Journal of Global Optimization (2005) 32: 385–400
DOI 10.1007/s10898-004-5910-6

The Million-Variable "March" for Stochastic Combinatorial Optimization

LEWIS NTAIMO and SUVRAJEET SEN
Department of Systems and Industrial Engineering, The University of Arizona, Tucson, Az 85721, USA (e-mail: nlewis@sie.arizona.edu)

(Received 1 May 2004; accepted in revised form 13 May 2004)

Abstract. Combinatorial optimization problems have applications in a variety of sciences and engineering. In the presence of data uncertainty, these problems lead to stochastic combinatorial optimization problems which result in very large scale combinatorial optimization problems. In this paper, we report on the solution of some of the largest stochastic combinatorial optimization problems consisting of over a million binary variables. While the methodology is quite general, the specific application with which we conduct our experiments arises in stochastic server location problems. The main observation is that stochastic combinatorial optimization problems are comprised of loosely coupled subsystems. By taking advantage of the loosely coupled structure, we show that decomposition-coordination methods provide highly effective algorithms, and surpass the scalability of even the most efficiently implemented backtracking search algorithms.

Key words: Combinatorial optimization, Stochastic mixed integer programming, Stochastic server location

1. Introduction

Combinatorial optimization problems provide some of the most important, and challenging problems in computer science and operations research (CS/OR) (Cook et al., 1998). These optimization problems are characterized by discrete-choice variables for which 0 or 1 are the only possible values that can be assigned. One example of such a problem is the network design problem that determines which nodes and arcs should be built so as to provide network services at least "cost." Despite several decades of advances in CS/OR, combinatorial optimization problems continue to evoke parallels with galactic dimensions. For instance, a combinatorial optimization problem (e.g. network design) with 1000 variables can lead to a search space with 2^{1000} decision states! This is sometimes referred to as "combinatorial explosion." Moreover, these notorious problems belong to the class of *NP*-hard problems for which "good" algorithms are unlikely. It is no surprise that combinatorial optimization remains a "Grand CS/OR Challenge" problem.

It turns out that the challenge of combinatorial optimization is magnified many-fold in cases where data is uncertain. For instance, a network design problem in which customer locations are unknown leads to a stochastic combinatorial optimization (SCO) problem. Even the smallest of practical SCO problems lead to an astronomical number of decision states, and easily exceed current computational capabilities. As one might expect, SCO problems provide an even grander challenge. In this paper, we report on the solution of instances with the largest data sets for SCO problems to date. These instances, one of which contains over a million binary (0–1) variables, represent data for stochastic server location problems (SSLP) in which servers have to be located prior to demand realization.

These problems (SSLP) are said to have had a significant role in the economic downturn associated with the telecommunications sector of the US economy. Unfortunately, the lack of algorithms for solving SSLP has prompted industry to use planning models that are based on one (deterministic) forecast. The telecommunications sector is now awash in unused server capacity that has resulted from a combination of inaccurate forecasts, and an over-reliance on deterministic planning. In a volatile economy, planning models should recognize that forecasts can be error prone, and should seek plans that are robust to forecasting errors. One approach that provides robust plans is known as stochastic programming (SP) (Birge and Louveaux, 1997) and incorporates multiple scenarios within a planning model. Over a decade ago, (Sen et al., 1994) used the SP planning methodology to an industrial-sized network planning problem for Sonet-Switched Networks (SSN), and demonstrated improved network performance due to the SP model. Subsequently, others have reported solving the SSN instance using advanced computing architectures such as grid-computing (Linderoth and Wright, 2003). However, the types of algorithms used to solve SSN are based on stochastic linear programming, and do not allow binary variables. Networks that are being implemented today consist of high speed optical fiber cables, and high speed optical switches which are very capital intensive. In addition, service providers like AT&T are unsure of customer demand (Doverspike, 2003). Network design models under these circumstances lead to SSLP, which is the class of models studied here. Unfortunately, network designers have not had access to algorithms that can solve SSLP of realistic dimensions.

In this paper, we present a SP model of SSLP, and report on the performance of an algorithm called the D^2 method (Sen and Higle, 2000; Sen et al., 2002). This method refers to disjunctive decomposition which combines a particular divide-and-conquer paradigm with cutting planes[1] based on disjunctive programming (Balas, 1979). Unlike pure backtracking-search

[1]A cut is a linear inequality that is implied by the binary restrictions and other inequalities defining the instance.

(branch-and-bound, B&B) algorithms that search the decision space, the divide-and-conquer approach of D^2 refers to "conquering" smaller models which are coordinated in such a manner as to solve the larger model. Mathematically such an approach is sometimes referred to as a decomposition-coordination approach. Our computational experiments illustrate that for all but the smallest of instances, standard B&B algorithms are unable to solve these SSLP instances. However, decomposition-coordination allows the D^2 algorithm to solve these very large instances with reasonable computational effort.

