Algorithms for Chance-constrained Multi-robot Task Allocation

A Dissertation Presented

by

Fan Yang

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Department of Mechanical Engineering

Stony Brook University

December 2020

Stony Brook University

The Graduate School

Fan Yang

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation.

Dr. Nilanjan Chakraborty - Dissertation Advisor
Assistant Professor, Department of Mechanical Engineering

Dr. Qiaode Jeffrey Ge - Chairperson of Defense
Professor, Department of Mechanical Engineering

Dr. Anurag Purwar - Committee Member
Research Associate Professor, Department of Mechanical Engineering

Dr. Joseph S.B. Mitchell - Committee Member (Outside)
Distinguished Professor, Department of Applied Mathematics and Statistics

Dr. Jie Gao - Committee Member (Outside)
Professor, Department of Computer Science, Rutgers University

This dissertation is accepted by the Graduate School

Eric Wertheimer

Dean of the Graduate School

Abstract of the Dissertation

Algorithms for Chance-constrained Multi-robot Task Allocation

by

Fan Yang

Doctor of Philosophy

in

Department of Mechanical Engineering

Stony Brook University

2020

Coordinating multi-robot systems (MRS) have a wide range of application domains including collaborative autonomous manufacturing, automated transport of goods in warehouses and factory floors, environmental monitoring, search and rescue, surveillance, and sensing data collection. Two fundamental problems arise naturally in the autonomous operation of the multi-robot system: (a) multi-robot task allocation; (b) multi-robot path planning. Generally speaking, multi-robot task allocation solves the following problem: given a set of robots and tasks with each robot obtaining a certain payoff for each task, compute a subset of tasks for each robot such that a team performance criterion is optimized subject to a set of constraints on the tasks and/or the robots. Multi-robot path planning solves the following problem: given the initial positions of a set of robots and target positions of the spatially distributed tasks, compute a collision-free path for each robot from its initial position to its target position.

Most current works on multi-robot task allocation consider as deterministic parameters like robot-task payoffs, resource consumption of robots performing tasks. However, these parameters may not be known exactly in practice, especially in the open environment in which there are other (uncontrolled) moving entities like humans or human-operated vehicles that occupy the space. One goal of this thesis is to formulate multi-robot task allocation

problems with uncertain parameters and develop solution algorithms for them. Further, in the extant literature, task allocation and path planning problems are commonly considered separately. In particular, it is assumed in task allocation that there are planned paths for all robot-task assignments and thus the payoff or the resource consumption to perform a task is available. In path planning, it is assumed that the tasks are assigned to robots and target positions are thus specified. When both problems have to be solved, the decoupling approach (solving two problems separately) is usually used. Although conceptually convenient, the decoupling approach may lead to sub-optimal solutions. Therefore another goal of this thesis is to simultaneously compute the allocation of tasks to robots as well as plan the path from the initial position of robots to the destinations.

We address five related problems of multi-robot task allocation under uncertainty: (a) task allocation with stochastic payoffs; (b) simultaneous task allocation and path planning with the sum objective and stochastic travel cost; (c) simultaneous task allocation and path planning with the MinMax objective and stochastic travel cost; (d) stochastic knapsack problem with application to multi-robot teaming; (e) generalized assignment problem with stochastic resource consumption. We formulate these problems as chance-constrained combinatorial optimization problems, which provide us the flexibility to tradeoff between performance and reliability. However, solving these problems directly is challenging. The main contribution of this thesis is developing a unified methodology of solving the task allocation with uncertainty in payoffs and resource consumption. By analyzing the problems on a two-dimensional plane called the variance-mean plane, we connect the chance-constrained problem to a deterministic version of the problem, called the risk-averse problem. As long as the deterministic problems can be solved with a certain guarantee, the solutions of the chance-constrained problems are proved to have the same guarantee.

Dedication

To my grandfather.

Acknowledgements

My deepest gratitude goes first to my advisor, Professor Nilanjan Chakraborty, for his mentorship, patient support, and continuous encouragement. His guidance helped me during all of the research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D. study.

I would like to express my thanks to my committee members, Professor Qiaode Jeffrey Ge, Professor Anurag Purwar, Professor Joseph S.B. Mitchell, and Professor Jie Gao for taking their precious time in attending my presentation, their insightful feedback and comment on my proposal.

Over the past several years, I have been fortunate to be in a wonderful lab. I greatly appreciate my colleagues from Interacting Robotic Systems Laboratory, Jiayin Xie and Anirban Sinha. Thanks also to my friend, Jiayuan Zhang.

Finally, I sincerely thank my parents, Hui Yang, Yanping Zhao, my uncle, Rui Zhao and my girlfriend, Ke Lu for giving me unconditional and endless love that I cherish most. I would not be here without their support and encouragement.

Table of Contents

List of Figures

Figure

List of Abbreviations

CC-LAP	**C**hance-**c**onstrained **L**inear **A**ssignment **P**roblem
RA-LAP	**R**isk-**a**verse **L**inear **A**ssignment **P**roblem
CC-STAP	**C**hance-**c**onstrained **S**imultaneous **T**ask **A**ssignment and **P**ath planning
RA-STAP	**R**isk-averse **S**imultaneous **T**ask **A**ssignment and **P**ath planning
CC-STAP-B	**C**hance-**c**onstrained **S**imultaneous **T**ask **A**ssignment and **P**ath planning with MinMax (**B**ottleneck) objective
RA-STAP-B	**R**isk-**a**verse **S**imultaneous **T**ask **A**ssignment and **P**ath planning with MinMax (**B**ottleneck) objective
LBAP	**L**inear **B**ottleneck **A**ssignment **P**roblem
CC-KNAP	**C**hance-**c**onstrained **K**napsack **P**roblem
RA-KNAP	**R**isk-averse **K**napsack **P**roblem
CC-GAP	**C**hance-**c**onstrained **G**eneralized **A**ssignment **P**roblem
RA-GAP	**R**isk-**a**verse **G**eneralized **A**ssignment **P**roblem

Chapter 1

Introduction

Coordinating multi-robot systems (MRS) have a wide range of application domains including collaborative autonomous manufacturing, automated transport of goods in warehouses and factory floors, environmental monitoring, search and rescue, surveillance, and sensing data collection [1, 2]. Multi-robot task allocation (or task assignment) and multi-robot path planning are two key classes of problems that arise in the autonomous operation of coordinating multi-robot systems.

Generally speaking, in multi-robot task allocation one solves the following: *Given a set of robots and a set of tasks with each robot obtaining some payoff (or incurring some cost) for each task, compute a subset of (usually spatially distributed) tasks for each robot such that a team performance criterion is optimized subject to a set of constraints on the tasks and/or the robots.* Multi-robot task allocation problems have been extensively studied in robotics with underlying roots in operations research, management science, and computer science. The focus in robotics has been on multi-robot task allocation problems where all problem parameters like the robot-task payoff or energy consumption of the robots for performing a task are deterministic. However, in practice, these parameters may not be known exactly. *Thus, one focus of this thesis is to formulate multi-robot task allocation problems with parameter uncertainty and develop solution algorithms for them.*

In multi-robot path planning, one solves the following problem: *Given the initial and goal positions of a set of robots, compute a collision-free path for each robot from its initial position to its goal position.* Multi-robot path planning has been widely studied in the

robotics literature. Much of the work in the extant literature has focused on a *closed environment* where the only moving objects are the robots. However, in many applications (like a factory floor or a warehouse), robots may work in an *open environment*, where there are other moving entities like humans or human-operated vehicles. In this thesis, we assume that robots will be working in an open environment.

For spatially distributed tasks, the robots have to both allocate the tasks among themselves as well as plan a collision-free path from their initial locations to the locations of the tasks allocated to them. However, in the extant literature, the task allocation and the path planning problems are usually considered separately. When solving the task allocation problem, it is usually assumed that there is a planned path from the robot location to the task location, which implies that one can compute the energy consumed to perform the task. On the other hand, when path planning problems are formulated, it is assumed that the goal locations (that arise from the task locations) are known. Thus, in practice, when both problems have to be solved, some inputs for the task allocation problem as well as the path planning problem are unknown. One possibility is to use a sequential approach, which is usually used in practice. First, some reasonable path cost is assumed and the task allocation problem is solved to compute the tasks (or targets) for each robot. Then the multi-robot path planning problem is solved with the positions of the allocated tasks as goals. The choice of path costs without knowing the path can lead to sub-optimal solutions. Therefore, in this thesis, *we also focus on simultaneously computing the allocation of tasks to robots as well as planning their paths as a single problem. This is done while considering the stochasticity of the path costs and assuming that the robots operate in an open environment.*

1.1 Motivating Applications

There are different types of multi-robot task allocation problems that have been studied in the literature. Point-to-point transfer of goods, parts or people is a class that has applications in factory floors, automated warehouses, and mobility-on-demand with autonomous

vehicles. In this thesis, we focus on task allocation problems that arise from point-to-point transfer of objects. We will now present two motivating examples, one from the object transfer domain and another from the mobility-on-demand domain.

1.1.1 Point-to-point Object Transfer

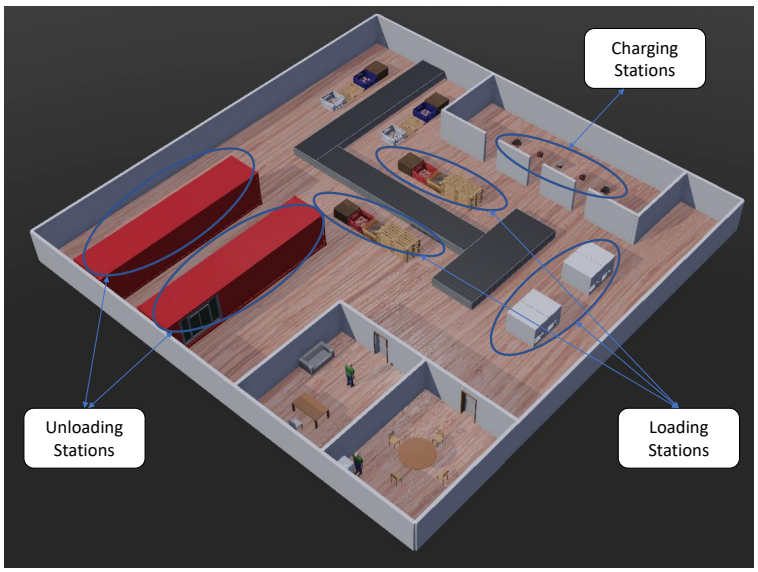


Figure 1.1: Multi-robot package delivery. The robots drive around the facility picking up packages from the loading stations and delivering them to unloading stations for storing or processing.

One motivating example is multi-robot packages delivery in an automated factory shown in Fig. 1.1. A number of packages are located at the loading stations of the central store. Each robot has to pick up packages from the loading station and deliver it to the unloading station for storing or processing. Each robot has an energy (resource) budget or battery life and consumes a certain amount of energy to deliver a package. Meanwhile, each robot obtains a certain payoff for delivering packages successfully. Influenced by the uncertain environment, either payoffs or resource consumption could be uncertain. The goal is to simultaneously assign packages to robots and compute paths for robots to deliver packages to the task locations such that the total payoffs are maximized and the resource constraint for each robot is not violated with a probabilistic guarantee.

1.1.2 Mobility-on-demand

Figure 1.2: Mobility-on-demand. A number of passengers are initially located at several places in the road map marked by squares. A team of autonomous robots are initially located at places marked by circles. Each robot should be assigned to a passenger and compute a path to pick up the assigned passenger. The time for robots moving to the target location is uncertain. The goal is to have the minimum time window that the robot team picks up all passengers with a high probabilistic guarantee.

As shown in Fig. 1.2, there is a team of autonomous robots located at places marked by circles and a number of passengers located at places marked by squares. Each robot should be assigned to a passenger and compute a path to the location of the assigned passenger. Influenced by the uncertain traffic condition, the time for a robot moving to the assigned location is uncertain. We consider the makespan of the robot team. The goal is to have the minimum amount of time such that the robot team picks up all passengers with a probabilistic guarantee.

1.2 Problem Setup and Assumptions

The payoff, performance criteria, and the constraints are the key components of any multi-robot task allocation problem. The *payoff* could be (a) the cost for a robot to perform a task such as the energy consumption, travel time and distance to reach a destination; (b) benefit for completing a task; (c) suitability, a characterization of "how well" a robot is

capable to perform a task (e.g., in automated factory scenario robot with high stability is suitable for the fragile part); (d) priority, the urgency of performing a task (e.g., in search and rescue scenario saving life is the first priority). In general, multi-robot task allocation considers combinations of those factors. For example, the payoffs of robots delivering the packages in Fig. 1.1 could be the combination of the benefit of packages and the traveling cost for robot visiting loading and unloading stations.

The *performance criteria* could be (a) the overall system performance, e.g, the sum of the individual utility. (b) The performance of a subset of robots. For example, in the package delivery system, the on-time delivery time span of the robot team is essentially the delivery time of the slowest robot. This is also known as the MinMax criterion.

The *constraints* on robots and among tasks naturally arise in multi-robot task allocation applications. It could be (a) resource constraint that the number of the tasks to perform by the robot is bounded by resource consumption; (b) task ordering constraints, e.g., a task must be performed before or after a set of others; (c) task deadline constraints that guarantee that a task is completed within a time window. Tasks may also have coupling constraints that two or more tasks must be performed at the same time.

This thesis is motivated by the multi-robot task allocation applications where the robots execute the tasks by visiting sites in the *open environment* where other (uncontrolled) mobile agents like people or manually operated vehicles can occupy the space. In this scenario, the robots have to navigate in the environments containing both static and dynamic obstacles.

We assume that there is a graph that represents the actual road network or pre-laid paths or a roadmap [3, 4, 5] which captures the collision-free configuration space of the robots (see Fig. 1.3). The weights of edges represent the stochastic traveling cost. It is assumed that the mean and variance of the random cost are known. Further, each robot is assumed to have a local collision avoidance scheme.

The setup of the multi-robot task allocation in this thesis is presented in Fig. 1.4.

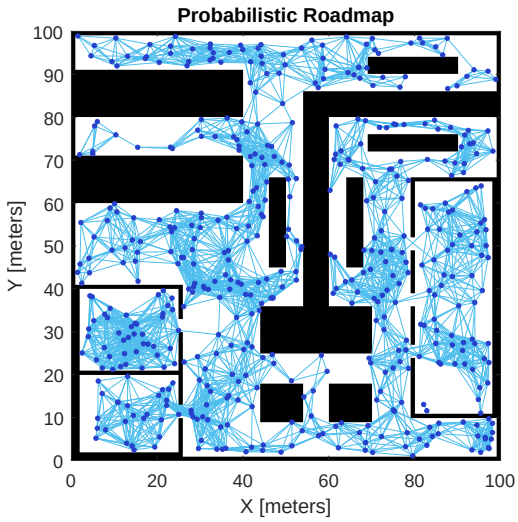


Figure 1.3: The probabilistic roadmap (PRM) of the warehouse in Fig. 1.1.

The system takes as input the task description, the roadmap of the environment, and the available resource for each robot. The task scheduler implements our algorithm and sends the delivery commands to each robot such that the system performance is optimized. Each robot moves along the path provided by the task scheduler and avoids the obstacles by the local collision avoidance scheme. The task scheduler can be either a stand-alone computing unit or be partially integrated into each robot.

1.3 Contributions of the Thesis

In this thesis, I have studied two problems of multi-robot task allocation, namely, (a) chance-constrained simultaneous task assignment and path planning for multiple robots with stochastic path costs (CC-STAP) [6, 7]; (b) multi-robot chance-constrained generalized assignment problem with stochastic resource consumption (CC-GAP) [8].

In CC-STAP, the initially unassigned robots and tasks are located at known positions in a roadmap or graph (e.g., charging stations in Fig. 1.1). The traveling costs of edges are random variables with known means and variances. We want to assign a unique task to each robot and compute a path for the robot to go to its assigned task location (e.g., the path connecting the robot's charging station, the loading and unloading stations of the

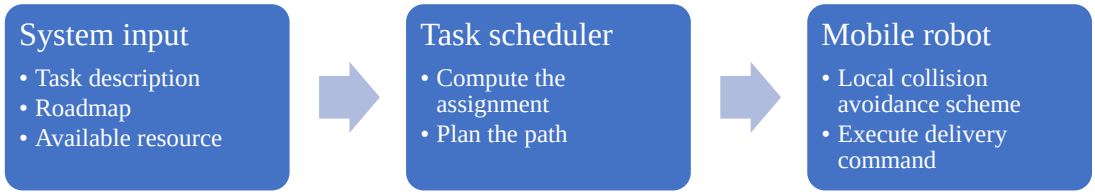


Figure 1.4: The setup of of the multi-robot task allocation in this thesis.

assigned package, see Fig. 1.1). The goal is to optimize the performance of the robot team with a probabilistic guarantee. CC-STAP arises mainly in multi-robot application scenarios where robots should drive around an environment, such as package delivery, search and rescue, automated service vehicles. People may have different demands on the performance of the multi-robot system and the performance criteria thus can be different. I have studied CC-STAP with two different team performance criteria. The first team performance is the sum of the individual robot costs (also called sum objective) [7]. The other one is the maximum of the individual robot costs (also called bottleneck objective and CC-STAP-B is the abbreviation of such problem) [6]. In particular, CC-STAP-B applies to two common scenarios (a) when the robot team should complete their tasks in a minimum time window; (b) when the energy usage in completing the task should be as balanced as possible or the maximum energy consumption by any robot should be as small as possible.

In CC-GAP, each robot has a resource constraint (e.g., battery life) and need to consume a certain amount of resource to obtain a payoff for performing a task. A robot can perform multiple tasks as long as the resource constraint is not violated. The resource consumed for performing a task is a random variable with known mean and variance. The goal is to find an assignment of robots to tasks with maximum team payoff such that each task is assigned to at most one robot and the resource constraint for each robot is not violated with very high probability [8]. Although the path planning is not considered (or the mean and variance of path cost are assumed to be known) in CC-GAP, the task in CC-GAP could have a more general definition than CC-STAP and is applicable to the context like material

handling, assembly, besides the pick-and-place type of tasks.

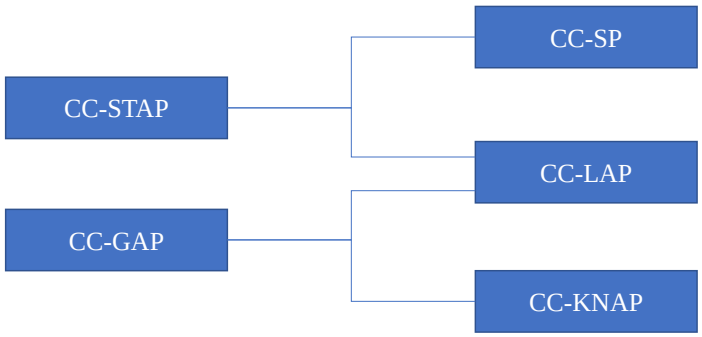


Figure 1.5: The roadmap of the thesis.

As shown in Fig. 1.5, the methods to solve CC-STAP and CC-GAP are built upon our work on several basic multi-robot task allocation and path planning problems with stochastic variables, which are the special cases of those two major problems. First, chance-constrained linear assignment problem (CC-LAP) [9] is a simpler version of CC-STAP in which the mean and variance of the payoffs of robots performing tasks are given. Each robot performs exactly one task, which is a special case of CC-GAP. Second, chance-constrained shortest path problem (CC-SP) which considers finding the path between a specific initial position and target position. The obtained path cost should be less than a minimized value with high probability. Third, chance-constrained knapsack problem [10] is a special case of CC-GAP in which only one robot is involved. However, CC-KNAP also arises in multi-robot application scenarios such as multi-robot patrolling in which a team of robots should be selected from the given set such that a certain route is covered by robots navigating one by one with high probability.

This thesis work could be organized in Table 1.1 with four columns, namely, (a) objective, the performance criteria for the robot team; (b) uncertainty, which components of the optimization problem are uncertain; (c) resource budget, whether the resource constraints are considered; (d) number of chance constraints (CC), how many chance constraints are involved.

Below I give a brief summary of the thesis contribution of these problems of multi-robot

Table 1.1: The characterization of multi-robot task allocation problems in this thesis

	Objective	Uncertainty	Resource Budget	Number of CC
CC-LAP	Sum	Objective	No	Single
CC-STAP	Sum	Objective	No	Single
CC-STAP-B	MinMax	Objective	No	Multiple
CC-KNAP	Sum	Constraint	Yes	Single
CC-GAP	Sum	Constraint	Yes	Multiple

task allocation under uncertainty.

(1) In CC-LAP, we design centralized and distributed algorithms that solve CC-LAP optimally. The primary contributions are the development of the connection between the value-at-risk problem and the risk-averse problem for linear assignment and the deterministic method to solve the CC-LAP. We prove that the optimal solution of CC-LAP is also an optimal solution to the risk-averse problem with a certain value of the risk-averse parameter. Based on the connection, we prove the existence of an upper bound on the optimal risk-averse parameter and give an algorithm to compute it. We prove that our algorithm will converge to the optimal solution in a finite number of iterations. The simulation results show that the algorithm is scalable to the number of robots.

(2) In CC-STAP, we design algorithms that solve CC-STAP optimally by solving a sequence of deterministic simultaneous task assignment and path planning problems (D-STAP). The travel cost of each edge of D-STAP is a linear combination of mean and variance of the edge cost. We also design an algorithm that solves D-STAP optimally by first computing the cost of shortest paths to the task locations for each robot-task pair and then solving a linear assignment problem with the shortest path costs. We present a distributed algorithm to solve CC-STAP based on the auction algorithm for solving deterministic linear assignment problems [11, 12].

(3) In CC-STAP-B, the key contribution is to prove that the optimal solution of CC-STAP-B 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 solutions of the CC-SP problems. We design a two-step deterministic approach to solve CC-STAP. Leveraging the work on CC-LAP, we also present a novel algorithm for solving the CC-SP problem optimally, which is more efficient than existing algorithms [13, 14]. We present the simulation results demonstrating the scalability of our algorithm to the increasing number of robots and tasks.

(4) In CC-KNAP, we model a multi-robot teaming problem as the chance-constrained knapsack problem. We design an iterative algorithm where we solve the chance-constrained knapsack problem optimally by methodically solving a sequence of risk-averse knapsack problems. We prove that the optimal solution of the chance-constrained knapsack problem is also optimal to the risk-averse knapsack problem with the proper value of the risk-averse parameter and the resource capacity. CC-KNAP boils down to a two-dimensional search on the risk-averse parameter and the resource capacity. The simulation results show that the proposed algorithm is scalable to the number of robots (or packages for the single robot application).

(5) In CC-GAP, we design an $(1+\alpha)$-approximation algorithm for CC-GAP where each robot solves a chance-constrained knapsack problem (CC-KNAP) sequentially. The algorithm is inspired by the idea from the deterministic GAP in [15]. Based on the optimal algorithm for CC-KNAP, we provide an α-approximation algorithm for CC-KNAP using any α-approximation algorithm for deterministic knapsack problem as a subroutine. 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.

1.4 Outline of the Thesis

This thesis is organized as follows: In Chapter 2 we present the related work. From Chapter 3 to Chapter 5, we discuss the multi-robot task allocation with uncertainty in the performance criteria (objective). In Chapter 3 we present our result for multi-robot task allocation with stochastic payoff (CC-LAP). In Chapter 4, we extend to multi-robot simultaneous path planning and task assignment on graphs with stochastic costs, in which the performance criteria is the total travel cost of the robot team (CC-STAP). In Chapter 5, we study another variation of CC-STAP where the performance criterion is the makespan of the travel cost of the robot team, namely CC-STAP-B. From Chapter 6 to Chapter 7, we discuss the multi-robot task allocation with uncertainty in the resource constraints. In Chapter 6 we present results for multi-robot teaming with stochastic robot resource consumption (CC-KNAP). In Chapter 7, we present the result for the generalized multi-robot task allocation with stochastic resource consumption (CC-GAP). In Chapter 8 we present the summary of our current work and future work.

Chapter 2

Related Work

Multi-robot coordination is important in many applications of multi-robot systems, e.g., multi-robot routing [16], multi-robot decision making [17], and other multi-robot coordination problems (see [2, 18]). There are different variations of the multi-robot coordination problem that have been studied in the literature depending on the assumptions about the tasks and the robots (see [1, 2, 19] for surveys), and there also exists multi-robot task allocation systems (e.g., Traderbot [20, 21], Hoplites [22], MURDOCH [23], ALLIANCE [24]) that build on different algorithms. Although the main objective of this thesis is to design algorithms for the multi-robot coordination problem under an uncertain environment (i.e., some variables in the problem are stochastic), the techniques for the deterministic problem are used to solve the stochastic problem. Therefore the related work discussed in this chapter is restricted to the following aspects: (a) deterministic multi-robot coordination; (b) existing techniques for stochastic optimization; (c) chance-constrained optimization.

2.1 Deterministic Multi-robot Coordination

The basic version of multi-robot coordination is the multi-robot task allocation (also known as the linear assignment problem in combinatorial optimization). The problem can be solved optimally in polynomial time by Hungarian algorithm [25, 26] which finds a maximum weighted matching problem for bipartite graphs. Bertsekas [11] proposed a decentralized algorithm with shared memory called auction algorithm which solves the problem almost optimally. A totally distributed algorithm without the shared memory assumption is pre-

sented in [12, 18] which combines the auction algorithm with the consensus algorithm. In the subsequent work from Bertsekas [11, 27], the auction algorithm was extended for solving the generalized assignment problem in which the robots are capable to perform multiple tasks. In other scenarios, it is meaningful to replace the sum objective by a so-called bottleneck objective function, e.g., minimizing the on-time delivery time span in the transportation system. This type of problem is known as the linear bottleneck assignment problem. This problem was introduced by Fulkerson, Glicksberg, and Gross[28]. Garfinkel [29] proposed a threshold algorithm that alternates between choosing threshold value, threshold matrix, and finding maximum matching in the bipartite graph obtained from the threshold matrix. By using [30] to obtain the maximum bipartite matching, the total complexity is $O(n^{2.5}/\sqrt{\log n})$ in dense case (the number of non-zero payoffs is $O(n^2)$). Unlike using the binary search to update the threshold value by the original threshold method, a dual method [31] updates the threshold value from minimum vertex cover. Augmenting path methods [32, 33, 34] mimicked the Hungarian method to solve the linear bottleneck assignment problem. In CC-STAP-B, we use the threshold algorithm to solve the risk-averse problem.

2.2 Stochastic Optimization

Since in many of the realistic applications, the variables are stochastic and we mainly consider the multi-robot coordination problem under the uncertain environment in this thesis, we will discuss stochastic combinatorial optimization techniques relevant to multi-robot coordination. The most widely applied and studied techniques of the stochastic programming model used in the task allocation are two-stage program and multi-stage program [35, 36, 37, 38, 39]. The general idea is to make an initial assignment of robots to tasks in the first stage. After the result of the assignment made in the first stage is known (e.g., the stochastic total payoff is realized), the decision-maker could make a recourse decision to compensate for any bad effects that might have experienced as a result of the first-stage assignment. A solution of the two-stage program includes an initial assignment in the first

stage and a decision rule in the second stage that instructs which action should be taken in response to each realization of the random event. In [40, 41, 42], two-stage problem was applied to stochastic generalized assignment problems. The objective is the expected cost. In [43], the author discussed a stochastic version of the elastic generalized assignment problem. The goal is to minimize the expected cost and expected penalties for violating the resource constraint. The stochastic shortest path problem was studied in [44, 45, 46, 47, 48]. The objective is minimizing the expected path length or cost, which does not apply to the scenarios considered in this thesis. In [49, 50], the objective of the stochastic shortest path is a combination of expected length and cost. The models here typically consider optimizing the expectation which does not incorporate risk.

One of the techniques that provide the risk-averse solution is robust optimization (see a survey [51] for detail). In robust optimization, the uncertain variables are assumed to lie within some range of a minimum and maximum value and the goal is to optimize the worst-case objective function. Although the solution of robust optimization is reliable however it may be overly conservative (bad performance or objective function value) when the realization of the worst case is a very low probability event. Further, the uncertainty in robust optimization has to be bounded which limits its applications.

Generally speaking, our approach lies between stochastic optimization that optimizes the expectation without considering the risk and the robust optimization which provides a reliable solution. Our method allows the flexibility to tradeoff between performance and reliability and allows users to choose the desired level of probability guarantee.

2.3 Chance-constrained Optimization

The stochastic problems considered in this thesis can be classified as stochastic optimization with chance constraint and the deterministic sub-problems are combinatorial optimization (e.g., linear assignment problem, knapsack problem, shortest path problem, bottleneck linear assignment problem). Most of the existing literature in chance-constrained

optimization considered the continuous optimization setting [52, 53, 54]. In general, chance-constrained optimization in discrete settings is difficult to solve (except for some special cases like linear optimization [52], minimum spanning tree [55]). In the chance-constrained knapsack problem (an important sub-problem for generalized multi-robot task allocation), the problem can be transformed into a second order conic program and can be solved in polynomial time by relaxing the binary integer variables. However, as an example presented by Goyal et al. [56] the integrality gap of the conic relaxation is $\Omega(\sqrt{n})$. Kleinberg et al. consider the case where the stochastic variables have Bernoulli-type distribution (two possible outcomes) and provide an $\mathcal{O}(\log \frac{1}{1-p})$ approximation algorithm where p is the probability [57]. Goel et al. in [58] assume stochastic variables have Poisson or exponential distribution. A polynomial time approximation scheme (PTAS) is provided however the chance constraint could be violated by a factor of $(1 + \epsilon)$. In [13] the authors have presented algorithms for chance-constrained shortest path problems. In [14], the method of [13] has been extended to a class of chance-constrained optimization problems, where the objective function is quasi-convex. The key aspect of [14] is to note that the chance-constrained problem (also known as the value-at-risk formulation) is equivalent to the mean-risk formulation (where the objective function is a sum of the mean and a constant times the standard deviation) and thus the value-at-risk model can be solved by solving the value-at-risk formulation. It is known [52] that the mean-risk problem is quasi-concave and the objective function depends on only the means and the variances of the feasible solutions. Thus, finding the optimal solution to the mean-risk problem reduces to finding extreme points over the projection of the feasible points in the mean-variance plane. The authors then go on to provide an exact solution where all extreme points in the mean-variance plane are enumerated. This enumeration is done by solving risk-averse problems with various values of the risk-aversion parameter, although the connection to the risk-averse problem formulation is implicit in the paper. In [59, 60] the authors model the multi-robot routing problem as a chance-constrained optimization problem. The payoffs are also time-sensitive in that the payoff of visiting a target site re-

duces with time. Hence all tasks need not be done by robots. The relationship between the chance-constrained and the risk-averse problem has not been identified or explored in [59, 60]. In [61], the authors have extended the work in [13] to develop faster algorithms for stochastic shortest path problems.

Our geometric view of the solutions and objectives in this thesis was inspired by work in [14]. We make the relationship between the risk-averse problem and the value-at-risk problem explicit. In multi-robot task allocation in Chapter 3, we obtain an upper bound on the risk-averse parameter λ such that solving a finite number of deterministic risk-averse problems with values of risk-aversion parameter below the bound is guaranteed to provide us the optimal solution for the value-at-risk problem. We present a methodical one-dimensional search method on the risk-aversion parameter and show through simulations on random data sets that our method is more efficient than the exact method proposed in [14]. Further in multi-robot teaming in Chapter 6, the problem cannot be modeled as value-at-risk formulation. We proved that the optimal solution can be obtained by solving deterministic knapsack problems with proper value of risk-averse parameter λ and distance to be traversed denoted by L. The problem becomes a two-dimensional search on λ and L. To the best of our knowledge, there are no available algorithms with theoretical guarantees on solution quality and better efficiency that can solve the chance-constrained multi-robot coordination problems studied in this thesis.

Chapter 3

Chance-constrained Linear Assignment Problem

3.1 Introduction

Multi-robot task allocation (MRTA) is a fundamental problem that arises in a wide variety of application scenarios like manufacturing, automated transport of goods, environmental monitoring, and surveillance [1, 2]. We will start this thesis with the basic version of multi-robot task allocation, which is the single-task robot, single-robot task, and instantaneous assignment (ST-SR-IA) MRTA problem [1]: *Given a set of robots and a set of tasks, with each robot obtaining some payoff (or incurring some cost) for each task, find a one-to-one assignment of robots to tasks so that the overall payoff (or team payoff) of all the agents is maximized (or cost incurred is minimized).* The ST-SR-IA task allocation is also known as the linear assignment problem (LAP). It can be solved optimally in polynomial time by centralized algorithms [25, 26] and decentralized algorithms [11, 12, 62]. Much of the research in linear assignment problems have assumed known payoffs (exceptions include [63, 64, 65]). As shown in Fig. 1.1, the ST-SR-IA multi-robot task allocation arises commonly in pick and place type of problems of transporting items from a start location to a target location where each item should be carried by only one robot. Such problems are widely considered in many application scenarios like automated package delivery (e.g., Amazon's drones delivery), automated warehouse (e.g., Kiva systems for warehouse automation), and automated ports. In applications, the payoffs of the robots for tasks may not be known exactly, and consequently, the overall performance of the robot system becomes uncertain. Thus, either for autonomous deployment or for human-supervised deployment, one may want to have

algorithms for linear assignment with performance certificates despite the uncertain payoffs. The goal of this chapter is to develop algorithms with probabilistic performance certificates for linear assignment problems with uncertain payoffs.

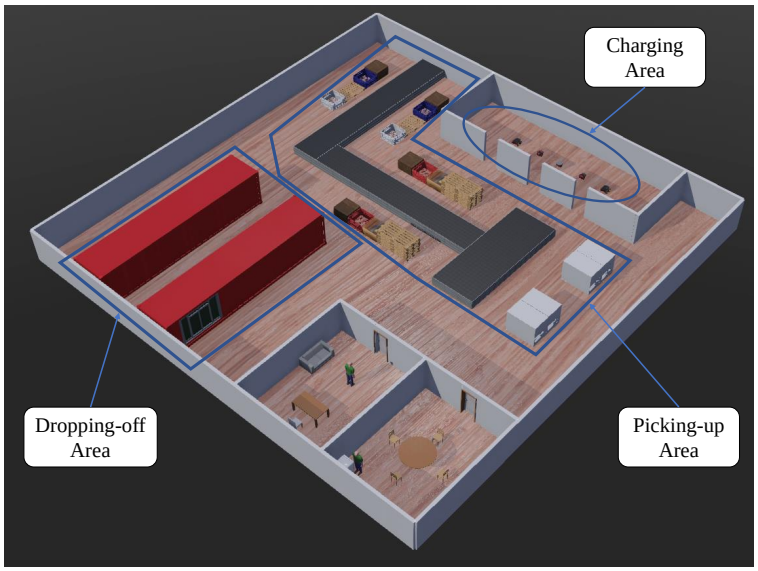


Figure 3.1: In this automated warehouse, the mobile robots are initially located at the charging area. The robots should pick up packages at loading stations and deliver them to the unloading stations in the dropping-off area.

We consider a set of n_r robots, $R = \{r_i\}$, $i = 1, \ldots, n_r$, and a set of n_t tasks, $T = \{t_j\}$, $j = 1, \ldots, n_t$. The payoff for a robot r_i for task t_j is a random variable a_{ij}. Each robot can do at most one task[1] . In task allocation problems with uncertain payoffs, one would like to have an assignment with some guarantees on the quality of assignment (i.e., total team payoff achievable in the assignment) irrespective of the realization of the random payoffs. Technically, different objectives can be used for the assignment problem depending on the model of uncertainty used for the payoffs. Two typical models of uncertainty are set-theoretic models and probabilistic models. Set-theoretic uncertainty model for the linear assignment has been considered in [64]. In the set-theoretic model of uncertainty, the uncertain payoffs are assumed to lie within some range of a minimum and maximum value and the objective is to maximize the worst-case payoff. In other words, the solution obtained maximizes the

[1] Note that the problem where we have to minimize the cost of an assignment is identical and therefore we will talk about the payoff maximization problem here.

total payoff for the worst possible realization of the individual payoffs. Thus, this solution may be overly conservative when the realization of the worst case is a very low probability event.

For probabilistic models of uncertainty, two popular objectives are the expected payoff and the chance-constrained payoff [59]. Although the expected payoff maximization is a quite popular objective for decision making problems under uncertainty, the solution obtained is meaningful only in situations where the same problem has to be executed multiple times by the robot team, and one is interested in the average performance of the team over the multiple scenarios. For any particular realization of individual robot-task payoffs, the actual team performance (or payoff) may be far off from the average team performance. Consequently, there is no guarantee on the performance of the robot team for any individual scenario. Another objective is to have a probabilistic guarantee on the quality of assignment for any given scenario by using a chance constraint formulation, which is what we pursue in this chapter. Our goal is to solve the following problem: *Find an assignment and the maximum possible team payoff value, y^*, such that the realized team payoff is always greater than y^* with (an a priori specified) probability p, under any realization of the random individual robot-task payoffs. The assignment has to also satisfy the integer constraints arising from LAP, i.e., one-to-one assignment of tasks to robots.* The assignment along with y^* is a probabilistic performance certificate for the robot team operating under uncertainty. When $p = 1$, we have a performance certificate with worst-case guarantees.

We formulate the above task allocation problem as a chance-constrained combinatorial optimization problem. In general, the stochastic program with chance constraint is a non-convex optimization problem, and solving the problem directly is challenging. However, the chance-constrained linear assignment problem (CC-LAP) is a special case of the class of combinatorial problems considered in [14]. The CC-LAP, which is also known as a value-at-risk formulation, can be reformulated as an equivalent deterministic non-linear integer program, called the mean-risk formulation. In [14], the author provides an exact solution for

the nonlinear integer program or mean-risk formulation, by solving a sequence of parametric deterministic LAPs, called the risk-averse problem, for various values of the risk-aversion parameter (say λ), although the connection to the risk-averse problem formulation is implicit in the paper.

Contributions: Our overall approach to solving the value-at-risk formulation of linear assignment or chance-constrained linear assignment problem builds on the work by [14]. We make the relationship between the risk-averse problem and the CC-LAP explicit. In particular, we prove that for the optimal solution of the chance-constrained problem, there is a value of the risk-averse parameter λ, say λ^*, for which the optimal solution to the chance-constrained problem is also a solution to the risk-averse problem. We call λ^*, the optimal risk-aversion parameter. Based on this connection between CC-LAP and risk-averse linear assignment, we prove the existence of an upper bound ($\bar{\lambda}$) on the optimal risk-aversion parameter and give an algorithm to compute $\bar{\lambda}$. We present a novel methodical one-dimensional search method on the risk-aversion parameter, λ, to solve the CC-LAP. We prove that our algorithm will converge to the optimal solution in a finite number of iterations. *The development of the connection between the value-at-risk problem and the risk-averse problem for linear assignment as well as the presented deterministic method to solve the CC-LAP are the primary contributions of this paper.* The connection between chance-constrained integer programs and risk-averse integer programs is of broader interest to other problems. This insight has been exploited by us in developing algorithms for CC-STAP and CC-GAP in this thesis.

