Adaptive Sample Size and Importance Sampling in Estimation-based Local Search for Stochastic Combinatorial Optimization: A complete analysis

Prasanna Balaprakash, Mauro Birattari, Thomas Stützle, and Marco Dorigo

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2007-015

September 2007

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2007-015

Adaptive Sample Size and Importance Sampling in Estimation-based Local Search for Stochastic Combinatorial Optimization: A complete analysis

Prasanna Balaprakash pbalapra@ulb.ac.be
Mauro Birattari mbiro@ulb.ac.be
Thomas Stützle stuetzle@ulb.ac.be
Marco Dorigo mdorigo@ulb.ac.be

IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium

September 10, 2007

Abstract

Metaheuristics and local search algorithms have received considerable attention as promising methods for tackling stochastic combinatorial optimization problems. However, in stochastic settings, these algorithms are usually simple extensions of the versions that are originally designed for deterministic optimization and often they lack rigorous integration with techniques that handle the stochastic character. In this paper, we discuss two generally applicable procedures that can be integrated into metaheuristics and local search algorithms that use Monte Carlo evaluation for estimating the solution cost. The first is an adaptive sampling procedure that selects the appropriate size of the sample to be used in Monte Carlo evaluation; the second is a procedure that adopts the importance sampling technique in order to reduce the variance of the cost estimator. We illustrate our approach and assess its performance using an *estimation-based* local search algorithm for the PROBABILISTIC TRAVELING SALESMAN PROBLEM. Experimental results show that an integration of the two procedures into the *estimation-based* local search increases significantly its effectiveness in cases where the variance of the cost estimator is high.

1 Introduction

Optimization problems that are both combinatorial and stochastic are important in a large number of practically relevant settings. Examples include portfolio management, vehicle routing, resource allocation, scheduling, and the modeling and simulation of large molecular systems (Fu, 2002). These problems are customarily tackled by considering the cost of each solution as a random variable and by finding a solution that minimizes some statistics of the cost. For a number of practical and theoretical reasons, the optimization is performed with respect to the expectation (Fu, 1994, 2002). The most widely used approach for this task is *empirical estimation*, where the expectation is estimated through Monte Carlo simulation.

In recent years, metaheuristics and local search algorithms that adopt the *empirical estimation* techniques emerged as the most promising approaches to tackle industrial and large scale stochastic combinatorial optimization problems (Fu, 1994, 2002). We refer the reader to Fu (1994) and Bianchi (2006) for surveys on solution techniques for stochastic combinatorial optimization problems. However, Fu (2002) and Gutjahr (2004) explicitly state that, although metaheuristics are widely used for practical applications, there is a huge gap between metaheuristics and research in specialized algorithms that handle the stochastic character of the problem. In particular, Fu (2002) forecasts that the effectiveness of metaheuristics and local search algorithms can be increased by a systematic integration of optimization algorithms and statistical procedures.

This paper focusses on techniques that aid local search algorithms to handle the stochastic character of the problem. More precisely, we adopt techniques that reduce the variance of the cost estimator, which are crucial for the effectiveness of the estimation techniques. We use two generally applicable procedures that can be integrated with a number of local search algorithms (such as iterative improvement, simulated annealing, iterated local search, tabu search, and variable neighborhood search), where the costs of the solutions are evaluated by *empirical estimation*. The first is an adaptive sampling procedure that selects the appropriate size of the sample with respect to the variance of the cost estimator; the second is a procedure that adopts the importance sampling technique in order to reduce the variance of the estimator when dealing with highly stochastic problem instances. We illustrate the methodology that we propose by integrating it into an *estimation-based* local search algorithm, `2.5-opt-EEs` (Birattari et al., 2007a). This is an iterative improvement algorithm that starts from some initial solution and then iteratively moves to an improving neighboring one until a local optimum is found. `2.5-opt-EEs` is an algorithm for tackling the PROBABILISTIC TRAVELING SALESMAN PROBLEM (PTSP) (Jaillet, 1985), a paradigmatic example of a stochastic combinatorial optimization problem.

The paper is organized as follows: In Section 2, we introduce the proposed approach; in Section 3, we study its performance and in Section 4, we conclude the paper.

2 Estimation-based iterative improvement algorithm for the PTSP

For the sake of completeness and in order to make this section self-contained, we first provide a short description of the PTSP and then we sketch the `2.5-opt-EEs` algorithm; finally, we describe the procedures introduced in this paper.

2.1 The probabilistic traveling salesman problem

The PROBABILISTIC TRAVELING SALESMAN PROBLEM (PTSP) is similar to the TSP with the difference that each node has a probability of requiring a visit. The *a priori* optimization approach (Bertsimas et al., 1990) for the PTSP consists in finding an *a priori* solution that visits all the nodes such that the expected cost of *a posteriori* solutions is minimized: The *a priori* solution must be found prior to knowing which nodes are to be visited; the associated *a posteriori* solution is computed *after* knowing which nodes need to be visited and it is obtained from the *a priori* solution by skipping the nodes that do not require to be visited and visiting the others in the order in which they appear in the *a priori* solution. Formally, the PTSP can be described as follows: Minimize $F(x) = E\big[f(x, \Omega)\big]$, subject to $x \in S$, where x is an *a priori* solution, S is the set of feasible solutions, the operator E denotes the mathematical expectation, and $f(x, \Omega)$ is the cost of the *a posteriori* solution that depends on a random variable Ω. In the PTSP, Ω is described by an n-variate Bernoulli distribution parameterized by P, where n is the number of nodes and $P = \{p_1, \ldots, p_n\}$ is a set of probabilities that for each node i specifies its probability p_i of requiring a visit. A realization ω of Ω is a binary vector of size n where a '`1`' in position i indicates that node i requires visit and a '`0`' indicates that it does not. See Figure 1 for an illustration.

The usage of effective *delta evaluation* procedures is of crucial importance for a fast local search for the PTSP. Currently, the state-of-the-art iterative improvement algorithms for the PTSP, namely, `2-p-opt` and `1-shift` use for the *delta evaluation* recursive closed-form expressions based on heavy mathematical derivations (Bertsimas, 1988; Chervi, 1988; Bertsimas and Howell, 1993; Bianchi et al., 2005; Bianchi, 2006; Bianchi and Campbell, 2007). Recently, we introduced a more effective algorithm called `2.5-opt-ACs`
(Birattari et al., 2007a). It also uses closed-form expressions but adopts the classical TSP neighborhood reduction techniques, which is not possible in `2-p-opt` and `1-shift` since these algo-

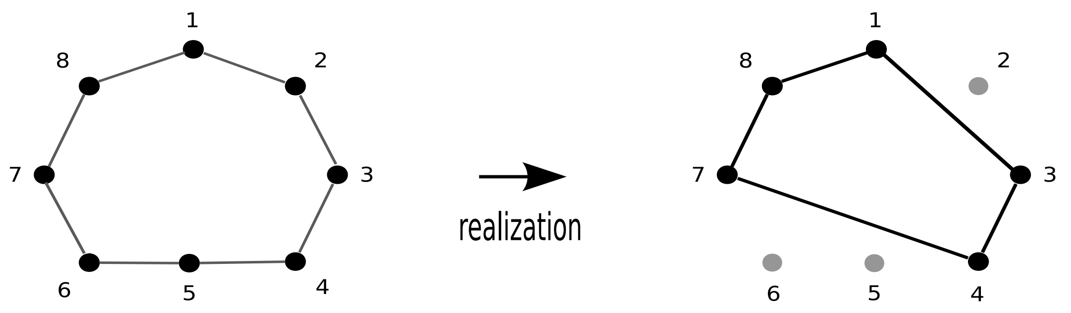


Figure 1: An *a priori* solution for a PTSP instance with 8 nodes. The nodes in the *a priori* solution are visited in following order: 1, 2, 3, 4, 5, 6, 7, 8, and 1. Assume that a realization of Ω prescribes that nodes 1, 3, 4, 7, and 8 are to be visited. The resulting *a posteriori* solution is obtained by visiting the nodes in the order in which they appear in the *a priori* solution and by skipping the nodes 2, 5, and 6, which do not require visit.

rithms require to search the neighborhood solutions in a fixed lexicographic order. `2.5-opt-EEs` (Birattari et al., 2007a), the algorithm on which we focus in this paper, makes use of *empirical estimation* techniques and the classical TSP neighborhood reduction techniques. The experimental results show that this approach is more effective than the state-of-the-art PTSP algorithms. However, highly stochastic PTSP instances, that is, instances where the probability of visiting nodes is very low, are not as effectively tackled by the *empirical estimation* technique as by the closed-form computations used in `2.5-opt-ACs`. The methodology that we propose in this paper addresses this issue.

2.2 The `2.5-opt-EEs` algorithm

In iterative improvement algorithms for the PTSP, we need to compare two neighboring solutions x and x' to select the one of lower cost. An unbiased estimator of $F(x)$ for a solution x can be computed on the basis of a sample of costs of *a posteriori* solutions obtained from M independent realizations of the random variable Ω (Gutjahr, 2004; Birattari et al., 2007a). Using the method of *common random numbers*, for x' an *unbiased* estimator of $F(x')$ can be estimated analogously to $F(x)$ using a same set of M independent realizations of Ω. The estimator $\hat{F}_M(x') - \hat{F}_M(x)$ of the cost difference is given by:

$$\hat{F}_M(x') - \hat{F}_M(x) = \frac{1}{M}\sum_{r=1}^{M}\Big(f(x',\omega_r) - f(x,\omega_r)\Big). \tag{1}$$

We implemented iterative improvement algorithms that use this way of estimating cost differences exploiting a neighborhood structure that consists of a *node-insertion neighborhood* on top of a *2-exchange neighborhood* structure, that is, the well-known *2.5-exchange neighborhood*: when checking for a *2-exchange* move on any two edges $\langle a, b\rangle$ and $\langle c, d\rangle$, it is also checked whether deleting any one of the nodes of an edge, say for example a, and inserting it between nodes c and d results in an improved solution (**?**)—see Figure 2. To make the computation of the cost differences as efficient as possible, given two neighboring *a priori* solutions and a realization ω, the algorithm needs to identify the edges that are not common to the two *a posteriori* solutions. These edges are found as follows: for every edge $\langle i, j\rangle$ that is deleted from x, one needs to find the corresponding edge $\langle i^*, j^*\rangle$ that is deleted in the *a posteriori* solution of x. We call this edge the *a posteriori edge*. It is obtained as follows. If node i requires visit, then $i^* = i$; otherwise, i^* is the first predecessor of i in x such that $\omega[i^*] = 1$, that is, the first predecessor for which the realization