This paper begins with a statement of the stochastic server location problem (SSLP) as a stochastic combinatorial optimization model. We then proceed with a brief summary of previous computational experience with SCO models. Following this, we present our experimental setup and our computational results with SSLP. These experiments demonstrate that new decomposition methods (e.g., D^2) provide a powerful tool for stochastic combinatorial optimization. The D^2 algorithm is derived in (Sen and Higle, 2000) and illustrated in (Sen et al., 2002).

2. Stochastic Server Location Problems

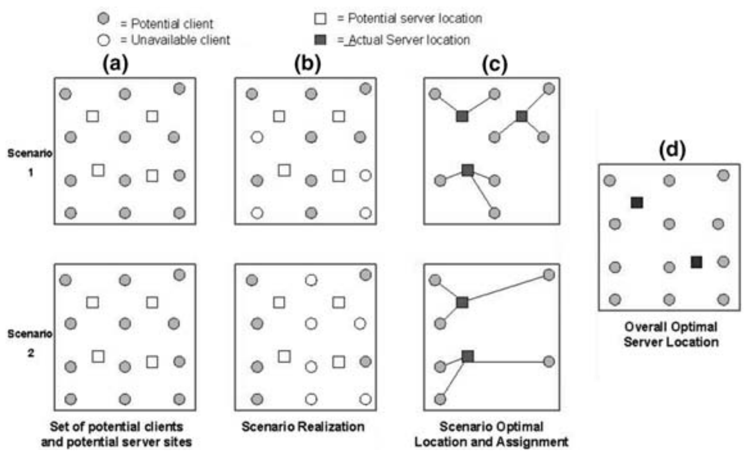
The stochastic server location problems (SSLP) we study find applications in a variety of domains (Wang et al., submitted for publication) such as network design for electric power, internet services, telecommunications, and water distribution. In particular, consider a set of possible customer buildings in a metropolitan area for which a service provider is interested in installing optical fibers and switching equipment in the most profitable manner. Due to the uncertainty regarding the customer base for high speed services, telecommunications providers often adopt a very conservative approach to capital investment, leading to potential losses in revenue (Doverspike, 2003). Such problems are common in practice (Doverspike, 2003) and can be formulated as the SSLP. Because of the variety of application domains for SSLP, we use the names *server* and *client* in a generic sense.

Stochastic programming (SP) approaches to such decision problems incorporate uncertainty via a collection of future scenarios. The plan (e.g. servers to be located) is to be chosen in such a way that the system performs well under those future scenarios that are deemed possible. By a "scenario" we refer to a set of potential clients that do materialize, and a plan is evaluated against all foreseeable future scenarios. In contrast, traditional (deterministic) combinatorial optimization models recommend decisions under the assumption that only one scenario is possible in the future. It is the need to accommodate a large number of future scenarios that leads to very large scale SCO models. So long as a SCO model allows a fairly large number of scenarios, there is no loss of generality in assuming that there are only a finite number of scenarios that are possible in the future. Then each scenario may be associated with a probability of occurrence, denoted p_ω, $\omega \in \Omega$ (the entire collection of scenarios). For such stochastic optimization models, it is common to optimize "expected costs" (Birge and Louveaux, 1997).

Figure 1 gives an illustration of the stochasticity in the SSLP. The figure shows two scenarios, 1 and 2. For *scenario* 1 we have 4 potential server locations and 12 potential clients (see panel a) of Figure 1. In (panel b), which depicts the clients that materialize in *scenario* 1, only 8 out of the 12 potential clients are available for service. Figure 1 panel c shows the optimal server locations if we only plan for *scenario* 1. For *scenario* 2 we again have 4 potential server locations and 12 potential clients. However in this scenario, only 6 out of the 12 potential clients are available for service (see panel b). In this case the optimal server locations are at the two sites shown in panel c). Although Figure 1 shows the optimal server locations and client-server assignments for each scenario, the goal of the SSLP is to find the overall optimal server locations (see panel d) which accommodate all foreseeable scenarios. In this case the overall optimal solution is not optimal for either of the two scenarios. In the next subsection we formally give two model formulations for the SSLP.

2.1. MODEL FORMULATION

Let $\mathcal{I}$ and $\mathcal{J}$ be index sets for the clients and servers, with $|\mathcal{I}| = n$ and $|\mathcal{J}| = m$. Let $\mathcal{Z}$ denote a given set of zones. For $i \in \mathcal{I}$ and $j \in \mathcal{J}$ we define the following:



● = Potential client
○ = Unavailable client
□ = Potential server location
■ = Actual Server location
(a)
(b)
(c)
(d)
Scenario 1
Scenario 2
Set of potential clients and potential server sites
Scenario Realization
Scenario Optimal Location and Assignment
Overall Optimal Server Location

Figure 1. SSLP illustration.

