Chance Constrained Simultaneous Path Planning and Task Assignment with Bottleneck Objective

Fan Yang[1] and Nilanjan Chakraborty[2]

***Abstract*— We present a novel algorithm for combined task assignment and path planning on a roadmap with stochastic costs. In this problem, the initially unassigned robots and tasks are located at known positions in a roadmap. We want to assign a unique task to each robot and compute a path for the robot to go to the task location. Given the means and variances of travel cost, our goal is to develop algorithms that guarantee that for each robot, with high probability, the total travel cost is below a minimum value in any realization of the stochastic travel costs. We prove that the solution can be obtained by solving (a) a chance-constrained shortest path problems for all robot-task pairs and (b) a linear bottleneck assignment problem in which the cost of an assignment is equal to the optimal objective value of the former problem. We propose algorithms for solving the chance-constrained shortest path problem either optimally or approximately by solving a number of deterministic shortest path problems that minimize some linear combination of means and variances of edge costs. We present simulation results on randomly generated networks and data to demonstrate that our algorithm is scalable with the number of robots (or tasks) and the size of the network.**

I. Introduction

In this paper we are considering multi-robot task allocation problems where robots have to move to target destinations for executing a task. Such problems arise in a number of applications including search and rescue, goods or parts transfer in warehouses or factory floors, automated service vehicles picking up or dropping off people or goods. In such problems the robots have to be assigned to a target destination and a path has to be planned for each robot such that a team performance objective is optimized. We call this problem the simultaneous task allocation and path planning (STAP) problem.

Each robot incurs a cost (time or energy) in executing a task. The overall team performance cost can be a sum of the individual robot costs (also called sum objective) or the maximum of the individual robot costs (also called bottleneck objective). The choice of using sum objective or bottleneck objective is context-dependent. In this paper, we consider the bottleneck objective. Two common contexts in which the bottleneck objective is used are: (a) when we want the robot team to complete their tasks in minimum time and (b) when we want the energy usage in completing the task to be as balanced as possible or the maximum energy usage by any robot to be as small as possible.

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

We assume that the robots move on a graph in an *open environment*, where other (uncontrolled) mobile agents like people or manually operated vehicles can occupy the space. The graph can represent an actual road network or pre-laid paths (e.g., in warehouses) or a roadmap [9], [10], [8] which captures the collision-free configuration space of the robots. Since there are uncontrolled agents in the environment, we assume that robots have local collision avoidance schemes to avoid mobile obstacles. To avoid collisions, robots may have to slow down or deviate locally from their planned paths, thus making travel costs (like time or energy consumed) stochastic. Therefore, we assume that the cost on each edge of the graph is a random variable with known mean and variance. Thus, the cost of any path between a robot-destination pair will be a random variable, hence, the team performance cost is a random variable. Our goal is to compute a solution for the STAP problem that provides probabilistic certificates on the team performance of the multi-robot system.

More precisely, the simultaneous task assignment and path planning (STAP) problem in the presence of uncertainty about the task execution costs is as follows: *Compute the assignment of tasks (targets) to robots as well as paths to reach the tasks and a minimum value (say y) of the team performance objective such that we have a guarantee that the robot team performance will be less than y with high probability (say 0.95) under any realization of the random costs*. Such a solution will provide a probabilistic performance certificate on the performance of the robot team. For example, if we are interested in the time of completion of the tasks by the robots, then y will provide the minimum time by which all robots will complete their tasks with high probability, irrespective of the actual values the random costs take in a given scenario.

Related Work: Technically, the STAP problem is a chance-constrained nonlinear integer optimization problem, which makes it challenging to solve. To the best of our knowledge, there are no available algorithms with theoretical guarantees on solution quality that can solve combined task assignment and planning problem with stochastic costs and bottleneck objective. In our previous work [28], we propose an algorithm for solving the STAP problem with the sum objective. When we consider the bottleneck objective instead of the sum objective, the technical problem becomes quite different. In [28], there is a single chance constraint for the total cost of the robot team, whereas for the bottleneck objective there are multiple chance constraints (one for each robot). As such, the method proposed in [28] cannot solve the problem with multiple chance constraints. In [16], the authors solve

a task allocation problem under uncertainty for analyzing the sensitivity of the optimal assignment with respect to the uncertainty in payoffs. In [23], a redundant robot assignment on graphs with uncertain edge costs is studied. In [29], the authors study chance-constrained discrete submodular maximization problem and provide an algorithm that produces solutions within a constant factor of the optimal and an additive term.

In the extant literature, there has been some effort in solving different variations of the stochastic shortest path problem [1], [2], [11], [13], [14], [15], [18], [19], [21], [22], in which one robot has to plan its motion to a destination node, with random costs on the edges. In [12], the authors considered a stochastic path planning problem for one robot that has to visit a set of nodes in a predefined sequence. Our problem involves both path planning and the task assignment which is not predefined. In the deterministic setting combined goal assignment and collision-free trajectory planning problem has been studied in [24], [25], [20]. The distinction of these papers from our problem is that we consider stochastic costs and our planning is on a discrete structure instead of continuous space.