Note that each risk-averse problem is essentially a deterministic linear assignment problem, with the values of the robot-task payoff dependent on the parameter λ. Thus, the CC-LAP can be solved by solving a sequence of methodically generated deterministic linear assignment problems, which can be solved by using centralized [25] or distributed algorithms [11, 12]. We show through simulations on randomly generated scenarios that our method is much more efficient than the exact method proposed in [14]. Simulation

results also show that during our first step of constructing the upper bound, we obtain a solution that is quite close to the optimal solution. This provides empirical evidence that in practice, we may use just the first step. Therefore, we also provide a distributed algorithm implementing the first step of solving the CC-LAP, by modifying the distributed auction method proposed in [11, 12]. We present simulation results on randomly generated scenarios that demonstrate empirically that we obtain nearly optimal solutions.

This chapter is organized as follows: In Section 3.2 provide preliminaries and a motivating example. In Section 3.3, we discuss the formulations for chance constrained linear assignment problem and the risk-averse linear assignment problem. In Section 3.4, we study the geometric relationship between those two models. In Section 3.5 and Section 3.6, we present our algorithm in centralized and distributed setting respectively. In Section 3.7 we present our simulation results. In Section 3.9, we summarize our contributions.

3.2 Preliminaries and Motivating Example

In this section, we will present the linear assignment problem (LAP) as a integer program (IP) and connect the abstract problem formulation to a robotics application scenario. Suppose there are n_r heterogeneous robots, $R = \{r_1, r_2, \ldots, r_{n_r}\}$ and n_t tasks, $T = \{t_1, t_2, \ldots, t_{n_t}\}$. Let $a_{ij} \in \mathbb{R}$ be the payoff for the assignment pair (r_i, t_j), i.e., the payoff for assigning robot r_i to task t_j. Without loss of generality, we assume that any robot can be assigned to any task. If a robot cannot be assigned to any task, it can be encoded by making the corresponding robot-task payoff $-\infty$ (or a very small number in practice). Each robot can perform exactly one task (second constraint in (3.1)) while each task can be assigned to exactly one robot (first constraint in (3.1)). Let f_{ij} be the binary decision variable that takes a value 1 if robot r_i is assigned to task t_j and 0 otherwise (third constraint in (3.1)). The goal is to assign all tasks to robots so that the total team payoff $(\sum_{i=1}^{n_r} \sum_{j=1}^{n_t} a_{ij} f_{ij})$ is maximized. For simplicity of exposition, we assume that the number

of robots is same with the number of tasks, i.e., $n_r = n_t = n$. The IP formulation of LAP is:

$$\begin{aligned} \max \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} f_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{n} f_{ij} = 1, \quad \forall j = 1, \ldots, n \\ & \sum_{j=1}^{n} f_{ij} = 1, \quad \forall i = 1, \ldots, n \\ & f_{ij} \in \{0, 1\}, \quad \forall i, j \end{aligned} \tag{3.1}$$

The linear assignment problem (LAP) is a classical problem in operations research and combinatorial optimization. Both centralized [25] and distributed [27, 12] polynomial time algorithms have been devised to solve LAP optimally.

Motivating Application: The linear assignment problem arises in many scenarios including parts transfer in automated factories, last-mile transfer in automated delivery, collection, and transfer of medical samples in remote areas using drones. The common structure in these scenarios is that the robots have to transfer objects from a central location to spatially distributed locations. Thus, robots have to move back and forth between a central location and spatially distributed points.

We elaborate on the scenario of multi-robot package delivery in an automated factory shown in Fig. 3.2. The robots initially located at the charging stations (labeled as Ci) drive around the facility picking up packages and delivering them to unloading stations for storing or processing. The packages are different in properties size and weight at different loading stations (labeled as Li) and have to be delivered to specific unloading stations (labeled as Ui).

The task for each robot is traveling from its initial charging station to the loading station of the assigned packages and then delivering the packages to the specific unloading station (in other words, the loading station and the charging station should have the same index). The payoffs of the assignment are the combinations of the benefits and fuel costs of delivering the package. The robots have heterogeneous capability. Therefore the payoff of

delivering a package is different by different robots.

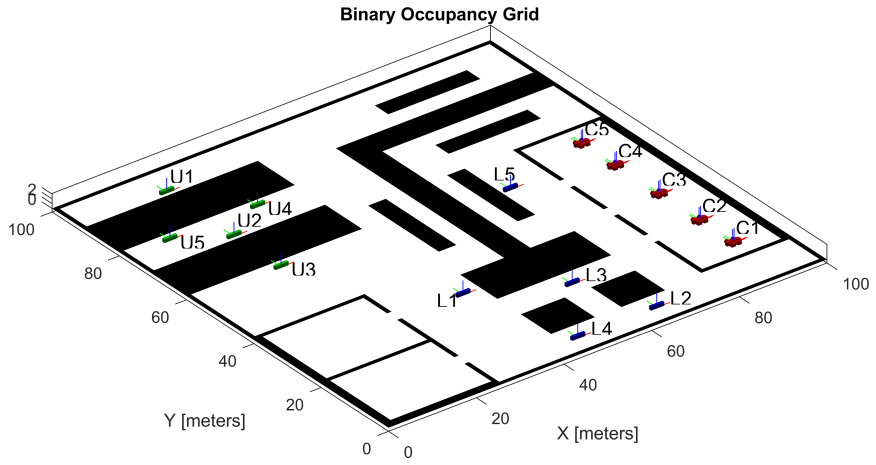


Figure 3.2: The multi-robot packing system in the warehouse. Robots are initially located at charging stations labeled as Ci and should pick up packages from the loading stations and deliver them to the unloading stations with the same index say Ui. All robots should avoid static and moving obstacles during the delivery.

This problem can be modeled as a linear assignment problem (LAP). In LAP it is usually assumed that the payoffs are known constants. However, in many application scenarios, the payoffs are uncertain because the costs of robots delivering packages are stochastic variables. The robot has to avoid moving obstacles (e.g., workers or robots of the external system) which may take place unexpectedly at different locations and times. Therefore the fuel consumption of a robot delivering the same package could be random. In this paper, we will present an algorithm to solve a stochastic variation of the linear assignment problem where the payoffs are random variables. We assume that payoffs are independent Gaussian random variables with known means and variances, i.e., $a_{ij} \sim \mathcal{N}(\mu_{ij}, \sigma_{ij}^2), \forall i, j$, where μ_{ij} is the mean and σ_{ij}^2 is the variance of a_{ij}. In contrast to the objective of LAP which is deterministic, the total payoff of a solution (assignment) in the stochastic linear assignment problem is random. Therefore, the objective in (3.1) is not meaningful and we formulate the problem as an integer program with chance constraint, called chance-constrained linear assignment problem (CC-LAP).

3.3 Problem Formulation

3.3.1 Chance-constrained Linear Assignment Problem

We now present the formal definition of the chance-constrained linear assignment problem with stochastic payoffs. The CC-LAP can be formulated as an integer program:

$$\begin{aligned} \max \quad & y \\ \text{s.t.} \quad & \mathbb{P}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} f_{ij} \geq y\right) \geq p \\ & \sum_{i=1}^{n} f_{ij} = 1, \quad \forall j = 1, \ldots, n \\ & \sum_{j=1}^{n} f_{ij} = 1, \quad \forall i = 1, \ldots, n \\ & f_{ij} \in \{0, 1\}, \quad \forall i, j \end{aligned} \tag{3.2}$$

where the first constraint is a probabilistic constraint on the total team payoff. The constraints in the second and third rows are the assignment constraints which ensure that each task is assigned to exactly one robot. In words, Equation (3.2) solves the following: *Given a probability, p (say 0.99), compute an assignment of robots to tasks with the maximum team payoff value (say y^*), such that with probability p it is guaranteed that the actual team payoff under* **any realization** *of the random payoff a_{ij} is greater than or equal to y^* with probability p.* Thus, the chance-constrained problems can provide strong probabilistic guarantees irrespective of the payoff realizations. The chance-constrained formulation is also known as the *value-at-risk (VaR) formulation.*

Since the chance constraints essentially involve integrating the probability density function of the total payoff, it is very hard to solve (3.2) directly. We will instead reformulate (3.2)

by restating the chance-constraint as an equivalent deterministic constraint [66]:

$$\begin{aligned} & \mathbb{P}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}f_{ij} \geq y\right) \\ = \; & \mathbb{P}\left(Z \geq \frac{y - \sum_{i=1}^{n}\sum_{j=1}^{n}\mu_{ij}f_{ij}}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_{ij}^2 f_{ij}}}\right) \end{aligned} \tag{3.3}$$

where

$$Z = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}f_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n}\mu_{ij}f_{ij}}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_{ij}^2 f_{ij}}}$$

Since all payoffs are normally distributed and independent. Therefore Z is a standard normal random variable with 0 mean and variance 1. Let $\Phi(\cdot)$ be the cumulative distribution function of the standard normal random variable. The chance constraint in (3.2) can be written as

$$\begin{aligned} & 1 - \Phi\left(\frac{y - \sum_{i=1}^{n}\sum_{j=1}^{n}\mu_{ij}f_{ij}}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_{ij}^2 f_{ij}}}\right) \geq p \\ \Leftrightarrow \; & y \leq \sum_{i=1}^{n}\sum_{j=1}^{n}\mu_{ij}f_{ij} - \Phi^{-1}(p)\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_{ij}^2 f_{ij}} \end{aligned} \tag{3.4}$$

Let $C = \Phi^{-1}(p)$. For $p > 0.5$, $C > 0$. Since the objective of CC-LAP in (3.2) is maximizing team payoff value y and the y is upper-bounded by the right-hand side of the inequality (3.4), CC-LAP can be *equivalently* formulated as

$$\begin{aligned} \max \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}\mu_{ij}f_{ij} - C\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}\sigma_{ij}^2 f_{ij}} \\ \text{s.t.} \quad & \sum_{i=1}^{n} f_{ij} = 1, \quad \forall j = 1, \ldots, n \\ & \sum_{j=1}^{n} f_{ij} = 1, \quad \forall i = 1, \ldots, n \\ & f_{ij} \in \{0, 1\}, \quad \forall i, j \end{aligned} \tag{3.5}$$

The formulation in (3.5) is also known as the *mean-risk formulation*. Note that (3.5) is a deterministic problem. However, the objective function is non-linear (in fact quadratic) and thus (3.5) is a non-linear integer program, which are, in general, difficult to solve. We will

now introduce another formulation called risk-averse linear assignment problem (RA-LAP), and subsequently show that we can solve CC-LAP by solving a sequence of RA-LAPs.

3.3.2 Risk-averse Linear Assignment Problem

Another way of formulating decision making problems under uncertainty is by using the notion of risk-aversion from the decision theory literature. Let $P = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} f_{ij}$ be the team payoff for a feasible assignment (i.e., assignment that satisfies the integer constraints in (3.1)). We define a utility function, U, for the robot team as

$$U = 1 - e^{-\lambda P} \tag{3.6}$$

where $\lambda \geq 0$ is the risk-aversion parameter for the robot team (in economics, this is known as the Arrow-Pratt index of absolute risk aversion). The higher the value of λ the more risk-averse the robot. The utility function U has the following properties. (a) It is non-negative with its value equal to 0 for any feasible assignment if $\lambda = 0$. (b) For any given λ, the utility U is a monotonically increasing function of the total payoff P.

Since a_{ij} is a random variable, we deal with uncertainty by maximizing the expected total utility of the assignment. In contrast to maximizing the expected total payoff, maximizing the expected total utility of the assignment also concern about the risk (high variance). In more detail, a high expectation of the total payoff cannot guarantee that a single realization of the corresponding assignment will achieve the same or higher total payoff. The utility function considers not only the expectation but also the variance of the total payoff of an assignment. Note that for any feasible assignment two robots will not be assigned to the same task. Therefore, for any assignment, $P = \sum_{j=1}^{n} a_{ij} f_{ij}$ is a sum of independent Gaussian random variables. Thus, P is a Gaussian random variable with mean $\mu = \sum_{i=1}^{n} \sum_{j=1}^{n} \mu_{ij} f_{ij}$ and variance $\sigma^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \sigma_{ij}^2 f_{ij}$. Therefore, from [67] the expected total utility of an assignment is

$$E[U] = 1 - e^{-2\lambda(\mu - \lambda\sigma^2)} \tag{3.7}$$

For any given λ maximizing $E[U]$ is equivalent to maximizing $\mu - \lambda\sigma^2$. Thus we can formulate

the risk-averse optimization formulation for the linear assignment problem (RA-LAP) as follows:

$$\begin{aligned} \max \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}(\mu_{ij} - \lambda\sigma_{ij}^2)f_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{n} f_{ij} = 1, \quad \forall j = 1, \ldots, n \\ & \sum_{j=1}^{n} f_{ij} = 1, \quad \forall i = 1, \ldots, n \\ & f_{ij} \in \{0, 1\}, \quad \forall i, j \end{aligned} \tag{3.8}$$

For a given value of λ, the RA-LAP is a deterministic linear assignment problem (3.1) with modified payoffs that are a linear combination of means and variances, i.e., $a_{ij} = \mu_{ij} - \lambda\sigma_{ij}^2$. Therefore RA-LAP can be solved optimally by methods like the Hungarian Algorithm in polynomial time. However, the challenge is that we do not know the risk-averse parameter λ which determines the quality of the solution and the problem formulation does not directly provide a probabilistic guarantee of a solution.

To take the advantages of CC-LAP and RA-LAP, the idea of our method is to solve CC-LAP which provides a probabilistic guarantee by solving a number of RA-LAPs and consequently obtaining a proper value for risk-averse parameter λ with which the optimal solution of RA-LAP is the same with the optimal solution of CC-LAP. In the next section, we will discuss the relationship between CC-LAP and RA-LAP and prove that the optimal solution of CC-LAP is the optimal solution of RA-LAP with a proper choice of risk-averse parameter λ.

3.4 Connection Between Chance-constrained and Risk-averse Formulations

The connection between CC-LAP and RA-LAP can be understood by a geometric analysis of the two problems on a two-dimensional space, called variance-mean plane presented in Fig. 3.3. Every assignment (or solution), for which the integer constraints are feasible, can be represented by a point in this plane. The horizontal coordinate σ^2 represents the

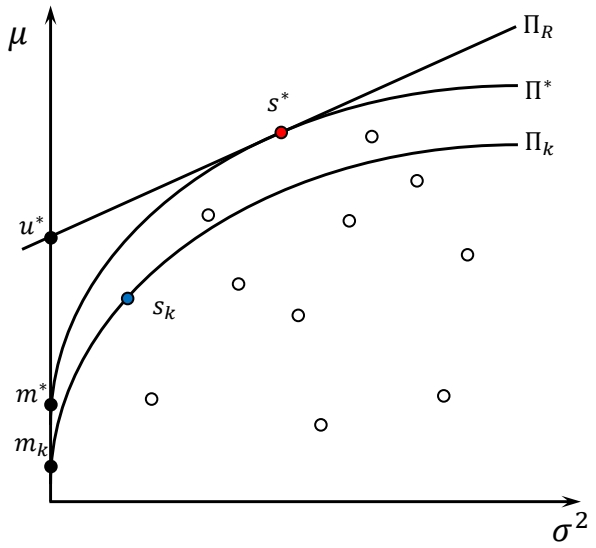


Figure 3.3: Illustration of the variance-mean plane. Any feasible solution of CC-LAP or RA-LAP can be mapped to a point on the variance-mean plane where the vertical and horizontal coordinates represent the variance and mean of the team payoff respectively. The level curve of the objective function of CC-LAP is the parabola with a vertical intercept equal to the objective function value. The level curve of RA-LAP is a straight line. Therefore the problem is to find the point s^* on the variance-mean plane so that the parabola going through this point has the highest vertical intercept.

variance of total team payoff, i.e., $\sigma^2 = \sum_{i,j=1}^{n} \sigma_{ij}^2 f_{ij}$, and vertical coordinate μ represents the mean of total team payoff, i.e., $\mu = \sum_{i,j=1}^{n} \mu_{ij} f_{ij}$.

We also call the point $s = (\sigma^2, \mu)$ on variance-mean plane as a feasible solution. The feasible set in the mean-variance plane will be denoted by S. For any $s \in S$, let $m = \mu - C\sigma$ be the value of the objective of (3.5) evaluated at s. Then the set of all points $\Pi \triangleq \{(\sigma^2, \mu) | \mu - C\sigma = m\}$ is a parabola passing through s with intercept m on the y-axis of the mean-variance plane (with abuse of notation m also denotes the vertical intercept). These curves passing through the points of S are level sets with different values of m (i.e., they will not intersect). For a fixed value of m, all points in S, either lie above, below or on a given parabola. Let $m_k = \mu_k - C\sigma_k$ for any point $s_k \in S$. If $m_k < m$, then s_k lies below the parabola, Π, and if $m_k > m$, s_k lies above the parabola, Π (e.g., s_k is below the level curve Π^* and $m_k < m^*$ in Fig. 3.3).

The value of C, which is a constant dependent on the pre-specified probability p, determines the shape of the parabola. Furthermore, the optimal solution to CC-LAP is the point s^*, such that the parabola passing through it has the highest vertical intercept, say m^*. The objective of (3.8) is a straight line through the point s with slope equal to λ. The above geometric interpretation provides us some useful facts to solve the CC-LAP.

Definition 1. *Let $\mathcal{S}$ be a point set. A point $s \in \mathcal{S}$ is an extreme point if there exists a straight line (hyperplane) through s such that all other points of $\mathcal{S}$ lie strictly on one side.*

Lemma 1. *Let s^* be an optimal solution for CC-LAP. Then, there is a value of λ, say λ^* such that s^* is the optimal solution of RA-LAP for $\lambda = \lambda^*$. Also, s^* is an extreme point of the set S.*

Proof. Let $s^* = (\sigma_{opt}^2, \mu_{opt})$ and the value of the objective of CC-LAP in (3.5) be $m^* = \mu_{opt} - C\sigma_{opt}$. Then, the level curve of the objective of CC-LAP, $\Pi^* \triangleq \{(\sigma^2, \mu) | \mu - C\sigma = m^*\}$ passing through s^* has vertical intercept at m^* (see Figure 3.3). Since s^* is the optimal solution, then for any point $s_k = (\sigma_k^2, \mu_k)$ in feasible solution set S,

$$\mu_{opt} - C\sigma_{opt} \geq \mu_k - C\sigma_k$$

Therefore

$$\mu_k \leq C\sigma_k + \mu_{opt} - C\sigma_{opt} = C\sigma_k + m^* \tag{3.9}$$

Hence, any feasible solution (σ_k^2, μ_k) is below the level curve Π^*.

Let the tangent of level curve Π^* at s^* denoted by $\Pi_R \triangleq \{(\sigma^2, \mu) | \mu - \lambda^* \sigma^2 = u^*\}$ where $u^* = \mu_{opt} - \lambda^* \sigma_{opt}^2$ and $\lambda^* = \frac{C}{2\sigma_{opt}}$ (the slope of curve Π^* at s^*). It is true that

$$(\sigma_{opt} - \sigma_k)^2 \geq 0$$

$$\iff \quad \lambda^*(\sigma_k^2 - \sigma_{opt}^2) \geq C(\sigma_k - \sigma_{opt})$$

$$\iff \quad (\mu_{opt} - \lambda^* \sigma_{opt}^2) + \lambda^* \sigma_k^2 \geq (\mu_{opt} - C\sigma_{opt}) + C\sigma_k$$

$$\iff \quad u^* + \lambda^* \sigma_k^2 \geq m^* + C\sigma_k \geq \mu_k \tag{3.10}$$

Therefore any feasible solution (σ_k^2, μ_k) is below the straight line Π_R. The conclusions from Equation (3.9) and (3.10) can also be easily observed from Fig. 3.3. The straight line Π_R is the level curve of the objective function of RA-LAP with $\lambda = \lambda^*$ in (3.8). The optimal solution of CC-LAP, s^* is also optimal to RA-LAP with λ^*. The optimal objective function value is equal to the vertical intercept u^*. By Definition 1, the optimal solution s^* is an extreme point of the feasible solution set S.

□

Remark 1. *Note that Lemma 1 gives the existence of an optimal risk-averse parameter λ^*. In fact, there is a range of values of λ^*, for which the solution of RA-LAP will give the optimal solution for CC-LAP, i.e., s^*. In the subsequent discussion, when we use λ^*, we will imply the minimum value of risk-averse parameter λ, such that we obtain the solution s^* by solving the RA-LAP.*

Corollary 1. *The optimal solution of RA-LAP in* (3.8) *with any non-negative risk-averse parameter, say λ_k is an extreme point on the variance-mean plane.*

Proof. Let $s_k = (\sigma_k^2, \mu_k)$ be the optimal solution of RA-LAP with λ_k and the level curve of its objective function at s_k is $\Pi_R = \{(\sigma^2, \mu) | \mu - \lambda_k \sigma^2 = u_k\}$ where $u_k = \mu_k - \lambda_k \sigma_k^2$. Let $s_a = (\sigma_a^2, \mu_a)$ any feasible solution. It is true that

$$\mu_k - \lambda_k \sigma_k^2 \geq \mu_a - \lambda_k \sigma_a^2$$

Therefore

$$u_k + \lambda_k \sigma_a \geq \mu_a \tag{3.11}$$

It implies that all feasible solutions are either on or below the straight line Π_k. By Definition 1, the optimal solution (σ_k^2, μ_k) of the RA-LAP with λ_k is an extreme point on the variance-mean plane. □

Lemma 1 motivates the method we use for solving CC-LAP. If we can obtain the λ^*, then solving RA-LAP with $\lambda = \lambda^*$ gives us s^*, the solution for CC-LAP. Finding λ^* is a one-dimensional parameter search problem on λ. By Corollary 1, as we change λ and solve RA-LAP, the solutions generate extreme points of the set S. Thus, one way to compute s^* is to enumerate all extreme points by searching for the risk-averse parameter λ over the range $[0, \infty)$ (which has been presented in [14] for solving chance-constrained shortest path problem). However, in the linear assignment problem, the number of extreme points could be large. In our method, we first find a range for λ, i.e., $\lambda^* \in [0, \bar{\lambda}]$ where $\bar{\lambda}$ denotes an upper bound for λ^*.

We will first discuss some properties of the objective value of RA-LAP as a function of λ. Let μ_i, σ_i^2 denote the mean and variance of the total payoffs of the optimal solution of RA-LAP with λ_i respectively. The corresponding objective function value of RA-LAP in (3.8) and CC-LAP in (3.5) is $\mu_i - \lambda_i \sigma_i^2$ and $\mu_i - C\sigma_i$.

Lemma 2. *The optimal objective function value $\mu - \lambda\sigma^2$ for RA-LAP in* (3.8) *is a strictly monotonically decreasing function of the risk-averse parameter λ, i.e., $\mu_i - \lambda_i \sigma_i^2 > \mu_j - \lambda_j \sigma_j^2$, $\forall\ \lambda_i < \lambda_j$.*

Proof. Let $\lambda_i < \lambda_j$ be two risk-averse parameters and $({\sigma_i}^2, \mu_i)$ and $({\sigma_j}^2, \mu_j)$ be solutions obtained from the optimally solving the corresponding RA-LAPs. Since $({\sigma_i}^2, \mu_i)$ is the optimal solution to RA-LAP for λ_i, we have

$$\mu_i - \lambda_i {\sigma_i}^2 \geq \mu_j - \lambda_i {\sigma_j}^2 \tag{3.12}$$

Since $\lambda_i < \lambda_j$, we have

$$\mu_j - \lambda_i {\sigma_j}^2 > \mu_j - \lambda_j {\sigma_j}^2 \tag{3.13}$$

Combining (3.12) and (3.13) we obtain $\mu_i - \lambda_i {\sigma_i}^2 > \mu_j - \lambda_j {\sigma_j}^2$. Thus, the optimal objective function of RA-LAP is a strictly monotonically decreasing function of λ. □

Lemma 3. *The variance of total payoff with the assignment obtained from RA-LAP is a non-increasing function of risk-averse parameter, i.e.,* $\sigma_i^2 \geq \sigma_j^2$, $\forall \lambda_i < \lambda_j$.

Proof. Let $\lambda_i < \lambda_j$ be two risk-averse parameters and $({\sigma_i}^2, \mu_i)$ and $({\sigma_j}^2, \mu_j)$ be solutions obtained from the optimally solving the corresponding RA-LAPs. Since $({\sigma_j}^2, \mu_j)$ is the optimal solution to RA-LAP for λ_j, we have

$$\mu_i - \lambda_j {\sigma_i}^2 \leq \mu_j - \lambda_j {\sigma_j}^2 \tag{3.14}$$

By negating (3.14) and adding it to (3.12), we obtain $(\lambda_j - \lambda_i){\sigma_i}^2 \geq (\lambda_j - \lambda_i){\sigma_j}^2$. Therefore ${\sigma_i}^2 \geq {\sigma_j}^2$. □

Lemma 4. *Let* $g(\lambda)$ *be the function that is equal to the optimal value of the objective of the RA-LAP in* (3.8) *for a given* λ. *Then,* $g(\lambda)$ *is a continuous function of* λ.

Proof. Let S be the set of all feasible solutions to RA-LAP, i.e., the set of all solutions that satisfies the integer constraints in (3.8). Let $s_k \in S$ be a feasible solution with mean μ_k and variance σ_k^2. Any feasible solution, (σ_k^2, μ_k), is optimal for RA-LAP, for a given value of λ, if $(\mu_k - \lambda\sigma_k^2) \geq (\mu_j - \lambda\sigma_j^2)$, $\forall s_j \in S \setminus \{s_k\}$. Thus, the objective $g(\lambda)$ can also be written as:

$$g(\lambda) = \max_{s_k \in S}\{\mu_k - \lambda\sigma_k^2\} \tag{3.15}$$

Since $g(\lambda)$ is the maximum of a finite number of linear functions in λ, the function $g(\lambda)$ is piecewise-linear [68] and hence continuous in λ. □

Let $f(\lambda)$ equal to the objective function value of CC-LAP with the solution obtained from RA-LAP, i.e., $f(\lambda) = \mu - C\sigma$ where $\mu - \lambda\sigma^2 = \max_{s_k \in S}\{\mu_k - \lambda\sigma_k^2\}$. Note that $f(\lambda)$ is a discontinuous function. The following lemma illustrates the relationship between $g(\lambda)$ and $f(\lambda)$.

Lemma 5. *As risk-averse parameter λ increases, the optimal objective function value of RA-LAP, $g(\lambda)$, is always greater than the objective function value of CC-LAP at the solution obtained from RA-LAP, $f(\lambda)$, until they are equal, i.e., $g(\lambda) = f(\lambda)$.*

Proof. Let λ_k be any risk-averse parameter, (σ_k^2, μ_k) be the optimal solution of RA-LAP with λ_k and $\bar{\lambda}$ be the risk-averse parameter such that $g(\bar{\lambda}) = f(\bar{\lambda})$. For $\lambda_0 = 0$, we have $\lambda_0\sigma_0 = 0 < C$ and thus $g(\lambda_0) > f(\lambda_0)$. For any risk-averse parameter λ_k such that $\lambda_0 \leq \lambda_k < \bar{\lambda}$ and $\lambda_k\sigma_k < C$. Define $\lambda_{k+1} \triangleq \frac{C}{\sigma_k}$ and let λ be any risk-averse parameter such that $\lambda_k < \lambda < \lambda_{k+1}$. From Lemma 3, we know $\sigma_{k+1} \leq \sigma \leq \sigma_k$. Therefore $\lambda\sigma < \lambda_{k+1}\sigma_k = C$ and $\lambda_{k+1}\sigma_{k+1} \leq \lambda_{k+1}\sigma_k = C$. Hence $g(\lambda) > f(\lambda)$ and $\lambda_{k+1} = \bar{\lambda}$ when the optimal solutions of RA-LAP with λ_k and λ_{k+1} are same, i.e., $\sigma_{k+1} = \sigma_k$. By induction, for any risk-averse parameter $0 < \lambda < \bar{\lambda}$ the optimal objective function value of RA-LAP with λ is always greater than the objective function value of CC-LAP at the same solution. □

Lemma 6. *There exists a risk-averse parameter, denoted by $\bar{\lambda}$, such that the optimal objective function value of RA-LAP with $\bar{\lambda}$ is equal to the objective value of CC-LAP, i.e., $\bar{\mu} - C\bar{\sigma} = \bar{\mu} - \bar{\lambda}\bar{\sigma}^2$. This objective value gives a lower bound for the optimal objective value of CC-LAP. Furthermore, the value of the risk-averse parameter, λ^*, that gives the optimal assignment for CC-LAP must lie in the interval $[0, \bar{\lambda}]$.*

Proof. First we need to prove that $\bar{\lambda}$ exists. Let $g(\lambda)$ be the optimal objective value for RA-LAP for a given λ and $f(\lambda)$ be the objective function value of CC-LAP at the optimal solution obtained from RA-LAP with λ. For $\lambda = \lambda_0 = 0$, let (σ_0^2, μ_0) be the optimal solution for RA-LAP. Then $g(\lambda_0) = \mu_0$ and $f(\lambda_0) = \mu_0 - C\sigma_0$. Thus $g(\lambda_0) > f(\lambda_0)$. Let λ_m be a value of λ such that, the optimal solution of RA-LAP, i.e., (σ_m^2, μ_m) satisfies $\lambda_m\sigma_m > C$

(image λ_m is a very large number). Then $g(\lambda_m) = \mu_m - \lambda\sigma_m^2 < \mu_m - C\sigma_m = f(\lambda_m)$. Thus $g(\lambda_m) < f(\lambda_m)$.

Thus we have the following three facts: (a) $g(\lambda)$ is a continuous (by Lemma 4) and monotonically decreasing function of λ (by Lemma 2); (b) $g(\lambda) > f(\lambda)$, for $\lambda = 0$ and $g(\lambda) < f(\lambda)$, for $\lambda = \lambda_m$; (c) as λ increases from 0, $g(\lambda)$ is always greater than $f(\lambda)$ until $g(\lambda) = f(\lambda)$ (by Lemma 5). Thus, from the intermediate value theorem, we can conclude that there will be a value of $\lambda \in [0, \lambda_m]$, at which $g(\lambda)$ intersects the curve $f(\lambda)$. Let $\bar{\lambda}$ denote the value of λ at the intersection points, where $g(\lambda) = f(\lambda)$, i.e., $\bar{\mu} - C\bar{\sigma} = \bar{\mu} - \bar{\lambda}\bar{\sigma}^2$ or more simply $\bar{\lambda}\bar{\sigma} = C$.

Now we need to prove that the objective values of CC-LAP obtained outside the range $[0, \bar{\lambda}]$ are smaller than the objective value obtained at $\bar{\lambda}$, i.e., $\bar{\mu} - C\bar{\sigma} > \mu - C\sigma$ for any $\lambda > \bar{\lambda}$. Suppose μ and σ^2 are obtained by solving RA-LAP with $\lambda > \bar{\lambda}$, there are two different cases:

(1) $\lambda\sigma \leq C$. Obviously $y = \mu - C\sigma \leq \mu - \lambda\sigma^2$. And because $\lambda > \bar{\lambda}$, from Lemma 2 we know that $\mu - \lambda\sigma^2 < \bar{\mu} - \bar{\lambda}\bar{\sigma}^2 = \bar{\mu} - C\bar{\sigma}$. Therefore $\bar{\mu} - C\bar{\sigma} > \mu - C\sigma$.

(2) $\lambda\sigma > C$. Let $\lambda' = C/\sigma$. Thus, $\lambda' < \lambda$. Since $\lambda > \bar{\lambda}$, from Lemma 3 we know $\sigma < \bar{\sigma}$. Thus, $\frac{C}{\sigma} > \frac{C}{\bar{\sigma}}$, which implies $\lambda' > \bar{\lambda}$. Therefore, $\mu - C\sigma = \mu - \lambda'\sigma^2 < \mu' - \lambda'\sigma'^2 < \bar{\mu} - \bar{\lambda}\bar{\sigma}^2 = \bar{\mu} - C\bar{\sigma}$.

Thus, $\bar{\mu} - C\bar{\sigma}$ is the lower bound of optimal objective value for CC-LAP. Since the risk-averse parameter, λ^*, that gives the optimal assignment is a positive number, λ^* must be in the range $[0, \bar{\lambda}]$. □

The conclusion above can also be observed on variance-mean plane (shown in Fig. 3.4). Suppose $\bar{s}$ is the optimal solution of RA-LAP with $\bar{\lambda}$. Therefore $\bar{\mu} - C\bar{\sigma} = \bar{\mu} - \bar{\lambda}\bar{\sigma}^2$ and thus the vertical intercept of black parabola and straight line are located at same position $\bar{m}$. Suppose $\hat{\lambda}$ is any risk-averse parameter greater than $\bar{\lambda}$ and the dotted line $m'\bar{s}$ is the line through $\bar{s}$ with slope equal $\hat{\lambda}$. The optimal solution of RA-LAP with $\hat{\lambda}$, denoted by $\hat{s}$, must be located at the triangular region $\Delta\bar{m}\bar{s}m'$. The objective function value of CC-LAP with

solution in this region is less than $\bar{s}$, i.e., the vertical intercept of the parabola ($\hat{m}$ in Fig. 3.4) is less than $\bar{m}$. Therefore $\bar{\lambda}$ is an upper bound for risk-averse parameter and objective value of CC-LAP at $\bar{s}$ is lower bound for the optimal solution.

3.5 Solution Approach and Algorithm

Based on the results of Section 3.4, we now present a two-step approach for computing the optimal solution for CC-LAP. In the first step, we compute the range $[0,\ \bar{\lambda}]$ where the optimal λ^* is located. Then, we methodically generate and solve RA-LAPs with $\lambda \in [0,\ \bar{\lambda}]$. The optimal solution is the solution with the highest objective value among those solutions.

3.5.1 Bounding the Optimal Risk-aversion Parameter

Algorithm 1 Find the range

Input: $C, \mu_{ij}, \sigma_{ij}^2\ \forall i, j = 1, 2, \ldots n.$
Output: $\bar{\lambda}, S_1.$

1: Initialization: $k = 0$, $\lambda_k = 0$, $S_1 = \emptyset$.
2: Solve risk-averse problem with risk-averse parameter λ_k and obtain solution s_k, and μ_k, σ_k^2.
3: Store the solution in set $S_1 = S_1 \cup \{s_k\}$
4: **while** $\lambda_k \sigma_k \neq C$ **do**
5: Compute risk-averse parameter for the next iterations by $\lambda_{k+1} = \frac{C}{\sigma_k}$, $k = k + 1$.
6: Solve RA-LAP with λ_k, obtain s_k and μ_k, σ_k^2.
7: Store the solution $S_1 = S_1 \cup \{s_k\}$.
8: **end while**
9: **return** $\bar{\lambda} = \lambda_k$, S_1.

The first step is shown in Algorithm 1. The initial iteration index k is 0 and the initial risk-averse parameter λ_k is 0. The solution set S_1, initialized as a empty set (line 1). It starts to solve RA-LAP with λ_k (line 2). Then we can obtain the mean and variance of the team payoff with the obtained solution s_k, i.e., $\mu_k = \sum_{ij}^{n} \mu_{ij} f_{ij}$ and $\sigma_k = \sqrt{\sum_{i,j=1}^{n} \sigma_{ij}^2 f_{ij}}$. The obtained solution is stored in set S_1 (line 3). If the objective of RA-LAP is greater than the objective of CC-LAP, i.e., $\mu_k - \lambda_k \sigma_k^2 > \mu_k - C\sigma_k$ or equivalently $\lambda_k \sigma_k < C$, we iterate by solving RA-KAP with $\lambda_{k+1} = C/\sigma_k$ and storing the obtained solution (line 4-8). If both

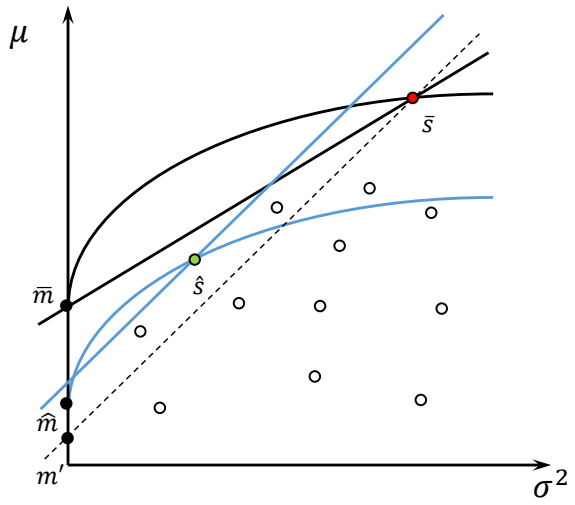


Figure 3.4: Illustration of Lemma 6. The solution, $\hat{s}$ obtained by solving RA-LAP with $\hat{\lambda} > \bar{\lambda}$ is within the triangular region $\Delta\bar{m}\bar{s}m'$. The objective function value of CC-LAP at $\hat{s}$ is equal to the vertical intercept of the level curve through $\hat{s}$, i.e., vertical coordinate of point $\hat{m}$, which is always less than $\bar{m}$, the value of solution $\bar{s}$.

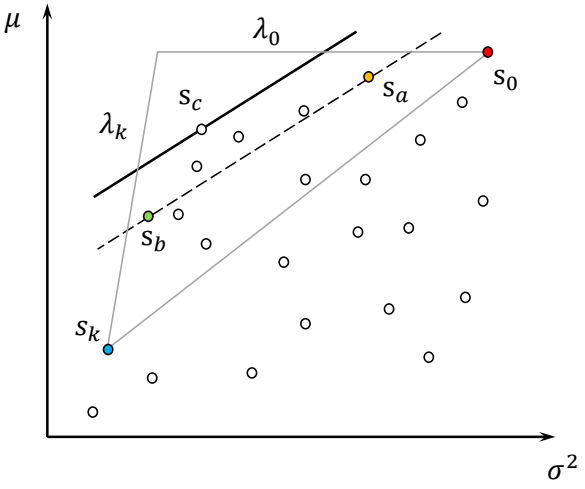


Figure 3.5: The search region is limited to this triangular region. The problem becomes enumerating extreme points in the search region. The risk-averse parameter λ is updated by connecting two adjacent extreme points, e.g. s_a and s_b, and λ is equal to the slope of the line connecting s_a and s_b.

objectives are same, i.e., $\lambda_k \sigma_k = C$, we stop with the upper bound of risk-averse parameter $\bar{\lambda}$ equal to λ_k (line 9).

In Algorithm 1, we are updating λ iteratively. We will now show that the update rule for λ, ensures that λ converges to the upper bound $\bar{\lambda}$.

