A Stochastic Programming Approach to Resource-Constrained Assignment Problems

Berkin Toktas[†] Joyce W. Yen[‡] Zelda B. Zabinsky[†]

12 February 2004

Abstract

We address the resource-constrained generalizations of the assignment problem with uncertain resource capacities, where the resource capacities have an unknown distribution that can be sampled. We propose three stochastic programming-based formulations that can be used to solve this problem, and provide exact and approximate solution techniques for the resulting models. We also present numerical results for a large set of numerical problems. The results indicate that the solutions obtained using the stochastic programming approaches perform significantly better than those obtained using expected values of capacities in a deterministic solution strategy. In addition, stochastic-programming-based approximations are computationally as efficient as deterministic techniques.

1 Introduction

Many real-life problems involving the assignment of a set of tasks to a set of agents are constrained by capacities of one or more resources that are consumed by these assignments. Due to this fact, generalizations of the classical assignment problem concerning resource constraints have been studied extensively in the literature. The majority of these studies assume deterministic conditions. However, the real-life applications of these problems often encounter uncertainty in different problem parameters, such as the assignment costs or the amount of resources needed for accomplishing tasks.

One major source of uncertainty in resource-constrained assignment problems is the uncertainty in resource capacities. In this study, we utilize stochastic programming methods to model capacity uncertainty in resource-constrained assignment problems, and present exact and approximate techniques to solve the resulting problems.

The rest of this article is organized as follows. In Section 2, we review resource-constrained generalizations of the assignment problem, and discuss possible sources of uncertainty that may affect these problems in Section 3. In Section 4, we identify three stochastic programming formulations to model resource-constrained assignment problems with capacity uncertainty. We present exact and approximate techniques to solve the resulting formulations in Section 5, and present the performance of these techniques on a large test set in Section 6. We conclude the paper with a brief summary and a discussion on future research directions in Section 7.

2 Background and Literature Survey

In the absence of resource constraints, the problem of finding the minimum-cost assignment of tasks to agents is called the classical assignment problem, which has been widely studied in the literature (see Kennington and Wang [15] for a survey of algorithms). The assignment problem is also of computational importance (Kuhn [16], Srinivasan and Thompson [24], Glover, Karney and Klingman [10], and others). The assignment problem itself has many applications such as facility location, personnel scheduling, and task assignment (see for example, Winston [27]).

[†]Industrial Engineering, University of Washington, Box 352650, Seattle, WA 98195
[‡]ADVANCE Center for Institutional Change, University of Washington, Box 352180, Seattle, WA 98195

To address real-life assignment decisions that are constrained by capacitated resources, resource-constrained generalizations of the assignment problem have also been studied extensively in the literature. There are two well-known resource-constrained generalizations of the assignment problem: the generalized assignment problem (GAP) (Ross and Soland [20]) and the assignment problem with side constraints (APSC) (Mazzola and Neebe [17]). Toktas, Yen and Zabinsky [25] define the assignment problem with individual capacities (APIC) and the collectively capacitated GAP (CCGAP) as other possible generalizations.

2.1 Generalized Assignment Problem (GAP)

The GAP considers the assignment of a given set of tasks, $\mathcal{I}$, to a set of agents, $\mathcal{J}$, in the presence of a single resource that is individually capacitated for each agent. First formally defined by Ross and Soland [20], the GAP has many applications such as resource scheduling (Zimokha and Rubinstein [28]), fixed-charge plant location (Ross and Soland [21]), and routing (Fisher and Jaikumar [5]). A binary programming formulation of the GAP can be given as follows:

$$\text{(GAP) Minimize} \quad \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}X_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{i\in\mathcal{I}} d_{ij}X_{ij} \le b_j, \quad j\in\mathcal{J}, \tag{2}$$

$$\sum_{j\in\mathcal{J}} X_{ij} = 1, \quad i\in\mathcal{I}, \tag{3}$$

$$X_{ij} = 0 \text{ or } 1, \quad i\in\mathcal{I}, \quad j\in\mathcal{J}, \tag{4}$$

where c_{ij} is the cost of assigning task i to agent j, d_{ij} is the capacity usage when task i is assigned to agent j, and b_j is the capacity of the single resource available to agent j. The binary variable X_{ij} takes the value 1 when task i is assigned to agent j and 0 otherwise. Equation (2) constrains the task-agent assignments such that the resource capacities are not exceeded. Equation (3) assures that each task is assigned to a single agent.

The GAP is NP-Hard (Fisher, Jaikumar and Van Wassenhowe [6]), and a considerable amount of research has been done to identify algorithms to solve GAP instances of reasonable size to optimality. The majority of these algorithms are based on Lagrangian relaxation (Fisher, Jaikumar and Van Wassenhove [6], Guignard and Rosenwein [11]), linear relaxation (Benders and van Nunen [1]), constraint deletion (Ross and Soland [20]) and decomposition (Jörnsten and Näsberg [14], Haddadi [12]). The reader is referred to the survey by Cattryse and Van Wassenhove [4] for an extensive review of such algorithms.

Gavish and Pirkul [7] extend the GAP to address multiple resources in the multi-resource generalized assignment problem (MRGAP), which has been used to model many multi-resource applications, such as database location in distributed computer systems (Pirkul [19]) and truck routing (Murphy [18]).

In the MRGAP, the agents consume multiple resources from a set $\mathcal{R}$ when accomplishing their assigned task. This extension replaces the constraint set (2) in the GAP formulation by

$$\sum_{i\in\mathcal{I}} a_{ijr}X_{ij} \le b'_{jr}, \quad j\in\mathcal{J}, \quad r\in\mathcal{R}, \tag{5}$$

where a_{ijr} represents the capacity usage of resource r when task i is assigned to agent j and b'_{jr} is the available capacity of resource r to agent j. In a later-dated study, Gavish and Pirkul [8] present a variety of effective solution procedures for the MRGAP based on its Lagrangian relaxation.

2.2 Assignment Problem with Side Constraints (APSC)

The MRGAP introduces an important extension to the GAP by allowing the modeling of multiple resource consumption. However, the need for an additional variation of this model arises if the multiple resources in the real system are capacitated not individually for each agent, but collectively for all agents in $\mathcal{J}$. The second well-known generalization of the assignment problem, the assignment problem with side-constraints

(APSC), addresses this need using the following binary programming formulation:

$$\begin{aligned}
\text{(APSC) Minimize} \quad & (1) \\
\text{subject to} \quad & \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}X_{ij} \le k_r, \quad r \in \mathcal{R}, && (6)\\
& \sum_{i\in\mathcal{I}} X_{ij} = 1, \quad j \in \mathcal{J}, && (7)\\
& (3) \text{ and } (4),
\end{aligned}$$

where k_r is the total available capacity of resource r to the agents in $\mathcal{J}$.

In addition to the structure in which the resources are capacitated, the second major difference between the formal definitions of the GAP, MRGAP and APSC is the fact that APSC requires a one-to-one matching of tasks to agents (hence $|\mathcal{I}| = |\mathcal{J}|$) whereas GAP and MRGAP do not.

Mazzola and Neebe [17] propose a branch-and-bound algorithm that utilizes subgradient optimization as a bounding strategy for solving the APSC to optimality. They also present an effective subgradient-based heuristic.

2.3 Other Generalizations

Toktas, Yen and Zabinsky [25] define two additional problems to complete the spectrum of different generalizations of the assignment problem between the GAP and the APSC. The first of these problems, called the collectively capacitated generalized assignment problem (CCGAP), allows for one-to-many matching between tasks and agents and incorporates collectively capacitated resources. The binary programming formulation of the CCGAP is as follows:

$$\begin{aligned}
\text{(CCGAP) Minimize} \quad & (1) \\
\text{subject to} \quad & (3),\ (4) \text{ and } (6).
\end{aligned}$$

A real-life application of this variation is the schedule recovery problem in air traffic management (see Berge, Hopperstad and Haraldsdottir [2]). The CCGAP can also be applied to other real-life applications of the GAP when the resources are collective. For instance, in resource scheduling, budget and equipment may be collectively constrained for all agents.

The second additional generalization allows for individually capacitated resources (as in the GAP), but address one-to-one matching of tasks to agents (as in the APSC). This assignment problem with individual capacities (APIC) is formulated using:

$$\begin{aligned}
\text{(APIC) Minimize} \quad & (1) \\
\text{subject to} \quad & (2),\ (3),\ (4) \text{ and } (7).
\end{aligned}$$

We summarize the possible types of resource-constrained assignment problems in Table 1.

3 Uncertainty in Resource Constrained Assignment

Most of the literature that studies resource-constrained assignment assumes that the problem parameters (such as assignment costs, resource usage vectors, and resource capacities) are perfectly known. These deterministic studies can be used to formulate problems where the conditions are known with high accuracy. However, for effective modelling of applications where conditions are not known perfectly, uncertainty should not be overlooked, as the solutions obtained using the aforementioned formulations can be highly sensitive to the problem parameters.

We can list different sources of uncertainty that may affect resource-constrained assignment problems. One is when the actual amount of resources needed by different agents to process the tasks (d_{ij}, a_{ijr}) is not known in advance. Similarly, the assignment costs (c_{ij}) might not be perfectly known. Another source of uncertainty is when the presence or absence of individual tasks or agents (sets $\mathcal{I}$ and $\mathcal{J}$) is not known for certain. The last possible source is the uncertainty in resource capacities (b_j, b'_{jr}, k_r).

Table 1: Classification of Resource-Constrained Assignment Problems

	Task-Agent Matching	
Resources	*One-to-One* $\sum_{j\in\mathcal{J}} X_{ij} = 1, \quad i \in \mathcal{I}$ $\sum_{i\in\mathcal{I}} X_{ij} = 1, \quad j \in \mathcal{J}$	*One-to-Many* $\sum_{j\in\mathcal{J}} X_{ij} = 1, \quad i \in \mathcal{I}$
Individually Capacitated $\sum_{i\in\mathcal{I}} a_{ijr} X_{ij} \leq b'_{jr}$, $j \in \mathcal{J}, r \in \mathcal{R}$	Assignment Problem with Individual Capacities (APIC)	Generalized Assignment Problem (GAP) Multi-Resource GAP (MRGAP)
Collectively Capacitated $\sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} a_{ijr} X_{ij} \leq k_r$, $r \in \mathcal{R}$	Assignment Problem with Side Constraints (APSC)	Collectively Capacitated GAP (CCGAP)

A somewhat straightforward (and common) approach to address parameter uncertainty is to 'plug in' the average values of the stochastic parameters, and solve the resulting deterministic formulation of the problem. Due to its simplicity, this approach may be suitable for certain cases. However, this approximation scheme may result in solutions that are not robust under the range of actual parameter values.

Another approach to address uncertainty in resource-constrained assignment is to implement stochastic-programming based approaches (see Birge and Louveaux [3] for an introductory text). The stochastic programming-based approaches fully incorporate, in the formulation itself, the relative performance of any solution under all possible realizations of uncertain parameters. Therefore, given a good understanding of the uncertainty structure, the stochastic programming solution to a resource-constrained assignment problem would be optimal in the expected sense.