Contributions: We model the stochastic STAP problem as a chance-constrained combinatorial optimization problem and call the problem chance-constrained simultaneous task assignment and planning (CC-STAP) problem. One *key contribution is to prove* that the optimal solution to CC-STAP can be obtained by solving two related sub-problems, namely, (a) chance-constrained shortest path (CC-SP) problems between all robot-destination pairs and (b) linear bottleneck assignment problem formed from the outputs of the CC-SP problems. This leads to a two-step deterministic approach to solving CC-STAP, which is another key contribution of this paper. We also leverage work on solving chance constrained combinatorial optimization problems [26], [27] and present a novel algorithm for solving the CC-SP problem, which is faster than existing algorithms [18], [17]. We present simulation results demonstrating the scalability of our algorithm for an increasing number of robots and tasks. Due to space constraints we have only presented the statements of the key Lemmas without the proofs. The proofs of the Lemmas are available in a supplementary document [5].

II. Chance Constrained Simultaneous Task Assignment and Path Planning

We now present the multirobot simultaneous task assignment and planning problem formally. We assume that there is a roadmap representing the free configuration space of the robots. Formally the roadmap is modeled as a graph $G = (V, E)$, where V is a set of nodes representing collision free configurations and E is a set of edges, which consist of collision free paths between two nodes. The nodes and edges are collision free with respect to the static obstacles. Given a source-destination pair $s, d \in V$, a path from s to d is a sequence of edges that connect a sequence of distinct nodes. The shortest path between s and d is the path with least cost. Let c_{uv} be the cost of an edge (u, v) and y be the cost of a path. Let x_{uv} be a binary decision variable which is 1 when edge (u, v) is included in the path and 0 otherwise.

Each robot is initially at a given position in the graph denoted by s_i and each task (unassigned initially), denoted by t_j, is located at given position. Let z_{ij} be a binary decision variable with $z_{ij} = 1$ when task t_j is assigned to robot r_i and 0 otherwise. Let ${}^i x_{uv}$ be another binary decision variable where ${}^i x_{uv} = 1$ if edge (u, v) is in the path of r_i, and 0 otherwise. Let $V' = V \setminus \{s_i, t_j\}$ and $\mathcal{N}(u)$ be the neighbors of a node u. Our goal is to assign each robot to a unique task and compute a path for it. The CC-STAP problem is

$$
\begin{aligned}
&\min \quad y \\
&\text{s.t.} \quad \mathbb{P}\left(\sum_{(u,v)\in E} {}^i c_{uv}\, {}^i x_{uv} \le y \right) \ge p, \quad i = 1, \ldots, n. \\
&\sum_{v\in\mathcal{N}(u)} {}^i x_{uv} - \sum_{v\in\mathcal{N}(u)} {}^i x_{vu} = \begin{cases} 1, & \text{if } u = s_i, \\ -z_{ij}, & \text{if } u = t_j, \\ 0, & \forall u \in V', \end{cases} \\
&\qquad\qquad\qquad\qquad i = 1, \ldots, n. \\
&\sum_{i=1}^{n} z_{ij} = 1, \ \forall j; \quad \sum_{j=1}^{n} z_{ij} = 1, \ \forall i; \quad z_{ij} \in \{0, 1\} \\
&{}^i x_{uv} \in \{0, 1\}.
\end{aligned}
\tag{1}
$$

A feasible solution for (1) is a set of paths such that (a) each robot can reach a unique task position through it (constraints in the second and third rows in (1)) and (b) the total path traversal cost for each robot in any realization is less than a value y with at least a pre-specified probability p (set of chance constraints in (1)). Note that there is one chance constraint for each robot. The second set of constraints comprises of one path constraint for each robot. These constraints are slightly different from the path constraints in the shortest path problems and has the variable z_{ij}. If $z_{ij} = 1$, i.e., robot r_i is assigned to task t_j, then the constraints become a path constraint for the robot to go from s_i to t_j. If $z_{ij} = 0$, then it implies that the robot either does not visit that task node or it can pass through the task node on the way to its destination. There are $n|V|$ constraints in the second set of constraints. The constraints for z_{ij} in the third row guarantee that each robot performs a single task and each task can be assigned to only one robot.

Note that the *technical challenge in solving* (1) *directly is that it is an integer nonlinear optimization problem where the nonlinearity arises due to the chance constraints*. In the rest of the paper, the feasible set corresponding to the second constraint and fourth constraint will be denoted by $\mathcal{X}_1$ and the third constraint will be denoted by $\mathcal{X}_2$.

