Solving Chance-Constrained Programs combining Tabu Search and Simulation

R. Aringhieri.

Solving Chance-Constrained Programs combining Tabu Search and Simulation.

In C. C. Ribeiro and S. L. Martins, editors, Experimental and Efficient Algorithms, volume 3059 of Lecture Notes in Computer Science, pages 30-41. Springer-Verlag Heidelberg, 2004.

DOI: 10.1007/978-3-540-24838-5_3

Solving Chance-Constrained Programs combining Tabu Search and Simulation

Roberto Aringhieri[1]

DTI, University of Milan - via Bramante 65 I-26013 Crema
roberto.aringhieri@unimi.it
http://www.dti.unimi.it/~aringhieri

Abstract. Real world problems usually have to deal with some uncertainties. This is particularly true for the planning of services whose requests are unknown *a priori*.
Several approaches for solving stochastic problems are reported in the literature. Metaheuristics seem to be a powerful tool for computing good and robust solutions. However, the efficiency of algorithms based on Local Search, such as Tabu Search, suffers from the complexity of evaluating the objective function after each move.
In this paper, we propose alternative methods of dealing with uncertainties which are suitable to be implemented within a Tabu Search framework.

1 Introduction

Consider the following *deterministic linear program*:

$$\begin{aligned} \textbf{LP}: \min \quad & \sum_j c_j x_j & & \\ \text{s.t.} \quad & \sum_j a_{ij} x_j \leq b_i\,, & i = 1,\ldots,m & \quad (1a) \\ & x_j \geq 0, & j = 1,\ldots,n. & \quad (1b) \end{aligned}$$

The *cost* coefficients c_j, the *technological* coefficients a_{ij}, and the right-hand side values b_i are the problem parameters. In practical applications, any or all of these parameters may not be precisely defined. When some of these parameters are modelled as random variables, a stochastic problem arises.

When c_j, a_{ij} or b_i are random variables having a known joint probability distribution, z is also a random variable. When only the coefficients c_j are random, the problem can be formulated as the minimization of the *expected value* of z. Otherwise, a stochastic programming approach must be used. Two main variants of stochastic programming (see e.g. [1]) are the *stochastic programming with recourse* and the *chance-constrained programming*.

Charnes and Cooper [2] proposed to replace constraints (1a) with a number of probabilistic constraints. Denoting with $\mathbb{P}(.)$, the probability of an event, we

consider the probability that constraint i is satisfied, i.e. :

$$\mathbb{P}\left(\sum_j a_{ij}x_j \leq b_i\right).$$

Let α_i be the maximum allowable probability that constraint i is violated, a **chance-constrained programming** formulation of **LP**, say **CCP**, can be obtained by replacing (1a) with the following *chance constraints*:

$$\mathbb{P}\left(\sum_j a_{ij}x_j \leq b_i\right) \geq 1-\alpha_i\,, \quad i=1,\ldots,m. \tag{2}$$

When all c_j's are known, this formulation minimizes z while forbidding the constraints to exceed certain threshold values α_i.

Moving constraints (2) to the objective function via Lagrangean multipliers, we obtain the following stochastic program:

$$\begin{aligned} \textbf{SPR}: \min \quad & \sum_j c_j x_j + \sum_i \lambda_i\, \mathbb{P}\left(\sum_j a_{ij}x_j > b_i\right) && (3)\\ \text{s.t.} \quad & x_j \geq 0, && j=1,\ldots,n. \end{aligned}$$

Using the Lagrangean multipliers, **SPR** directly considers the *cost of recourse*, i.e. the cost of bringing a violated constraint to feasibility.

When the deterministic mathematical model involves also binary and/or integer variables, as in many real applications, the complexity of the associated stochastic program increases. To this purpose, several solution approaches are reported in literature such as the Integer L-Shaped Method [3], heuristics (see e.g. those for Stochastic Vehicle Routing Problem [4, 5]), methods based on *a priori optimization* [6] or *sample-average approximation* [7].