Lemma 7. *In Algorithm 1, the risk-averse parameter λ converges to $\bar{\lambda}$, and thus, the algorithm terminates in a finite number of iterations.*

Proof. For $k = 0$, $\lambda_0 = 0$, and we obtain λ_1 as $\lambda_1 = \frac{C}{\sigma_0}$. Since C and σ_0 are both positive $\lambda_1 > \lambda_0$. For any $k > 0$, $\lambda_{k+1}\sigma_k = C$. However, $\lambda_k \sigma_k < C$. Therefore $\lambda_{k+1} > \lambda_k$. Thus, we have a strictly increasing sequence of values of λ, namely, $0 < \lambda_1 < \lambda_2 < \cdots$. Before termination, in each iteration a new extreme point is found by solving RA-LAP with λ_k. The algorithm terminates when the obtained extreme point is same with the previous iteration, which implies $\lambda_{k+1}\sigma_{k+1} = \lambda_{k+1}\sigma_k = C$. Since the number of the extreme points of the solution set is finite, λ_k converges to $\bar{\lambda}$ in finite number of step. Note that the corresponding variance, σ_k^2, is not monotonically decreasing and remains the same when λ is large enough. Since $\bar{\lambda}\bar{\sigma}$ is a constant, C, therefore, convergence of λ_k to $\bar{\lambda}$ implies convergence of $\lambda_k \sigma_k$ to C. Thus, Algorithm 1 terminates. □

Remark 2. *Note that Algorithm 1 is generating extreme points of the feasible set S. Since the point set S is finite, the number of extreme points in S is finite. Furthermore, when the algorithm has not converged, we are always generating a new extreme point at each step. Therefore Algorithm 1 converges in a finite number of iterations. The maximum number of iterations that Algorithm 1 will take to converge depends on the number of extreme points in S that can be generated by lines with slopes between 0 and $\bar{\lambda}$. However, this number is problem parameter dependent and it is hard to give a priori bounds. We show through extensive simulation in Section 3.7 that the number of steps to convergence is quite small and does not change much with an increase in the number of robots or tasks.*

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.

Let $S' = S_1$ and compute point pair set $S'_0 = \{(s_0, s_1), (s_1, s_2), ..., (s_{k-1}, s_k)\}$.
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 RA-LAP with λ.
        if the obtained solution say $s \notin \{s_{j-1}, s_j\}$ **then**
            Store s in S' and store point pairs (s_{j-1}, s), (s, s_j) in S'_{i+1}.
        end if
    end for
    $i = i + 1$.
end while
$s^* = \arg\max_{s_j \in S'} \ \mu_j - C\sigma_j$, $y^* = \max_{s_j \in S'} \ \mu_j - C\sigma_j$.
return s^*, y^*.

3.5.2 Searching for the Optimal Solution

Let $S_2 \subset S$ be the set of all extreme points of feasible point set S that can be generated by $\lambda \in [0, \bar{\lambda}]^2$. As stated in Lemma 6, the optimal solution s^* is an extreme point generated by some $\lambda \in [0, \bar{\lambda}]$. In Algorithm 1, we computed the set $S_1 \subseteq S_2$ of extreme points. In Algorithm 2, we present a procedure for generating the whole set S_2, starting from S_1.

Figure 3.5 presents a conceptual sketch, where s_0 and s_k are the points obtained by solving RA-LAP for $\lambda = 0$ and $\lambda = \bar{\lambda} = \lambda_k$ respectively. Since s_k is an extreme point for λ_k, there is no solution above the straight line with slope λ_k going through s_k. Similarly there is no solution above the horizontal line going through s_0. The search region for our next step thus reduce to a triangular area shown in Figure 3.5 where we need to find other extreme points in $S_2 \setminus S_1$. The procedure is presented in Algorithm 2. The input is the solution set S_1 obtained from the previous step including all extreme points computed so far. Then we need to find the rest of the extreme points in this search region. More precisely, we should find extreme points upon the straight line connecting any consecutive pair of solutions in S_1 if it

[2] In fact, S_2 is the set of all extreme points on the upper convex hull that can be generated by $\lambda \in [0, \bar{\lambda}]$. For the sake of simplicity description, we call it the set of all extreme points

exists, e.g., (s_a, s_b) in Figure 3.5. The solution in S_1 obtained in Algorithm 1 are stored in S'. 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), we can obtain the risk-averse parameter λ by computing the slope of 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 5). The algorithm solves RA-LAP with λ (line 6). 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 set S'. Two new point pairs (s_{j-1}, s) and (s, s_j) are stored in S'_{i+1} (line 7-8). If the obtained solution is same with s_{j-1} or s_j, there is no extreme above the line through s_{j-1} and s_j. After solving all RA-LAPs generated from S'_i, algorithm checks whether S'_{i+1} is empty. If it is not empty, in other words there is new extreme point obtained from iteration i, we move on to next iteration $i+1$ and repeat the same procedures starting from line 3. It stops when S'_{i+1} is empty, which means no new extreme point is obtained and we have found all extreme points in the search region, i.e., $S' = S_2$. The optimal solution is the solution with the highest team payoff value in extreme point set S' (line 13).

Lemma 8. *Algorithm 2 terminates in a finite number of iterations.*

Proof. In Algorithm 2, by solving the RA-LAP we either obtain a new extreme point or an extreme point that was already obtained previously. If the latter happens, the procedure stops generating new RA-LAPs. The number of new extreme points obtained by Algorithm 2 is $|S_2| - |S_1|$. The number of RA-LAPs that produce the repeated solutions in Algorithm 2 is $|S_2| - 1$. The total number of RA-LAPs solved in Algorithm 2 is thus $2|S_2| - |S_1| - 1$. Since $|S_2|$ is finite and $|S_1| \leq |S_2|$, Algorithm 2 solves a finite number of RA-LAPs and terminates in a finite number of iterations. □

3.6 Distributed Algorithm for Chance-constrained Linear Assignment

We will now present a distributed version of our algorithm that requires no central unit or shared memory. The distributed algorithm implements the first step of our centralized

algorithm and obtains an almost optimal solution (results are presented in Section 3.7). We assume that each robot has a range-limited wireless communication capability. For simplicity of exposition, we also assume that if robot r_i can communicate with r_j, then robot r_j can communicate with r_i. Thus, the communication network of n robots can be modeled as an undirected graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, n\}$ are the set of nodes representing the robots in the network and $\mathcal{E} = \{(i, j)|\ i, j \in \mathcal{V}\}$ denotes the set of edges connecting two nodes. An edge (i, j) exists in $\mathcal{G}$, if robot r_i and r_j can communicate with each other directly. We assume that $\mathcal{G}$ is a connected graph, i.e., there is a (possibly multi-hop) communication path between any two robots in the team. We denote the neighbors of node i by $\mathcal{N}_i = \{j|\ (i, j) \in \mathcal{E}\}$.

3.6.1 Algorithm Overview

Recall that our centralized algorithm consists of two separate parts, namely, solving RA-LAP for a given value of the risk-averse parameter λ, and updating risk-averse parameter λ. For the first part, we can use the auction algorithm proposed in [12] to solve RA-LAP in a distributed fashion without the shared memory. However, CC-LAP is solved by solving a sequence of RA-LAPs with different risk-averse parameter λ. To implement the update rule for λ (line 5 in Algorithm 1) in a distributed manner, each robot should have the knowledge of the variance of the total payoff of the assignment, i.e., $\sum_{i=1}^{n} \sigma_{ij}^2 f_{ij}$. Therefore, we modify the auction algorithm in [12], to design a distributed algorithm (Algorithm 4) that not only solve RA-LAP for a given λ but also enable each robot to evaluate the variance of the total payoff in the computed assignment and thus update risk-averse parameter for next iteration on its own.

Initial Knowledge of the Robots: We assume that each robot, r_i, only knows the pre-specified probability p (hence C), and means and variances of it's own payoffs for performing any task t_j, i.e., μ_{ij} and $\sigma_{ij}^2,\ \forall j$.

Algorithm 3 describes the computations by each robot r_i in our distributed algorithm.

The risk-averse parameter is initialized as 0 (line 1). At any iteration k (line 2 or 6), robot r_i solves RA-LAP with λ_k by auction algorithm $\mathcal{A}(\lambda_k)$ (which is described in Algorithm 4). The outputs are the variance of the total payoff of the assignment, σ_k^2 and its assigned t_{α_i} where α_i denotes the index of its assigned task. If $\lambda_k \sigma_k \neq C$, robot updates risk-averse parameter by $\lambda_{k+1} = \frac{C}{\sigma_k}$ (line $4-5$) and continues to solve RA-LAP (line 6). Otherwise the algorithm stops and the assigned task in the last RA-LAP solved is the assignment for r_i (line 8).

Algorithm 3 The distributed algorithm for robot r_i solving CC-LAP

Input: $C, \mu_{ij}, \sigma_{ij}^2 \ \forall j = 1, 2, \ldots n$.
Output: The assigned task, $t^*_{\alpha_i}$, in the (near) optimal solution.
1: Initialization: $k = 0$, $\lambda_k = 0$.
2: Solve RA-LAP with λ_k by auction algorithm $\mathcal{A}(\lambda_k)$ and obtain σ_k and assigned task t_{α_i}.
3: **while** $\lambda_k \sigma_k \neq C$ **do**
4: Update risk-averse parameter by $\lambda_{k+1} = \frac{C}{\sigma_k}$.
5: $k = k + 1$.
6: Solve RA-LAP with λ_k by auction algorithm $\mathcal{A}(k)$, obtain σ_k and assigned task t_{α_i}.
7: **end while**
8: **return** $t^*_{\alpha_i} = t_{\alpha_i}$.

3.6.2 Distributed Algorithm for LAP with a Given Risk-avers Parameter

In this subsection, we present the distributed auction algorithm presented for solving RA-LAP with a given value of λ. We modify the auction algorithms from [27] and [12] so that each robot can also compute the variance of the team payoff at the end of the auction procedure. Nevertheless, we present the algorithm here for completeness.

For any given value of λ, let $\ell_{ij} = \mu_{ij} - \lambda \sigma_{ij}^2$ be the payoff for robot r_i performing task t_j. Let p_j be an auxiliary variable associated with each task t_j, which is also called the price of the task t_j. Every robot prefers to perform the task with the greatest net value, i.e., $\max_{1 \leq j \leq n} \{\ell_{ij} - p_j\}$. The form of auction algorithm proposed by Bertsekas in [27] is an iterative method to find the optimal prices and an assignment that maximizes the net value.

Algorithm 4 Auction algorithm $\mathcal{A}(\lambda_k)$ for robot r_i

Input: Risk-averse parameter λ_k.
Output: Assigned task t_{α_i}, variance of the total payoffs σ_k^2.

Initialization: $\ell_{ij} = \mu_{ij} - \lambda_k \sigma_{ij}^2, \forall j$.
Compute initial assigned task by $\alpha_i = \arg\max_{1 \le j \le n_t} \ell_{ij} - p_{ij}$ and assigned task variance by $\beta_{i\alpha_i} = \sigma_{i\alpha_i}^2$.
while $\{p_{ij}\}_{j=1}^{n_t}$ changes in the last Δ auction iterations **do**
    Extract message $\{p_{hj}, b_{hj}, \beta_{hj}\}_{j=1}^{n_t}$ from neighbors, i.e., $\forall h \in \mathcal{N}_i$.
    for each task t_j **do**
        Find the neighbor r_d offering the highest price for t_j, where $d = \arg\max_{h \in D} b_{hj}$, $D = \arg\max_{h \in \{i, \mathcal{N}_i\}} p_{hj}$.
        Update local variables $\{p_{ij}, b_{ij}, \beta_{ij}\}$ by $p_{ij} = p_{dj}$, $b_{ij} = b_{dj}$ and $\beta_{ij} = \beta_{dj}$.
    end for
    if $p_{i\alpha_i}$ increases or $p_{i\alpha_i}$ unchanged but $b_{i\alpha_i} \neq i$ **then**
        Update the assigned task t_{α_i} by $\alpha_i = \arg\max_{1 \le j \le n_t} \ell_{ij} - p_{ij}$.
        Increase the price for the assigned task $p_{i\alpha_i}$ by γ_i.
        Update $b_{i\alpha_i} = i$ and $\beta_{i\alpha_i} = \sigma_{i\alpha_i}^2$.
    end if
    Send local variables $\{p_{ij}, b_{ij}, \beta_{ij}\}_{j=1}^{n_t}$ to neighbors.
end while
return t_{α_i} and $\sigma_k^2 = \sum_{j=1}^{n_t} \beta_{ij}$.

At the end of the auction algorithm, every robot is at *almost equilibrium*:

$$a_{i\alpha_i} - p_{\alpha_i} \geq \max_{1 \leq j \leq n} \{\ell_{ij} - p_j\} - \epsilon \tag{3.16}$$

The total payoff of the final solution is within $n\epsilon$ of being optimal. With the proper choice of ϵ and scaling of payoffs as described in [27], the solution obtained is optimal.

Unlike the auction algorithm that uses a shared memory network in [27] where each robot has access to the global task price p_j, a distributed algorithm presented in [12] does not require a shared memory. Each robot r_i has a local copy of task price p_{ij} and it updates the prices of all tasks with information from its neighbors. We use $\mathcal{N}_i$ to denote the set of indices of r_i's neighbors.

In our algorithm, we do not assume the shared memory. In order to update the risk-averse parameter λ as described in Algorithm 3, our auction algorithm needs to compute the sum of the variances of the payoff of the assignment, i.e., $\sum_{i=1}^{n_r} \sigma_{i\alpha_i}^2$. Therefore, we consider that each robot r_i has local variables $\{p_{ij}, b_{ij}, \beta_{ij}\}_{j=1}^{n_t}$. These local variables are also in the message communicated by a robot to its neighbors. In particular, p_{ij} is the price that r_i has to pay to be assigned to t_j, b_{ij} is the index of the highest bidder for task t_j and β_{ij} is the variance of task t_j in the assignment. Each robot r_i first computes the values that are used as payoffs in the auction by $\ell_{ij} = \mu_{ij} - \lambda\sigma_{ij}^2$ based on the input λ_k (line 1). Then r_i computes its initial assigned task t_{α_i} which has the maximum net value and variance of task t_{α_i} by $\beta_{i\alpha_i} = \sigma_{i\alpha_i}^2$ (line 2).

The computation and communication done by each robot in one iteration of auction procedure is described from line 4 to line 14. For each task t_j, robot r_i determines the robot(s) offering the highest price in the neighborhood denoted by r_d (line 5-6). Note that because there is no shared memory, a *tie* in the bids for a task could happen, i.e., there could be multiple robots, denoted by $h \in D$, that have same highest price. We choose a consistent protocol for all robots to break the tie, that is the robot with the largest index of bidder b_{hj} for t_j is identified as r_d, i.e., $d = \arg\max_{h \in D} b_{hj}$. Next r_i updates the local variables for task t_j (line 7). After updating local variables for all tasks, r_i determines whether the

updated price of the current assigned task $p_{i\alpha_i}$ increases or the price does not change but the highest bidder of the task $b_{i\alpha_i}$ changes due to the tie breaking rule (line 9). If yes, r_i should re-compute the assigned task t_{α_i} with the highest net value based on the updated prices (line 10). Let q_i and w_i denote the highest and second highest net value for r_i. Then the price of the updated assigned task t_{α_i} is increased by $\gamma_i = q_i - w_i + \epsilon$ (line 11). Now, in its neighborhood, r_i provides the highest price for the assigned task t_{α_i}. Therefore the highest bidder of the assigned tasks $b_{i\alpha_i}$ and the task variance $\beta_{i\alpha_i}$ are updated to i and $\sigma^2_{i\alpha_i}$ respectively (line 12). Then r_i sends updated local variables to its neighbors (line 14).

The auction iteration continues until the local prices do not change for Δ iterations where $\Delta \leq n-1$ is the maximum possible diameter of the network (line 3). At this moment, all robots are at *almost equilibrium* (because of ϵ). The local knowledge of the task prices and variances $\{p_{ij}, \beta_{ij}\}_{j=1}^{n_t}$ of all the robots are the same. Each robot r_i outputs its assigned task t_{α_i} and the variance of total payoff, $\sigma_k^2 = \sum_{j=1}^{n_t} \beta_{ij}$. As stated earlier this algorithm is designed based on earlier work in [27, 12] with modifications made to keep track of the variance of the assigned tasks in the final assignment. Since the core computations for assigning tasks have not changed, the convergence proof and complexity analysis in [27, 12] are still valid for our algorithm.

3.7 Simulation Results

In Sections 3.5 and 3.6, we provided both centralized and distributed algorithms for solving the CC-LAP. The key idea in our approach is to solve multiple instances of the RA-LAP to compute a solution for the CC-LAP. Therefore, a good measure of the efficiency of the algorithms is the number of RA-LAP problems that are solved. Moreover, we proved and presented an upper bound, $\bar{\lambda}$, for the risk-averse parameter. We would like to know the closeness of the solution obtained from $\bar{\lambda}$ to the optimal solution from λ^*. In distributed algorithms, robots solve RA-LAP with the auction algorithm. We use the number of total rounds of message communications and computations for solving CC-LAP (in other words,

the sum of the number of rounds for convergence of multiple auctions) as a measurement efficiency of the distributed algorithm. The probability in the chance constraint of CC-LAP (3.2) is set to be 0.99. In this section, we use the above metrics to evaluate the performance of the centralized algorithm and the distributed algorithm. The simulations are done on a computer with Intel i7-4720HQ CPU 2.60GHz and 16.0 GB RAM. Without loss of generality, we assume that the number of robots is the same as the number of tasks.

The environment of the simulation is shown in Fig. 3.1 and 3.6. Each package is generated randomly with size and weight, and the locations of loading and unloading stations. In particular, the size and weight are categorized as small, medium and large. The loading and unloading stations of packages are randomly located inside the picking up area (near the L-shape conveyor) and dropping off area (near the two rectangular containers). The robots move from the initial charging stations and pick up packages from loading stations and deliver the packages to the unloading stations with the same indices as the loading stations. The means of the assignment payoffs are generated based on the distance of the route from the charging station to the loading and unloading station. The variances of assignment payoff are generated based on the weight and size of the package. (e.g., the package with a large size and weight has larger variance). More precisely, the means and variance are created from eight intervals shown in Table 3.1. Based on the distance of the route, each assignment of the robot to a task is associated with an interval of the mean and an interval of the variance accordingly (the longer the distance is, the interval of the mean with higher magnitude is selected). Then the variance of assignment payoff is selected from the corresponding interval of variance based on the weight and size of the package. For example, if an assignment of the robot to a task has the longest distance, the mean of the payoff is created from the uniform distribution $\mathcal{U}(90, 100)$ and the variance is created from interval $[25^2, 30^2]$. Further, if the package is large and heavy, the variance has a higher value within the interval. One solution

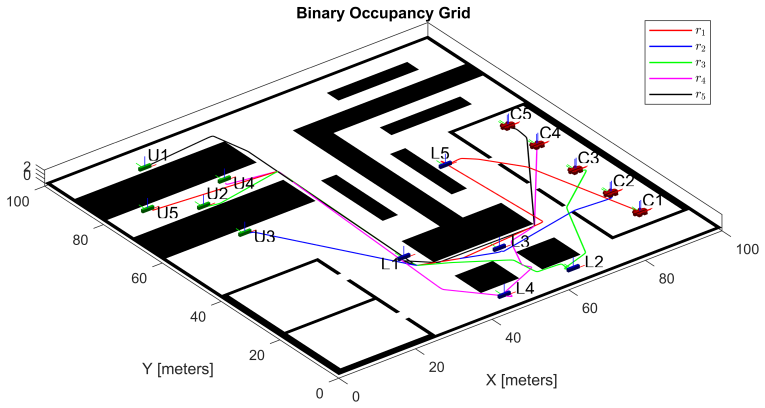


Figure 3.6: The environment of the simulation. The robots are initially located at the charging area. The loading stations are randomly located near the L-shape conveyor. The unloading stations are randomly located near the two rectangular containers. In this example, robot 1 to 5 are assigned to package 5, 3, 2, 4, 1 respectively.

under this environment is presented in Fig. 3.6.

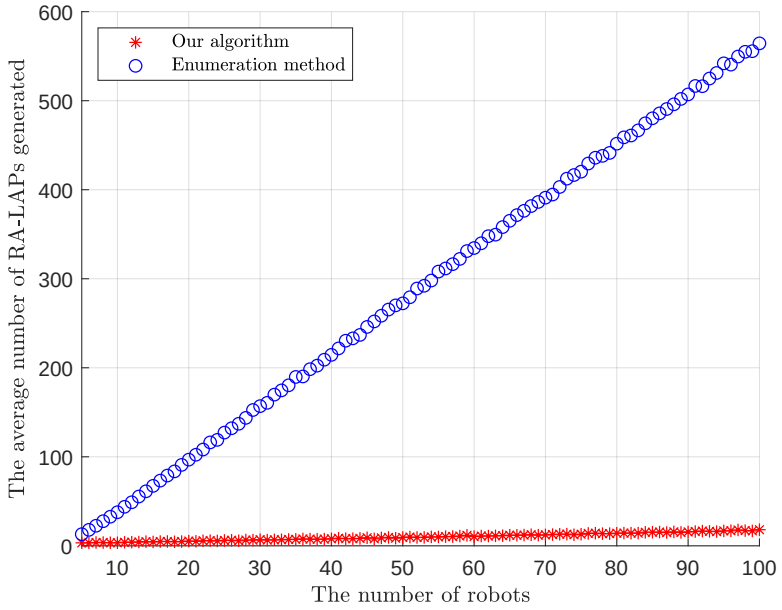
Table 3.1: The scheme to generate means and variances of payoffs

μ_{ij}	$[20, 30)$	$[30, 40)$	$[40, 50)$	$[50, 60)$	$[60, 70)$	$[70, 80)$	$[80, 90)$	$[90, 100]$
σ_{ij}^2	$[2^2, 3^2]$	$[5^2, 6^2]$	$[8^2, 9^2]$	$[12^2, 14^2]$	$[16^2, 17^2]$	$[19^2, 20^2]$	$[23^2, 26^2]$	$[25^2, 30^2]$

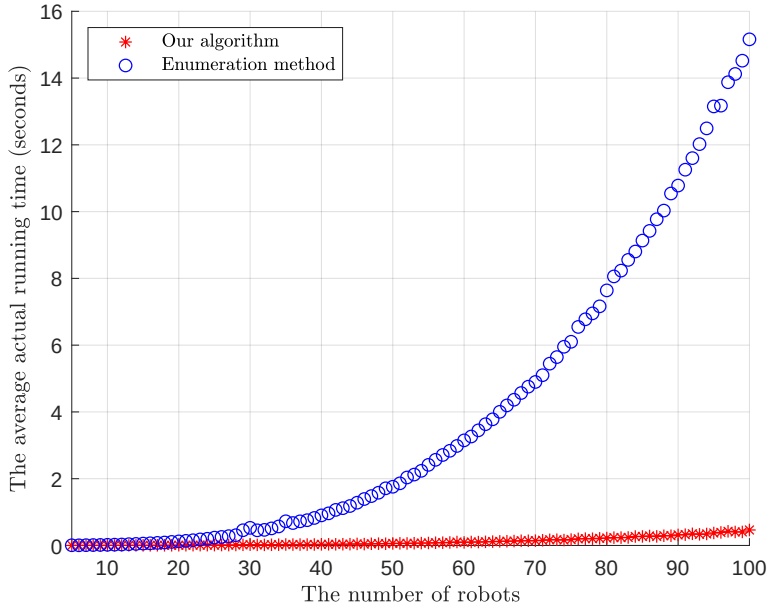
3.7.1 Centralized Algorithm

We measure the scalability of our centralized algorithm producing both approximate solution and exact solution to the number of robots (tasks) in Fig. 3.8a, and evaluate the closeness of the solution obtained from $\bar{\lambda}$ in Fig. 3.8b. Nikolova presented algorithms for risk-averse combinatorial optimization in which CC-LAP is an instance of the class of problems. The exact algorithm in [14] can also be used to solve the CC-LAP. We compare the performance of our exact algorithm and exact algorithm (called enumeration method in this thesis) mentioned from [14] in Fig. 3.7a. We also measure the actual running time of algorithms to provide a sense of performance in practice in Fig. 3.7b.

We use our algorithm to solve a large number of CC-LAPs with an increasing number

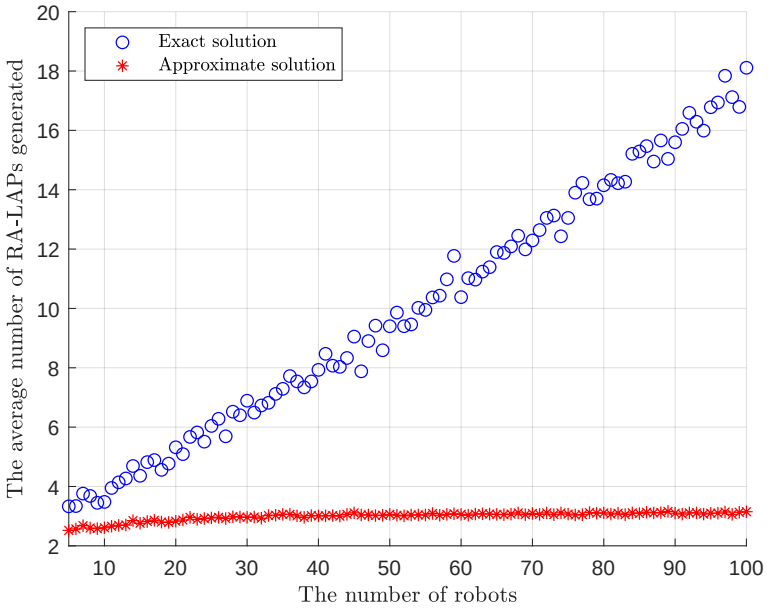


(a) The number of RA-LAPs solved by our algorithm and enumeration method with respect to the number of robots.

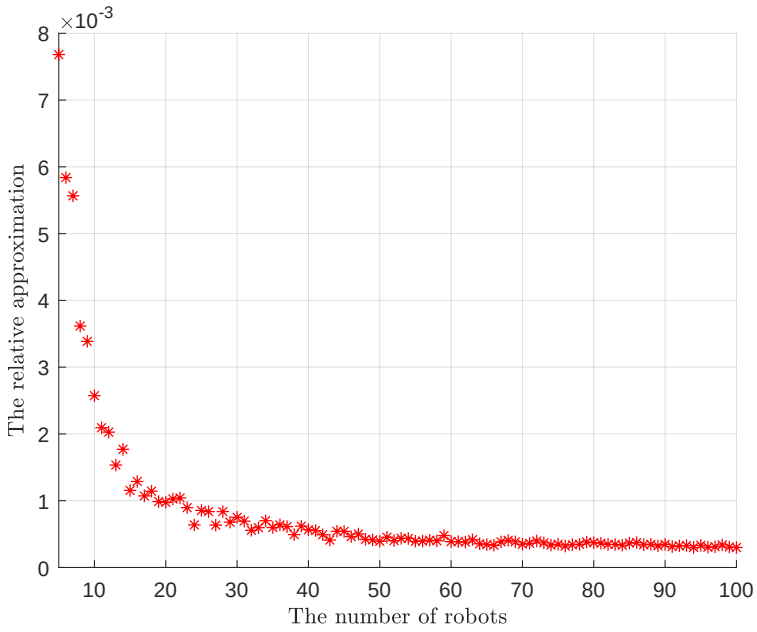


(b) Actual running time for our algorithm and enumeration method with respect to the number of robots.

Figure 3.7: The scalability of the centralized algorithm as a function of the number of robots and tasks.



(a) The number of RA-LAPs solved by our complete two-step algorithm and the approximate algorithm.



(b) The relative difference of approximate solution and the exact solution with respect to the number of robots.

Figure 3.8: The comparison between the complete two-step algorithm which produces the exact solution and the approximate algorithm that implements the first step of our algorithm.

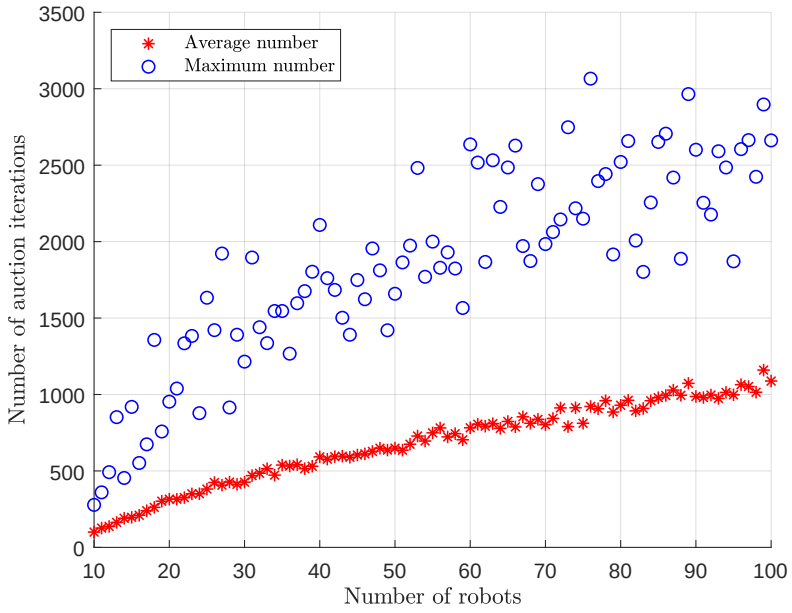
of robots from 5 to 100 with an increment of 1. We assume the number of robots and tasks are the same. The problem with a given number of robots is solved for 100 times with different randomly generated instances including means and variances of payoffs. We compute the average number of calls to the risk-averse problem. We also use the method that enumerates all extreme points on the variance-mean plane to solve CC-LAP under the same setup and parameters. Figure 3.7a presents a comparison between our procedure for computing the optimal solution and the method from [14]. The number of calls to the risk-averse problem increases for both algorithms linearly with the number of robots. However, the rate of growth for our algorithm is much smaller compared to [14]. Further, the actual running time for both our algorithm and exact algorithm in [14] are provided in Fig. 3.7b.

Figure 3.8a measures the scalability of Algorithm 1 with the number of robots and compared it with our complete two-step algorithm. Figure 3.8b evaluates the closeness of the solution obtained from Algorithm 1 by computing the relative difference[3] between the objective values of the solution obtained from $\bar{\lambda}$ and the optimal objective. As we can see from Figure 3.8a, the number of calls to RA-LAPs is relatively constant, whereas the number of calls to RA-LAP, although small, increases linearly. Furthermore, from Figure 3.8b the relative difference between the two solutions is very small (of the order of 1×10^{-03}). This seems to suggest that in practical situations, Algorithm 1 produces a good enough solution with a fewer number of RA-LAPs solved.

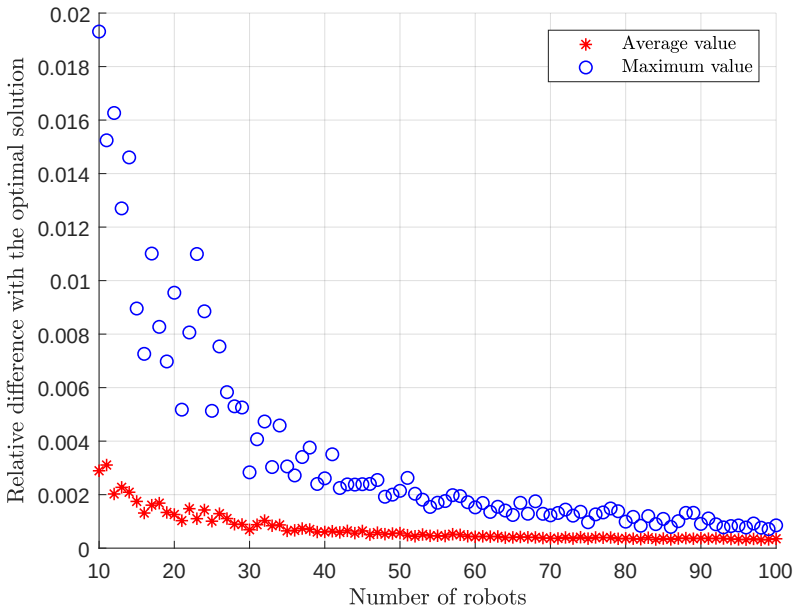
3.7.2 Distributed Algorithm

In the distributed algorithm, robots proceed multiple times the auction algorithm for solving RA-LAPs and obtaining the sum of the variance to update the risk-averse parameter λ. For any single auction algorithm, robots need to proceed with a number of auction iterations until the convergence. In this set of simulations, we want to study the scalability of our distributed algorithm to the number of robots. We will use the total number of

[3] The relative difference is defined as $\frac{V^*-V}{V^*}$ where V is the evaluated objective function value and V^* is the optimal objective function value.



(a) The number of auction iterations taken for solving CC-LAP. The average number and the maximum number of iterations are computed from 100 randomly generated instances.



(b) The relative difference of the approximate solution obtained from the distributed algorithm and the optimal solution.

Figure 3.9: The scalability of the distributed algorithm to the number of robots and the closeness of the obtained solution to the optimal solution.

auction iterations as a measurement of the convergence time for our distributed algorithm solving CC-LAP. For example, let the number of RA-LAPs solved be u and for solving each RA-LAP the iterations taken in the auction algorithm is q_i we compute the total number of auction iterations, i.e., $\sum_{i=1}^{u} q_i$.

To create a communication network, the robots are randomly located in the charging area. Any two robots are able to communicate with each other if the distance between them is within the communication range. The chosen communication network are connected graph. The value of ϵ in the auction algorithm is 0.1. As shown in Fig. 3.9a, for any number of the robots in the range from 10 to 100, we compute the average and the maximum number of auction iterations over 100 randomly generated instances. The average number of auction iterations grows linearly. Since our distributed algorithm implements only the first step of our algorithm and the auction algorithm provides a sub-optimal solution because of ϵ, we obtain an approximate solution from the distributed algorithm. In Fig. 3.9b, we present the relative differences of the approximate solution obtained from the distributed algorithm with the exact solution obtained from our centralized solution. The average and the maximum value are obtained from the same instances in Fig. 3.9a. The average values are all less than 0.4%. The results show that the distributed algorithm is scalable to the number of robots and the solution is nearly optimal. Note that our distributed algorithm is designed based on the auction algorithm in [12]. Its performance is also influenced by the choice of ϵ and initial prices. With the proper choice of ϵ, we can obtain results with a smaller number of auction iterations or a more accurate solution with the cost of a larger number of auction iterations. However, it is not our main point in this simulation. Our contribution is that we design an algorithm that solves CC-LAP by solving a *small* number of RA-LAP which influences the total number of auction iterations used for solving CC-LAP.

3.8 Relaxing the Gaussian Assumption

Thus far, in our presentation, we have assumed that individual robot-task payoffs are Gaussian random variables. In this section, we discuss how we can relax this assumption and consider the case that we only know the first two moments (i.e., mean and variance) of each robot-task payoff pair.

In such situations, where we have no further information on the payoff distributions, we can use $C = \sqrt{\frac{p}{1-p}}$ (instead of $C = \Phi^{-1}(p)$) in the mean-risk formulation given by Equation (3.5). This would ensure that the chance constraint in (3.2) is satisfied, as we show below. By the one-sided Chebyshev inequality (or Cantelli's lemma), we have

$$\mathbb{P}\left(\sum_{i,j}^{n} a_{ij}f_{ij} \leq \sum_{i,j}^{n} \mu_{ij}f_{ij} - A\right) \leq \frac{\sum_{i,j}^{n} \sigma_{ij}^2 f_{ij}}{\sum_{i,j}^{n} \sigma_{ij}^2 f_{ij} + A^2}$$

Let $A = C\sqrt{\sum_{i,j}^{n} \sigma_{ij}^2 f_{ij}}$. The inequality above becomes

$$\mathbb{P}\left(\sum_{i,j}^{n} a_{ij}f_{ij} \leq \sum_{i,j}^{n} \mu_{ij}f_{ij} - C\sqrt{\sum_{i,j}^{n} \sigma_{ij}^2 f_{ij}}\right) \leq \frac{1}{1+C^2}$$

and equivalently

$$\mathbb{P}\left(\sum_{i,j}^{n} a_{ij}f_{ij} > \sum_{i,j}^{n} \mu_{ij}f_{ij} - C\sqrt{\sum_{i,j}^{n} \sigma_{ij}^2 f_{ij}}\right) \geq 1 - \frac{1}{1+C^2}$$

For the right hand side to be p, we need to choose $C = \sqrt{\frac{p}{1-p}}$. Consequently, we get

$$\mathbb{P}\left(\sum_{i,j}^{n} a_{ij}f_{ij} > \sum_{i,j}^{n} \mu_{ij}f_{ij} - C\sqrt{\sum_{i,j}^{n} \sigma_{ij}^2 f_{ij}}\right) \geq p$$

Therefore the optimization problem in Equation (3.5) with $C = \sqrt{\frac{p}{1-p}}$ gives us a solution for the CC-LAP.

3.9 Conclusion

In this chapter, we presented algorithms for computing solutions with a probabilistic team performance certificate for task allocation in multi-robot systems with payoff uncertainty. We formulated a chance-constrained linear assignment problem and developed a

novel deterministic technique to solve this chance-constrained problem. Adopting the notion of risk-aversion from the economics literature, we formulate a risk-averse task allocation problem. We prove that by repeatedly solving the risk-averse task allocation problem using a one-dimensional search on the risk-averse parameter we find a solution for the chance-constrained optimization problem. We provide simulation results on randomly generated scenarios to demonstrate our approach and also compare our method to existing approaches. The geometric insight of the chance-constrained linear assignment problem in this chapter is the key idea to solved other chance-constrained multi-robot coordination problems in this thesis.

Chapter 4

Chance-constrained Multi-robot Simultaneous Task Assignment and Path Planning with Sum Objective

In Chapter 3, we discussed a basic version of the chance-constrained multi-robot task allocation, namely, the chance-constrained linear assignment problem. It is assumed in CC-LAP that the means and variances of the payoffs for robots performing tasks are available before allocating the tasks. The application scenarios include part packaging, material handling, and security patrolling. However, in many application scenarios, the payoff information might not be given *a priori*, especially when the tasks are spatially distributed. In this chapter, we discuss a class of multi-robot task coordination that has applications in the automated warehouse, factory floors, and mobility-on-demand with autonomous vehicles. As shown in the motivation example in Fig. 4.1, 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 its assigned task location. Given the mean and variance of travel cost of each edge, our goal is to develop algorithms that, with high probability, the total path cost of the robot team is below a minimum value in any realization of the stochastic travel costs.