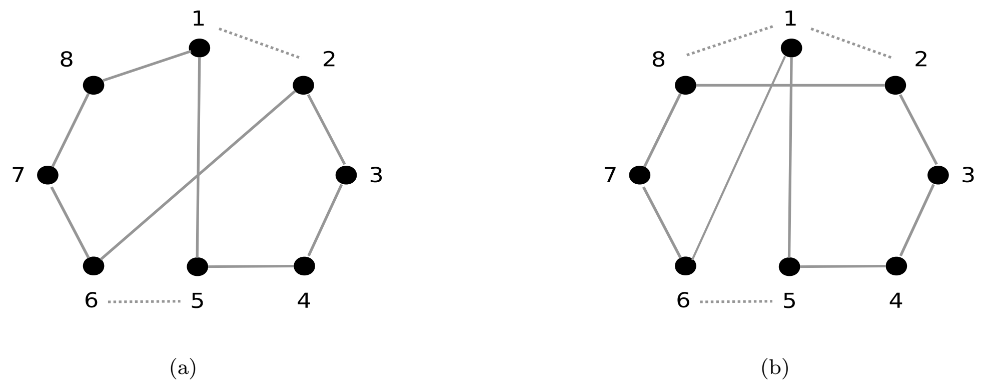


Figure 2: Plot 2(a) shows a *2-exchange* move that is obtained by deleting the two edges $\langle 1, 2\rangle$ and $\langle 5, 6\rangle$ of the solution and by replacing them with $\langle 1, 5\rangle$ and $\langle 2, 6\rangle$. Plot 2(b) shows a *node-insertion* move obtained by deleting node 1 from its current position in the solution and inserting it between nodes 5 and 6.

is one, indicating it requires visit. If node j requires visit, then $j^* = j$; otherwise, j^* is the first successor of j such that $\omega[j^*] = 1$. Recall that in a *2-exchange* move, the edges $\langle a, b\rangle$ and $\langle c, d\rangle$ are deleted from x and replaced by $\langle a, c\rangle$ and $\langle b, d\rangle$. For a given realization ω and the corresponding *a posteriori edges* $\langle a^*, b^*\rangle$ and $\langle c^*, d^*\rangle$, the cost difference between the two *a posteriori* solutions is given by $c_{a^*,c^*} + c_{b^*,d^*} - c_{a^*,b^*} - c_{c^*,d^*}$, where $c_{i,j}$ is the cost of edge $\langle i, j\rangle$. The procedure described can be directly extended to *node-insertion* moves. Furthermore, the algorithm adopts neighborhood reduction techniques such as *fixed-radius search*, *candidate lists*, and *don't look bits* (**??**Johnson and McGeoch, 1997). This algorithm is called `2.5-opt-EEs`. Note that `2.5-opt-EEs` uses the *first-improvement* rule. For a more detailed explanation, see Birattari et al. (2007a).

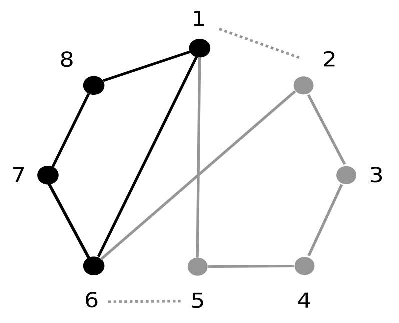
2.3 Advanced sampling methods

In this section, we focus on the main contribution of the paper, that is, the two procedures that we adopt in order to increase the effectiveness of `2.5-opt-EEs`.

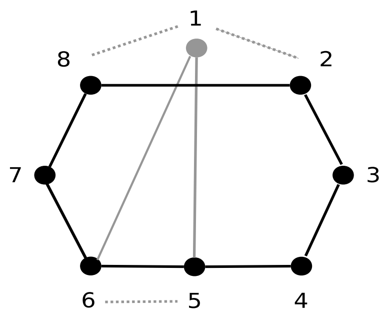
2.3.1 Adaptive sampling

Intuitively, the variance of the cost difference estimator depends on the probabilities associated with the nodes. The smaller the probability values, the higher the variance. A discussion about this issue is given in Section 3.1.2. In this case, averaging over a large number of realizations reduces the variance of the estimator. However, using a large number of realizations for high probability values is simply a waste of time. In order to address this issue, we adopt an adaptive sampling procedure that saves computation time by selecting the most appropriate number of realizations for each estimation. This procedure is realized using *Student's t-test* in the following way: Given two neighboring *a priori* solutions, the cost difference between their corresponding *a posteriori* solutions is sequentially computed on a number of realizations. As soon as the *t-test* rejects the null hypothesis that the value of the cost difference estimator is equal to zero, the computation is stopped. If no statistical evidence is gathered, then the computation is continued until a maximum number M of realizations is considered, where M is a parameter of the algorithm. The sign of the estimator determines the solution of lower cost.

The *estimation-based* iterative improvement algorithm that adds the adaptive sampling procedure to `2.5-opt-EEs` will be called `2.5-opt-EEas`.



(a) Assume that a realization of Ω prescribes that nodes 1, 6, 7, and 8 are to be visited. The *2-exchange* neighboring solutions shown in Figure 2(a) lead to the same *a posteriori* solution. The cost difference is therefore zero.



(b) Assume that a realization of Ω prescribes that nodes 2, 3, 4, 5, 6, 7, and 8 are to be visited. The *node-insertion* neighboring solutions shown in Figure 2(b) lead to the same *a posteriori* solution. Since the two *a posteriori* solutions are the same, the cost difference is zero.

Figure 3: Some degenerate cases that can occur in the evaluation of cost differences.

2.3.2 Importance sampling

A difficulty in the adoption of the *t-test* is that for low probability values often the test statistic cannot be computed: since the nodes involved in the cost difference computation may not require visit in the realizations considered, the sample mean and the sample variance of the cost difference estimator are null. Some degenerate cases in which this problem appears are the following. In a *2-exchange* move that deletes the edges $\langle a, b\rangle$ and $\langle c, d\rangle$, and where no node between the nodes b and c (or between a and d) requires visit, the difference between the two *a posteriori* solutions is zero—see Figure 3(a) for an illustration. In particular, this case is very frequent when the number of nodes between b and c (or between a and d) is small. In a *node-insertion* move, if the insertion node does not require visit, the cost difference between the two *a posteriori* solutions is zero—see Figure 3(b). A naïve strategy to handle this problem consists in postponing the *t-test* until non-zero sample mean and sample variance are obtained. However, this might increase the number of realizations needed for the cost difference computation. The key idea to address this issue consists in forcing the nodes involved in the cost difference computation to appear frequently in the realizations. More in general, we need to reduce the variance of the cost difference estimator for low probability values. For this purpose, we use the variance reduction technique known as *importance sampling* (Srinivasan, 2002).

In order to compute the cost difference between two *a posteriori* solutions, importance sampling, instead of using realizations of the given variable Ω parameterized by P, considers realizations of another variable Ω^* parameterized by P^*; this so-called biased distribution P^* biases the nodes involved in the cost difference computation to occur more frequently. This is achieved by choosing probabilities in P^* greater than the probabilities in P. The resulting biased cost difference between two *a posteriori* solutions for each realization is then corrected for the adoption of the biased distribution: the correction is given by the likelihood ratio of the original distribution with respect to the biased distribution and the unbiased cost difference is simply given by the product of the biased cost difference and the likelihood ratio.

The adoption of importance sampling can be made more effective: As illustrated in Figure 3, among all the nodes that are involved in the cost difference computation, few nodes are more important than others—in a *2-exchange* move, the nodes in the shorter segment; in a *node-insertion* move, the insertion node. We use importance sampling to bias these few important nodes in the following way:

- in a *2-exchange* move, importance sampling is used to bias the nodes between b and c, only if their number is less than min_{is}, where min_{is} is a parameter of the algorithm. The likelihood ratio is given by the product of the likelihood ratio of each node i between b and c, which has been used in finding *a posteriori edges*. The likelihood ratio of each node i is given by

$$LR_i^{2ex} = \frac{(p_i)^{\omega^{'}[i]} \cdot (1-p_i)^{1-\omega^{'}[i]}}{(p_i')^{\omega^{'}[i]} \cdot (1-p_i')^{1-\omega^{'}[i]}}, \tag{2}$$

where p_i' is the biased probability of node i to be used in a *2-exchange* move and $\omega^{'}[i]$ is sampled with probability p_i'.

- in a *node-insertion* move, importance sampling is used to bias only the insertion node i to appear in a realization and the likelihood ratio is given by

$$LR_i^{ins} = \frac{(p_i)^{\omega^{''}[i]} \cdot (1-p_i)^{1-\omega^{''}[i]}}{(p_i'')^{\omega^{''}[i]} \cdot (1-p_i'')^{1-\omega^{''}[i]}}, \tag{3}$$

where p_i'' is the biased probability of the insertion node i to be used in a *node-insertion* move and $\omega^{''}[i]$ is sampled with probability p_i''.

We denote `2.5-opt-EEais` the algorithm that adds to `2.5-opt-EEas` the above described importance sampling procedure.

2.4 Implementation-specific details

In order to implement `2.5-opt-EEais` efficiently, we use the same data structure as that of `2.5-opt-EEs`, which is composed of a doubly circularly linked list and some auxiliary arrays as described in Birattari et al. (2007a). Additionally, for each node three realization arrays, ω, ω', and ω'' are stored, each of size M, indexed from 1 to M. Element r of a realization array is either `1` or `0` indicating whether node i requires visit or not in a realization and it is obtained as follows: first a random number between 0 and 1 is generated; if this number is less than or equal to p_i, p_i', or p_i'', node i requires visit in realization ω_r, ω_r', or ω_r'', respectively. Moreover, if the biased probability p_i' is less than the original probability p_i, then p_i' is set to p_i. The same condition holds for p_i''. Given p_i, p_i', and p_i'' for a node i, the likelihood ratio is pre-computed and stored when the algorithm starts.

