2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Algorithm for Multi-Robot Chance-Constrained Generalized Assignment Problem with Stochastic Resource Consumption

Fan Yang[1] and Nilanjan Chakraborty[2]

Abstract— **We present a novel algorithm for the multi-robot generalized assignment problem (GAP) with stochastic resource consumption. In this problem, each robot has a resource (e.g., battery life) constraint and it consumes a certain amount of resource to perform a task. In practice, the resource consumed for performing a task can be uncertain. Therefore, we assume that the resource consumption is a random variable with known mean and variance. The objective is to find an assignment of the robots to tasks that maximizes the team payoff. Each task is assigned to at most one robot and the resource constraint for each robot has to be satisfied with very high probability. We formulate the problem as a chance-constrained combinatorial optimization problem and call it the chance-constrained generalized assignment problem (CC-GAP). This problem is an extension of the deterministic generalized assignment problem, which is a NP-hard problem. We design an iterative algorithm for solving CC-GAP in which each robot maximizes its own objective by solving a chance-constrained knapsack problem in an iterative manner. The approximation ratio of our algorithm is $(1+\alpha)$, assuming that the deterministic knapsack problem is solved by an α-approximation algorithm. We present simulation results to demonstrate that our algorithm is scalable with the number of robots and tasks.**

I. Introduction

Multi-robot task assignment problem is a fundamental problem that arises in a wide variety of application scenarios like manufacturing, automated transport of goods, environmental monitoring, and surveillance. Multi-robot generalized assignment problem (GAP) [1], [2], [3], is one formalization of task assignment problems that arises in situations where robots have resource constraints and have to consume a certain amount of the resource to obtain the payoff for performing a task. Each robot can perform multiple tasks as long as the total resource consumed to execute the tasks is within the resource budget. Each task is assigned to at most one robot. The objective is to find the assignment with the maximum team payoff. GAP is a NP-hard combinatorial optimization problem, which has been extensively studied both in operations research [4], [5] and theoretical computer science [6], [7], [2], [8]. However in many multi-robot application scenarios, the resource consumption in executing tasks are influenced by uncertain environmental factors and may not be known exactly before task execution. Consequently, the resource consumption could be stochastic. Therefore, in this paper, we study the generalized assignment problem with stochastic resource consumption constraints.

GAP with stochastic resource consumption is useful in many multi-robot applications. The resource could be energy, task execution time, or any other consumable resources depending on specific application. One application is the multi-robot part delivery in an automated factory. A team of robots should pick up parts from certain clustered storage locations and deliver it to processing stations. In this situation, the energy consumed by each robot to travel between the storage location and delivery station could be influenced by different sources of uncertainty. For example, robot may consume more energy to avoid static and moving obstacles.

In this paper, we present algorithms for stochastic generalized assignment problem where the resource consumption are random variables with known means and variances. The problem is formulated as the chance-constrained generalized assignment problem (CC-GAP). More formally: *Given n_r robots and n_t tasks, with a payoff for each robot-task pair, and a constraint on the resource (e.g., battery life) of each robot, find an assignment of robots to tasks so that the team payoff is maximized and each robot can execute its tasks with a guarantee that its resource constraint will not be violated with high probability (say* 0.99*) irrespective of the actual value of the random resource consumption taken in any realization*. The team payoff is the sum of the individual robot payoffs.

Contributions: We model the stochastic generalized assignment problem as a chance-constrained combinatorial optimization problem. Inspired by the idea for solving deterministic GAP in [2], we design an algorithm for CC-GAP where each robot solves a chance-constrained knapsack problem (CC-KNAP) sequentially. Leveraging algorithms for chance-constrained knapsack problem optimally, from our previous work [9], we provide an α-approximation algorithm for CC-KNAP using any α-approximation algorithm for deterministic knapsack problem as a subroutine. The solution of CC-GAP is proved to be a $(1+\alpha)$-approximation algorithm (which is 2 if the CC-KNAP is solved optimally). We present simulation results demonstrating the scalability of our algorithm with the number of robots and tasks. To the best of our knowledge, this is the first algorithm that solves CC-GAP with $(1+\alpha)$-approximation ratio.

II. Related Work

Task allocation is a key problem in many applications of multi-robot systems, e.g., point-to-point parts transfer [10], multi-robot routing [11], multi-robot decision making [12], and other multi-robot coordination problems (see [13], [14], [15]). There are different variations of the multi-robot task

[1]Fan Yang and [2]Nilanjan Chakraborty are with the Mechanical Engineering Department at Stony Brook University. Email: {fan.yang.3, nilanjan.chakraborty}@stonybrook.edu This work was supported in part by AFOSR award FA9550-15-1-0442 and NSF award CMMI 1853454.

assignment problem that have been studied in the literature depending on the assumptions about the tasks and the robots (see [16], [13], [17], [18] for taxonomies and surveys). We will focus on the literature about multi-robot task allocation under uncertainty.

In [19], the authors analyze the sensitivity of the optimal assignment with respect to the uncertainty in payoffs. In [20], a redundant robot assignment on graphs with uncertain edge costs is studied. Task allocation or team formation with chance constraints have been studied in [21], [22], [23], [9], [24]. In [24], the authors study discrete submodular maximization problem with payoff uncertainty. In [25], we have studied the chance-constrained simultaneous task assignment and path planning for multiple robots with uncertain path cost. In all of the papers above, the uncertainty is in the objective which is the *total* path cost of the robot team. The chance-constrained generalized assignment considered in this paper has uncertain resource consumption which is in the constraint of the formulation and we have to make sure that the resource constraint for *each* robot is not violated with probabilistic guarantee.

Most of the current literature on stochastic generalized assignment problem (SGAP) study the problem in which the resource consumption is uncertain before an initial assignment is made and a recourse decision is allowed to compensate for the adverse effect of the initial assignment after the actual consumption is known (usually with incurred penalty). The two-stage (or multi-stage) programs are widely used here [26], [27], [28], [29], [30]. However, in robotics applications, the tasks have to be executed before the random resource cost is known and thus the two-stage models are not applicable. Therefore, we model SGAP as a chance-constrained combinatorial optimization problem.

The stochastic knapsack problem is a special case of SGAP when there is only one robot. Based on the sources of the uncertainty, it can be categorized into random payoffs-deterministic weights and random weights-deterministic payoffs. The former problems are studied in [31], [32], [33], [34]. The latter problems are also widely studied. The multiple-stage model is used in [35], [36]. The chance constraint model is used in [37], [38], [39], [9], [40], [41]. A PTAS for chance-constrained knapsack problem with Gaussian random variable is proposed in [39]. A Relaxed FPTAS is presented in [41]. In our previous work [9], we present method that compute the optimal solution of chance-constrained knapsack problem with the assumption that the deterministic knapsack problem is solved optimally.