4.1 Introduction

Multi-robot task coordination problems where robots have to move to target destinations arise in a number of applications including search and rescue, and goods or parts transfer in warehouses. In such scenarios, the robots have to navigate in environments containing both static and dynamic obstacles. There are two related problems to be solved,

namely, (a) multi-robot task allocation problems, wherein, robots have to be assigned to a destination, (b) multi-robot path planning problems, wherein, collision-free paths have to be planned for each robot between their origin and destination. The two problems are usually decoupled. In task allocation problems, it is commonly assumed that the cost of the paths between robot-destination pairs are known, which implicitly implies that a single collision-free path has been pre-computed between each robot-destination pair. In path planning problems the origin and destination of each robot are usually given, which implicitly implies that the task assignment for each robot has been computed.

Further, for task allocation, the path costs are usually assumed to be deterministic. However, the path costs may be stochastic, especially in *open environments*, where other (uncontrolled) mobile agents like people or other cars can occupy the space. 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. The decoupling of the task allocation and path planning problems, although conceptually convenient can lead to sub-optimal solutions. Therefore, in this paper, we consider robot task coordination problems where the robots have to simultaneously plan paths and select target destinations (or tasks) under uncertainty about the travel costs.

We assume that the robots move on a graph. The graph may represent an actual road network or a roadmap which captures the collision-free configuration space of the robots (see Fig. 1.1 and 1.2). It is assumed that robots have local collision avoidance schemes to avoid mobile obstacles. There are many algorithms like PRM [3] or RRT [4] or their optimal variants PRM* and RRT* [5] that can generate the roadmaps for any given environment (see Fig. 1.3). We assume that the cost on each edge of the graph is a random variable with known mean and variance. Thus the total path cost of the robot team is a random variable. Our goal is to *simultaneously 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 cost objective such that we have a guarantee that the task execution cost of the robot team will be less than y with high*

probability (say 0.99*) under any realization of the random costs.* Such a solution will provide a quality guarantee (albeit probabilistic) on the solution of the simultaneous task assignment and planning (STAP) problem in the presence of uncertainty about the task execution costs.

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. We prove that the optimal solution of CC-STAP can be obtained by solving a sequence of deterministic simultaneous task assignment and path planning (D-STAP) problems, in which the travel cost of each edge is a linear combination of mean and variance of the edge cost. We show that the D-STAP problem can be solved by first computing the cost of the shortest paths to the task locations for each robot-task pair and then solving a linear assignment problem with the shortest path costs. The algorithms to solve CC-STAP optimally and also the algorithm to solve D-STAP optimally are the primary contributions of this work. Based on the work of CC-LAP in Chapter 3, we also present a distributed algorithm to solve CC-STAP that builds on the auction algorithm for solving linear assignment problems [11, 12].

To the best of our knowledge, there are no available algorithms with theoretical guarantees on solution quality that can solve the combined task assignment and planning problem with stochastic costs. In the extant literature, there are centralized algorithms such as the Hungarian algorithm [25], distributed algorithms with shared memory [11], and totally distributed algorithms [12] for solving deterministic task assignment problem. In [65], 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 [69], a redundant robot assignment on graphs with uncertain edge costs is studied. In [60], authors present a distributed chance-constrained task allocation framework. There has been some effort in solving different variations of the stochastic shortest path problem [45, 44, 70, 71, 46, 49, 13, 50, 47, 48], in which one robot has to plan its motion to a destination node with random costs on the edges. In [72], 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 not only path planning but also the task assignment which is not predefined. In the deterministic setting combined goal assignment and collision-free trajectory planning problem has been studied in [73, 74, 75]. 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.

4.2 Problem Formulation

We are given a graph, $G = (V, E)$, a set of N_r heterogeneous robots, $\{r_i\}_{i=1}^{N_r}$, with known initial positions, and a set of N_t tasks or target destinations, $\{t_j\}_{j=1}^{N_t}$. Let ${}^i c_{uv}$ be the travel cost of a robot, r_i, through an edge, e_{uv}. We assume that ${}^i c_{uv}$ is a random variable with known mean ${}^i\mu_{uv}$ and variance ${}^i\sigma_{uv}^2$. The goal is to compute the path for each robot to a unique target such that the total path cost of robot team is less than a team cost value y with probability at least p. Our objective is to minimize the team cost value y.

$$
\begin{aligned}
&\min \quad y \\
&\text{s.t. } \mathbb{P}\left(\sum_{i=1}^{N_r}\sum_{u,v=1}^{|V|} {}^i c_{uv}\ {}^i x_{uv} \le y\right) \ge p \\
&\sum_{v\in\mathcal{N}(u)} {}^i x_{uv} - \sum_{v\in\mathcal{N}(u)} {}^i x_{vu} = \begin{cases} 1, & \text{if } u = s_{r_i}, \\ -z_{ij}, & \text{if } u = t_j, \ \ \forall i \\ 0, & \text{otherwise} \end{cases} \qquad (4.1) \\
&{}^i x_{vu} \in \{0,1\}, \quad \forall u, v \in V,\ i = 1, \ldots, N_r. \\
&\sum_{i=1}^{N_r} z_{ij} = 1,\ \forall j; \quad \sum_{j=1}^{N_t} z_{ij} = 1,\ \forall i; \quad z_{ij} \in \{0,1\}\ \forall i, j
\end{aligned}
$$

The solution to this problem includes two parts: assignments, indicated by decision variable $\{z_{ij}\}, \forall i, j$, and paths, indicated by $\{{}^i x_{uv}\}, \forall u, v \in V, \forall i$. In particular, $z_{ij} = 1$ when r_i is assigned to t_j, and ${}^i x_{uv} = 1$ when edge e_{uv} is an edge on the path of r_i. The probabilistic guarantee on the performance of the robot team is ensured by the chance constraint which guarantees that the total travel cost for all robots, $\sum_{i=1}^{N_r}\sum_{u,v=1}^{|V|} {}^i c_{uv}\, {}^i x_{uv}$, is at most the cost

value y in any realization of the random travel cost with probability at least p. The second set of constraints is the path constraints for every robot. In particular, for any robot r_i the difference in the number of the edges along the path leaving and entering one node, say u, is equal to 1 when u is the source node s_{r_i}, equal to $-z_{ij}$ when u is a node for any task, say t_j, and equal to 0 otherwise. Thus, if $u = t_j$, the difference is equal to -1 when t_j is the task assigned to r_i and is 0 when t_j is not assigned to r_i (in other words u is an intermediate point on the path or a point not on the path). The constraints for z_{ij} are the assignment constraints that each robot performs only one task and each task is assigned to one robot.

The chance constraint can be written as inequality (4.2) based on different assumptions on the probability distribution of the travel cost. If the travel costs are independent Gaussian random variables, i.e., ${}^ic_{uv} \sim \mathcal{N}({}^i\mu_{uv}, {}^i\sigma^2_{uv})$, the total path cost for the robot team is a Gaussian random variable with mean and variance equal to the sum of means and variances of the edges along the paths. Following the arguments in Chapter 3, the chance constraint becomes:

$$\sum_{i=1}^{N_r} \sum_{u,v=1}^{|V|} {}^i\mu_{uv}\ {}^ix_{uv} + C\sqrt{\sum_{i=1}^{N_r} \sum_{u,v=1}^{|V|} {}^i\sigma^2_{uv}\ {}^ix_{uv}} \leq y \tag{4.2}$$

where $C = \Phi^{-1}(p)$ and $\Phi^{-1}(\cdot)$ denotes the inverse cumulative distribution function of the standard Gaussian distribution $\mathcal{N}(0, 1)$.

Note that the independence and Gaussian assumptions are for simplification and brevity of the mathematical exposition. Our framework can also consider scenarios where only means and variances of travel cost distributions are available, and any other distributional information is absent. In particular, from Chebyshev's inequality, the chance constraint can be replaced by (4.2), where $C = \sqrt{\frac{p}{1-p}}$. Further with appropriate graph transformation, our method could be extended to the case where the travel costs are dependent. A graph transformation example was provided in [14].

Keep constraints of ${}^ix_{uv}$ and z_{ij} in (4.1) except the chance constraint. Let the objective value y equal to the left-hand side of inequality (4.2). The formulation in (4.1) can be

equivalently written as

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{N_r} \sum_{u,v=1}^{|V|} {}^i\mu_{uv}\, {}^ix_{uv} + C\sqrt{\sum_{i=1}^{N_r} \sum_{u,v=1}^{|V|} {}^i\sigma^2_{uv}\, {}^ix_{uv}} \\
\text{s.t.} \quad & \sum_{v\in\mathcal{N}(u)} {}^ix_{uv} - \sum_{v\in\mathcal{N}(u)} {}^ix_{vu} = \begin{cases} 1, & \text{if } u = s_{r_i}, \\ -z_{ij}, & \text{if } u = t_j, \\ 0, & \text{otherwise} \end{cases} \quad \forall i \\
& {}^ix_{vu} \in \{0,1\}, \quad \forall u, v \in V,\ i = 1, \ldots, N_r. \\
& \sum_{i=1}^{N_r} z_{ij} = 1,\ \forall j; \quad \sum_{j=1}^{N_t} z_{ij} = 1,\ \forall i; \quad z_{ij} \in \{0,1\}\ \forall i, j
\end{aligned}
\tag{4.3}
$$

This problem is a non-linear integer program which is difficult to solve in general.

4.3 Geometric Analysis

Based on the insight of CC-LAP from Chapter 3, we now present a geometric analysis of CC-STAP. A feasible solution for (4.1) or (4.3) is a set of paths, one for each robot. For each feasible solution or set of paths, s, we can compute the mean $\mu(s)$ and variance $\sigma^2(s)$ of sum of the cost of all edges in the solution, namely, $\mu(s) = \sum_{i=1}^{N_r} \sum_{u,v=1}^{|V|} {}^i\mu_{uv}\, {}^ix_{uv}$ and $\sigma^2(s) = \sum_{i=1}^{N_r} \sum_{u,v=1}^{|V|} {}^i\sigma^2_{uv}\, {}^ix_{uv}$. Thus, a feasible solution, s, of (4.3), can be represented in the variance-mean plane, with coordinates $(\sigma^2(s), \mu(s))$. Figure 4.2 shows a schematic sketch of the variance-mean plane with the circles representing the feasible solutions. The level curve of the objective function in (4.3) is a parabola (Fig. 4.2 (a)). The objective value of a feasible solution, s, is equal to the vertical intercept of the parabola through s. *Therefore the optimal solution of the chance-constrained problem* (4.3) *is the point, s^*, with the smallest vertical intercept of the parabolic level curve through it.* Similar to CC-LAP, the optimal solution of 4.3 is an extreme point of the set of feasible solutions on variance-mean plane[1] . Therefore a risk-averse formulation in (4.4) with proper choice of the risk-averse

[1] The proof can be found in Lemma 1 of Chapter 3.

Figure 4.1: A motivation example of CC-STAP. Three autonomous cars are initially located at the places labeled by circles. Three passengers are calling for pick up. Due to the uncertain traffic condition, travel costs are random variables. The goal is to assign each autonomous car to passenger such that the actual travel cost of the team is less than a minimized cost value with a high probability.

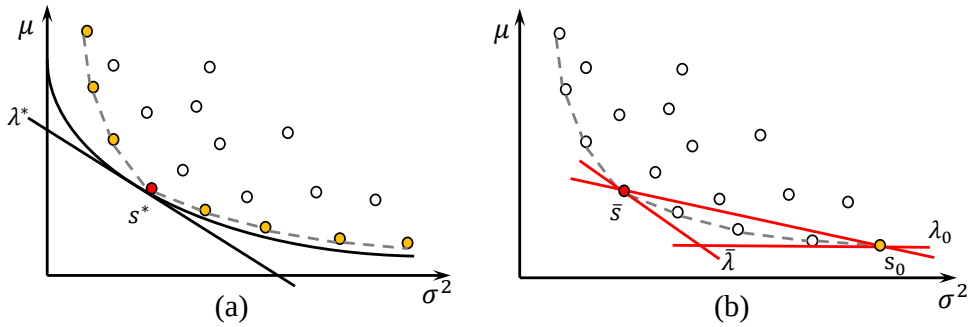


Figure 4.2: Geometric interpretation of optimization problems (4.3) and (4.4) on variance-mean plane. The level curve of objective function of (4.3) is a parabola (black solid line in (a)) and the level curve of objective function of (4.4) is a straight line with slope λ, the risk-aversion parameter. Circles denote the feasible solutions (not known *a priori*) and the dashed lines show the lower convex hull of the set of feasible solutions. (a) The optimal solution, s^*, is an extreme point of the set of feasible solutions. The parabola passing through s^* has the smallest vertical intercept and we can compute s^* by solving (4.4) with a proper risk-averse parameter, say, λ^* (negative of the slope of solid black line). (b) Our method computes the restricted search region (depicted by red triangle) that allows us to obtain s^* by solving a smaller number of risk-averse problems (4.4), rather than enumerating all extreme points.

parameter, λ is able to produce the optimal solution of (4.3).

$$
\begin{aligned}
&\min \quad \sum_{i=1}^{N_r} \sum_{u,v=1}^{|V|} ({}^{i}\mu_{uv} + \lambda {}^{i}\sigma_{uv}^{2})\ {}^{i}x_{uv} \\
&\text{s.t.} \quad \sum_{v \in \mathcal{N}(u)} {}^{i}x_{uv} - \sum_{v \in \mathcal{N}(u)} {}^{i}x_{vu} = \begin{cases} 1, & \text{if } u = s_{r_i}, \\ -z_{ij}, & \text{if } u = t_j, \\ 0, & \text{otherwise} \end{cases} \quad \forall i \\
&{}^{i}x_{vu} \in \{0,1\}, \quad \forall u, v \in V,\ i = 1, \ldots, N_r. \\
&\sum_{i=1}^{N_r} z_{ij} = 1,\ \forall j; \quad \sum_{j=1}^{N_t} z_{ij} = 1,\ \forall i; \quad z_{ij} \in \{0,1\}\ \forall i, j
\end{aligned}
\tag{4.4}
$$

This problem is a deterministic version of (4.1), where the edge costs are linear combination of means and variances i.e., ${}^{i}\mu_{uv} + \lambda {}^{i}\sigma_{uv}^{2}$. The parameter λ is the risk-averse parameter which balances between the mean and variance of edge cost. The higher the value of λ, the more risk-averse the robot team. In (4.4), we are simultaneously computing the assignment of robots to tasks and the path for robots to reach the tasks so as to minimize the total path cost of the robot team. The level curve of the objective function in (4.4) is a straight line in the variance-mean plane with a slope equal to $-\lambda$. The set of feasible solutions of (4.4) and (4.3) are the same, since the constraints are the same. Geometrically, for a given value of λ, solving (4.4) optimally is equivalent to finding a feasible point, s, such that the line with slope $-\lambda$ through s has the lowest vertical intercept. Hence, there is no other feasible point below the straight line. Therefore, *for a given λ, the optimal solution of* (4.4) *is an extreme point of the feasible point set.* The following lemma demonstrates the relationship between risk-averse problem (4.4) and chance-constrained problem (4.3).

Lemma 9. *The optimal solution, s^* of problem* (4.3) *is the optimal solution of* (4.4) *with proper value of λ.*

Proof. As shown in Fig. 4.2 (a), the optimal solution of (4.3) is the point, s^*, such that the parabolic level curve through it has the lowest vertical intercept. Therefore, there is no

feasible point below the level curve. There is always a straight line through s^* such that there is also no feasible point below it, e.g., the tangent to the parabola at s^*. Let λ^* be the negative of the slope of the tangent. Then, s^* must be the optimal solution of (4.4) with $\lambda = \lambda^*$. Therefore solving risk-averse problem (4.4) with $\lambda = \lambda^*$ will find the optimal solution of chance-constrained problem (4.3). □

Notice that Lemma 9 is proved by analyzing the geometric relationship between the level curve of the objective functions of (4.3) and (4.4), which is straightforward and intuitive. The formal mathematical proof is similar to the proof of Lemma 1 in Chapter 3. The difference is that in CC-LAP the optimal solution is the point that the parabolic level curve through it has the highest vertical intercept while in CC-STAP, the optimal solution is the point that the parabolic level curve through it has the lowest feasible point. Further, we consider extreme points as the vertices of the lower convex hull in CC-STAP instead of the upper convex hull. These differences can also be observed in Fig. 3.3 and 4.2.

4.4 Solution Approach

As discussed in Section 4.3, the optimal solution of CC-STAP is an extreme point of the feasible solution set and is the optimal solution of risk-averse problem (4.4) with a proper value of the risk-averse parameter, say λ^*. Therefore the problem becomes a one-dimensional search on risk-averse parameter λ. The idea of generating RA-LAP and finding extreme points on the variance-mean plane in Chapter 3 can be used for solving CC-STAP. We present a two-step algorithm to solve CC-STAP: (a) First, by computing an upper bound for λ^*, we find a search region (shown in Fig. 4.2 (b)) for the extreme points within which the optimal solution is guaranteed to lie. (b) We then enumerate extreme points in this search region by solving a sequence of risk-averse problems (4.4) with methodically generated λ.

Let μ_k and σ_k^2 denote the sum of the mean and variance of the edges on the path computed from risk-averse problem (4.4) with $\lambda = \lambda_k$, i.e., $\mu_k = \sum_{i=1}^{N_r} \sum_{u,v=1}^{|V|} {}^i\mu_{uv}\, {}^ix^*_{uv}$, $\sigma_k^2 = \sum_{i=1}^{N_r} \sum_{u,v=1}^{|V|} {}^i\sigma^2_{uv}\, {}^ix^*_{uv}$. Further let $\sigma_k = \sqrt{\sum_{i=1}^{N_r} \sum_{u,v=1}^{|V|} {}^i\sigma^2_{uv}\, {}^ix^*_{uv}}$. Therefore the optimal

solution of (4.4) is a point with coordinate (σ_k^2, μ_k) and the optimal objective function value for (4.4) is $\mu_k + \lambda_k \sigma_k^2$. The objective function value for (4.3) at this point is $\mu_k + C\sigma_k$.

We will present the lemma used to design the first step of our algorithm. Let $\lambda_k < \lambda_{k+1}$ be two different risk-averse parameters.

Lemma 10. *The optimal objective function value $\mu + \lambda\sigma^2$ for risk-averse problem (4.4) is a strictly monotonically increasing function of the risk-averse parameter λ.*

Proof. Let $\lambda_k < \lambda_{k+1}$ be two risk-averse parameters. It is obvious that

$$\mu_{k+1} + \lambda_k \sigma_{k+1}^2 < \mu_{k+1} + \lambda_{k+1}\sigma_{k+1}^2$$

Since $({\sigma_k}^2, \mu_k)$ is the optimal solution to risk-averse problem (4.4) with λ_k, we have

$$\mu_k + \lambda_k {\sigma_k}^2 \leq \mu_{k+1} + \lambda_k \sigma_{k+1}^2 \tag{4.5}$$

Combining two inequalities, we obtain the conclusion

$$\mu_k + \lambda_k \sigma_k^2 < \mu_{k+1} + \lambda_{k+1}\sigma_{k+1}^2$$

□

Lemma 11. *The variance of total travel cost of the optimal solution of the risk-averse problem (4.4), i.e., σ^2 is a non-increasing function of risk-averse parameter λ.*

Proof. Since $(\sigma_{k+1}^2, \mu_{k+1})$ is the optimal solution of the risk-averse problem (4.4) with λ_{k+1}, we have

$$\mu_k + \lambda_{k+1} \sigma_k^2 \geq \mu_{k+1} + \lambda_{k+1}\sigma_{k+1}^2 \tag{4.6}$$

By negating (4.5) and adding it to (4.6), we obtain $(\lambda_{k+1} - \lambda_k){\sigma_k}^2 \geq (\lambda_{k+1} - \lambda_k)\sigma_{k+1}^2$. Therefore ${\sigma_k}^2 \geq \sigma_{k+1}^2$. □

Lemma 12. *Let $\bar{\lambda}$ be a risk-averse parameter, such that the optimal objective function value of (4.4) with $\bar{\lambda}$ is equal to the objective value of (4.3) at the same solution, i.e., $\bar{\mu} + C\bar{\sigma} = \bar{\mu} + \bar{\lambda}\bar{\sigma}^2$. This objective value gives an upper bound for the optimal objective value of (4.3). The optimal risk-averse parameter λ^* must lie in the interval $[0, \bar{\lambda}]$.*

Proof. Let μ and σ^2 be obtained from the optimal solution of (4.4) with $\lambda > \bar{\lambda}$. We need to prove that the objective values of (4.3) at the solution obtained from the optimal solution of (4.4) with λ are greater than the objective value obtained at $\bar{\lambda}$, i.e., $\bar{\mu} + C\bar{\sigma} < \mu + C\sigma$ for any $\lambda > \bar{\lambda}$.

There are two different cases:

(a) $\lambda\sigma \leq C$. Obviously $\mu + C\sigma \geq \mu + \lambda\sigma^2$. And because $\lambda > \bar{\lambda}$, from Lemma 10 we know that $\mu + \lambda\sigma^2 > \bar{\mu} + \bar{\lambda}\bar{\sigma}^2 = \bar{\mu} + C\bar{\sigma}$. Therefore $\bar{\mu} + C\bar{\sigma} < \mu + C\sigma$.

(b) $\lambda\sigma > C$. Let $\lambda' = C/\sigma$. Thus, $\lambda' < \lambda$. Since $\lambda > \bar{\lambda}$, from Lemma 11 we know $\sigma < \bar{\sigma}$. Thus, $\frac{C}{\sigma} > \frac{C}{\bar{\sigma}}$, which implies $\lambda' > \bar{\lambda}$. Therefore, $\mu + C\sigma = \mu + \lambda'\sigma^2 \geq \mu' + \lambda'\sigma'^2 > \bar{\mu} + \bar{\lambda}\bar{\sigma}^2 = \bar{\mu} + C\bar{\sigma}$. Thus, $\bar{\mu} + C\bar{\sigma}$ is the upper bound of optimal objective value for (4.3). The team cost obtained by solving risk-averse problem with $\lambda > \bar{\lambda}$ is thus greater than or equal to $\bar{\mu} + C\bar{\sigma}$. Therefore we should focus the searching on the optimal risk-averse parameter λ^* within the interval $[0, \bar{\lambda}]$. □

Once $\bar{\lambda}$ is computed, we can obtain a triangular search region shown in Fig. 4.2 (b).

4.5 Algorithms

In this section, we present algorithms including three components: (a) an algorithm to find the upper bound of the optimal risk-averse parameter $\bar{\lambda}$ (Algorithm 5), (b) an algorithm to enumerate the extreme points in the search region and obtain the optimal solution of CC-STAP (Algorithm 6) and (c) an algorithm (Algorithm 7) to solve the risk-averse problem (4.4) and the distributed implementation of our algorithm.

Although the overall structure of the presented algorithms is similar to the algorithms of CC-LAP, several differences that should not be neglected. First, the risk-averse problem (4.4) in CC-STAP includes the path planning and the initial payoffs of the auction are not given. Second, the objective functions of both chance-constrained and risk-averse formulations are different and thus the update rule for the risk-averse parameter is different. Third, the distributed algorithms implement our complete two-step algorithm, which requires additional

variables stored in the robot's memory and additional communication message among robots. To have a self-contained chapter, we will provide a detailed description of the algorithms for CC-STAP.

4.5.1 Searching for the Bound

Algorithm 5 is the procedure for any robot r_i to compute $\bar{\lambda}$ in the first step of our algorithm. At any iteration k (line 1 or line 6), r_i solves risk-averse problem (4.4) with λ_k (initially $\lambda_0 = 0$). Let α_i be the index of the task assigned to r_i. We obtain the assigned task t_{α_i}, the path to the assigned task ${}^i\Pi_k = \{{}^i x^*_{uv}\}$, and the mean and variance of path cost, $\mu_k = \sum_{i=1}^{N_r} \sum_{u,v=1}^{|V|} {}^i\mu_{uv} {}^i x^*_{uv}, \sigma_k^2 = \sum_{i=1}^{N_r} \sum_{u,v=1}^{|V|} {}^i\sigma^2_{uv} {}^i x^*_{uv}$. Robot r_i store the solution s_k including the path to the assigned task ${}^i\Pi_k$ and the coordinates of the obtained solution, i.e., (σ_k^2, μ_k) in the solution set S and compute σ_k (line 2 or line 7). Then r_i determines whether λ_k is the upper bound $\bar{\lambda}$ by the condition $\lambda_k \sigma_k = C$ (line 3, from lemma 12). If no, the risk-averse parameter is updated by $\lambda_{k+1} = \frac{C}{\sigma_k}$ (line 4). Then r_i moves on to the next iteration (line 4-7). The procedure repeats until the condition is satisfied. Now λ_k is equal to upper bound $\bar{\lambda}$. Finally r_i outputs solution set S that will be used in the next step. For the proof of the convergence of Algorithm 5, interested readers are referred to Lemma 7 in Chapter 3.

Algorithm 5 Robot r_i searching for the upper bound $\bar{\lambda}$

Input: C, ${}^i\mu_{uv}$, ${}^i\sigma^2_{uv}$, $\forall e_{uv} \in E$.
Output: The solution set S.
1: Let $k = 0$, solve risk-averse problem with $\lambda_k = 0$.
2: Store the solution s_k in set S and compute σ_k from the output.
3: **while** $\lambda_k \sigma_k \neq C$ **do**
4: Update the risk-averse parameter by $\lambda_{k+1} = \frac{C}{\sigma_k}$.
5: $k = k + 1$.
6: Solve risk-averse problem with λ_k.
7: Store the solution s_k in set S and compute σ_k from the output.
8: **end while**
9: **return** Solution set S.

4.5.2 Enumerating Extreme Points within the Search Region

Algorithm 6 Enumerate the extreme points within the search region

Input: Solution set S.
Output: The optimal path ${}^{i}\Pi^{*}$ for r_i and the optimal team cost y^{*}.
1: Compute point pair set $S_m = \{(s_0, s_1), (s_1, s_2), ..., (s_{k-1}, s_k)\}$, where $m = 0$.
2: **while** S_m is not empty **do**
3: Create empty point pair set S_{m+1}.
4: **for** Each point pair in S_m, say (s_{j-1}, s_j) **do**
5: Compute risk-averse parameter $\lambda = -\frac{\mu_j - \mu_{j-1}}{\sigma_j^2 - \sigma_{j-1}^2}$.
6: Solve risk-averse problem (4.4) with λ.
7: **if** the obtained point say $(\sigma^2, \mu) \notin \{(\sigma_{j-1}^2, \mu_{j-1}), (\sigma_j^2, \mu_j)\}$ **then**
8: Store solution s in S and store point pairs (s_{j-1}, s), (s, s_j) in S_{m+1}.
9: **end if**
10: **end for**
11: $m = m + 1$.
12: **end while**
13: $s^{*} = \arg\max_{s_j \in S} \ \mu_j + C\sigma_j$, $y^{*} = \max_{s_j \in S} \ \mu_j + C\sigma_j$.
14: **return** ${}^{i}\Pi^{*}, y^{*}$.

In this subsection, we provide the algorithm for any robot r_i to enumerate the extreme points in the search region and compute the optimal solution of CC-STAP.

Given the solution set S obtained from Algorithm 5, r_i computes a set of point pairs, as $S_0 = \{(s_0, s_1), (s_1, s_2), ...\}$ (line 1). For each pair of solutions, say (s_{j-1}, s_j), robot r_i computes the slope of the line connecting both points and let a risk-averse parameter λ equal to the negation, i.e., $\lambda = -\frac{\mu_j - \mu_{j-1}}{\sigma_j^2 - \sigma_{j-1}^2}$ (line 4-5). Then r_i solves risk-averse problem with λ by Algorithm 7 (line 6). If the output solution s has different coordinates with s_{j-1} and s_j, then s is a new extreme point and r_i stores two pairs of points (s_{j-1}, s) and (s, s_j) in S_{m+1} that are used for the next iteration (line 7-9). When all pairs of points in the current iteration are processed (line 11), r_i moves on to the next iteration and computes λ by same procedure for each pair of points generated from last iteration (line 3-11). The procedure continues until no new extreme point is obtained. The optimal solution is the one with smallest objective value of problem (4.3). The optimal path Π_i^{*} can be obtained from the optimal solution s^{*}.

4.5.3 Solving Risk-averse Problem and the Distributed Implementation

The risk-averse problem in (4.4) is a deterministic STAP where the edge costs are linear combination of means and variance, i.e., ${}^i\mu_{uv} + \lambda {}^i\sigma^2_{uv}$. The following lemma shows the idea of our algorithm.

Lemma 13. *The risk-averse problem in* (4.4) *can be solved optimally by solving a linear assignment problem with assignment cost equal to the shortest path cost to the task in the graph with deterministic edge cost* ${}^i\mu_{uv} + \lambda\ {}^i\sigma^2_{uv}$.

Proof. Our first claim is that, in the optimal solution, each path should be the shortest path from the robot to the task. This is because, if any path is not the shortest path, we can always obtain a better solution using the shortest path for the same robot and task. Therefore we can use the shortest path costs for all robot-task pairs as the assignment costs. Further, the optimal assignment of robots to tasks should provide the minimum total path cost and satisfy the assignment constraints for z_{ij} in (4.4) , which are the constraints for the linear assignment problem. Therefore the optimal assignment can be computed by solving such linear assignment problem. □

Algorithm 7 is the auction algorithm without shared memory for risk-averse problem. The original auction algorithm was presented in [12]. We modify the auction algorithm to enable each robot to obtain the mean and variance of the path cost of the robot team so that each robot can update the risk-averse parameter locally. It is executed by each robot r_i when r_i solves risk-averse problem with λ_k in any iteration of Algorithm 5 or 6. First, robot r_i computes the shortest path and path cost to all tasks, denoted by $\{\pi_{ij}, \ell_{ij}\}_{j=1}^{N_t}$, by Dijkstra algorithm (line 1). The edge cost of the graph is ${}^i\mu_{uv} + \lambda_k {}^i\sigma^2_{uv}$. The assignment cost in the auction algorithm is $-\ell_{ij}$ (because of minimization problem). Robot r_i should also compute the mean and variance of the obtained paths from its initial position s_{r_i} to all tasks on the graph with the stochastic edge cost in (4.1), i.e., $\{\tau_{ij}, v_{ij}\}_{j=1}^{N_t}$ where $\tau_{ij} = \sum_{u,v=1}^{|V|} {}^i\mu_{uv} {}^ix^*_{uv}$, $v_{ij} = \sum_{u,v=1}^{|V|} {}^i\sigma^2_{uv} {}^ix^*_{uv}$ and $\{{}^ix^*_{uv}\}$ is the binary decision variables for r_i's shortest path to t_j, namely π_{ij} obtained from Dijkstra algorithm.

Algorithm 7 Auction algorithm for robot r_i solving risk-averse problem

Input: Risk-averse parameter λ_k.

Output: Path to the assigned task ${}^i\Pi_k$, mean μ_k and variance of the team cost. σ_k^2

Solve the shortest path problem from the initial position s_{r_i} to all tasks on the graph with edge cost ${}^i\mu_{uv} + \lambda_k \, {}^i\sigma_{uv}^2$. The solution includes $\{\pi_{ij}, \ell_{ij}, v_{ij}, \tau_{ij}\}_{j=1}^{N_t}$.

Compute initial assigned task by $\alpha_i = \arg\max_{1 \le j \le N_t} -\ell_{ij} - p_{ij}$, mean and variance of path cost to assigned task by $\psi_{i\alpha_i} = \tau_{i\alpha_i}$ and $\beta_{i\alpha_i} = v_{i\alpha_i}$.

while $\{p_{ij}\}_{j=1}^{N_t}$ changes in the last Δ auction iterations **do**

    Extract message $\{p_{hj}, b_{hj}, \psi_{hj}, \beta_{hj}\}_{j=1}^{N_t}$ from neighbors, i.e., $\forall h \in \mathcal{N}_i$.

    for each task t_j **do**

        Find the neighbor r_d offering the highest price for t_j, where $d = \arg\max_{h \in D} b_{hj}$, and $D = \arg\max_{h \in \{i, \mathcal{N}_i\}} p_{hj}$.

        Update variables $\{p_{ij}, b_{ij}, \psi_{ij}, \beta_{ij}\}$ by $p_{ij} = p_{dj}$, $b_{ij} = b_{dj}$, $\psi_{ij} = \psi_{dj}$, $\beta_{ij} = \beta_{dj}$.

    end for

    if $p_{i\alpha_i}$ increases or unchanged but $b_{i\alpha_i} \neq i$ **then**

        Update the assigned task t_{α_i} by $\alpha_i = \arg\max_{1 \le j \le N_t} -\ell_{ij} - p_{ij}$.

        Increase the price for the assigned task $p_{i\alpha_i}$ by γ_i.

        Update $b_{i\alpha_i} = i$, $\psi_{i\alpha_i} = \tau_{i\alpha_i}$ and $\beta_{i\alpha_i} = v_{i\alpha_i}$.

    end if

    Send local variables $\{p_{ij}, b_{ij}, \psi_{ij}, \beta_{ij}\}_{j=1}^{N_t}$ to neighbors.

end while

return α_i, ${}^i\Pi_k = \pi_{i\alpha_i}$, $\mu_k = \sum_{j=1}^{N_t} \psi_{ij}$ and $\sigma_k^2 = \sum_{j=1}^{N_t} \beta_{ij}$.

The input of our auction procedures for each robot r_i is $\{\ell_{ij}, \pi_{ij}, \tau_{ij}, v_{ij}\}_{j=1}^{N_t}$. The messages communicated among robots within the range are $\{p_{ij}, b_{ij}, \psi_{ij}, \beta_{ij}\}_{j=1}^{N_t}$ where p_{ij} is the price that r_i has to pay in order to be assigned to t_j and the initial price p_{ij} could be 0.[2] The variable b_{ij} is the local knowledge of index of the highest bidder for a task t_j (the purpose of b_{ij} is to break the tie in the bidding). The variables ψ_{ij} and β_{ij} are the additional messages used in our auction iteration, which indicates the local knowledge of the mean and variance of the path to task t_j in current auction iteration. The initial assignment t_{α_i} for r_i is the task with the highest net value (line 2). The initial value of variables $\psi_{i\alpha_i}$ and $\beta_{i\alpha_i}$ are equal to $\tau_{i\alpha_i}$ and $v_{i\alpha_i}$ respectively.

A single auction iteration of our algorithm is from line 4 to 14 in Algorithm 7. For each task t_j, robot r_i determines the robot offering the highest price from its neighborhood, denoted by $r_d \in \mathcal{N}_i$, based on the message obtained from $\mathcal{N}_i$ (line 6). If there are multiple highest bidders, robot with greatest b_{hj} ($h \in \mathcal{N}_i$) is identified as r_d. Next r_i updates the local variables by copying the message from r_d (line 7). After updating local variables for all tasks, r_i determines whether the updated price of the current assigned task $p_{i\alpha_i}$ increases or the price does not change but the highest bidder of the task $b_{i\alpha_i}$ changes (line 9). If yes, r_i should re-compute the assigned task t_{α_i} with the highest net value, i.e., $-\ell_{ij} - p_{ij}$ based on the updated prices (line 10). Let q_i and w_i denote the highest and second highest net value for r_i. Then the price of the updated assigned task t_{α_i} is increased by $\gamma_i = q_i - w_i + \epsilon$ where ϵ is to prevent the cycle in the auction (line 11). Now in its neighborhood, r_i provides the highest price for the assigned task t_{α_i}. Therefore the highest bidder $b_{i\alpha_i}$, the mean $\psi_{i\alpha_i}$ and variance $\beta_{i\alpha_i}$ of the assigned task are updated to i, $\tau_{i\alpha_i}$ and $v_{i\alpha_i}$ respectively (line 12). Then r_i sends updated local variables to its neighbors (line 14). The auction iteration continues until the local prices does not change for Δ iterations where $\Delta \leq n-1$ is the maximum diameter of the network (line 15). The local knowledge of the task prices, mean and variance of path cost to tasks $\{p_{ij}, \psi_{ij}, \beta_{ij}\}_{j=1}^{N_t}$ is now identical with the global information. Each robot r_i outputs the its assigned task t_{α_i}, the path to the assigned task denoted by ${}^i\Pi_k$, the mean μ_k

[2] It may improve the efficiency to use the prices vector at the end of Algorithm 7 in the last call.

and variance σ_k^2 of total path cost (line 16).

Similar to the proposed distributed algorithm for CC-LAP in Chapter 3, we could design a simplified version of distributed algorithm for CC-STAP. In particular, each robot r_i only implements Algorithm 5 and outputs the solution of risk-averse problem with $\bar{\lambda}$. Therefore r_i does not need to store all of the obtained solutions s_k. The variance of the team cost σ_k^2 is the only variable required to update the risk-averse parameter λ_k. Consequently, variables for the mean of path cost, namely τ_{ij} and ψ_{ij}, are not required in Algorithm 7. In Section 4.6, we show through the extensive simulations that the solution of the simplified version of the distributed algorithm is very close to the optimal solution (less than 5% relative difference).

4.6 Simulation Results

The computational cost for each robot is $K(T_1 + I \cdot T_2)$ where K is the number of risk-averse problems (4.4) solved, $T_1 = O(|E| + |V| \log |V|)$ is the computational cost for Dijkstra algorithm, $I = O(\Delta N_r^2 \lceil \frac{\max_{i,j} \ell_{ij} - \min_{ij} \ell_{ij}}{\epsilon} \rceil)$ is the number of auction iterations [12] and T_2 is the computational cost for single auction iteration. The value for K depends on the number of extreme points on the variance-mean plane. However, this number is problem parameter dependent and it is hard to give *a priori* bounds. In this section, we study the values for K and KI with different number of robots and the size of graphs. We show through extensive simulations that: (a) our algorithm is scalable with the number of robots (tasks) and the size of the maps. (b) Our algorithm is more efficient than enumerating all extreme points to obtain the optimal solution. (c) Our distributed algorithm is efficient and the solution is nearly optimal.

The simulations were done on a computer with Intel i7 2.60GHZ CPU and 16GB RAM. We assume the number of robots is same with the number of tasks. The desired probability p in chance constraint is 99% and ϵ of auction iteration in Algorithm 7 is 10. We create different instances with randomly generated means and variances for edge cost, i.e. ${}^i\mu_{uv}, {}^i\sigma_{uv}^2$.

The means are generated from a continuous uniform distribution $\mathcal{U}(20, 100)$ and variances are generated from a continuous uniform distribution with different magnitude of the range so that the edge with a higher mean also tends to have a higher variance.

We compare three algorithms (shown in plot (a) of Fig. 4.3 and 4.4) : (1) Distributed algorithm presented in this paper that implements the first step of our method. It produces an approximate solution. The results are represented by the red line. (2) The centralized algorithm that implements our two-step method. It solves CC-STAP optimally. The results are represented by the blue line. (3) The method that enumerates all extreme points. The solution is optimal. The results are shown by the black line. All methods solve CC-STAP by solving a sequence of risk-averse problems. We evaluate the performance by computing the number of risk-averse problems solved, namely K the number of iterations. We further evaluate the performance of our distributed algorithm by computing the number of auction iterations taken by each robot for solving CC-STAP, i.e., KI.

