S-ACO: An Ant-Based Approach to Combinatorial Optimization under Uncertainty

Walter J. Gutjahr

Dept. of Statistics and Decision Support Systems, University of Vienna
walter.gutjahr@univie.ac.at,
http://mailbox.univie.ac.at/walter.gutjahr/

Abstract. A general-purpose, simulation-based algorithm S-ACO for solving stochastic combinatorial optimization problems by means of the ant colony optimization (ACO) paradigm is investigated. Whereas in a prior publication, theoretical convergence of S-ACO to the globally optimal solution has been demonstrated, the present article is concerned with an experimental study of S-ACO on two stochastic problems of fixed-routes type: First, a pre-test is carried out on the probabilistic traveling salesman problem. Then, more comprehensive tests are performed for a traveling salesman problem with time windows (TSPTW) in the case of stochastic service times. As a yardstick, a stochastic simulated annealing (SSA) algorithm has been implemented for comparison. Both approaches are tested at randomly generated problem instances of different size. It turns out that S-ACO outperforms the SSA approach on the considered test instances. Some conclusions for fine-tuning S-ACO are drawn.

1 Introduction

The application of exact or heuristic methods of combinatorial optimization to real-world problems often faces the difficulty that for a particular problem solution considered, there is uncertainty on the objective function value achieved by it. A traditional way to represent uncertainty is by using a stochastic model. Based on such a model, the objective function becomes dependent not only on the solution, but also on a random influence, i.e., it becomes a random variable. Most frequently, the practical aim is then to optimize the *expected value* of this random variable.

When the expected value of the objective function can be represented as an explicit mathematical expression or at least be easily computed numerically, the solution of the stochastic combinatorial optimization problem needs not to be essentially different from that of a deterministic problem: The stochastic structure is then encapsulated in the representation of the expected objective function, and (possibly heuristic) techniques of deterministic optimization can be used. Very often, however, it is only possible to determine *estimates* of the expected objective function by means of sampling or simulation. This is the starting point for the area of *simulation optimization*, which has been a topic of intense research for several decades, but seems to undergo an interesting and

challenging shift at present, the key features of which have recently been outlined by Michael C. Fu [8]. In Fu's opinion, there is a gap between research in stochastic optimization concentrated on sophisticated specialized algorithms on the one hand, and the current boom of integration of optimization routines into commercial simulation software packages, mainly based on metaheuristics such as Genetic Algorithms or Neural Nets, on the other hand. Fu argues that traditional stochastic optimization algorithms are often not (or not easily) adaptable to complex real-world simulation applications, whereas the mentioned metaheuristic approaches, in their stochastic variants, frequently lack methodological rigor and are not provably convergent.

As a promising candidate of a metaheuristic that holds the potential for an extension to simulation optimization purposes, Fu explicitly names the Ant Colony Optimization (ACO) metaheuristic approach as introduced by Dorigo and Di Caro [6]. To make ACO satisfy these expectations, one would have to fill the gap mentioned above by developing ACO algorithms for combinatorial optimization under uncertainty that are both broadly applicable and theoretically well-founded. The present article aims at a step in this direction by presenting first experimental results for S-ACO, an ACO-based general-purpose stochastic combinatorial optimization algorithm for which the convergence to the optimal solution has been shown in [11].

Before presenting the S-ACO approach, let us briefly refer to some alternative techniques. Traditional methods for optimization under uncertainty such as Stochastic Approximation or the Response Surface Method are not well-suited for an application in the context of *combinatorial* optimization. Problems that are both stochastic and combinatorial can, however, be treated by Sample Average Approximation [16], Variable-Sample Random Search Methods [14], the Stochastic Branch-and-Bound Method [18], the Stochastic Ruler Method [2], or the Nested Partition Method [19]. As approaches drawing from metaheuristic algorithms ideas, we mention Stochastic Simulated Annealing ([12], [1]) or Genetic Algorithm for Noisy Functions [9].

In the field of ACO, an early paper on a stochastic combinatorial optimization problem has been published by Bianchi, Gambardella and Dorigo [4], it investigates the solution of the Probabilistic Travelling Salesman Problem (PTSP). For the PTSP, an explicit formula for the expectation of the objective function value is known, so the chosen solution technique cannot be generalized to problems where sampling is necessary to obtain estimates of this expectation. We shall use, however, the PTSP as a benchmark for pre-tests with our general-purpose algorithm.