There exist two studies in the literature (Albareda-Sambola, van der Vlerk and Aréizaga [22], Spoerl and Wood [23]) that consider uncertainty in resource-constrained assignment problems. Both of these studies use stochastic programming to address parameter uncertainty in the GAP. Albareda-Sambola, van der Vlerk and Aréizaga [22] consider the GAP where only a random subset of the given set of tasks are required to be actually processed, according to independent Bernoulli distributions. They assume that the assignment of each task to an agent is decided a priori, and once the actual set of tasks ($\mathcal{I}$) are known, reassignments can be performed if there are overloaded agents. They construct a convex approximation of the objective function, the minimal expected cost of assignments, and present an algorithm to solve the resulting problem. Spoerl and Wood [23] also consider the GAP, but address uncertainty in the amount of resources (d_{ij}) used by the task-agent assignments. They consider normally distributed resource usage parameters, and study a stochastic programming formulation in which excess capacity usage is penalized under actual conditions.

In this study, we consider capacity uncertainty in resource-constrained assignment problems. Hence, the resource capacities are no longer deterministic values, but random variables. We denote these random variables by $\tilde{b}$ for GAP and APIC, $\tilde{b}'$ for MRGAP, and $\tilde{k}$ for APSC and CCGAP.

Unlike [22] and [23], we do not assume that the probability distribution that governs the capacities is known. Hence, the stochastic programming formulations or our solution methodologies presented in the following sections are independent of the governing probability distribution. We do assume that it is possible to sample resource capacities from the underlying probability distribution. This allows for the implementation of our techniques to address applications where there is limited information about the uncertainty structure.

In this stochastic setting, we have to make a set of assignment decisions (X) without full information on the resource capacities. X denotes the first-stage decisions. Later, when full information about capacities becomes available, corrective actions (Y) can be taken to achieve feasibility under the actual set of resource capacities (b, b' or k). These corrective actions are our second-stage decisions.

The deterministic formulations introduced in the previous section no longer remain valid, as the right hand sides of Equations (2), (5) and (6) are stochastic. We can, however, introduce generic two-stage stochastic programming formulations of GAP, MRGAP, APSC, CCGAP and APIC.

A generic stochastic programming formulation for the generalizations with one-to-many matching (i.e. GAP, MRGAP and CCGAP) can be defined as:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}X_{ij} + EQ(X) \\ \text{subject to} \quad & (3) \text{ and } (4). \end{aligned} \tag{8}$$

The expected second-stage value function $EQ(X)$ is given by

$$EQ(X) = E_{\Omega}[Q(X,\omega)], \tag{9}$$

where $\omega \in \Omega$ denote the set of possible realizations of the stochastic parameters ($\tilde{b}$ for GAP, $\tilde{b'}$ for MRGAP and $\tilde{k}$ for CCGAP). For a given realization ω,

$$Q(X,\omega) = min_Y\{qY \,|\, Y \in \Upsilon(X,\omega)\} \tag{10}$$

is defined as the second-stage value function, thus incorporating the randomness of resource capacities to the objective function of the stochastic program. The vector Y denotes the set of second-stage variables, while q denotes their cost coefficients in the second-stage value function. The set Υ governs the relationship between the first-stage and second-stage decisions under the given set of resource capacities.

While the first-stage decisions are the assignment of tasks to agents, the definition of the second-stage decisions depend on the formulation of Equation (10) and, specifically, $\Upsilon(X,\omega)$. Alternative formulations for $Q(X,\omega)$ will be discussed in detail in the next section. The stochastic programming formulations for generalizations with one-to-one matching can be defined in the same manner, by adding Equation (7) in the set of constraints.

Throughout the rest of this article, we focus on CCGAP when presenting alternative formulations and solution techniques, and provide additional discussions for other resource-constrained assignment problems where applicable and necessary.

4 Stochastic Programming Formulation for SCCGAP

Using the generic formulation presented in Section 3, the stochastic programming formulation for the stochastic CCGAP (SCCGAP) can be given as:

$$\begin{aligned} \text{(SCCGAP) Minimize} \quad & \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}X_{ij} + E_{\tilde{k}}[Q(X,\tilde{k})] \\ \text{subject to} \quad & (3) \text{ and } (4). \end{aligned} \tag{11}$$

In the following subsections, we present three alternative second-stage value functions $Q(X,\tilde{k})$, hence three alternative formulations, for the SCCGAP. In all three formulations, we assume that the set of resource capacities is a multi-dimensional random variable, $\tilde{k}$, that follows an unknown probability distribution. A group of independent samples, $\{k_f = (k_{1f}, k_{2f}, \ldots, k_{Rf}),\ f \in \mathcal{F} = \{1,\ldots,F\}\}$, can be taken from this distribution. The formulations presented next utilize the information obtained from these samples when finding a solution to the CCGAP with unknown (but now forecasted) resource capacities.

4.1 Alternative 1: Simple Recourse on Amounts of Infeasibility

The first stochastic programming formulation alternative allows infeasibilities in the capacity constraints for a subset of possible outcomes. That is, it allows, with some penalties, tasks to be assigned to agents even

if, for some samples, these assignments result in resource usage in excess of their capacities. As an example, consider a project scheduling problem where we assign tasks to a team of engineers and resources are yearly budgets. For this example, the first formulation allows for going over budget at some years, and penalizes how much over-budget the project is.

In this formulation, the second-stage value function for a given realization k_f of the resource capacities can be defined as

$$Q_1(X, k_f) = min_U\{\sum_{r\in\mathcal{R}} \beta_r U_{rf} | \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}X_{ij} \leq k_{rf} + U_{rf},\, U_{rf} \geq 0,\, r \in \mathcal{R}\}, \tag{12}$$

where U_{rf} is the continuous second-stage variable that represents the amount of resource r used in excess of its capacity, and β_r is the positive marginal cost for each unit of excess capacity usage of resource r, given in the same units as the assignment costs.

Since the set of possible outcomes of resource capacities is approximated by the sampled sets of capacities, the SCCGAP formulation associated with Q_1 is equivalent to the following mixed-integer formulation for the stochastic CCGAP with simple recourse on amounts of infeasibility (SCCGAP-SRA),

(SCCGAP-SRA)

$$\text{Minimize} \quad \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}X_{ij} + \frac{1}{F}\sum_{r\in\mathcal{R}}\sum_{f\in\mathcal{F}} \beta_r U_{rf} \tag{13}$$

$$\text{subject to} \quad \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}X_{ij} \leq k_{rf} + U_{rf}, \quad r \in \mathcal{R}, \quad f \in \mathcal{F}, \tag{14}$$

$$U_{rf} \geq 0, \quad r \in \mathcal{R}, \quad f \in \mathcal{F}, \tag{15}$$

(3) and (4).

We assume the samples are equally likely expectations of the actual capacities. In general, probabilities can be assigned to each sample and incorporated in the objective function.

Stochastic programming formulations that allow excess capacity usage for the other resource-constrained assignment problems can be defined in similar fashion using a continuous recourse variable. For example, the stochastic APSC with simple recourse on amounts of infeasibility (SAPSC-SRA) can be given as

(SAPSC-SRA) Minimize (13)

subject to (3), (4), (7), (14) and (15).

4.2 Alternative 2: Simple Recourse on Number of Infeasibilities

Our second stochastic programming formulation accounts for the number of infeasibilities instead of their magnitudes. Similar to the first alternative formulation, excess capacity usage is allowed with some penalties. However, these penalties are not incurred for each unit of capacity violation, but rather for each resource with excess capacity usage. In the project scheduling example, this second formulation would penalize each year the project is over-budget, regardless of the amount over budget.

The recourse on the number of infeasibilities can be modelled using the following second-stage value function, for a given realization k_f of the resource capacities:

$$\begin{aligned} Q_2(X, k_f) \quad = \quad & min_V\{\sum_{r\in\mathcal{R}} \psi_r V_{rf} | \\ & \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}X_{ij} \leq k_{rf} + MV_{rf},\, V_{rf} = 0 \text{ or } 1,\, r \in \mathcal{R}\}, \end{aligned} \tag{16}$$

where V_{rf} is a binary variable that is equal to 1 if resource r is used in excess of its capacity and 0 otherwise, M is a sufficiently large number (for example, $M = \max\{0, \sum_{i\in\mathcal{I}} \max_{j\in\mathcal{J}}\{a_{ijr}\} - k_{rf}\}$) and ψ_r is the cost of violating the capacity constraint for resource r, in the same units as the assignment costs.

The formulation for the stochastic CCGAP with simple recourse on the number of infeasibilities (SCCGAP-SRN) using the second-stage value function given by Equation (16) has the following equivalent mixed-integer

program:

(SCCGAP-SRN)

$$\text{Minimize} \quad \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}X_{ij} + \frac{1}{F}\sum_{r\in\mathcal{R}}\sum_{f\in\mathcal{F}} \psi_r V_{rf} \tag{17}$$

$$\text{subject to} \quad \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}X_{ij} \le k_{rf} + MV_{rf},\ r\in\mathcal{R},\ f\in\mathcal{F}, \tag{18}$$

$$V_{rf} = 0 \text{ or } 1, \quad r\in\mathcal{R}, \quad f\in\mathcal{F}, \tag{19}$$

(3) and (4).

4.3 Alternative 3: Simple Recourse on Cancellations

The third stochastic programming formulation alternative focuses on assignments rather than resources. Now, excess resource usage is not allowed, but the task-agent assignments are allowed to be "cancelled" as recourse decisions. For the project scheduling example, the third formulation does not allow being over-budget at any year. Instead, we have to cancel certain tasks to keep the project within budget.

Suppose that the cost of *not* assigning task i to any agent, that is cancelling task i, is identified as n_i. Then the second-stage value function that models the recourse on cancellations for a given realization k_f of the capacities can be given as follows:

$$\begin{aligned} Q_3(X,k_f) \quad = \quad & min_Y\{\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}(n_i - c_{ij})Y_{ijf}| \\ & -\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}Y_{ijf} + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}X_{ij} \le k_{rf}, \\ & Y_{ijf} \le X_{ij},\ Y_{ijf} = 0 \text{ or } 1,\ i\in\mathcal{I},\ j\in\mathcal{J}\}, \end{aligned} \tag{20}$$

where Y_{ijf} is the binary second-stage variable that takes the value of 1 if the assignment of task i to agent j is cancelled under capacities k_f, and 0 otherwise. Constraints $Y_{ijf} \le X_{ij}$ ensure that only existing assignments are cancelled. Note that if the assignment of task i to agent j is cancelled, we no longer incur the assignment cost (c_{ij}), but we incur the cancellation cost (n_i). Hence the cancellation coefficient in the above second-stage value function is given as $n_i - c_{ij}$.

Again, the SCCGAP that uses the above second-stage value function can be reduced to an ordinary mixed-integer program since the set of possible realizations of $\tilde{k}$ is approximated by the sampled sets of capacities. The resulting stochastic CCGAP with simple recourse on cancellations (SCCGAP-SRC) is formulated as

(SCCGAP-SRC)

$$\text{Minimize} \quad \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}X_{ij} + \frac{1}{F}\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}(n_i - c_{ij})\sum_{f\in\mathcal{F}} Y_{ijf} \tag{21}$$

$$\text{subject to} \quad \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}(X_{ij} - Y_{ijf}) \le k_{rf},\ r\in\mathcal{R},\ f\in\mathcal{F}, \tag{22}$$

$$Y_{ijf} \le X_{ij}, \quad i\in\mathcal{I}, \quad j\in\mathcal{J}, \quad f\in\mathcal{F}, \tag{23}$$

$$Y_{ijf} = 0 \text{ or } 1, \quad i\in\mathcal{I}, \quad j\in\mathcal{J}, \quad f\in\mathcal{F}, \tag{24}$$

(3) and (4).

