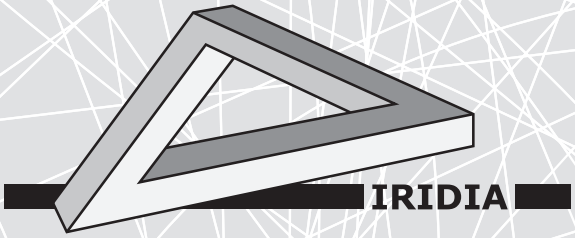

Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Estimation-based Ant Colony Optimization and Local Search for the Probabilistic Traveling Salesman Problem

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

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2008-020

September 2008

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/2008-020

Estimation-based Ant Colony Optimization and Local Search for the Probabilistic Traveling Salesman Problem

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

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

September 30, 2008

Abstract

The use of ant colony optimization for solving stochastic optimization problems has received a significant amount of attention in recent years. In this paper, we present a study of enhanced ant colony optimization algorithms for tackling a stochastic optimization problem, the probabilistic traveling salesman problem. In particular, we propose an empirical estimation approach to evaluate the cost of the solutions constructed by the ants. Moreover, we use a recent estimation-based iterative improvement algorithm as a local search. Experimental results on a large number of problem instances show that the proposed ant colony optimization algorithms outperform the current best algorithm tailored to solve the given problem, which also happened to be an ant colony optimization algorithm. As a consequence, we have obtained a new state-of-the-art ant colony optimization algorithm for the probabilistic traveling salesman problem.

1 Introduction

Ant colony optimization (ACO) (Dorigo and Stützle, 2004) has become a very successful and widely used swarm intelligence method (Dorigo and Birattari, 2007) for solving hard optimization problems. Its success has been proved not only by the large number of problems to which it has been applied, but also by the very good performance ACO algorithms have achieved in many fields, especially for routing problems with complex features, such as stochastic information, or time-varying data (Di Caro and Dorigo, 1998).

In this paper, we study the application of ACO algorithms to the PROBABILISTIC TRAVELING SALESMAN PROBLEM (PTSP) (Jaillet, 1985), which is also known as traveling salesman problem with stochastic customers (Gendreau et al., 1996). It is a stochastic extention of the classical traveling salesman problem (TSP). In the PTSP, it is unknown in advance whether a node requires being visited, but its probability of requiring a visit is given. The most widely used approach to tackle the PTSP is to construct an *a priori* solution before knowing which nodes require being visited. An *a priori* solution is a permutation of all the nodes of the given instance. Once the set of nodes that require being visited is known, the *a posteriori* solution is derived by visiting the nodes that require being visited in the order prescribed by the *a priori* solution and by skipping the nodes that do not require being visited. The objective of the PTSP is to find an *a priori* solution, such that the expected cost of its associated *a posteriori* solution is minimized.

The PTSP is an $\mathcal{NP}$-hard problem. The stochastic nature and the complexity of the problem limit the applicability of exact algorithms. The so far best performing exact solution technique has been proposed by Laporte et al. (1994), who formulated the problem as an integer linear stochastic program and solved it by a branch-and-cut approach. The experimental results showed that instances of size up to 50 can be solved to optimality.

Recent approaches to the PTSP mainly involve the application of stochastic local search (SLS) methods (Hoos and Stützle, 2005), among which ACO algorithms appear to be currently the best-performing. SLS methods for the PTSP can be grouped into two classes: *analytical computation* and *empirical estimation*.

Much of the early ACO algorithms for the PTSP are based on analytical computation, that is, they compute the exact expected cost of the *a priori* solution using a closed-form expression derived by Jaillet (1985). Bianchi et al. (2002a,b) adopted this closed-form expression in an ant colony system (ACS) and compared it with a version of ACS for the TSP. The preliminary results showed that the PTSP-specific approach is more effective than its' TSP counterpart when the instance probability values are less than 0.5. Branke and Guntsch (2004) explored the idea to employ an ad-hoc approximation to replace the exact PTSP objective function, and showed that the computation time can be significantly reduced without major loss in solution quality. Bianchi (2006) and Bianchi and Gambardella (2007) proposed `pACS+1-shift`, which integrates the PTSP-specific ACS with `1-shift` (Bertsimas and Howell, 1993; Bianchi et al., 2005), a local search tailored for the PTSP. The experimental results showed that `pACS+1-shift` significantly outperforms all other algorithms proposed so far in the literature, and it is up to now considered as the best-performing SLS method for the PTSP.