In this paper we propose a new algorithmic approach which combines Tabu Search and simulation for Chance-Constrained Programming. Glover and Kelly have described the benefits of applying Simulation to the solution of $\mathcal{NP}$-hard problems [8]. Tabu Search [9] is a well-known metaheuristic algorithm which has proved effective in a great number of applications.

The paper is organized as follows. The motivations and the basic idea of combining Tabu Search and Simulation are presented and discussed in section 2. In section 3 we introduce two optimization problems which are used to evaluate the efficiency of the proposed algorithms. Section 4 describes the two problems. Section 5 reports about the planning and the results of the computational experiments. Finally, ongoing work is discussed in section 6.

2 Motivations and Basic Ideas

In the following, we refer to the general model **IP** derived from **LP** setting the x_j's to be integer.

Tabu Search (TS) explores the solution space by moving at each iteration to the best neighbor of the current solution, even if the objective function is not improved. In order to avoid cycling, each move is stored in a list of *tabu* moves for a number of iterations: a tabu move is avoided until it is deleted from the list. A basic scheme of TS algorithm, say BTS, is depicted in Algorithm 1.

Algorithm 1 BTS

```
k := 1;
x := InitialSol();
while (not STOP) do
   N(x) := NeighborhoodOf(x);
   N^(k)(x) := N(x) \ T^(k)(x) ∪ A^(k)(x);
   x' := BestOf(N^(k)(x));
   x := x';
   k := k + 1;
end while
```

Note that: $T^{(k)}(x)$ is the set of *tabu solutions* generated by x using tabu moves at iteration k; $A^{(k)}(x)$ is the set of tabu solution which are evaluated since they respect some *aspiration criteria* (e.g. their objective function value improves that of current best solution). The algorithm usually stops after a given number of iterations or after a number of not improving iterations.

From a computational point of view, the computation of $N(x)$ and its evaluation (the choice of the best move) are the most time consuming components. For instance, a linear running time to evaluate the objective function of a single move is usually considered acceptable for a deterministic problem.

Unfortunately, this is not always true for stochastic programs. If we consider both **SPR** and **CCP** models, we observe that a move evaluation requires to compute a quite complex probability function. For instance, in [5], the authors proposed a **SPR** formulation for Vehicle Routing Problem with Stochastic Demands and Customers: the proposed TS algorithm, TABUSTOCH, requires at least $O(n^3)$ to evaluate a single move, where n is the number of demand locations. More generally, the evaluation of a new move involves probability and, at least, two stages of computation [1].

Our main concern is to reduce the computational complexity required for neighborhood exploration by introducing simulation methods within TS framework for solving a **CCP** programs.

The idea is based on a different way of dealing with random parameters: instead of computing directly the probability function, which is computationally expensive, we use simulation to evaluate random parameters. Then, we use these simulated random parameters, within the TS framework, in order to avoid moves which lead to unfeasible solutions, i.e. moves which make unfeasible the chance-constraints (2).

For the sake of simplicity, we assume that only the b_i's are random. Clearly, the following remarks can be extended straightforwardly to the other problem parameters. In order to simulate random parameters, we introduce the following notation:

$$\begin{aligned} x^{(k)} &= \text{ the variable } x \text{ at } k\text{-th iteration,} \\ \widetilde{b}_i^{(t)} &= \text{ the } t\text{-th simulated value of } b_i \quad (t = 1, \ldots, T), \\ \delta_i^{(k,t)} &= \widetilde{b}_i^{(t)} - \sum_j a_{ij} x_j^{(k)} \quad (t = 1, \ldots, T). \end{aligned}$$

Let $S_i^{(k)}$ be given as follows:

$$S_i^{(k)} = \sum_{t=1}^{T} S_i^{(k,t)}, \text{ where } S_i^{(k,t)} = \begin{cases} 1 & \text{if } \delta_i^{(k,t)} > 0 \\ 0 & \text{otherwise} \end{cases}. \tag{4}$$

The value of $S_i^{(k)}$ counts the number of *successes* for the i-th constraint (i.e. the constraint is satisfied) The value $\bar{S}_i^{(k)} = \dfrac{S_i^{(k)}}{T}$ estimates the probability of constraint i to be satisfied at iteration k

Taking into account **CCP** models, we are interested in computing solutions such that the chance-constraints (2) are satisfied for a given probability.