For low probability values, the computational results of the state-of-the-art local searches show that the *2-exchange* neighborhood relation is not effective (Bianchi, 2006; Birattari et al., 2007a,b). In order to alleviate this problem in `2.5-opt-EEais`, the way in which importance sampling is applied to a *2-exchange* move is slightly modified. Instead of biasing all the nodes between b and c when the number of nodes between them is less than min_{is}, only a certain number of nodes, w (a parameter), close to b and c are biased: Given the nodes from b to c as $[b, b', b'', \ldots, c'', c', c]$, if w is set to 1, then the nodes that are biased are b and c; if w is set to 2, then the nodes that are biased are b, b' and c, c', etc. The usage of min_{is} and w is illustrated in Figure 4.

`2.5-opt-EEais` uses a same set of realizations for all iterative improvement steps. In the context of the PTSP, this strategy is more effective than changing realizations for each improvement or for each comparison (Birattari et al., 2007a). However, the order in which the realizations are considered for the cost difference computation is randomly shuffled for each improvement.

The following techniques are used to speed up the computations involved in the *t-test*: the critical values of the *Student's t-distribution* are hand-coded and stored in a lookup table; the sample mean and the sample variance of the cost difference estimator are computed recursively.

The implementation of `2.5-opt-EEas` is similar to `2.5-opt-EEais` with the difference that the importance sampling procedure and pre-computations required for `2.5-opt-EEais` are excluded.

3 Experimental analysis

In this section, we present the experimental setting considered and the empirical results. We analyze the algorithms by classifying them in two groups: the *analytical computation based* algorithms

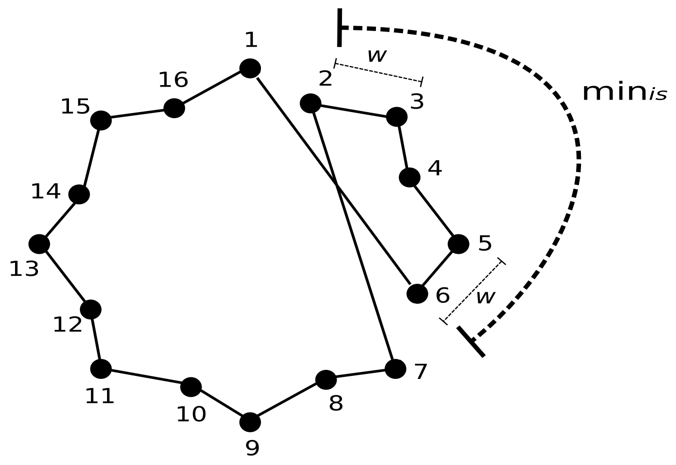


Figure 4: In this example, a *2-exchange* move is obtained by deleting the two edges $\langle 1, 2\rangle$ and $\langle 6, 7\rangle$ and by replacing them with $\langle 1, 6\rangle$ and $\langle 2, 7\rangle$. The parameter min_{is} is set to 8. Since the number of nodes in the segment $[2, \ldots, 6]$ is less than min_{is}, importance sampling is used to bias the nodes between them. However, instead of biasing all the nodes between 2 and 6, only a certain number of nodes that are close to them are biased. In this example, w is set to 2, therefore the nodes that are biased are 2, 3, 5, and 6.

and the *estimation-based* algorithms. We select the best from each group and we finally compare them. For what concerns the comparison of the different *estimation-based* algorithms, our goal is to show that the integration of the adaptive sample size and the importance sampling procedures into the *estimation-based* local search increases significantly its effectiveness—in particular, for the problem instances where the variance of the cost estimator is high. In Section 3.1, we describe the setup of the experiments; in Section 3.2 and 3.3, we present the results of the *analytical computation based* algorithms and the *estimation-based* algorithms, respectively; finally in Section 3.4, we show the effectiveness of the `2.5-opt-EEais` by comparing it with the state-of-the-art algorithm, `2.5-opt-ACs`.

3.1 Experimental setup

In this section, we present the experimental setting considered and the empirical results. We analyzed the algorithms by classifying them in two groups: the *analytical computation based* algorithms and the *estimation-based* algorithms. We selected the best from each group and we finally compared them. Our analysis is based on PTSP instances that we obtained from TSP instances generated with the DIMACS instance generator (Johnson et al., 2001). They are homogeneous PTSP instances, where all the nodes of an instance have a same probability p of appearing in a realization. We carried out experiments on clustered instances of 1000 nodes, where the nodes are arranged in a number of clusters, in a $10^6 \times 10^6$ square. We considered the following probability levels: $[0.050, 0.200]$ with a step size of 0.025 and $[0.3, 0.9]$ with a step size of 0.1. For a PTSP instance of size 1000, the algorithms `2.5-opt-ACs`, `2-p-opt`, and `1-shift` suffer from numerical problems for $p > 0.5$ (Birattari et al., 2007a). Therefore, these algorithms are examined only for probability values up to 0.5. Note that the algorithms based on *empirical estimation* do not suffer from this numerical problem.

All algorithms were implemented in `C` and the source codes were compiled with `gcc`, version 3.3. Experiments were carried out on AMD Opteron$^{\text{TM}}$244 1.75 GHz processors with 1 MB L2-Cache and 2 GB RAM, running under the Rocks Cluster GNU/Linux.

The nearest-neighbor heuristic is used to generate initial solutions. The candidate list is set to size 40 and it is constructed with the quadrant nearest-neighbor strategy (Penky and Miller, 1994; Johnson and McGeoch, 1997). Each iterative improvement algorithm is run until it reaches a local optimum.

In `2.5-opt-EEas` and `2.5-opt-EEais`, the minimum number of realizations used in the adaptive sampling procedure before applying the *t-test* is set to five. The null hypothesis is rejected at a significance level of 0.05. If the test statistic cannot be computed after five realizations, then the cost difference computation is stopped and the algorithm considers the next neighboring solution. The maximum number M of realizations is set to one thousand.

For the *homogenous* PTSP with probability p and size n, given an *a priori* solution x, the exact cost $F(x)$ of x can be computed using the formula $F(x) = \sum_{u=1}^{n} \sum_{v=1}^{n-1} p^2 (1-p)^{v-1} c_{(x(u),x(v))}$, where $x(u)$ and $x(v)$ are the nodes at index u and v in x, respectively (Jaillet, 1985). We use this formula only for post-evaluation purposes: for each algorithm, whenever an improved solution is found, we record the solution. In order to compare the cost of the *a priori* solutions reached by the algorithms, we use this formula to compute the cost of the recorded solutions obtained from each algorithm.

In addition to tables, the results are visualized using runtime development plots. These plots show how the cost of solutions develops over computation time and they can be used to compare the performance of several algorithms over time. In these plots, the x-axis indicates computation time in logarithmic scale and the y-axis indicates the cost of the solutions found, averaged over 100 instances. For comparing several algorithms, one of them has been taken as a reference: for each instance, the computation time and the cost of the solutions of the algorithms are normalized by the average computation time and the average cost of the local optima obtained by the reference algorithm.

In order to test whether the observed differences between the expected solution costs of different algorithms are significant in a statistical sense, a paired Wilcoxon test with $\alpha = 0.05$ is adopted; the two sided p-value is computed for each comparison and it is adjusted by Holm's method in the case of multiple comparison.

3.1.1 Parameter tuning

`2.5-opt-EEais` is a parameterized algorithm and its full potential cannot be achieved unless its parameters are fine tuned. We used `Iterative F-Race` (Balaprakash et al., 2007) to tune its parameters. The parameters tuned are the following: (i) min_{is}, the parameter that is used to decide whether importance sampling should be adopted or not in a *2-exchange* move; if the number of nodes between b and c (or between a and d) is less than min_{is}, then importance sampling is employed; (ii) w, the number of nodes close to b and c (or a and d) that should be biased in a *2-exchange* move; (iii) p'_i, the biased probability of node i to be used in a *2-exchange* move; (iv) p''_i, the biased probability of node i to be used in a *node-insertion* move. Instead of tuning p'_i for each node i, a parameter p' has been tuned and is given as the biased probability for all the nodes. Similarly, for the *node-insertion* biased probability, p'' has been tuned. The range for each parameter given to the tuning algorithm and the selected value are shown in Table 1.

The selected values, $min_{is} = 8$, $w = 1$ and $p' = 0.11$ allow `2.5-opt-EEais` to use importance sampling in *2-exchange* moves only occasionally. This is due to the ineffectiveness of the *2-exchange* neighborhood relation for low probability values as discussed in Section 2.4. We also made some tests in which importance sampling is completely disabled in *2-exchange* moves. The results showed that indeed the occasional usage of importance sampling in *2-exchange* moves resulted in solutions' costs that were slightly better than the ones in which importance sampling was completely disabled. It should be noted that the tuned biased probability for *node-insertion* moves, $p'' = 0.60$, allows `2.5-opt-EEais` to use importance sampling in all the *node-insertion* moves up to $p < 0.6$.

3.1.2 A note on the variance of the cost estimator

The relationship between the probabilities associated with the nodes of the PTSP and the variance of the cost estimator can be shown empirically. To do so, first we generated a TSP instance with 1000 nodes, from which we obtained 14 PTSP instances with different probability levels:

Table 1: Parameter values considered for tuning `2.5-opt-EEais` and the values selected by `Iterative/F-Race`

parameter	range	selected value
min_{is}	$[1, 50]$	8
w	$[0, 10]$	1
p'	$[0.0, 1.0]$	0.11
p''	$[0.0, 1.0]$	0.60

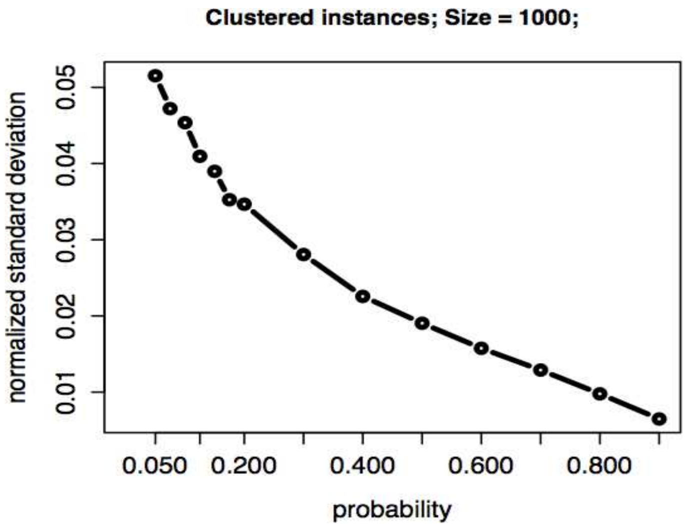


Figure 5: Relationship between the probability levels and the variance of the cost estimator. The y-axis represents the normalized standard deviation, an indicator of the variance of the cost estimator. This is computed on 1000 *a posteriori* solution costs.