III. Overview of Solution Approach

We will now rewrite the probabilistic constraint in (1) under different assumptions on the random edge costs. If the edge costs ${}^i c_{uv}$ are independent Gaussian random variables, the total path cost for each robot is a Gaussian random variable with mean and variance equal to the sum of means and variances, i.e., $\sum_{u,v} {}^i c_{uv}\, {}^i x_{uv} \sim$

$\mathcal{N}(\sum_{u,v} {}^i\mu_{uv}\, {}^ix_{uv}, \sum_{u,v} {}^i\sigma^2_{uv}\, {}^ix_{uv})$. If we standardize the Gaussian random variable in each chance constraint and let C denote $\Phi^{-1}(p)$, the chance constraint is written as

$$\sum_{(u,v)\in E} {}^i\mu_{uv}\, {}^ix_{uv} + C\sqrt{\sum_{(u,v)\in E} {}^i\sigma^2_{uv}\, {}^ix_{uv}} \leq y,\ \forall i \tag{2}$$

Note that the Gaussian assumption is convenient but not necessary. As long as the inverse cumulative distribution function, Φ^{-1}, is known, the value of C can be computed. Furthermore, for other distributions, we could obtain an upper bound for the optimal objective value by using $C = \sqrt{\frac{p}{1-p}}$ which comes from Chebyshev's inequality.

Let ${}^iy$ denote the CC path cost for robot r_i to its assigned task. Thus $y = \max_i {}^iy$, which implies that the optimal solution to (1) can be obtained by solving

$$\begin{aligned} &\min \max_i\ {}^iy \\ &\text{s.t.} \sum_{(u,v)\in E} {}^i\mu_{uv}\, {}^ix_{uv} + C\sqrt{\sum_{(u,v)\in E} {}^i\sigma^2_{uv}\, {}^ix_{uv}} \leq {}^iy,\ \forall i \\ &{}^ix_{uv} \in \mathcal{X}_1,\ z_{ij} \in \mathcal{X}_2 \end{aligned} \tag{3}$$

Definition 1: In a feasible solution of (3), the robot with the greatest CC path cost, ${}^iy$, is called the *bottleneck robot*.

Lemma 1: Let r_i be the bottleneck robot and d_i be its assigned task location in the optimal solution to Problem (3). Then, the path for bottleneck robot r_i to d_i in the optimal solution is the CC shortest path of robot r_i (corresponding to optimal solution of Problem (4) for robot r_i to destination d_i).

Proof: See supplementary document [5]. ■

Lemma 1 implies that the the optimal solution of problem (3) is also optimal for problem (1). To see this, note that the feasible solutions in problem (3) are always feasible to problem (1) in which $y = \max_i {}^iy$. Hence the optimal solution of problem (3) is feasible to (1). From Lemma 1, the optimal solution of problem (1) can always be converted to a feasible solution to problem (3) with the same objective value, by replacing each path with CC shortest path having same starting point and destination, and ${}^iy$ equal to its CC shortest path. Furthermore, $y = \max_i {}^iy$. Therefore, the optimal solution of problem (3) is also optimal to problem (1).

The above discussion leads to our two-step method to solve (3) which consists of (1) Formulate and solve a chance-constrained shortest path problem for each robot-task pair (r_i, t_j). Thus, a total of n^2 chance-constrained shortest path problems have to be solved at this step. (2) Using the optimal solutions from the CC-SP for each robot-task pair, solve a bottleneck assignment problem.

The first problem to compute the CC shortest paths for a robot-task pair $(r_i,\ t_j)$ is formulated as:

$$\begin{aligned} &\min\ {}^iy_j \\ &\text{s.t.} \sum_{(u,v)\in E} {}^i\mu_{uv}\, {}^ix_{uv} + C\sqrt{\sum_{(u,v)\in E} {}^i\sigma^2_{uv}\, {}^ix_{uv}} \leq {}^iy_j \\ &\quad {}^ix_{uv} \in \mathcal{X}_1 \end{aligned} \tag{4}$$

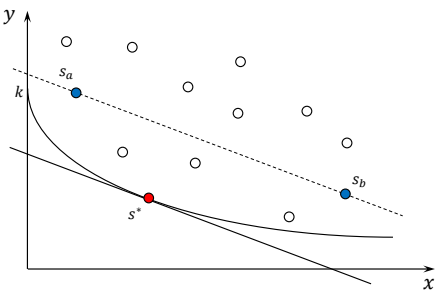

Fig. 1. In the variance-mean plane, any path is a point with coordinates equal to the sum of variances and means of the edge cost in the path. The objective function value in (6) of the path is equal to the vertical intercept of the parabola through the associated point. The optimal path is the point with the smallest vertical intercept of the parabola through it.