Instead of using analytical computation, estimation-based algorithms estimate the expected cost of the *a priori* solution by Monte Carlo simulation. Gutjahr (2003, 2004) proposed S-ACO, a general-purpose estimation-based ACO algorithm for tackling stochastic combinatorial optimization problems, and took the PTSP as a test bed. In S-ACO, the solutions produced at a given iteration

are compared on the basis of a single realization; then the iteration-best solution is compared with the best-so-far solution on the basis of a number of realizations that increases with the number of iterations according to a static scheme. Gutjahr (2004) also studied a variant of S-ACO called S-ACOa, in which the number of realizations needed for comparing the iteration-best with the best-so-far solution is determined dynamically based on a statistical test. Birattari et al. (2006) proposed ACO/F-Race, which adopts the F-Race procedure (Birattari, 2004). In this algorithm, the solutions produced at a given iteration, together with the best-so-far solution, are compared using a pairwise statistical test for multiple comparisons. The preliminary results showed that for the instances with probability values less than 0.5, ACO/F-Race achieved solution costs that are significantly better than those of S-ACO and S-ACOa. However, S-ACO, S-ACOa, and ACO/F-Race are not expected to perform as well as `pACS+1-shift`. This is due to the following facts: Firstly, these algorithms are proposed as proof-of-concepts; secondly, they are based on ant system, which is not typically as well performing as ACS; thirdly, they do not use any local search as a subsidiary solution improvement procedure. Note that the adoption of local search is crucial to the performance of ACO algorithms (Dorigo and Stützle, 2004). Bianchi (2006) and Bianchi and Gambardella (2007) also considered the estimation-based solution evaluation approach of Gutjahr (2003, 2004) in `pACS+1-shift`, but concluded that this variant is significantly worse performing than the analytical computation variant of `pACS+1-shift`.

Besides the current research effort in solving the PTSP, our recent research results in the context of local search algorithms (Balaprakash et al., 2007b; Birattari et al., 2008) have shown that (i) the estimation-based approaches are effective alternatives to the analytical computation approaches; (ii) the new estimation-based local search procedure `2.5-opt-EEais` that we developed is very effective for solving the PTSP: it reaches significantly better solutions for a wide range of instances and it is by two to three orders of magnitude faster than `1-shift`. These recent results indicate that there is also a significant potential to improve over `pACS+1-shift`. All the aforementioned factors motivated us to develop a new state-of-the-art algorithm that adopts the estimation-based approach in the ACO framework, with the goal of effectively solving the PTSP.

In order to assess the impact of each algorithmic component that we use, we adopt the following systematic bottom-up design: in Section 4, we integrate `2.5-opt-EEais` into `pACS` and we show that `pACS+2.5-opt-EEais` outperforms `pACS+1-shift`; in Section 5, we bring the estimation-based solution evaluation into `pACS+2.5-opt-EEais` and we show that the cost evaluation performed by the estimation-based approach is comparable to that of the analytical computation approach; in Section 6, we customize three high performing ACO variants, $\mathcal{MAX}$–$\mathcal{MIN}$ ant system, rank-based ant system, and best-worst ant system. We compare the three variants to ACS and we show that the differences in solution costs among the four ACO variants are minor, once their parameters are fine tuned. It should be noted that all the four estimation-based ACO variants outperform the previously best ACO algorithm, `pACS+1-shift`.

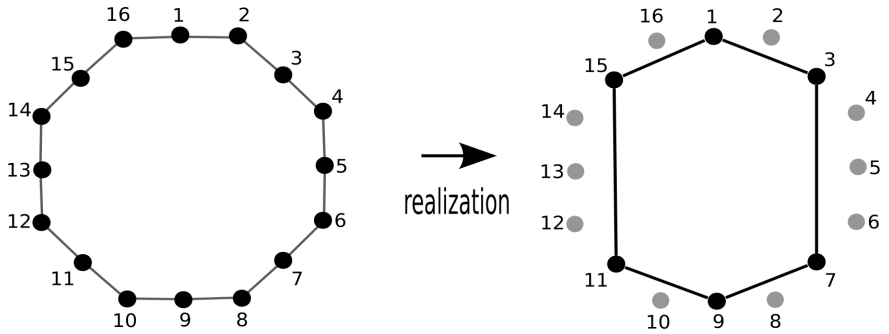

Figure 1: An *a priori* solution for a PTSP instance with 16 nodes, where the nodes are visited in the following order: 1, 2, 3,..., 15, 16, and 1. Let us assume that the nodes 1, 3, 7, 9, 11, and 15 are prescribed to be visited by a realization of $\boldsymbol{\omega}$. The *a posteriori* solution visits the nodes following the *a priori* solution but skipping the gray nodes that do not require to be visited.

