in $O(k^2)$ time. We conclude with empirical results for the optimality of this solution and compare it to various baselines, followed by a discussion of future work.

II. STOCHASTIC RESILIENCE

A. Problem Formulation

For illustration, consider a scenario in which you wish deploy a set of autonomous robots (agents) to survey a set of objects in an area (tasks), where the importance (associated profit) of each object has been determined ahead of time. You know that each robot has a chance of suffering a malfunction that will prevent it from completing its assignment, where the probability of such a failure occurring is dependent on the task at hand. Intuitively, this associates a difficulty level to each task. As before, each object only requires one successful robot to be accurately surveyed (i.e., there is no benefit gained from having more than one). Assume these malfunctions occur independently with probability p_i for an agent assigned to task i. What assignment will maximize the *expected value* of the total profit? This scenario is modeled by the following non-linear mixed-integer problem.

Problem 1 (Stochastic Failure). *Given task values $T \in \mathbb{R}^k$, number of agents $N \in \mathbb{Z}_{\geq 0}$, and failure probability $p \in$*

$[0, 1]^k$, *find an assignment* $x \in \mathbb{Z}^k$ *that*

maximizes $\quad \displaystyle\sum_{i=1}^{k} t_i(1 - p_i^{x_i})$

subject to $\quad \displaystyle\sum_{i=1}^{k} x_i \leq N \quad and \quad x_i \geq 0 \;\; for\; all\;\; 1 \leq i \leq k$

The key notion introduced here is that of redundancy. It may now be advisable to assign multiple agents to the same tasks in the hopes that if one agent fails, the rest will be able to take its place. This comes at the cost of leaving the less valuable tasks unattempted. For example, given $N = 3$, $T = [70, 30, 10]$, $p = [0.3, 0.3, 0.3]$, the optimal solution is $x^* = [2, 1, 0]$

B. Exact Solution

We begin by presenting an intuitive greedy algorithm that solves Problem 1 exactly in pseudo-polynomial time. Since its runtime is dependent on the number of agents, we then refine our algorithm by using continuous relaxation to achieve strongly-polynomial runtime.

A greedy algorithm will give the optimal solution to Problem 1. Intuitively, we iterate over the agents, assigning them sequentially to the task that results in the largest marginal increase in expected value. By using a max heap to keep track of the most profitable tasks, we can implement the approach given in Algorithm 1 in $O(k + N \log k)$.

Algorithm 1 Greedy solution to Problem 1

$x \leftarrow [0, 0, \ldots, 0]$
for $i = 1$ to N **do**
$\quad z \in \arg\max_{1 \leq j \leq k} \left(t_j \cdot p_j^{x_j}(1 - p_j) \right)$
$\quad x_z \leftarrow x_z + 1$
end for
return x

Since the runtime is dependent on N, the greedy algorithm is pseudo-polynomial. This makes it satisfactory for only sufficiently small values of N. Alternatively, the problem can be solved exactly in strongly polynomial time by first solving the continuous relaxation (i.e., allowing for fractional agents) and then converting the solution back to the discrete case.

We modify Problem 1 to allow $x_i \in \mathbb{R}$ and denote the objective function f. Also note that since an optimal solution will never leave agents un-assigned, we can assume that $\sum x_i = N$. Now, we deduce that an optimal assignment must form a solution to the following system of equations [24]:

$$\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial x_2} = \cdots = \frac{\partial f}{\partial x_k}$$
$$\Rightarrow -\ln(p_1)t_1 p_1^{x_1} = -\ln(p_2)t_2 p_2^{x_2} = \cdots = -\ln(p_k)t_k p_k^{x_k}$$
$$\Rightarrow \ln(-\ln(p_1)t_1) + x_1 \ln(p_1) = \ln(-\ln(p_2)t_2) + x_2 \cdots$$

Let $a_i = \ln(p_i)$ and $b_i = \ln(-\ln(p_i)t_i)$. We now have the following linear system:

$$\begin{cases} a_1 x_1 + b_1 = a_2 x_2 + b_2 = \cdots = a_k x_k + b_k \\ x_1 + x_2 + \cdots + x_k = N \end{cases}$$

which can be solved in linear time:

$$Z \leftarrow \left(N + \sum_{i=1}^{k} \frac{b_i}{a_i} \right) \Big/ \left(\sum_{i=1}^{k} \frac{1}{a_i} \right)$$
$$x_i \leftarrow \frac{1}{a_i}(Z - b_i)$$

In most cases, this method yields a valid solution. However, we have not yet fully enforced our constraints. If T contains a particularly wide range of values, then the assignment produced may actually have a negative number of agents assigned to one or more of the least valuable tasks. If this is the case, we remove the problematic tasks from T and recompute with the same method, repeating this process until all values are non-negative. This is equivalent to optimizing over the boundary formed by setting these values identically to 0. Due to the convexity of f, this iterative procedure will always yield the global maximum subject to our constraints.