III. Problem Formulation

We will now introduce the chance-constrained generalized assignment problem and a sub-problem of CC-GAP called the chance-constrained knapsack problem. We will show the relationship between the chance-constrained knapsack problem and the deterministic risk-averse knapsack problem, which is key for designing algorithms to solve CC-GAP.

We consider a set of robots $\{r_1, \dots, r_{n_r}\}$ and a set of tasks $\{t_1, \dots, t_{n_t}\}$. Let a_{ij} be the payoff for assigning robot r_i to task t_j. Let w_{ij} be the resource (e.g., energy) consumed by robot r_i to perform task t_j. Each robot has a resource capacity W_i. Let x_{ij} be the binary decision variable that is 1 when t_j is assigned to r_i and 0 otherwise. The deterministic generalized assignment problem is

$$\begin{aligned} \max \quad & \sum_{i=1}^{n_r} \sum_{j=1}^{n_t} a_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{n_t} w_{ij} x_{ij} \le W_i, \quad \forall i = 1, \dots, n_r \\ & \sum_{i=1}^{n_r} x_{ij} \le 1, \quad \forall j = 1, \dots, n_t \\ & x_{ij} \in \{0, 1\}, \quad \forall i, j \end{aligned} \tag{1}$$

where the first constraint in (1) is the *resource constraint* for each robot. The second constraint guarantees that each task is exclusively assigned to at most one robot.

In (1), the resource consumption variable, w_{ij}, is assumed to be deterministic. However, in practice, the resource consumption is usually a random variable. Thus, the satisfaction of the resource constraint in (1) is dependent on the actual realization of the random variables, which cannot be known unless the task is executed. Therefore, we formulate a variation of (1), in which the resource constraint is represented as a stochastic constraint.

A. Chance-constrained generalized assignment problem

We assume that the variables, w_{ij}, are independent Gaussian random variables with known mean μ_{ij} and variance σ_{ij}^2. Therefore, the resource constraint in (1), is written as a probabilistic constraint, as shown in (2). The goal of CC-GAP is to *compute the assignment of robots to tasks with the maximum payoff such that each robot is able to finish the assigned tasks with a high probability (say* 0.99*).*

$$\begin{aligned} \max \quad & \sum_{i=1}^{n_r} \sum_{j=1}^{n_t} a_{ij} x_{ij} \\ \text{s.t.} \quad & \mathbb{P}\left(\sum_{j=1}^{n_t} w_{ij} x_{ij} \le W_i \right) \ge p, \quad \forall i = 1, \dots, n_r \\ & \sum_{i=1}^{n_r} x_{ij} \le 1, \quad \forall j = 1, \dots, n_t \\ & x_{ij} \in \{0, 1\}, \quad \forall i, j \end{aligned} \tag{2}$$

The first constraint in (2) is the chance constraint which guarantees that for each robot, r_i, the resource consumption for executing its assigned tasks $\sum_{j=1}^{n_t} w_{ij} x_{ij}$ does not exceed its resource capacity W_i with probability at least p, in any realization of the random resource consumption. Note that there is one chance constraint for each robot, so there are a total of n_r chance constraints.

It is well known in stochastic optimization [23], [42] that probabilistic constraints of the form in (2) can be re-written

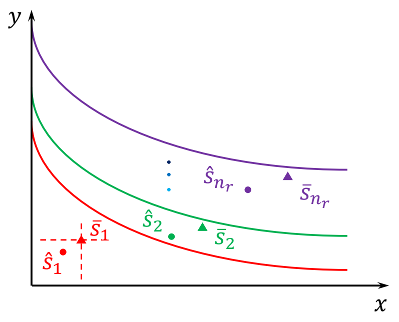

Fig. 1: The assignment of any robot is represented by a point on variance-mean plane. The feasible solution of CC-GAP is a set of points that are under the parabola defined by the corresponding chance constraints, e.g., $\{\hat{S}_i\}_{i=1}^{n_r}$.

as a deterministic constraint

$$\sum_{j=1}^{n_t} \mu_{ij} x_{ij} + C \sqrt{\sum_{j=1}^{n_t} \sigma_{ij}^2 x_{ij}} \le W_i \tag{3}$$

where C is a constant that depends on the probability, p. For a distribution with known cumulative distribution function ϕ, $C = \Phi^{-1}(p)$. For distributions with information only about the mean and variance, and no other information, $C = \sqrt{\frac{p}{1-p}}$.

Geometric Interpretation of (3): For any robot, r_i, there is a point on a two-dimensional space in which the horizontal and vertical coordinate are equal to the sum of variances and means of the random resource consumption of the assigned tasks, i.e., $x = \sum_{j=1}^{n_t} \sigma_{ij}^2 x_{ij}$ and $y = \sum_{j=1}^{n_t} \mu_{ij} x_{ij}$. We call the plane as the variance-mean plane (see Fig. 1). Equation (3) implies that the chance constraint for each robot is a parabola in the variance-mean plane with vertical intercept equal to W_i. Any feasible assignment for r_i should be located below the corresponding parabola. Therefore *the feasible solution of CC-GAP in* (2) *is a set of points on variance-mean plane which are located in the region below the corresponding parabola defined by the chance constraint* (3). Besides, the assignments in the feasible solution of CC-GAP should also satisfy the assignment constraints in (2).

The idea of our method is to compute the feasible assignment that satisfies the chance constraint for each robot ($\bar{S}_i$ in Fig. 1) and then re-assign the tasks such that each task is assigned to at most one robot while the chance constraints are not violated (e.g., the initial assignment $\bar{S}_1$ changes to $\hat{S}_1$ in Fig. 1).