2 The probabilistic traveling salesman problem

A PTSP instance can be defined on a weighted graph $G = \{V, E, C, P\}$ with $V = \{1, 2, ..., n\}$ being a set of nodes, $E = \{\langle i, j\rangle | i, j \in V, i \neq j\}$ being a set of edges that connect the nodes, $C = \{c_{ij} | \langle i, j\rangle \in E\}$ being a set of travel costs, where c_{ij} is the travel cost imposed on edge $\langle i, j\rangle \in E$, and $P = \{p_i | i \in V, 0 \leq p_i \leq 1\}$ being a set of probabilities, where p_i is the probability that node i requires being visited. It is assumed that an event of visiting a node is independent of visiting other nodes. The probabilistic information can be modeled using a random variable $\boldsymbol{\omega}$ that follows an n-variate Bernoulli distribution. A realization of $\boldsymbol{\omega}$ is a vector of binary values, where a value `'1'` in position i indicates that node i requires being visited whereas a value `'0'` means that it does not need a visit. Note that the travel costs are assumed to be symmetric—for all pairs of nodes i, j we have $c_{ij} = c_{ji}$. A PTSP instance is called homogeneous if all probabilities in the set P are the same, and it is called heterogeneous otherwise.

The PTSP is usually tackled by the *a priori* optimization approach in two stages. First, before the realization of ω is known, a Hamitonian tour containing all the nodes is constructed, which is called an *a priori* solution; once the nodes that require being visited are known, the *a posteriori* solution is obtained by following the nodes in the order of the *a priori* solution and by excluding the nodes that need not be visited. The goal is to find an *a priori* solution with the minimum expected *a posteriori* solution cost.

Suppose $x = (\pi(1), \pi(2), \ldots, \pi(n), \pi(1))$ is an *a priori* solution for the PTSP, where π is a permutation of the set V. The analytical computation approach for evaluating the expected cost $F(x)$ of the *a priori* solution x uses the following

closed-form expression (Jaillet, 1985):

$$F(x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{\pi(i)\pi(j)}\ p_{\pi(i)} p_{\pi(j)} \prod_{k=i+1}^{j-1} (1 - p_{\pi(k)}) + \sum_{j=1}^{n} \sum_{i=1}^{j-1} c_{\pi(j)\pi(i)}\ p_{\pi(i)} p_{\pi(j)} \prod_{k=j+1}^{n} (1 - p_{\pi(k)}) \prod_{k=1}^{i-1} (1 - p_{\pi(k)}). \quad (1)$$

Note that for a homogenous instance with probability p and size n, Equation 1 reduces to $F(x) = \sum_{i=1}^{n} \sum_{j=1}^{n-1} p^2 (1-p)^{j-1}\ c_{\pi(i),\pi(i+j)}$.

In the empirical estimation approach for the PTSP, the cost $F(x)$ is empirically estimated on the basis of sample costs of *a posteriori* solutions $f(x, \omega_1), f(x, \omega_2), \ldots, f(x, \omega_M)$ obtained from M independent realizations $\omega_1, \omega_2, \ldots, \omega_M$ of the random variable $\boldsymbol{\omega}$:

$$\hat{F}_M(x) = \frac{1}{M} \sum_{r=1}^{M} f(x, \omega_r). \quad (2)$$

As it can be shown easily, $\hat{F}_M(x)$ is an *unbiased* estimator of $F(x)$.