Let ℓ_{ij} be the optimal objective value of the problem above, i.e., $\ell_{ij} = \min\ {}^iy_j$. Then the task assignment problem is:

$$\begin{aligned} &\min \max_{\mathrm{i}} \sum_{j=1}^{n} \ell_{ij} z_{ij} \\ &\text{s.t.} \sum_{i=1}^{n} z_{ij} = 1,\ \sum_{j=1}^{n} z_{ij} = 1,\ z_{ij} \in \{0,1\}\ \forall i,j. \end{aligned} \tag{5}$$

Given CC shortest path costs for all robot-task pairs ℓ_{ij}, the problem above is essentially a linear bottleneck assignment problem. The goal is to find proper task assignment to minimize the CC path cost of the bottleneck robot which is same as the objective in problem (3). z_{ij} is the decision variable for the task assignment with the same definition in formulation (1). It is guaranteed that that each robot is assigned to a unique task which is equivalent to $\mathcal{X}_2$ in problem (3). Further, since ℓ_{ij} is obtained from problem (4), a feasible solution in problem (5) also satisfies the chance constraint and $\mathcal{X}_1$ in problem (3). Therefore the feasible solution of problem (5) is feasible to problem in (3) and vice versa. This leads to:

Lemma 2: The optimal solution of problem (3) can be obtained by solving problem (4) and (5).

Proof: See supplementary document [5]. ■

IV. Chance Constrained Shortest Path Problem

The CC-SP problem in (4) can be written as an equivalent mean-risk model as follows:

$$\begin{aligned} &\min \sum_{(u,v)\in E} {}^i\mu_{uv}\, {}^ix_{uv} + C\sqrt{\sum_{(u,v)\in E} {}^i\sigma^2_{uv}\, {}^ix_{uv}} \\ &\text{s.t.}\ \ {}^ix_{uv} \in \mathcal{X}_1 \end{aligned} \tag{6}$$

This is a non-convex combinatorial optimization problem where the number of integer variables, ${}^ix_{uv}$, is equal to the number of edges in the graph. However, this problem can be viewed as an optimization problem in a two-dimensional (2D) plane shown in Fig. 1. Consequently, (6) can be solved exactly by solving a number of deterministic problems that minimize some linear combination of means and variances of the edge cost.

We will first present a geometric view of the optimization problem in (6), as shown in Fig. 1. Let $\{^i x_{uv}\}$ be a feasible solution to (6) (i.e., a path). For each path, we define $\bar{\mu} = \sum_{u,v} {}^i\mu_{uv} {}^i x_{uv}$ and $\bar{\sigma}^2 = \sum_{u,v} {}^i\sigma^2_{uv} {}^i x_{uv}$ as the y and x coordinates of the point corresponding to the path. Thus, any feasible solution (path) is a point on variance-mean plane. For the path $\{^i x_{uv}\}$, we can evaluate the objective in (6) and we call it k. Then $\mu + C\sigma = k$ is a parabola passing through the point $(\bar{\sigma}^2, \bar{\mu})$. Thus, for each point we can associate a parabola passing through it, and all these parabolas are level curves of the objective in (6). The vertical intercept of the parabola is the cost of the path. The optimal solution denoted by s^* is a point on the variance-mean plane with the smallest vertical intercept of the associated parabola.

Lemma 3: The optimal solution, s^* is an extreme point of the lower convex hull of the set of all feasible points on variance-mean plane.

Proof: See supplementary document [5]. ■

Note that although (6) can be viewed as an optimization problem on a 2D point set, the challenge is that the number of points can be quite large (exponential in the number of variables) and thus cannot be enumerated or known a priori. Therefore, we formulate the risk-averse shortest path problem, which can be used to solve CC-SP in (6)

$$\min \sum_{(u,v)\in E} ({}^i\mu_{uv} + \lambda {}^i\sigma^2_{uv}) {}^i x_{uv} \text{ s.t. } {}^i x_{uv} \in \mathcal{X}_1, \ \forall i \quad (7)$$

For a given value of the risk-averse parameter, λ, this problem is essentially a deterministic shortest path problem where the edge cost is a linear combination of the mean and variance. The objective function of (7) on variance-mean plane is a straight line. Thus, minimizing the objective will find an extreme point of the solution set. By Lemma 3, the optimal solution of (6) is an extreme point which can be computed by solving a risk-averse shortest path problem in (7) for some value of λ. The challenge is to find the proper value of λ. In general, we could enumerate all extreme points on variance-mean plan by solving risk-averse problems (7) with different choices of λ in the range $[0, +\infty)$. However, the total number of extreme points could be large and thus enumeration of all extreme points could be expensive.

Let the optimal objective function value for risk-averse problem in (7) with λ_i be denoted by $\mu_i + \lambda\sigma_i^2$. The following lemmas help to find the upper bound of $\bar{\lambda}$.