4.6.1 Scalability with the Number of Robots

We study the scalability of our algorithm with the number of robots varying from 20 to 100 with an increment of 20. Robots are working on a graph with 500 nodes and 8470 edges. The results are provided in Fig. 4.3. For a certain number of robots (x coordinate) in plot (a), we compute the average number of iterations taken by three methods from 100 instances with randomly generated ${}^i\mu_{uv}, {}^i\sigma^2_{uv}$. The black line has the highest value and growth rate. The value for our two-step algorithm increases slowly while the value for our distributed algorithm is nearly constant (less than 4). As shown in Fig. 4.5(a), the average relative difference[3] of the distributed algorithm is less than 3% (could reduce to order of 1×10^{-4} with smaller ϵ at the cost of a higher number of auction iterations). In plot (b), we further present the average (red) and maximum (blue) number of auction iterations taken by each robot over 100 instances. The average number grows linearly.

[3] The relative difference is defined as $\frac{V^*-V}{V^*}$ where V is the evaluated objective function value and V^* is the optimal objective function value.

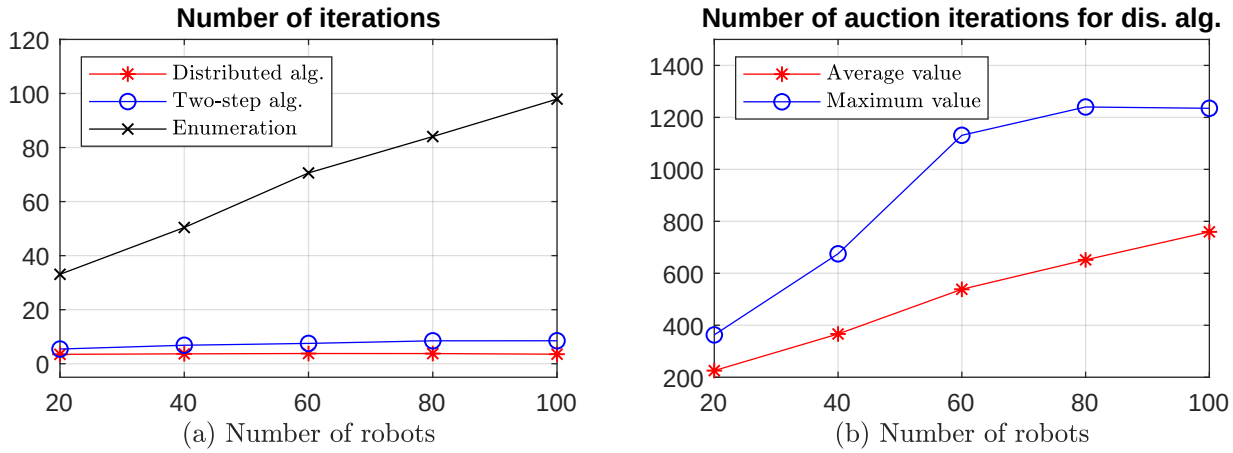


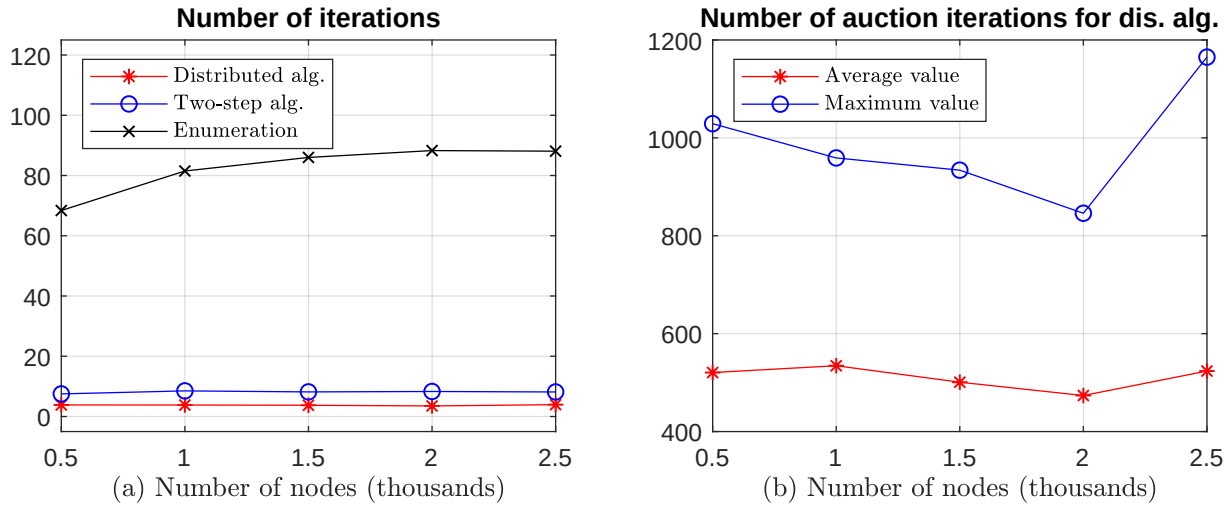
Figure 4.3: Scalability with the number of robots. (a) The average number of risk-averse problems solved and (b) the total number of auction iterations for each robot solving CC-STAP with different number of robots.

It implies that the search region (see Fig. 4.2 (b)) obtained from Algorithm 3 is small and excludes a large percentage of extreme points. Our algorithms solve less number of risk-averse problems than enumerating all extreme points and is scalable with the number of robots. Further, our distributed algorithm produces a good approximate solution with a small number of iterations.

4.6.2 Scalability with the Size of the Map

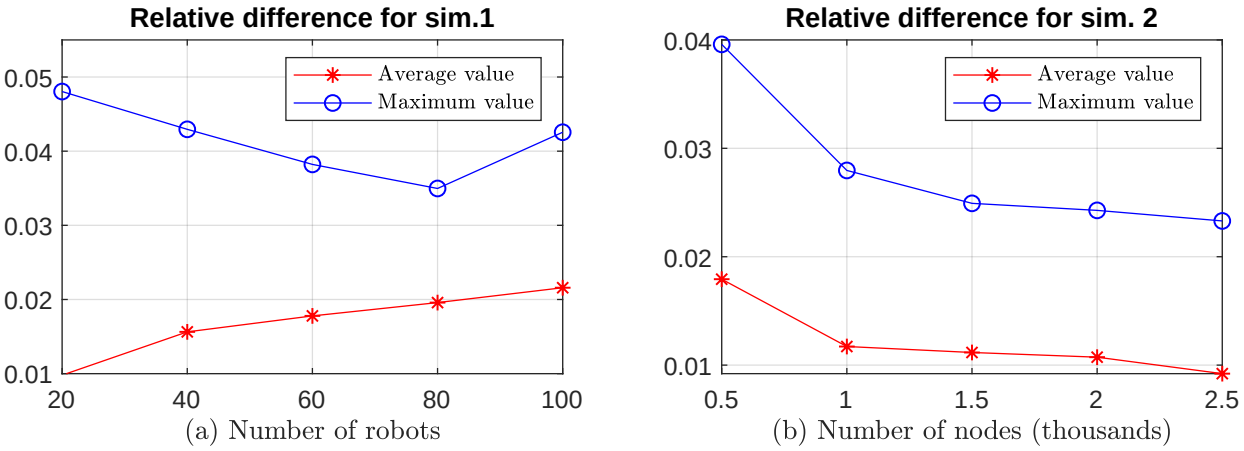
We also study the scalability of our algorithms with the number of nodes in the graph varying from 500 to 2500 with 500 increment. The number of edges increases from about 8000 to 27000 accordingly. The number of robots is 60. The results are presented in Fig. 4.4.

For a certain number of nodes in plot (a), we compute the average number over 100 instances under different graphs having the same number of nodes and randomly generated ${}^i\mu_{uv}, {}^i\sigma^2_{uv}$. The values for our distributed algorithm and two-step algorithm are nearly constant while the number for the black line grows slowly. As shown in Fig. 4.5(b), the average relative difference for the distributed algorithm is less than 2%. In plot (b) we present both average (red) and maximum (blue) number of the auction iterations for each robot. The results show that the number of nodes does not have a great influence on the number of



(a) Number of nodes (thousands) (b) Number of nodes (thousands)

Figure 4.4: Scalability with the size of the graph. (a) The average number of risk-averse problems solved and (b) the total number of auction iterations for each robot solving CC-STAP with different number of nodes in the graph.



(a) Number of robots (b) Number of nodes (thousands)

Figure 4.5: The relative difference between the objective values of the optimal solution and the approximate solution: (a) first set of simulations; (b) second set of simulations.

iterations and auction iterations. It influences more on the complexity of Dijkstra solving shortest path problems than the efficiency of our way of generating risk-averse problems.

4.7 Conclusion

We presented a novel algorithm for solving chance-constrained simultaneous task assignment and path planning on a graph (or roadmap) with stochastic edge costs. We proved that CC-STAP can be solved optimally by solving a sequence of deterministic STAP. We proved that the deterministic STAP can be solved optimally by a linear assignment problem with costs equal to the shortest path to the task location. We also present a distributed algorithm to solve CC-STAP based on the auction algorithm. The simulation results show that both our distributed algorithm and centralized two-step algorithm is scalable with the number of robots and the size of the graph.

Chapter 5

Chance-constrained Multi-robot Simultaneous Task Assignment and Path Planning with the MinMax Objective

In Chapter 4, we discuss the chance-constrained multi-robot simultaneous task assignment and path planning problem with the sum objective. In CC-STAP, team performance is the sum of the individual robot costs. However, in some multi-robot application scenarios, the team performance is considered as the maximum of the individual robot costs (makespan of team cost). In this chapter, we will discuss the chance-constrained multi-robot simultaneous task assignment and path planning problem with the MinMax objective (CC-STAP-B where B is for the bottleneck). In particular, CC-STAP-B is applicable to two common scenarios (a) when the robot team should complete their tasks in a minimum time window; (b) when the energy usage in completing the task should be as balanced as possible or the maximum energy consumption by any robot should be as small as possible. Similar to CC-STAP, we want to assign a unique task to each robot and compute a path for the robot going to the task location, given the means and variances of travel cost. However, the constraint of CC-STAP-B is that the path cost for *each* robot is below a minimum cost value in any realization of the stochastic travel costs with high probability. We formulate the problem as a combinatorial optimization problem with multiple chance constraints. We prove that the solution can be obtained by solving chance-constrained shortest path problems (CC-SP) for all robot-task pairs and a linear bottleneck assignment problem in which the cost of an assignment is equal to the optimal objective value of the former problem. We provide a two-step deterministic approach to solve CC-STAP. Leveraging methods solving CC-LAP in Chapter 3 and CC-STAP 4, we design algorithms to solve the sub-problem, CC-SP.

5.1 Introduction

As discussed in Chapter 4, the commonly used idea of decoupling the task allocation and path planning problems can lead to sub-optimal solutions. Furthermore, stochasticity of the path cost is usually inevitable except in very controlled situations. In many applications, the team performance is evaluated by the maximum of the individual robot costs. Therefore, in this chapter, we consider multi-robot coordination problems where the robots have to simultaneously plan paths and select target destinations (or tasks) under uncertainty about the travel costs with the MinMax objective. 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. If we are interested in balancing the energy consumed, then y will give the minimum value such that all robots will consume less energy than y with high probability.

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 combi-

natorial optimization problems [9, 10] and present a novel algorithm for solving the CC-SP problem, which is faster than existing algorithms [13, 14]. We present simulation results demonstrating the scalability of our algorithm to an increasing number of robots and tasks.

5.2 Mathematical Preliminaries

In this section, we introduce the chance-constrained shortest path (CC-SP) problem, which is a key sub-problem that arises while solving the CC-STAP problem.

5.2.1 Shortest Path Problem

We assume that there is a roadmap representing the free configuration space of the robots (see Fig. 1.3). 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. Note that the nodes and edges are collision-free 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 path with the least cost between s and d, is called the shortest path between them. 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 equal to 1 when edge (u, v) is included in the path and 0 otherwise. Given a source ($s \in V$) and destination ($d \in V$), the shortest path problem is:

$$
\begin{aligned}
\min \quad & \sum_{(u,v)\in E} c_{uv} x_{uv} \\
\text{s.t.} \quad & \sum_{v\in\mathcal{N}(u)} x_{uv} - \sum_{v\in\mathcal{N}(u)} x_{vu} = \begin{cases} 1, & \text{if } u = s, \\ -1, & \text{if } u = d, \\ 0 & \forall u \in V \setminus \{s, d\} \end{cases} \\
& x_{uv} \in \{0, 1\}, \quad \forall u, v \in V
\end{aligned}
\tag{5.1}
$$

where the objective is the cost of the path. The first constraint ensures that the selected edges form a path. For the source node, s, there is only one leaving edge, and for the

destination, d, there is one entering edge. Other nodes are either not on the path or there is an edge entering and an edge leaving. There are many algorithms to solve the shortest path in a graph [76, 77].

5.2.2 Chance-constrained Shortest Path Problem

We assume that the cost of an edge is a random variable. Hence the path cost is also a random variable. Since the edge costs and hence path costs are stochastic, the actual (realized) cost of a path can vary and hence the notion of the shortest path becomes unclear. *We define the stochastic shortest path as a path with the minimum value of y, such that the realized path cost is always less than y with very high probability (that we can set a priori).* Note that y is not the actual path cost, but the minimum value such that with very high probability, the total cost of the path is less than y under any realization of the random edge costs. To distinguish y from the actual path cost, we name y as the *CC path cost* and we also name the minimum y as *CC shortest path cost*. The optimal path with the CC shortest path cost of CC-SP (5.2) is named as *CC shortest path*. Let x_{uv} be a binary decision variable which is equal to 1 when edge (u, v) is included in the path and 0 otherwise. The CC-SP problem can be written as:

$$
\begin{aligned}
&\min \quad y \\
&\text{s.t.} \quad \mathbb{P}\left(\sum_{(u,v)\in E} c_{uv}x_{uv} \le y\right) \ge p, \\
&\qquad \sum_{v\in\mathcal{N}(u)} x_{uv} - \sum_{v\in\mathcal{N}(u)} x_{vu} = \begin{cases} 1, & \text{if } u = s, \\ -1, & \text{if } u = d, \\ 0 & \forall u \in V \setminus \{s, d\} \end{cases} \\
&\qquad x_{uv} \in \{0, 1\}, \quad \forall u, v \in V
\end{aligned}
\tag{5.2}
$$

The first constraint states that the total edge cost in any realization denoted as $\sum_{u,v} c_{uv}x_{uv}$, is no greater than y with at least a pre-specified probability p. The objective is to find such path with the smallest bound y. The second set of constraints is a path constraint that

guarantees that the solution is a sequence of edges starting from node s and ending at a destination d (i.e., task t).

5.3 Problem Formulation

We now present the multi-robot simultaneous task assignment and planning problem formally. We are given a graph, $G = (V, E)$, a set of N_r heterogeneous robots, $\{r_i\}_{i=1}^{N_r}$ and a set of N_t tasks or target destinations, $\{t_j\}_{j=1}^{N_t}$. Without loss of generality, we assume the number of robots is equal to the number of tasks, i.e., $N_r = N_t = n$. Each robot is initially at a given position in the graph denoted by s_{r_i} and each task (unassigned initially), denoted by t_j, is located at given position. Let z_{ij} be a binary decision variable for the assignment which is equal to 1 when task t_j is assigned to robot r_i and 0 otherwise. Let ${}^i x_{uv}$ be the binary decision variable for the edge of the path which is equal to 1 if edge (u, v) is in the path of r_i, and 0 otherwise. Let $V' = V \setminus \{s_{r_i}, t_j\}$ and $\mathcal{N}(u)$ be the neighbors of a node u.

Our goal is to assign each robot a unique task and compute a path for each robot. The CC-STAP-B problem is

$$
\begin{aligned}
&\min \quad y \\
&\text{s.t.} \;\; \mathbb{P}\left(\sum_{(u,v)\in E} {}^i c_{uv}\, {}^i x_{uv} \leq y \right) \geq p, \quad i = 1, \ldots, n. \\
&\sum_{v\in\mathcal{N}(u)} {}^i x_{uv} - \sum_{v\in\mathcal{N}(u)} {}^i x_{vu} = \begin{cases} 1, & \text{if } u = s_{r_i}, \\ -z_{ij}, & \text{if } u = t_j, \\ 0, & \forall u \in V', \end{cases} \quad \forall i \\
&\sum_{i=1}^{n} z_{ij} = 1, \;\; \forall j; \quad \sum_{j=1}^{n} z_{ij} = 1, \;\; \forall i; \quad z_{ij} \in \{0, 1\} \\
&{}^i x_{uv} \in \{0, 1\}.
\end{aligned}
\tag{5.3}
$$

A feasible solution for (5.3) 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 (5.3)); (b) the path 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 (5.3)).

The second set of constraints contain one unspecified path constraint for each robot. These constraints are slightly different from the path constraints in the shortest path problem (5.1) and (5.2) because the assignment of the source and target is not specified *a priori* and binary decision variable z_{ij} is thus included in the path constraints. 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_{r_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 are the assignment constraints that each robot performs a single task and each task can be assigned to only one robot.

Note that the *technical challenge* in solving CC-STAP-B (5.3) directly is that (a) there are n chance constraints which bring the nonlinearity to the integer optimization problem; (b) the task allocation and path planning are coupled and the existing path planning and task assignment algorithms are not able to solve both problems simultaneously and obtain the optimal solution.

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., $\sum_{u,v} {}^i c_{uv} {}^i x_{uv} \sim \mathcal{N}(\sum_{u,v} {}^i \mu_{uv} {}^i x_{uv}, \sum_{u,v} {}^i \sigma^2_{uv} {}^i x_{uv})$. As discussed in Chapter 3, the chance constraint for any robot r_i is equivalently written as

$$\sum_{(u,v)\in E} {}^i \mu_{uv} {}^i x_{uv} + C \sqrt{\sum_{(u,v)\in E} {}^i \sigma^2_{uv} {}^i x_{uv}} \leq y \tag{5.4}$$

If the distribution of the edge cost is not Gaussian distribution, we could obtain an upper bound for the optimal objective value by using a conservative value for $C = \sqrt{\frac{p}{1-p}}$ which comes from Chebyshev's inequality. In fact, for scenarios where only means and variance of the edge costs are known and there is no information about the distributions, the best one can do is to upper-bound the probability tail in this value-at-risk model [13].

The CC-STAP-B formulation (5.3) is equivalent to (5.5)

$$
\begin{aligned}
&\min\ y \\
&\text{s.t.}\quad \sum_{u,v} {}^{i}\mu_{uv}\,{}^{i}x_{uv} + C\sqrt{\sum_{u,v} {}^{i}\sigma^2_{uv}\,{}^{i}x_{uv}} \le y,\ \forall i \\
&\qquad \sum_{v\in\mathcal{N}(u)} {}^{i}x_{uv} - \sum_{v\in\mathcal{N}(u)} {}^{i}x_{vu} = \begin{cases} 1, & \text{if } u = s_{r_i}, \\ -z_{ij}, & \text{if } u = t_j, \\ 0, & \forall u \in V', \end{cases} \ \forall i \\
&\qquad \sum_{i=1}^{n} z_{ij} = 1,\ \forall j;\quad \sum_{j=1}^{n} z_{ij} = 1,\ \forall i;\quad z_{ij} \in \{0,1\} \\
&\qquad {}^{i}x_{uv} \in \{0,1\}.
\end{aligned}
\tag{5.5}
$$

Let ${}^{i}y$ denote the CC path cost for robot r_i (the target is not specified). Thus $y = \max_i {}^{i}y$, which implies that the optimal solution to (5.3) can be obtained by solving

$$
\begin{aligned}
&\min \max_i\ {}^{i}y \\
&\text{s.t.}\quad \sum_{(u,v)\in E} {}^{i}\mu_{uv}\,{}^{i}x_{uv} + C\sqrt{\sum_{(u,v)\in E} {}^{i}\sigma^2_{uv}\,{}^{i}x_{uv}} \le {}^{i}y,\ \forall i \\
&\qquad \sum_{v\in\mathcal{N}(u)} {}^{i}x_{uv} - \sum_{v\in\mathcal{N}(u)} {}^{i}x_{vu} = \begin{cases} 1, & \text{if } u = s_{r_i}, \\ -z_{ij}, & \text{if } u = t_j, \\ 0, & \forall u \in V', \end{cases} \ \forall i \\
&\qquad \sum_{i=1}^{n} z_{ij} = 1,\ \forall j;\quad \sum_{j=1}^{n} z_{ij} = 1,\ \forall i;\quad z_{ij} \in \{0,1\} \\
&\qquad {}^{i}x_{uv} \in \{0,1\}.
\end{aligned}
\tag{5.6}
$$

Define the robot with the highest CC path cost as the *bottleneck robot*, i.e., r_i where $i = \arg\max_j {}^{j}y$. We provide formal proof in Lemma 14 and 15 to show that problem (5.5) is equivalent to (5.5).

Lemma 14. *Let r_i be the bottleneck robot and d_i be its assigned task location in the optimal solution to Problem* (5.6)*. 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 (5.2) *for robot r_i to destination d_i).*

Proof. To prove by contradiction, we assume that the path for the bottleneck robot r_i in the optimal solution s^* is not the CC shortest path. Let ${}^iy$ be the CC path cost for r_i in solution s^*. Since s^* is optimal, the CC path costs of the bottleneck robots in *all other feasible solutions* are greater than or equal to ${}^iy$.

There exists a solution denoted by $\hat{s}$ that can be obtained from s^* by replacing the path for bottleneck robot r_i with the CC shortest path to its assigned task location d_i. Let ${}^i\hat{y}$ be the CC shortest path cost for r_i going to its destination d_i. Now consider the bottleneck robot in $\hat{s}$. There will be two different cases.

(1) Bottleneck robot, r_i in the optimal solution s^* is no longer the bottleneck in solution $\hat{s}$. Instead other robot say r_j is the bottleneck robot in $\hat{s}$. Denote its CC path cost in $\hat{s}$ as ${}^j\hat{y}$, which is equal to its CC path cost in the optimal solution, denoted by ${}^jy$. Since the path for r_j in $\hat{s}$ is the same path in the optimal solution s^* in which r_i is the bottleneck, it implies that the CC path cost of the bottleneck robot in $\hat{s}$ is less than or equal to the CC path cost of the bottleneck robot in $\hat{s}$, i.e., ${}^iy \geq {}^jy = {}^i\hat{y}$. Therefore, the new solution $\hat{s}$ is better than the optimal solution s^*.

(2) Second, r_i is still the bottleneck in $\hat{s}$. Now the CC path cost of the bottleneck robot in $\hat{s}$ is less than or equal to the CC path cost of the bottleneck robot in the optimal solution s^*. It indicates that the solution $\hat{s}$ is better than the optimal solution s^*.

In both cases, we conclude that there is a solution that has objective value better than the optimal solution. This contradicts the statement that the paths of the bottleneck robot in *all other feasible solutions* are greater than or equal to ${}^iy$. Therefore, the path for r_i must be a CC shortest path in s^*. □

Lemma 14 implies the following key conclusion:

Lemma 15. *The optimal solution of problem* (5.6) *is also optimal for problem* (5.3).

Proof. The feasible solutions in problem (5.6) are always feasible to problem (5.3) in which $y = \max_i {}^i y$. Hence the optimal solution of problem (5.6) is feasible to (5.3). From lemma 14, the optimal solution of problem (5.3) can always be converted to a feasible solution to problem (5.6) with same objective value by replacing each path with CC shortest path with same starting point and destination, and ${}^i y$ is equal to its CC shortest path cost and $y = \max_i {}^i y$. Therefore the optimal solution of problem (5.6) is also optimal to problem (5.3). □

Lemma 15 above suggest the idea of our two-step method to solve (5.6) 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 path for a robot-task pair (r_i, t_j) is formulated as:

$$
\begin{aligned}
&\min {}^i y_j \\
&\text{s.t.} \quad \sum_{(u,v)\in E} {}^i\mu_{uv}\, {}^i x_{uv} + C\sqrt{\sum_{(u,v)\in E} {}^i\sigma^2_{uv}\, {}^i x_{uv}} \le {}^i y_j \\
&\qquad \sum_{v\in\mathcal{N}(u)} {}^i x_{uv} - \sum_{v\in\mathcal{N}(u)} {}^i x_{vu} = \begin{cases} 1, & \text{if } u = s_{r_i}, \\ -z_{ij}, & \text{if } u = t_j, \\ 0, & \forall u \in V' \end{cases} \\
&\qquad {}^i x_{uv} \in \{0, 1\}.
\end{aligned}
\tag{5.7}
$$

Problem (5.7) is the formulation for the CC-SP for r_i from s_{r_i} to t_j, therefore the decision variable of the assignment z_{ij} is implicitly equal to 1. This formulation is equivalent to the

CC-SP formulation in (5.2) with $s = s_{r_i}$ and $d = t_j$. Let ℓ_{ij} be the optimal objective value of the problem above, i.e., $\ell_{ij} = \min {}^{i}y_j$. Then the task assignment problem is:

$$\begin{aligned} &\min \max_{\mathrm{i}} \sum_{j=1}^{n} \ell_{ij} z_{ij} \\ &\text{s.t. } \sum_{i=1}^{n} z_{ij} = 1,\ \sum_{j=1}^{n} z_{ij} = 1,\ z_{ij} \in \{0,1\}\ \forall i,j. \end{aligned} \tag{5.8}$$

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 the proper task assignment to minimize the CC path cost of the bottleneck robot which is same with the objective in formulation (5.6). The decision variable for the task assignment z_{ij} has the same definition in formulation (5.6). It is guaranteed that each robot is assigned to a unique task which is equivalent to the assignment constraints in formulation (5.6). Further since ℓ_{ij} is obtained from problem in (5.7), a feasible solution in formulation (5.8) also satisfies the chance constraint and path constraints in formulation (5.6). Therefore the feasible solution of formulation (5.8) is feasible to formulation (5.6) and vice versa. This illustrates the following lemma:

Lemma 16. *The optimal solution of problem* (5.3) *can be obtained by solving problem* (5.7) *and* (5.8).

Proof. The feasible set defined by problem (5.7) and (5.8) is same with that of problem (5.6). By definition $\ell_{ij} = \min {}^{i}y_j$, the objective of problem (5.8) is thus same with that of (5.6), i.e., ${}^{i}y = \sum_j \ell_{ij} z_{ij}$. Therefore problem (5.6) is equivalent to problem (5.8) with ℓ_{ij} equal to the optimal objective value of problem (5.7). And by Lemma 15, the optimal solution of problem (5.8) is optimal to problem (5.3). □

5.4 Solution Approach

In this section, we will discuss the algorithm to solve CC-SP (5.9) and linear bottleneck assignment problem in (5.8).

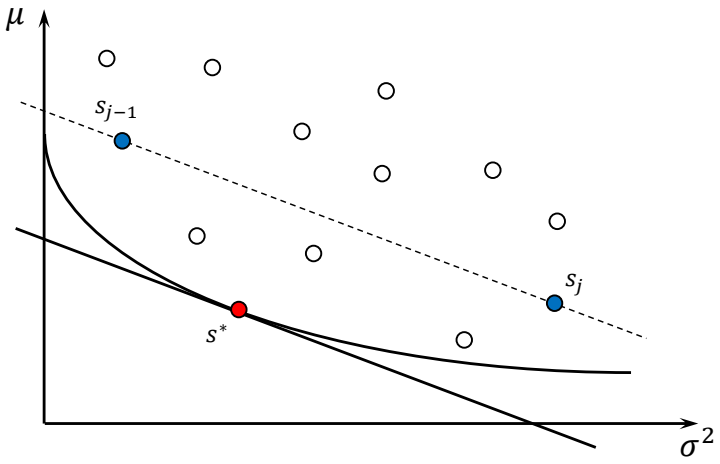


Figure 5.1: On the variance-mean plane, any path is mapped as a point with coordinates equal to the sum of variances and means of the edge cost in the path respectively. The objective function value in (5.9) of the path is equal to the vertical intercept of the parabola through the associated point. The optimal path is the point with the smallest vertical intercept of the parabola through it.

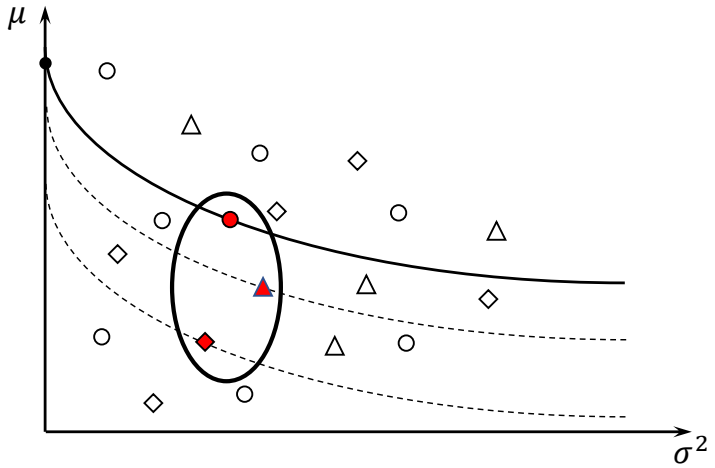


Figure 5.2: The feasible paths for one robot is represented by points with the same shape. The feasible solution for CC-STAP-B is a group of points representing the path for each robot. The red dot is the bottleneck robot with the highest level curve through it. The optimal solution is the set of points such that the level curve through the bottleneck robot has the smallest vertical intercept.

5.4.1 Chance-constrained Shortest Path Problem

The CC-SP problem in (5.7) can be written as an equivalent mean-risk model as follows:

$$\begin{aligned}
\min \quad & \sum_{(u,v)\in E} {}^{i}\mu_{uv}\,{}^{i}x_{uv} + C\sqrt{\sum_{(u,v)\in E} {}^{i}\sigma^2_{uv}\,{}^{i}x_{uv}} \\
\text{s.t.} \quad & \sum_{v\in\mathcal{N}(u)} {}^{i}x_{uv} - \sum_{v\in\mathcal{N}(u)} {}^{i}x_{vu} = \begin{cases} 1, & \text{if } u = s_{r_i}, \\ -1, & \text{if } u = t_j, \\ 0, & \forall u \in V' \end{cases} \\
& {}^{i}x_{uv} \in \{0,1\}.
\end{aligned} \tag{5.9}$$

This non-convex combinatorial optimization problem has a similar structure as CC-LAP formulation (3.5) in Chapter 3 and CC-STAP with sum objective formulation (4.3) in Chapter 4. Any feasible path can be mapped as a point on variance-mean plane (shown in Fig. 4.2) in which the horizontal and vertical coordinates are equal to the sum of the variances and means of the edges in the path, i.e., $\sigma^2 = \sum_{(u,v)\in E} {}^{i}\sigma^2_{uv}\,{}^{i}x_{uv}$, $\mu = \sum_{(u,v)\in E} {}^{i}\mu_{uv}\,{}^{i}x_{uv}$. The objective function value of any solution is thus equal to the vertical intercept of the parabolic level curve $\Pi = \{(\sigma^2,\mu)|\mu + C\sigma = m\}$. Hence the optimal solution of CC-SP (5.9), denoted by s^* is a point such that the level curve Π through it has the minimum vertical intercept. There always exists a straight line $\Pi_R = \{(\sigma^2,\mu)|\mu+\lambda\sigma^2 = m'\}$ through the optimal solution s^* such that all feasible points is above it. Therefore the optimal solution of CC-SP (5.9) is an extreme point on variance-mean plane which can be solved by solving the risk-averse problem (5.10).

$$\begin{aligned}
\min \quad & \sum_{(u,v)\in E} ({}^{i}\mu_{uv} + \lambda\,{}^{i}\sigma^2_{uv})\,{}^{i}x_{uv} \\
\text{s.t.} \quad & \sum_{v\in\mathcal{N}(u)} {}^{i}x_{uv} - \sum_{v\in\mathcal{N}(u)} {}^{i}x_{vu} = \begin{cases} 1, & \text{if } u = s_{r_i}, \\ -1, & \text{if } u = t_j, \\ 0, & \forall u \in V' \end{cases} \\
& {}^{i}x_{uv} \in \{0,1\}.
\end{aligned} \tag{5.10}$$

The formulation above is essentially a deterministic shortest path problem on the graph with edge cost equal to ${}^i\mu_{uv} + \lambda^i\sigma^2_{uv}$. It can be solved optimally by Dijkstra's algorithm with $O(|E| + |V| \log |V|)$ computational complexity.

Note that the objective function of CC-LAP (3.5), CC-STAP (4.3) and CC-SP (5.9) are all quadratic function of μ and σ^2. Although the constraints in CC-LAP, CC-STAP and CC-SP are different, it only influences the distribution of the feasible points on the variance-mean plane. Listed below are the important conclusions from CC-LAP and CC-STAP that are still valid for CC-SP.

(1) The optimal solution of CC-SP (5.9) can be obtained by solving risk-averse problem (5.10) with proper value of risk-averse parameter, denoted by λ^*. The proof can be found in Lemma 1 in Chapter 3 and Lemma 9 in Chapter 4.

(2) There is an upper bound for the optimal risk-averse parameter, denoted by $\bar{\lambda}$. The optimal objective function value of risk-averse problem (5.10) with $\bar{\lambda}$ is equal to the objective function value of CC-SP (5.9) at the same solution, i.e., $\bar{\mu} + C\bar{\sigma} = \bar{\mu} + \bar{\lambda}\bar{\sigma}^2$. The proof can be found in Lemma 6 in Chapter 3 and Lemma 12 in Chapter 4.

Therefore CC-SP (5.9) can be solved optimally by solving methodically generated risk-averse problem (5.10). Leveraging our algorithms for CC-LAP, we present a two-step algorithm to solve CC-SP. In the first step of our algorithm, we compute the upper bound for risk-averse parameter $\bar{\lambda}$. In the second step, we enumerate the possible solutions (extreme points on the variance-mean plane) in the search region and find the optimal solution.

The first step of our algorithm is presented in Algorithm 8. It starts with solving risk-averse problem (5.10) with λ_k equal to 0 (line 1-2). The obtained solution is stored in the solution set S. Further, the coordinate associated with this solution (σ^2_k, μ_k) on the variance-mean plane is obtained (line 3). Then the algorithm determines if $\lambda_k\sigma_k = C$. If no, the risk-averse parameter is updated, $\lambda_{k+1} = \frac{C}{\sigma_k}$ and the algorithm goes back to step 2 (line 4-5). If yes, the current λ_k is the upper bound for the risk-averse parameter. The

Algorithm 8 Find the upper bound for risk-averse paraemeter $\bar{\lambda}$

Input: $C, {}^{i}\mu_{uv}, {}^{i}\sigma_{uv}$.
Output: $\bar{\lambda}, S$.
1: Initialization: let $k = 0, \lambda_k = 0$.
2: Solve risk-averse problem (5.10) with λ_k.
3: Compute (σ_k^2, μ_k) and store it to solution set S.
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: **end if**
7: **return** $\bar{\lambda} = \lambda_k$ and S.

algorithm terminates in a finite number of iterations (the proofs are provided in Lemma 7). Note that the optimal solution of the risk-averse problem (5.10) with $\bar{\lambda}$ is a nearly optimal solution. We compare the objective value of the obtained solution and the optimal solution in Section 5.5.

The second step of our algorithm is presented in Algorithm 9. The algorithm enumerates all extreme points in the search region by solving the risk-averse problem (5.10) with $\lambda \in [0,\ \bar{\lambda}]$. Given the solution set S containing the extreme points obtained in Algorithm 8, we need to find the rest of the extreme points in the search region. Inspired by the quickhull algorithm for convex hull [78], we find extreme points below the straight line connecting any consecutive pair of solutions in S if it exists, e.g., (s_{j-1}, s_j) in Figure 5.1. 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), we can obtain the risk-averse parameter λ by computing the negation of slope of 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 5). The algorithm solve risk-averse problem with λ (line 6). 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 set S. Two new point pairs (s_{j-1}, s) and (s, s_j) are stored in S_{i+1} (line 7-8). If the obtained solution is same with s_{j-1} or s_j, there is no extreme below the line through s_{j-1} and s_j. After solving all risk-averse problems generated from S_i, algorithm determines whether the updated set S_i is empty. If it is not empty, in other words there

is new extreme point obtained from last iteration, we move on to next iteration and repeat the same procedures starting from line 3. It stops when S_i is empty, which means no new extreme point is obtained and we have found all extreme points (vertices of the lower convex hull) in the search region. The optimal solution can be obtained from the obtained solution set S (line 13).

Algorithm 9 Compute the optimal solution of CC-SP

Input: Solution set S.
Output: Optimal solution s^* and optimal objective value y^*.
1: Compute point pair set $S_i = \{(s_1, s_2), (s_2, s_3), ..., (s_{k-1}, s_k)\}$ where $i = 0$.
2: **while** S_i is not empty **do**
3: Create empty point pair set S_{i+1}.
4: **for** each point pair in S_i, say (s_{j-1}, s_j) **do**
5: Compute risk-averse parameter $\lambda = -\frac{\mu_j - \mu_{j-1}}{\sigma_j^2 - \sigma_{j-1}^2}$.
6: Solve risk-averse problem (5.10) with λ.
7: **if** the obtained solution say $s \notin \{s_{j-1}, s_j\}$ **then**
8: Store s in S and store point pairs (s_{j-1}, s), (s, s_j) in S_{i+1}.
9: **end if**
10: **end for**
11: $i = i + 1$.
12: **end while**
13: $s^* = \arg\min_{s_j \in S} \ \mu_j + C\sigma_j$, $y^* = \min_{s_j \in S} \ \mu_j + C\sigma_j$.
14: **return** s^*, y^*.

5.4.2 Algorithm for Linear Bottleneck Assignment Problem

Now consider the CC-STAP-B (5.6) on variance-mean plane shown in Fig. 5.2. As discussed in Section 5.4.1, the path for any robot is mapped as one point on the variance-mean plane. The CC shortest path for one robot is an extreme point, say (σ_i^2, μ_i) of the set of feasible paths for the *same source-target pair* (may not be the extreme point of the set of all feasible paths though). The CC shortest path cost is equal to the vertical intercept of a parabolic level curve $\Pi = \{(\sigma^2, \mu) | \mu + C\sigma = m\}$ where $m = \mu_i + C\sigma_i$. As shown in Fig. 5.2, the feasible solution is a group of points representing the path for each robot. The bottleneck robot (red dot) corresponds to the point with the highest level curve through any solution (in other words all points in the group are on one side of the level curve). The

optimal solution is the set of points such that the level curve through the bottleneck robot has the smallest vertical intercept.

Lemma 14 implies that the optimal solution of CC-STAP-B is or can be converted to a group of CC-SP. Solving CC-SP for all robot-task pairs provides us the candidate paths for the optimal solution. The remaining problem is to compute the assignment of the robot to the task (selects a subset of the candidate paths) such that the CC shortest path cost of the bottleneck robot is minimized. Lemma 16 states that given the CC shortest path for all robot-task pairs, i.e., $\ell_{ij}, \forall i, j$ as the assignment costs, the CC-STAP-B is a linear bottleneck assignment problem (5.8), which can be solved by many algorithms [28, 29, 79, 34]. In this chapter, we use a threshold algorithm presented in [29], which is presented in Algorithm 10 for completeness.