2 Problem Description and Cost Function Estimation

In the S-ACO approach, stochastic combinatorial optimization problems of the following very general form are considered:

$$\text{Minimize } F(x) = \mathrm{E}\left(f(x,\omega)\right) \quad \text{subject to} \quad x \in S. \tag{1}$$

Therein, x is the decision variable, f is the cost function, ω denotes the influence of randomness, E denotes the mathematical expectation, and S is a finite set of feasible decisions.

It is not necessary that $\mathrm{E}\,(f(x,\omega))$ is numerically computable, since it can be estimated by sampling: For this purpose, draw N random *scenarios* $\omega_1,\ldots,\omega_N$ independently from each other. A *sample estimate* is given by

$$\mathcal{E}F(x) \;=\; \frac{1}{N}\sum_{\nu=1}^{N} f(x,\omega_\nu) \;\approx \mathrm{E}\,(f(x,\omega)). \tag{2}$$

Obviously, $\mathcal{E}F(x)$ is an unbiased estimator for $F(x)$.

It should be mentioned that, contrary to its deterministic counterpart, problem (1) is typically nontrivial already for a very small number $|S|$ of feasible solutions: Even for $|S| = 2$, except when $F(x)$ can be computed directly, a nontrivial statistical hypothesis testing problem is obtained (see [18]).

3 Algorithms

3.1 The S-ACO Algorithm

In [11], the algorithm S-ACO indicated in Fig. 1 has been proposed for solving problems of type (1). S-ACO works based on the encoding of a given problem instance as a *construction graph* $\mathcal{C}$, a directed graph with a distinguished start node. (For examples, see section 4.) The stepwise construction of a solution is represented by a random walk in $\mathcal{C}$, beginning in the start node. Following the definition of the construction graph encoding given in [10], the walk must satisfy the condition that each node is visited at most once; already visited nodes are infeasible. There may also be additional rules defining particular nodes as infeasible after a certain partial walk has been traversed. When there is no feasible unvisited successor node anymore, the walk stops and is decoded as a complete solution for the problem. The conceptual unit performing such a walk is called an *ant*.

The encoding must be chosen in such a way that to each feasible walk in the sense above, there corresponds exactly one feasible solution. (The converse property that to each feasible solution there corresponds exactly one feasible walk is *not* required.) Since, if the indicated condition is satisfied, the objective function value is uniquely determined by a feasible walk, we may denote a walk by the same symbol x as a solution and consider S as the set of feasible walks.

When constructing a walk in the algorithm, the probability p_{kl} to go from a node k to a feasible successor node l is chosen as proportional to $\tau_{kl}\cdot\eta_{kl}(u)$, where τ_{kl} is the so-called *pheromone value*, a memory value storing how good step (k,l) has been in previous runs, and $\eta_{kl}(u)$ is the so-called *visibility*, a pre-evaluation of how good step (k,l) will presumably be, based on some problem-specific heuristic. $\eta_{kl}(u)$ is allowed to depend on the given partial walk u up to now. For the biological metaphors behind the notions "pheromone" and "visibility", we refer the readers to the basic texts on ACO, e.g., [7] [6].

In our stochastic context, the computation of the visibility values $\eta_{kl}(u)$ needs some additional explanation. The difficulty may arise that certain variables possibly used by a problem-specific heuristic are not known with certainty, because they depend on the random influence ω. This difficulty can be solved either by taking the *expected values* (with respect to the distribution of ω) of the required variables as the base of the visibility computation (these expected values are often directly given as model parameters), or by taking those variable values that result from a random scenario ω drawn for the current round.

Feasibility of a continuation (k, l) of a partial walk u ending with node k is defined in accordance with the condition above that node l is not yet contained in u, and none of the (eventual) additional rules specifies l as infeasible after u has been traversed.

As in some frequently used ACO variants for deterministic problems, we determine in each round a *round-winner*. In the stochastic context, we do this by comparing all walks that have been performed in this round on a *single* random scenario ω, drawn specifically for this round. We also experimented with determining the round winner based on *several* random scenarios, but this only increased the runtime and did not improve the solution quality.