Data:

c_j	Cost of locating a server at location j
q_{ij}	Revenue from client i being served by server at location j
d_{ij}	Client i resource demand from server at location j
u	Server capacity
v	An upper bound on the total number of servers that can be located
w_z	Minimum number of servers to be located in zone $z \in \mathcal{Z}$
$\mathcal{J}_z$	The subset of server locations that belong to zone z

$$h^i(\omega) = \begin{cases} 1, & \text{it if client } i \text{ is present in scenario } \omega, \ \omega \in \Omega, \\ 0, & \text{otherwise.} \end{cases}$$

$h(\omega)$	Client availability vector for scenario $\omega \in \Omega$ with elements $h^i(\omega)$, $i \in \mathcal{I}$
p_ω	Probability of occurrence for scenario $\omega \in \Omega$

Decision variables:

$$x_j = \begin{cases} 1, & \text{if a server is located at site } j, \\ 0, & \text{otherwise.} \end{cases}$$

$$y_{ij}^{\omega} = \begin{cases} 1, & \text{if client } i \text{ is served by a server at location } j \text{ under scenario } \omega, \\ 0, & \text{otherwise.} \end{cases}$$

The essence of the SSLP may be described as follows. Suppose that we place a server at location j. Then, this allocation costs c_j, and provides enough capacity to serve up to u amount of resource to clients. The revenue generated by serving client i from location j, is denoted q_{ij}. There is also a shortage cost (penalty) q_{jo} for each unit of demand that remains unserved among the clients assigned to server j. If client i is served by a server at location j, it uses d_{ij} units of resource from the server. Note that the dependence of resource utilization (d_{ij}) on the client–server pair allows us to model losses occurring from assigning client i to server j. Such considerations are important in certain networks like electricity and water. In cases where the losses are negligible (at least for planning purposes), one could use the same value of d_{ij} for all j, as long as i is fixed.

As far as operational considerations are concerned, we allow only one server to be installed at each location and each client can only be served by one server. There is also a requirement that a minimum number of servers denoted w_z, $z \in \mathcal{Z}$ be located in a given area or zone. Finally, the problem is to choose locations of servers and client–server assignments that maximize the total expected revenue subject to the given constraints. The SSLP can be stated as follows:

$$\text{Min} \quad \sum_{j \in \mathcal{J}} c_j x_j - \sum_{\omega \in \Omega} p_\omega \left(\sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} q_{ij}^\omega y_{ij}^\omega - \sum_{j \in \mathcal{J}} q_{j0}^\omega y_{jo}^\omega \right) \tag{1a}$$

$$\text{S.t.} \quad \sum_{j \in \mathcal{J}} x_j \leqslant v, \tag{1b}$$

$$\sum_{j \in \mathcal{J}_z} x_j \geqslant w_z, \quad \forall z \in \mathcal{Z} \tag{1c}$$

$$\sum_{i \in \mathcal{I}} d_{ij} y_{ij}^\omega - y_{j0}^\omega \leqslant u x_j, \quad \forall j \in \mathcal{J},\ i \in \mathcal{I},\ \omega \in \Omega, \tag{1d}$$

$$\sum_{j \in \mathcal{J}} y_{ij}^\omega = h^i(\omega), \quad \forall i \in \mathcal{I},\ \omega \in \Omega, \tag{1e}$$

$$x_j \in \{0,1\}, \quad \forall j \in \mathcal{J}. \tag{1f}$$

$$y_{ij}^\omega \in \{0,1\}, \quad \forall i \in \mathcal{I}, \quad j \in \mathcal{J},\ \omega \in \Omega. \tag{1g}$$

$$y_{j0}^\omega \geqslant 0, \quad \forall j \in \mathcal{J},\ \omega \in \Omega. \tag{1h}$$

Formulation (1a–1h) is the so called deterministic equivalent problem (DEP) in stochastic programming. When one solves this problem, one obtains a recommendation to locate servers in locations that will hedge against a variety of scenarios in which certain clients do not materialize. The variables y_{ij}^ω are decisions that will be implemented in the future, when scenario ω is finally observed. The location variables (x) are referred to as first-stage decisions, and the assignment variables (y^ω) are referred to as recourse (or second-stage) decisions. Unlike the first-stage variables, the latter are dependent on the scenario ω.

The constraints provide a mechanism to impose the operational requirements. Thus constraints (1b) satisfy the requirement that only up to a total of v available servers can be installed. The zonal requirements that specify how many servers are necessary in each zone are given by constraint (1c). Constraints (1d) dictate that a server located at site j can serve only up to its capacity u. We have introduced a variable y_{j0}^ω in this constraint to accommodate any overflows that are not served due to limitations in server capacity. These overflows result in a loss of revenue at a rate of q_{j0}^ω. Unlike the deterministic version of such problems, the inclusion of an artificial variable may allow a client to be assigned to servers that are not located. However, penalty costs associated with such an assignment may result in such high costs as to preclude it in an optimal solution, unless server capacity is so limited that some clients have to be turned away. For cases in which the server capacities are severely restricted, linear overflow costs may not provide an appropriate modeling tool and an extension of this model may be necessary. Since most of our experiments will be conducted on instances with sufficient server capacity, overflows will be zero, and linear overflow costs will suffice.