Algorithm 10 Threshold algorithm for solving LBAP

Input: Cost matrix $L = (\ell_{ij}) \in \mathbb{R}^{n \times n}$.

Output: The decision variables z_{ij} and the optimal bottleneck cost ℓ^* (minimum time window).

1: Initialization: Obtain the bipartite graph $G = (U, V, E)$ from L, let M indicates the maximum matching which is initialized as an empty set.
2: Find the initial threshold $\bar{\ell} = \max_k(\min_i \ell_{ik}, \min_i \ell_{ki})$.
3: Compute the cost matrix $L[\bar{\ell}]$ and the corresponding bipartite graph $G[\bar{\ell}]$.
4: Compute the maximum matching M and the minimum cover (I, J) of $G[\bar{\ell}]$.
5: **if** $|M| \neq n$ **then**
6: Update the threshold $\bar{\ell} = \min_{i \notin I, j \notin J} \ell_{ij}$ and go back to step 3.
7: **end if**
8: **return** The optimal time window ℓ^* and decision variables z_{ij}.

After solving CC-shortest path problem for all robot-task pairs, we can thus compute a cost matrix $L = (\ell_{ij}) \in \mathbb{R}^{n \times n}$ where ℓ_{ij} is the optimal objective value of CC-SP with source s_i and target t_j. Given the cost matrix L, we could build a bipartite graph $G = (U, V, E)$ (line 1). The first threshold $\bar{\ell}$ is initialized as the maximum of the row minima and column minima which is a lower bound of the optimal threshold (line 2). Then a threshold matrix $L[\bar{\ell}]$ is defined as $L[\bar{\ell}] = (\bar{\ell}_{ij}) \in \mathbb{R}^{n \times n}$ where $\bar{\ell}_{ij} = 1$ if $\ell_{ij} < \bar{\ell}$ and 0 otherwise. $G[\bar{\ell}]$ is the unweighted bipartite graph built from $L[\bar{\ell}]$ (line 3). Further, the algorithm computes the

unweighted maximum matching M and the minimum cover (I, J) where $I \subseteq U$, $J \subseteq V$ are the index set for the covered rows and covered columns respectively. Since in the final assignment every robot has a unique target, the number of edges in the matching should be equal to the number of the robot n. Therefore we check the cardinality of the matching $|M|$. If it is less than n, the threshold is updated and the algorithm repeats the procedure from step 3 (line 5-6). This is because the optimal objective value must be in the remaining uncovered entries and we have to find a new threshold from entries larger than the current threshold. The new bipartite graph $G[\bar{\ell}]$ has all edges in the previous graph and at least one new edge. The procedure continues until the cardinality of the matching is equal to n. In other words, the matching is now a perfect matching. The CC shortest path of the bottleneck robot (optimal objective value of CC-STAP-B) is equal to the maximum CC shortest path of the robot-task pair (edges) in the matching and for those pair included in the matching say (i, j) the decision variable z_{ij} is equal to 1.

5.5 Simulation Results

We motivate the application of our algorithms in a mobility-on-demand scenario. There are a team of autonomous robots located at places marked by circles and a number of passengers located at places marked by squares on the roadmap shown in Fig. 5.3. Each robot should be assigned to a passenger and compute a path to the location of the assigned passenger. Influenced by the uncertain traffic condition, the time for a robot moving to the assigned location is uncertain. The performance criterion is the makespan of the robot team. The goal is to have the minimum amount of time such that the robot team picks up all passengers with a high probabilistic guarantee.

The region in Fig. 5.3 is selected from OpenStreetMap. The OSM data of this region is converted to a graph data structure in Matlab. The means and variances of travel cost of edges are computed based on the distance and tags of the road from OSM file (e.g., edge with tag highway has smaller variance). The probability is defined as 90%. We compare

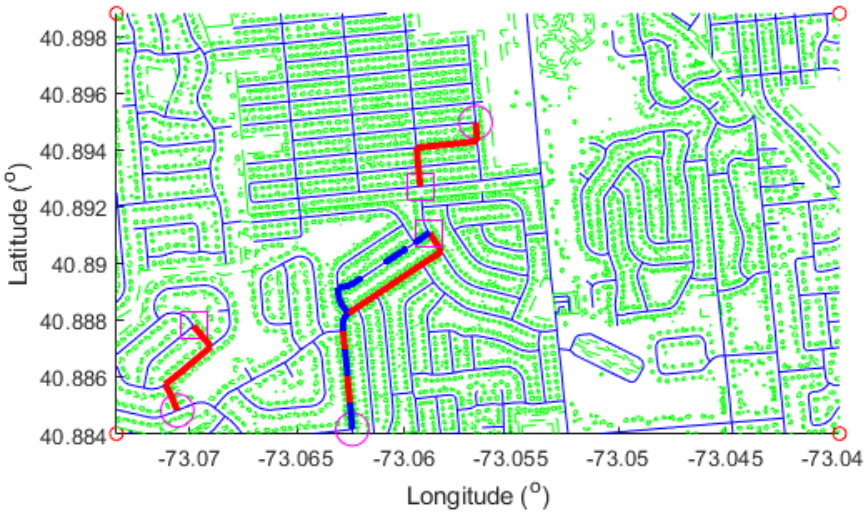


Figure 5.3: A number of passengers are initially located at several places in the road map marked by squares. A team of autonomous robots are initially located at places marked by circles. Each robot should be assigned to a passenger and compute a path to pick up the assigned passenger. The time for robots moving to the target location is uncertain. The goal is to have the minimum time window that the robot team picks up all passengers with a high probabilistic guarantee. The red lines indicate the paths obtained from our algorithm while the blue dashed line is the path of the bottleneck robot obtained by optimizing the expectation.

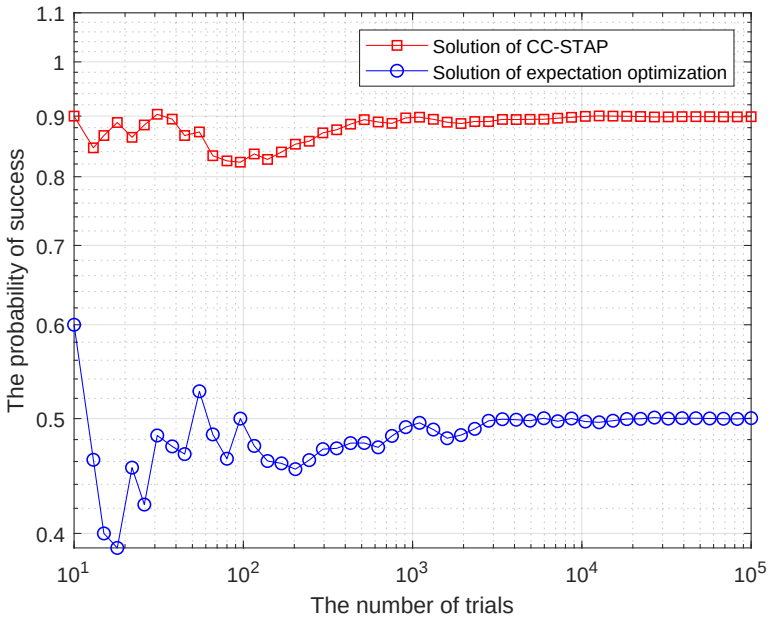


Figure 5.4: The probability that the travel times of the bottleneck robot do not exceed the optimal time window obtained by solving CC-STAP and optimizing the expected travel time.

our algorithm with a commonly used method optimizing the expectation. The solutions of our algorithm are shown as red paths in Fig. 5.3. Under the same setup, we also solve the problem by optimizing the expectation of the time window instead of using chance constraints. The path for the bottleneck robot (the blue dashed line) is different from our solution. We compute the probability that the travel time of each robot is within the time window obtained by both methods. The probabilities for the bottleneck robot (which takes the longest time than other robots) are shown in Fig. 5.4. The probability for our solution converges to 90% while the probability converges to 50% for the method optimizing the expectation. In other words, in any realization, the bottleneck robot may take a longer time than the optimal expected time window to pick up a passenger. The paths and time window obtained by optimizing the expectation might not be a robust solution to the application in which the safety or success rate is important. This is one of the advantages to use the chance constraint formulation.

Besides the case study, we perform extensive simulations to test our algorithm in more general scenarios (e.g., larger number of robots, size of the 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 deterministic shortest path problem (risk-averse problem (5.10)), which is $O(|E| + |V| \log |V|)$ for Dijkstra's algorithm.

We will study the number of deterministic shortest path problems solved by a single robot, i.e. $\sum_{j=1}^{n} K_j$ which is a key measure for the efficiency of our algorithm. However, this number is problem parameter dependent and it is hard to give *a priori* bounds. We will study the values for $\sum_{j=1}^{n} K_j$ with different numbers of robots and the size of the 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-SP

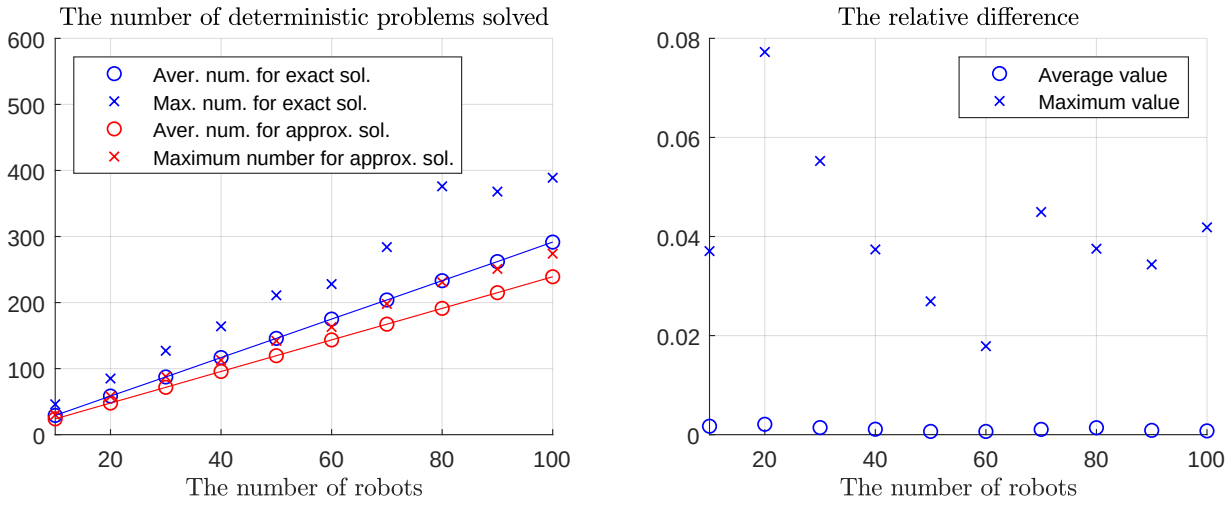


Figure 5.5: Scalability to the number of robots: (a) The number of deterministic shortest path problems solved by a single robot and (b) the relative difference of the travel costs obtained by our exact algorithm and approximate algorithm.

is scalable with the number of robots (tasks) and the size of the maps. (c) The solution obtained from the first step of our algorithm in Algorithm 8, i.e., the optimal solution of risk-averse problem (5.10) with $\bar{\lambda}$, is a nearly optimal solution.

The simulations were done on the computer with Intel i7 2.60GHZ CPU and 16G RAM. We assume the number of robots is same with the number of tasks. The desired probability p in chance constraint is 99%. We create different instances with randomly generated means and variance 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 variances are generated from a continuous uniform distribution with different magnitude of the range so that the edge with a higher mean also tends to have a higher variance. We test both proposed algorithm namely the exact algorithm and algorithm in which the subroutine solving CC-SP implements only the first step of our algorithm, namely the approximate algorithm.

5.5.1 Scalability with the Number of Robots

In the first set of simulations, we study the scalability of our algorithm to the number of robots varying from 20 to 100 with 10 increment. Robots are working on a graph with

2500 nodes and 27099 edges. The results are provided in Fig. 5.5. For a certain number of robots (x coordinate) in plot (a), we compute both average and maximum number of deterministic problems solved by a single robot from 100 instances with randomly generated ${}^{i}\mu_{uv}, {}^{i}\sigma_{uv}^{2}$. The average and maximum numbers for the exact algorithm are presented by blue circles and crosses. The results of the approximate algorithm are presented by red circles and crosses. It is shown that the average numbers for both exact and approximate algorithms grow almost linearly. It implies that the average number of deterministic problems solved for the CC-SP is close to a constant, irrespective of the number of robots. The approximate algorithm solves a 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 maximum relative difference[1] of the cost value between the exact solution and approximate solution obtained from 100 instances with a certain number of robots. The average differences are all less than 1% and the highest difference over the total number 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 from 3 to 300 seconds as the number of robots increase.

5.5.2 Scalability with the Size of Graph

In the second set of simulations, we study the scalability of our algorithms to the number of nodes in graphs varying from 500 to 2500 with 500 increment. The number of edges increases from about 8000 to 27000 accordingly. The number of robots is 50. The results are presented in Fig. 5.6. For a certain amount of nodes, we generate 100 instances under different graphs with the same amount of nodes and randomly generated ${}^{i}\mu_{uv}, {}^{i}\sigma_{uv}^{2}$. Similar to the first set of simulations, we compute the number of the deterministic shortest path problem solved by a single robot. As shown in Fig. 5.6 (a), the average numbers of the deterministic problem solved by both exact and approximate algorithms are nearly

[1] The relative difference is defined as $\frac{V^*-V}{V^*}$ where V is the evaluated objective function value and V^* is the optimal objective function value.

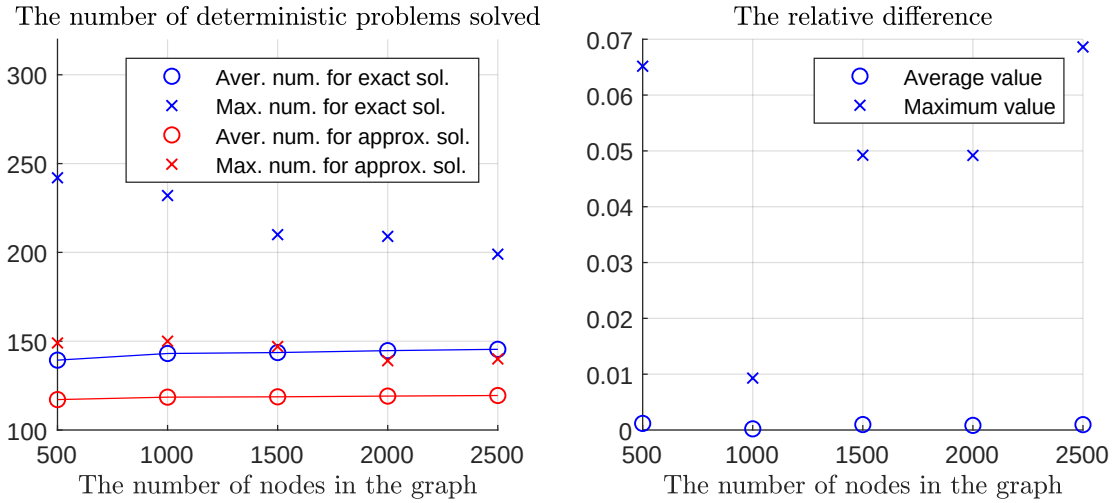


Figure 5.6: Scalability to the number of nodes of the graph: (a) The number of deterministic shortest path problems solved by a single robot and (b) the relative difference of the travel costs obtained by our exact algorithm and approximate algorithm.

constant. It implies that the size of the graph does not have a great influence on the number of deterministic problems solved by each robot. It influences more on the complexity of algorithm solving deterministic shortest path problems, i.e., $O(|E| + |V| \log |V|)$. Fig. 5.6 (b) shows that our approximate solution compute the nearly optimal solution (less than 1% average relative difference). The actual running time varies from 60 to 80 seconds as the number of nodes increases.

5.6 Conclusion

In this chapter, we present algorithms to solve the simultaneous task assignment and path planning problem under stochastic travel costs with the MinMax objective. We formulate the problem as a chance-constrained combinatorial optimization problem, called CC-STAP-B (5.3). We prove that the CC-STAP-B 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-constraint shortest path (CC-SP) 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 to 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.

Chapter 6

Chance-constrained Knapsack Problem

6.1 Introduction

In Chapters 3-5, we have discussed multi-robot task allocation with uncertainty in the performance criteria (more specifically, uncertainty in payoffs of a robot-task pair). In these problems, it is assumed that the number of tasks that one robot can perform is bounded (by a predefined number). The bound on the number of tasks a robot can perform is an approximate model of the finite energy resources or battery life the robot has during a mission. This model implicitly assumes that each robot consumes the same unit amount of resources for performing a task. However, due to the heterogeneity of robots and tasks, robots may consume a different amount of resources to perform a task. Further, when robots are working in an uncertain environment, the (energy) resource consumed by one robot to execute a task could be uncertain. The point-to-point task assignment problems in such scenarios are the stochastic generalized assignment problem (GAP), with uncertainty in resource consumption. In Chapters 6 and-7, we will discuss the problems of multi-robot coordination with uncertain resource consumption. In this chapter, we first address a sub-problem of stochastic generalized assignment, namely the chance-constrained knapsack problem (CC-KNAP).

The knapsack problem is a fundamental problem in combinatorial optimization that has multiple applications in task allocation and team formation in multi-robot systems, especially when we consider the limited battery life of the robots. For example, in algorithms to solve the generalized assignment problem for multiple robots, the knapsack problem is a sub-problem that needs to be solved [80]. The CC-KNAP problem can also be useful in

applications by itself, for example, in multi-robot team formation problems, where robots with limited battery life (and therefore a constraint on the distance they can travel) have to cover a given distance in the course of their task execution. The distance the robots have to cover is assumed to be much larger than the distance an individual robot can cover. Furthermore, the distance a robot can cover is a stochastic variable since it may depend on variables that are unknown at the team formation time. Such situations arise quite naturally in many applications including point-to-point material transfer, patrolling, and persistent surveillance.

Consider a perimeter patrolling application (see Figure 6.1) by a team of quadrotors, in which the robots with limited battery life have to move along a route of a given length (possibly multiple times). The total distance that has to be covered is much larger than the distance an individual robot can fly. Furthermore, the distance an individual robot can travel is uncertain because it depends on uncertain environmental variables (like wind speed for unmanned aerial vehicles). Thus, the distance a robot can travel can vary from one run of the robot to the next depending on the environmental conditions. There is an operating cost for each robot. The total cost of covering the route is a sum of the individual costs of robots. Our goal is to select a team of robots of minimum size or minimum total operating cost from a group of heterogeneous robots that covers the route with high probability (specified *a priori*), irrespective of the realization of the random distance that a robot can travel. Such a solution has the advantage that the selected team would be able to perform the patrolling task despite high variability in the environmental conditions.

The deterministic version of this problem where the travel distances are known constants can be formulated as a 0-1 knapsack problem. The classical 0-1 knapsack problem can be stated as follows: *Given a set of items, each with a weight and a value, determine the items to include in a knapsack so that the total weight is less than or equal to the weight carrying capacity of the knapsack and the total value is as large as possible [81].* To see the correspondence of our problem to the knapsack problem, note that the total length to be

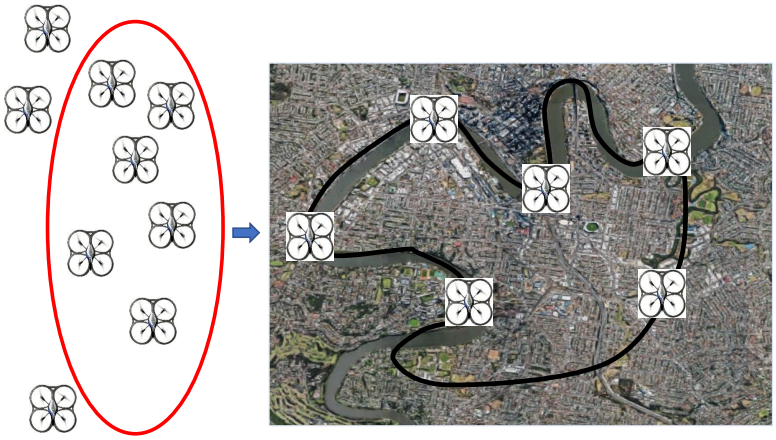


Figure 6.1: Perimeter patrolling application. A patrolling route for perimeter surveillance with a team of quadrotors. The goal is to select a team of robots from the given set so that the operating cost of patrolling (or the number of robots in the team) is minimized.

traveled by the robots corresponds to the capacity of the knapsack and the weights of the items correspond to the distance an individual robot can fly. At first glance, the deterministic version of our problem seems to be different from the classical knapsack problem since we are minimizing the total cost. However, the knapsack problem we are considering is equivalent to the classical 0-1 maximization knapsack problem. In fact, we can think of the minimization problem as the following maximization problem: find a subset of robots such that the total cost of unchosen robots is maximized and the total travel length of the robots is less than the total length of all robots in the given set minus the length of the given route. Therefore the algorithms that solve the classical 0-1 knapsack problem can also be used to solve the deterministic version of our problem. There are many methods to solve the knapsack problem such as dynamic programming [82], branch and bound method [83] and other methods that combine both dynamic programming and branch and bound [84, 85, 86]. Although solving the knapsack problem is NP-hard, there is a fully polynomial time approximation scheme [82].

Contributions: In this chapter, we present a novel algorithm that solves the 0-1 knapsack problem with a chance constraint. In [56], the authors consider a stochastic knapsack

problem similar to our setting and provide a polynomial time approximation scheme (PTAS) by using a parametric linear program. We present algorithms that solve chance-constrained knapsack optimally. The idea is to solve a sequence of methodically generated deterministic knapsack problems called the risk-averse knapsack problem. By analyzing the feasible region of both chance-constrained and risk-averse knapsack problems on the variance-mean plane, we prove that there exists a risk-averse knapsack problem such that the optimal solution of the chance-constrained knapsack problem is also the optimal solution of the risk-averse knapsack problem. We use this insight to develop an iterative algorithm where we solve the chance-constrained problem by repeatedly solving a sequence of the risk-averse knapsack problem. The key aspect of our algorithm is that we maintain a probabilistic guarantee irrespective of the realization of the random variables (the lengths the robots could move). We present simulation results on randomly generated data which show that our algorithm works efficiently.

6.2 Related Work

Chance-constrained optimization problems are a class of stochastic optimization problem [52, 87]. They are usually hard to solve (except for some special cases like linear optimization [52], minimum spanning tree [55]). Chance-constrained shortest path problems have been studied in [13] and the algorithm has been extended to a class of chance-constrained optimization problems, where the objective function is quasi-convex [14]. In our previous work [9] (presented in Chapter 3), we presented an algorithm for solving the chance-constrained linear assignment problem. In [9], we demonstrate the connection between the chance-constrained problem and risk-averse linear assignment problem and show that the optimal solution is obtained by using a one-dimensional search on the risk-averse parameter. In contrast to the stochastic linear assignment problem where the stochastic variables are in the objective of the deterministic problem, in this problem the stochastic variables are travel distances of robots which are in the resource constraint of the deterministic knap-

sack problem. Therefore the value-at-risk model used in [9] cannot be used to formulate the stochastic knapsack problem. Within the robotics literature on task allocation or team formation, problems with stochastic resource constraints have been studied in [59, 60, 65].

There are different stochastic variations of the classical 0-1 knapsack problem that have been studied in the extant literature. In [88, 89, 90], the authors have studied the stochastic knapsack problem with deterministic weights and random costs whereas in our problem we have deterministic costs and random weights. In fact, the algorithms for CC-LAP presented in Chapter 3 are able to solve such problems by using algorithm solving deterministic knapsack problems. In [91], the authors compute a solution policy that optimizes the expected total values. Optimizing expected values provide no performance guarantees on a particular realization of the random variables. We want to develop methods that ensure the constraints are satisfied with a high probability irrespective of the realization of the random weights. An algorithm is designed to obtain good solutions to the chance-constrained problem in [92], by running a sequence of robust problems. The algorithm provides an optimal solution when the costs are identical or the uncertain weights present all the same characteristics. In this chapter, our method computes the optimal solution in a more general situation. Literature [57, 58, 56] considers chance-constrained knapsack problems similar to the problem studied in this chapter. In [57], the item sizes (travel distance in our case) are assumed to have Bernoulli-type distribution. In other words, there are two possible outcomes of the item size. The authors present an $\mathcal{O}(\log \frac{1}{1-p})$ approximation algorithm where p is the probability. Authors in [58] assume item sizes have Poisson or exponential distribution. A polynomial time approximation scheme (PTAS) is provided however the chance constraint could be violated by a factor of $(1+\epsilon)$. In [56], the authors provide a PTAS by using a parametric linear programming reformulation. A relaxed FPTAS for chance-constrained knapsack is presented by [93]. However, the knapsack capacity is relaxed.

6.3 Problem Formulation

Let L be the length of the closed curve (or a route) that a team of robots has to cover. We have a collection of heterogeneous robots with different battery life and they can fly for different lengths. Let ℓ_i be the distance that robot i can fly. Each robot has a different operating and maintenance cost denoted by c_i. The variable ℓ_i is assumed to be a Gaussian random variable with mean μ_i and variance σ_i^2, i.e., $\ell_i \sim \mathcal{N}(\mu_i, \sigma_i^2), i = 1, ..., n$. Our goal is to find a set of robots from the collection of n robots that can cover the total length L with probability at least p (where $0.5 \leq p \leq 1$) while minimizing the total cost. Let f_i be an integer variable that takes the value 1 if robot i is selected and 0 otherwise. The integer program formulation of our problem is as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} c_i f_i \\ \text{s.t.} \quad & \mathbb{P}\left(\sum_{i=1}^{n} \ell_i f_i \geq L\right) \geq p \\ & f_i \in \{0,1\}, \quad \forall i = 1, \ldots, n \end{aligned} \tag{6.1}$$

The formulation above is a stochastic variation of the minimization version of 0-1 knapsack problem, which we call the chance-constrained knapsack problem (CC-KNAP). In the deterministic knapsack problem, the probabilistic constraint in (6.1) constraint is replaced by the deterministic constraint $\sum_{i=1}^{n} \ell_i f_i \geq L$. The minimization version of the knapsack (which we are generalizing with the stochastic constraint) is equivalent to the maximization version [82]. Intuitively, minimizing the cost of robots in the team is equivalent to maximizing the cost of robots that are not chosen in the team. Note that if we set the costs to be identical (say 1), we find the robot team with the minimum number of robots such that the route is covered by the team with high probability, despite the uncertainty in the length that each robot can fly. The formulation in (6.1) is a chance-constrained combinatorial optimization. As discussed in CC-LAP, the chance constraint can be written as a quadratic inequality with variable $\sum_{i=1}^{n} \mu_i f_i$ and $\sum_{i=1}^{n} \sigma_i^2 f_i$. Therefore formulation of CC-KNAP (6.1)

can be equivalently written as following.

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} c_i f_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} \mu_i f_i - C\sqrt{\sum_{i=1}^{n} \sigma_i^2 f_i} \geq L \\ & f_i \in \{0,1\}, \quad \forall i = 1, \ldots, n \end{aligned} \tag{6.2}$$

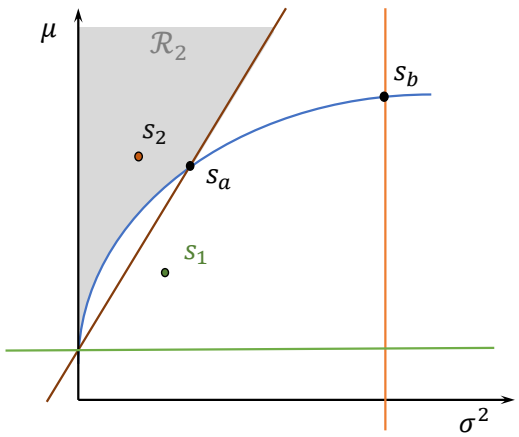
where $C = \Phi^{-1}(p)$ with Φ^{-1} representing the inverse cumulative distribution function of a standard normal distribution with zero mean and unit variance.

If we relax f_i, the problem in (6.2) is a second order cone program with integrality gap $\Omega(\sqrt{n})$ [56]. In [56], the authors converted the formulation of CC-KNAP (6.2) to a parametric linear program and presented an algorithm that gives a $1 - 3\epsilon$ approximate solution with running time $O(\frac{1}{\epsilon^2} n^{\frac{1}{\epsilon}})$. We present an alternate parametric formulation, where different choices of the parameter lead to different knapsack problems. In the discussion below we will refer to both (6.1) and (6.2) as chance-constrained knapsack problem (CC-KNAP), which is a chance-constrained integer optimization problem and is hard to solve in general. Instead of solving CC-KNAP directly, we show that the solution to CC-KNAP can be obtained by solving a number of deterministic knapsack problems (given below), which is called the risk-averse knapsack problem (RA-KNAP).

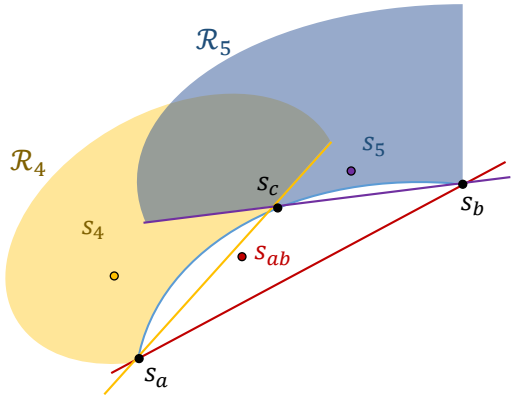
$$\begin{aligned} \min \quad & \sum_{i=1}^{n} c_i f_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} \mu_i f_i - \lambda \sum_{i=1}^{n} \sigma_i^2 f_i \geq L' \\ & f_i \in \{0,1\}, \quad \forall i = 1, \ldots, n \end{aligned} \tag{6.3}$$

Here λ is the risk-averse parameter that performs a weighted combination of the mean and variance of the travel lengths of each robot. The parameter L' is the constraint for the total length in RA-KNAP. Consider the situation when $\lambda = 0$ and $L' = L$, the problem becomes the classical 0-1 knapsack problem. Our solution strategy is to iteratively search over λ and L' to find the appropriate λ and L' such that the solution to the RA-KNAP gives a solution to the CC-KNAP. We justify this in the next section.

6.4 Geometric Interpretation



(a) Find a feasible solution of CC-KNAP by solving RA-KNAPs with increasing λ.



(b) Search the rest of feasible region of CC-KNAP by solving RA-KNAPs with methodically updated λ and L'

Figure 6.2: Geometric interpretation of the CC-KNAP and illustration of our algorithm. Any solution is related to a point on the variance-mean plane. The feasible region for CC-KNAP is the space above the parabola while the feasible region for RA-KNAP with given λ and L is the space above the line whose slope is equal to λ and vertical intercept is equal to L

In this section, we present a geometric interpretation of the CC-KNAP on the variance-mean plane in which the horizontal and vertical coordination represent the variance and mean of the travel distance of the selected robots respectively, i.e., $\sigma^2 = \sum_i^n \sigma_i^2 f_i$ and $\mu = \sum_i^n \mu_i f_i$. Therefore given any particular solution $s = \{f_1, ..., f_n\}$, we can identify this solution with a point with coordinates (σ^2, μ) on the variance-mean plane. Thus the space of all possible robot teams can be identified with points in the variance-mean plane (however, we do not construct this explicitly because the number of such points will be exponential in the number of robots). Moreover, we can find the feasible region of solution for the CC-KNAP based on the chance constraint in Formulation (6.2). As shown in Fig. 6.2a, the feasible region for CC-KNAP is space above the parabola in the first quadrant on the variance-mean plane. Since the first constraint in the RA-KNAP is a linear inequality of σ^2 and μ, the feasible region for RA-KNAP is the space above the line whose slope is equal to risk-averse parameter

λ and vertical intercept is equal to the length of the route L'. Based on the geometric interpretation, we present the following lemmas:

Lemma 17. *The optimal solution of RA-KNAP that satisfies the chance constraint provides an upper bound of the optimal solution of CC-KNAP.*

Proof. When the optimal solution of RA-KNAP satisfies the chance constraint, the corresponding point on the variance-mean plane must be in the intersection of feasible regions of CC-KNAP and RA-KNAP (e.g., s_2 in Fig. 6.2a). If the optimal solution of CC-KNAP is in this intersection, then the optimal objective function values of CC-KNAP and RA-KNAP are the same. If the optimal solution of CC-KNAP is not in the intersection, it implies that the optimal objective value of RA-KNAP is greater than the optimal objective value of CC-KNAP. Therefore the optimal objective function value of RA-KNAP must be greater than or equal to the optimal objective function value of CC-KNAP. □

Lemma 18. *If the constraint of RA-KNAP is tangent to the parabola of chance constraint, the optimal solution of this particular RA-KNAP must be a feasible solution of CC-KNAP.*

Proof. When the linear constraint of RA-KNAP is tangent to the parabola, the feasible region of RA-KNAP is the subset of the feasible region of CC-KNAP. Therefore the points of all feasible solutions including the optimal solution of RA-KNAP are in the feasible region of CC-KNAP. □

Lemma 19. *There exists one RA-KNAP such that the optimal solution of CC-KNAP is also the optimal solution of the RA-KNAP.*

Proof. As it is stated in the proof of Lemma 18, the feasible region of RA-KNAP in which the constraint is tangent to the parabola is a subset of the feasible region of CC-KNAP. There always exists a straight line tangent to the parabola (the slope and vertical intercept are denoted by $\bar{\lambda}$ and $\bar{L}'$ respectively), such that the half-space defined by the resource constraint in (6.3) contains the point corresponding to the optimal solution of CC-KNAP (say s^*). Thus s^* is also the optimal solution of RA-KNAP. This is also true for RA-KNAP

with parameters different with $\bar{\lambda}$ and $\bar{L}'$ as long as the upper half-space of the 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 the variance-mean plane (change parameter λ and L' of RA-KNAP) in such a way that s^* is above it and no feasible solution of RA-KNAP below the parabola has an objective value less than s^*), the optimal solution of the corresponding RA-KNAP is still s^*. □

Lemma 19 indicates that the key to solving the CC-KNAP is to solve the RA-KNAP with the particular slope $\bar{\lambda}$ and vertical intercept $\bar{L}'$ whose optimal solution satisfies the chance constraint. However Lemma 17 tells that solving the RA-KNAP for a particular choice of λ and L' does not mean we have obtained the optimal solution, probably could be a feasible solution providing an upper bound instead. Therefore we need to design a method to iterate over different values of λ and L' methodically to cover the whole area of the feasible region of CC-KNAP. The optimal solution would be the solution with the smallest total cost among feasible solutions. Our algorithm provides a systematic approach to perform this search.

6.5 Algorithm

In this section, we present our algorithm to solve the CC-KNAP. As shown in Fig. 6.2a, the feasible region of CC-KNAP (6.2) denoted by $\mathcal{R}$ is the non-convex space above the parabola $\{(\sigma^2, \mu)|\mu - C\sigma = L\}$ from the chance constraint. The feasible region of RA-KNAP (6.3) is the space above the straight line $\{(\sigma^2, \mu)|\mu - \lambda\sigma^2 = L'\}$ from the resource constraint. Let the intersection of the feasible regions of CC-KNAP (6.2) and the risk-averse problem, RA-KNAP (6.3), in the first quadrant be $\mathcal{R}_i$ where i is the index for the RA-KNAP. Lemma 17 implies that the optimal solution of RA-KNAP with λ_i and L'_i that satisfies the chance constraints is also optimal for CC-KNAP within the feasible region $\mathcal{R}_i$ (e.g., s_2 is the optimal solution of CC-KNAP within the region $\mathcal{R}_2$ in Fig. 6.2a). Let $\mathbb{S}$ be the set of solutions obtained from RA-KNAP that satisfy the chance constraint. For ease

of exposition, $\mathbb{S}$ also denotes the set of indices (subscripts) of those solutions by abuse of notation. Our idea is to solve multiple RA-KNAP and obtain solutions satisfying the chance constraint such that the union of corresponding feasible regions $\mathcal{R}_i$ covers the feasible region of chance-constrained problem, i.e., $\mathcal{R} \subset \bigcup_{i \in \mathbb{S}} \mathcal{R}_i$.

Our algorithm has two steps: (a) Find a feasible solution to CC-KNAP by solving RA-KNAP with increasing λ in (6.3) (line 1-7); (b) search for the optimal solution within the rest of the feasible region of CC-KNAP by solving RA-KNAP with methodically updated λ and L' (line 8-10). The procedures are shown in Algorithm 11 and Algorithm 12.

The first step starts with solving RA-KNAP with $\lambda_k = 0$, $L'_k = L$, $k = 0$ (line 1-2). If the optimal solution of RA-KNAP does not satisfy the chance constraint, in other words the corresponding point is below the parabola, e.g. s_1 in Fig. 6.2a, the algorithm computes the resource constraint of the RA-KNAP for the next iteration by updating λ from equation $\lambda_{k+1} = C/\sigma$ where $\sigma = \sqrt{\sum_{j=1}^{n} \sigma_j^2 f_j}$ (line 4). From a geometric point of view, the update procedure can be treated as the straight line from RA-KNAP rotating counter-clockwise at vertical intercept $(0,\ L)$. The new straight line is guaranteed to be located above the previous point. The procedure continues until we obtain a feasible solution of chance-constrained problem, e.g., s_2 in shaded region $\mathcal{R}_2$ above the parabola in Figure 6.2a (line 3-6). The feasible solution s_2 is stored in the solution set $\mathbb{S}$ (line 7). Now the subset of feasible region of CC-KNAP, $\mathcal{R}_2$ is dominated by s_2, i.e., $s_2 = \arg\min_{s \in \mathcal{R}_2} v(s)$ where $v(\cdot)$ denotes the objective function of CC-KNAP (6.2). Note that this procedure terminates in finite number of iterations because λ increases as algorithm proceeds. In worst case, RA-KNAP becomes a infeasible problem, i.e., $\sum_{j=1}^{n} \mu_j - \lambda \sum_{j=1}^{n} \sigma_j^2 < L$. In other words, in this deterministic knapsack problem the travel distance for robots are too small to cover the route even all available robots are used. At this moment, the procedure moves to line 7 with solution set $\mathbb{S}$ as an empty set.