B. Chance-constrained knapsack problem

If we consider the CC-GAP problem for one robot, say r_k, we get the chance-constrained knapsack problem, shown in (4). The objective is to find the maximum payoff assignment for a single robot say r_k such that the resource consumption is within its capacity W_k with high probability. CC-KNAP is a combinatorial optimization problem with a non-convex feasible region, which is difficult to solve in general.

$$\begin{aligned} \max \quad & \sum_{j=1}^{n_t} a_{kj} x_{kj} \\ \text{s.t.} \quad & \sum_{j=1}^{n_t} \mu_{kj} x_{kj} + C \sqrt{\sum_{j=1}^{n_t} \sigma_{kj}^2 x_{kj}} \le W_k \\ & x_{kj} \in \{0,1\}, \quad \forall j = 1, \dots, n_t \end{aligned} \tag{4}$$

If we relax the integer constraint for x_{kj}, the problem is a second order cone program with integrality gap $\Omega(\sqrt{n_t})$ [39]. In our previous work [9], we showed that the CC-KNAP problem in (4) can be solved by solving a sequence of deterministic problems called the risk-averse knapsack problem (RA-KNAP). The RA-KNAP, given below, is a deterministic knapsack problem in which the resource consumption is equal to a linear combination of the mean and variance, i.e., $\mu_{kj} + \lambda \sigma_{kj}^2$ and the resource capacity is equal to W' (not necessarily equal to W_k in (4)).

$$\begin{aligned} \max \quad & \sum_{j=1}^{n_t} a_{kj} x_{kj} \\ \text{s.t.} \quad & \sum_{j=1}^{n_t} \mu_{kj} x_{kj} + \lambda \sum_{j=1}^{n_t} \sigma_{kj}^2 x_{kj} \le W' \\ & x_{kj} \in \{0,1\}, \quad \forall j = 1, \dots, n_t \end{aligned} \tag{5}$$

The parameter λ is called the risk-aversion index (or parameter). Figure 2 shows the geometric relationship between the CC-KNAP and RA-KNAP on the variance-mean plane.

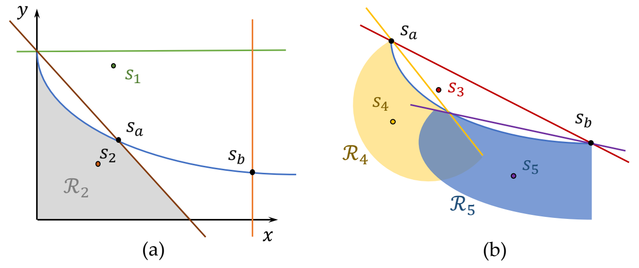

Fig. 2: The feasible assignments of CC-KNAP and RA-KNAP are located below the parabola defined by chance constraint in (4) and straight line defined by the resource constraint in (5) respectively. $\mathcal{R}_i$ denotes the intersection of feasible region of CC-KNAP and RA-KNAP.

Geometric Relationship between CC-KNAP and RA-KNAP: As shown in Figure 2, any feasible assignment of CC-KNAP is a point under the parabola defined by the chance constraint in (4). The feasible region for RA-KNAP is the space below the straight line shown in Fig. 2(a). The slope of the straight line is $-\lambda$ and W' is equal to its vertical intercept. Depending on the position of the straight line defined by the resource constraint in (5), the optimal solution of (5) could be either below or above the parabola defined by the chance constraint in (4). In particular, the optimal solution of (5) is feasible to (4) if the solution is below the parabola, e.g., s_2 in Fig. 2(a). The solution is thus a lower bound of optimal solution of (4). In a special case, the straight line from RA-KNAP in (5) is tangent to the parabola from CC-KNAP in (4). Then any feasible solution of (5) is below the parabola and therefore feasible to (4). Based on these facts, we have the following important lemma.

Lemma 1: There exists a problem (5) with proper values of λ and W' such that the optimal solution of (5) is also the optimal solution to problem (4).

Proof: Let s^* be the optimal solution of CC-KNAP in (4). There always exists a straight line tangent to the parabola defined by chance constraint in (4) such that its lower half-space covers s^*. Suppose $-\overline{\lambda}$ and $\overline{W}'$ denote the slope and vertical intercept respectively. The feasible region of the resource constraint with $\overline{\lambda}$ and $\overline{W}'$ of RA-KNAP in (5) is a subset of the feasible region defined by the parabola from (4). Therefore s^* is the optimal solution of RA-KNAP with $\overline{\lambda}$ and $\overline{W}'$. This is also true for RA-KNAP with parameters different from $\overline{\lambda}$ and $\overline{W}'$ as long as the lower half-space of straight line of RA-KNAP covers s^*, and the optimal solution of RA-KNAP is feasible to the chance constraint of CC-KNAP. For example, if we rotate and translate the straight line on variance-mean plane (change λ and W') in such a way that s^* is below it and the objective value of all feasible solutions of RA-KNAP above parabola is less than that of s^* (or no feasible solution of RA-KNAP is above the parabola), the optimal solution of the corresponding RA-KNAP is still s^*. ■

Lemma 1 implies that one way to solve the CC-KNAP is to perform a two-dimensional search on the parameters λ and W' to find the proper values of λ and W'.

IV. Solution Approach

In this section, we will introduce the algorithms to solve CC-KNAP and CC-GAP. The algorithm for CC-KNAP iterates over different values of λ and W' methodically to cover the whole feasible region of CC-KNAP. To solve CC-GAP, each robot solves CC-KNAP sequentially with updated payoffs in each iteration. We will prove that the obtained solution of CC-KNAP is α-approximate when the approximation ratio for the subroutine solving RA-KNAP is α. Consequently, the obtained solution of CC-GAP is $(1+\alpha)$-approximate.

Definition 1: A solution (assignment), $\overline{s}$, of a maximization problem is *α-approximate*, with $\alpha \geq 1$, if and only if $\alpha v(\overline{s}) \geq \max_{s \in \mathcal{R}} v(s)$ where $v(\cdot)$ denotes the objective function value and $\mathcal{R}$ denotes the feasible region.

A. Algorithm for CC-KNAP

The algorithm for solving CC-KNAP follows [9]. However, the performance analysis of the algorithm differs from the analysis in [9]. In [9], we assumed that the deterministic knapsack problems (i.e., RA-KNAPs) can be solved optimally. However, here we prove that the solution of algorithm for CC-KNAP has the same approximation ratio as the solution of the deterministic knapsack problem. This provides us the flexibility to select any algorithm that solves the deterministic knapsack problem as a subroutine for CC-KNAP, based on the application. To make the paper self-contained, we will now provide a brief description for solving the CC-KNAP.

Let $\mathcal{R}$ denote the feasible region defined by the chance constraint of CC-KNAP and $\mathcal{R}_i$ denote the intersection region of $\mathcal{R}$ and the feasible region defined by the resource constraint of RA-KNAP with λ_i (e.g., $\mathcal{R}_2$ in Fig. 2(a)). The algorithm has two parts: find a feasible solution of CC-KNAP by solving RA-KNAPs with increasing λ in (5) (line 1-6, Alg. 1), search the rest of feasible region of CC-KNAP by solving RA-KNAPs with methodically updated λ and W' (line 7-8, Alg. 1). The procedures are shown in Algorithm 1 and Algorithm 2.