3 The `pACS+1-shift` algorithm

`pACS+1-shift` (Bianchi, 2006; Bianchi and Gambardella, 2007) is currently the best performing ant colony optimization algorithm for the PTSP. It is a standard ACS algorithm (Dorigo and Gambardella, 1997) in which, at each iteration, m ants construct solutions in the following way: with a probability q_0, an ant k at node i chooses to move to the node j that maximizes the product $\tau_{ij}\eta_{ij}^{\beta}$; with probability $1 - q_0$, the next node j is chosen with probability $p_{ij}^k = \tau_{ij}\eta_{ij}^{\beta} / \sum_{l \in N_i^k} \tau_{il}\eta_{il}^{\beta}$ (the random proportional rule); τ_{ij} and $\eta_{ij} = 1/c_{ij}$ are the pheromone value and the heuristic value associated with edge $\langle i, j \rangle$, respectively; β is a parameter that determines the relative influence of the heuristic information; N_i^k is the set of nodes to which it is feasible to move from node i. When an ant moves from node i to node j, the pheromone value associated with the edge $\langle i, j \rangle$ is updated to $\tau_{ij} = (1-\varphi) \cdot \tau_{ij} + \varphi \cdot \tau_0$, where $\varphi \in (0, 1]$ is a parameter, and τ_0 is the initial value of the pheromone. At the end of each iteration, the pheromone value associated with each edge $\langle i, j \rangle$ of the best-so-far solution is updated to $\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij}^{best}$, where $\rho \in (0, 1]$ is a parameter. The solutions generated at each iteration are evaluated by Equation 1.

`1-shift` local search, a PTSP-specific iterative improvement algorithm, is applied to all solutions constructed by the ants prior to the pheromone update. The algorithm proceeds in two phases: the first phase consists in exploring a swap-neighborhood, where the set of neighbors of a given solution contains all the solutions that can be obtained by swapping two consecutive nodes. The second phase explores the node-insertion neighborhood in a fixed lexicographic order. The cost difference of neighboring solutions is obtained by

delta evaluation, a technique that considers only the cost contribution of solution components that are not common between the two solutions. This is done by recursive closed-form expressions, which are based on heavy mathematical derivations (Bianchi et al., 2005; Bianchi, 2006; Bianchi and Campbell, 2007).

4 Effectiveness of `2.5-opt-EEais` in `pACS`

In this section, we show that the adoption of `2.5-opt-EEais` instead of `1-shift` as a subsidiary solution improvement procedure significantly improves the effectiveness of `pACS`.

`2.5-opt-EEais` (Balaprakash et al., 2007b) is the state-of-the-art iterative improvement algorithm for the PTSP. `2.5-opt-EEais` differs from `1-shift` in the following three elements: it adopts an empirical estimation technique in the delta evaluation; it uses the 2.5-exchange neighborhood relation that combines the 2-exchange and node-insertion neighborhoods (Bentley, 1992); and it exploits typical TSP neighborhood reduction techniques such as fixed-radius search, candidate lists, and don't look bits (Martin et al., 1991; Bentley, 1992; Johnson and McGeoch, 1997). The effectiveness of the algorithm is further enhanced by the usage of variance reduction techniques such as the method of common random numbers, adaptive sample size, and importance sampling. In particular, importance sampling is used to tackle effectively the instances with very low probability values and it is applied as follows: in a 2-exchange move, whenever the number of nodes in a segment (a 2-exchange move always leads to two segments) is less than $min_{is}\%$ of the instance size, $w\%$ nodes of the shorter segment are biased with probability p'. See Figure 2 for an example. For the node-insertion move, only the insertion node is biased with a value p''. For a more detailed explanation of `2.5-opt-EEais`, we refer the reader to Balaprakash et al. (2007b). We denote `pACS+2.5-opt-EEais` the algorithm obtained by combining `pACS` with `2.5-opt-EEais`.

We tuned the four parameters of `2.5-opt-EEais` through a parameter tuning algorithm, Iterative F-Race (Balaprakash et al., 2007a). For tuning, we used homogeneous instances obtained as follows: TSP instances are generated with the DIMACS instance generator (Johnson et al., 2001) from which the PTSP instances are obtained by associating a probability value to each node. We used clustered instances of 1000 nodes, in which the nodes are arranged in a number of clusters in a $10^6 \times 10^6$ square. We considered values for p from 0.050 to 0.200 with an increment of 0.025 and from 0.3 to 0.5 with an increment of 0.1. We focus on probability values up to 0.5 because Bianchi and Gambardella (2007) showed that the instances with probability values greater than 0.5 can effectively be solved as a TSP by the concorde solver (Applegate et al., 2001). The generated instances are grouped into three classes according to the value of p: $\{0.050, 0.075, 0.100\}$ (Class-I), $\{0.150, 0.175, 0.200\}$ (Class-II), $\{0.300, 0.400, 0.500\}$ (Class-III); for each probability level we generated 30 instances. The parameters of `2.5-opt-EEais` are fine tuned on each instance class. Table 1 shows the range of each parameter