In an ACO implementation for a deterministic problem, it is always reasonable to store the best solution seen so far in a special variable. A crucial difference to the deterministic case is that in the stochastic context, it is not possible anymore to decide with certainty whether a current solution x is better than the solution currently considered as the best found, $\hat{x}$, or not. We can only "make a good guess" by sampling: After a current round-winner x has been determined, x is compared with the solution considered currently as the overall best solution, $\hat{x}$. This is done based on a sample of N_m randomly drawn scenarios used by both solutions. Also these scenarios are round-specific, i.e., in the next round, new scenarios will be drawn. The larger N_m, the more reliable is the decision. The winner of the comparison is stored as the new "global-best" $\hat{x}$.

Both the solution $\hat{x}$ considered so far as global-best and the round-winner are reinforced on each of their arcs by pheromone increments. The parameters $c_1 > 0$ and $c_2 > 0$ in the algorithm determine the amount of pheromone increment on global-best and round-best walks, respectively. Experiments showed that c_2 should be chosen small compared to c_1, but a small *positive* c_2 produced better results than setting $c_2 = 0$. The parameters c_1 and c_2 are allowed to depend on the walks $\hat{x}$ resp. x; in particular, it is reasonable to choose them inversely proportional to the lengths of $\hat{x}$ resp. x, such that the overall amount of pheromone increment is constant. (In the construction graphs used in this paper, all walks have the same lengths, so c_1 and c_2 are chosen as constants.)

In [11], it has been shown that a slight modification of the algorithm S-ACO of Fig. 1 with $c_2 = 0$ (only global-best reinforcement) converges, on certain mild conditions, with probability one to the globally optimal solution of (1). The essential condition is that the sample size N_m grows at least linearly with the round number m. The mentioned modification consists in an extension of the pheromone update rule above by a rule that additionally uses a lower pheromone

bound (cf. [20]) of a certain type. In the present paper, we did not apply lower pheromone bounds.

Procedure S-ACO

set $\tau_{kl} = 1$ for all (k,l);
for round $m = 1, 2, \ldots$ {
  for ant $\sigma = 1, \ldots, s$ {
    set k, the current position of the ant, equal to the start node of $\mathcal{C}$;
    set u, the current walk of the ant, equal to the empty list;
    while (a feasible continuation (k,l) of the walk u of the ant exists) {
      select successor node l with probability p_{kl}, where

$$p_{kl} = \begin{cases} 0, \text{ if } (k,l) \text{ is infeasible,} \\ \tau_{kl} \cdot \eta_{kl}(u) \,/\, \left(\sum_{(k,r)} \tau_{kr} \cdot \eta_{kr}(u) \right), \text{ else,} \end{cases}$$

      the sum being over all feasible (k,r);
      set $k = l$, and append l to u;
    }
    set $x_\sigma = u$;
  }
  based on one random scenario ω, select the best walk x out of the walks $x_1, \ldots, x_s$;
  if $(m = 1)$ set $\hat{x} = x$; // $\hat{x}$ is the candidate for the best solution
  else {
    based on N_m random scenarios ω_ν, compute a sample estimate $\mathcal{E}(F(x) - F(\hat{x})) = \frac{1}{N_m} \sum_{\nu=1}^{N_m} (f(x, \omega_\nu) - f(\hat{x}, \omega_\nu))$ for the difference between the costs of x and $\hat{x}$,
    if $(\mathcal{E}(F(x) - F(\hat{x})) < 0)$ set $\hat{x} = x$;
  }
  evaporation: set $\tau_{kl} = (1 - \rho)\, \tau_{kl}$ for all (k,l);
  global-best reinforcement: set $\tau_{kl} := \tau_{kl} + c_1$ for all $(k,l) \in \hat{x}$;
  round-best reinforcement: set $\tau_{kl} := \tau_{kl} + c_2$ for all $(k,l) \in x$;
}

Fig. 1. Pseudocode S-ACO.