In the second step, the algorithm computes the intersection point of parabola and the straight line of the last RA-KNAP in previous step, e.g., s_a in Fig. 6.2. Further it

computes the right-most point on the parabola with horizontal coordinate $\sigma^2 = \sum_{i=1}^{n} \sigma_i^2$, e.g., s_b in Fig. 6.2 (line 8). Therefore the search region is limited to the space embraced by the parabola from s_a to s_b, vertical straight line through s_b and the straight line from the last RA-KNAP. Then the algorithm calls a recursive function $\mathcal{A}(s_a, s_b, \mathbb{S})$ presented in Algorithm 12 (line 9). First, $\mathcal{A}(s_a, s_b, \mathbb{S})$ computes the equation of the straight line through s_a and s_b. And then solve RA-KNAP with λ and L' equal to 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 $L' = \mu_a - \lambda \sigma_a^2$ (line 1). If the obtained solution $s_{ab} = (\sigma_{ab}^2, \mu_{ab})$ does not satisfy the chance constraint, algorithm computes 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 it computes a intersection point s_c 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 prevents the cycling (line 2-3). Then it calls the recursive functions $\mathcal{A}(s_a, s_c, \mathbb{S})$ and $\mathcal{A}(s_c, s_b, \mathbb{S})$. 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. When all recursive functions terminate, we obtain a set of feasible solutions such that the feasible region of CC-KNAP, $\mathcal{R}$ is covered by the union of intersection region corresponding to those feasible solutions in $\mathbb{S}$, i.e., $\mathcal{R} \subset \bigcup_{i \in \mathbb{S}} \mathcal{R}_i$. The optimal solution is one with the smallest objective function value. Algorithm 11 terminates (line 10).

Algorithm 11 Algorithm to solve CC-KNAP

Input: $L,\ p,\ \mu_i,\ \sigma_i^2\ \forall i = 1, \ldots, n.$
Output: The optimal solution s^* and the optimal objective value v^*.
1: Initialize $k = 0$, $\lambda_k = 0$, $L'_k = L$.
2: Solve RA-KNAP with λ_k and L'_k and obtain solution s_k with objective value v_k.
3: **while** s_k does not satisfy the chance constraint **do**
4: Update risk-averse parameter $\lambda_{k+1} = C/\sigma$, $L'_{k+1} = L'_k$ and $k = k + 1$.
5: Solve RA-KNAP with λ_k and L'_k and obtain solution s_k.
6: **end while**
7: Store solution s_k in set $\mathbb{S}$, .
8: Compute intersection point s_a and s_b.
9: Call recursive function $\mathcal{A}(s_a, s_b, \mathbb{S})$.
10: **return** $v^* = \min_{s \in \mathbb{S}}\ v(s)$ and solution $s^* = \arg\min_{s \in \mathbb{S}}\ v(s)$.

Algorithm 12 $\mathcal{A}(s_a, s_b, \mathbb{S})$

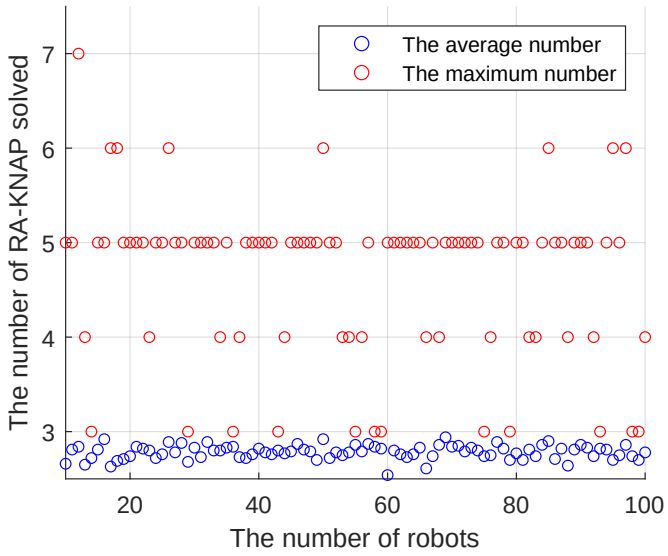
Input: $s_a, s_b, \mathbb{S}$.
Output: $\mathbb{S}$.
1: Solve RA-KNAP with $\lambda = \frac{\mu_a - \mu_b}{\sigma_a^2 - \sigma_b^2}$ and $L' = \mu_a - \lambda\sigma_a^2$.
2: **if** the solution $s_{ab} = (\sigma_{ab}^2, \mu_{ab})$ does not satisfy the chance constraint **then**
3: Compute intersection 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 $\mathcal{A}(s_c, s_b, \mathbb{S})$.
5: **end if**
6: **return** $\mathbb{S} = \{\mathbb{S}, s_{ab}\}$.

6.6 Simulation Results

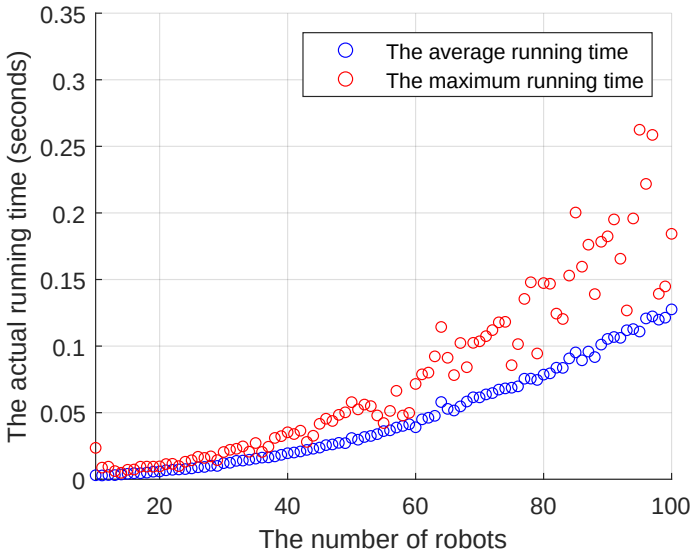
In this section, we present simulation results to characterize the performance of our algorithm. The goal of the simulation studies is to understand the scalability of our algorithm as a function of the number of robots and the uncertainty in knowledge about the distance the robots can travel (i.e., variances). Our algorithm solves the chance-constrained knapsack problem by solving a number of deterministic risk-averse knapsack problems (RA-KNAP). Thus, the efficiency of our algorithm depends on the number of deterministic knapsack problems solved. To understand the effects of parameters such as the number of robots and the variance of the travel distance of robots, we generated different scenarios based on randomly generated parameter values. We will first present simulation results in which the mean and variances of the travel distance of robots are randomly generated and the number of robots is varying from 10 to 100 (Fig. 6.3). Then we will present the simulation results in which the variances of travel distance are varied (Fig. 6.4). Our simulations were performed on the computer with Intel 2.60 GHz processor and 16.0 GB RAM.

6.6.1 Scalabity to the Number of Robots

In the first simulation, we test the effect of the number of robots on the number of RA-KNAPs solved which in turn determines the algorithm performance. We use dynamic programming to solve the RA-KNAP optimally in pseudo-polynomial time, $\mathcal{O}(n^2P)$, where n is the number of robots and P is the largest cost among all robots [82]. As mentioned in



(a) The average and maximum number of RA-KNAPs solved.



(b) The average and maximum of the actual running time.

Figure 6.3: The scalability of our algorithm to the number of the given robots. For a certain number of given robots, the average and maximum value are computed from 100 simulations with randomly generated means and variances for stochastic travel distances

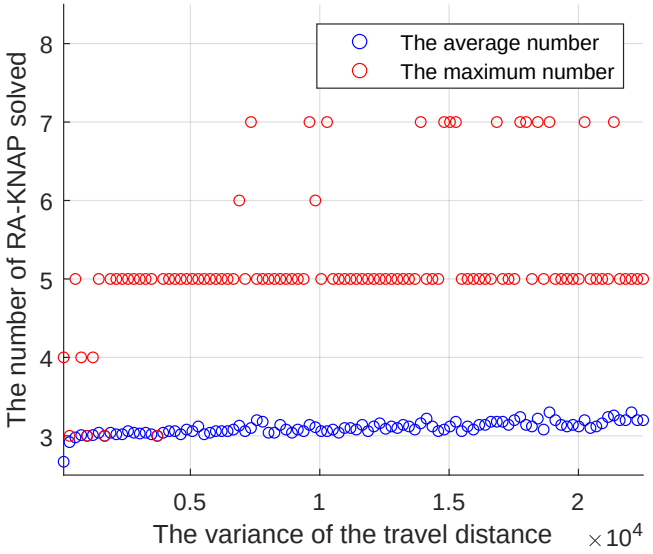
Section 6.3, the travel distance of each robot is a Gaussian random variable. The means and variances of these random variables are generated independently from a uniform distribution

$\mu_i \sim \mathcal{U}(1000, 3000)$ and $\sigma_i^2 \sim \mathcal{U}(10000, 12500)$. The total length that the robots have to traverse as a team, L, is $10,000$ meters. The probability that the robots cover the route is 0.99. In other words, the total distance the chosen robots can cover is greater than $10,000$ with a probability 0.99, despite the actual realization of their travel length in any given scenario. Moreover the operation and maintenance cost for each robot is drawn uniformly from 50 to 150, i.e., $c_i \sim \mathcal{U}(50, 150)$. We set $\epsilon = 1 \times 10^{-7}$. We count the number of RA-KNAPs for solving CC-KNAP when the number of robots is equal to $10, 11, \ldots, 100$. For each case with a given number of robots, we generate 100 random scenarios and compute both the average and maximum number of RA-KNAP solved from 100 scenarios.

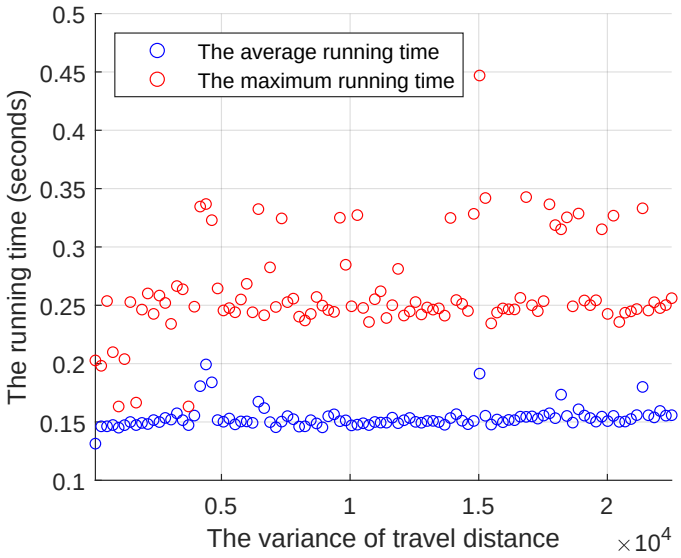
Fig. 6.3 show the performance of our algorithm with different numbers of robots. In Fig. 6.3a, the blue dots represent the average number of RA-KNAPs to solve CC-KNAP with a given number of robots while the red dots represent the maximum number of calls to RA-KNAPs. Each dot is generated with the results from 100 simulations. As it is shown, the average number of RA-KNAP solved is almost constant (between 2.5 to 3) irrespective of the number of robots. In Fig. 6.3a, the maximum numbers of RA-KNAPs solved are between 3 and 7, which is a small value for practical applications. The results reveal that the number of robots does not have a significant influence on the speed of our algorithm and the number of calls to RA-KNAPs is nearly a constant. Another way to measure the speed of our algorithm is the actual running time. As shown in Fig. 6.3b, the maximum running time is less than 0.3 seconds and both the average and maximum numbers increase polynomially with respect to the number of robots. This is because the algorithm used to solve deterministic RA-KNAP is a $O(n^2P)$ algorithm while the number of calls RA-KNAPs is nearly a constant.

6.6.2 Scalability to the Variances

In the second simulation, we study the effect of the uncertainty of travel distance of the robots on the performance of our algorithm by counting the number of calls to RA-KNAPs



(a) The average and maximum number of RA-KNAPs solved.



(b) The average and maximum of the actual running time.

Figure 6.4: The scalability of our algorithm to the uncertainty in knowledge about the travel distance of robots (variance). For a certain value of variances, the average and maximum value are computed from 100 simulations with 100 given robots and randomly generated means for stochastic travel distances.

and the actual running time for our algorithm solving CC-KNAP with variance equal to $100, 324, 548, \ldots, 22500$. For each value of variance, we generate means of the travel distance

of robots randomly, by sampling from the uniform distribution, $\mathcal{U}(1000, 3000)$. For each value of variance, we generate 100 different scenarios. The number of robots is 100 and the length of the route is $50,000$ meters for all scenarios in this simulation. The other parameters such as the cost, c_i, the probability in the chance constraint, p, and ϵ are the same as that in the first simulation. Fig. 6.4a shows that the average number of calls (blue dots) to RA-KNAPs is practically constant as the variance of travel distance increases. The maximum numbers of calls (red dots) range from 3 to 7. Fig. 6.4b shows that the average running time is about 0.15 seconds and the maximum running times are less than 0.45 seconds. This is because the increased variance does not increase the computational burden for the algorithm solving the deterministic RA-KNAP and the number of calls is constant. Fig. 6.4a and Fig. 6.4b imply that our algorithm scales well when the uncertainty of the travel distance of robots is high.

6.7 Conclusion

In this chapter, we presented an algorithm for computing the optimal solution for chance-constrained knapsack problem with a deterministic objective and stochastic knapsack constraint. The key idea in our approach is to convert CC-KNAP to a deterministic discrete optimization problem on the variance-mean plane, where each point on the plane can be identified with an assignment of items to the knapsack. By exploiting the geometry of the non-convex feasible set of the CC-KNAP in the variance-mean plane, we showed that CC-KNAP can be solved optimally by solving a sequence of deterministic risk-averse knapsack problems. We presented an algorithm to systematically generate the sequences of RA-KNAPs to solve the CC-KNAP.

Our problem formulation is motivated by team selection problems in multi-robot systems, where robots have limited battery life, under uncertain knowledge about their durability in the mission. We presented simulation results on randomly generated data to show empirically that the number of RA-KNAPs required to solve the CC-KNAP is a small con-

stant (less than 7), irrespective of the number of robots or the uncertainty in the knowledge about the distance that a robot can travel. Thus, our algorithm scales well with the number of robots from which we can choose the team and also with the variance of the battery life, i.e., the distance that a robot can travel.

Although empirically our algorithm seems to be computationally efficient, we do not have any theoretical guarantees about the computational complexity of our algorithm. We will work on analyzing the computational complexity of our algorithm as a part of the future work of this thesis. We extend our method to solve the stochastic generalized assignment problem, presented in Chapter 7.

Chapter 7

Multi-robot Chance-constrained Generalized Assignment Problem with Stochastic Resource Consumption

As the knapsack problem is a sub-problem for the generalized assignment problem, we will extend our algorithm for CC-KNAP to the stochastic generalized assignment problem (SGAP).

7.1 Introduction

Multi-robot generalized assignment problem (GAP) [79, 15, 94], 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 payoff. GAP is an NP-hard combinatorial optimization problem which has been extensively studied both in operations research [95, 96] and theoretical computer science [97, 98, 15, 99]. However, in many multi-robot application scenarios, the resource consumption of tasks is influenced by uncertain environmental factors and may not be known exactly before the execution. Consequently, the resource consumption could be stochastic. Therefore in this chapter, we design algorithms that provide assignment with *a priori* probabilistic guarantee on the stochastic resource consumption of the generalized assignment problem.

GAP with stochastic resource consumption is related to many multi-robot applications. The resource could be energy, task execution time, or any other consumable resources

depending on the 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, the robot may consume more energy when there are moving obstacles on the way to the task location.

In this chapter, 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). The goal to solve CC-GAP is: *find the assignment of robots to tasks with maximum team payoff such that each robot performs the tasks successfully and its resource constraint is not violated with high probability (say* 0.99*) irrespective of the actual value the random resource consumption taken in any realization.* Our solution provides $(1 + \alpha)$ approximate solution to CC-GAP if the subroutine algorithm to solve the deterministic knapsack problem is α approximation ($\alpha \geq 1$).

Contributions: We model the stochastic generalized assignment problem as a chance-constrained combinatorial optimization problem. Inspired by the idea from the deterministic GAP in [15], we design an algorithm for CC-GAP where each robot solves a chance-constrained knapsack problem (CC-KNAP) sequentially. Leveraging optimal algorithm for CC-KNAP from Chapter 6, 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 $(1 + \alpha)$-approximation (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.

Most of the current literature on stochastic generalized assignment problem (SGAP) study the problem in which the resource consumption is uncertain before an initial assign-

ment 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 in those literature [40, 41, 43, 100, 42]. However, in robotics applications, the one-off assignment must be made and the task has to be executed before the random resource cost is known. The two-stage programs cannot be applied in these applications. We model SGAP as the chance-constrained combinatorial optimization problem, which allows us to obtain a risk-averse solution.

7.2 Problem Formulation

We will introduce the chance-constrained generalized assignment problem and the subproblem CC-KNAP. We will show that the relationship between the chance-constrained knapsack problem and the deterministic risk-averse knapsack, which is important to design the algorithms for CC-GAP.

We consider a set of robots $\{r_1, \ldots, r_{n_r}\}$ and a set of tasks $\{t_1, \ldots, t_{n_t}\}$. 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

$$\begin{aligned} \max \quad & \sum_{i=1}^{n_r} \sum_{j=1}^{n_t} a_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{n_t} w_{ij} x_{ij} \leq W_i, \quad \forall i = 1, \ldots, n_r \\ & \sum_{i=1}^{n_r} x_{ij} \leq 1, \quad \forall j = 1, \ldots, n_t \\ & x_{ij} \in \{0, 1\}, \quad \forall i, j \end{aligned} \tag{7.1}$$

where the first constraint in (7.1) is the *resource constraint* for each robot. The second constraint guarantees that each task is exclusively assigned to at most one robot.

In (7.1), the resource consumption variable, w_{ij}, is assumed to be deterministic. However, in practice, the resource consumption is usually a random variable. Thus, the satisfac-

tion of the resource constraint in (7.1) 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 (7.1), in which the resource constraint is represented as a stochastic constraint.

7.2.1 Chance-constrained Generalized Assignment Problem

We assume that resource consumption w_{ij} are independent Gaussian random variables with known mean μ_{ij} and variance σ_{ij}^2. To obtain the probabilistic guarantee for the one-off decision (assignment) making, the problem is formulated as the chance-constrained generalized assignment problem (CC-GAP) in (7.2). 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*).*

$$\begin{aligned} \max \quad & \sum_{i=1}^{n_r}\sum_{j=1}^{n_t} a_{ij}x_{ij} \\ \text{s.t.} \quad & \mathbb{P}\left(\sum_{j=1}^{n_t} w_{ij}x_{ij} \le W_i\right) \ge p, \quad \forall i = 1,\dots,n_r \\ & \sum_{i=1}^{n_r} x_{ij} \le 1, \quad \forall j = 1,\dots,n_t \\ & x_{ij} \in \{0,1\}, \quad \forall i,j \end{aligned} \tag{7.2}$$

$$\begin{aligned} \max \quad & \sum_{j=1}^{n_t} a_{kj}x_{kj} \\ \text{s.t.} \quad & \mathbb{P}\left(\sum_{j=1}^{n_t} w_{kj}x_{kj} \le W_k\right) \ge p, \\ & x_{kj} \in \{0,1\}, \quad \forall j \end{aligned} \tag{7.3}$$

The first constraint in (7.2) is the chance constraint which guarantees that for each robot, r_i, the resource consumption of 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.

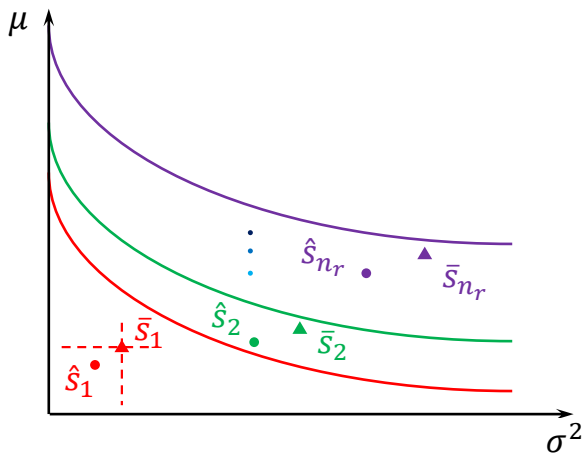


Figure 7.1: The assignment of any robot is represented by a point on the variance-mean plane. The feasible solution of CC-GAP is a set of points that are under the parabola defined by the corresponding chance constraints, e.g., $\{\hat{S}_i\}_{i=1}^{n_r}$.

Due to the assumption on the random resource consumption w_{ij}, the total resource consumption for each robot is a Gaussian random variable with mean $\mu = \sum_{j=1}^{n_t} \mu_{ij} x_{ij}$ and variance $\sigma^2 = \sum_{j=1}^{n_t} \sigma_{ij}^2 x_{ij}$. Therefore the chance constraint can be equivalently written as a quadratic inequality of μ and σ^2 (details are provided in Section 3.3). The CC-GAP formulation (7.2) is equivalent to formulation

$$\begin{aligned} \max \quad & \sum_{i=1}^{n_r} \sum_{j=1}^{n_t} a_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j=1}^{n_t} \mu_{ij} x_{ij} + C \sqrt{\sum_{j=1}^{n_t} \sigma_{ij}^2 x_{ij}} \leq W_i, \quad \forall i = 1, \ldots, n_r \\ & \sum_{i=1}^{n_r} x_{ij} \leq 1, \quad \forall j = 1, \ldots, n_t \\ & x_{ij} \in \{0, 1\}, \quad \forall i, j \end{aligned} \tag{7.4}$$

where $C = \Phi^{-1}(p)$ and $\Phi^{-1}(\cdot)$ is the inverse cumulative distribution function of standard Gaussian distribution.

Now consider the assignment for any robot r_i as a point on variance-mean plane 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 respectively, i.e., $\sigma^2 = \sum_{j=1}^{n_t} \sigma_{ij}^2 x_{ij}$

and $\mu = \sum_{j=1}^{n_t} \mu_{ij} x_{ij}$ (see Fig. 7.1). Eq. (3.4) implies that the chance constraint for each robot is a parabola with vertical intercept equal to W_i.

Any feasible assignment for r_i should be located below the parabola obtained from the chance constraint of (7.4). Therefore *the feasible solution of CC-GAP in* (7.2) *and* (7.4) *is a set of points below the corresponding parabola defined by the chance constraint on the variance-mean plane.* Besides, the assignments in the feasible solution of CC-GAP should also satisfy the assignment constraints in (7.2).

The idea of our method is to compute the feasible assignment that satisfies the chance constraint for each robot ($\bar{S}_i$ in Fig. 7.1) 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. 7.1).

7.2.2 Chance-constrained Knapsack Problem

Now consider the problem for a single robot, say r_k, which is a special case of CC-GAP. The problem is the chance-constrained knapsack problem (CC-KNAP). 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. The formulation of CC-KNAP is presented below.

$$\begin{aligned} \max \quad & \sum_{j=1}^{n_t} a_{kj} x_{kj} \\ \text{s.t.} \quad & \sum_{j=1}^{n_t} \mu_{kj} x_{kj} + C \sqrt{\sum_{i=j}^{n_t} \sigma_{kj}^2 x_{kj}} \leq W_k \\ & x_{kj} \in \{0,1\}, \quad \forall j = 1, \ldots, n_t \end{aligned} \tag{7.5}$$

As discussed in Chapter 6, the optimal solution of CC-KNAP can be solved by solving methodically generated deterministic knapsack problems, called RA-KNAP. The formulation

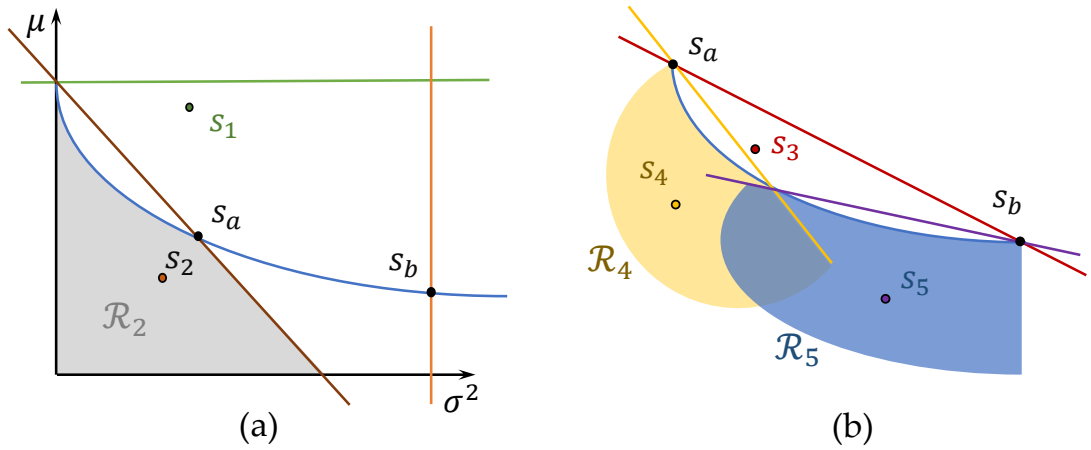


Figure 7.2: The feasible assignments of CC-KNAP and RA-KNAP are located below the parabola defined by chance constraint in (7.5) and the straight line defined by the resource constraint in (7.6) respectively. $\mathcal{R}_i$ denotes the intersection of the feasible region of CC-KNAP and RA-KNAP.

of RA-KNAP is given as following.

$$\begin{aligned} \max \quad & \sum_{j=1}^{n_t} a_{kj} x_{kj} \\ \text{s.t.} \quad & \sum_{j=1}^{n_t} \mu_{kj} x_{kj} + \lambda \sum_{j=1}^{n_t} \sigma_{kj}^2 x_{kj} \leq W' \\ & x_{kj} \in \{0,1\}, \quad \forall j = 1, \ldots, n_t \end{aligned} \tag{7.6}$$

As shown in Fig. 7.2(a), the feasible region defined by the resource constraint is the space below the straight line. The slope of the straight line is equal to $-\lambda$ where $\lambda > 0$ is the risk-averse parameter. The vertical intercept of the straight line is equal to W', which is the resource capacity of the robot. The resource consumption of the robot is equal to the linear combination of mean and variance, i.e., $\mu_{kj} + \lambda\sigma_{kj}^2$.

Depending on the position of straight line defined by the resource constraint in (7.6), the optimal solution of (7.6) could be either below or above the parabola defined by chance constraint in (7.5). In particular, the optimal solution of (7.6) is feasible to (7.5) if the solution is below the parabola, e.g., s_2 in Fig. 7.2(a). The solution is thus a lower bound of optimal solution of (7.5). In a special case, the straight line from RA-KNAP in (7.6)

is tangent to the parabola from CC-KNAP in (7.5). Then any feasible solution of (7.6) is below the parabola and therefore feasible to (7.5). It implies an important statement that *there exists a problem* (7.6) *with proper values of* λ *and* W' *such that the optimal solution of* (7.6) *is also the optimal to problem* (7.5)[1] . The key to solve the CC-KNAP is to find proper λ and W'.

7.3 Algorithms

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 area of the 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-approximation when the approximation ratio for subroutine solving RA-KNAP is α. Consequently, the obtained solution of CC-GAP is $(1+\alpha)$-approximation.

Definition 2. *A solution of an optimization problem* $\bar{s}^*$ *is* α*-approximation, if and only if* $\alpha v(\bar{s}^*) \geq \max_{s\in\mathcal{R}} v(s)$ *where* $v(\cdot)$ *denotes the objective function and* $\mathcal{R}$ *denotes the feasible region.*

7.3.1 Algorithm for CC-KNAP

In Chapter 6, we study a multi-robot team formation problem which is formulated as a knapsack problem with stochastic traveling distance and the objective is to minimize the total traveling cost of the robot team. The solution is obtained by solving a sequence of deterministic knapsack problems. The assumption is that the deterministic knapsack problem is solved optimally. In this chapter, we further analyze and prove that the solution of the algorithm for CC-KNAP has the same approximation ratio with the solution of the deterministic problem. This provides us the flexibility to select any algorithm that solves

[1] The proof for this statement is provided in Lemma 19 of Chapter 6

the deterministic knapsack problem as a subroutine for CC-KNAP based on the application. For completeness, we provide a description of the algorithm for 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 feasible region defined by the resource constraint of RA-KNAP with λ_i, e.g., $\mathcal{R}_2$ in Fig. 7.2(a). The algorithm has two parts: (a) find a feasible solution of CC-KNAP by solving RA-KNAPs with increasing λ in (7.6) (line 1-6); (b) search the rest of feasible region of CC-KNAP by solving RA-KNAPs with methodically updated λ and W' (line 7-8). The procedures are shown in Algorithm 13 and Algorithm 14.

The first part starts with solving RA-KNAP with $\lambda_i = 0$, $W'_i = W_k$, where $i = 1$ is the index of the iteration. (line 1-2). 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. 7.2 (a), the algorithm computes the resource constraint of the RA-KNAP for the next iteration by updating risk-averse parameter from equation $\lambda_{i+1} = C/\sigma_i$ where $\sigma_i = \sqrt{\sum_{j=1}^{n_t} \sigma_{kj}^2 x_{kj}}$ (line 4). From a geometric point of view, the update 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. 7.2(a) (line 3-5). The feasible solution s_2 is stored in the solution set $\mathbb{S}$ (line 6).[2] 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-approximation). Note that this procedure terminates in finite number of iterations because λ_i increases as 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 tasks. At this moment, the first part of Algorithm 13 (line 3-6) terminates and $\mathbb{S}$ is an empty set.

[2] 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 the solution by abuse of notation.

We then need to find the optimal (α-approximation) solution of the rest of the feasible region by the second part of Algorithm 13, e.g., $\mathcal{R} \setminus \mathcal{R}_2$ in Fig. 7.2(a) (line 7-8). It computes the intersection point of parabola and the straight line of the last RA-KNAP's constraint, e.g., s_a in Fig. 7.2. Further it computes the point on parabola with horizontal coordinate $\sigma^2 = \sum_{j=1}^{n_t} \sigma_{kj}^2 x_{kj}$, e.g., s_b in Fig. 7.2 (line 7). Therefore the search region is now limited to the space embraced 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. 7.2(a)). Then the algorithm calls a recursive function $\mathcal{A}(s_a, s_b, \mathbb{S})$ presented in Algorithm 14 (line 9). Let (σ_a^2, μ_a) and (σ_b^2, μ_b) be the coordinate of s_a and s_b respectively. The recursive function $\mathcal{A}(s_a, s_b, \mathbb{S})$ computes the equation of the straight line through s_a and s_b. And then solve 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). If the obtained solution $s_{ab} = (\sigma_{ab}^2, \mu_{ab})$ does not satisfy the chance constraint(e.g., s_3 in Fig. 7.2(b)), algorithm computes 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 it computes 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 prevents the cycling (line 2-3). As shown in Fig. 7.2(b), the current search region is thus covered by union of $\mathcal{R}_4$ and $\mathcal{R}_5$. Then it calls $\mathcal{A}(s_a, s_c, \mathbb{S})$ and $\mathcal{A}(s_c, s_b, \mathbb{S})$. 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. Now s_{ab} dominates the corresponding intersection region, e.g, s_4 dominates $\mathcal{R}_4$ in Fig. 7.2(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 one with the highest objective function value (line 10 of Algorithm 13). The output is the indices of tasks assigned to r_k, denoted by $\bar{J}_k^*$.

Lemma 20. *If the solution of the RA-KNAP in* (7.6) *is α-approximation, the solution of CC-KNAP is also α-approximation.*

Proof. Since the approximation ratio of algorithm that solves RA-KNAP is α and $\mathcal{R}_i$ is

Algorithm 13 Algorithm to solve CC-KNAP for r_k

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

a subset of feasible region defined by the resource constraint of RA-KNAP in (7.6), 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 Algorithm 13 finishes, the union of $\mathcal{R}_i$ cover 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 $\mathbb{I}$ denotes 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

$$v^*_i = v(s^*),\ \forall i \in \mathbb{I} \tag{7.7}$$

Because $\bar{s}^*$ the solution of CC-KNAP obtained by Algorithm 13 is the one with the highest objective value in $\mathbb{S}$, it is clear that

$$v(\bar{s}^*) \geq v(s_i),\ \forall i \in \mathbb{S} \tag{7.8}$$

Therefore,

$$\alpha v(\bar{s}^*) \geq \alpha v(s_i) \geq v^*_i = v(s^*), \forall i \in \mathbb{I} \tag{7.9}$$

Since $\mathbb{I}$ is a non-empty set, the solution of CC-KNAP obtained by Algorithm 13 is α-approximation. □

Algorithm 14 Recursive function $\mathcal{A}(s_a, s_b, \mathbb{S})$

Input: $s_a, s_b, \mathbb{S}$.
Output: $\mathbb{S}$.
1: Solve RA-KNAP (7.6) 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 (7.5) **then**
3: Compute 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** $\mathbb{S} = \{\mathbb{S}, s_{ab}\}$.
7: **end if**

If RA-KNAP is solved optimally, Algorithm 13 provides the optimal solution of CC-KNAP.

7.3.2 Algorithm for CC-GAP

The algorithm that solves CC-GAP is presented in Algorithm 15. The iterative procedure is executed by single robot 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 Algorithm 13. 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 tasks say t_u in the current solution has been assigned to a robot with a smaller index say $r_b, (b < k)$, this task is removed from the previous set $\bar{J}_b^*$ (line 5-7). 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 8). 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 11-12).

Lemma 21. *(Feasibility) The assignments for all robots S obtained from Algorithm 15 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-

Algorithm 15 Algorithm to solve CC-GAP

Input: a_{ij}, W_i, μ_{ij}, σ_{ij}^2, p, $\forall i = 1, ..., n_r, j = 1, ..., n_t$.
Output: S.

```
Let initial payoff value ^k a_ij = a_ij, k = 1, ∀i = 1, ..., n_r, ∀j = 1, ..., n_t.
for robot r_1 to r_{n_r} do
    Solve CC-KNAP with payoffs value {^k a_kj}_{j=1}^{n_t} by Algorithm 13.
    Obtain solution J̄*_k = {j | x_kj = 1}.
    if J̄*_k ⋂ {J̄*_i}_{i=1}^{k-1} ≠ ∅ then
        Remove all t_u ∈ J̄*_k ⋂ {J̄*_i}_{i=1}^{k-1} from the original set.
    end if
    Update ^{k+1} a_ij = ^k a_ij − ^k a_kj, ∀i > k and ∀j ∈ J̄*_k (the payoffs for unassigned tasks
    remain the same).
    k = k + 1.
end for
Let Ĵ*_i = J̄*_i, ∀i = 1, ..., n_r.
return S = {Ĵ*_i}_{i=1}^{n_r}.
```

mean plane (see $\bar{S}_1$ in Fig. 7.1). During Algorithm 15, 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 $\hat{J}_k^*$ are smaller. The point of the final assignment must be on the southwest side of the initial assignment, e.g., $\hat{S}_1$ in Fig. 7.1. 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 22. *(Optimality) If the algorithm for CC-KNAP* (7.5) *is α-approximation, then the algorithm for CC-GAP is a $(1+\alpha)$-approximation.*

Lemma 22 is proved by induction. The base case is that the assignment of the last robot r_{n_r} in the final solution of Algorithm 15, 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 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 $(1+\alpha)$-approximation to the CC-GAP in which the payoffs are equal to

${}^{k}a_{ij}, \forall i = k, ..., n_r, \forall j = 1, ..., n_t$. If this is true, it implies

$$(1+\alpha)\sum_{i=1}^{n_r}\sum_{j\in\hat{J}_i^*} {}^{1}a_{ij} \geq \max\{\sum_{i=1}^{n_r}\sum_{j=1}^{n_t} {}^{1}a_{ij}x_{ij}\} \tag{7.10}$$

As shown in the first line of Algorithm 15, ${}^{1}a_{ij} = a_{ij}$. Therefore $\{\hat{J}_i^*\}_{i=1}^{n_r}$ is $(1+\alpha)$-approximation to the original CC-GAP in (7.2). The proof that $\{\hat{J}_i^*\}_{i=k}^{n_r}$ is $(1+\alpha)$ approximation is similar to the deterministic GAP. Interested readers are referred to [15] (deterministic GAP). We also provide a detailed proof of Lemma 22 in Appendix A.

7.4 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) (7.6) solved by r_i and T_i denotes the computational cost for solving knapsack problem. We use a pseudo-polynomial algorithm in [101] and the computational cost for solving knapsack is thus $O(n_i^2 A_i)$ 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., $A_i = \max_{j=1}^{n_i} {}^{i}a_{ij}$.

Since our algorithm solves CC-GAP by solving a sequence of deterministic knapsack problems, the key measures 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 numbers are problem parameter dependent and it is hard to provide *a priori* bounds. In this section, we will study these numbers with different numbers of robots and tasks. The extensive simulations show that (a) the algorithm to solve CC-GAP is scalable with the number of robots and tasks; (b) the algorithm solves chance-constrained knapsack problem by solving a small number of deterministic knapsack problems.

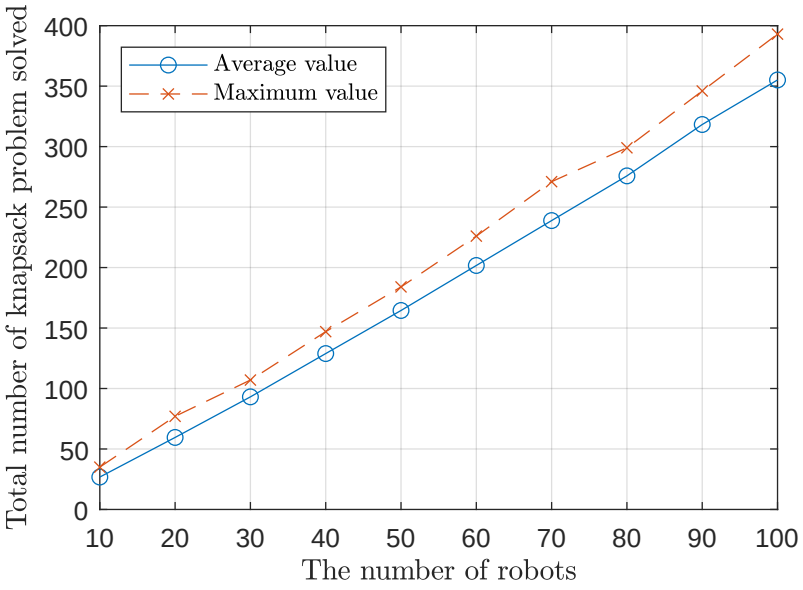
The motivating scenario that we consider for this simulation study is that of multi-robot parts delivery in an automated factory (presented in Fig. 1.1). A number of parts are located at a central store. A robot has to pick up a part from the central store and deliver it

to a specific workstation for assembly. Each robot obtains a 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 parts 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 capabilities), 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 99%. The simulations were done on a computer with Intel i7 2.6 GHz CPU and 16 GB RAM.