Algorithm 2 Relaxed solution to Problem 1

$x \leftarrow [0, 0, \ldots, 0]$ // Solve continuous relaxation
$C \leftarrow 0$
repeat
$\quad Z \leftarrow \left(N + \sum_{i=1}^{k} \frac{b_i}{a_i} \right) / \left(\sum_{i=1}^{k} \frac{1}{a_i} \right)$
$\quad c \leftarrow 0$
$\quad$**for** $i = 1$ to k **do**
$\quad\quad x_i \leftarrow \frac{1}{a_i}(Z - b_i)$
$\quad\quad$**if** $x_i < 0$ **then**
$\quad\quad\quad c \leftarrow c + 1$
$\quad\quad$**end if**
$\quad$**end for**
$\quad k \leftarrow k - c$
$\quad C \leftarrow C + c$
until $x_i \geq 0$ for all $1 \leq i \leq k$
for $i = 1$ to k **do**   // Convert to discrete
$\quad x_i \leftarrow \lfloor x_i \rfloor$
end for
$L \leftarrow N - \sum_{i=1}^{k} x_i$
for $i = 1$ to L **do**
$\quad z \in \arg\max_{1 \leq j \leq k} \left(t_j \cdot p^{x_j}(1 - p) \right)$
$\quad x_z \leftarrow x_z + 1$
end for
return $[x_1, x_2, \cdots, x_k, \underbrace{0, 0, \cdots, 0}_{C}]$

Now, to convert this solution back into the discrete case, we simply take the floor of each value, calculate the number of remaining unassigned agents, and then use the greedy

procedure from Algorithm 1 to assign them sequentially. Algorithm 2 demonstrates this approach.

Theorem 1 (Performance of Algorithm 2). *Given an instance of Problem 1, the performance of Algorithm 2 is bounded as follows:*

  1) (Optimality) *The assignment returned yields the maximum possible expected value for total profit.*
  2) (Runtime) *The overall time complexity is $O(k^2)$, which is independent of the number of agents.*

It is evident from our derivation that Algorithm 2 is optimal for the relaxed version. The conclusion that it is also optimal for the original version follows from the correctness of Algorithm 1 and is further supported by empirical results: In 10^5 random trials, the optimal solution was found in each case. The runtime is explained by the following. Since computing Z and its corresponding assignment is accomplished in $O(k)$, and at most k such instances are required, the continuous relaxation is solved in $O(k^2)$. Further, since $L \leq k$, converting to a discrete assignment costs at most $O(k \log k)$. Thus, the overall complexity of Algorithm 2 is $O(k^2)$.

III. ADVERSARIAL RESILIENCE

A. Problem Formulation

Problem 1 effectively models scenarios in which we wish to maximize the expected value of our profit. However, this is not always the best metric for success in failure-prone environments. Consider a scenario in which completing a set of tasks is mission-critical. That is, if the total profit gained from attempting to complete these tasks falls below some threshold, then the overall mission will fail. In cases like these, it may be helpful to consider the *worst-case* profit of a given assignment as opposed to the *average* profit as we did in Problem 1.

In order to meaningful define the worst-case, we must constrain the space of possible failures in some way. One very general way is to take some small $\varepsilon \in [0, 1]$ and only consider the set of failures which occur with probability at least ε. We then optimize under the assumption that the worst such failure will always occur. This provides a guarantee of performance in "all but the most unlikely" of occurrences. In the interest of tractability, for this model we will take some positive integer $\alpha < N$ and only consider the set of failures in which the number of agents is at most α. This cardinality constraint provides a good approximation for the probabilistic constraint, especially when the variance in p_i is low.

Definition (α-tolerance). *Given an assignment x, a profit threshold $F \in \mathbb{R}$, and a failure limit $\alpha \in \mathbb{Z}$, we say that x is α-tolerant iff, for all cases in which at most α agents fail, the resulting profit is at least F.*

In other words, an assignment is α-tolerant if it guarantees profit F even in the worst case. In addition to providing an approximation for the for the probabilistic constraint (sometimes called ε-tolerance), this definition encapsulates the notion of robustness in a way that exactly models certain real world scenarios. Two examples are hard drive failures in RAID arrays (where a certain number of drives are required to retain data integrity) [19] and cooperative swarm robotics (where a certain number of failures can lead to a total break in communication) [20].