In this case, we can introduce a concept similar to that of a tabu move. The idea is to avoid all the moves leading to solutions which make unfeasible the respective chance-constraint. More formally, a move is *probably tabu* at iteration k if

$$\bar{S}_i^{(k)} < 1 - \alpha_i\,, \quad i = 1, \ldots, m. \tag{5}$$

Let $P^{(k)}(x)$ be the set of probably tabu solutions generated by x at iteration k. Then, the corresponding TS algorithm, say `SIMTS-CCP`, can be obtained from Algorithm 1 by modifying the computation of $N^{(k)}(x)$ as

$$\bar{N}^{(k)}(x) := N(x) \setminus T^{(k)}(x) \setminus P^{(k)}(x) \cup A^{(k)}(x). \tag{6}$$

The `SIMTS-CCP` procedure is sketched in Algorithm 2.

Finally, TS offers to the researchers a great flexibility. For instance, a common practice is to use a *simple* neighborhood structure and a penalized objective function to take into account the unfeasibility when some constraints are violated (see e.g. [10]). A general form of penalized function can be the following:

$$z + \sum_i \beta_i p_i(x) \tag{7}$$

where $\beta_i > 0$ is usually a self-adjusting penalty coefficient and $p_i(x) \geq 0$ is a measure of how much the i-th constraint is unfeasible.

In the same way, we can adapt the function in (7) to take into account the unfeasibility of chance-constraints. For instance, the $p_i(x)$ function for the i-th

Algorithm 2 SIMTS-CCP

$k := 1$;
$x :=$ `InitialSol()`;
while (`not STOP`) **do**
  $N(x) :=$ `NeighborhoodOf(`x`)`;
  $\bar{N}^{(k)}(x) := N(x) \setminus T^{(k)}(x) \setminus P^{(k)}(x) \cup A^{(k)}(x)$;
  $x' :=$ `BestOf(`$N^{(k)}(x)$`)`;
  $x := x'$;
  $k := k + 1$;
end while

constraint (2) can have the following general form:

$$p_i(x) = 1 - \alpha_i - \mathbb{P}\left(\sum_j a_{ij}x_j \leq b_i\right), \quad i = 1, \ldots, m. \tag{8}$$

3 Test Problems

In order to test the SIMTS-CCP algorithm,we will consider two $\mathcal{NP}$-hard optimization problems arising in the design of telecommunication networks based on SONET technology. For this class of problems, extensive computational experiences have been made and efficient tabu search algorithms are available [11].

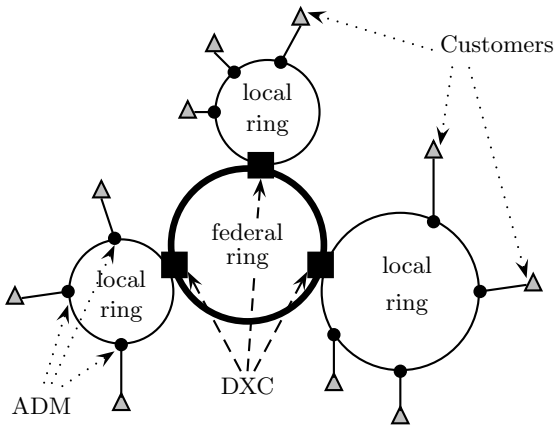

Fig. 1. A SONET network with DXC

The SONET network is a collection of rings connecting a set of customer sites. Each customer needs to transmit, receive and relay a given traffic with a subset of the other customers. Add-Drop Multiplexers (ADM) and Digital

Cross Connectors (DXC) are the technologies allowing the connection between customers and rings. Since they are very expensive, the main concern is to reduce the number of DXCs used.