5 Solution Techniques

Due to the structural similarity between the stochastic programming formulations of the five resource-constrained assignment problems, we present detailed solution techniques only for SCCGAP. However, as in earlier sections, we discuss extensions to the other formulations.

We present two techniques for solving the stochastic models: a branch-and-bound technique that finds exact optimal solutions and a more efficient, approximate technique that finds near-optimal solutions. Both

of these techniques are based on the Lagrangian relaxation of the capacity constraints, which can be seen as an extension of the Lagrangian relaxation used by Fisher, Jaikumar and Van Wassenhove [6] for the deterministic GAP.

A component of the Lagrangian relaxation approach is to solve the subproblems where the capacity constraints (Equations (14), (18) and (22)) are dualized. In the next subsection, we derive trivial solutions to these subproblems for all three stochastic programming models. These trivial solutions will provide *lower bounds* on the objective functions of the original problems. We then present, in subsection 5.2, techniques to construct primal feasible solutions based on these lower bound solutions. These primal feasible solutions yield *upper bounds* on the original objective function values.

In subsection 5.3, we utilize these lower and upper bounds in a subgradient search algorithm. This subgradient search algorithm is included in a branch-and-bound scheme, described in subsection 5.4, for finding exact solutions to the stochastic programming models. Lastly, we discuss an approximate solution technique that investigates the root node of the branch-and-bound tree, to find near-optimal solutions to the original problems efficiently.

5.1 Lagrangian Relaxation

A close examination of the three stochastic programming formulations reveals that the capacity constraints (Equations (14), (18) and (22)) make these these models difficult to solve. The Lagrangian relaxations of SCCGAP-SRA, SCCGAP-SRN and SCCGAP-SRC, presented next, relax these constraints and incorporate them in the objective function using dual prices. We show these relaxed problems are easier to solve due to the absence of capacity constraints, and use their solutions to find lower bounds on the original objective function values.

5.1.1 Lagrangian Relaxation of SCCGAP-SRA

In order to obtain a lower bound on the optimal objective function of SCCGAP-SRA, we first dualize the constraint set (14) and obtain the following Lagrangian relaxation:

(SCCGAP-SRA-LR)

$$z^{\text{SRA-LR}}(\lambda) = \min_{U,X} \quad \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}X_{ij} + \frac{1}{F}\sum_{r\in\mathcal{R}}\sum_{f\in\mathcal{F}} \beta_r U_{rf} + \sum_{r\in\mathcal{R}}\sum_{f\in\mathcal{F}} \lambda_{rf}\Big(\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}X_{ij} - k_{rf} - U_{rf}\Big) \tag{25}$$

$$\text{subject to} \quad (3),\ (4) \text{ and } (15),$$

$$U_{rf} \le \max\{0, \sum_{i\in\mathcal{I}} \max_{j\in\mathcal{J}}\{a_{ijr}\} - k_{rf}\}, \qquad r\in\mathcal{R}, \quad f\in\mathcal{F}, \tag{26}$$

where $\lambda = \{\lambda_{rf}, r\in\mathcal{R}, f\in\mathcal{F}\}$ is a set of dual variables associated with the capacity constraints. Note that Equation (26) introduces an upper bound on the excess capacity usage variable for each resource and each sample. This constraint set is added to the SCCGAP-SRA-LR in order to produce tighter bounds, without changing the solution space.

For given λ, the Lagrangian relaxation decomposes into two subproblems. The first of these subproblems can be given as:

$$\text{(SCCGAP-SRA-LR1) Minimize} \quad \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\{c_{ij} + \sum_{r\in\mathcal{R}}\sum_{f\in\mathcal{F}}\lambda_{rf}a_{ijr}\}X_{ij} \tag{27}$$

$$\text{subject to} \quad (3) \text{ and } (4),$$

with the following trivial optimal solution:

$$X_{ij}^{\text{SRA-LR}}(\lambda) \;=\; \begin{cases} 1 & \text{for one } j \in \arg\min_{j'\in\mathcal{J}}\{c_{ij'} + \sum_{r\in\mathcal{R}}\sum_{f\in\mathcal{F}}\lambda_{rf}a_{ij'r}\}, \\ 0 & \text{otherwise,} \end{cases} \tag{28}$$

for $i \in \mathcal{I}$, $j \in \mathcal{J}$.

The second subproblem is

$$\begin{aligned} \text{(SCCGAP-SRA-LR2) Minimize} \quad & \sum_{r\in\mathcal{R}}\sum_{f\in\mathcal{F}}(\frac{1}{F}\beta_r - \lambda_{rf})U_{rf} - \lambda_{rf}k_{rf} \\ \text{subject to} \quad & \text{(15) and (26)}, \end{aligned} \tag{29}$$

which also has a trivial optimal solution:

$$U_{rf}^{\text{SRA-LR}}(\lambda) \;=\; \begin{cases} \max\{0, \sum_{i\in\mathcal{I}} \max_{j\in\mathcal{J}}\{a_{ijr}\} - k_{rf}\} & \text{if } \frac{1}{F}\beta_r - \lambda_{rf} < 0, \\ 0 & \text{otherwise,} \end{cases} \tag{30}$$

for $r \in \mathcal{R}$, $f \in \mathcal{F}$.

Note that, the term $\sum_{r\in\mathcal{R}}\sum_{f\in\mathcal{F}}\lambda_{rf}k_{rf}$ in Equation (29) is constant for given λ values, and is a byproduct of the Lagrangian relaxation. It does not change the optimal solution of SCCGAP-SRA-LR but affects its objective function value, which is used to determine the bounds on the primal problem. Hence, we arbitrarily include this term in the SCCGAP-SRA-LR2 formulation.

The above decomposition structure holds for resource-constrained assignment problems other than CCGAP. The only difference occurs when the matching of tasks to agents is one-to-one. In that case, the Lagrangian relaxation on capacity constraints still produces two subproblems. Although the second subproblem will have the same structure, the first subproblem will not have the above trivial solution due to the addition of the constraint set (7). The subproblem becomes an assignment problem, which can be solved using the Hungarian algorithm [16].

5.1.2 Lagrangian Relaxation of SCCGAP-SRN

Consider the Lagrangian relaxation of the SCCGAP-SRN obtained by dualizing Equation (18):

$$\begin{aligned} &\text{(SCCGAP-SRN-LR)} \\ z^{\text{SRN-LR}}(\lambda) = \min_{V,X} \quad & \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}X_{ij} + \frac{1}{F}\sum_{r\in\mathcal{R}}\sum_{f\in\mathcal{F}}\psi_r V_{rf} \\ & + \sum_{r\in\mathcal{R}}\sum_{f\in\mathcal{F}}\lambda_{rf}(\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}X_{ij} - k_{rf} - MV_{rf}) \\ \text{subject to} \quad & \text{(3), (4) and (19)}, \end{aligned} \tag{31}$$

where λ is the set of dual variables associated with the capacity constraints.

The resulting formulation is very similar to SCCGAP-SRA-LR, and has a very similar solution structure. It follows that, for given λ, the optimal solution to SCCGAP-SRN-LR is given by:

$$X_{ij}^{\text{SRN-LR}}(\lambda) = \begin{cases} 1 & \text{for one } j \in \arg\min_{j'\in\mathcal{J}}\{c_{ij'} + \sum_{r\in\mathcal{R}}\sum_{f\in\mathcal{F}}\lambda_{rf}a_{ij'r}\}, \\ 0 & \text{otherwise,} \end{cases} \quad i \in \mathcal{I},\ j \in \mathcal{J}, \tag{32}$$

and

$$V_{rf}^{\text{SRN-LR}}(\lambda) = \begin{cases} 1 & \text{if } \frac{1}{F}\psi_r - M\lambda_{rf} < 0, \\ 0 & \text{otherwise,} \end{cases} \quad r \in \mathcal{R},\ f \in \mathcal{F}. \tag{33}$$

5.1.3 Lagrangian Relaxation of SCCGAP-SRC

As before, we dualize the capacity constraints given by Equation (22). The resulting Lagrangian relaxation of SCCGAP-SRC is given by:

$$\begin{aligned} &\text{(SCCGAP-SRC-LR)} \\ z^{\text{SRC-LR}}(\lambda) = \min_{Y,X} \quad & \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}X_{ij} + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\sum_{f\in\mathcal{F}}\frac{n_i - c_{ij}}{F}Y_{ijf} \\ & + \sum_{r\in\mathcal{R}}\sum_{f\in\mathcal{F}}\lambda_{rf}(\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}(X_{ij} - Y_{ijf}) - k_{rf}) \\ \text{subject to} \quad & \text{(3), (4), (23) and (24)}, \end{aligned} \tag{34}$$

where λ is the set of dual variables associated with the capacity constraints.

While the Lagrangian relaxation does not decompose into two subproblems due to constraints (23), it is still possible to obtain a trivial solution for given λ. The Lagrangian relaxation becomes

$$\begin{aligned} \text{(SCCGAP-SRC-LR1) Minimize} \quad & \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}(c_{ij}+\sum_{r\in\mathcal{R}}\sum_{f\in\mathcal{F}}\lambda_{rf}a_{ijr})X_{ij} \\ & +\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\sum_{f\in\mathcal{F}}(\frac{n_i-c_{ij}}{F}-\sum_{r\in\mathcal{R}}\lambda_{rf}a_{ijr})Y_{ijf} \\ & -\sum_{r\in\mathcal{R}}\sum_{f\in\mathcal{F}}\lambda_{rf}k_{rf} \qquad (35) \\ \text{subject to} \quad & \text{(3), (4), (23) and (24).} \end{aligned}$$

The optimal solution to SCCGAP-SRC-LR is based on the optimal cancellation policies (cancellations with negative objective function coefficients) in the relaxed problem. Based on this optimal cancellation policy, each task is assigned to the agent with the least expected cost. The solutions can be given as

$$X_{ij}^{\text{SRC-LR}}(\lambda)=\begin{cases} 1 & \text{for one } j\in\arg\min_{j'\in\mathcal{J}}\{c_{ij'}+\sum_{r\in\mathcal{R}}\sum_{f\in\mathcal{F}}\lambda_{rf}a_{ij'r} \\ & \qquad +\sum_{f\in\mathcal{F}}\min\{\frac{n_i-c_{ij'}}{F}-\sum_{r\in\mathcal{R}}\lambda_{rf}a_{ij'r},0\}\}, \\ 0 & \text{otherwise,} \end{cases} \qquad (36)$$

for $i\in\mathcal{I}$, $j\in\mathcal{J}$, and

$$Y_{ijf}^{\text{SRC-LR}}(\lambda)=\begin{cases} 1 & \text{if } X_{ij}^{\text{SRC-LR}}(\lambda)=1 \text{ and } n_i-c_{ij}\le F\sum_{r\in\mathcal{R}}\lambda_{rf}a_{ijr}, \\ 0 & \text{otherwise,} \end{cases} \qquad (37)$$

for $i\in\mathcal{I}$, $j\in\mathcal{J}$, $f\in\mathcal{F}$.