$[0.050, 0.200]$ with a step size of 0.025 and $[0.3, 0.9]$ with a step size of 0.1. The local optimum of each PTSP instance, obtained using `2.5-opt-EEais`, was then evaluated on 1000 freshly generated realizations. In order to visualize the variance of the cost estimator in a simple way, we plot the normalized standard deviation, which is an indicator of the variance of the cost estimator. This normalized standard deviation is computed on 1000 *a posteriori* solution costs. Its relationship with different probability levels is shown in Figure 5. From this plot, we can see that the variance of the cost estimator increases with decreasing probability values, in particular, it increases rapidly for $p \leq 0.2$.

3.2 Preliminary experiments

For *homogeneous* PTSP with $p \geq 0.1$, `2.5-opt-ACs` has already been shown to be more effective than `1-shift` and `2-p-opt` (Birattari et al., 2007a,b). In Figure 6, we show that the same tendency holds also for low probability values, that is, for $p < 0.1$. Concerning the time required to reach local optima, irrespective of the probability levels, `2.5-opt-ACs` is faster than `2-p-opt` by approximately a factor of five. In the case of `1-shift`, the same tendency holds when $p \geq 0.2$. However, for small values of p, the difference in speed between `2.5-opt-ACs` and `1-shift` becomes small. Concerning the average cost of local optima found, `2.5-opt-ACs` is between 2% and 5% better than `2-p-opt` and it is 1% and 4% better than `1-shift`. For absolute values, see Table 3. The p-values of the paired Wilcoxon test given in Table 2 show that the cost of the local optima obtained by `2.5-opt-ACs` is significantly lower than that of `1-shift` and `2-p-opt` for all probability levels. Therefore, in the following sections, we take `2.5-opt-ACs` as a yardstick for

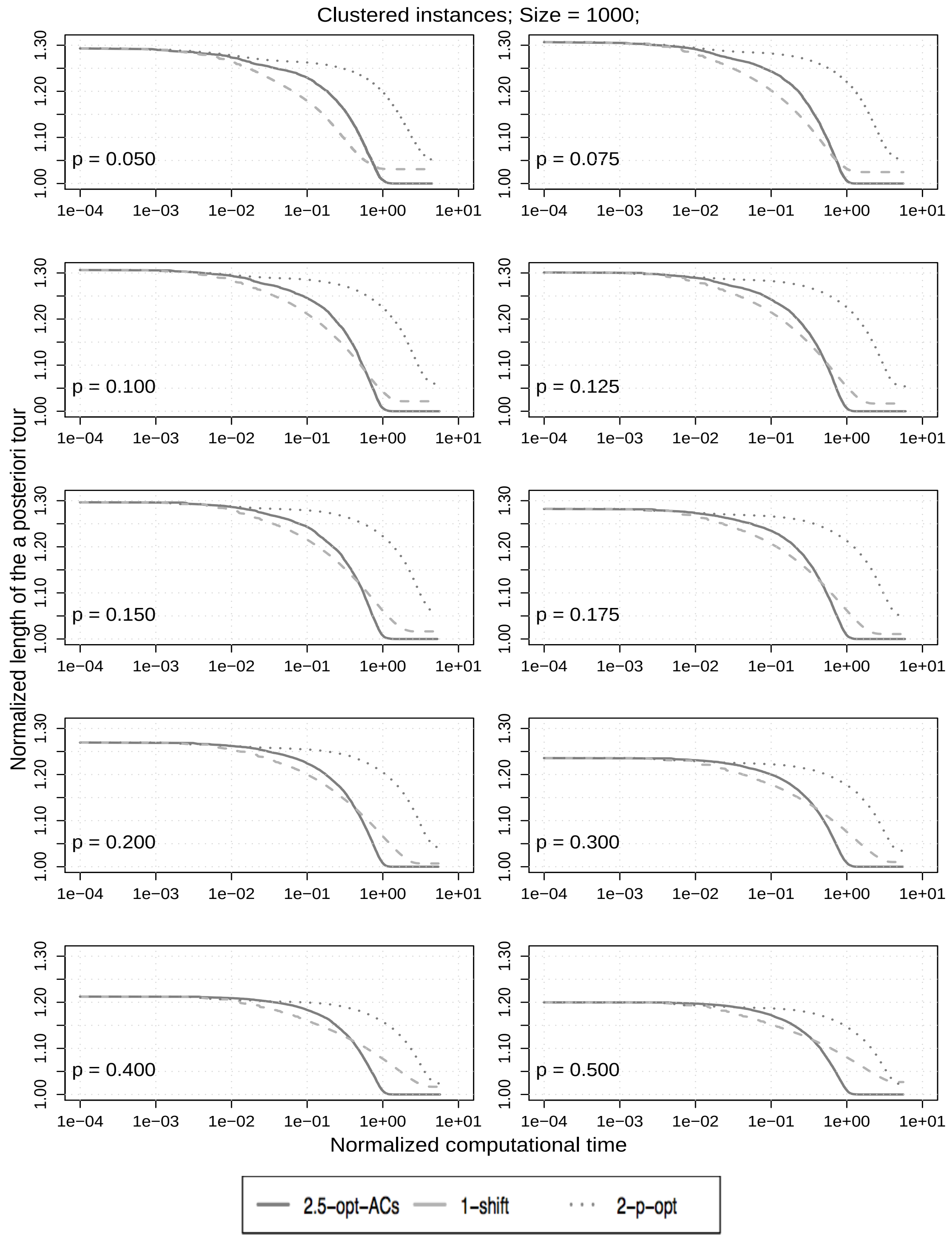


Figure 6: Experimental results on clustered homogeneous PTSP instances of size 1000. The plots represent the average cost of the solutions obtained by `2-p-opt` and `1-shift` normalized by the one obtained by `2.5-opt-ACs`. Each algorithm is stopped when it reaches a local optimum.

measuring the effectiveness of the proposed algorithms.

Table 2: The p-values of the comparison of `2.5-opt-ACs`, `2-p-opt` and `1-shift` on clustered instances of size 1000. Each algorithm is allowed to run until it reaches a local optimum. The statistical test adopted is the paired Wilcoxon test with p-values adjusted by Holm's method. The confidence level is 95%. Values in bold mean that the algorithm in the row performs significantly better than the algorithm in the column, while values in italic mean that the algorithm in the column performs significantly better than the algorithm in the row.

		p-values		
		2.5-opt-ACs	1-shift	2-p-opt
$p = 0.050$	2.5-opt-ACs	-	**0.000**	**0.000**
	1-shift	*0.000*	-	**0.000**
	2-p-opt	*0.000*	*0.000*	-
		2.5-opt-ACs	1-shift	2-p-opt
$p = 0.075$	2.5-opt-ACs	-	**0.000**	**0.000**
	1-shift	*0.000*	-	**0.000**
	2-p-opt	*0.000*	*0.000*	-
		2.5-opt-ACs	1-shift	2-p-opt
$p = 0.100$	2.5-opt-ACs	-	**0.000**	**0.000**
	1-shift	*0.000*	-	**0.000**
	2-p-opt	*0.000*	*0.000*	-
		2.5-opt-ACs	1-shift	2-p-opt
$p = 0.125$	2.5-opt-ACs	-	**0.000**	**0.000**
	1-shift	*0.000*	-	**0.000**
	2-p-opt	*0.000*	*0.000*	-
		2.5-opt-ACs	1-shift	2-p-opt
$p = 0.150$	2.5-opt-ACs	-	**0.000**	**0.000**
	1-shift	*0.000*	-	**0.000**
	2-p-opt	*0.000*	*0.000*	-
		2.5-opt-ACs	1-shift	2-p-opt
$p = 0.175$	2.5-opt-ACs	-	**0.017**	**0.000**
	1-shift	*0.017*	-	**0.000**
	2-p-opt	*0.000*	*0.000*	-
		2.5-opt-ACs	1-shift	2-p-opt
$p = 0.200$	2.5-opt-ACs	-	**0.043**	**0.000**
	1-shift	*0.043*	-	**0.000**
	2-p-opt	*0.000*	*0.000*	-
		2.5-opt-ACs	1-shift	2-p-opt
$p = 0.3$	2.5-opt-ACs	-	**0.000**	**0.000**
	1-shift	*0.000*	-	**0.000**
	2-p-opt	*0.000*	*0.000*	-
		2.5-opt-ACs	1-shift	2-p-opt
$p = 0.4$	2.5-opt-ACs	-	**0.000**	**0.000**
	1-shift	*0.000*	-	**0.001**
	2-p-opt	*0.000*	*0.001*	-
		2.5-opt-ACs	1-shift	2-p-opt
$p = 0.5$	2.5-opt-ACs	-	**0.000**	**0.000**
	1-shift	*0.000*	-	*0.032*
	2-p-opt	*0.000*	**0.032**	-

3.3 Experiments on *estimation-based* algorithms

In this section, we study the performance of `2.5-opt-EEas` and `2.5-opt-EEais` by comparing their solution cost and computation time to `2.5-opt-EEs`. In the case of `2.5-opt-EEs`, we consider samples of size 10, 100, and 1000; we denote these algorithms by `2.5-opt-EEs-10`, `2.5-opt-EEs--100`, and `2.5-opt-EEs-1000` (note that these algorithms do not use the adaptive sample size and the importance sampling procedures). The results of the comparison of the five algorithms are given in Figure 7, where `2.5-opt-EEs-1000` is taken as a reference. Tables 3 and 4 show the absolute values and the p-values of the paired Wilcoxon test, respectively.