Two main topologies for the design of a SONET network are available. The first topology consists in the assignment of each customer to exactly one ring by using one ADM and allowing connection between different rings through a unique *federal ring* composed by one DXC for each connected ring. The objective of this problem, say SRAP and depicted in Figure 1, is to minimize the number of DXCs.

Under some distance requirements, a second topology is possible: the use of a federal ring is avoided assigning each traffic between two different customers to only one ring. In this case, each customer can belong to different rings. The objective of this problem, say IDP and depicted in Figure 2, is to minimize the number of ADMs. For further details see e.g. [12, 11, 13].

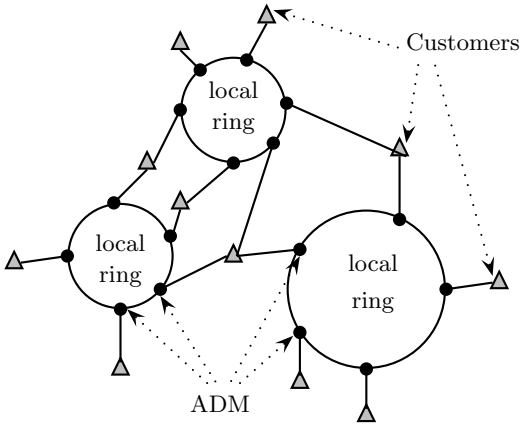

Fig. 2. A SONET network without DXC

In order to formulate these problems, we consider the undirected graph $G = (V, E)$: the node set V ($|V| = n$) contains one node for each customer; the edge set E has an edge $[u, v]$ for each pair of customers u, v such that the amount of traffic d_{uv} between u and v is greater than 0 and $d_{uv} = d_{vu}$, $\forall u, v \in V, u \neq v$. Given a subset of edges $E_i \subset E$, let $V(E_i) \subseteq V$ be the subset of nodes induced by E_i, i.e. $V(E_i) = \{u, v \in V : [u, v] \in E_i\}$.

SRAP formulation

Given a partition of V into r subsets $V_1, V_2, \ldots V_r$, the corresponding SRAP network is obtained by defining r local rings, connecting each customer of subset V_i

to the i-th local ring, and one federal ring, connecting the r local rings by using r DXCs. The resulting network uses n ADMs and r DXCs.

Solving SRAP corresponds to finding the partition $V_1, \dots V_r$ minimizing r, and such that

$$\sum_{u \in V_i} \sum_{\substack{v \in V \\ v \neq u}} d_{uv} \leq B, \quad i = 1, \dots, r \tag{9a}$$

$$\sum_{i=1}^{r-1} \sum_{j=i+1}^{r} \sum_{u \in V_i} \sum_{v \in V_j} d_{uv} \leq B \tag{9b}$$

Constraints (9a) and (9b) impose, respectively, that the capacity bound B is satisfied for each local ring and for the federal ring.

IDP formulation

Given a partition of E into r subsets $E_1, E_2, \dots E_r$, the corresponding IDP network can be obtained by defining r rings and connecting each customer of $V(E_i)$ to the i-th ring by means of one ADM. The resulting network uses $\varphi = \sum_{i=1}^{r} |V(E_i)|$ ADMs and no DXC.

Solving IDP corresponds to finding the partition $E_1, \dots E_r$ minimizing φ and such that

$$\sum_{[u,v] \in E_i} d_{uv} \leq B, \quad i = 1, \dots, r \tag{10}$$

Constraints (10) assure that the traffic capacity bound B for each ring is not exceeded. we finally remark that IDP has always a feasible solution, e.g. the one with $|E|$ rings composed by a single edge.