The above solution structure holds for resource-constrained assignment problems other than CCGAP. However, if the matching of tasks to agents is one-to-one, the solution for the first-stage variables is determined by solving the following assignment problem:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}\hat{c}_{ij}(\lambda)X_{ij} \qquad (38) \\ \text{subject to} \quad & \text{(3), (4) and (7)} \end{aligned}$$

where

$$\hat{c}_{ij}(\lambda)=\{c_{ij}+\sum_{r\in\mathcal{R}}\sum_{f\in\mathcal{F}}\lambda_{rf}a_{ijr}+\sum_{f\in\mathcal{F}}\min\{\frac{n_i-c_{ij}}{F}-\sum_{r\in\mathcal{R}}\lambda_{rf}a_{ijr},0\}\}. \qquad (39)$$

5.2 Primal Feasible Solutions

The trivial solutions obtained for the Lagrangian relaxations in the previous section provide lower bounds on the primal objective function values of the three stochastic programming formulations, given $\lambda\ge 0$ (Geoffrion [9]). Although these trivial solutions are feasible (and optimal) for the relaxed problems, they may not be feasible with respect to the original set of constraints in the stochastic programming formulation. Specifically, these trivial solutions may fail to satisfy the capacity constraints (Equations (14), (18) and (22)) for some $r\in\mathcal{R}$ and $f\in\mathcal{F}$.

In order to obtain an upper bound on the optimal objective function values of the three stochastic programming models, we need to find solutions that satisfy the capacity constraints. Next, we present techniques to generate primal feasible solutions that satisfy these constraints, by modifying the trivial solutions to the relaxed problems.

5.2.1 Primal Feasible Solutions for SCCGAP-SRA

Remember that, for a given λ, the trivial solution to the Lagrangian relaxation of SCCGAP-SRA is given by Equations (28) and (30). For the same λ, we construct a primal feasible solution to the original problem by keeping the first-stage assignment variables ($X^{\text{SRA-LR}}(\lambda)$) as in Equation (28), and modifying the values of the second-stage variables ($U^{\text{SRA-LR}}(\lambda)$) such that they are feasible with respect to the capacity constraints:

$$X_{ij}^{\text{SRA-LR}'}(\lambda) = X_{ij}^{\text{SRA-LR}}(\lambda), \qquad i \in \mathcal{I},\, j \in \mathcal{J}, \tag{40}$$

$$U_{rf}^{\text{SRA-LR}'}(\lambda) = \max\left\{0, \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr} X_{ij}^{\text{SRA-LR}}(\lambda) - k_{rf}\right\}, \quad r \in \mathcal{R},\, f \in \mathcal{F}, \tag{41}$$

where $X^{\text{SRA-LR}'}(\lambda)$ and $U^{\text{SRA-LR}'}(\lambda)$ give a primal feasible solution.

5.2.2 Primal Feasible Solutions for SCCGAP-SRN

The solution to the Lagrangian relaxation of SCCGAP-SRN is given by Equations (32) and (33) for a given λ. Similar to the first formulation, we construct a primal feasible solution as follows:

$$X_{ij}^{\text{SRN-LR}'}(\lambda) = X_{ij}^{\text{SRN-LR}}(\lambda), \qquad i \in \mathcal{I},\, j \in \mathcal{J}, \tag{42}$$

$$V_{rf}^{\text{SRN-LR}'}(\lambda) = \begin{cases} 1 & \text{if } \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr} X^{\text{SRN-LR}}(\lambda) > k_{rf}, \\ 0 & \text{otherwise,} \end{cases}, \quad r \in \mathcal{R},\, f \in \mathcal{F}, \tag{43}$$

where $X^{\text{SRN-LR}'}(\lambda)$ and $V^{\text{SRN-LR}'}(\lambda)$ satisfy the capacity constraints and provide a primal feasible solution.

5.2.3 Primal Feasible Solutions for SCCGAP-SRC

Similar to the first two formulations, we construct a primal feasible solution to SCCGAP-SRC for a given λ, by keeping the first-stage assignment variables equal to their trivial solution from the Lagrangian relaxation solution as in Equation (36). Now, modifying the second-stage (cancellation) variables to guarantee primal feasibility is equivalent to picking a subset of tasks to cancel for each sample, such that the capacity constraints are satisfied. For a given λ, we systematically select the tasks to cancel (hence construct a primal feasible solution to the original problem) using the following algorithm:

Step 0. Set $X^{\text{SRC-LR}'}(\lambda) = X^{\text{SRC-LR}}(\lambda)$ as in (36).
Step 1. Set $Y^{\text{SRC-LR}'}(\lambda) = Y^{\text{SRC-LR}}(\lambda)$ as in (37).
Step 2. Set $\gamma_{rf} = \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}(X_{ij}^{\text{SRA-LR}}(\lambda) - Y_{ijf}^{\text{SRA-LR}}(\lambda)) - k_{rf}$, for $r \in \mathcal{R}$ and $f \in \mathcal{F}$.
Step 3. Set $f = 1$.
Step 4. If $f = F$, stop, primal feasible solution found.
Step 5. If $\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}(X_{ij}^{\text{SRC-LR}'}(\lambda) - Y_{ijf}^{\text{SRC-LR}'}(\lambda)) \le k_{rf}$ for all $r \in \mathcal{R}$, then $f \leftarrow f + 1$, go to *Step 4*.
Step 6. Let $\tau_i = \sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{J}} a_{ijr} X^{\text{SRC-LR}'}(\lambda)\gamma_{rf}, \quad i \in \mathcal{I}$.
Set $Y_{ijf}^{\text{SRC-LR}'}(\lambda) = 1$ for one $i \in \arg\max_{i'\in\mathcal{I},\, \sum_j Y_{i'jf}^{\text{SRC-LR}'}(\lambda)\neq 1}\{\tau_i\}$.
Go to *Step 4*.

The algorithm above ranks the tasks using the τ values, for each sample that the capacity constraints are violated. The value τ_i measures the contribution of task i to infeasibilities for the sample under consideration. The algorithm then iteratively cancels the tasks that rank the highest, until all capacity constraints regarding that sample is satisfied. At the time the algorithm terminates, $X^{\text{SRC-LR}'}(\lambda)$ and $Y^{\text{SRC-LR}'}(\lambda)$ is a primal feasible solution to SCCGAP-SRC.

5.3 Subgradient Search

We know that $z^{\text{SRA-LR}}(\lambda)$, $z^{\text{SRN-LR}}(\lambda)$ and $z^{\text{SRC-LR}}(\lambda)$ provide lower bounds on the optimal objective function vales, $z^{*\text{SRA}}$, $z^{*\text{SRN}}$ and $z^{*\text{SRC}}$ respectively, for any $\lambda \geq 0$ (Geoffrion [9]), and that the best such bounds can be obtained by solving the Lagrangian dual.

In our solution technique, we use the method of subgradient optimization (Held and Karp [13] and others) to find an approximation to the optimal value of the Lagrangian dual. Subgradient optimization is a method for iteratively searching through possible values of λ, such that the solution to the relaxed problem associated with those λ values converges to the highest lower bound that can be obtained from the Lagrangian dual.

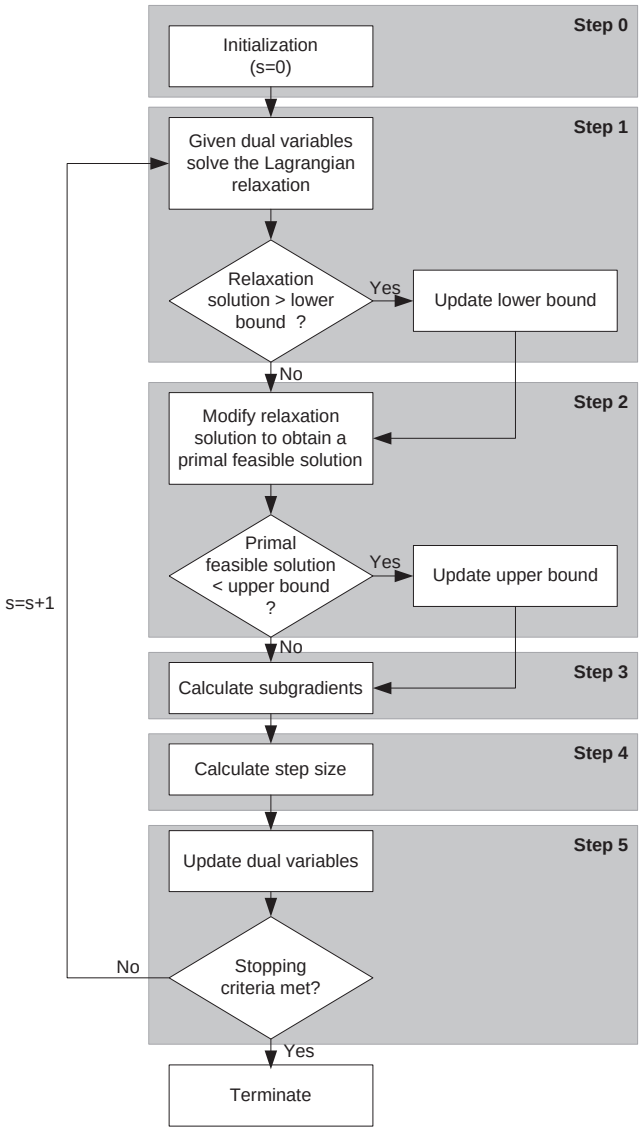


Figure 1: Subgradient Search Algorithm

Our subgradient search algorithm is outlined in Figure 1, and is essentially equivalent for all three formulations. Let the problem under consideration be denoted by $\mathbb{P}$ (SRA, SRN or SRC). Then we define this algorithm as follows:

Step 0. Set $\lambda^{(0)} = 0$, $s = 0$, $\mu = 1$, $\underline{z}^{\mathbb{P}} = 0$, $\overline{z}^{\mathbb{P}} = \infty$.

Step 1. *(Lower bounds)*

Step 1a. Given $\lambda^{(s)}$, solve $\mathbb{P}$-LR and obtain $X^{\mathbb{P}\text{-LR}}(\lambda^{(s)})$ and $U^{\text{SRA-LR}}(\lambda^{(s)})$ if $\mathbb{P}$=SRA, $V^{\text{SRN-LR}}(\lambda^{(s)})$ if $\mathbb{P}$=SRN, $Y^{\text{SRC-LR}}(\lambda^{(s)})$ if $\mathbb{P}$=SRC.

Step 1b. Evaluate $z^{\mathbb{P}\text{-LR}}(\lambda^{(s)})$ using $X^{\mathbb{P}\text{-LR}}(\lambda^{(s)})$ and $U^{\text{SRA-LR}}(\lambda^{(s)})$ if $\mathbb{P}$=SRA, $V^{\text{SRN-LR}}(\lambda^{(s)})$ if $\mathbb{P}$=SRN, $Y^{\text{SRC-LR}}(\lambda^{(s)})$ if $\mathbb{P}$=SRC.