3.2 Stochastic Simulated Annealing

In order to be able to compare the performance of the S-ACO algorithm with that of an alternative approach, we also implemented the Stochastic Simulated Annealing (SSA) algorithm described in [12]. SSA follows the philosophy of a standard Simulated Annealing algorithm. The only difference is that each time an objective function evaluation is required, which is the case when a current solution x is to be compared with a neighbor solution y, the evaluation is based on a sample estimate with some sample size N. In our implementations, we tried to keep equal conditions for S-ACO and SSA. Therefore, in analogy to S-ACO, the comparative evaluation of x and y in SSA is done by means of

the sample estimate $\mathcal{E}(F(x) - F(y))$, which is used in SSA as the input Δ for the probabilistic acceptance rule. As S-ACO, also SSA requires growing sample sizes N to satisfy theoretical convergence conditions, so we applied comparable sample size growth schemata to both algorithms. For two reasons, we did not use advanced SA concepts as *reheating*: First, also our S-ACO implementation was kept very basic (in particular, neither visibility values nor pheromone bounds were used). Secondly, only little experience is available at present about how to apply more elaborate SA concepts to *stochastic* problems.

4 Experimental Results

4.1 Pre-Test on the PTSP

In order to get a first impression of the performance of S-ACO and an idea about suitable parameter choices, it seemed convenient to test the algorithm on a problem where, for comparison, the exact value of $F(x)$ can be determined, because a closed formula for the computation of the expectation in (1) exists. A good candidate for this purpose is the *Probabilistic Traveling Salesman Problem* (PTSP) introduced by Jaillet [15]. The PTSP consists in finding a fixed closed tour containing each of n customer nodes exactly once, such that the *expected* tour length is minimized, where uncertainty comes from the fact that each customer i has only a given probability p_i of requiring a visit. A realization of a tour only contains the subset of customers who actually require a visit, but the sequence in which these customers are visited is taken from the fixed a-priori tour x. For the PTSP, it is possible to compute the expected costs $F(x)$ directly, although by a somewhat clumsy formula containing double sums over double products. In [4], the homogeneous version of the PTSP where all p_i are equal has been investigated within an ACO context. We experimented with the more general inhomogeneous version, where the p_i may differ from each other.

We tested four different problem instances of sizes $n = 9$, 14, 29 and 256, respectively. The distance matrices $D = (d_{ij})$ were taken from diverse TSP benchmarks available in the Internet. These data were extended by probabilities p_i drawn uniformly at random in an interval $[\lambda, 1]$ with $\lambda = 0.3$, 0.4, 0.5 and 0.5, respectively, to obtain complete test instances. This part of the experiments was performed on a PC Pentium 933 Mhz, 256 MB RAM.

The 12 parameter combinations resulting from the following values were investigated for each problem instance: (i) number of ants: $s = 50$, 500, (ii) evaporation factor: $\rho = 0.05$, 0.01, (iii) visibility values: $\eta_{kl}(u) = 1/(d_{kl})^{\beta}$ with $\beta = 2$, 3, 4. For the PTSP experiments, we only applied global-best reinforcement, i.e., we chose $c_2 = 0$. The increment c_1 was chosen as 4ρ.

As the natural construction graph $\mathcal{C}$ of the S-ACO implementation for this problem, we took the complete graph on node set $\{1, \ldots, n\}$, where the nodes represent the customers. Node 1 was taken as the start node.

A sample scenario ω is obtained by making for each customer i a random decision (with probability p_i) whether (s)he requires a visit or not ($i = 1, \ldots, n$).

Let us briefly outline the main findings. The case $n = 14$ was the largest for which we were still able to compute the *exact* solution of the problem by the explicit formula, combined with complete enumeration. Using S-ACO, best results were achieved in this case by the parameter combination $s = 50$, $\rho = 0.05$ and $\beta = 2$. Already after 1 sec computation time, relatively good solutions (only 1.9 % worse than the optimal solution in the average) were obtained, with only moderate improvements later.

For the largest instance ($n = 256$), the optimal solution value is unknown. Best results were obtained by letting both $\beta = 2$ and the number $s = 50$ of ants unchanged, but decreasing ρ to the value 0.01, and increasing the number of rounds considerably: steepest decrements of the expected cost function were observed between 5 and 20 minutes after the start of the computation.