Stochastic formulations

The stochastic version of SRAP and IDP considers the demand d_{uv} as random parameters. The corresponding chance-constrained programs are obtained by replacing constraints (9a) and (9b) with

$$\mathbb{P}\left(\sum_{u \in V_i} \sum_{\substack{v \in V \\ v \neq u}} d_{uv} \leq B \right) \geq 1 - \alpha_i, \quad i = 1, \dots, r \tag{11a}$$

$$\mathbb{P}\left(\sum_{i=1}^{r-1} \sum_{j=i+1}^{r} \sum_{u \in V_i} \sum_{v \in V_j} d_{uv} \leq B \right) \geq 1 - \alpha_0 \tag{11b}$$

for SRAP, and constraints (10) with

$$\mathbb{P}\left(\sum_{[u,v] \in E_i} d_{uv} \leq B \right) \geq 1 - \alpha_i, \quad i = 1, \dots, r \tag{12}$$

for IDP.

4 The Algorithms for SRAP and IDP

In [11] the authors proposed a short-term memory TS guided by a variable objective function: the main idea of the variable objective function z_v is to lead the search within the solution space from unfeasible solutions to feasible ones, as in Table 1.

Table 1. Variable objective function z_v

	SRAP			IDP	
	feas. sol	not feas. sol		feas. sol	not feas. sol
feas. sol	$k\,B + BN$	$(k+1)\widetilde{BN}$	feas. sol	$\varphi\,B + M$	$2\varphi\,\widetilde{M}$
not feas. sol	$k\,B$	$n\,\widetilde{BN}$	not feas. sol	$\varphi\,B$	$2\varphi\,\widetilde{M}$
where	BN is the maximum value between maximum ring and federal ring capacities				
	$\widetilde{BN}$ is the value of BN associated to an unfeasible solution				
	M is the maximum ring capacity value				
	$\widetilde{M}$ is the value of M associated to an unfeasible solution				

A diversification strategy is implemented by varying multiple neighborhoods during the search. More specifically, `DMN` uses mainly a neighborhood based on moving one customer or demand at a time in such a way that the receiving ring does not exceed the bound B. Otherwise, if B is exceeded, we consider also the option of switching two customers or demands belonging to two different rings. After Δ consecutive non improving iterations, a second neighborhood is used for few moves. During this phase, `DMN` empties a ring by moving its elements (customers or demands respectively for SRAP and IDP) to the other rings disregarding the capacity constraint while locally minimizing the objective function.

To turn the computation of each move efficient, some data structures, representing the traffic within and outside a ring, are maintained along the computation.

The whole algorithm is called Diversification by Multiple Neighborhoods, say `DMN`. For a more detailed description of the whole algorithm, refer to the description given in [11].

The `simts-ccp` Algorithms

In order to devise the `SIMTS-CCP` algorithms for our problems, we need to implement the evaluation of chance-constraints through the generation of random parameters d_{uv}.

As described in section 2, we need T observations of $S_i^{(k,t)}$ to evaluate the value of $\bar{S}_i^{(k)}$ defined in (4). Moreover, each $S_i^{(k,t)}$ needs the generation of all d_{uv} traffic demands values according to their probability distribution function.

Algorithm 3 Computation of $\bar{S}_i^{(k)}$

for all $t = 1, \ldots, T$ **do**
  generate random d_{uv}, $\forall u, v \in V, u \neq v$;
  for $i = 1, \ldots, r$ **do** compute $\delta_i^{(k,t)}$;
end for
for $i = 1, \ldots, r$ **do** compute $\bar{S}_i^{(k)}$;

Algorithm 3 describes, with more details, the evaluation of $\bar{S}_i^{(k)}$. Since the number of traffic demands d_{uv} is $O(n^2)$, the complexity of the computation of $\bar{S}_i^{(k)}$ values is $O(T \times n^2)$. Note that the complexity of $\delta_i^{(k,t)}$ can be reduced employing the current traffic values available in the traffic data structures.