Step 1c. If $z^{\mathbb{P}\text{-LR}}(\lambda^{(s)}) > \underline{z}^{\mathbb{P}}$, update the lower bound using $\underline{z}^{\mathbb{P}} = z^{\mathbb{P}\text{-LR}}(\lambda^{(s)})$, $\underline{X}^{\mathbb{P}} = X^{\mathbb{P}\text{-LR}}(\lambda^{(s)})$ and $\underline{U}^{\text{SRA}} = U^{\text{SRA-LR}}(\lambda^{(s)})$ if $\mathbb{P}$=SRA, $\underline{V}^{\text{SRN}} = V^{\text{SRN-LR}}(\lambda^{(s)})$ if $\mathbb{P}$=SRN, $\underline{Y}^{\text{SRC}} = Y^{\text{SRC-LR}}(\lambda^{(s)})$ if $\mathbb{P}$=SRC.

Step 2. *(Upper bounds)*

Step 2a. Construct a primal feasible solution based on the lower bound, and obtain $X^{\mathbb{P}\text{-LR}'}(\lambda^{(s)})$ and $U^{\text{SRA-LR}'}(\lambda^{(s)})$ if $\mathbb{P}$=SRA, $V^{\text{SRN-LR}'}(\lambda^{(s)})$ if $\mathbb{P}$=SRN, $Y^{\text{SRC-LR}'}(\lambda^{(s)})$ if $\mathbb{P}$=SRC.

Step 2b. Evaluate $z^{\mathbb{P}\text{-LR}'}(\lambda^{(s)})$ using $X^{\mathbb{P}\text{-LR}'}(\lambda^{(s)})$ and $U^{\text{SRA-LR}'}(\lambda^{(s)})$ if $\mathbb{P}$=SRA, $V^{\text{SRN-LR}'}(\lambda^{(s)})$ if $\mathbb{P}$=SRN, $Y^{\text{SRC-LR}'}(\lambda^{(s)})$ if $\mathbb{P}$=SRC.

Step 2c. If $z^{\mathbb{P}\text{-LR}'}(\lambda^{(s)}) < \overline{z}^{\mathbb{P}}$, update the upper bound using $\overline{z}^{\mathbb{P}} = z^{\mathbb{P}\text{-LR}'}(\lambda^{(s)})$, $\overline{X}^{\mathbb{P}} = X^{\mathbb{P}\text{-LR}'}(\lambda^{(s)})$ and $\overline{U}^{\text{SRA}} = U^{\text{SRA-LR}'}(\lambda^{(s)})$ if $\mathbb{P}$=SRA, $\overline{V}^{\text{SRN}} = V^{\text{SRN-LR}'}(\lambda^{(s)})$ if $\mathbb{P}$=SRN, $\overline{Y}^{\text{SRC}} = Y^{\text{SRC-LR}'}(\lambda^{(s)})$ if $\mathbb{P}$=SRC.

Step 3. Calculate subgradients:
$\gamma_{rf}^{(s)} = \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr} X_{ij}^{\text{SRA-LR}}(\lambda^{(s)}) - k_{rf} - U_{rf}^{\text{SRA-LR}}(\lambda^{(s)})$, $r \in \mathcal{R}$, $f \in \mathcal{F}$ if $\mathbb{P}$=SRA,
$\gamma_{rf}^{(s)} = \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr} X_{ij}^{\text{SRN-LR}}(\lambda^{(s)}) - k_{rf} - M V_{rf}^{\text{SRN-LR}}(\lambda^{(s)})$, $r \in \mathcal{R}$, $f \in \mathcal{F}$ if $\mathbb{P}$=SRN,
$\gamma_{rf}^{(s)} = \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}(X_{ij}^{\text{SRC-LR}}(\lambda^{(s)}) - Y_{ijf}^{\text{SRC-LR}}(\lambda^{(s)})) - k_{rf}$, $r \in \mathcal{R}$, $f \in \mathcal{F}$ if $\mathbb{P}$=SRC.

Step 4. Calculate step size:
$\theta^{(s)} = \mu \frac{\overline{z}^{\mathbb{P}} - \underline{z}^{\mathbb{P}}}{\sum_{r\in\mathcal{R}}\sum_{f\in\mathcal{F}}\left(\gamma_{rf}^{(s)}\right)^2}$.

Step 5. Set $\lambda_{rf}^{(s+1)} = \max\{0, \lambda_{rf}^{(s)} + \theta^{(s)}\gamma_{rf}^{(s)}\}$, $r \in \mathcal{R}$, $f \in \mathcal{F}$.
If $|\lambda_{rf}^{(s+1)} - \lambda_{rf}^{(s)}| < \epsilon$, $r \in \mathcal{R}$, $f \in \mathcal{F}$ or $s = s_{limit}$, stop.
Otherwise, $s \leftarrow s+1$, go to *Step 1a.*

In the algorithm, $\epsilon \in \mathbb{R}^+$ and $s_{limit} \in \mathbb{Z}^+$ are user-defined variables that govern the stopping criteria. We set the lower bound on $z^{*\mathbb{P}}$ to $\underline{z}^{\mathbb{P}}$ and the upper bound to $\overline{z}^{\mathbb{P}}$ at the time the subgradient search stops.

The subgradient search algorithm outlined above applies to the primal problems with no fixed variables and corresponds to the initial node of our branch-and-bound tree. Next we describe our branch-and-bound technique.

5.4 Branch-and-Bound Technique

In this section, we present a branch-and-bound technique that utilizes the bounds obtained from the subgradient search algorithm outlined in Section 5.3. It solves any of the three stochastic programming formulations

to optimality, and is a variation of the solution technique proposed by Mazzola and Neebe [17] for the deterministic APSC.

In our branch-and-bound technique, we use an $|\mathcal{I}|+1$ level search tree that searches through the possible values of the first-stage variables. Each level of this search tree corresponds to a task in $\mathcal{I}$.

We determine which task is associated with each level of the search tree after we investigate the level-0 (root) node where there is no fixing of variables. We first measure each task's contribution to excess capacity usage in the level-0 primal feasible solution by calculating

$$\tau_i = \sum_{r\in\mathcal{R}}\sum_{f\in\mathcal{F}}\sum_{j\in\mathcal{J}} a_{ijr}\overline{X}_{ij}\max\{0,\left(\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}\overline{X}_{ij}\right)-k_{rf}\},\quad i\in\mathcal{I}, \tag{44}$$

where $\overline{X}$ is the value of assignment variables in the level-0 solution, i.e., $\overline{X}=\overline{X}^{*\mathbb{P}}$ in Step 2c of the subgradient search algorithm.

We sort the set $\mathcal{I}$ according to these τ values, that is

$$\tau_{[1]}\geq\tau_{[2]}\geq\ldots\geq\tau_{[|\mathcal{I}|]}. \tag{45}$$

Each branch that emanates from a node at level i corresponds to fixing the assignment of task $[i]$ to an agent in $\mathcal{J}$. An example search tree is depicted in Figure 2.

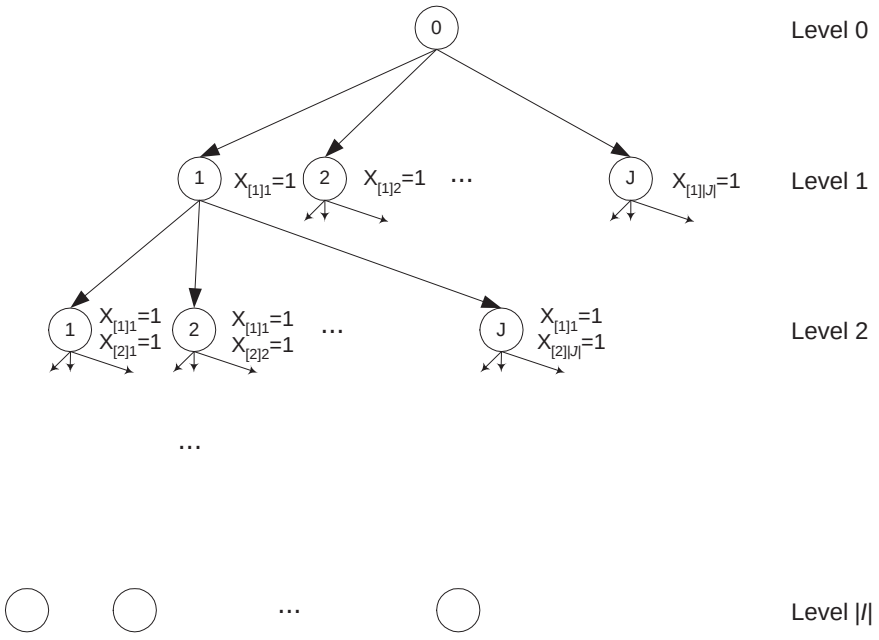


Figure 2: Branch-and-Bound Tree

We use a depth-first branching strategy, and at each node, we obtain lower and upper bounds using the subgradient search schemes from Section 5.3. At any node in the search tree, except node 0, some variables are fixed and we obtain bounds by setting $c_{ij}=M$, $j\in\mathcal{J}\setminus\{l\}$ if X_{il} is fixed to 1. We update the incumbent solution, $\hat{X}$ and $\hat{U}$, $\hat{V}$ or $\hat{Y}$ if any primal feasible (upper bound) solution obtained using the subgradient search has a lower objective function value than the previous incumbent. If the lower bound at any node is worse than the objective function value of the incumbent, or if the node is at level $|\mathcal{I}|$, we fathom that node. If the node is not fathomed, we branch from this node by generating $|\mathcal{J}|$ nodes at the next level. The incumbent solution at the end of the search is an optimal solution to the problem under consideration.

5.5 Approximate Solution Technique

As approximate solutions to SCCGAP-SRA, SCCGAP-SRN and SCCGAP-SRC, we use the upper bound primal feasible solution at the level-0 node of the branch-and-bound search tree. That is, we utilize a subgradient search technique that uses the Lagrangian relaxation of the capacity constraints with no fixed variables, and set the approximate solution $z^{A\mathbb{P}} = \overline{z}^{\mathbb{P}}$ after a single run of the subgradient search, for any of the three formulations.

In fact, the incumbent solution at any node of the search tree can be used as the approximate solution. An advantage of the branch-and-bound procedure outlined above is that an upper bound solution is calculated at every node, and even if the search is terminated (for example, after a certain time limit), we are still able to obtain a feasible near-optimal – if not optimal – solution.

6 Results

To analyze the performance of the alternative solution techniques, a set of experiments were carried out on a group of random test problems. Next we discuss the performance evaluation technique, test problem generation, and the results obtained from these experiments.

6.1 Performance Evaluation

In our experiments, regardless of which approach is used, the solution it produces is a set of task-agent assignments. Let the set of assignments obtained using a given alternative approach be $X^* = \{X^*_{ij},\, i \in \mathcal{I},\, j \in \mathcal{J}\}$, before the actual capacities are realized. Note that the *planned cost* of this solution is $\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}X^*_{ij}$. We are interested in the actual performance of this set of assignments under the actual set of resource capacities (k), which are unknown at the time the assignments are made.

We consider three measures when evaluating the performance of each solution: *actual cost of amount of infeasibilities*, *actual cost of number of infeasibilities*, and *actual cost of cancellations*.