The computational results show that `2.5-opt-EEais` is more effective than the other algorithms—in particular, for low probability levels. For what concerns the comparison of `2.5-opt-EEais` and `2.5-opt-EEas`, the results show that the adoption of importance sampling allows the former to achieve high quality solutions for very low probability values, that is, for $p < 0.2$—the average cost of the local optima obtained by `2.5-opt-EEais` is between 1% and 3% less than that of `2.5-opt--EEas`. The observed differences are significant in a statistical sense—see Table 4. For $p \geq 0.3$, the average cost of the solutions and the computation time of `2.5-opt-EEais` are comparable to the ones of `2.5-opt-EEas`.

Concerning the comparison of `2.5-opt-EEais` and `2.5-opt-EEs-1000`, the former achieves an average cost similar to that of the latter. However, the advantage of `2.5-opt-EEais` is the computation time: it is faster than `2.5-opt-EEs-1000` approximately by a factor of four.

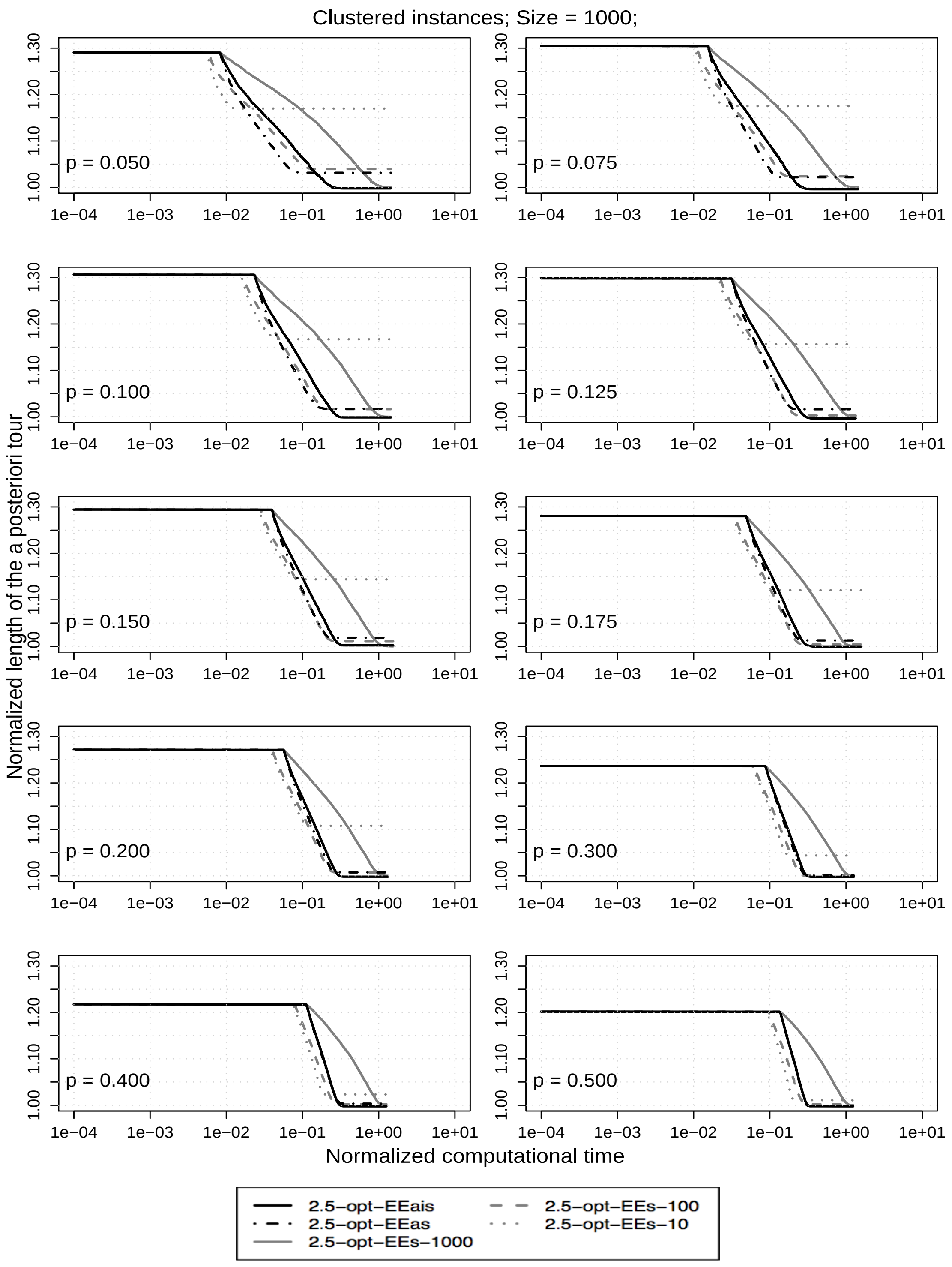


Figure 7: Experimental results on clustered homogeneous PTSP instances of size 1000. The plots represent the cost of the solutions obtained by `2.5-opt-EEas`, `2.5-opt-EEais`, `2.5-opt-EEs-10`, and `2.5-opt-EEs-100` normalized by the one obtained by `2.5-opt-EEs-1000`. Each algorithm is stopped when it reaches a local optimum.

Regarding the comparison of `2.5-opt-EEais` and `2.5-opt-EEs-100`, for low probability values, $p \leq 0.2$, the average cost of the solutions obtained by the former is between 1% and 3% lower that that of `2.5-opt-EEs-100`. This clearly shows that the adoption of 100 realizations is not sufficient for these probability levels. Note that these differences are significant according to the paired Wilcoxon test. On the other hand, for $p \geq 0.3$, the two algorithms are comparable to one another with respect to solution quality and computation time.

Although faster, `2.5-opt-EEs-10` achieves a very poor solution quality: the average cost of the solutions obtained by `2.5-opt-EEs-10` is between 17% and 2% higher that that of `2.5-opt-EEais`.

For probability levels greater than 0.5, the numerical results show that the algorithms achieve equivalent results with respect to the solution quality. Moreover, the results of `2.5-opt-EEs-10` show that a sample size of 10 is sufficient to tackle instances with $p > 0.5$. For what concerns the computation time, `2.5-opt-EEais` and `2.5-opt-EEs-100` are comparable to `2.5-opt-EEs--10`. However, `2.5-opt-EEais` is faster than `2.5-opt-EEs-1000` by a factor of three. Note that `2.5-opt-EEais` and `2.5-opt-EEas` are essentially the same for these probability levels.

Taking into account both the computation time and the cost of the solutions obtained, we can see that `2.5-opt-EEais` emerges as a clear winner among the considered *estimation-based* algorithms.

Table 3: Experimental results for `2.5-opt-EEas`, `2.5-opt-EEais`, `2.5-opt-EEs--10`, `2.5-opt-EEs-100`, `2.5-opt-EEs-1000`, `2.5-opt-ACs`, `2-p-opt` and `1-shift` on clustered instances of size 1000. Each algorithm is allowed to run until it reaches a local optimum. The table gives mean and standard deviation (s.d.) of final solution cost and computation time in seconds. The results are given for 100 instances at each probability level. The symbol × indicates that the algorithms do not produce meaningful results due to the numerical problem. The algorithms based on the *empirical estimation* do not suffer from this problem.

	Algorithm	Solution Cost		Computation Time	
		mean	s.d.	mean	s.d.
$p = 0.050$	`2.5-opt-EEais`	3979071	396999	10.985	1.506
	`2.5-opt-EEas`	4113281	400094	3.017	0.506
	`2.5-opt-EEs-1000`	3987514	414951	41.725	7.145
	`2.5-opt-EEs-100`	4145766	411885	4.378	0.718
	`2.5-opt-EEs-10`	4665151	438176	0.477	0.039
	`2.5-opt-ACs`	3981009	387310	785.410	113.257
	`1-shift`	4104861	445292	634.526	146.398
	`2-p-opt`	4187971	421438	2204.397	649.341
$p = 0.075$	`2.5-opt-EEais`	4547516	412236	6.119	0.820
	`2.5-opt-EEas`	4665403	410482	2.606	0.322
	`2.5-opt-EEs-1000`	4565251	425792	22.624	2.829
	`2.5-opt-EEs-100`	4673597	411925	3.411	0.386
	`2.5-opt-EEs-10`	5364829	452497	0.535	0.037
	`2.5-opt-ACs`	4559679	428530	587.990	71.020
	`1-shift`	4672661	453726	688.678	114.010
	`2-p-opt`	4800235	436085	1884.282	455.744
$p = 0.100$	`2.5-opt-EEais`	5061985	438942	4.125	0.470
	`2.5-opt-EEas`	5157133	437331	2.279	0.241
	`2.5-opt-EEs-1000`	5068797	448162	14.808	1.930
	`2.5-opt-EEs-100`	5152785	438530	2.685	0.308
	`2.5-opt-EEs-10`	5915152	446106	0.583	0.049
	`2.5-opt-ACs`	5068223	450709	452.868	60.388
	`1-shift`	5178144	469977	674.985	87.745
	`2-p-opt`	5365486	449318	1505.189	333.144
$p = 0.125$	`2.5-opt-EEais`	5515565	451263	3.076	0.343
	`2.5-opt-EEas`	5625240	468537	2.013	0.215
	`2.5-opt-EEs-1000`	5534344	449688	10.917	1.372
	`2.5-opt-EEs-100`	5550938	446171	2.218	0.222
	`2.5-opt-EEs-10`	6400616	472372	0.619	0.053
	`2.5-opt-ACs`	5522776	439088	363.594	47.541
	`1-shift`	5615434	469014	660.757	90.551
	`2-p-opt`	5820694	467656	1303.610	264.699
$p = 0.150$	`2.5-opt-EEais`	5943406	455610	2.487	0.280