Here we propose two `SIMTS-CCP` algorithms derived from `DMN`. The basic idea is to add the computation of $\bar{S}_i^{(k)}$ to the original framework of `DMN` at the end of neighborhood exploration: starting from solution x, the algorithms generate each possible move x' as in `DMN` using the mean value of d_{uv}; then, the stochastic feasibility (respecting to the chance-constraints) is tested through the computation of $\bar{S}_i^{(k)}$. The algorithms differ in how the test is used to reject, or not, solution x'.

The first one, called `DMN-STOCH-1`, is the simplest one: each move not belonging to $\bar{N}^{(k)}(x)$, defined in (6), is avoided. In other words, `DMN-STOCH-1` allows only moves which satisfy the chance-constraints. Note that the objective function remains the one in Table 1.

Table 2. Penalized variable objective function $\bar{z}_v$

	SRAP			IDP	
	feas. sol	not feas. sol		feas. sol	not feas. sol
feas. sol	$z_v + Bs(x)$	$z_v + \widetilde{BN}s(x)$	feas. sol	$z_v + Bs(x)$	$z_v + \widetilde{M}s(x)$
not feas. sol	$z_v + Bs(x)$	$z_v + \widetilde{BN}s(x)$	not feas. sol	$z_v + Bs(x)$	$z_v + \widetilde{M}s(x)$

On the contrary, the second one, say `DMN-STOCH-2`, allows moves in $P^{(k)}(x)$ but penalizes them using an objective function which also measures how unfeasible the chance-constraints are. Referring to the general form reported in (7) and (8), our penalized objective function $\bar{z}_v$ is depicted in Table 2 where $s(x) =$

$\sum_i p_i(x)$ and

$$p_i(x) = \begin{cases} 100(1 - \alpha_i - \bar{S}_i^{(k)}) & \text{if } \bar{S}_i^{(k)} < 1 - \alpha_i \\ 0 & \text{otherwise} \end{cases}.$$

5 Preliminary Computational Results

In this section, we report the planning of computational experiments and the preliminary results.

In our computational experiments, we used the well-known Marsaglia-Zaman generator [14, 15], say RANMAR. This algorithm is the best known random number generator available since it passes *all* of the tests for random number generators. The algorithm is a combination of a Fibonacci sequence (with lags of 97 and 33, and operation "*subtraction plus one, modulo one*") and an "*arithmetic sequence*" (using subtraction), which gives a period of 2^{144}. It is completely portable and gives an identical sequence on all machines having at least 24 bit mantissas in the floating point representation.

Results of the deterministic version

Table 3. DMN: best computational results.

			gaps				avg. time in ms	
	z	# opt	# 1	# 2	# 3	# >3	optimal	all
SRAP	z_v	118	0	0	0	0	60.4	60.4
IDP	z_v	129	23	2	0	0	808.9	846.4

Considering the set of 160 benchmark instances generated by Goldschmidt *et al.* [13], DMN solves to optimality all the 118 instances, which are known to be feasible for SRAP, with an average computing time of 60 mseconds. On the same benchmark but considering IDP, for which the optimal solution value is known only for 154 instances, DMN solves 129 instances to optimality, 23 instances with a gap of 1 and the remaining two instances with a gap of 2. The average computing time is 850 mseconds. The overall results are reported in Table 3.

Results of the stochastic version

We have tested our algorithms on the same benchmark varying the parameters in the following ranges: $T \in \{50, 75, 100\}$, $\alpha_i = \alpha \in \{0.3, 0.2, 0.1\}$ and maintaining unaltered those giving the best result for the deterministic version (see [11]).

Table 4. DMN-STOCH-1: best and worst computational results.

	type of		gaps				avg. time in ms	
	results	# opt	# 1	# 2	# 3	# >3	optimal	all
SRAP	best	103	3	3	4	5	341.4	397.1
SRAP	worst	99	3	8	1	7	412.3	441.3
IDP	best	105	27	8	4	10	3127.3	3301.5
IDP	worst	101	25	10	3	15	3108.2	3309.9

The comparisons are made with the optimal value of the deterministic version, that is the values obtained using the mean value of traffic demands. Our tests try to investigate how far the solution computed by the SIMTS-CCP is from the one computed by its deterministic version.