There is a strong correspondence between each of the three performance measures and the objective functions of the three stochastic programming formulations. Indeed, each performance measure evaluates the cost of the assignment decisions under the *actual* capacities, under each of the three recourse definitions given in Section 4.

The first measure, *actual cost of amount of infeasibilities*, calculates and penalizes the amount of resources used in excess of their actual capacities (k) under the given set of assignment decisions (X^*), using

$$AC_1(X^*, k) = \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}X^*_{ij} + \sum_{r\in\mathcal{R}} \beta_r \max\{0, \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}X^*_{ij} - k_r\}. \tag{46}$$

Similarly, the second performance measure, *actual cost of number of infeasibilities*, calculates and penalizes the number of resources used in excess of their actual capacities under the given set of assignment decisions:

$$AC_2(X^*, k) = \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}X^*_{ij} + \sum_{r\in\mathcal{R}} \psi_r \mathbf{I}_{\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}X^*_{ij} > k_r}, \tag{47}$$

where $\mathbf{I}_{\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}X^*_{ij} > k_r}$ is an indicator function that takes the value of 1 if $\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} a_{ijr}X^*_{ij} > k_r$ and 0 otherwise.

In our final performance measure, *actual cost of cancellations*, we need to determine the optimal cancellation policy to maintain resource usage feasibility given the actual capacities (k). We define the following binary program to find an optimal subset of tasks to cancel:

$$\text{Minimize} \quad \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}X^*_{ij}(1 - Y_i) + \sum_{i\in\mathcal{I}} n_i Y_i \tag{48}$$

$$\text{subject to} \quad \sum_{i\in\mathcal{I}} \left(\sum_{j\in\mathcal{J}} a_{ijr}X^*_{ij}\right)(1 - Y_i) \le k_r,\ r \in \mathcal{R}, \tag{49}$$

$$Y_i = 0 \text{ or } 1, \quad i \in \mathcal{I}, \tag{50}$$

where the binary variable Y_i is 1 if task i is cancelled and 0 otherwise. Equation (48) minimizes the additional cost incurred by the cancellations to maintain feasibility with respect to actual resource capacities, which are declared in Equation (49). Note that Equation (48) can be rewritten as:

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}X^*_{ij} + \sum_{i\in\mathcal{I}}\left(n_i - \sum_{j\in\mathcal{J}} c_{ij}X^*_{ij}\right)Y_i, \tag{51}$$

where the first term ($\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}X^*_{ij}$) is a constant.

Like the deterministic CCGAP solver, we use a branch-and-bound algorithm to solve this problem. The search tree again has $|\mathcal{I}|+1$ levels, but now each node at level i corresponds to the fixing of i of the $|\mathcal{I}|$ variables to 1. For a given node κ, let this subset of variables be $\mathcal{I}_\kappa \subseteq \mathcal{I}$, such that $Y_i = 1$, $i \in \mathcal{I}_\kappa$.

As before, we utilize a depth-first branching strategy, and at each node κ, we obtain a lower bound using:

$$\underline{z}(\kappa) = \sum_{i\in\mathcal{I}\setminus\mathcal{I}_\kappa}\sum_{j\in\mathcal{J}} c_{ij}X^*_{ij} + \sum_{i\in\mathcal{I}_\kappa} n_i, \tag{52}$$

where the first term calculates the assignment cost of all uncancelled tasks and the second term accounts for the cost of cancelled tasks at node κ.

During the search, when we reach a feasible solution at any node, we update the incumbent solution, $\hat{Y}$, when this solution has a lower objective function value. If it has a higher objective function value, we fathom that node. We also fathom a node if it does not correspond to a feasible solution and has a lower bound value that is worse than the incumbent. If a node κ at level i is not fathomed, we branch from this node by generating $|\mathcal{I}| - i$ nodes at the next level, each corresponding to the fixing of a variable in $\mathcal{I} \setminus \mathcal{I}_\kappa$. The incumbent solution, $\hat{Y}$, at the end of the search is an optimal solution, Y^*.

We then use this optimal solution to calculate the actual cost of cancellations of X^* using

$$AC_3(X^*, k) = \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}X^*_{ij}(1 - Y^*_i) + \sum_{i\in\mathcal{I}} n_i Y^*_i. \tag{53}$$

In our experiments, the performance of any solution generated by the alternative approaches is measured by AC_1, AC_2, and AC_3.

6.2 Experimental Setup

Our experimental setup is built around a variety of numerical CCGAP instances that are randomly generated using the following problem generation scheme, which is based on the scheme used by Mazzola and Neebe [17] for the deterministic APSC.

For the problem size, we consider ten combinations of number of tasks, I, and number of agents, J. In each combination, J is set to 30% of I, and we consider $I = \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$.

For the smaller problems ($I = \{10, 20\}$), we generate problems with the number of resources, R, equal to $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. For medium-sized problems ($I = \{30, 40, 50\}$), we consider $R = \{1, 2, 3, 4, 5\}$, and for large-sized problems ($I = \{60, 70, 80, 90, 100\}$), we consider a single resource ($R = 1$). Hence, we have a total of 40 *problem sizes*.

For each problem size, we generate three independent *test problems*, totaling 120 test problems overall. In each test problem, the assignment costs, c_{ij}, are uniformly distributed between 0 and 5. The resource usages, a_{ijr}, are uniform between 0 and 10. The cost of each unit of excess capacity usage, β_r, is set to 1, and the cost of each resource with excess capacity usage, ψ_r, is set to 10 for all $r \in \mathcal{R}$. The cost of cancelling an assignment, n_i, is set to 10 for all $i \in \mathcal{I}$.

We consider three probability distributions for the resource capacities: normal, bimodal, and exponential. For a given resource, these three distribution functions are given in Table 4 and depicted in Figure 3. Note that the probability distribution for each resource is independent of that of the other resources.

For each of the 120 test problems in our experimental setup, we have 15 *capacity cases*, five each for the three probability distributions. In each capacity case, we generate five sampled sets of resource capacities (which are used in the alternative approaches; $F = 5$) and a single set of actual capacities (which is used in the performance evaluation) from the corresponding distribution function.

Table 4: Capacity Distributions in the Experimental Setup

Normal*†

$$p^n(k_r) = \begin{cases} \frac{1}{\sigma_r\sqrt{2\pi}}\, e^{\frac{-(k_r-\mu_r)^2}{2\sigma_r^2}} & \text{if } k_r \geq 0, \\ \int_{-\infty}^{0} \frac{1}{\sigma_r\sqrt{2\pi}}\, e^{\frac{-(k_r-\mu_r)^2}{2\sigma_r^2}} & \text{if } k_r = 0, \\ 0 & \text{if } k_r < 0, \end{cases} \quad r \in \mathcal{R}.$$

Bimodal*†

$$p^b(k_r) = \begin{cases} \frac{0.3}{\sigma_r\sqrt{2\pi}}\, e^{\frac{-(k_r-0.8\mu_r)^2}{2\sigma_r^2}} + \frac{0.7}{\sigma_r\sqrt{2\pi}}\, e^{\frac{-(k_r-1.2\mu_r)^2}{2\sigma_r^2}} & \text{if } k_r \geq 0, \\ \int_{-\infty}^{0} \left(\frac{0.3}{\sigma_r\sqrt{2\pi}}\, e^{\frac{-(k_r-0.8\mu_r)^2}{2\sigma_r^2}} + \frac{0.7}{\sigma_r\sqrt{2\pi}}\, e^{\frac{-(k_r-1.2\mu_r)^2}{2\sigma_r^2}} \right) & \text{if } k_r = 0, \\ 0 & \text{if } k_r < 0, \end{cases} \quad r \in \mathcal{R}.$$

Exponential*

$$p^e(k_r) = \begin{cases} \frac{1}{\mu_r}\, e^{-\frac{k_r}{\mu_r}} & \text{if } k_r \geq 0, \\ 0 & \text{if } k_r < 0, \end{cases} \quad r \in \mathcal{R}.$$

$* \ \mu_r = \left(\sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} a_{ijr} \right) / J, \ r \in \mathcal{R}.$

$\dagger \ \sigma_r = 0.1\mu_r, \ r \in \mathcal{R}.$

We evaluate each test problem, under each capacity case, using the exact and approximate techniques for the three formulations. For each test problem, we also investigate the performance of the solutions obtained using two deterministic-formulation-based techniques for comparison purposes: using expected values of resource capacities, and using risk-averse trimmed mean of resource capacities in a deterministic formulation. The selection of these two techniques for comparison is due to their robust performance in a study by Toktas, Yen and Zabinsky [25]. The study investigates deterministic-formulation-based approaches to address capacity uncertainty in resource-constrained assignment problems. In this context, using expected values of resource capacities in a deterministic formulation is equivalent to setting

$$k_r = \frac{\sum_{f \in \mathcal{F}} k_r^f}{F}, \quad r \in \mathcal{R}, \tag{54}$$

when solving the deterministic CCGAP formulation. Similarly, using risk-averse trimmed mean of resource capacities is equivalent to setting

$$k_r = \frac{\sum_{f \leq \lfloor \rho F \rfloor} k_r^{[f]}}{\lfloor \rho F \rfloor}, \quad r \in \mathcal{R}, \tag{55}$$

where ρ, the fraction of samples to be used in the trimmed mean, is a number between $\frac{1}{F}$ and 1. Among the numerical results presented next, the solutions corresponding to the risk-averse trimmed mean approximation is obtained using $\rho = 0.8$.

6.3 Numerical Results

The solution techniques were coded and compiled using Compaq Visual Fortran 6.6, and the tests were made on a personal computer with a 1-GHz Intel Pentium III processor.

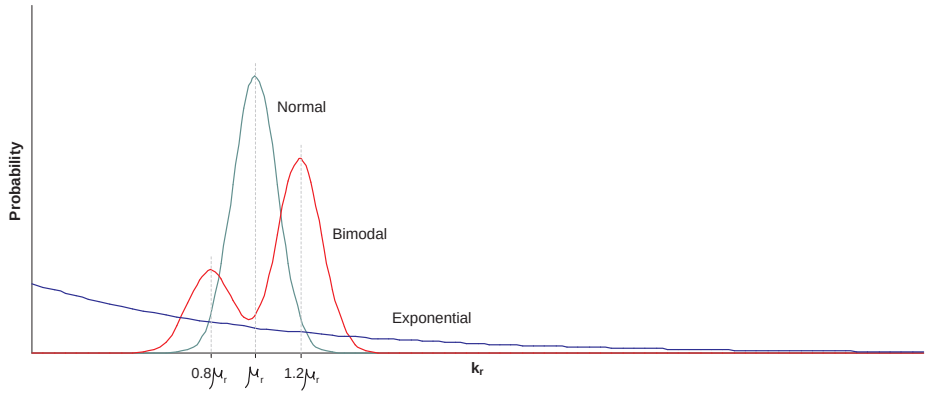


Figure 3: Capacity Distributions in the Experimental Setup

In all of the subgradient search algorithms, ϵ was set to 10^{-5} and n_{limit} was set to 10,000. The branch-and-bound algorithms were terminated if they failed to return an optimal solution in 1 hour of CPU time, and the incumbent solution at the time of termination was used.

To provide a best-case scenario for each test problem, the actual cost under perfect information (AC_0) was calculated, supposing that perfect information about the resource capacities was available. Note that, AC_0 is the planned cost of the optimal solution to the deterministic CCGAP under the actual capacities ($k' = k$).