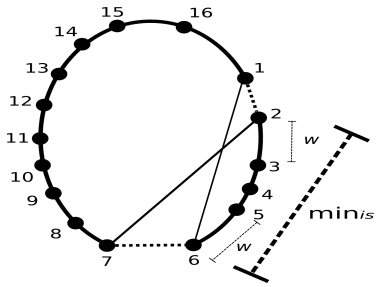

Figure 2: In this example, the two edges $\langle 1, 2\rangle$ and $\langle 6, 7\rangle$ are deleted and replaced with $\langle 1, 6\rangle$ and $\langle 2, 7\rangle$ by a 2-exchange move. Assume that min_{is} and w are set to 50 and 40, respectively. Since the number of nodes in the segment $[2, \ldots, 6]$ is less than 50% of 16, which is eight, importance sampling is used to bias 40% of 5, that is, two nodes, between 2 and 6, one on each side of the segment. The nodes that are biased are 2, 3, 5, and 6.

Table 1: Parameters values for `2.5-opt-EEais`

algorithm	parameters	range	selected value		
			Class-I	Class-II	Class-III
2.5-opt-EEais	min_{is}	[0.0, 50.0]	42.0	46.0	2.40
	w	[0.0, 20.0]	13.0	16.0	5.80
	p'	[0.0, 1.0]	0.003	0.47	0.70
	p''	[0.0, 1.0]	0.92	0.67	0.95

given to the tuning algorithm and the selected value. For `pACS`, we adopted the parameter values suggested by Bianchi and Gambardella (2007); Bianchi (2006).

`pACS+1-shift` and `pACS+2.5-opt-EEais` are evaluated on the homogeneous PTSP instances used by Bianchi (2006), which are obtained by assigning a probability value to each node for TSPLIB instances, ch150, d198, lin318, att532, and rat783. The algorithms were implemented in `C` and compiled with `gcc`, version 3.3. Experiments were carried out on AMD Opteron^{TM}244 1.75 GHz processors with 1 MB L2-Cache and 2 GB RAM, running under Rocks Cluster GNU/Linux. We used the stopping criterion suggested by Bianchi and Gambardella (2007) and by Bianchi (2006), where each algorithm is allowed to run for a computation time of $n^2/100$ CPU seconds. The computational results obtained on the instance rat783 are shown in Table 2 and Figure 3.

The results show that the adoption of `2.5-opt-EEais` in `pACS` is indeed very effective. The average cost of the solutions found by `pACS+2.5-opt-EEais` is between 2.0% to 12.1% less than those of `pACS+1-shift` and the observed difference is significant according to the Student t-test. An exception is for $p = 0.050$, where `pACS+1-shift` obtains an average solution cost that is 0.236% less than that of `pACS+2.5-opt-EEais`. The general trends of the experimental results obtained on the other instance sizes are similar; we refer the reader to

Table 2: Comparison of the average cost obtained by `pACS+2.5-opt-EEais` and `pACS+1-shift` over 30 independent runs on instance rat783. See footnote 1 on this page for an explanation of the contents and the typographic conventions adopted in the table.

	`pACS+2.5-opt-EEais` vs. `pACS+1-shift`	
p	Difference	95% CI
0.050	*+0.236*	[+0.014, +0.459]
0.075	**−2.022**	[−2.509, −1.534]
0.100	**−3.029**	[−3.430, −2.628]
0.150	**−5.214**	[−5.784, −4.644]
0.175	**−5.798**	[−6.270, −5.326]
0.200	**−6.214**	[−6.661, −5.766]
0.300	**−9.401**	[−9.924, −8.877]
0.400	**−10.760**	[−11.451, −10.069]
0.500	**−12.176**	[−12.759, −11.593]

Balaprakash et al. (2008) for the complete set of results and for the absolute values.

5 Estimation-based ant colony system

In order to design a complete estimation-based ACS, in particular, to make the solution evaluation approach of ACS coherent with that of the underlying iterative improvement algorithm, we modified `pACS+2.5-opt-EEais` in such a way that the solution costs are evaluated using Equation 2 instead of Equation 1. In particular, for each solution x^i, an *unbiased* estimator $\hat{F}_{M_i}(x^i)$ of $F(x^i)$ is obtained through M_i independent realizations of $\boldsymbol{\omega}$. Estimating solution costs with low variance is crucial to the effectiveness of the estimation approach. In a similar spirit as for `2.5-opt-EEais`, we address this issue using two variance reduction techniques: (i) the method of common random numbers and (ii) an adaptive sample size.