The first part starts with solving RA-KNAP for $\lambda_i = 0$, $W'_i = W_k$, where $i = 1$ is the index of the iteration (line 1-2, Alg. 1). If the optimal solution of RA-KNAP does not satisfy the chance constraint, in other words the corresponding point is above the parabola, e.g. s_1 in Fig. 2 (a), the algorithm computes the resource constraint of the RA-KNAP for the next iteration by updating λ as $\lambda_{i+1} = C/\sigma_i$ where $\sigma_i = \sqrt{\sum_{j=1}^{n_t} \sigma_{kj}^2 x_{kj}}$ (line 4, Alg. 1). From a geometric viewpoint, on the variance-mean plane, the updating procedure can be treated as the straight line from RA-KNAP's constraint rotating clockwise at vertical intercept $(0,\ W_k)$. The new straight line is guaranteed to be located below the previous point so that the previous solution will not be obtained again. The procedure continues until we obtain a feasible solution of CC-KNAP, e.g., s_2 in shaded region $\mathcal{R}_2$ below the parabola in Fig. 2(a) (line 3-5, Alg. 1). The feasible solution s_2 is stored in the solution set $\mathbb{S}$ (line 6)*. Now we can conclude that the subset of feasible region of CC-KNAP, $\mathcal{R}_2$ is dominated by s_2, i.e., $s_2 = \arg\max_{s \in \mathcal{R}_2} v(s)$ (or $\alpha v(s_2) \geq \max_{s \in \mathcal{R}_2} v(s)$ if the solution of RA-KNAP is α-approximate). Note that this procedure terminates in finite number of iterations because λ_i increases as the algorithm proceeds. The resource consumption of all tasks in RA-KNAP thus increase while the resource capacity remains the same. In the worst case, the resource consumption becomes so large that the robot cannot perform any of the task. At this moment, the first part of Alg. 1 terminates and $\mathbb{S}$ is an empty set.

We then need to find the optimal (α-approximate) solution of the rest of the feasible region using the second part of Alg. 1, e.g., $\mathcal{R} \backslash \mathcal{R}_2$ in Fig. 2(a). We compute the intersection point of parabola and the straight line of the last RA-KNAP's constraint, e.g., s_a in Fig. 2. Further, we compute the point on the parabola with horizontal coordinate $\sigma^2 = \sum_{j=1}^{n_t} \sigma_{kj}^2 x_{kj}$, e.g., s_b in Fig. 2 (line 7, Alg. 1). Therefore the search region is now limited to the space enclosed by the parabola from s_a to s_b, vertical straight line through s_b and the straight line from the last RA-KNAP's constraint (see Fig. 2(a)). Then the algorithm calls a recursive function $\mathcal{A}(s_a, s_b, \mathbb{S})$ presented in

* Formally, $\mathbb{S}$ is the set of all solutions obtained from RA-KNAP that satisfy the chance constraint. For ease of exposition, $\mathbb{S}$ also denotes the set of indices (subscripts) of solution by abuse of notation.

Alg. 2. Let (σ_a^2, μ_a) and (σ_b^2, μ_b) be the coordinates of s_a and s_b respectively. $\mathcal{A}(s_a, s_b, \mathbb{S})$ computes the equation of the straight line through s_a and s_b, and then solves RA-KNAP with λ and W' equal to the negation of the slope and vertical intercept of the obtained straight line equation respectively, i.e., $\lambda = -\frac{\mu_a-\mu_b}{\sigma_a^2-\sigma_b^2}$ and $W' = \mu_a + \lambda\sigma_a^2$ (line 1, Alg. 2). If the obtained solution $s_{ab} = (\sigma_{ab}^2, \mu_{ab})$ does not satisfy the chance constraint(e.g., s_3 in Fig. 2(b)), we compute the slope of a straight line through s_a and s_{ab}, i.e., $\frac{\mu_a-\mu_{ab}}{\sigma_a^2-\sigma_{ab}^2}$. Then we compute the point s_c which is the intersection of the parabola and a straight line through s_a with slope $-(\frac{\mu_a-\mu_{ab}}{\sigma_a^2-\sigma_{ab}^2} + \epsilon)$ where ϵ is a small value that prevent the cycling (line 2-3, Alg. 2). As shown in Fig. 2(b), the current search region is thus covered by union of $\mathcal{R}_4$ and $\mathcal{R}_5$. Then it calls itself with input $(s_a, s_c, \mathbb{S})$ and $(s_c, s_b, \mathbb{S})$. When solution s_{ab} satisfies the chance constraint, s_{ab} is stored in $\mathbb{S}$ and the recursive function $\mathcal{A}(s_a, s_b, \mathbb{S})$ terminates. s_{ab} dominates the corresponding intersection region, e.g, s_4 dominates $\mathcal{R}_4$ in Fig. 2(b). When all recursive functions terminate, we obtain a set of feasible solutions such that $\mathcal{R}$ is covered by the union of intersection region corresponding to those feasible solutions in $\mathbb{S}$, i.e., $\mathcal{R} \subseteq \bigcup_{i\in\mathbb{S}} \mathcal{R}_i$. The optimal ($\alpha$-approximation) solution $\bar{s}^*$ is the one with the highest objective function value (line 9 in Alg. 1). The output is the indices of tasks assigned to r_k, denoted by $\bar{J}_k^*$.

Algorithm 1 Algorithm to solve CC-KNAP for r_k

```
Input: $W_k$, $p$, $\mu_{kj}$, $\sigma_{kj}^2$ $\forall j = 1, \ldots, n_t$.
Output: $\bar{J}_k^*$.
1: Initialize $i = 1$, $\lambda_i = 0$, $W_i' = W_k$.
2: Solve RA-KNAP with $\lambda_i$, $W_i'$ and obtain solution $s_i$.
3: while $s_i$ does not satisfy the chance constraint do
4:     Update risk-averse parameter $\lambda_{i+1} = C/\sigma_i$ and capacity $W_{i+1}' = W_i'$, and $i = i + 1$.
5:     Solve RA-KNAP with $\lambda_i$ and $W_i'$, and obtain solution $s_i$.
6: Store solution $s_i$ in set $\mathbb{S}$.
7: Compute intersection point $s_a$ and $s_b$.
8: Call recursive function $\mathcal{A}(s_a, s_b, \mathbb{S})$.
9: Compute the optimal objective value $\bar{v}^* = \max_{s_i\in\mathbb{S}} v(s_i)$ and solution $\bar{s}^* = \arg\max_{s_i\in\mathbb{S}} v(s_i)$.
10: return $\bar{J}_k^*$ which contains all assigned tasks in $\bar{s}^*$.
```

Lemma 2: If the solution of the RA-KNAP in (5) is α-approximate, Algorithm 1 computes an α-approximate solution to CC-KNAP.