An important goal of the pre-test was an answer to the question whether sample size functions N_m growing linearly in m, as prescribed by theory, would turn out as useful. Indeed, the scheme $N_m = 50 + (0.0001 \cdot n^2) \cdot m$ yielded best results out of several alternatives, such as omitting the intercept 50, choosing factors 0.001 or 0.00001, or working with a sample size independent of m. The proportionality to n^2 has been chosen to parallel the space complexity $O(n^2)$ of the algorithm. Future work should address the question whether schemes N_m that are only sub-linear in m suffice to give good results.

4.2 The TSPTW with Stochastic Service Times

Our main experimental results concern the Travelling Salesman Problem with Time Windows and Stochastic Service Times (TSPTW-SST). Let us briefly recapitulate this NP-hard problem (cf. [5] and [13]):

As in the ordinary TSP, a set of customers $\{1, \ldots, n\}$ and a distance matrix $D = (d_{ij})$ are given. Distances are interpreted as driving times. Let us imagine that the travelling person is a service engineer. To each customer i, a time window $[a_i, b_i]$ can be assigned, indicating that customer i requests a visit by the service engineer starting at time t_i with $a_i \leq t_i \leq b_i$. Not every customer needs to have a time window for the visit. The service at customer i takes some time Y_i, where Y_i is a random variable with known distribution. After finishing the service at customer i, the service engineer drives to the next customer on the list given by the chosen permutation x of customers. The aim is to minimize the total driving time. As in the case of the PTSP, we restrict ourselves to the *fixed-routes* variant of the problem: The sequence of customers must be fixed in advance and cannot be changed when information on actual service times gets available.

We study a variant of this problem where time-window violations are possible, but penalized by two cost terms (which are added to the driving-time component of the objective function): If the service engineer arrives at customer i at a time t_i before time a_i, (s)he must wait until time a_i. This is penalized by a (low) cost factor C_w, multiplied by the waiting time $a_i - t_i$. If, on the other hand, the service engineer arrives too late, i.e., at a time t_i after time b_i, a (high) penalty C_t, multiplied by the tardiness $t_i - b_i$, is incurred.

4.3 Test Instance Generation

To perform a comparative test of S-ACO and SSA, we generated 20 problem instances of different sizes and different degrees of difficulty at random. In the case of each problem instance, n customer points were selected uniformly at random from a square. Distances were computed as Euclidean distances between these points. It was assumed that traversing an edge of the square takes 10 time units. For each customer, a random decision was made on wether or not to assign a time window, using a fixed probability p_{TW} for the existence of a time window. If a time window was assigned, its length was selected uniformly at random between 6 and 60 time units, and its start time was selected uniformly at random between time 0 and the maximum time such that the whole time window was still contained in an interval of 120 time units. The service time distributions were chosen as uniform distributions on the interval between 2 and 6 time units.

We experimented with the following parameter values: For n, we investigated the cases $n = 10$ and $n = 20$. For p_{TW}, we considered the cases $p_{TW} = 0.25$ and $p_{TW} = 0.50$ if $n = 10$, and the cases $p_{TW} = 0.20$ and $p_{TW} = 0.40$ if $n = 20$. For each of these four combinations, five random test instances were generated. For each of the 20 resulting test instances, we performed 20 test runs with each of the considered algorithms and their variants described below. The penalties for waiting time and tardiness were set to the values $C_w = 1$ and $C_t = 20$, respectively. In other words, it was assumed that the cost of waiting one time unit is considered equal to that of having to drive an additional time unit, and the cost of being tardy by one time unit is considered as equal to that of having to drive 20 additional time units.

4.4 Implementation Details and Outcome Evaluation

We chose the construction graph shown in Fig. 2. The start node is the leftmost node. A visit of node v_{ij} means that the ith visited customer is customer j. The additional constraint imposed on the walks in this solution encoding is that after some node v_{ij} has been visited, all nodes v_{kj} get infeasible for the rest of the walk. (An equivalent solution encoding has been used by Bauer et al. [3] and by Merkle and Middendorf [17] for scheduling problems.)