Lemma 4: Let $\bar{\lambda}$ be a risk-averse parameter and ${}^i x^*_{uv}$ be the optimal solution of the risk-averse problem (7) with $\bar{\lambda}$. Further define $\bar{\sigma} = \sqrt{\sum_{u,v} {}^i\sigma^2_{uv} {}^i x^*_{uv}}$. The risk-averse parameter $\bar{\lambda}$ is an upper bound of the optimal risk-averse parameter when the condition $\bar{\lambda}\bar{\sigma} = C$ is satisfied.

Proof: See supplementary document [5]. ■

A. Algorithm for Solving Chance Constrained Shortest Path

Alg. 1 finds such upper bound for λ. It starts with solving problem (7) with λ_k equal to 0 (line 2). The obtained solution is stored in the solution set S_1. Further, the coordinate associated with this solution (σ_k^2, μ_k) on the variance-mean plane is obtained (line 3). Then we check if $\lambda_k\sigma_k = C$. If no, the risk-averse parameter is updated as $\lambda_{k+1} = \frac{C}{\sigma_k}$ and we go back to step 2 (lines $4-5$). If yes, the current λ_k is the upper bound for the risk-averse parameter. The algorithm terminates in a finite number of iterations. Note that the optimal solution of the risk-averse parameter equal to $\bar{\lambda}$ is a nearly optimal solution (see Section V).

In the second part of our algorithm, we enumerate all extreme points in the search region restricted by $\bar{\lambda}$ by solving the risk-averse problem with $\lambda \in (0, \ \bar{\lambda}]$. The procedure is presented in Alg. 2. The input is the solution set S_1 obtained from the previous step including all extreme points computed thus far. These points are all in the search region. We need to find the rest of the extreme points. More precisely, we should find extreme points below the straight line connecting any consecutive pair of solutions in S_1 if it exists, e.g., (s_a, s_b) in Figure 1. We create a point pair set S'_i which is initialized as $S'_0 = \{(s_0, s_1), ...(s_{k-1}, s_k)\}$ (line 1). In any iteration, say i, we first create a empty point pair set S'_{i+1} storing the point pairs for the next iteration (line 3). For each point pair in S'_i, say (s_{j-1}, s_j), the risk-averse parameter λ is the negative of the slope of the line connecting s_{j-1} and s_j on variance-mean plane, i.e., $\lambda = -\frac{\mu_j - \mu_{j-1}}{\sigma_j - \sigma_{j-1}}$ (line 4). The algorithm solves the risk-averse problem with λ (line 5). If the obtained solution s is different from s_{j-1} and s_j, the solution s is a new extreme point which is then stored in the set S_1. Two new point pairs (s_{j-1}, s) and (s, s_j) are stored in S'_{i+1} (lines $6-7$). If the obtained solution is same as s_{j-1} or s_j, there is no extreme point below the line through s_{j-1} and s_j. After solving all the risk-averse problems generated from S'_i, we check whether the updated S'_i is empty. If it is not empty, or in other words, there is a new extreme point obtained from the previous step, we move on to the next iteration and repeat the same procedures starting from line 3. We stop when S'_i is empty, which means no new extreme point is obtained and we have found all extreme points in the search region. The optimal solution can be obtained from the extreme point set S_1 (lines $12-13$).

Algorithm 1 Find the upper bound for λ

Input: $C, {}^i\mu_{uv}, {}^i\sigma_{uv}$.
Output: $\bar{\lambda}, S_1$.
1: Initialization: let $k = 0, \lambda_k = 0$.
2: Solve problem (7) with λ_k.
3: Compute (σ_k^2, μ_k) and store it to solution set S_1.
4: **if** $\lambda_k\sigma_k \neq C$ **then**
5: Update $\lambda_{k+1} = \frac{C}{\sigma_k}$ and go back to step 2.
6: **else**
7: **return** $\bar{\lambda} = \lambda_k$ and S_1.

The second sub-problem that we have to solve is the linear bottleneck assignment problem in which the greatest CC path cost in the assignment should be minimized. It is a well-studied problem and there are many algorithms to solve it [6], [7], [3], [4]. In this paper, we use a threshold algorithm published in [7].

Algorithm 2 Compute the optimal solution of CC-LAP

Input: Solution set S_1.

Output: Optimal solution s^* and optimal objective value y^* for CC-LAP.