Proof: Since the approximation ratio of the algorithm that solves RA-KNAP is α and $\mathcal{R}_i$ is a subset of feasible region defined by the resource constraint of RA-KNAP in (5), any obtained solution from RA-KNAP that satisfies the chance constraint, say $s_i \in \mathbb{S}$ is α-approximation in $\mathcal{R}_i$. Let v_i^* be the optimal objective value in $\mathcal{R}_i$. It is true that $\alpha v(s_i) \geq v_i^*, \forall i \in \mathbb{S}$.

When Alg. 1 finishes, the union of $\mathcal{R}_i$ covers the whole feasible region $\mathcal{R}$, i.e., $\mathcal{R} \subseteq \bigcup_{i\in\mathbb{S}} \mathcal{R}_i$. Therefore, the optimal solution of CC-KNAP denoted by s^* must be in at least one of the intersection region $\mathcal{R}_i$. Let $\mathbb{I}$ denote the set of indices of $\mathcal{R}_i$ which contain s^* ($\mathbb{I} \subset \mathbb{S}$ is a non-empty set). Since $\mathcal{R}_i \subset \mathcal{R}$, $v_i^* = \max_{s\in\mathcal{R}_i} v(s)$, $v(s^*) = \max_{s\in\mathcal{R}} v(s)$ and $s^* \in \mathcal{R}_i, \forall i \in \mathbb{I}$, it implies that $v_i^* = v(s^*), \forall i \in \mathbb{I}$.

Algorithm 2 Recursive function $\mathcal{A}(s_a, s_b, \mathbb{S})$

```
Input: $s_a, s_b, \mathbb{S}$.
Output: $\mathbb{S}$.
1: Solve RA-KNAP (5) with $\lambda = -\frac{\mu_a-\mu_b}{\sigma_a^2-\sigma_b^2}$ and $W' = \mu_a + \lambda\sigma_a^2$.
2: if the solution $s_{ab} = (\sigma_{ab}^2, \mu_{ab})$ does not satisfy the chance constraint of CC-KNAP in (4) then
3:     Compute the intersection point $s_c$ of parabola and straight line through $s_a$ with slope $-(\frac{\mu_a-\mu_{ab}}{\sigma_a^2-\sigma_{ab}^2} + \epsilon)$.
4:     Call $\mathcal{A}(s_a, s_c, \mathbb{S})$ and then call $\mathcal{A}(s_c, s_b, \mathbb{S})$
5: else
6:     return $\mathbb{S} = \{\mathbb{S}, s_{ab}\}$.
```

Because $\bar{s}^*$, the solution of CC-KNAP obtained by Alg. 1 is the one with the highest objective value in $\mathbb{S}$, it is clear that $v(\bar{s}^*) \geq v(s_i), \forall i \in \mathbb{S}$.

Therefore we can conclude that $\alpha v(\bar{s}^*) \geq \alpha v(s_i) \geq v_i^* = v(s^*), \forall i \in \mathbb{I}$. Since $\mathbb{I}$ is a non-empty set, the solution of CC-KNAP obtained by Alg. 1 is α-approximate. ■

If RA-KNAP is solved optimally using dynamic program, our algorithm provides the optimal solution of CC-KNAP.

B. Algorithm for CC-GAP

The algorithm for solving CC-GAP is presented in Algorithm 3. The iterative procedure is executed by the robots sequentially from r_1 to r_{n_r} (line 2-8). The initial payoff value of CC-KNAP for r_1 performing any task, ${}^1a_{1j}$ is equal to $a_{1j}, \forall j$. In any iteration k, CC-KNAP for r_k with payoffs $\{{}^k a_{kj}\}_{j=1}^{n_t}$ is solved by Alg. 1. The solution is denoted by $\bar{J}_k^*$ which is the set for all assigned tasks for r_k (line 3-4). If any of the task say t_u in the current solution has been assigned to a robot with smaller index say $r_b, (b < k)$, this task is removed from previous set $\bar{J}_b^*$ (line 5-6). Hence each task is assigned to at most one robot. Then the payoffs of all tasks in $\bar{J}_k^*$ to robots with greater indices are equal to ${}^k a_{ij} - {}^k a_{kj}, \forall i > k$ and $\forall j \in \bar{J}_k^*$. The payoffs of other tasks remain the same (line 7). Then r_{k+1} executes the same procedure with updated payoffs. Once the last robot r_{n_r} finishes the procedure, the current assignment of robots to tasks is the solution, i.e., $\{\hat{J}_i^*\}_{i=1}^{n_r}$ (line 9-10).

Lemma 3: (Feasibility) The assignments for all robots S obtained from our algorithm always satisfy the chance constraints.

Proof: The initial assignment of any robot $\bar{J}_k^*$ obtained by solving CC-KNAP of r_k satisfies the chance constraint. Consider it as a point below the corresponding parabola on variance-mean plane (see $\bar{S}_1$ in Fig. 1). During Alg. 3, the assigned tasks for r_k might be re-assigned to other robots. The final assignment of r_k, $\hat{J}_k^*$ is a subset of $\bar{J}_k^*$. Therefore the sum of the variances and means of the assigned tasks in

Algorithm 3 Algorithm to solve CC-GAP

```
Input: $a_{ij}$, $W_i$, $\mu_{ij}$, $\sigma_{ij}^2$, $p$, $\forall i = 1, ..., n_r, j = 1, ..., n_t$.
Output: $S$.
 1: Let initial payoff value ${}^1a_{ij} = a_{ij}, \forall i = 1, ..., n_r, \forall j = 1, ..., n_t$.
 2: for robot $r_1$ to $r_{n_r}$ do
 3:     Solve CC-KNAP with payoffs value $\{{}^k a_{kj}\}_{j=1}^{n_t}$ by Alg. 1.
 4:     Obtain solution $\bar{J}_k^* = \{j | x_{kj} = 1\}$.
 5:     if $\bar{J}_k^* \bigcap \{\bar{J}_i^*\}_{i=1}^{k-1} \neq \emptyset$ then
 6:         Remove all $t_u \in \bar{J}_k^* \bigcap \{\bar{J}_i^*\}_{i=1}^{k-1}$ from original set.
 7:     Update ${}^{k+1}a_{ij} = {}^k a_{ij} - {}^k a_{kj}, \forall i > k$ and $\forall j \in \bar{J}_k^*$ (the payoffs for unassigned tasks remain the same).
 8:     $k = k + 1$.
 9: Let $\hat{J}_i^* = \bar{J}_i^*, \forall i = 1, ..., n_r$.
10: return $S = \{\hat{J}_i^*\}_{i=1}^{n_r}$.
```