An equivalent way of viewing this notion of robustness is to suppose that agents fail, not by chance, but instead by the actions of an intelligent adversary. In particular, we consider scenarios in which first, a defender assigns agents to tasks, and then an attacker disables at most α of the agents with the intention of minimizing the defenders profit. The attacker has full knowledge of the defender's assignment (x) and the defender's valuations of the tasks (T). The targets selected by the attacker constitutes the worst-case failure for the given assignment. Any assignment that yields profit at least F in the adversarial environment is clearly α-tolerant in the stochastic environment (and vice-versa). This perspective also lends itself to various real-world scenarios, the most obvious of which being military applications [25].

We begin by formalizing the set of valid target selections available to the attacker in response to assignment x with failure limit α:

$$\Delta_\alpha(x) = \left\{ \delta \in \mathbb{Z}_{\geq 0}^k \;\middle|\; \left(\sum_{i=1}^k \delta_i \leq \alpha \right) \wedge \left(\forall_i (\delta_i \leq x_i) \right) \right\}$$

Selection $\delta \in \Delta_\alpha(x)$ represents the attack that disables δ_i of the agents assigned to task i. Thus, $x - \delta$ gives the agents remaining after the attack has occurred. When an attacker behaves optimally, the δ selected will result in the minimum profit for the defender. The problem of finding the best assignment in the face of an optimal attacker is modeled by the following.

Problem 2 (Adversarial Failure). *Given task values $T \in \mathbb{R}^k$, number of agents $N \in \mathbb{Z}_{\geq 0}$, and failure limit $\alpha \in \mathbb{Z}_{\geq 0}$ (with $\alpha \leq N$), find an assignment $x \in \mathbb{Z}^k$ that*

$$maximizes \quad \min_{\delta \in \Delta_\alpha(x)} \sum_{i=1}^k t_i \cdot H(x_i - \delta_i)$$

$$subject \ to \quad \sum_{i=1}^k x_i \leq N \quad and \quad x_i \geq 0 \ for \ all \ \ 1 \leq i \leq k.$$

Solving the inner minimization problem reflects the role of the optimal attacker, and the overall maximin problem the role of the defender. Problem 2 may be interpreted as a two-stage perfect information zero-sum sequential game (sometimes called a "Stackelberg" game) between these two players [10]. Consider the following example. Given $N = 9$, $T = [90, 65, 55, 30, 15]$, $\alpha = 3$, the defender should implement the assignment $x^* = [3, 2, 2, 1, 1]$. In response,

the attacker will implement $\delta = [0, 2, 0, 1, 0]$, resulting in a total profit of 160.

B. Exact Solution

We begin by fully characterizing the behavior of the optimal attacker in response to assignment x. Observe that the attacker only serves to gain when they target all agents assigned to a given task. Thus, the attacker's problem can be simplified to that of selecting *tasks* instead of agents. Specifically, the attacker wishes to select a subset of tasks with maximal total value, such that the sum of the agents assigned to those tasks is at most α (thus maximizing the harm to the defender). This is an instance of the 0-1 Knapsack Optimization Problem, where the size of the knapsack is α, the weight of item i is x_i, and the value of item i is t_i. This is itself an NP-Hard problem. However, it omits a dynamic programming solution which can be implemented in pseudo-polynomial time: $O(\alpha k)$.

With this in mind, it is now possible to implement a brute force solution that iterates over every valid assignment, solving the knapsack instance for each one, and returning the assignment that yields the greatest profit. The runtime of this approach is given by:

$$O(\alpha k) \cdot |X| = O\left(\frac{Nk(N+k)!}{N!k!} \right)$$

where X denotes the set of all valid assignments. The cardinality of X follows from the observation that X represents the number of compositions of N into at most k parts. This can be greatly improved by the following observation.