Compute point pair set $S'_0 = \{(s_1, s_2), (s_2, s_3), ..., (s_{k-1}, s_k)\}$ (suppose $\lambda_k = \bar{\lambda}$).
while S'_i is not empty **do**
  Create empty point pair set S'_{i+1}
  for each point pair in S'_i, say (s_{j-1}, s_j) **do** compute risk-averse parameter $\lambda = -\frac{\mu_j - \mu_{j-1}}{\sigma_j^2 - \sigma_{j-1}^2}$.
    Solve risk-averse problem (7) with λ.
    if the obtained solution say $s \notin \{s_{j-1}, s_j\}$ **then**
      Store s in S_1 and store point pairs (s_{j-1}, s), (s, s_j) in S'_{i+1}.
    end if
  end for
  $i = i + 1$.
end while
$s^* = \arg\min_{s_j \in S_1} \ \mu_j + C\sigma_j$, $y^* = \min_{s_j \in S_1} \ \mu_j + C\sigma_j$
return s^*, y^* =0

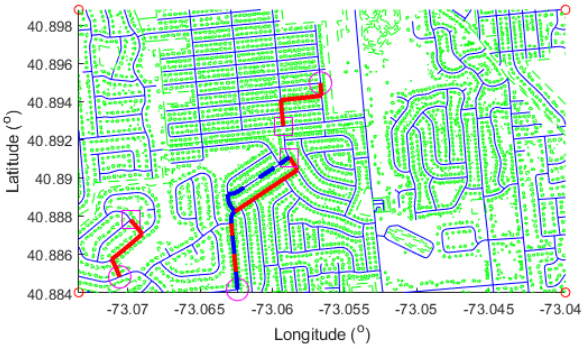

Fig. 2. A CC-STAP example. The alarms go off in several places in the roadmap marked by squares. A team of robots are initially located at places marked by circles. The computed assignments and paths are shown in red.

V. Simulation Results

We will first present a case study in a multi-robot emergency response scenario. A team of robots are located at the positions marked by circles while the alarms go off in places marked by squares in Fig. 2. The travel time for a robot to a task location is dependent on the traffic condition, hence, it is a random variable. We need to assign a task to each robot and compute their path to the alarm location. Our algorithm computes the earliest time such that each robot is guaranteed to finish the job within the time window with a high probability.

The region in Fig. 2 is selected from OpenStreetMap (OSM). The means and variances of the travel cost of the edges are generated based on the distance and tags of the roads from the OSM file (e.g., edge with tag highway has smaller variance). The probability, p, is set as 0.9. The solutions of our algorithm are shown as red paths in Fig. 2. Under the same setup, we also solve the problem by optimizing the expectation of the travel cost. The path for the bottleneck robot (the blue dashed line) is different from our solution. For both methods, we use Monte Carlo simulation to compute the probability that the travel time of each robot is within the time window obtained. The probabilities for the bottleneck robot (which takes the longest time) are shown in Fig. 3. The probability for our solution converges to 0.90 while the probability converges to 0.5 for the method optimizing the expectation. In other words, in a particular realization, the robot may take longer time than the optimal expected time window to attend the emergency. The paths and time windows obtained by optimizing expectation might not be a robust solution to the application in which the safety or success rate are important. This is one of the advantages of using the chance constrained formulation.

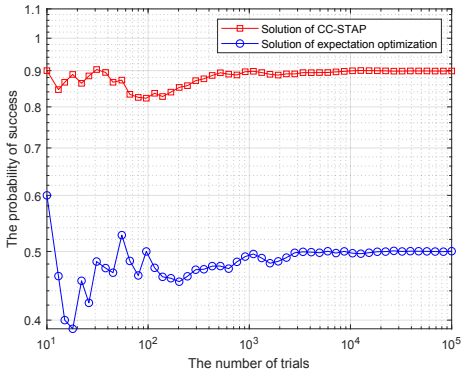

Fig. 3. The probability that the travel times of the bottleneck robot do not exceed the optimal time window obtained by solving CC-STAP and optimizing expected travel time.

Besides the case study, we perform extensive simulations to test our algorithm in more general scenarios (e.g., larger number of robots, size of map). The computational cost of our algorithm is $\sum_{i=1}^{n} T_i + T'$ where n is the number of robots (tasks), T_i is the computational cost of each robot, T' is the computational cost for solving a linear bottleneck assignment problem. The computational cost of each robot T_i is equal to $\bar{T} \cdot \sum_{j=1}^{n} K_j$ where K_j is the number of deterministic shortest path problems solved for solving a CC shortest path problem from r_i to task t_j and $\bar{T}$ is the computational cost for solving the shortest path problem, which is $O(|E| + |V| \log |V|)$ for Dijkstra's algorithm.

A key efficiency measure of our algorithm is the number of deterministic shortest path problems solved by a single robot, i.e., $\sum_{j=1}^{n} K_j$. However, this number is problem parameter dependent and it is hard to give *a priori* bounds. We will plot the values of $\sum_{j=1}^{n} K_j$ with different number of robots and the size of graph. We show through extensive simulations that: (a) our algorithm is scalable with the number of robots (tasks) and the size of the maps. (b) the subroutine we propose to solve CC shortest path problem is scalable with the number of robots (tasks) and the size of the maps. (c) With only the first step of our algorithm in Alg. 1, we can compute nearly optimal solutions.