$\hat{J}_k^*$ are smaller. It is clear that the point of the final assignment must be on the southwest side of the initial assignment, e.g., $\hat{S}_1$ in Fig. 1. Hence the final assignment must be under the corresponding parabola. This is true for all robots. Therefore the final assignments $\{\hat{J}_i^*\}_{i=1}^{n_r}$ satisfy the chance constraints. ■

Lemma 4: (Optimality) If the algorithm for CC-KNAP is α-approximate, then the algorithm for CC-GAP is a $(1+\alpha)$-approximate.

Lemma 4 is proved by induction. The base case is that the assignment of the last robot r_{n_r} in the final solution of Alg. 3, i.e, $\hat{J}_{n_r}^*$ is $(1+\alpha)$-approximation to the CC-GAP in which the payoffs are equal to ${}^{n_r}a_{ij}$ and r_{n_r} is the only robot involved (in other words, $i = n_r$). This is true because the initial assignment of r_{n_r} obtained by solving CC-KNAP, i.e., $\bar{J}_{n_r}^*$, is an α-approximation to the optimal solution of CC-KNAP, which is equivalent to the considered CC-GAP and the initial assignment of r_{n_r}, $\bar{J}_{n_r}^*$, is same with the final assignment $\hat{J}_{n_r}^*$. Then we need to prove that the final assignment of robots from r_k to r_{n_r} $(1 \leq k < n_r)$, i.e, $\{\hat{J}_i^*\}_{i=k}^{n_r}$ is $(1+\alpha)$-approximate to the CC-GAP in which the payoffs are equal to ${}^k a_{ij}, \forall i = k, ..., n_r, \forall j = 1, ..., n_r$. If this is true, it implies

$$(1+\alpha)\sum_{i=1}^{n_r}\sum_{j\in\hat{J}_i^*} {}^1a_{ij} \geq \max\{\sum_{i=1}^{n_r}\sum_{j=1}^{n_t} {}^1a_{ij}x_{ij}\} \qquad (6)$$

As shown in the first line of Alg. 3, ${}^1a_{ij} = a_{ij}$. Therefore $\{\hat{J}_i^*\}_{i=1}^{n_r}$ is $(1+\alpha)$-approximate to the original CC-GAP in (2). The proof that $\{\hat{J}_i^*\}_{i=k}^{n_r}$ is $(1+\alpha)$-approximate is similar to the deterministic GAP. We omit this proof here due to space constraints and the fact that it is similar to the proof in [2] for deterministic GAP. Please refer to [2] for a detailed description.

V. Simulation Results

The computational cost of our algorithm is $\sum_{i=1}^{n_r} K_i T_i$, where K_i denotes the number of the deterministic knapsack problems (RA-KNAP) (5) solved by r_i, and T_i denotes the computational cost for solving the knapsack problem. We use a pseudo-polynomial algorithm from [43] for solving the knapsack problem. Thus, the computational cost for solving one knapsack problem is $O(n_i^2 A_i)$ where n_i denotes the number of tasks of the knapsack problem solved by r_i and A_i denotes the highest payoff, i.e., $A_i = \max_{j=1}^{n_i} {}^i a_{ij}$.

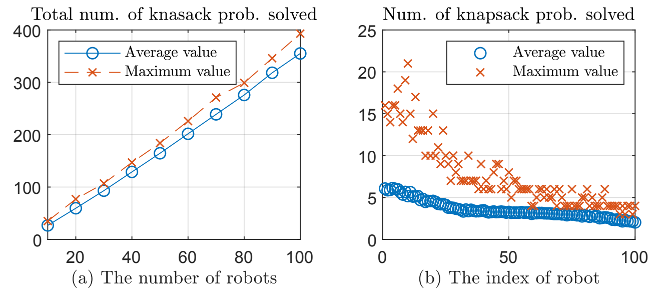

Fig. 3: The scalability with the number of robots: (a) the total number of deterministic knapsack problems solved by all robots; (b) The number of deterministic knapsack problems solved by each robot (with index from 1 to 100).

Since our algorithm solves CC-GAP by solving a sequence of deterministic knapsack problems, the key metrics for the efficiency of our algorithm are the number of deterministic knapsack problem solved by each robot, i.e., K_i and the total number of the knapsack problems solved by all robots, i.e., $\sum_{i=1}^{n_r} K_i$. However these metrics are problem parameter dependent and it is hard to provide *a priori* bounds. In this section, we will study the dependence of these metrics on the number of robots and tasks. Our extensive simulations show that (a) the algorithm to solve CC-GAP is scalable with the number of robots and tasks, and (b) the algorithm solves chance-constrained knapsack problem by solving small number of deterministic knapsack problems.

The motivating scenario that we consider for this simulation study is that of multi-robot parts delivery in an automated factory. A number of parts are located at a central store. A robot has to pick up a part from the central store and deliver it to a specific workstation for assembly. Each robot obtains certain payoff for delivering parts successfully. The payoffs a_{ij} are generated from a discrete uniform distribution $\mathcal{U}(20, 100)$. The resource consumed by a robot to deliver a part is the fuel cost for r_i traveling from its initial position to the central store and from the central store to the work station of the assigned part. The parts may be heterogeneous, (i.e., of different weights and values) and the robots are also heterogeneous (i.e., have different weight carrying capability), so the energy consumed depends on the robot-task pair. The means, μ_{ij}, and variances, σ_{ij}^2, of the random fuel cost are created from continuous uniform distribution $\mathcal{U}(20, 100)$ and $\mathcal{U}(9, 36)$ respectively. Each robot has limited fuel capacity. The capacity W_i for each robot is generated from continuous uniform distribution $\mathcal{U}(350, 400)$. The probability p that robots deliver parts successfully is 0.99. The simulations were done on a computer with Intel $i7$ 2.6 GHZ CPU and 16 GB RAM.