	`2.5-opt-EEas`	6041281	454513	1.776	0.161
	`2.5-opt-EEs-1000`	5930674	441599	8.674	1.202
	`2.5-opt-EEs-100`	5997015	458452	1.857	0.200
	`2.5-opt-EEs-10`	6785563	485912	0.650	0.050
	`2.5-opt-ACs`	5920742	472133	308.862	39.959
	`1-shift`	6018464	481164	635.003	88.903
	`2-p-opt`	6256054	476219	1117.438	226.919
	`2.5-opt-EEais`	6330797	457716	2.041	0.209
	`2.5-opt-EEas`	6414473	476255	1.602	0.142
	`2.5-opt-EEs-1000`	6333479	466420	7.066	0.854
$p = 0.175$	`2.5-opt-EEs-100`	6360582	474837	1.596	0.153
	`2.5-opt-EEs-10`	7097122	535544	0.663	0.047
	`2.5-opt-ACs`	6326607	468850	264.233	35.623
	`1-shift`	6394590	491996	599.314	87.826
	`2-p-opt`	6643060	474838	992.219	192.012
	`2.5-opt-EEais`	6674573	493872	1.775	0.157
	`2.5-opt-EEas`	6736742	504954	1.474	0.137
	`2.5-opt-EEs-1000`	6686638	491334	6.102	0.641
$p = 0.200$	`2.5-opt-EEs-100`	6735824	502516	1.378	0.128
	`2.5-opt-EEs-10`	7408193	520731	0.655	0.049
	`2.5-opt-ACs`	6697814	480609	226.968	26.906
	`1-shift`	6744906	494658	580.355	74.488
	`2-p-opt`	6978848	477593	888.112	160.210
	`2.5-opt-EEais`	7879210	525179	1.158	0.090
	`2.5-opt-EEas`	7899805	553771	1.078	0.086
	`2.5-opt-EEs-1000`	7895340	522287	3.948	0.410
$p = 0.300$	`2.5-opt-EEs-100`	7902533	551358	0.948	0.080
	`2.5-opt-EEs-10`	8240490	548645	0.617	0.048
	`2.5-opt-ACs`	7901717	524412	150.528	22.110
	`1-shift`	7982502	531786	480.545	64.266
	`2-p-opt`	8175024	547814	576.815	103.943
	`2.5-opt-EEais`	8789879	573340	0.898	0.075
	`2.5-opt-EEas`	8840983	574245	0.872	0.071
	`2.5-opt-EEs-1000`	8810714	579258	3.093	0.334
$p = 0.400$	`2.5-opt-EEs-100`	8832988	592100	0.763	0.061
	`2.5-opt-EEs-10`	9015844	583343	0.547	0.037
	`2.5-opt-ACs`	8848198	549139	109.622	15.532
	`1-shift`	8995824	567472	385.921	51.510
	`2-p-opt`	9060178	551378	433.499	73.642
	`2.5-opt-EEais`	9559588	619031	0.757	0.056
	`2.5-opt-EEas`	9565592	619559	0.745	0.049
	`2.5-opt-EEs-1000`	9582160	630991	2.550	0.257
$p = 0.500$	`2.5-opt-EEs-100`	9602311	615180	0.670	0.048
	`2.5-opt-EEs-10`	9684247	599151	0.494	0.029
	`2.5-opt-ACs`	9597432	599270	89.415	12.716
	`1-shift`	9856073	579796	326.975	46.330
	`2-p-opt`	9799469	594428	347.427	61.300
	`2.5-opt-EEais`	10232166	651824	0.665	0.035
	`2.5-opt-EEas`	10227292	652924	0.659	0.036
	`2.5-opt-EEs-1000`	10271570	650493	2.216	0.197
$p = 0.600$	`2.5-opt-EEs-100`	10280701	653690	0.606	0.037
	`2.5-opt-EEs-10`	10254630	610135	0.459	0.028
	`2.5-opt-ACs`	×	×	×	×
	`1-shift`	×	×	×	×
	`2-p-opt`	×	×	×	×
	`2.5-opt-EEais`	10815818	661729	0.613	0.032
	`2.5-opt-EEas`	10815731	661732	0.607	0.030
	`2.5-opt-EEs-1000`	10812671	665515	1.999	0.184
$p = 0.700$	`2.5-opt-EEs-100`	10834997	630350	0.566	0.038
	`2.5-opt-EEs-10`	10866704	673214	0.427	0.022
	`2.5-opt-ACs`	×	×	×	×
	`1-shift`	×	×	×	×
	`2-p-opt`	×	×	×	×
	`2.5-opt-EEais`	11314317	669416	0.574	0.028
	`2.5-opt-EEas`	11314319	669565	0.569	0.028
	`2.5-opt-EEs-1000`	11329129	702036	1.811	0.154
$p = 0.800$	`2.5-opt-EEs-100`	11353344	682321	0.536	0.028

	`2.5-opt-EEs-10`	11342206	674372	0.410	0.023
	`2.5-opt-ACs`	×	×	×	×
	`1-shift`	×	×	×	×
	`2-p-opt`	×	×	×	×
$p = 0.900$	`2.5-opt-EEais`	11775623	718848	0.544	0.021
	`2.5-opt-EEas`	11774558	718768	0.540	0.022
	`2.5-opt-EEs-1000`	11776547	718035	1.685	0.132
	`2.5-opt-EEs-100`	11771051	705646	0.517	0.029
	`2.5-opt-EEs-10`	11777405	698347	0.401	0.021
	`2.5-opt-ACs`	×	×	×	×
	`1-shift`	×	×	×	×
	`2-p-opt`	×	×	×	×

Table 4: The p-values of the pairwise comparisons of `2.5-opt-EEas`, `2.5-opt-EEais`, `2.5-opt--EEs-10`, `2.5-opt-EEs-100`, and `2.5-opt-EEs-1000` on clustered instances of size 1000 for probability levels less than 0.5. Values in bold mean that the algorithm in the row performs significantly better than the algorithm in the column, while values in italic mean that the algorithm in the column performs significantly better than the algorithm in the row.

		p-values				
		2.5-opt-EEais	2.5-opt-EEas	2.5-opt-EEs-1000	2.5-opt-EEs-100	2.5-opt-EEs-10
	2.5-opt-EEais	-	**0.000**	**0.045**	**0.000**	**0.000**
	2.5-opt-EEas	*0.000*	-	*0.000*	**0.003**	**0.000**
$p = 0.050$	2.5-opt-EEs-1000	*0.045*	**0.000**	-	**0.000**	**0.000**
	2.5-opt-EEs-100	*0.000*	*0.003*	*0.000*	-	**0.000**
	2.5-opt-EEs-10	*0.000*	*0.000*	*0.000*	*0.000*	-
		2.5-opt-EEais	2.5-opt-EEas	2.5-opt-EEs-1000	2.5-opt-EEs-100	2.5-opt-EEs-10
	2.5-opt-EEais	-	**0.000**	0.206	**0.000**	**0.000**
	2.5-opt-EEas	*0.000*	-	*0.000*	0.732	**0.000**
$p = 0.075$	2.5-opt-EEs-1000	0.206	**0.000**	-	**0.000**	**0.000**
	2.5-opt-EEs-100	*0.000*	0.732	*0.000*	-	**0.000**
	2.5-opt-EEs-10	*0.000*	*0.000*	*0.000*	*0.000*	-
		2.5-opt-EEais	2.5-opt-EEas	2.5-opt-EEs-1000	2.5-opt-EEs-100	2.5-opt-EEs-10
	2.5-opt-EEais	-	**0.000**	0.899	**0.000**	**0.000**
	2.5-opt-EEas	*0.000*	-	*0.000*	0.423	**0.000**
$p = 0.100$	2.5-opt-EEs-1000	0.899	**0.000**	-	**0.000**	**0.000**
	2.5-opt-EEs-100	*0.000*	0.423	*0.000*	-	**0.000**
	2.5-opt-EEs-10	*0.000*	*0.000*	*0.000*	*0.000*	-
		2.5-opt-EEais	2.5-opt-EEas	2.5-opt-EEs-1000	2.5-opt-EEs-100	2.5-opt-EEs-10
	2.5-opt-EEais	-	**0.000**	0.974	**0.010**	**0.000**
	2.5-opt-EEas	*0.000*	-	*0.000*	*0.000*	**0.000**
$p = 0.125$	2.5-opt-EEs-1000	0.974	**0.000**	-	0.108	**0.000**
	2.5-opt-EEs-100	*0.010*	**0.000**	0.108	-	**0.000**
	2.5-opt-EEs-10	*0.000*	*0.000*	*0.000*	*0.000*	-
		2.5-opt-EEais	2.5-opt-EEas	2.5-opt-EEs-1000	2.5-opt-EEs-100	2.5-opt-EEs-10
	2.5-opt-EEais	-	**0.000**	0.725	**0.000**	**0.000**
	2.5-opt-EEas	*0.000*	-	*0.000*	*0.020*	**0.000**
$p = 0.150$	2.5-opt-EEs-1000	0.725	**0.000**	-	**0.000**	**0.000**
	2.5-opt-EEs-100	*0.000*	**0.020**	*0.000*	-	**0.000**
	2.5-opt-EEs-10	*0.000*	*0.000*	*0.000*	*0.000*	-
		2.5-opt-EEais	2.5-opt-EEas	2.5-opt-EEs-1000	2.5-opt-EEs-100	2.5-opt-EEs-10
	2.5-opt-EEais	-	**0.000**	0.857	0.052	**0.000**
	2.5-opt-EEas	*0.000*	-	*0.000*	*0.004*	**0.000**
$p = 0.175$	2.5-opt-EEs-1000	0.857	**0.000**	-	0.111	**0.000**
	2.5-opt-EEs-100	0.052	**0.004**	0.111	-	**0.000**
	2.5-opt-EEs-10	*0.000*	*0.000*	*0.000*	*0.000*	-
		2.5-opt-EEais	2.5-opt-EEas	2.5-opt-EEs-1000	2.5-opt-EEs-100	2.5-opt-EEs-10
	2.5-opt-EEais	-	**0.000**	0.999	**0.001**	**0.000**
	2.5-opt-EEas	*0.000*	-	*0.001*	0.999	**0.000**
$p = 0.200$	2.5-opt-EEs-1000	0.999	**0.001**	-	**0.015**	**0.000**
	2.5-opt-EEs-100	*0.001*	0.999	*0.015*	-	**0.000**
	2.5-opt-EEs-10	*0.000*	*0.000*	*0.000*	*0.000*	-
		2.5-opt-EEais	2.5-opt-EEas	2.5-opt-EEs-1000	2.5-opt-EEs-100	2.5-opt-EEs-10
	2.5-opt-EEais	-	0.244	1.000	1.000	**0.000**
	2.5-opt-EEas	0.244	-	1.000	1.000	**0.000**
$p = 0.300$	2.5-opt-EEs-1000	1.000	1.000	-	1.000	**0.000**
	2.5-opt-EEs-100	1.000	1.000	1.000	-	**0.000**
	2.5-opt-EEs-10	*0.000*	*0.000*	*0.000*	*0.000*	-
		2.5-opt-EEais	2.5-opt-EEas	2.5-opt-EEs-1000	2.5-opt-EEs-100	2.5-opt-EEs-10
	2.5-opt-EEais	-	0.195	0.441	0.441	**0.000**
	2.5-opt-EEas	0.195	-	0.112	0.441	**0.000**
$p = 0.400$	2.5-opt-EEs-1000	0.441	0.112	-	0.441	**0.000**
	2.5-opt-EEs-100	0.441	0.441	0.441	-	**0.000**
	2.5-opt-EEs-10	*0.000*	*0.000*	*0.000*	*0.000*	-
		2.5-opt-EEais	2.5-opt-EEas	2.5-opt-EEs-1000	2.5-opt-EEs-100	2.5-opt-EEs-10
	2.5-opt-EEais	-	1.000	1.000	0.973	**0.000**
	2.5-opt-EEas	1.000	-	0.973	1.000	**0.000**
$p = 0.500$	2.5-opt-EEs-1000	1.000	0.973	-	1.000	**0.000**
	2.5-opt-EEs-100	0.973	1.000	1.000	-	**0.000**
	2.5-opt-EEs-10	*0.000*	*0.000*	*0.000*	*0.000*	-