The actual costs (AC_1, AC_2 and AC_3) of the solutions obtained using each alternative approach were then scaled using

$$SC_1 = \frac{AC_1 - AC_0}{AC_0},\ SC_2 = \frac{AC_2 - AC_0}{AC_0},\ SC_3 = \frac{AC_3 - AC_0}{AC_0}, \tag{56}$$

for each test problem. In this scale, the value zero corresponds to the performance with perfect information, hence lower values denote better performance.

We summarize the numerical results obtained from the experiments in Figure 4 for problems with normally, bimodally and exponentially distributed capacities. In these charts, we report the scaled values of the three performance measures for the solutions obtained using exact and approximate techniques for all three formulations, and the two deterministic approaches, averaged over the 600 problems with the corresponding capacity distribution. The horizontal axes in these charts list the methodologies used. The average values of scaled actual costs (SC_1, SC_2 and SC_3) are measured on the vertical axes, on the left-hand-sides. The three overlapping bars in each column correspond to the average values of the three performance measures for each approach.

For all of the three distributions, the results show that the six stochastic programming approaches dominate the deterministic-formulation-based approaches in all three performance measures. Among the stochastic programming-based approaches, each of the three formulations had the best performance in terms of their corresponding performance measures. That is, the exact and approximate techniques for SCCGAP-SRA had the lowest actual cost in terms of AC_1, SCCGAP-SRN in terms of AC_2 and SCCGAP-SRC in terms of AC_3.

There are a few notable differences between the results under different capacity distributions. The actual costs under bimodally distributed capacities are relatively high when compared to those under normal and exponential. The intuition behind this observation ties to the nature of the bimodal distribution used in the experiments. Under bimodal distribution, the solutions produced by alternative approaches are more likely to expect high resource capacities, as the second mode (higher capacities) of the distribution has a larger weight than the first mode. However, across multiple tests, there is indeed a possibility (30 percent for each resource, due to the way the distribution is constructed) that the resources have, on the average, lower capacities than planned around. As a result, the cost for assuring feasibility under actual capacities will be

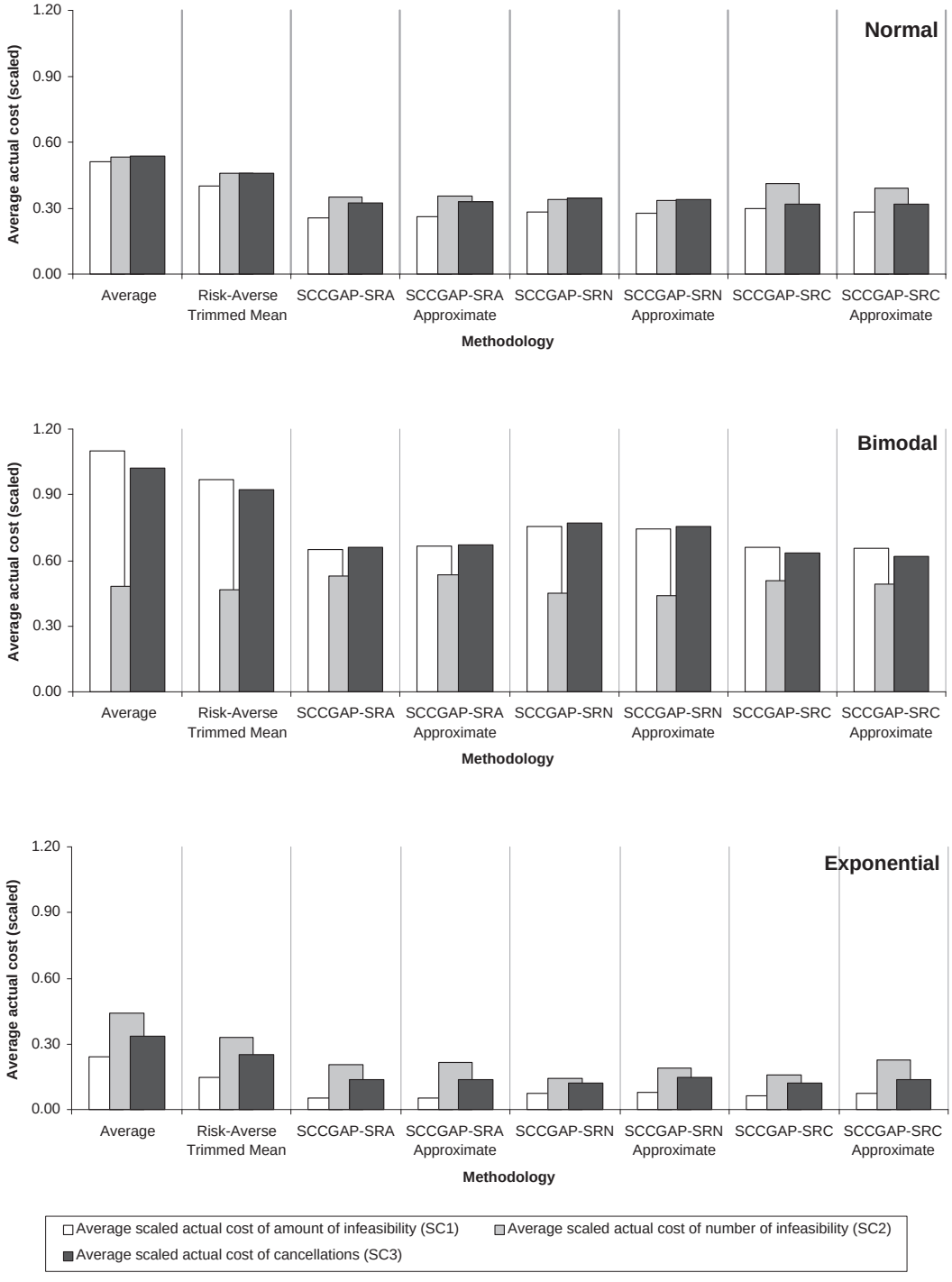


Figure 4: Computational Results

more than those under normal and exponential distributions. This observation signifies the importance of the probability distribution that governs the resource capacities.

It can also be noted that, in the bimodal distribution results, the solutions have lower actual costs of number of infeasibilities than the other performance measures. This observation relates to the costing scheme used in our experiments. In the experimental setup, the cost of each resource with excess capacity usage (ψ_r) is set to ten times the cost of each unit of excess capacity usage (β_r). Given the hypothetical bimodal distribution used, the intuition is as follows: if a resource plans for the second (high-capacity) mode, but the first (low-capacity) mode is realized, the assignments become infeasible with respect to that resource. Even if the *number* of these resources is low (as the results show), the distance between the two modes of the distribution in our tests exceeds ten units of resource, on the average, and the total *amount* of infeasibilities can be large.

We summarize the results of the experiments over all three probability distributions in Figure 5. In the top graph, the horizontal axis lists alternative methodologies and the average values of scaled actual costs are measured on the vertical axes. The three overlapping bars in each column correspond to the average values of the three performance measures for each approach, across 1,800 problems. In the bottom graph, the median CPU times are reported for each approach across all test problems.

Figure 5 illustrates the higher computational complexity of the exact (branch-and-bound) techniques for the stochastic programming formulations relative to the deterministic approaches. We also observe that the third formulation (SCCGAP-SRC) has the highest CPU times. An important observation is that the approximate approaches for all three stochastic programming formulations produced solutions in very low CPU times, outperforming the deterministic formulations.

The dominance of the stochastic programming approaches in performance is evident in this summary chart. Although there is no clear winner among these approaches in terms of all three performance measures, each of the three stochastic programming formulations produce solutions with the lowest actual costs under their respective recourse definition.

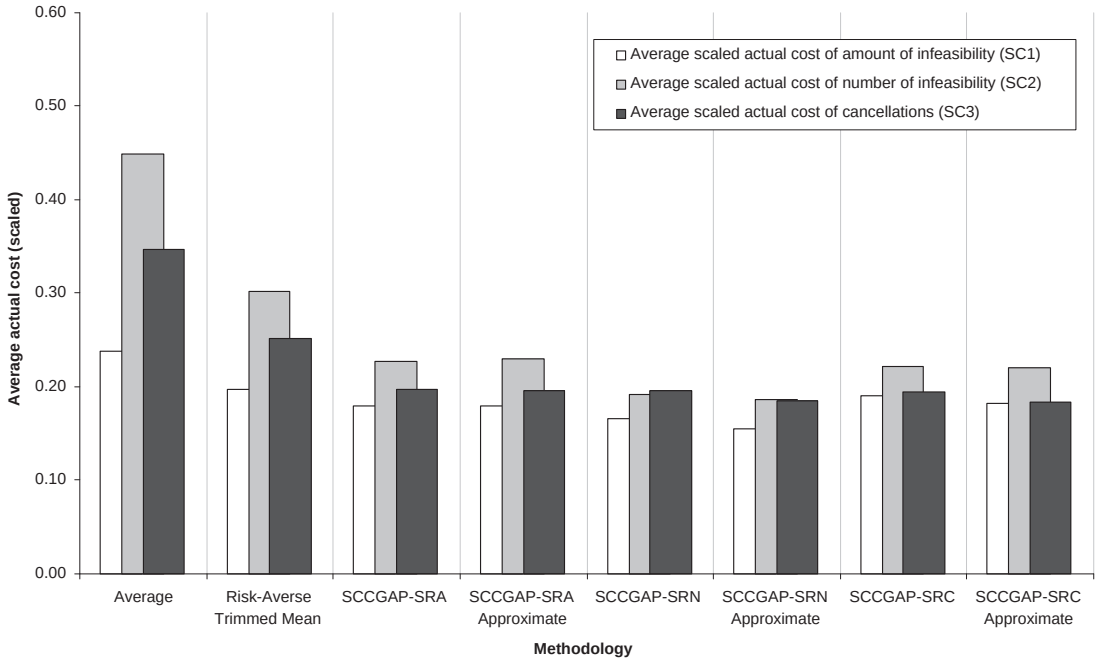
When both performance and efficiency are considered, the stochastic programming-based approximate solution strategies dominate. These approximate approaches can produce solutions that are remarkably close to those obtained using the corresponding branch-and-bound approaches. This fact can be observed in Table 5, which presents the average deviations of the approximate solution performance from the respective branch-and-bound solution performance under each of the three capacity distributions, measured in terms of AC_1, AC_2 and AC_3. Furthermore, the stochastic-programming based approximate strategies were more efficient than an exact deterministic approach (for example using expected values of capacities), producing solutions in less than 1 CPU second for all of the 1,800 problems.

Table 5: Average Deviation of Approximate Solution Strategy Performance from Branch-and-Bound Performance

	SCCGAP-SRA			SCCGAP-SRN			SCCGAP-SRC		
	AC_1	AC_2	AC_3	AC_1	AC_2	AC_3	AC_1	AC_2	AC_3
Normal	0.7%	1.0%	0.7%	1.7%	2.1%	1.9%	1.3%	2.0%	1.6%
Bimodal	1.3%	0.8%	1.1%	2.1%	1.6%	1.7%	1.2%	1.2%	1.1%
Exponential	0.5%	1.8%	0.8%	2.0%	5.6%	3.7%	1.9%	7.3%	3.1%

The reason for the high performance of approximate approaches can be tied to the formulation structures. The formulation of the Lagrangian relaxation on the capacity constraints structurally resembles the stochastic programming formulation. In both the relaxation and the stochastic programming formulations, over-capacity usage is penalized according to the relevant recourse definition. In the Lagrangian relaxation, the penalties are indirectly incorporated via the dual variables, whereas in the stochastic programming approaches, the penalties are defined in the primal problem directly.