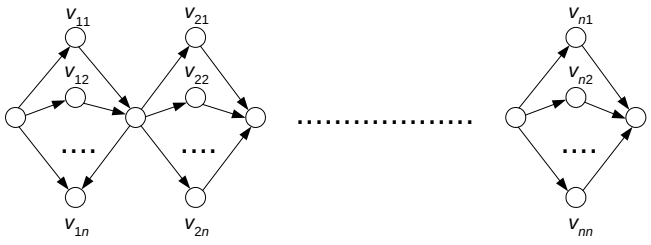

Fig. 2. Construction graph for the TSPTW-SST.

We did not use visibility values, i.e., we chose $\eta_{kl} = 1$.

A sample scenario ω is obtained by drawing service times independently for each customer according to the described uniform distribution, and $f(x, \omega)$ is evaluated by simulating the arrival and departure times connected with the actual tour according to the drawn service times.

Besides the standard implementations of S-ACO and SSA as described in the previous section, we also experimented with variants of each approach obtained by the following possible modifications: First, inspired by ideas in [8] and [14], we also implemented variants where the sample size is not increased according to a fixed scheme, but computed in an adaptive way: If two solutions x and $\hat{x}$ in the case of S-ACO or two solutions x and y in the case of SSA are to be compared, the sample size is gradually increased until the absolute value of the sample estimate of the difference between the two objective function values is larger than the triple standard deviation of this sample estimate. In other words, we take the sample size in any case just sufficiently large to reject (at a significance level of 99.93 %) the null hypothesis that both objective function values are identical. This saves sampling time in cases where the objective function values have a large difference anyway, so that it can be decided soon that one of them is better. The resulting modifications of S-ACO and SSA will be denoted by S-ACOa and SSAa, respectively.

Second, since it is well-known that metaheuristic tour optimization approaches can gain from postoptimization by local search, we considered variants where after the run of the corresponding general-purpose algorithm, a 2-opt local optimization procedure is applied to the resulting tour, where each function evaluation is based on a random sample of size 5000. These variants are denoted by appending the suffix "+ls".

The following parameter values were used: For S-ACO, we executed 2000 resp. 4000 rounds for $n = 10$ resp. $n = 20$. We set $s = 50$, $c_1 = 2\rho$, $c_2 = 0.02\rho$, $N_m = 2m$. Furthermore, we set $\rho = 0.005$ resp. $\rho = 0.002$ for $n = 10$ resp. $n = 20$. For SSA, we chose the initial resp. final value of the temperature parameter as 1000 resp. 0.0001, reduced the temperature parameter by 5 % at each iteration, executed $2n$ neighbor selections and accept/reject decisions at each temperature level, and chose the sample size N_t at the tth temperature level as $20t$. The sample size increments for S-ACO and SSA were designed in such a way that, for both algorithms, a final sample size in the same order of magnitude resulted.

To evaluate the solutions produced by the different approaches, a final brute-force simulation run with sample size $N = 10^6$ was applied to each of them. By computing variances, it was verified that in this way, an accuracy of about one digit after the decimal point could usually be reached. Only for the $n = 10$ problems it was possible to determine estimates of optimal solution values by complete enumeration (CE) within a feasible time (about 4 hours per test instance). The CE runs were based on a sample size of 10000 for each permutation, so there is no guarantee that the produced values are optimal in an exact sense, but they certainly give a fairly good estimate of the average difference between the optimal values and those produced by the metaheuristic approaches.

4.5 Results

Table 1 summarizes the results for the four instance parameter combinations and the eight metaheuristic variants. Each entry contains: (i) the achieved average cost function value, averaged over the 100 runs in total (5 random instances, each with 20 test runs), (ii) the computation time (in seconds) for a run, averaged over the 100 runs, (iii) the superiority counts explained below. The TSPTW-SST computations were performed on a PC Pentium 2.4 Ghz, 1 GB RAM.