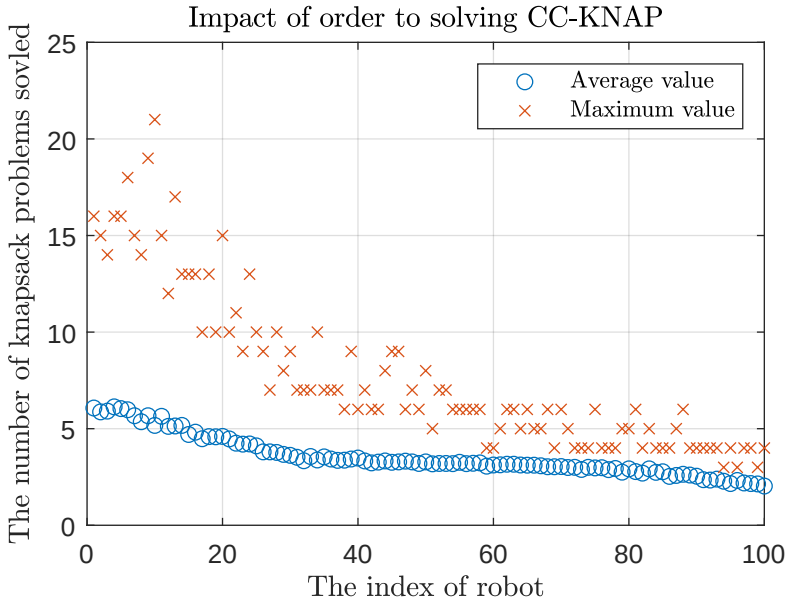
7.4.1 Scalability with the Number of Robots

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 an increment of 10. The number of tasks is always 4 times the number of robots. The results are presented in Fig.7.3a. For a fixed number of robots, we create 100 problem instances with randomly generated parameters and compute both average and the 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 average and maximum numbers grow almost linearly.

To study the influence of the order of robots solving chance-constrained knapsack problems, we present the average and the maximum number of knapsack problems solved by each robot (i.e., $K_i, \forall i$). The results are computed from same 100 instances with 100 robots used in Fig. 7.3a. The results are shown in Fig. 7.3b, in which the horizontal axis represents the index of each robot. The index reflects the order for robots to solve the chance-constrained knapsack problem, e.g., the robot with index 1 is the first robot to

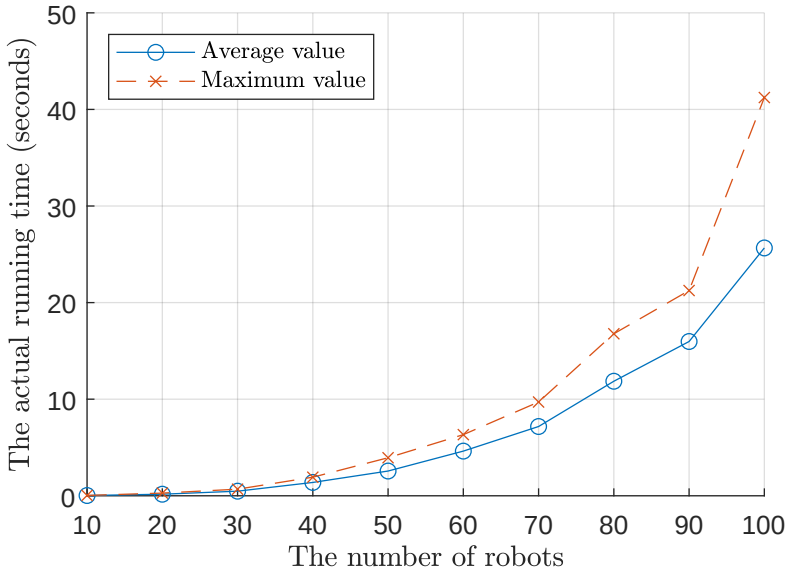


(a) The total number of deterministic knapsack problems solved by all robots.

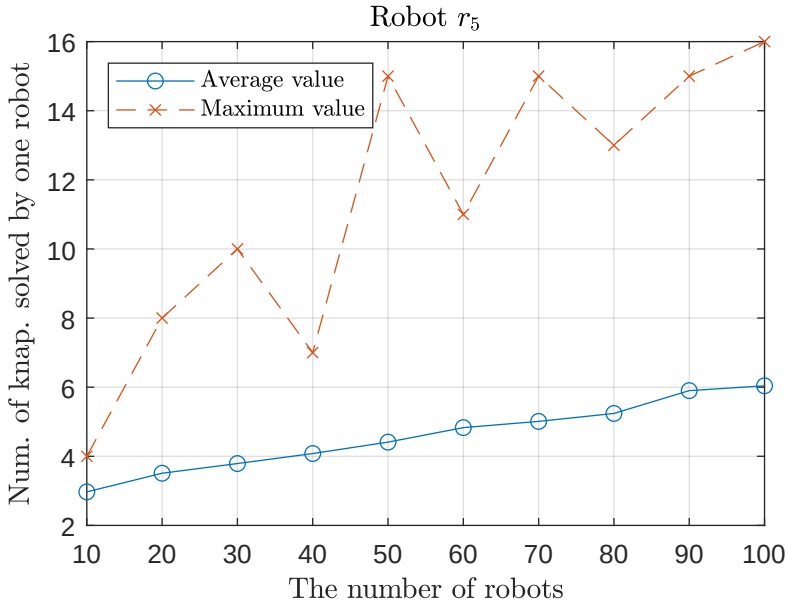


(b) The number of deterministic knapsack problem solved by each robot (with index from 1 to 100).

Figure 7.3: Scalability with the number of robots, part I.

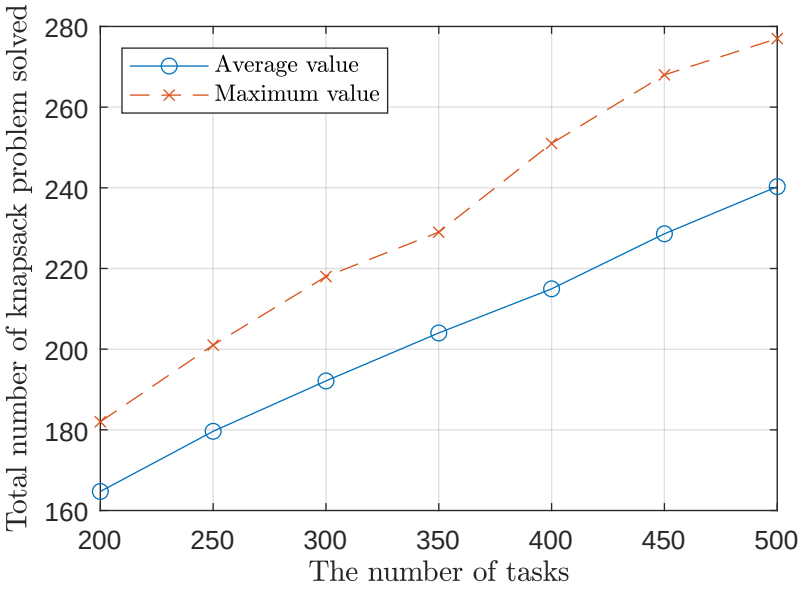


(a) The actual running time to solve CC-GAP (seconds).

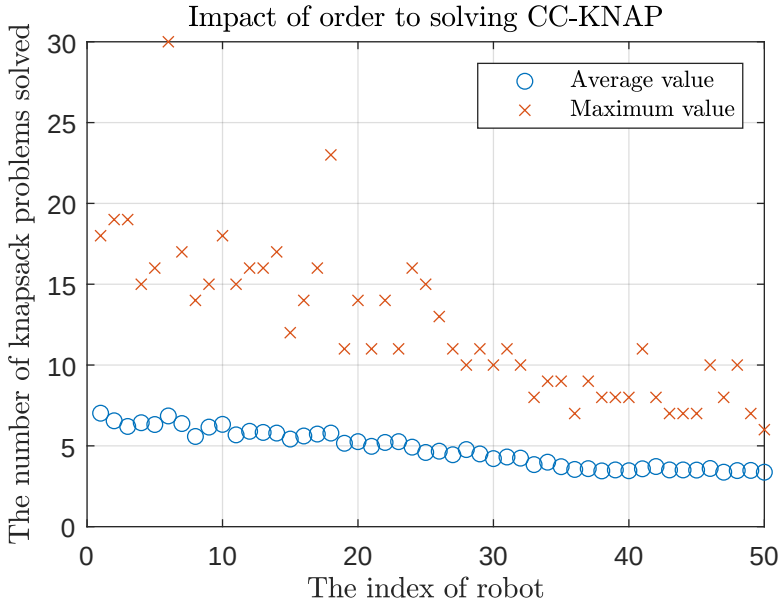


(b) The number of deterministic knapsack problem solved by one robot (r_5).

Figure 7.4: Scalability with the number of robots, part II.

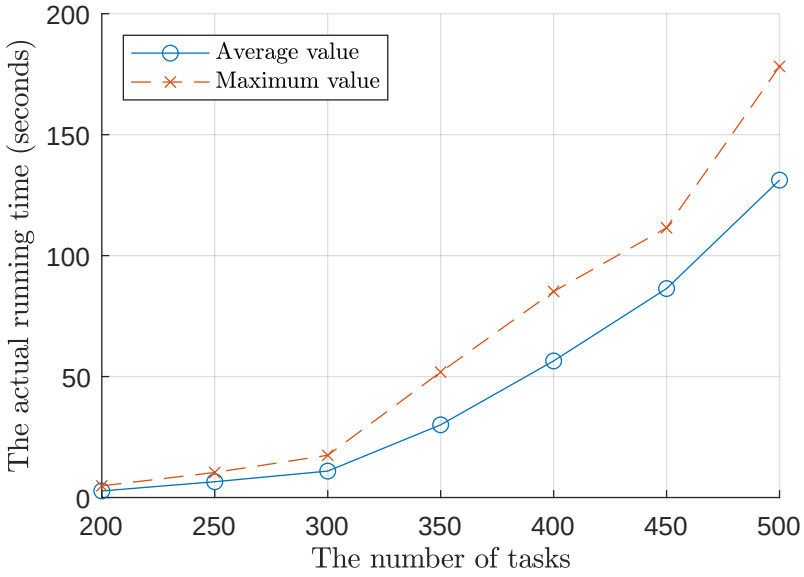


(a) The total number of deterministic knapsack problems solved by all robots..

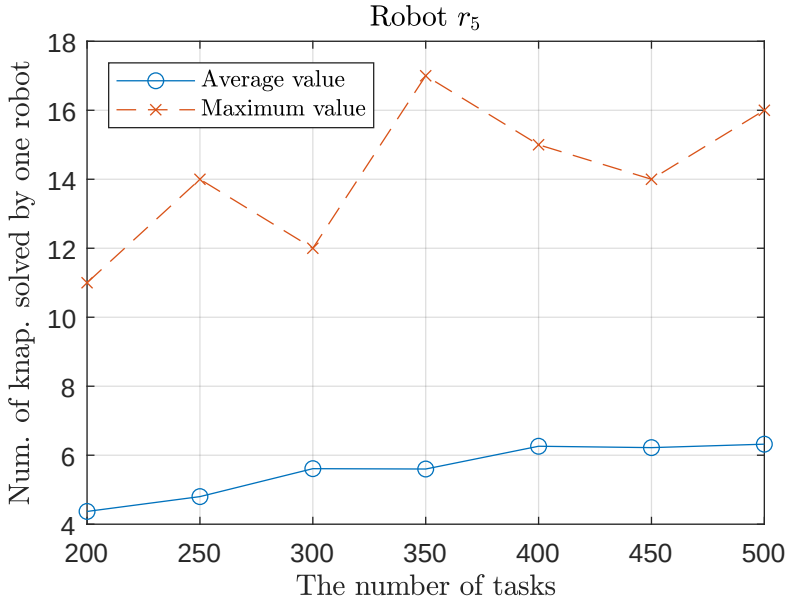


(b) The number of deterministic knapsack problem solved by each robot (with index from 1 to 50).

Figure 7.5: Scalability with the number of tasks, part I.



(a) The actual running time to solve CC-GAP (seconds).



(b) The number of deterministic knapsack problem solved by one robot (r_5).

Figure 7.6: Scalability with the number of tasks, part II.

solves CC-KNAP (7.5). It shows that the robot with a larger index obtains the solution by solving a smaller number of knapsack problems. The reason is that the chance-constrained knapsack problem for the robot with a larger index is simpler than 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. 7.3b 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 a particular index as the number of robots increases (i.e., $K_i, i \in \{1, ..., 10\}$ as n_r increases). The results for robot r_5 are presented in Fig. 7.4a. The average number of knapsack problems solved grows linearly in general and the maximum numbers are all less than 17. It implies that our algorithm for the chance-constrained knapsack problem is scalable with the number of robots, which agree with the results shown in Fig. 6.3a of Chapter 6. For reference, the average and maximum actual running time of the simulations are provided in Fig. 7.4b.

7.4.2 Scalability with the 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 varies from 200 to 500 with an increment 50. The results are presented in Fig. 7.5a. Similar to the first set of simulations, for a given number of tasks, the average and the maximum number of deterministic knapsack problems solved by all robots (i.e., $\sum_{i=1}^{n_r} K_i$) are computed by 100 randomly generated scenarios. Generally speaking, the average and the maximum numbers of knapsack problems solved increase linearly.

In Fig 7.5b, we present the average and the 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. 7.5a. The horizontal axis represents the index of the robot. It shows that the robot with a larger index tends to solve a 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 a particular index as the number of tasks increase (i.e., $K_i, i \in \{1, ...50\}$ as n_t increase). The results for robot r_5 is presented in Fig. 7.6a. It shows that the algorithm for the chance-constrained knapsack problem is scalable with the number of tasks. The actual running of the simulations is presented in Fig. 7.6b.

7.5 Conclusion

In this chapter, 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.

Chapter 8

Conclusion and Future Work

8.1 Thesis Summary

In this thesis, we study problems of multi-robot task allocation under uncertainty. The goal of this work is to design algorithms that compute solution with *a priori* probabilistic performance guarantee to various multi-robot task allocation with random parameters. We formulate these problems as chance-constrained optimization problems, which provide us the flexibility to tradeoff between performance and reliability. Since solving the chance-constrained problem directly is difficult, we study the connection between the chance-constrained problem and the risk-averse problem. We prove that the solution of the chance-constrained problem can be obtained by computing solutions to the risk-averse problem with the proper value of the parameters. The chance-constrained problem thus boils down to one or two-dimensional search on those parameters. By analyzing the problem on the variance-mean plane, we design methods to methodically search the parameters. We design algorithms to solve multi-robot task allocation with stochastic variables in the objective, which is first designed for solving CC-LAP and then extend to CC-STAP and CC-SP. We also design algorithms to solve multi-robot task allocation with stochastic variables in the resource constraints, which is first designed to solve CC-KNAP and extend to CC-GAP.

In CC-LAP, we present algorithms for computing solutions with a probabilistic team performance guarantee for the multi-robot task allocation with uncertain payoffs. We prove that the optimal solution of the chance-constrained problem can be found by repeatedly solving risk-averse problems using a one-dimensional search on the risk-averse parameter.

In CC-STAP, we consider chance-constrained simultaneous task assignment and path planning on a graph with stochastic edge costs. We prove that the chance-constrained problem can be solved optimally by solving a sequence of the deterministic version of the problems. We prove that the deterministic problem can be solved optimally by a linear assignment problem with the cost equal to the shortest path to the task location. We also present a distributed algorithm to solve CC-STAP based on the auction algorithm.

In CC-STAP-B, we present algorithms to solve CC-STAP with the MinMax objective. We prove that the CC-STAP-B is equivalent to a linear bottleneck assignment problem in which each assignment cost is equal to the chance-constrained shortest path cost for each robot-task pair. By leveraging the method for CC-LAP, we propose an algorithm to solve the chance-constrained shortest path problem.

In CC-KNAP, we consider a multi-robot teaming problem with the deterministic objective and stochastic resource constraints. It is proved that the solution of CC-KNAP can be obtained by solving a sequence of methodically generated deterministic knapsack problems. If all deterministic knapsack problems are solved optimally, the solution of CC-KNAP is also optimal.

In CC-GAP, we consider a generalized assignment problem with deterministic payoffs and stochastic consumption. By leveraging the method to solve CC-KNAP, we present an $(1 + \alpha)$-approximate algorithm where α is the approximation ratio for solving risk-averse knapsack problems. The best approximation ratio of our algorithm for CC-GAP is 2.

8.2 Future Work

Future work could be conducted in the following directions.

(1) We have designed algorithms for the multi-robot task allocation with stochastic variables in objective and stochastic variables in resource constraints separately. We will extend our algorithms to more general task allocation problems which contain both stochastic payoffs and stochastic resource constraints. One example is the

knapsack problem with the stochastic payoff and stochastic resource consumption. We consider a set of n tasks and one robot with resource capacity W. The payoffs and resource consumption of robots performing tasks are random variable a_i and w_i. We want to assign a subset of tasks to each robot and compute a performance certificate Y such that irrespective of the realization of the random variables, the total realized payoffs are greater than Y, and the resource capacity of the robot is not violated with a high probability respectively. The formulation is given as following.

$$\begin{aligned} \max \quad & Y \\ \text{s.t.} \quad & \mathbb{P}\left(\sum_{i=1}^{n} a_i x_i \geq Y\right) \geq p_a \\ & \mathbb{P}\left(\sum_{i=1}^{n} w_i x_i \leq W\right) \geq p_w \\ & x_i \in \{0,1\}, \quad \forall i = 1, \ldots, n \end{aligned} \tag{8.1}$$

where $p_a > 0.5$ is the predefined probability for payoffs and $p_w > 0.5$ is the predefined probability for resource consumption. This problem could be extended to other problems related to the knapsack, like generalized assignment problem and multiple-choice knapsack problem.

(2) In CC-STAP and CC-STAP-B, we assume that robots move on a graph that captures the collision-free configuration space of the robots and robots have local collision avoidance schemes to avoid mobile obstacles. A potential future direction is to evaluate or develop priority-based local collision avoidance schemes that can be applied in our algorithms.

(3) Although we demonstrated empirically that our algorithm is quite efficient, in the future, we would like to have a theoretical analysis of the computational complexity of our algorithm. In particular, we want to obtain a theoretical bound on the number of calls to the risk-averse problem for our algorithms.

Bibliography

[1] B. P. Gerkey and M. J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. International Journal of Robotics Research, 23(9):939–954, 2004.

[2] M.B. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. Proceedings of the IEEE, 94(7):1257–1270, jul. 2006.

[3] L. E. Kavraki, P. Svestka, Jean-Claude 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.

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

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

[6] Fan Yang and Nilanjan Chakraborty. Multirobot simultaneous path planning and task assignment on graphs with stochastic costs. In 2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), pages 86–88, 2019.

[7] Fan Yang and Nilanjan 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), pages 6661–6667, 2020.

[8] Fan Yang and Nilanjan Chakraborty. Algorithm for multi-robot chance-constrained generalized assignment problem with stochastic resource consumption. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.

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

[10] Fan Yang and Nilanjan 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.

[11] D. P. Bertsekas. The auction algorithm: A distributed relaxation method for the assignment problem. Annals of Operations Research, 14:105–123, 1988.

[12] M. M. Zavlanos, L. Spesivtsev, and G. J. Pappas. A distributed auction algorithm for the assignment problem. In Proc. 47th IEEE Conf. Decision and Control, pages 1212–1217, 2008.

[13] Evdokia Nikolova, Jonathan A. Kelner, Matthew Brand, and Michael Mitzenmacher. Stochastic Shortest Paths Via Quasi-convex Maximization, pages 552–563. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[14] Evdokia Nikolova. Approximation Algorithms for Reliable Stochastic Combinatorial Optimization, pages 338–351. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[15] Reuven Cohen, Liran Katzir, and Danny Raz. An efficient approximation for the generalized assignment problem. Information Processing Letters, 100:162–166, 2006.

[16] 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.

[17] C. Bererton, G. Gordon, S. Thrun, and P. Khosla. Auction mechanism design for multi-robot coordination. In NIPS, 2003.

[18] H.-L. Choi, L. Brunet, and J. How. Consensus-based decentralized auctions for robust task allocation. IEEE Transactions on Robotics, 25(4):912–926, 2009.

[19] Alejandro R. Mosteo and Luis Montano. A survey of multi-robot task allocation. Technical report, Instituto de Investigacin en Ingeniera de Aragn (I3A), 2010.

[20] A. Stentz and M. B. Dias. A free market architecture for coordinating multiple robots. Technical report, CMU Robotics Institute, 1999.

[21] M. B. Dias and A. Stentz. A free market architecture for distributed control of a multirobot system. In 6th International Conference on Intelligent Autonomous Systems (IAS-6), pages 115–122, July 2000.

[22] N. Kalra, D. Ferguson, and A. Stentz. Hoplites: A market-based framework for planned tight coordination in multirobot teams. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pages 1170–1177, April 2005.

[23] B. P. Gerkey and M. J. Mataric. Sold!: Auction methods for multirobot coordination. IEEE Transactions on Robotics, 18(5):758–768, October 2002.

[24] L.E. Parker. Alliance: an architecture for fault tolerant multirobot cooperation. IEEE Transactions on Robotics and Automation, 14(2):220–240, apr 1998.

[25] H. W. Kuhn. The Hungarian method for the assignment problem. Naval Research Logistics, 2(1-2):83–97, March 1955.

[26] R. Burkard, M. Dell'Amico, and S. Martello. Assignment Problems, chapter 4, pages 79–87. Society for Industrial and Applied Mathematics, 2009.

[27] D. P. Bertsekas. The auction algorithm for assignment and other network flow problems: A tutorial. Interfaces, 20(4):133–149, 1990.

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

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

[30] H. Alt, N. Blum, K. Mehlhorn, and M. Paul. Computing a maximum cardinality matching in a bipartite graph in time $\mathcal{O}(n^{1.5}\sqrt{m/\log n})$. Information Processing Letters, 37(4):237–240, February 1991.

[31] G. Carpaneto and P. Toth. Algorithm for the solution of the bottleneck assignment problem. Computing, 27(2):179–187, Jun 1981.

[32] O. Gross. The bottleneck assignment problem. Technical Report P-1630, The Rand Corporation, Santa Monica, CA, 1959.

[33] E.S. Page. A note on assignment problems. The Computer Journal, 6, 1963.

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

[35] David B. Shmoys and Chaitanya Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. J. ACM, 53(6):978–1012, November 2006.

[36] Anupam Gupta, Martin Pál, Ramamoorthi Ravi, and Amitabh Sinha. What about wednesday? approximation algorithms for multistage stochastic optimization. In Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, pages 86–98, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[37] Irit Katriel, Claire Kenyon-Mathieu, and Eli Upfal. Commitment under uncertainty: Two-stage stochastic matching problems. Theoretical Computer Science, 408(2):213–223, 2008. Excursions in Algorithmics: A Collection of Papers in Honor of Franco P. Preparata.

[38] Anupam Gupta, Martin Pál, R. Ravi, and Amitabh Sinha. Boosted sampling: Approximation algorithms for stochastic optimization. In Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing, STOC '04, pages 417–426, New York, NY, USA, 2004. ACM.

[39] Nicole Immorlica, David Karger, Maria Minkoff, and Vahab S. Mirrokni. On the costs and benefits of procrastination: Approximation algorithms for stochastic combinatorial optimization problems. In Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '04, pages 691–700, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[40] Maria Albareda-Sambola and Elena Fernández. The stochastic generalised assignment problem with bernoulli demands. Top, 8(2):165–190, Dec 2000.

[41] Maria Albareda-Sambola, Maarten H. van der Vlerk, and Elena Fernández. Exact solutions to a class of stochastic generalized assignment problems. European Journal of Operational Research, 173(2):465–487, 2006.

[42] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. The online stochastic generalized assignment problem. In Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, pages 11–25, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[43] R. Kevin Wood David R. Spoerl. A stochastic generalized assignment problem. Technical report, Department of Operations Research, Naval Postgraduate School Monterey, California 93943, USA, January 2004.

[44] Justin Boyan, Michael Mitzenmacher, and Michael 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.

[45] Dimitri P. Bertsekas. Dynamic Programming and Optimal Control, Vol. II. Athena Scientific, 3rd edition, 2007.

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

[47] Christos H. Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. Theoretical Computer Science, 84(1):127–150, 1991.

[48] George H. Polychronopoulos and John N. Tsitsiklis. Stochastic shortest path problems with recourse. NETWORKS, 27:133–143, 1996.

[49] John Mote, Ishwar Murthy, and David L. Olson. A parametric approach to solving bicriterion shortest path problems. European Journal of Operational Research, 53(1):81–92, 1991.

[50] Stefano Pallottino and Maria Grazia 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.

[51] D. Bertsimas, D. Brown, and C. Caramanis. Theory and applications of robust optimization. SIAM Review, 53(3):464–501, 2011.

[52] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczynski. Lectures on Stochastic Programming: Modeling and Theory, Second Edition. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2014.

[53] E. Erdoğan and G. Iyengar. On two-stage convex chance constrained problems. Mathematical Methods of Operations Research, 65(1):115–140, Feb 2007.

[54] A. Nemirovski and A. Shapiro. Convex approximations of chance constrained programs. SIAM Journal on Optimization, 17(4):969–996, 2006.

[55] Hiroaki Ishii, Shōgo Shiode, Toshio Nishida, and Yoshikazu Namasuya. Stochastic spanning tree problem. Discrete Applied Mathematics, 3(4):263–273, 1981.

[56] Vineet Goyal and R. Ravi. A ptas for the chance-constrained knapsack problem with random item sizes. Operations Research Letters, 38(3):161–164, 2010.

[57] Jon Kleinberg, Yuval Rabani, and Éva Tardos. Allocating bandwidth for bursty connections. In Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing, STOC '97, pages 664–673, New York, NY, USA, 1997. ACM.

[58] A. Goel and P. Indyk. Stochastic load balancing and related problems. In 40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039), pages 579–586, Oct 1999.

[59] Sameera S. Ponda, Luke B. Johnson, and Jonathan P. How. Distributed chance-constrained task allocation for autonomous multi-agent teams. In American Control Conference (ACC), June 2012.

[60] Sameera S. Ponda, Luke B. Johnson, and Jonathan P. How. Risk allocation strategies for distributed chance-constrained task allocation. In American Control Conference (ACC), June 2013.

[61] Sejoon Lim, Christian Sommer, Evdokia Nikolova, and Daniela Rus. Practical route planning under delay uncertainty: Stochastic shortest path queries. In Proceedings of Robotics: Science and Systems, Sydney, Australia, July 2012.

[62] Lantao Liu and Dylan A. Shell. An anytime assignment algorithm: From local task swapping to global optimality. Autonomous Robots, 35(4):271–286, Nov 2013.

[63] M. Alighanbari and J. P. How. A robust approach to the UAV task assignment problem. International Journal of Robust and Nonlinear Control, 18(2):118–134, January 2008.

[64] L. Liu and D. A. Shell. Assessing optimal assignment under uncertainty: an interval-based algorithm. International Journal of Robotics Research, 30:936–953, 2011.

[65] 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.

[66] Peter Kall and Stein W. Wallace. Stochastic Programming. John Wiley & Sons, 1994.

[67] T.J. Sargent. Macroeconomic Theory. Academic Press, 1979.

[68] Stephen Boyd and Lieven Vandenberghe. Convex Optimization. Cambridge University Press, New York, NY, USA, 2004.

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

[70] Sejoon Lim, Hari Balakrishnan, David Gifford, Samuel Madden, and Daniela Rus. Stochastic motion planning and applications to traffic. The International Journal of Robotics Research, 30(6):699–712, 2011.

[71] Sejoon Lim, Christian Sommer, Evdokia Nikolova, and Daniela 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.

[72] Sejoon Lim and Daniela 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.

[73] Matthew Turpin, Nathan Michael, and Vijay Kumar. Trajectory planning and assignment in multirobot systems. In Algorithmic Foundations of Robotics X, pages 175–190, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[74] Matthew Turpin, Nathan Michael, and Vijay Kumar. Capt: Concurrent assignment and planning of trajectories for multiple robots. The International Journal of Robotics Research, 33(1):98–112, 2014.

[75] Dimitra Panagou, Matthew Turpin, and Vijay 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.

[76] Edsger W. Dijkstra. A note on two problems in connexion with graphs. Numerische mathematik, 1(1):269–271, 1959.

[77] Richard Bellman. On a routing problem. Quarterly of Applied Mathematics, 16:87–90, 1958.

[78] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. ACM Trans. Math. Softw., 22(4):469–483, December 1996.

[79] Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. Assignment Problems. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009.

[80] Lingzhi Luo, Nilanjan Chakraborty, and Katia P. Sycara. Distributed algorithm design for multi-robot generalized task assignment problem. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013, pages 4765–4771, 2013.

[81] Hans Kellerer, Ulrich Pferschy, and David Pisinger. Knapsack problems. Springer, 2011.

[82] Vijay V. Vazirani. Approximation Algorithms. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[83] Silvano Martello and Paolo Toth. Knapsack Problems: Algorithms and Computer Implementations. John Wiley & Sons, Inc., New York, NY, USA, 1990.

[84] Vincent Poirriez, Nicola Yanev, and Rumen Andonov. A hybrid algorithm for the unbounded knapsack problem. Discrete Optimization, 6(1):110 – 124, 2009.

[85] Silvano Martello, David Pisinger, and Paolo Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. Manage. Sci., 45(3):414–424, March 1999.

[86] Silvano Martello and Paolo Toth. A mixture of dynamic programming and branch-and-bound for the subset-sum problem. Management Science, 30(6):765–771, 1984.

[87] Erick Delage and Shie Mannor. Percentile optimization for markov decision processes with parameter uncertainty. Operations Research, 58(1):203–213, 2010.

[88] Robert L. Carraway, Robert L. Schmidt, and Lawrence R. Weatherford. An algorithm for maximizing target achievement in the stochastic knapsack problem with normal returns. Naval Research Logistics (NRL), 40(2):161–173, 1993.

[89] Mordechai I. Henig. Risk criteria in a stochastic knapsack problem. Operations Research, 38(5):820–825, 1990.

[90] Moshe Sniedovich. Preference order stochastic knapsack problems: Methodological issues. The Journal of the Operational Research Society, 31(11):1025–1032, 1980.

[91] Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. Mathematics of Operations Research, 33(4):945–964, 2008.

[92] Olivier Klopfenstein and Dritan Nace. A robust approach to the chance-constrained knapsack problem. Operations Research Letters, 36(5):628 – 632, 2008.

[93] Galia Shabtai, Danny Raz, and Yuval Shavitt. A relaxed FPTAS for chance-constrained knapsack. In 29th International Symposium on Algorithms and Computation, ISAAC, volume 123 of LIPIcs, pages 72:1–72:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[94] Lingzhi Luo, Nilanjan Chakraborty, and Katia 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.

[95] G. Ross and Richard Soland. A branch and bound algorithm for the generalized assignment problem. Mathematical Programming, 8:91–103, 12 1975.

[96] Martin Savelsbergh. A branch-and-price algorithm for the generalized assignment problem. Operations Research, 45:831–841, 12 1997.

[97] David Shmoys and Éva Tardos. Approximation algorithm for the generalized assignment problem. Math. Program., 62:461–474, 02 1993.

[98] Chandra Chekuri and Sanjeev Khanna. A ptas for the multiple knapsack problem. Proc. of SODA, page 213–222, 09 2001.

[99] Lisa Fleischer, Michel X. Goemans, Vahab S. Mirrokni, and Maxim Sviridenko. Tight approximation algorithms for maximum general assignment problems. In Proc. of ACM-SIAM SODA, pages 611–620, 2006.

[100] Berkin Toktas, Joyce W. Yen, and Zelda B. Zabinsky. Addressing capacity uncertainty in resource-constrained assignment problems. Computers and Operations Research, 33(3):724–745, March 2006.

[101] Katherine Lai and Michel Goemans. The knapsack problem and fully polynomial time approximation schemes. Technical report, Massachusetts Institute of Technology, Department of Mathematics, 3 2006.

Appendix A

Proof of Lemma 22

Definition 3. *The inductive CC-GAP(k) is the chance-constrained generalized assignment problem which involves robots from r_k to r_{n_r} and the payoffs equal to ${}^k a_{ij}$ (the payoffs for CC-KNAP in Algorithm 15). The formulation is presented in* (A.1).

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

It is clear that the original CC-GAP in (7.2) is equivalent to the inductive CC-GAP(1).

Lemma 22. *(Optimality) If the algorithm for CC-KNAP* (7.5) *is α-approximation, then the algorithm for CC-GAP is a $(1+\alpha)$-approximation.*

Proof. Remind that $\bar{J}_i^*$ denotes the assignment (tasks) obtained by solving CC-KNAP for r_i at iteration i in Algorithm 15; ${}^i a_{ij}$ is the payoff of CC-KNAP; $\hat{J}_i^*$ denotes assignment of r_i at the end of the algorithm (some of tasks in $\bar{J}_i^*$ might be reassigned to other robots with higher index).

Since the solution of CC-KNAP for the last robot $\bar{J}_{n_r}^*$ is α-approximation and $\bar{J}_{n_r}^* = \hat{J}_{n_r}^*$. The solution is also α-approximation to the optimal solution of the inductive CC-GAP(n_r).

Obviously, it is true that

$$(1+\alpha)\sum_{i=n_r}^{n_r}\sum_{j\in\hat{J}_i^*}{}^{n_r}a_{ij} \geq \max\{\sum_{i=n_r}^{n_r}\sum_{j=1}^{n_t}{}^{n_r}a_{ij}x_{ij}\} \tag{A.2}$$

To prove by induction, we need to prove that the assignments for r_k to r_{n_r} is $(1+\alpha)$-approximation to the optimal solution of the inductive CC-GAP(k), i.e.,

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

assuming it is true for $\{\hat{J}_i^*\}_{i=k+1}^{n_r}$ to inductive CC-GAP($k+1$).

Now we introduce variables ${}^{k}b_{ij}$ and ${}^{k}c_{ij}, \forall i = k, ..., n_r, \forall j = 1, .., n_t$ which are defined in equation (A.3) and (A.4) respectively. The nice properties are ${}^{k}a_{ij} = {}^{k}b_{ij} + {}^{k}c_{ij}$ and ${}^{k}c_{ij} = {}^{k+1}a_{ij}, \forall i > k$.

$${}^{k}b_{ij} = \begin{cases} {}^{k}a_{kj}, & i = k \text{ or } j \in \bar{J}_k^*, \\ 0 & \text{otherwise} \end{cases} \tag{A.3}$$

$${}^{k}c_{ij} = \begin{cases} {}^{k}a_{ij} - {}^{k}a_{kj}, & i > k \text{ and } j \in \bar{J}_k^*, \\ 0, & i = k, \\ {}^{k}a_{ij} & \text{otherwise} \end{cases} \tag{A.4}$$

Since $\bar{J}_k^*$ is the α-approximation solution of CC-KNAP of r_k and ${}^{k}b_{kj} = {}^{k}a_{kj}$, it is true that

$$\alpha\sum_{j\in\bar{J}_k^*}{}^{k}b_{kj} \geq \max\{\sum_{j=1}^{n_t}{}^{k}b_{kj}x_{kj}\} \tag{A.5}$$

Eq. (A.3) implies that

$$\max\{\sum_{i=k+1}^{n_r}\sum_{j=1}^{n_t}{}^{k}b_{ij}x_{ij}\} = \sum_{j\in\bar{J}_k^*}{}^{k}b_{kj} \tag{A.6}$$

Eq. (A.5) and (A.6) imply that

$$(1+\alpha)\sum_{j\in\bar{J}_k^*}{}^{k}b_{ij} \geq \max\{\sum_{j=1}^{n_t}{}^{k}b_{kj}x_{kj}\} + \max\{\sum_{i=k+1}^{n_r}\sum_{j=1}^{n_t}{}^{k}b_{ij}x_{ij}\} \geq \max\{\sum_{i=k}^{n_r}\sum_{j=1}^{n_t}{}^{k}b_{ij}x_{ij}\} \tag{A.7}$$

Further because $\{\hat{J}_i^*\}_{i=k}^{n_r}$ includes all tasks in $\bar{J}_k^*$ and ${}^k b_{ij} = 0$, when $i > k+1$ and $j \notin \bar{J}_k^*$, it is true that $\sum_{i=k}^{n_r} \sum_{j \in \hat{J}_i^*} {}^k b_{ij} = \sum_{j \in \bar{J}_k^*} {}^k b_{ij}$. Then we have the following inequality

$$(1+\alpha) \sum_{i=k}^{n_r} \sum_{j \in \hat{J}_i^*} {}^k b_{ij} \geq \max\{\sum_{i=k}^{n_r} \sum_{j=1}^{n_t} {}^k b_{ij} x_{ij}\} \tag{A.8}$$

Eq. (A.4) indicates that ${}^k c_{kj} = 0, \forall j$ and ${}^k c_{ij} = {}^{k+1} a_{ij}, \forall i > k$. It is clear that

$$\sum_{i=k}^{n_r} \sum_{j \in \hat{J}_i^*} {}^k c_{ij} = \sum_{i=k+1}^{n_r} \sum_{j \in \hat{J}_i^*} {}^{k+1} a_{ij}$$

and

$$\sum_{i=k}^{n_r} \sum_{j=1}^{n_t} {}^k c_{ij} x_{ij} = \sum_{i=k+1}^{n_r} \sum_{j=1}^{n_t} {}^{k+1} a_{ij} x_{ij}$$

By the assumption that $\{\hat{J}_i^*\}_{i=k+1}^{n_r}$ is $(1+\alpha)$-approximation to inductive CC-GAP$(k+1)$, i.e., $(1+\alpha) \sum_{i=k+1}^{n_r} \sum_{j \in \hat{J}_i^*} {}^{k+1} a_{ij} \geq \max\{\sum_{i=k+1}^{n_r} \sum_{j=1}^{n_t} {}^{k+1} a_{ij} x_{ij}\}$. We can conclude that

$$(1+\alpha) \sum_{i=k}^{n_r} \sum_{j \in \hat{J}_i^*} {}^k c_{ij} \geq \max\{\sum_{i=k}^{n_r} \sum_{j=1}^{n_t} {}^k c_{ij} x_{ij}\} \tag{A.9}$$

Because ${}^k a_{ij} = {}^k b_{ij} + {}^k c_{ij}$ and combine Eq. (A.9) with (A.8), it is clear that the assignments for r_k to r_{n_r} is $(1+\alpha)$-approximation to the optimal solution of the inductive CC-GAP(k), i.e.,

$$(1+\alpha) \sum_{i=k}^{n_r} \sum_{j \in \hat{J}_i^*} {}^k a_{ij} \geq \max\{\sum_{i=k}^{n_r} \sum_{j=1}^{n_t} {}^k a_{ij} x_{ij}\} \tag{A.10}$$

Therefore, $\{\hat{J}_i^*\}_{i=1}^{n_r}$ is $(1+\alpha)$-approximation to the inductive CC-GAP(1), which is equivalent to the original CC-GAP in (7.2). □