Continuing with a description of the rest of the model, the requirement that each available client is served by only one server is given by constraints (1e). Constraints (1f) and (1g) are the binary restrictions on the decision variables. Finally, constraints (1h) are the nonnegativity requirements on the overflow variables.

If we denote the number of scenarios by S, where $S = |\Omega|$, and the number of zones by $|\mathcal{Z}|$, then this DEP formulation has $m(1 + nS)$ variables and $m + |\mathcal{Z}| + (m + n)S$ constraints. The number of scenarios can be very large in general and therefore, this formulation is a large scale problem and can get out of hand very quickly. Hence, the need to decompose it. For example, for a problem instance with 10 potential servers, 50 potential clients and 2000 scenarios, with no zonal constraints, we have 1,000,010 binary variables and 120,010 constraints!

The above formulation can be considered as a basic model from which other SSLP models can be extended. For instance, we can easily make an extension to a SSLP with multiple server types each with a different capacity. Thus we can simply define new variables for each server type and add new constraints on the server type requirements to the model. In any event, the additional constraints would depend on the specific application at hand.

We will now decompose the DEP into a two stage SMIP with complete recourse and binary variables in both stages. The two stage SSLP can be formally stated as follows:

$$\text{Min} \sum_{j \in \mathcal{J}} c_j x_j - E[f(x, \tilde{\omega})] \tag{2a}$$

$$\text{S.t.} \sum_{j \in \mathcal{J}} x_j \leqslant v, \tag{2b}$$

$$\sum_{j \in \mathcal{J}_z} x_j \geqslant w_z, \quad \forall z \in \mathcal{Z} \tag{2c}$$

$$x_j \in \{0, 1\}, \quad \forall j \in \mathcal{J}, \tag{2d}$$

where $E[\cdot]$ is the usual mathematical expectation with

$$E[f(x, \tilde{\omega})] = \sum_{\omega \in \Omega} p_\omega f(x, \omega),$$

and for any x satisfying the constraints (2b)–(2d) and $\omega \in \Omega$ we define

$$f(x,\omega) = \text{Min} - \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} q_{ij}y_{ij} + \sum_{j\in\mathcal{J}} q_{j0}y_{j0} \tag{3a}$$

$$\text{S.t.} \quad \sum_{i\in\mathcal{I}} d_{ij}y_{ij} - y_{j0} \leqslant ux_j, \quad \forall j \in \mathcal{J}, \tag{3b}$$

$$\sum_{j\in\mathcal{J}} y_{ij} = h^i(\omega), \quad \forall i \in \mathcal{I}, \tag{3c}$$

$$y_{ij} \in \{0,1\}, \quad \forall i \in \mathcal{I}, \quad j \in \mathcal{J}, \tag{3d}$$

$$y_{j0} \geqslant 0, \quad \forall j \in \mathcal{J}. \tag{3e}$$

As before, the x_j's are the first-stage decision variables. We observe that although the second-stage (recourse) variables y_{ij}'s continue to depend on the outcome ω, this dependence is not explicitly indicated here. This is because the subproblem for each outcome ω is decoupled from all other outcomes once a vector x is given. This formulation emphasizes the loosely coupled nature of SCO problems, and while this decomposition framework has been extensively used for stochastic linear programming (Cook et al., 1998), its use for SCO problems has been limited. Readers familiar with the current state of computations with SCO should feel free to proceed to Section 4.

3. The "March" Thus Far: Previously Solved SCO Instances

Before presenting our computational results, it is appropriate to briefly examine previous computational experiments with SCO problems. This section also serves to illustrate the variety of applications in which uncertainty must be accommodated within combinatorial optimization. Table I gives a summary of some of the largest SCO instances that have been reported. The first three instances are available on the SIP test problem library

Table I. Summary of previously reported of SCO problems (DEP)

Name (cite)	Scenarios	Variables	Binaries	Integers	Constraints
dcap243_500 (Ahmed and Garcia, in press, Ahmed et al., in press)	500	18,018	18,006		9012
SEMI4 (Barahona et al., 2001)	4	39,820		612	23,370
SIZES10 (Jorjani et al., 1995)	10	825	110		341
SSCh_c5 (Alonso-Ayuso et al., 2003)	23	3768	114		3933
SGAP_28 (Albareda-Sambola et al., 2002)	45	2745	2745		2835
SVRP_100 (Laporte et al., 2002)	a	10,000	10,000		b
E160-2_FRP (Alonso et al., 2000)	15	16,753	16,753		32,455

a – The recourse function has a closed-form expression. b – IP formulation has exponentially many constraints.

(http://www.isye.gatech.edu/~sahmed/siplib/). The sizes of the instances shown are for the corresponding DEP formulation.

Problem dcap243_500 is a two-stage stochastic integer program arising in dynamic capacity acquisition and allocation under uncertainty. The problem has complete recourse, mixed-integer first-stage variables, pure binary second-stage variables, and discrete distributions. The formulation and computational results for this class of problems are reported in (Ahmed and Garcia, in press) and (Ahmed et al., in press), respectively.