Proposition 1 (Decreasing Optimal Assignment). *For any instance of Problem 2, there exists an optimal assignment for which the agents are distributed in decreasing order. Formally, $\exists \tilde{x} \in X$ s.t. $f(\tilde{x}) = \max_{x \in X} f(x)$ and $\tilde{x}_i \geq \tilde{x}_j$ for all $1 \leq i < j \leq k$.*

We now restrict the search space to only the assignments that satisfy this decreasing property. Denote this smaller set $\tilde{X}$. In contrast with X, $\tilde{X}$ represents the *partitions* of N into at most k parts, which has a much lower asymptotic bound [26, Eq. 26.9.10]. The runtime of brute force is now given by:

$$O(\alpha k) \cdot |\tilde{X}| = O\left(\frac{N^k}{(k-1)!^2} \right)$$

C. NP-Hardness

It is easy to see that the decision version of Problem 2 is strictly NP-Hard. Any algorithm that would be able to verify if a given assignment achieves profit at least F in polynomial time would also be able to solve the Knapsack Decision Problem (KDP) in polynomial time. This follows precisely from the behavior of the attacker as discussed in the previous section. However, it is well known that KDP is NP-Complete [27]. Thus, unless $P = NP$, a solution to the decision version cannot be verified in polynomial time,

---

Algorithm 3 Approximate Solution to Problem 2

$U_0 \leftarrow 0$ // Construct prefix sum array
for $i = 0$ to k **do**
    $U_{i+1} \leftarrow U_i + t_i$
end for
$p_{max} \leftarrow -\infty$
for $m = 1$ to k **do**
    $x \leftarrow [0, 0, \cdots, 0]$ // Construct assignment
    $c \leftarrow \lfloor N/m \rfloor$
    $d \leftarrow N - cm$
    for $i = 1$ to m **do**
       $x_i \leftarrow c$
       if $i \leq d$ **then**
          $x_i \leftarrow x_i + 1$
       end if
    end for
    $p_{min} \leftarrow \infty$ // Evaluate assignment
    for $r = 0$ to $\min \left(d, \lfloor \frac{\alpha}{c+1} \rfloor \right)$ **do**
       $s \leftarrow \min \left(m - d, \lfloor \frac{\alpha - r(c+1)}{c} \rfloor \right)$
       $p \leftarrow (U_d - U_r) + (U_m - U_{d+s})$
       $p_{min} \leftarrow \min(p_{min}, p)$
    end for
    if $p > p_{max}$ **then**
       $p_{max} \leftarrow p$
       $x_{max} \leftarrow x$
    end if
end for
return x_{max}

---

making it strictly NP-Hard. Since the optimization version of Problem 2 is at least as hard as the decision version, it is also strictly NP-Hard.

D. Approximate Solutions

Our approximation approach involves sampling $\tilde{X}$ by selecting a subset of plausible assignments, calculating the resulting post-attack profit for each one (by solving the inner minimization problem), and returning the assignment which yields the maximum. We restrict the number of assignments sampled to be polynomial in k. In particular, we only sample the assignments which *evenly* distribute all N agents amongst the m most valuable tasks for some $1 \leq m \leq k$. When N is not evenly divisible by m, we give the most valuable tasks precedence in receiving the additional agents. Formally, for every m, we set $c = \lfloor N/m \rfloor$ and sample the assignment

$$x_i = \begin{cases} c+1 & 1 \leq i \leq N - cm \\ c & N - cm < i \leq m \\ 0 & m < i \leq k \end{cases}$$

For example, $m = 2$ gives $\left[\frac{N}{2}, \frac{N}{2}, 0, 0, \cdots, 0 \right]$ when N is even and $\left[\frac{N+1}{2}, \frac{N-1}{2}, 0, 0, \cdots, 0 \right]$ when N is odd. Restricting our search space in this manner allows us to find adequate
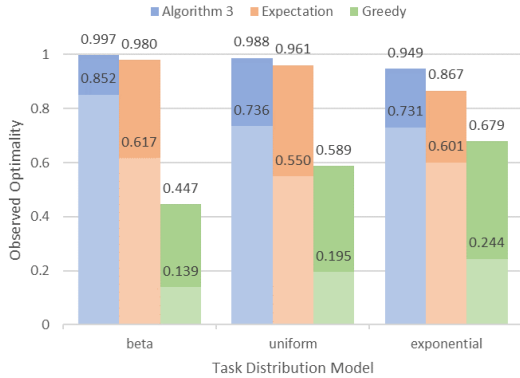
Fig. 2: Optimality of Algorithm 3 compared to baselines under various distributions. Both average and minimum observed ratios are shown for each case.

solutions while avoiding the combinatorial intractability of $\tilde{X}$. However, one potential problem is that we must solve the inner minimization problem for each sample. As shown before, this reduces to the knapsack optimization problem in the general case, which can be costly even when using dynamic programming. However, since the assignments in question are of a specific form, we can actually solve these knapsack instances exactly in polynomial time. The key insight is that the number of agents assigned to each task is limited to one of three values: 0, c, or $c + 1$. This greatly reduces the number of feasible attacks and allows us to efficiently iterate over the boundary of the space.