Table 5: The p-values of the comparison of `2.5-opt-EEas`, `2.5-opt-EEais`, `2.5-opt-EEs-10`, `2.5-opt-EEs-100`, and `2.5-opt-EEs-1000` on clustered instances of size 1000 for probability levels greater than 0.5. Each algorithm is allowed to run until it reaches a local optimum. The statistical test adopted is the paired Wilcoxon test with p-values adjusted by Holm's method. The confidence level is 95%. The statistical test does not reject the null hypothesis that the algorithms achieve equivalent results.

		p-values				
		2.5-opt-EEais	2.5-opt-EEas	2.5-opt-EEs-1000	2.5-opt-EEs-100	2.5-opt-EEs-10
$p = 0.600$	2.5-opt-EEais	-	1.000	0.144	0.493	0.493
	2.5-opt-EEas	1.000	-	0.085	0.412	0.395
	2.5-opt-EEs-1000	0.144	0.085	-	1.000	1.000
	2.5-opt-EEs-100	0.493	0.412	1.000	-	1.000
	2.5-opt-EEs-10	0.493	0.395	1.000	1.000	-
		2.5-opt-EEais	2.5-opt-EEas	2.5-opt-EEs-1000	2.5-opt-EEs-100	2.5-opt-EEs-10
$p = 0.700$	2.5-opt-EEais	-	1.000	1.000	1.000	0.137
	2.5-opt-EEas	1.000	-	1.000	1.000	0.137
	2.5-opt-EEs-1000	1.000	1.000	-	1.000	0.059
	2.5-opt-EEs-100	1.000	1.000	1.000	-	0.699
	2.5-opt-EEs-10	0.137	0.137	0.059	0.699	-
		2.5-opt-EEais	2.5-opt-EEas	2.5-opt-EEs-1000	2.5-opt-EEs-100	2.5-opt-EEs-10
$p = 0.800$	2.5-opt-EEais	-	1.000	1.000	1.000	0.951
	2.5-opt-EEas	1.000	-	1.000	1.000	0.951
	2.5-opt-EEs-1000	1.000	1.000	-	1.000	1.000
	2.5-opt-EEs-100	1.000	1.000	1.000	-	1.000
	2.5-opt-EEs-10	0.951	0.951	1.000	1.000	-
		2.5-opt-EEais	2.5-opt-EEas	2.5-opt-EEs-1000	2.5-opt-EEs-100	2.5-opt-EEs-10
$p = 0.900$	2.5-opt-EEais	-	1.000	1.000	1.000	1.000
	2.5-opt-EEas	1.000	-	1.000	1.000	1.000
	2.5-opt-EEs-1000	1.000	1.000	-	1.000	1.000
	2.5-opt-EEs-100	1.000	1.000	1.000	-	1.000
	2.5-opt-EEs-10	1.000	1.000	1.000	1.000	-

3.4 Final assessment

In this section, we compare `2.5-opt-EEais` with `2.5-opt-ACs`. For this purpose, we generated another 100 instances for each probability level. The rationale behind the adoption of a new set of instances is the following: `2.5-opt-EEais` and `2.5-opt-ACs` are selected as winners from a set of 5 and 3 algorithms, respectively, where all of them are evaluated on a same set of instances. Basing the comparison of `2.5-opt-EEais` and `2.5-opt-ACs` on the same set of instances might possibly introduce a bias in favor of `2.5-opt-EEais`. This issue is known as *over-tuning*; we refer the reader to Birattari (2004) for further discussion.

The computational results given in Figure 8 and Table 6 show that `2.5-opt-EEais` is very competitive. Regarding the time required to reach local optima, irrespective of the probability levels, `2.5-opt-EEais` is approximately 2 orders of magnitude faster than `2.5-opt-ACs`. The average cost of local optima obtained by `2.5-opt-EEais` is comparable to the one of `2.5-opt-ACs`.

In Table 8, we report the observed relative difference between the cost of the local optima obtained by the two algorithms and a 95% confidence bound on this relative difference. This bound is obtained through a one-sided paired Wilcoxon test. Table 8 confirms that, concerning the average cost of the local optima found, `2.5-opt-EEais` is essentially equivalent to `2.5-opt-ACs`. On average, `2.5-opt-EEais` obtains slightly better results, even if our experiments were not able to detect statistical significance except for $p = 0.400$. Nonetheless, irrespective of probability levels, should ever the average cost obtained by `2.5-opt-EEais` be higher than the one obtained by `2.5-opt-ACs`, the difference would be at most 0.39%.

4 Conclusion and Future Work

Motivated by the lack of systematic integration between optimization algorithms and specialized simulation techniques for stochastic combinatorial optimization, we integrated the adaptive sample size and the importance sampling procedures into an *estimation-based* iterative improvement algorithm. We used the PROBABILISTIC TRAVELING SALESMAN PROBLEM as a test bed and we showed that the two procedures are effective. The proposed methodology is general purpose, conceptually simple, easy to implement, scalable to large instances of large sizes and can be applied

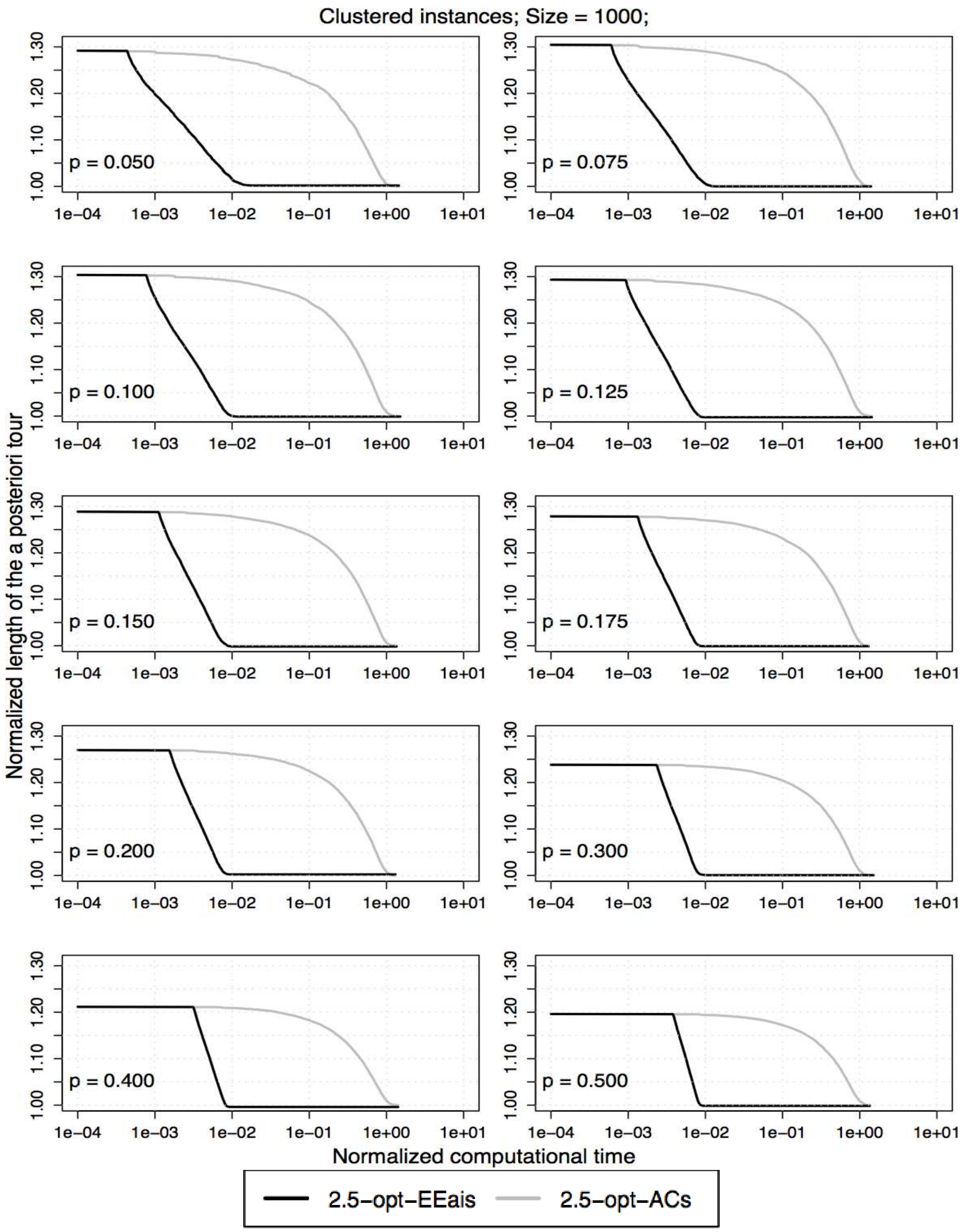


Figure 8: Experimental results on clustered homogeneous PTSP instances of size 1000. The plots represent the cost of the solutions obtained by `2.5-opt-EEais` normalized by the one obtained by `2.5-opt-ACs`. Each algorithm is stopped when it reaches a local optimum.