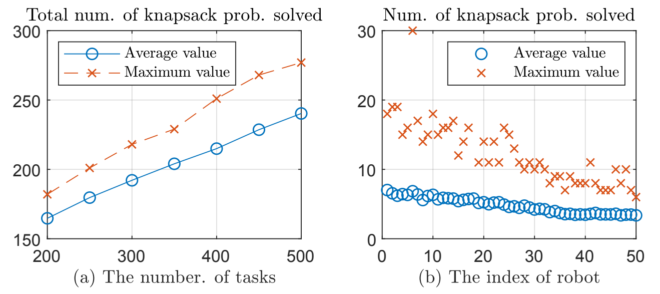

Fig. 4: The scalability with the number of tasks: (a) the total number of deterministic knapsack problems solved by all robots; (b) the number of deterministic knapsack problems solved by each robot (with index from 1 to 50).

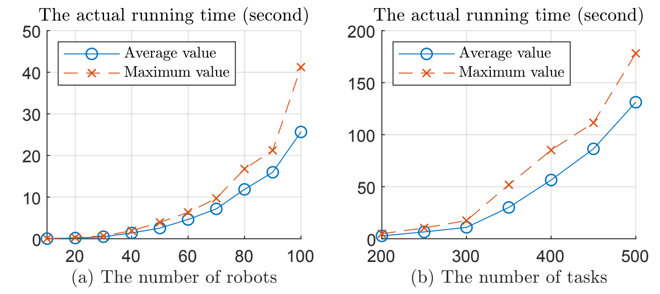

Fig. 5: The actual running time (in seconds) as a function of (a) number of robots (first set of simulations) and (b) number of tasks (second set of simulations).

Scalability with number of robots: In the first set of simulations, we study the scalability of our algorithm with the number of robots, which are varied from 10 to 100 with increment of 10. The number of tasks is always 4 times of the number of robots. The results are presented in Fig. 3(a). For a fixed number of robots, we create 100 problem instances with randomly generated parameters, and compute both average and maximum number of deterministic knapsack problems solved by all robots (i.e., $\sum_{i=1}^{n_r} K_i$) over these instances. It is shown that both the average and the maximum number of deterministic knapsack problems solved grow almost linearly.

Since our algorithm is sequential, the position of a robot in the sequence, may affect the number of calls to the deterministic knapsack problem. We use the index of a robot for its position in the sequence. For example, the robot with index 1 is the first robot to solve the problem (4). To study the influence of the index of the robots, we present the average and maximum number of knapsack problems solved by each robot (i.e., $K_i, \forall i$). The results are computed from the same 100 instances with 100 robots used in Fig. 3(a). The results are shown in Fig. 3(b), in which the horizontal axis represents the index of each robot. It shows that robots with larger indices solve lesser number of knapsack problems. The reason is that the chance-constrained knapsack problem for a robot with larger index is simpler than the problems solved before. In particular, the payoffs become smaller in general and some of the payoffs could be negative (the task in this case can be ignored). More importantly, Fig. 3(b) shows that the number of knapsack problems solved by each robot is small (less than 23 for all scenarios). We also study the number of knapsack problems solved by a single robot with particular index as the number of robots increases (i.e., $K_i, i \in \{1, ..., 10\}$ as n_r increases). The results for different robots show that the number of problems solved grows linearly. It implies that our algorithm that solve chance-constrained knapsack problem is scalable with the number of robots. For reference, the average and maximum actual running time of the simulations are provided in Fig. 5(a).

Scalability with number of tasks: In the second set of simulations, we study the scalability of our algorithm with the number of tasks. The number of robots is 50 while the number of tasks vary from 200 to 500 with increment of 50. The results are presented in Fig. 4(a). Similar to the first set of simulations, for a given number of tasks, the average and maximum number of deterministic knapsack problems solved by all robots (i.e., $\sum_{i=1}^{n_r} K_i$) are computed from 100 randomly generated scenarios. Generally speaking, the average and maximum number of knapsack problems solved increases linearly. In Fig 4(b), we present the average and maximum number of knapsack problems solved by each robot (i.e., $K_i, \forall i$). The results are computed from 100 randomly generated scenarios with 500 tasks used in Fig. 4(a). The horizontal axis represents the index of robot. As before, it shows that the robots with larger index tend to solve smaller number of knapsack problems. The number of knapsack problems solved by a single robot is no greater than 30. Further, we study the number of knapsack problems solved by a single robot with particular index as the number of tasks increase (i.e., $K_i, i \in \{1, ...50\}$ as n_t increase). For example, the number of problems solved by r_1 increases almost linearly from 4.5 to 7.5. The numbers for other robots are smaller than r_1 and increase in a similar manner to r_i. It implies that the algorithm that solves chance-constrained knapsack problem is scalable with the number of tasks. The actual running time of the simulations is shown in Fig. 5(b).

VI. Conclusion

In this paper, we present an algorithm to solve the generalized assignment problem with stochastic resource consumption. We formulate the problem as a chance-constrained combinatorial optimization problem, called CC-GAP. We propose an algorithm that solves CC-GAP by solving a closely related sub-problem, CC-KNAP, for each individual robot. Assuming that the random resource consumption follows a Gaussian distribution, we presented an α-approximate solution for CC-KNAP with α being the approximation ratio for solving a deterministic knapsack problem. The obtained solution for CC-KNAP is optimal if RA-KNAP is solved optimally. Further, we prove that our algorithm for computing CC-GAP is $(1 + \alpha)$-approximate when we use an α-approximation algorithm for CC-KNAP for each robot. Thus, the best approximation ratio of our algorithm is 2. We prove that our solution always satisfies the chance constraint. The simulation results demonstrate that our algorithm is scalable with the number of robots and tasks. In the future, we would like to study the distributed implementation and the theoretical computational complexity of our algorithm.

REFERENCES

[1] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2009.

[2] R. Cohen, L. Katzir, and D. Raz, "An efficient approximation for the generalized assignment problem," *Information Processing Letters*, vol. 100, pp. 162–166, 2006.

[3] L. Luo, N. Chakraborty, and K. Sycara, "Distributed algorithm design for multi-robot generalized task assignment problem," in *Proceedings of International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan*, November 2013.

[4] G. Ross and R. Soland, "A branch and bound algorithm for the generalized assignment problem," *Mathematical Programming*, vol. 8, pp. 91–103, 12 1975.

[5] M. Savelsbergh, "A branch-and-price algorithm for the generalized assignment problem," *Operations Research*, vol. 45, pp. 831–841, 12 1997.