Algorithm 3 utilizes this approach to find an approximate solution in polynomial time. For each of the k different values of m, the construction and evaluation of x are both $O(k)$. All pre-computation is also $O(k)$. Thus, the overall runtime of Algorithm 3 is $O(k^2)$. For further illustration of Algorithm 3's behavior, observe the following non-trivial test case. Inputs $N = 20, T = [89, 78, 71, 68, 57, 56, 53, 43], \alpha = 7$ yield the following assignments and profits.

- Optimal (Brute Force): $[4, 4, 2, 2, 2, 2, 2, 2] \Rightarrow 319$
- Approx. (Algorithm 3): $[3, 3, 3, 3, 2, 2, 2, 2] \Rightarrow 313$

In this case, Algorithm 3 achieved 98% optimal profit. The general effectiveness is discussed in the next section.

E. Empirical Results

In order to determine the effectiveness of our approximation scheme, Algorithm 3 was run against a large number of test cases (i.e., instances of Problem 2). For each case, the optimal solution was computed via the brute-force method described in section 3.2 and compared to the approximate by taking the ratio of their respective post-attack profits. Two additional baseline solutions were also computed: the "greedy" baseline, which naively assigns $\alpha + 1$ agents to as many of the tasks as possible (in the order of decreasing profits), randomly assigning any remaining agents; and the "expectation" baseline, which treats the problem as if it were

purely stochastic, calculating p_i as α/N and then applying Algorithm 2.

Three distinct test suites were run, each with a different model for the distribution of task values. Specifically, the distributions tested were the continuous distribution on $[0, 1]$, the exponential distribution with $\lambda = 2$, and the beta distribution with $\alpha = 6$, $\beta = 2$. Respectively, these represent scenarios in which the task values are skewed symmetrically, positively, and negatively [28]. For each distribution, 10,000 trials were run, and the values for the number of tasks, number of agents, and failure limit were randomly selected from $\{(k, N, \alpha) \in \mathbb{Z}^3 \mid 2 \leq k \leq N \leq 30, \ 2 < \alpha < N\}$.

Figure 1 shows the results of these tests, displaying both the average and minimum observed profit ratios for each type. Table 1 shows the average runtime of each algorithm over all three test suites. Evidently, Algorithm 3 achieved an average of over 95% optimal in all three cases, with performance as high as 99% in the case of the beta distribution. This significantly outperforms both baselines. Further, all 30,000 test cases achieved at least 70% optimal, implying that Algorithm 3 has reliable worst-case performance. We hypothesize that Algorithm 3 will always achieve at least $\frac{2}{3}$ optimal but have not yet been able to formally prove this conjecture.

Algorithm	Runtime (ms)
Brute Force	453.1
Algorithm 3	0.176
Expectation	0.176
Greedy	0.017

Table 1: Average runtimes of exact, approximate, and baseline algorithms. Tests were limited to cases with 30 or fewer agents. Run in Python 3 on an Intel i7-4790k @ 4.5GHz.

IV. CONCLUSION AND FUTURE WORK

We take key steps towards ensuring robustness in multi-agent systems that suffer from failures. In particular, we formulate the Robust Multi-Agent Assignment Problem for stochastic and adversarial environments within a set framework and demonstrate the need for such models. We provide an efficient exact solution for the stochastic case and an approximate solution for the adversarial. We demonstrate and analyze the effectiveness of these algorithms with custom simulations and discuss guaranteed performance. Notably, the results of this paper extend to any multi-agent system that fits the framework, not just those that arise in robotics.

This work is still ongoing, with our primary focus placed on increasing the complexity of the model to account for a broader class of problems. For instance, the framework could be expanded in various ways to include heterogeneous agents. There is also work to be done regarding general ε-tolerance in adversarial systems. There are potentially better ways (compared to the cardinality constraint) to approximate ε-tolerance while remaining tractable. The notion of robustness in multi-agent systems is both potentially important and largely unexplored.

References

[1] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[2] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International journal of robotics research*, vol. 23, no. 9, pp. 939–954, 2004.

[3] H. Lee, J. Motes, and N. M. Amato, "Optimal multi-agent multi-task planning."