The simulations were done on a computer with Intel

$i7$ 2.60GHZ CPU and 16GB RAM. We assume that the number of robots is same as the number of tasks. The desired probability p in the chance constraints is 0.99. We created different instances with randomly generated means and variances for edge cost, i.e. ${}^i\mu_{uv}, {}^i\sigma^2_{uv}$. The means are generated from a continuous uniform distribution $\mathcal{U}(20, 100)$ and the variances are generated from a continuous uniform distribution with different magnitude of the range so that the edge with higher mean also tends to have a higher variance. We test both the proposed algorithms, namely, the exact algorithm and the algorithm in which the subroutine that solves CC shortest path problem implements only the first step of our algorithm, namely, the approximate algorithm.

Scalability with the number of robots: In the first set of simulations, we study the scalability of our algorithm with the number of robots. The number of robots vary from 20 to 100 with increment of 10. Robots are working on a graph with 2500 nodes and 27099 edges. The results are provided in Fig. 4. Each data point is obtained from 100 instances with randomly generated ${}^i\mu_{uv}, {}^i\sigma^2_{uv}$ For a certain number of robots ($x-$coordinate) in plot (a), we compute both the average and the maximum number of deterministic problems solved by a single robot. The average and the maximum number for the exact algorithm are presented by blue circles and crosses. The results for the approximate algorithm are presented by red circles and crosses. It is shown that the average numbers for both exact and approximate algorithm grow almost linearly. It implies that the average number of deterministic problems solved for the CC shortest path problem is close to a constant, irrespective of the number of robots. The approximate algorithm solves smaller number of deterministic problems than the exact algorithm. The maximum numbers are less than 400 and are scalable to the number of robots. The plot (b) presents the average and the maximum relative difference of the cost value between the exact solution and the approximate solution. The average differences are all less than 1% and the highest difference over a total of 1000 scenarios in this set of simulations is less than 8%. Hence the solution obtained from the approximate algorithm is nearly optimal. The actual running time varies between 3 to 300 seconds as the number of robots increase.

Scalability with the size of the graph: In the second set of simulations, we study the scalability of our algorithms with the number of nodes of the roadmap. The number of nodes graphs vary from 500 to 2500 with increment of 500. The number of edges increase from about 8000 to 27000 accordingly. The number of robots is 50. The results are presented in Fig. 5. For a given number of nodes, we generate 100 different graphs with randomly generated ${}^i\mu_{uv}, {}^i\sigma^2_{uv}$. Similar to the first set of simulations, we compute the number of deterministic shortest path problems solved by a single robot. As shown in Fig. 5 (a), the average numbers of deterministic problems solved by both the exact and the approximate algorithm are nearly constant. It implies that the size of the graph does not have great influence on the number of deterministic problems solved by each robot.

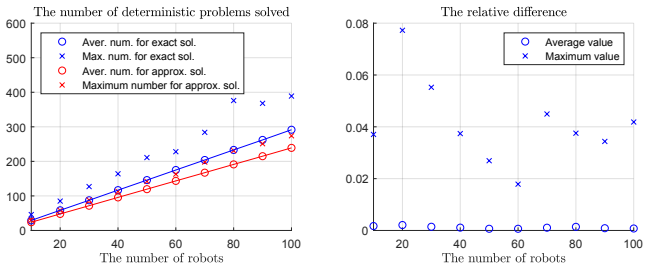

Fig. 4. Scalability with the number of robots: (a) The number of deterministic shortest path problems solved by a single robot and (b) the relative difference of the travel costs obtained by our exact algorithm and approximate algorithm.

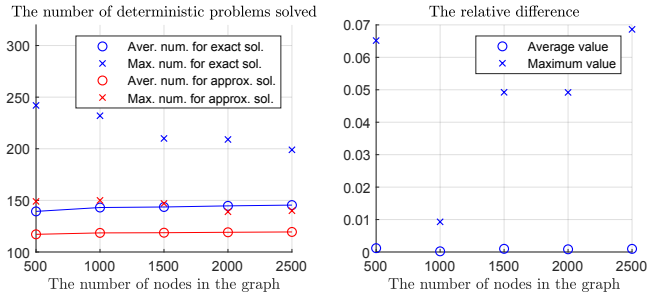

Fig. 5. Scalability with the number of nodes of the graph: (a) The number of deterministic shortest path problems solved by a single robot and (b) the relative difference of the travel costs obtained by our exact algorithm and approximate algorithm.

Fig. 5 (b) shows that our approximate solution is nearly optimal (relative difference of less than 1% on average). The actual running time varies from 60 to 80 seconds as the number of nodes increase.