Table 5. DMN-STOCH-2: best and worst computational results.

	type of		gaps				avg. time in ms	
	results	# opt	# 1	# 2	# 3	# >3	optimal	all
SRAP	best	106	4	5	2	1	394.1	432.7
SRAP	worst	104	5	4	2	3	443.5	457.9
IDP	best	118	22	6	5	3	3212.3	3287.2
IDP	worst	108	25	8	4	9	3459.1	3501.4

The results, reported in Table 4 and 5, show the impact of $\bar{S}_i^{(k)}$ computation: although the increase in the average computation time is quite remarkable with respect the deterministic version, we observe that the quality of solutions computed by both algorithms is acceptable.

6 Conclusions and further works

The paper addresses the problem of solving chance-constrained optimization problems combining Tabu Search and Simulation. After a brief introduction to stochastic programming, the class of SIMTS-CPP algorithms is proposed. The reported computational results show that the solutions computed have a quality comparable to those computed by the deterministic version. Also the increase in the average running time is acceptable.

Further work will be mainly concerned with two topics. The first one is the extension of computational experiments regarding SIMTS-CCP. The second one concerns the study of a similar algorithm for **SPR** problems.

Acknowledgments

The author wish to thank Paola Valpreda and Roberto Cordone for their help during the proofreading process.

References

1. Birge, J., Louveaux, F.: Introduction to Stochastic Programming. Springer (1997)
2. Charnes, A., Cooper, W.: Chance-constrained programming. Management Science **6** (1959) 73–79
3. Laporte, G., Louveaux, F.V.: The Integer L-Shaped Method for Stochastic Integer Problems with Complete Recourse. Operations Research Letters **13** (1993) 133–142
4. Dror, M., Trudeau, P.: Stochastic vehicle routing with modified savings algorithm. European Journal of Operational Research **23** (1986) 228–235
5. Gendreau, M., Laporte, G., Seguin, R.: A Tabu Search heuristic for the Vehicle Routing Problem with Stochastic Demands and Customers. Operations Research **44** (1996) 469–447
6. Bertsimas, D., Jaillet, P., Odoni, A.R.: A Priori Optimization. Operations Research **38** (1990) 1019–1033
7. Linderoth, J., Shapiro, A., Wright, S.: The Empirical Behavior of Sampling Methods for Stochastic Programming. Technical report, Computer Science Department, University of Wisconsin-Madison (2001)
8. Glover, F., Kelly, J.: New Advanced combining Optimization and Simulation. (In: Airo 1999 Proceedings)
9. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Boston (1997)
10. Cordeau, J.F., Gendreau, M., Laporte, G., Potvin, J.Y., Semet, F.: A guide to vehicle routing heuristics. Journal of the Operational Research Society **53** (2002) 512–522
11. Aringhieri, R., Dell'Amico, M.: Comparing Intensification and Diversification Strategies for the SONET Network Design Problem. Technical report, DISMI (2003) submitted to Journal of Heuristics.
12. Aringhieri, R., Dell'Amico, M., Grasselli, L.: Solution of the SONET Ring Assignment Problem with capacity constraints. Technical Report 12, DISMI (2001) To appear in "Adaptive Memory and Evolution: Tabu Search and Scatter Search", Eds. C. Rego and B. Alidaee.
13. Goldschmidt, O., Laugier, A., Olinick, E.V.: SONET/SDH Ring Assignment with Capacity Constraints. Discrete Applied Mathematics (2003) 99–128
14. Marsaglia, G., Zaman, A.: Toward a Universal Random Number Generator. Technical Report FSU-SCRI-87-50, Florida State University (1987)
15. Marsaglia, G., Zaman, A.: A New Class of Random Number Generators. Annals of Applied Probability **3** (1991) 462–480