[4] C. Zhang and J. A. Shah, "Co-optimizing multi-agent placement with task assignment and scheduling." in *IJCAI*, 2016, pp. 3308–3314.

[5] S. Biswas, S. G. Anavatti, and M. A. Garratt, "A time-efficient co-operative path planning model combined with task assignment for multi-agent systems," *Robotics*, vol. 8, no. 2, p. 35, 2019.

[6] G. Arslan, J. R. Marden, and J. S. Shamma, "Autonomous Vehicle-Target Assignment: A Game-Theoretical Formulation," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 584–596, 04 2007. [Online]. Available: https://doi.org/10.1115/1.2766722

[7] J. Marden and A. Wierman, "Distributed welfare games with applications to sensor coverage," 01 2009, pp. 1708 – 1713.

[8] T. Yamaguchi, K. Yahiro, and M. Yokoo, "Student-project-resource matching-allocation problems: Game theoretic analysis," 2019.

[9] S. Timotheou, "Asset-task assignment algorithms in the presence of execution uncertainty," *The Computer Journal*, vol. 54, no. 9, pp. 1514–1525, 2011.

[10] R. B. Myerson, *Game Theory: Analysis of Conflict*. Harvard Univ. Pres, 2007.

[11] M. Alighanbari and J. P. How, "A robust approach to the uav task assignment problem," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 18, no. 2, pp. 118–134, 2008.

[12] L. E. Parker, "Alliance: An architecture for fault tolerant multirobot cooperation," *IEEE transactions on robotics and automation*, vol. 14, no. 2, pp. 220–240, 1998.

[13] N. Grigoropoulos, M. Koutsoubelias, and S. Lalis, "Byzantine fault tolerance for centrally coordinated missions with unmanned vehicles," in *Proceedings of the 17th ACM International Conference on Computing Frontiers*, 2020, pp. 165–173.

[14] E. Melnik, A. Klimenko, and I. Safronenkova, "An estimation of distributed algorithms of the fault-tolerant management in the robot groups," in *International Conference on Interactive Collaborative Robotics*. Springer, 2020, pp. 212–221.

[15] N. Ayanian, "Dart: Diversity-enhanced autonomy in robot teams," *The International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1329–1337, 2019.

[16] M. J. Matarić, G. S. Sukhatme, and E. H. Østergaard, "Multi-robot task allocation in uncertain environments," *Autonomous Robots*, vol. 14, no. 2-3, pp. 255–263, 2003.

[17] L. Vig and J. A. Adams, "Multi-robot coalition formation," *IEEE transactions on robotics*, vol. 22, no. 4, pp. 637–649, 2006.

[18] R. Alami, F. F. Ingrand, and S. Qutub, "A scheme for coordinating multi-robots planning activities and plans execution." in *ECAI*. Citeseer, 1998, pp. 617–621.

[19] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant arrays of inexpensive disks (raid)," *SIGMOD Rec.*, vol. 17, no. 3, p. 109–116, Jun. 1988. [Online]. Available: https://doi.org/10.1145/971701.50214

[20] Y. Mohan and S. G. Ponnambalam, "An extensive review of research in swarm robotics," in *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, Dec 2009, pp. 140–145.

[21] L. Zhou, V. Tzoumas, G. J. Pappas, and P. Tokekar, "Resilient active target tracking with multiple robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 129–136, 2018.

[22] ——, "Distributed attack-robust submodular maximization for multi-robot planning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2479–2485.

[23] G. Yu, "On the max-min 0-1 knapsack problem with robust optimization applications," *Operations Research*, vol. 44, no. 2, pp. 407–415, 1996. [Online]. Available: http://www.jstor.org/stable/171805

[24] D. P. Bertsekas, *Convex Optimization Theory*. Athena Scientific, Belmont, Massachusetts, 2009.

[25] P. R. Young and P. E. Lehner, "Applications of a theory of automated adversarial planning to command and control," *IEEE transactions on systems, man, and cybernetics*, vol. 16, no. 6, pp. 806–812, 1986.

[26] "*NIST Digital Library of Mathematical Functions*," http://dlmf.nist.gov/, Release 1.0.25 of 2019-12-15, 2019, f. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain, eds. [Online]. Available: http://dlmf.nist.gov/

[27] R. Karp, "Reducibility among combinatorial problems," vol. 40, 01 1972, pp. 85–103.

[28] D. Forsyth, *Probability and statistics for computer science*. Cham, Switzerland: Springer, 2018.