SEMI4 is a two-stage multi-period stochastic integer problem that arises in planning semiconductor tool purchases. This model, which has mixed-integer first-stage variables and continuous second-stage variables, was solved by researchers at IBM (Barahona et al., 2001). Problem SIZES10 is an instance of a two-stage multi-period stochastic mixed-integer program arising in the product substitution applications. The problem formulation and data are given in (Jorjani et al., 1995).

Problem SSCh_c5 is an instance of a strategic supply chain planning problem under uncertainty with continuous and binary variables. This problem appears in (Alonso-Ayuso et al., 2003) as problem *c*5. SGAP_28 is an instance of a stochastic generalized assignment problem and is reported in (Albareda-Sambola et al., 2002) as problem instance number 28. The reformulation of this problem has both continuous and binary variables. Problem SVRP_100 is a stochastic vehicle routing problem (SVRP) in which the first stage decisions chart a route for each vehicle and the second stage calculates an expected penalty cost for not completing a route in case of excess demand (Laporte et al., 2002). The last instance in the table is a full recourse policy model for the air traffic flow management problem (TFMP) under uncertainty in airport arrival and departure and airspace due to weather conditions (Alonso et al., 2000).

An examination of the data in Table I reveals that the total number of integer variables in these SCO problems is not more that 20,000 for the DEP formulation. One may consider SCO instances of this size as representing the current state of the art.

4. A "Leap:" Computations with Million Variables

In this section, we report our computational experience in using the D^2 algorithm to solve instances whose DEP formulations are an order-of-magnitude larger than those discussed in the previous section. We compare our computational results with those obtained by the ILOG CPLEX 7.0 programming system (ILOG, 2000). It is widely recognized that the latter is among the more efficient commercial systems that implement B&B (backtracking search).

4.1. PROBLEM INSTANCE GENERATION

A number of instances of problem (2–3) were generated randomly as follows. Problem data were randomly generated from the uniform distribution while scenario data were generated from the Bernoulli distribution. The server location costs were generated randomly from the uniform distribution in the interval [40, 80] and the client demands were generated in the interval [0, 25]. The client-server revenue were set at one unit per unit of client demand. The overflow costs q_{j0}, for all $j \in \mathcal{J}$, were fixed at 1000, which was a high enough penalty cost to warranty no overflows in the optimal solution. The maximum number of servers to locate (v) was set to the total number of potential locations in the problem instance.

The scenario data were generated as follows. The availability of a potential client in each scenario was generated from the Bernoulli distribution with $p = 0.5$, with a 1 indicating the presence (availability) of the client and a 0 indicating the absence (unavailability) of the client. For each problem instance the different sets of scenarios were generated using different random seeds to allow for independent scenarios. Each scenario was given an equal probability of occurrence and contained the outcomes for all the clients in the problem instance. All scenarios were checked to make sure that there were no duplicate scenarios in a given problem instance.

The degree of difficulty of an instance can be controlled by the ratio (r) of the total server capacity to the maximum possible total demand. This ratio is defined by $r = v \cdot u / \sum_{i \in \mathcal{I}} \mathrm{Max}_{j \in \mathcal{J}}\{d_{ij}\}$, where the numerator is the total server capacity and the denominator is the total maximum demand. It reflects how much total server capacity is available to satisfy possible maximum overall client demand. A value of $r \geqslant 1.0$ means that the servers can satisfy the total client demand while a value of $r < 1.0$ implies that server capacity may be insufficient to satisfy client demand. Instances in which the server capacity is highly limited, piecewise linear overflow costs may be more appropriate.

As a mnemonic, the instances were named SSLP_*m*_*n*, where m is the number servers and n the number of clients. The number of servers ranged through $m = 5$, 10, and 15 while the number of clients were set at $n = 25$, 30, 45, and 50. The number of scenarios considered range from $S = 5$ to 2000. In particular, we report results on the problem instances SSLP_5_25, SSLP_5_50, SSLP_10_50, and SSLP_15_45, and briefly mention about the other instances.

4.2. COMPUTATIONAL RESULTS

The D^2 algorithm was implemented in C, with all small models (LP and MIP) solved by using the ILOG CPLEX 7.0 (ILOG, 2000) callable library. As a benchmark we applied the CPLEX MIP solver to the large scale DEP formulation (1) for each of the two-stage problem instances with the CPLEX parameters set at the following values: “set mip emphasis 1” (emphasizes looking for feasible solutions), “set mip strategy start 4” (uses barrier at the root), and “branching priority order on x” (first branches on any fractional component of x before branching on y). A CPU time limit of 10,000 s was imposed and any problem instance run taking more than this time limit was considered a “failure.” All the problems that took less than this time limit converged to an optimal solution and the percentage gap between the lower bound and the upper bound was equal to 0%. In all our computational experiments the 0-1 master programs were solved to optimality at each iteration.