VI. Conclusion

In this paper, we present algorithms to solve the simultaneous task assignment and path planning problem under stochastic travel costs. We formulate the problem as a chance-constrained combinatorial optimization problem, called CC-STAP. We prove that the CC-STAP problem is equivalent to a linear bottleneck assignment problem in which each assignment cost is equal to the CC shortest path cost for each robot-task pair. We propose an algorithm to solve the chance-constrained shortest path by solving a number of deterministic shortest path problems. Our approach is able to compute both exact solution (assuming the random edge costs are independent and Gaussian) and approximate solution. The simulation results demonstrate that our algorithm is scalable with the number of robots as well as the size of the road network. Further, simulation results also demonstrate that the approximate algorithm obtains a nearly optimal solution. Although we provided empirical evidence that our algorithm runs reasonably fast, a limitation of our current work is that there is no theoretical computational complexity guarantees of our algorithm. In the future we would like to study this question.

REFERENCES

[1] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. II*. Athena Scientific, 3rd edition, 2007.

[2] J. Boyan, M. Mitzenmacher, and M. Mitzenmacher. Improved results for route planning in stochastic transportation. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 895–902, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

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

[4] U. Derigs and U. Zimmermann. An augmenting path method for solving linear bottleneck assignment problems. *Computing*, 19(4):285–295, Dec 1978.

[5] Fan Yang and Nilanjan Chakraborty. Supplementary Document:. `http://bit.ly/ICRA21_STAP`, 2020.

[6] D. Fulkerson, I. Glicksberg, and O. Gross. *A Production Line Assignment Problem*. Research memorandum. Rand Corporation, 1953.

[7] R. S. Garfinkel. An improved algorithm for the bottleneck assignment problem. *Operations Research*, 19(7):1747–1751, 1971.

[8] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[9] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, Aug 1996.

[10] S. M. Lavalle, J. J. Kuffner, and Jr. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308, 2000.

[11] S. Lim, H. Balakrishnan, D. Gifford, S. Madden, and D. Rus. Stochastic motion planning and applications to traffic. *The International Journal of Robotics Research*, 30(6):699–712, 2011.

[12] S. Lim and D. Rus. Stochastic motion planning with path constraints and application to optimal agent, resource, and route planning. In *2012 IEEE International Conference on Robotics and Automation*, pages 4814–4821, May 2012.

[13] S. Lim, C. Sommer, E. Nikolova, and D. Rus. Practical route planning under delay uncertainty: Stochastic shortest path queries. In *Robotics: Science and Systems*, pages 249–256, United States, 1 2013. MIT Press Journals.

[14] E. D. Miller-Hooks and H. S. Mahmassani. Least expected time paths in stochastic, time-varying transportation networks. *Transportation Science*, 34(2):198–215, 2000.

[23] A. Prorok. Redundant robot assignment on graphs with uncertain edge costs. In *Distributed Autonomous Robotic Systems*, pages 313–327. Springer International Publishing, 2019.

[15] J. Mote, I. Murthy, and D. L. Olson. A parametric approach to solving bicriterion shortest path problems. *European Journal of Operational Research*, 53(1):81 – 92, 1991.

[16] C. Nam and D. A. Shell. Analyzing the sensitivity of the optimal assignment in probabilistic multi-robot task allocation. *IEEE Robotics and Automation Letters*, 2(1):193–200, Jan 2017.

[17] E. Nikolova. Approximation algorithms for reliable stochastic combinatorial optimization. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 338–351, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[18] E. Nikolova, J. A. Kelner, M. Brand, and M. Mitzenmacher. Stochastic shortest paths via quasi-convex maximization. In *Proceedings of the 14th Conference on Annual European Symposium - Volume 14*, ESA'06, pages 552–563, London, UK, UK, 2006. Springer-Verlag.

[19] S. Pallottino and M. G. Scutellà. Shortest path algorithms in transportation models: Classical and innovative aspects. In *Equilibrium and Advanced Transportation Modelling*, pages 245–281, Boston, MA, 1998. Springer US.

[20] D. Panagou, M. Turpin, and V. Kumar. Decentralized goal assignment and trajectory generation in multi-robot networks: A multiple lyapunov functions approach. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6757–6762, May 2014.

[21] C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84(1):127 – 150, 1991.

[22] G. H. Polychronopoulos and J. N. Tsitsiklis. Stochastic shortest path problems with recourse. *NETWORKS*, 27:133–143, 1996.

[24] M. Turpin, N. Michael, and V. Kumar. Trajectory planning and assignment in multirobot systems. In E. Frazzoli, T. Lozano-Perez, N. Roy, and D. Rus, editors, *Algorithmic Foundations of Robotics X*, pages 175–190, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[25] M. Turpin, N. Michael, and V. Kumar. Capt: Concurrent assignment and planning of trajectories for multiple robots. *The International Journal of Robotics Research*, 33(1):98–112, 2014.

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

[27] 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)*, pages 1043–1049, May 2018.

[28] 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.

[29] 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.