As in `2.5-opt-EEais`, the adoption of the method of common random numbers for ACS consists in using a same set of realizations to evaluate the solutions produced at each iteration. The adaptive sample size in `2.5-opt-EEais` is implemented using the sequential application of a parametric statistical test, Students t-test, which is appropriate for comparing two solutions. However, since in ACS more than two solutions are compared at each iteration, we use

[1] For a given comparison A vs. B, the table reports the observed relative difference between the two algorithms A and B and a 95% confidence interval (CI) obtained through the t-test. Concerning the relative difference, if the value is positive (negative), the algorithm A (B) obtained an average cost that is larger (smaller) than the one obtained by the algorithm B (A). In this case, the value is typeset in boldface (italics) if it is significantly different from zero according to the t-test, at a confidence of 95%.

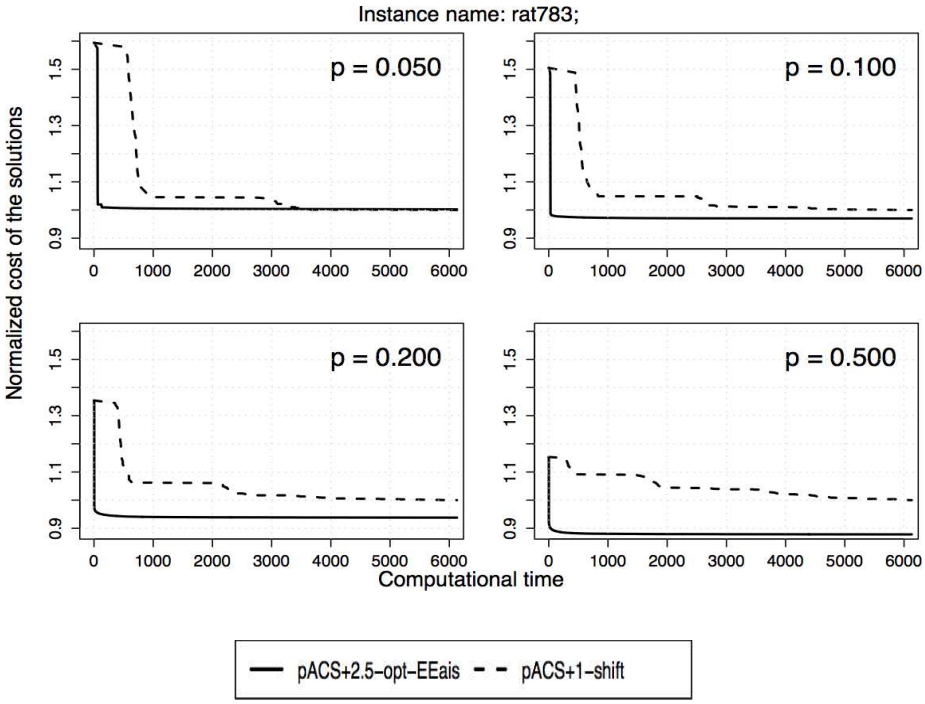

Figure 3: Experimental results on the instance rat783. The plots represent the cost of the solutions obtained by `pACS+2.5-opt-EEais` normalized by the one obtained by `pACS+1-shift`. The normalization is done on a run-by-run basis for 30 runs; the normalized solution cost is then aggregated.

a parametric statistical test based on analysis of variance (ANOVA) (Fisher, 1925) and Tuckeys' honestly significant differences (HSD) test (Tukey, 1949) for multiple comparison. We implemented the adaptive sample size procedure as a racing algorithm (Birattari, 2004): at each iteration, the *a posteriori* solution cost of each *a priori* solution is computed sequentially on realizations. Once M_{min} realizations have been used, where M_{min} is a parameter, ANOVA is applied on a realization-by-realization basis to test the null hypothesis that the cost estimates of all solutions are equal. The rejection of the null hypothesis indicates that there is at least one solution whose cost estimate is significantly worse than the one with best cost estimate. This particular worse solution is identified using Tuckeys' HSD and it is eliminated from further evaluation. The procedure terminates when a single solution remains or when a maximum number M of realizations is used, where M is a parameter. If more than one solution survives at the end, the solution with the best cost estimate is selected as the best solution. We denote this procedure ANOVA-Race. Note that the aforementioned cost evaluation procedure takes place after the solutions constructed by the ants are improved by `2.5-opt-EEais`. We denote the complete

estimation-based algorithm `ACS-EE`, where `EE` stands for empirical estimation.