All the experiments were run on a Sun 280R with 2 UltraSPARC-III+ CPUs running at 900 MHz. The results for the first set of experiments are summarized in Tables II–V. The numbers of variables and constraints shown are that of the DEP formulation. While most of the column headings in the tables are self-explanatory, we should clarify the term % Z_{IP} Gap. Entries in this column indicate the percentage difference between the optimal value of the SCO instance, and its continuous relaxation (which can be solved using stochastic linear programming). Another observation from the tables is that the number of D^2 cuts added to the second stage SMIP is less than the number of algorithmic iterations. This is because D^2 cuts are not generated in those iterations where all second stage subproblem relaxations yield binary solutions. Finally, each CPU time

Table II. Computational results for problem instance SSLP_5_25

Scenarios	Binaries	Constraints	% Z_{IP} gap	D^2 iterations	D^2 cuts	CPU time (secs)	
						D^2	CPLEX
5	630	151	33.11	16	1	0.13	0.12
10	1255	301	21.25	17	5	0.22	0.46
25	3130	751	23.47	17	10	0.42	1.82
50	6255	1501	24.03	17	6	0.53	4.58
100	12,505	3001	24.93	17	10	1.03	14.69

Table III. Computational results for problem instance SSLP_5_50

Scenarios	Binaries	Constraints	% Z_{IP} gap	D^2 iterations	D^2 cuts	CPU time (secs)	
						D^2	CPLEX
5	1255	276	21.75	28	6	0.52	0.30
10	2505	551	22.37	26	4	0.50	0.79
25	6255	1376	22.57	26	4	0.58	3.52
50	12,505	2751	20.74	33	11	1.64	10.35
100	25,005	5501	20.48	32	13	3.95	33.25

Table IV. Computational results for problem instance SSLP_10_50

Scenarios	Binaries	Constraints	% Z_{IP} gap	D^2 iterations	D^2 cuts	CPU time (secs)		CPLEX % gap
						D^2	CPLEX	
5	2510	301	10.49	209	189	78.25	80.53	
10	5010	601	11.38	264	257	171.49	Failed	0.19
25	12,510	1501	10.81	286	281	248.81	Failed	0.34
50	25,010	3001	10.89	252	250	295.95	Failed	0.44
100	50,010	6001	11.07	300	299	480.46	Failed	9.02
500	250,010	30,001	10.75	309	307	1902.20	Failed	38.17
1000	500,010	60,001	11.07	322	321	5410.10	Failed	99.60
2000	1,000,010	120,001	11.01	308	307	9055.29	Failed	46.24

shown in the tables records an average of three runs for the problem instance.

The experimental results for problems SSLP_5_25 and SSLP_5_50 are given in Tables II and III, respectively. Note that for all problem instances, the gap between the SLP objective of the DEP and the SMIP objective is over 20%. Hence the stochastic linear programming (continuous) relaxation of these SSLP instances does not provide very good approximations, and combinatorial optimization becomes necessary. The D^2 algorithm performs better than CPLEX for all the problem instances except the smallest instance of the second problem. We expect CPLEX to perform better on smaller problem instances since there is some overhead in decomposing small sized problems.

Table IV shows the results for the problem SSLP_10_50, which is much larger and takes substantially much more time to solve. As expected

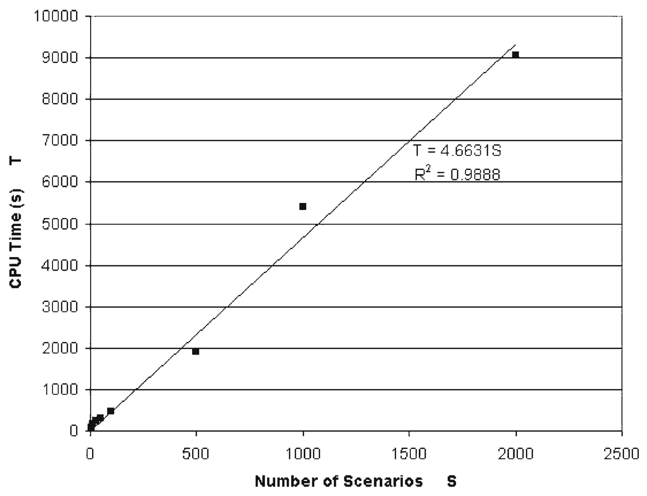


Figure 2. CPU time for SSLP_10_50 using the D^2 method.