The decision of which of the three heuristic approaches to use relies on the specific application addressed and, in particular, the definition of recourse. In our tests, each of the three heuristics dominate the others in terms of their respective recourse definition and performance measure. The branch-and-bound techniques for the stochastic programming formulations can also be used in applications where added performance is desirable at the expense of reduced computational efficiency.



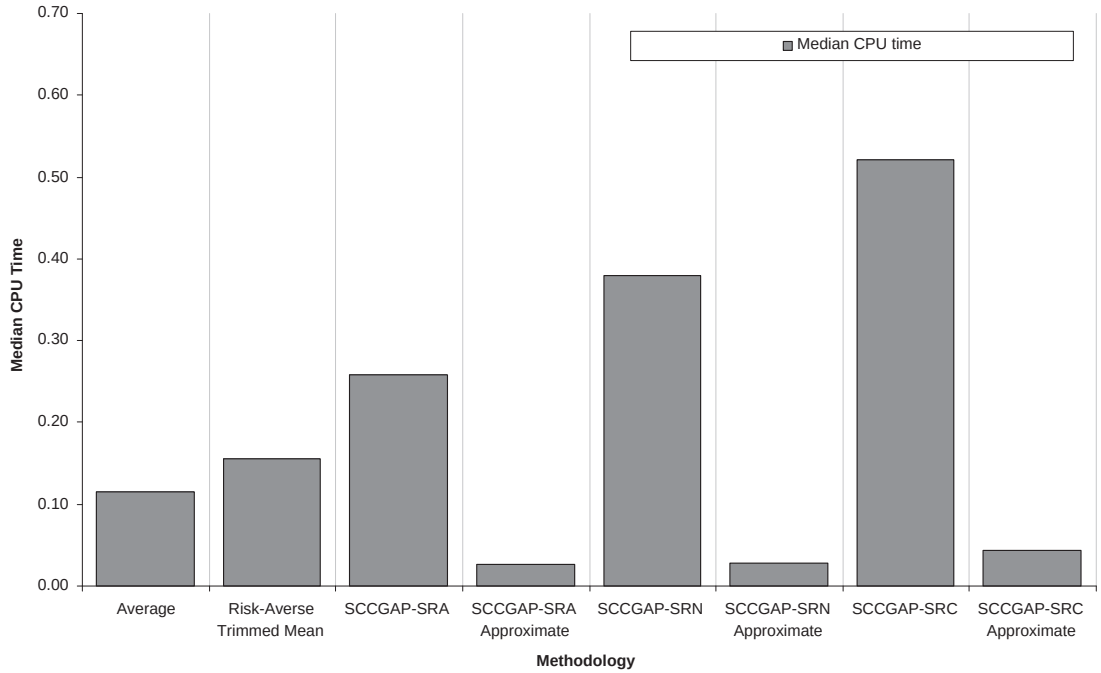


Figure 5: Computational Results, Summary

In general, if actual cost and computational efficiency are the only criteria, approximate stochastic programming approaches dominate the common approach of using expected values of capacities in a deterministic strategy, and provide, on average, approximately 40 % cost savings in less CPU time.

The only exception that would deem deterministic-formulation based approaches preferable is the case when there exist complicating side constraints in the application under consideration, making stochastic programming formulations either inefficient or not possible to implement. It should be noted, however, that certain types of side constraints have structure which can be accommodated in the proposed solution techniques (see, for example, Toktas [26]).

7 Conclusion

In this article we investigated stochastic-programming-based methodologies to address capacity uncertainty in resource-constrained assignment problems. After a brief review of the resource-constrained generalizations of the classical assignment problem, we summarized possible sources of uncertainty that can affect these problems.

We focused on problems with unknown resource capacities and identified three alternative stochastic programming formulations. The three alternative formulations differ mainly in their recourse definitions, and allow for the modelling of applications where there is limited information on the uncertainty structure. We also presented exact and approximate techniques to solve the resulting formulations.

In order to analyze the performance of these solution techniques, we focused on the CCGAP as a specific generalization of the assignment problem. We discussed three alternative ways to measure each approach's solution performance under actual capacities. For one of these performance measures, we presented a feasibility solver to determine the optimal subset of assignments to be cancelled after the actual capacities are realized.

We utilized these algorithms when carrying out experiments on a large set of CCGAP instances, for which the resource capacities were uncertain. The results from these experiments show that, regardless of the probability distribution that the resource capacities follow, stochastic programming-based approaches were superior to deterministic-formulation-based techniques. Moreover, approximate techniques for the stochastic programming formulations could produce solutions that are remarkably close to those obtained using the corresponding branch-and-bound approaches, in significantly low computational time. These approximate solutions were better than solutions obtained using expected values of capacities in a deterministic formulation (as is commonly done) in terms of both solution performance and computation time.

There exist several future research directions on the topics presented in this article. A possible direction is to explore strategies other than Lagrangian relaxation to solve the stochastic programming formulations presented. Such alternative strategies could include linear relaxation, decomposition or heuristic approaches. The efficiency and effectiveness of any alternative strategy should be compared to those presented.

Another possible extension is investigating uncertainty in parameters other than resource capacities in resource-constrained assignment problems. In this case, the stochastic programming formulations would be significantly different. A new development of solution techniques would need to redefine recourses and take advantage of a different problem structure. Finally, it is also possible to extend the experimentation, particularly by including tests on resource-constrained assignment problems other than CCGAP.

The resource-constrained generalizations of the assignment problem have many application areas, and a typical issue is the effects of uncertainty. This research demonstrates that stochastic programming is a viable approach to address specific resource-constrained assignment problems with uncertainty in resource capacities. The stochastic-programming-based approximations are computationally efficient and provide a methodology for near-optimal solutions.

Acknowledgement

This study was supported in part by a grant from The Boeing Company for collaborative research on air traffic flow management under temporary capacity constraints.

References

[1] Benders, J.F., van Nunen, J.A.E.E. : A Property of Assignment Type Mixed Integer Linear Programming Problems. Oper. Res. Lett. **2**(2), 47-52 (1983)

[2] Berge, M.E., Hopperstad, C.A., Haraldsdottir, A. : Airline Schedule Recovery in Collaborative Flow Management with Airport and Airspace Capacity Constraints. The 2003 US/Europe Air Traffic Management R&D Seminar, Budapest, June 2003

[3] Birge, J.R., Louveaux, F. : Introduction to Stochastic Programming. Springer-Verlag, New York, 1997

[4] Cattrysse, D.G., van Wassenhove, L.N. : A Survey of Algorithms for the Generalized Assignment Problem. Eur. J. Oper. Res. **60**(3), 260-272 (1992)

[5] Fisher, M.L., Jaikumar, R. : A Generalized Assignment Heuristic for Vehicle Routing. Networks **11**, 109-124 (1981)

[6] Fisher, M.L., Jaikumar, R., van Wassenhove, L.N. : A Multiplier Adjustment Method for the Generalized Assignment Problem. Manage. Sci. **32**(9), 1095-1103 (1986)

[7] Gavish, B., Pirkul, H. : Efficient Algorithms for the Multiconstraint Generalized Assignment Problem. Working Paper QM 8523, University of Rochester, 1985

[8] Gavish, B., Pirkul, H. : Algorithms for the Multi-Resource Generalized Assignment Problem. Manage. Sci. **37**(6), 695-713 (1991)

[9] Geoffrion, A.M. : Lagrangian Relaxation and Its Uses in Integer Programming. Math. Program. Study Ser. **2**, 82-114 (1974)

[10] Glover, F., Karney, D., Klingman, D. : Implementation and Computational Comparisons of Primal, Dual, and Primal-Dual Computer Codes for Minimum Cost Network Flow Problems. Networks **4**, 191-212 (1974)

[11] Guignard, M., Rosenwein, M. : An Improved Dual-Based Algorithm for the Generalized Assignment Problem. Oper. Res. **37**(4), 658-663 (1989)

[12] Haddadi, S. : Lagrangian Decomposition Based Heuristic for the Generalized Assignment Problem. INFOR **37**(4), 392-402 (1999)

[13] Held, M., Karp, R.M. : The Travelling Salesman Problem and Minimum Spanning Trees. Oper. Res. **18**, 1138-1162 (1970)

[14] Jornsten, K., Nasberg, M. : A New Lagrangian-Relaxation Approach to the Generalized Assignment Problem. Eur. J. Oper. Res. **27**(3), 313-323 (1986)

[15] Kennington, J.L., Wang, Z. : An Empirical Analysis of the Dense Assignment Problem: Sequential and Parallel Implementations. ORSA J. Comput. **3**, 299-306 (1991)

[16] Kuhn, A.W. : The Hungarian Method for the Assignment Problem. Nav. Res. Logist. **2**, 83-97 (1955)

[17] Mazzola, B., Neebe, A.W. : Resource-Constrained Assignment Scheduling. Oper. Res. **34**(4), 560-572 (1986)

[18] Murphy, R.A. : A Private Fleet Model with Multi-Stop Backhaul. Working Paper 103, Optimal Decision Systems, Green Bay, WI, 1986

[19] Pirkul, H. : An Integer Programming Model for the Allocation of Databases in a Distributed Computer System. Eur. J. Oper. Res. **26**(3), 401-411 (1986)

[20] Ross, G.T. and Soland, R.M. : A Branch-and-Bound Algorithm for the Generalized Assignment Problem. Math. Program. **8**, 91-103 (1975)

[21] Ross, G.T. and Soland, R.M. : Modeling Facility Location Problems as Generalized Assignment Problems. Manage. Sci. **24**(3), 345-357 (1977)

[22] Albareda-Sambola, M., van der Vlerk, M.H., Fernández, E. : Exact Solutions to a Class of Stochastic Generalized Assignment Problems. Stochastic Programming E-Print Series 2002-4, 2002

[23] Spoerl, D., Wood, R.K. : A Stochastic Generalized Assignment Problem. Naval Postgraduate School, Working Paper, 2003

[24] Srinivasan, V., Thompson, G.L. : Benefit-Cost Analysis of Coding Techniques for the Primal Transportation Algorithm. J. Assoc. Comput. Mach. **20**, 194-213 (1973)

[25] Toktas, B. Yen, J.W., Zabinsky, Z.B. : Addressing Capacity Uncertainty in Resource-Constrained Assignment Problems. Technical Report, University of Washington, Seattle, WA, 2003

[26] Toktas, B. : Addressing Capacity Uncertainty in Resource-Constrained Assignment Problems. Ph.D. Thesis, University of Washington, Seattle, WA, 2004

[27] Winston, W. : Introduction to Mathematical Programming, Duxbury Press, 1995

[28] Zimokha, V.A., Rubinstein, M.I. : R&D Planning and the Generalized Assignment Problem. Autom. Remote Control **49**, 484-492 (1988)