We evaluate `ACS-EE` and `pACS+2.5-opt-EEais` on clustered homogeneous PTSP instances of size 1000, which are generated afresh using the DIMACS instance generator as described in Section 4. For the instance size 1000, Bianchi (2006) and Bianchi and Gambardella (2007) used 10000 ($= n^2/100$) CPU seconds as a stopping criterion, which allowed `pACS+1-shift` to perform more than five iterations. Such a high computation time is needed because the computational complexity of `1-shift` is very high. Since `2.5-opt-EEais` is between two and three orders of magnitude faster than `1-shift` (Birattari et al., 2008; Balaprakash et al., 2007b), we study the algorithms under 100 and 1000 CPU seconds. The adoption of 10 CPU seconds is not insightful because it does not allow the algorithms to perform more than five iterations. Note that Equation 1 is used for the post-evaluation of the best-so-far solutions found by `ACS-EE`.

The parameters of the adaptive sampling procedure are fixed *a priori*: M_{min} is set to 5 and M is set to 1000. The null hypothesis is rejected at a significance level of 0.05. `ACS-EE` adopts the same parameter values as `pACS+2.5-opt-EEais`. `ACS-EE` uses a same set of realizations for all iterations. In the context of the PTSP, this strategy is more effective than changing realizations for each iteration (Birattari et al., 2008). However, the realizations are selected randomly from the given set for each iteration. Note that the implementation of `ACS-EE` and `pACS+2.5-opt-EEais` is based on ACOTSP (Stützle, 2002) and the two algorithms differ only in the solution evaluation procedure.

The computational results in Table 3 show that for both stopping criteria the two algorithms are comparable to each other. With 95% confidence, under the current experimental setting, we can state that should ever the expected cost of solutions found by `ACS-EE` be higher than the one of those found by `pACS+2.5-opt-EEais`, the difference between the expected costs would be at most 0.73% and 0.48% under 100 CPU seconds and 1000 CPU seconds, respectively.

We also tested the algorithms on instances with $p > 0.5$, where we found that `ACS-EE` is significantly better than `pACS+2.5-opt-EEais`. This can be explained as follows: instances with high probability values have low coefficient of variation (Balaprakash et al., 2007b). In this case, ANOVA-Race needs only few realizations to select the best solution. This allows `ACS-EE` to perform more iterations when compared to `pACS+2.5-opt-EEais`, which eventually results in higher quality solutions.

Note that the results presented in this section and in Section 4 contradict the ones reported in Bianchi and Gambardella (2007), where, in `pACS+1-shift` the adoption of an estimation-based approach is shown to be less effective than that of the analytical computation approach. This contradiction is due to the fact that the estimation-based approach adopted by Bianchi and Gambardella (2007) is not tailored to the PTSP: it is a general purpose procedure proposed for ACO that is allowed to run for a relatively long computation time without any iterative improvement algorithm.

Table 3: Comparison of the average cost obtained by `ACS-EE` and `pACS+2.5-opt-EEais` over 30 clustered homogeneous instances. See footnote 1 on page 8 for an explanation of the contents and the typographic conventions adopted.

100 CPU seconds			1000 CPU seconds		
	`ACS-EE` vs. `pACS+2.5-opt-EEais`			`ACS-EE` vs. `pACS+2.5-opt-EEais`	
p	Difference	95% CI	p	Difference	95% CI
0.050	−0.540	[−1.246, +0.166]	0.050	+0.117	[−0.254, +0.488]
0.075	+0.139	[−0.458, +0.736]	0.075	+0.024	[−0.046, +0.095]
0.100	+0.044	[−0.273, +0.361]	0.100	+0.019	[−0.039, +0.077]
0.150	−0.061	[−0.392, +0.270]	0.150	+0.041	[−0.063, +0.145]
0.175	+0.129	[−0.116, +0.374]	0.175	+0.111	[−0.033, +0.256]
0.200	−0.077	[−0.389, +0.235]	0.200	+0.043	[−0.113, +0.199]
0.300	−0.192	[−0.437, +0.054]	0.300	−0.030	[−0.139, +0.078]
0.400	−0.001	[−0.209, +0.208]	0.400	−0.068	[−0.154, +0.019]
0.500	−0.156	[−0.411, +0.099]	0.500	−0.052	[−0.154, +0.050]
0.600	**−0.324**	[−0.555, −0.094]	0.600	−0.019	[−0.116, +0.077]
0.700	**−0.486**	[−0.763, −0.208]	0.700	−0.089	[−0.257, +0.078]
0.800	**−0.618**	[−0.809, −0.426]	0.800	**−0.206**	[−0.308, −0.104]
0.900	**−0.994**	[−1.221, −0.766]	0.900	**−0.155**	[−0.294, −0.015]