[6] D. Shmoys and . Tardos, "Approximation algorithm for the generalized assignment problem," *Math. Program.*, vol. 62, pp. 461–474, 02 1993.

[7] C. Chekuri and S. Khanna, "A ptas for the multiple knapsack problem," *Proc. of SODA*, pp. 213–222, 09 2001.

[8] L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko, "Tight approximation algorithms for maximum general assignment problems," in *Proc. of ACM-SIAM SODA*, 2006, pp. 611–620.

[9] F. Yang and N. Chakraborty, "Algorithm for optimal chance constrained knapsack problem with applications to multi-robot teaming," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 1043–1049.

[10] L. Luo, N. Chakraborty, and K. Sycara, "Provably-good distributed algorithm for constrained multi-robot task assignment for grouped tasks," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 19–30, Feb 2015.

[11] M. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson, and S. Jain, "Auction-based multi-robot routing," in *Robotics Science and Systems*, 2005.

[12] C. Bererton, G. Gordon, S. Thrun, and P. Khosla, "Auction mechanism design for multi-robot coordination," in *NIPS*, 2003.

[13] M. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257 –1270, jul. 2006.

[14] H.-L. Choi, L. Brunet, and J. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 912–926, 2009.

[15] L. Luo, N. Chakraborty, and K. Sycara, "Distributed algorithms for multirobot task assignment with task deadline constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 876–888, July 2015.

[16] B. P. Gerkey and M. J. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[17] A. R. Mosteo and L. Montano, "A survey of multi-robot task allocation," Instituto de Investigacin en Ingeniera de Aragn (I3A), Tech. Rep., 2010.

[18] E. Nunes, M. Manner, H. Mitiche, and M. Gini, "A taxonomy for task allocation problems with temporal and ordering constraints," *Robotics and Autonomous Systems*, vol. 90, pp. 55–70, 2017, special Issue on New Research Frontiers for Intelligent Autonomous Systems.

[19] C. Nam and D. A. Shell, "Analyzing the sensitivity of the optimal assignment in probabilistic multi-robot task allocation," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 193–200, Jan 2017.

[20] A. Prorok, "Redundant robot assignment on graphs with uncertain edge costs," in *Distributed Autonomous Robotic Systems*. Springer International Publishing, 2019, pp. 313–327.

[21] S. S. Ponda, L. B. Johnson, and J. P. How, "Distributed chance-constrained task allocation for autonomous multi-agent teams," in *American Control Conference (ACC)*, June 2012.

[22] ——, "Risk allocation strategies for distributed chance-constrained task allocation," in *American Control Conference (ACC)*, June 2013.

[23] F. Yang and N. Chakraborty, "Algorithm for optimal chance constrained linear assignment," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 801–808.

[24] L. Zhou and P. Tokekar, "An approximation algorithm for risk-averse submodular optimization," in *2018 International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2018.

[25] F. Yang and N. Chakraborty, "Chance constrained simultaneous path planning and task assignment for multiple robots with stochastic path costs," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[26] M. Albareda-Sambola and E. Fernández, "The stochastic generalised assignment problem with bernoulli demands," *Top*, vol. 8, no. 2, pp. 165–190, Dec 2000.

[27] M. Albareda-Sambola, M. H. van der Vlerk, and E. Fernndez, "Exact solutions to a class of stochastic generalized assignment problems," *European Journal of Operational Research*, vol. 173, no. 2, pp. 465–487, 2006.

[28] D. R. Spoerl and R. K. Wood, "A stochastic generalized assignment problem," Department of Operations Research, Naval Postgraduate School Monterey, California 93943, USA, Tech. Rep., January 2004.

[29] B. Toktas, J. W. Yen, and Z. B. Zabinsky, "Addressing capacity uncertainty in resource-constrained assignment problems," *Computers and Operations Research*, vol. 33, no. 3, pp. 724–745, Mar. 2006.

[30] S. Alaei, M. Hajiaghayi, and V. Liaghat, "The online stochastic generalized assignment problem," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 11–25.

[31] R. L. Carraway, R. L. Schmidt, and L. R. Weatherford, "An algorithm for maximizing target achievement in the stochastic knapsack problem with normal returns," *Naval Research Logistics (NRL)*, vol. 40, no. 2, pp. 161–173, 1993.

[32] M. I. Henig, "Risk criteria in a stochastic knapsack problem," *Operations Research*, vol. 38, no. 5, pp. 820–825, 1990.

[33] M. Sniedovich, "Preference order stochastic knapsack problems: Methodological issues," *The Journal of the Operational Research Society*, vol. 31, no. 11, pp. 1025–1032, 1980.

[34] A. J. Kleywegt and J. D. Papastavrou, "The dynamic and stochastic knapsack problem," *Operations Research*, vol. 46, no. 1, pp. 17–35, 1998.

[35] B. C. Dean, M. X. Goemans, and J. Vondrk, "Approximating the stochastic knapsack problem: The benefit of adaptivity," *Mathematics of Operations Research*, vol. 33, no. 4, pp. 945–964, 2008.

[36] A. Bhalgat, A. Goel, and S. Khanna, "Improved approximation results for stochastic knapsack problems," in *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*. USA: Society for Industrial and Applied Mathematics, 2011, pp. 1647–1665.

[37] J. Kleinberg, Y. Rabani, and E. Tardos, "Allocating bandwidth for bursty connections," in *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, ser. STOC '97. New York, NY, USA: ACM, 1997, pp. 664–673.

[38] A. Goel and P. Indyk, "Stochastic load balancing and related problems," in *40th Annual Symposium on Foundations of Computer Science*, Oct 1999, pp. 579–586.

[39] V. Goyal and R. Ravi, "A ptas for the chance-constrained knapsack problem with random item sizes," *Operations Research Letters*, vol. 38, no. 3, pp. 161–164, 2010.

[40] A. De, "Boolean function analysis meets stochastic optimization: An approximation scheme for stochastic knapsack," in *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2018, pp. 1286–1305.

[41] G. Shabtai, D. Raz, and Y. Shavitt, "A Relaxed FPTAS for Chance-Constrained Knapsack," in *29th International Symposium on Algorithms and Computation (ISAAC 2018)*, vol. 123, Dagstuhl, Germany, 2018, pp. 72:1–72:12.

[42] E. Nikolova, "Approximation algorithms for reliable stochastic combinatorial optimization," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 338–351.

[43] K. Lai and M. Goemans, "The knapsack problem and fully polynomial time approximation schemes," Massachusetts Institute of Technology, Department of Mathematics, Tech. Rep., 3 2006.