Table 6: Experimental results for `2.5-opt-EEais` and `2.5-opt-ACs`, on clustered instances of size 1000. Each algorithm is allowed to run until it reaches a local optimum. The table gives the mean and the standard deviation (s.d.) of the final solution cost and the computation time in seconds. The results are given for 100 instances at each probability level.

	Algorithm	Solution Cost		Computation Time	
		mean	s.d.	mean	s.d.
$p = 0.050$	`2.5-opt-EEais`	4005088	383890	11.133	1.780
	`2.5-opt-ACs`	3998071	373847	791.594	112.789
$p = 0.075$	`2.5-opt-EEais`	4574323	399872	6.082	0.773
	`2.5-opt-ACs`	4574761	406584	577.353	90.806
$p = 0.100$	`2.5-opt-EEais`	5077261	415954	4.134	0.473
	`2.5-opt-ACs`	5083587	412256	449.297	67.028
$p = 0.125$	`2.5-opt-EEais`	5543550	448682	3.036	0.320
	`2.5-opt-ACs`	5557740	443416	374.634	53.923
$p = 0.150$	`2.5-opt-EEais`	5950184	444116	2.470	0.281
	`2.5-opt-ACs`	5959829	446971	311.533	40.494
$p = 0.175$	`2.5-opt-EEais`	6341762	481888	2.042	0.197
	`2.5-opt-ACs`	6347748	489415	261.905	32.857
$p = 0.200$	`2.5-opt-EEais`	6714236	507386	1.742	0.154
	`2.5-opt-ACs`	6699148	481917	224.652	28.431
$p = 0.300$	`2.5-opt-EEais`	7896103	541727	1.758	5.884
	`2.5-opt-ACs`	7890603	537486	148.077	22.784
$p = 0.400$	`2.5-opt-EEais`	8825878	596968	0.898	0.061
	`2.5-opt-ACs`	8860822	563736	110.226	16.785
$p = 0.500$	`2.5-opt-EEais`	9622254	616871	0.753	0.046
	`2.5-opt-ACs`	9637212	651691	90.801	13.193

to other classes of problems in which the cost difference cannot be expressed in a closed-form.

Further research will be devoted to assess the behavior of the proposed approach when used as an embedded heuristic in metaheuristics such as iterated local search, ant colony optimization and memetic algorithms.

Acknowledgments

This research has been supported by COMP^{2}SYS, a Marie Curie Early Stage Research Training Site funded by the European Community's Sixth Framework Programme under contract number MEST-CT-2004-505079, and by the ANTS project, an *Action de Recherche Concertée* funded by the Scientific Research Directorate of the French Community of Belgium. Prasanna Balaprakash, Thomas Stützle, and Marco Dorigo acknowledge support from the Belgian FNRS of which they are an Aspirant, a Research Associate, and a Research Director, respectively. The information provided is the sole responsibility of the authors and does not reflect the opinion of the sponsors. The European Community is not responsible for any use that might be made of data appearing in this publication.

References

P. Balaprakash, M. Birattari, and T. Stützle. Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement. In T. Bartz-Beielstein, M. Blesa, C. Blum, B. Naujoks, A. Roli, G. Rudolph, and M. Sampels, editors, *Hybrid Metaheuristics*, volume 4771 of *LNCS*, pages 113–127, Berlin, Germany, 2007. Springer-Verlag.

D. Bertsimas. *Probabilistic Combinatorial Optimization Problems*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1988.

D. Bertsimas and L. Howell. Further results on the probabilistic traveling salesman problem. *European Journal of Operations Research*, 65(1):68–95, 1993.

D. Bertsimas, P. Jaillet, and A. Odoni. A priori optimization. *Operations Research*, 38(6):1019–1033, 1990.

Table 7: The p-values of the comparison of `2.5-opt-ACs`, `2-p-opt` and `1-shift` on clustered instances of size 1000. Each algorithm is allowed to run until it reaches a local optimum. The statistical test adopted is the paired Wilcoxon test with p-values adjusted by Holm's method. The confidence level is 95%. Values in bold mean that the algorithm in the row performs significantly better than the algorithm in the column, while values in italic mean that the algorithm in the column performs significantly better than the algorithm in the row.

		p-values	
		`2.5-opt-EEais`	`2.5-opt-ACs`
$p = 0.050$	`2.5-opt-EEais`	-	0.064
	`2.5-opt-ACs`	0.064	-
		`2.5-opt-EEais`	`2.5-opt-ACs`
$p = 0.075$	`2.5-opt-EEais`	-	0.612
	`2.5-opt-ACs`	0.612	-
		`2.5-opt-EEais`	`2.5-opt-ACs`
$p = 0.100$	`2.5-opt-EEais`	-	0.968
	`2.5-opt-ACs`	0.968	-
		`2.5-opt-EEais`	`2.5-opt-ACs`
$p = 0.125$	`2.5-opt-EEais`	-	0.524
	`2.5-opt-ACs`	0.524	-
		`2.5-opt-EEais`	`2.5-opt-ACs`
$p = 0.150$	`2.5-opt-EEais`	-	0.436
	`2.5-opt-ACs`	0.436	-
		`2.5-opt-EEais`	`2.5-opt-ACs`
$p = 0.175$	`2.5-opt-EEais`	-	0.761
	`2.5-opt-ACs`	0.761	-
		`2.5-opt-EEais`	`2.5-opt-ACs`
$p = 0.200$	`2.5-opt-EEais`	-	0.859
	`2.5-opt-ACs`	0.859	-
		`2.5-opt-EEais`	`2.5-opt-ACs`
$p = 0.3$	`2.5-opt-EEais`	-	0.916
	`2.5-opt-ACs`	0.916	-
		`2.5-opt-EEais`	`2.5-opt-ACs`
$p = 0.4$	`2.5-opt-EEais`	-	**0.001**
	`2.5-opt-ACs`	*0.001*	-
		`2.5-opt-EEais`	`2.5-opt-ACs`
$p = 0.5$	`2.5-opt-EEais`	-	0.430
	`2.5-opt-ACs`	0.430	-

Table 8: Results on the comparison between `2.5-opt-EEais` and `2.5-opt-ACs`. Given is the observed relative difference and a 95% confidence bound on the relative difference obtained through a one-side paired Wilcoxon test. Concerning the relative difference, if the value is positive, `2.5-opt-EEais` obtained an average cost that is larger than the one obtained by `2.5-opt-ACs`; if it is negative, `2.5-opt-EEais` reached solutions of lower average cost. In both cases, a value is typeset in boldface if it is statistically significant. Concerning the bound, a positive value $+d\%$ indicates that `2.5-opt-EEais` is not more than $d\%$ worse than `2.5-opt-ACs`; a negative value $-d\%$ indicates that `2.5-opt-EEais` at least $d\%$ better.

p	Difference	Bound
0.050	+0.176%	+0.222%
0.075	−0.010%	+0.261%
0.100	−0.124%	+0.298%
0.125	−0.255%	+0.229%
0.150	−0.162%	+0.206%
0.175	−0.094%	+0.297%
0.200	+0.225%	+0.388%
0.300	+0.070%	+0.394%
0.400	−**0.394**%	−0.283%
0.500	−0.155%	+0.157%

L. Bianchi. *Ant Colony Optimization and Local Search for the Probabilistic Traveling Salesman Problem: A Case Study in Stochastic Combinatorial Optimization.* PhD thesis, Université Libre de Bruxelles, Brussels, Belgium, 2006.

L. Bianchi and A. Campbell. Extension of the 2-p-opt and 1-shift algorithms to the heterogeneous probabilistic traveling salesman problem. *European Journal of Operations Research*, 176(1):131–144, 2007.

L. Bianchi, J. Knowles, and N. Bowler. Local search for the probabilistic traveling salesman problem: Correction to the 2-p-opt and 1-shift algorithms. *European Journal of Operational Research*, 162(1):206–219, 2005.

M. Birattari. *The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective.* PhD thesis, Université Libre de Bruxelles, Brussels, Belgium, 2004.

M. Birattari, P. Balaprakash, T. Stützle, and M. Dorigo. Estimation-based local search for stochastic combinatorial optimization. Technical Report TR/IRIDIA/2007-003, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2007a. URL `http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2007-003r001.pdf`. Submitted for journal publication.

M. Birattari, P. Balaprakash, T. Stützle, and M. Dorigo. Extended empirical analysis of estimation-based local search for stochastic combinatorial optimization. IRIDIA Supplementary page, 2007b. URL `http://iridia.ulb.ac.be/supp/IridiaSupp2007-001/`.

P. Chervi. A computational approach to probabilistic vehicle routing problems. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 1988.

M. C. Fu. Optimization via simulation: A review. *Annals of Operations Research*, 53:199–248, 1994.

M. C. Fu. Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing*, 14:192–215, 2002.

W. Gutjahr. S-ACO: An ant based approach to combinatorial optimization under uncertainity. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, editors, *Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2004*, volume 3172 of *LNCS*, pages 238–249, Berlin, Germany, 2004. Springer-Verlag.

P. Jaillet. *Probabilistic Traveling Salesman Problems.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1985.

D. S. Johnson and L. A. McGeoch. The travelling salesman problem: A case study in local optimization. In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley & Sons, Chichester, United Kingdom, 1997.

D. S. Johnson, L. A. McGeoch, C. Rego, and F. Glover. 8th DIMACS implementation challenge, 2001. URL `http://www.research.att.com/~dsj/chtsp/`.

J. F. Penky and D. Miller. A staged primal-dual algorithm for finding a minimum cost perfect two-matching in an undirected graph. *ORSA Jornal on Computing*, 6(1):68–81, 1994.

R. Srinivasan. *Importance Sampling - Applications in Communications and Detection.* Springer-Verlag, 2002.