In general, averaging across different problem instances is problematic since they may have incompatible ranges of cost values (cf. [21]). Note, however, that in Table 1, averaging has only been performed across different random instances within the same application class, say, $n = 10$ and $p_{TW} = 0.25$. Thus, each reported cost value has a clear interpretation as an estimate of the expected cost produced by the considered algorithm if it is applied to a problem instance randomly selected — according to the given distribution — from the given application class. By *not* re-normalizing the results within an application class, we give "hard" random test cases (which are of a high practical importance in view of robustness) their natural weight in comparison with "easy" test cases. Nevertheless, to present also output data that do not depend on aggregation, we have indicated in Table 1 for how many random test instances of a given application class the S-ACO variant performed better than the corresponding SSA variant, and vice versa. E.g., the "superiority count" 4 in the entry for S-ACO, $n = 10$ and $p_{TW} = 0.25$ indicates that S-ACO outperformed SSA in 4 of the the 5 test instances of this application class. By the parameter-free *sign test*, based on the total of the 20 test instances, we judged which S-ACO variant outperformed the corresponding SSA variant *significantly* more often than vice versa.

In total, the S-ACO variants produced better results than the SSA variants, both with and without adaptive sample size, and both in the case of local postoptimization and that without it. Quantified by the number of test instances for which a variant of S-ACO outperforms the corresponding SSA variant, this result is statistically significant at the level 0.05 for the comparisons S-ACOa / SSAa and S-ACO+ls / SSA+ls, significant even at the level 0.01 for the comparison S-ACOa+ls / SSAa+ls, but only weakly significant (level 0.10) for the comparison S-ACO / SSA.

Postoptimation by local search improved all variants. However, for the test cases with $n = 10$, SSA profited more from this postoptimization than S-ACO (possibly because the S-ACO variants came already rather close to the optimal solutions in this case). For the test cases with $n = 20$, S-ACO profited more from postoptimization than SSA.

Adaptive sample size did not improve the results in *each* case. For S-ACO, it did not even always decrease the runtime, at least not for the smaller test instances. For the larger test instances, adaptive sample size *did* save time, and it achieved about the same level of solution quality. SSA always profited from adaptive sample size both in solution quality and in runtime, except in the most "difficult" parameter instance combination, $n = 20$ and $p_{TW} = 0.40$, where a slightly worse solution quality resulted.

5 Conclusions

A general-purpose algorithm, S-ACO, for solving stochastic combinatorial optimization problems has been tested experimentally on two problems of fixed-routes type, a probabilistic travelling salesman problem (PTSP) and a travelling salesman problem with time windows and stochastic service times (TSPTW-SST). Encouraging results have been obtained. In particular, it has been shown that on the randomly generated test instances for the TSPTW-SST, the S-ACO algorithm outperformed a stochastic version of simulated annealing (SSA).

Future research should be directed to broader experiments with S-ACO on diverse other stochastic combinatorial problems in routing, scheduling, subset selection and many other areas, and to the experimental investigation of modifications of S-ACO as those outlined in [11], section 6. Furthermore, not only SSA, but also the other approaches indicated in the Introduction should be included into the comparative experiments. An especially challenging research topic for the future is the development of methods for analyzing stochastic combinatorial problems from a *multiobjective* point of view. Ongoing research investigates a possible combination of the S-ACO approach presented here with different ant-based multiobjective optimization techniques that have been developed in the ACO literature.

Acknowledgment. The author wants to express his thanks to Christian Grundner for his help in implementation and test of the PTSP part of the experiments.

Table 1. Average achieved cost values, runtimes (in sec), and superiority counts.

	$n = 10$ $p_{TW} = 0.25$			$n = 10$ $p_{TW} = 0.50$			$n = 20$ $p_{TW} = 0.20$			$n = 20$ $p_{TW} = 0.40$		
CE	52.8			57.0								
S-ACO	53.5	5	4	57.9	4	5	90.4	33	1	138.4	35	3
S-ACOa	53.6	8	4	57.9	10	5	89.2	19	2	139.8	21	2
S-ACO+ls	53.5	5	3	57.8	5	5	87.6	37	3	136.3	38	4
S-ACOa+ls	53.6	8	3	57.8	8	5	87.2	21	3	136.4	29	5
SSA	57.3	14	1	59.7	13	0	95.5	52	4	159.6	56	1
SSAa	57.1	7	0	59.7	5	0	91.3	15	2	160.6	14	3
SSA+ls	56.9	13	1	59.0	13	0	94.8	56	2	159.3	58	1
SSAa+ls	56.7	7	0	59.0	6	0	90.4	17	2	160.4	17	0

References

1. Alrefaei, M.H., Andradóttir, S., "A simulated annealing algorithm with constant temperature for discrete stochastic optimization", Management Sci. 45 (1999), pp. 748–764.