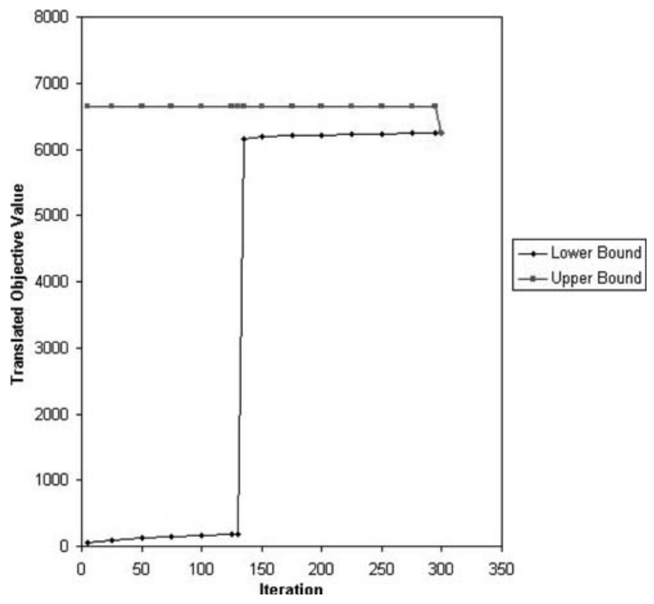


Figure 3. Convergence of the D^2 algorithm for problem instance SSLP_10_50 with 100 scenarios.

CPLEX has the smallest CPU time on the smallest instance but fails to solve the rest of the problems. On the other hand, the D^2 method solves all the problems in a reasonable amount of time. The last problem instance in the table is the largest and has 1,000,010 variables and 120,010 constraints! As shown in Figure 2 the performance of the D^2 algorithm is linear with increasing problem size. This is a desired algorithmic behavior for seal-ability. We also got similar results by increasing the number of clients to 75 (problem SSLP_10_75) but the computation times are about 1.4 times larger on average for all the scenarios.

Figure 3 shows a typical graph of convergence of upper and lower bounds when applying the D^2 method to SSLP_10_50. The results shown are for problem instance SSLP_10_50 with 100 scenarios. As can be seen in the figure, the lower bound increases close to the optimal value in less than half the total number of iterations. However, good upper bounds are calculated only after first-stage solutions stabilize and this causes the method to continue for the remaining iterations without changing the lower bound significantly. Once no changes are detected in the first-stage solution, a good upper bound is calculated by solving the MIP subproblems. This immediately lowers the upper bound. Moreover, a cut proposed in (Laporte and Louveaux, 1993) is added without any additional computation, and the method typically stops after this iteration. For smaller instances however (e.g., SSLP_5_25), the D^2 cuts are sufficient to provide linear relaxations which yield binary solutions for all scenarios. As the size

Table V. Computational results for problem instance SSLP_15_45

Scenarios	Binaries	Constraints	% Z_{IP} gap	D^2 iterations	D^2 cuts	CPU time (secs)		CPLEX % gap
						D^2	CPLEX	
5	3390	301	6.88	146	145	110.34	Failed	1.19
10	6765	601	6.53	454	453	1494.89	Failed	0.27
15	10,140	901	5.62	814	813	7210.63	Failed	0.72

Table VI. Problem instance SSLP_10_50 with 100 scenarios for different values of r

Ratio r	% Z_{IP} gap	D^2 iterations	D^2 cuts	D^2 time
0.90	3.35	618	617	7896.97
1.00	6.03	543	542	5296.02
1.25	8.38	348	347	900.77
1.50	11.07	300	299	480.46
1.75	14.19	236	227	240.85
2.00	17.45	243	198	207.09

of m and n increase, it becomes difficult for the linear relaxation to provide binary solutions, and solving the MIP becomes necessary to improve the upper bound.

In Table V we report the results for problem SSLP_15_45 with the number of scenarios ranging over the set 5,10 and 15. These instances are evidently much more challenging to solve since CPLEX could not even solve the smallest instance. The D^2 algorithm solves all the problem instances but takes much longer than the time it takes to solve instances of problem SSLP_10_50. These results show that while an increase in the number of scenarios do not affect the scalability of the D^2 algorithm in an adverse way, increases in the size of the master problem, (2a)–(2d), as well as the size of the subproblems, (3a)–(3e), do have an adverse effect on scalability.

Table VI gives the results obtained for an experiment aimed at studying the effect of the ratio of the total server capacity to the total maximum client demand on the problem computational difficult. In particular, we consider the performance of the D^2 algorithm on the problem instance SSLP_10_50 with 100 scenarios for different values of r ranging from 0.9 to 2.0. The CPLEX solver could not solve any of the DEP instances and so we excluded the results from Table VI.

Decreasing r results in increased computational times, an indication that tightly constrained instances are more computatioanlly demanding. The D^2 algorithm solves all the problem instances, but it takes substantially longer to solve the instances with $r \leqslant 1.00$. Despite the fact that the instances associated with $r = 1$, $r = 0.9$ resulted in instances with smaller gaps between an SLP relaxation, and the SMIP instance, tighter capacity constraints lead to several iterations in which the first stage solution leads to overflows in the second stage. Therefore, the algorithm has to overcome this "infeasibility" by generating possibly more first stage solutions (implying more algorithmic iterations) before converging to the optimal solution.