2. Alrefaei, M.H., Andradóttir, S., "A modification of the stochastic ruler method for discrete stochastic optimization", *European J. of Operational Research* 133 (2001), pp. 160–182.
3. Bauer, A., Bullnheimer, B., Hartl, R.F., Strauss, C., "Minimizing Total Tardiness on a Single Machine Using Ant Colony Optimization", *Central European Journal of Operations Research* 8 (2000), pp. 125–141.
4. Bianchi, L., Gambardella. L.M., Dorigo, M., "Solving the homogeneous probabilistic travelling salesman problem by the ACO metaheuristic", *Proc. ANTS '02*, 3rd Int. Workshop on Ant Algorithms (2002), pp. 177–187.
5. Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M.M., Sounis, F., "VRP with Time Windows", in: *The Vehicle Routing Problem*, P. Toth and D. Vigo (eds.), SIAM Monographs: Philadelphia (2002), pp. 157–194.
6. Dorigo, M., Di Caro, G., "The Ant Colony Optimization metaheuristic", in: *New Ideas in Optimization*, D. Corne, M. Dorigo, F. Glover (eds.), pp. 11-32, McGraw–Hill (1999).
7. Dorigo, M., Maniezzo, V., Colorni, A., "The Ant System: Optimization by a colony of cooperating agents", *IEEE Trans. on Systems, Man, and Cybernetics* 26 (1996), pp. 1–13.
8. Fu, M.C., "Optimization for simulation: theory vs. practice", *INFORMS J. on Computing* 14 (2002), pp. 192–215.
9. Fitzpatrick, J.M., Grefenstette, J.J., "Genetic algorithms in noisy environments", *Machine Learning* 3 (1988), pp. 101–120.
10. Gutjahr, W.J., "A graph–based Ant System and its convergence", *Future Generation Computer Systems* 16 (2000), pp. 873–888.
11. Gutjahr, W.J., "A converging ACO algorithm for stochastic combinatorial optimization", *Proc. SAGA 2003* (Stochastic Algorithms: Foundations and Applications), eds.: A. Albrecht and K. Steinhöfl, Springer LNCS 2827 (2003), pp. 10–25.
12. Gutjahr, W.J., Pflug, G., "Simulated annealing for noisy cost functions", *J. of Global Optimization*, 8 (1996), pp. 1–13.
13. Hadjiconstantinou, E., Roberts, D., "Routing Under Uncertainty: An Application in the Scheduling of Field Service Engineers", in: *The Vehicle Routing Problem*, P. Toth and D. Vigo (eds.), SIAM Monographs: Philadelphia (2002), pp. 331–352.
14. Homem-de Mello, T., "Variable-sample methods for stochastic optimization", *ACM Trans. on Modeling and Computer Simulation* 13 (2003), pp. 108–133.
15. Jaillet, P., *Probabilistic Travelling Salesman Problems*, PhD thesis, MIT, Cambridge, MA (1985).
16. Kleywegt, A., Shapiro, A., Homem-de-Mello, T., "The sample average approximation method for stochastic discrete optimization", *SIAM J. Optim.* 12 (2001), pp. 479–502.
17. Merkle, D., Middendorf, M., "Modelling the Dynamics of Ant Colony Optimization Algorithms", *Evolutionary Computation* 10 (2002), pp. 235–262.
18. Norkin, V.I., Ermoliev, Y.M., Ruszczynski, A., "On optimal allocation of indivisibles under uncertainty", *Operations Research* 46 (1998), pp. 381–395.
19. Shi, L., Olafsson, S., "Nested partition method for global optimization", *Operations Reseacrh* 48, pp. 390–407.
20. Stützle, T., Hoos, H.H., "The MAX-MIN Ant system and local search for the travelling salesman problem", in: T. Baeck, Z. Michalewicz and X. Yao (eds.), *Proc. ICEC '97* (Int. Conf. on Evolutionary Computation) (1997), pp. 309–314.
21. Zlochin, M., Dorigo, M., "Model-based search for combinatorial optimization: a comparative study", *Proc. PPSN* (Parallel Problem Solving from Nature) '02 (2002), pp. 651–664.