5. Conclusions

This paper presents computational results with the largest stochastic combinatorial optimization (SCO) instances to date. While the methods discussed in this paper are applicable to a variety of SCO problems, our computational results are presented for the stochastic server location problem (SSLP). These and other SCO problems result in very large scale instances which are comprised of loosely coupled subsystems. By taking advantage of the loosely coupled structure of SCO problems, we show that the divide-and-conquer paradigm of decomposition-coordination methods provide a highly effective algorithm, and surpasses the scalability of even the most efficiently implemented backtracking search algorithms. Notwithstanding the success reported here, the challenge of solving more general SMTP problems still remains (Sen, 2003). The design and analysis of approximations/heuristics for large scale SCO also remains an open research area (Stougie and van der Vlerk, 2003).

Acknowledgments

This research was funded by grants from the OR Program (DMII-9978780), and the Next Generation Software Program (CISE-9975050) of the National Science Foundation. We would like to thank Bob Bixby for confirming our experience with CPLEX and guiding the choice of parameter settings. The finishing touches on this paper were completed during the second author's visit to Carisma, Brunel University, under an EPSRC grant to Gautam Mitra. This collaboration is gratefully acknowledged. We are grateful to Bob Doverspike (AT&T Research), Madhav Marathe (Los Alamos National Labs), David Morton (University of Texas, Austin) and Maarten van der Vlerk (University of Groningen, Netherlands) for their comments on an earlier version of the paper.

References

Ahmed, S. and Garcia, R. (in press), Dynamic capacity acquisition and assignment under uncertainty, *Annals of Operational Research*.

Ahmed, S., Tawarmalani, M. and Sahinidis, N.V. (in press), A finite branch and bound algorithm for two-stage stochastic integer programs, *Mathematical Programming*. http://www.isye.gatech.edu/so/publications/.

Albareda-Sambola, M., van der Vlerk, M.H. and Fernandez, E. (2002), Exact solutions to a class of stochastic generalized assignment problems. *Research Report 02A11*, SOM, University of Groningen, The Netherlands, http://som.rug.nl.

Alonso, A., Escudero, L.F. and Ortuño, M.T. (2000), A stochastic 0-1 program based approach for the air traffic flow management problem. *European Journal of Operations Research* 120, 47–62.

Alonso-Ayuso, A., Escudero, L.F., Garín, A., Ortuño, M.T. and Perez, G. (2003), An approach for strategic supply chain planning under uncertainty based on stochastic 0–1 programming. *Journal of Global Optimization* 26, 97–124.

Balas, E. (1979), Disjunctive programming. *Annals of Discrete Mathematics* 5, 3–51.

Barahona, F., Bermon, S., Gunluk, O. and Hood, S. (2001), Robust capacity planning in semiconductor manufacturing. *IBM Research Report RC22196*, IBM.

Birge, J.R. and Louveaux, F.V. (1997), *Introduction to Stochastic Programming*, Springer, New York.

Cook, W.J., Cunningham, W.H., Pulleyblank, W.R. and Schrijver, A. (1998), *Combinatorial Optimization*, John Wiley and Sons, New York.

Doverspike, R.D. (2003), Private communication.

ILOG, IL. (2000), *CPLEX 7.0 Reference Manual*, ILOG CPLEX Division, Incline Village, NV.

Jorjani, S., Scott, C.H. and Woodruff, D.L. (1995), Selection of an optimal subset of sizes. *Technical report*, University of California, Davis, CA.

Laporte, G. and Louveaux, F.V. (1993), The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters* 1, 133–142.

Laporte, G., Louveaux, F.V. and van Hamme, L. (2002), An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands, *Operations Research* 50, 415–423.

Linderoth, J. and Wright, S.J. (2003), Decomposition algorithms for stochastic programming on a computational grid. *Computational Optimization and Applications* 24, 207–250.

Sen, S. (2003), Algorithms for stochastic mixed-integer programming models. In: Aardal, K., Nemhauser, G. and Weismantel, R. (eds.), *Stochastic Integer Programming Handbook*, Dordrecht, The Netherlands, Chapter 18. http://www.sie.arizona.edu/MORE/papers/SIPHbook.pdf.

Sen, S., Doverspike, R.D. and Cosares, S. (1994), Network planning with random demand. *Telecommunications Systems* 3, 11–30.

Sen, S. and Higle, J.L. (2000), The C^3 theorem and a D^2 algorithm for large scale stochastic integer programming: Set convexification. *Stochastic E-Print Series*. http://dochost.rz.hu-berlin.de/speps/.

Sen, S., Higle, J.L. and Ntaimo, L. (2002), A summary and illustration of disjunctive decomposition with set convexification. In: Woodruff, D.L. (ed.), *Stochastic Integer Programming and Network Interdiction Models*, Kluwer Academic Press, Dordrecht, The Netherlands, Chapter 6.

Stougie, L. and van der Vlerk, M.H. (2003), Approximation in stochastic integer programming. *Research Report 03A14,* SOM, Univ. of Groningen, The Netherlands. http://www.ub.rug.n1/eldoc/som/a/03A14/03A14.pdf.