6 Comparison between estimation-based ACO variants

One may wonder why ACS was adopted in Bianchi et al. (2002a,b), Bianchi (2006), and Bianchi and Gambardella (2007). It is rather likely that this choice is due to the authors background and their expertise with ACS, which has been developed by Dorigo and Gambardella (1997). Although ACS is a high performing ACO variant, we cannot rule out other existing variants as promising for the PTSP. This is due to the fact that there is no theoretical justification or empirical evidence in the PTSP literature that ACS is the best choice. We address this issue by comparing several ACO variants. In addition to ACS, we have considered the following three high performing variants: $\mathcal{MAX}$–$\mathcal{MIN}$ ant system (MMAS) (Stützle and Hoos, 2000), rank-based ant system (RAS) (Bullnheimer et al., 1999), and best-worst ant system (BWAS) (Cordón et al., 2002).

In all three variants, m ants construct solutions only using the random proportional rule and they differ from ACS with respect to the pheromone update procedure. In MMAS, only the iteration-best or best-so-far ant updates the pheromone trail associated with each edge $\langle i, j \rangle$ to $\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \Delta\tau_{ij}^{best}$, where ρ is a parameter and $\Delta\tau_{ij}^{best} = 1/C^{best}$. The value of C^{best} is equal to the cost of the iteration-best or best-so-far solution depending on which of the two

is chosen. The pheromone values are limited within a maximum and a minimum value in order to reduce the risk of search stagnation; in case of search stagnation, the search is restarted by re-initializing the pheromone values. In RAS, at each iteration, from m solutions only the $(w-1)$ best ranked solutions and the best-so-far solution are allowed to update the pheromone values using the following equation: $\tau_{ij} = \tau_{ij} + \sum_{r=1}^{w-1}(w-r)\Delta\tau_{ij}^{r} + w\Delta\tau_{ij}^{best}$, where r is the rank of the solution obtained by sorting m solutions by increasing cost, $\Delta\tau_{ij}^{r} = 1/C^{r}$, and $\Delta\tau_{ij}^{best} = 1/C^{best}$ if edge $\langle i,j\rangle$ belongs to the best-so-far solution; C^{r} and C^{best} are the cost of the solution with rank r and the cost of the best-so-far solution, respectively. In BWAS, only the best-so-far solution is allowed to update the pheromone values; the pheromone values of the edges that belong to the worst ant but not to the best-so-far solution are reduced. To avoid premature convergence, BWAS uses pheromone re-initialization and pheromone mutation.

All the aforementioned variants are extended to solve the PTSP by using ANOVA-Race to evaluate the solution costs and by using `2.5-opt-EEais` as the underlying solution improvement procedure. We denote them `MMAS-EE`, `RAS-EE`, and `BWAS-EE`.

Similar to `ACS-EE`, the implementations of `MMAS-EE`, `RAS-EE`, and `BWAS-EE` were based on ACOTSP (Stützle, 2002). We evaluate all the variants on a new set of clustered homogeneous PTSP instances of size 1000 obtained using the DIMACS instance generator. We allowed each variant to run for 100 and 1000 CPU seconds. Concerning the parameter values of each variant, we use two sets of values: default parameter values and tuned parameter values. We present the empirical results in the following three sections.

6.1 Experiments with default parameter values

The default parameter values for each variant are chosen reasonably close to the values proposed in the ACO literature for the TSP (Dorigo and Stützle, 2004; Bullnheimer et al., 1999; Cordón et al., 2002): in all the variants m, α, and β are set to 10, 1.0, and 2.0, respectively; in `ACS-EE`, ρ and q_0 are set to 0.1 and 0.98, respectively; in `MMAS-EE`, ρ is set to 0.2; in `RAS-EE`, ρ and w are set to 0.5 and 6, respectively; in `BWAS-EE`, ρ is set to 0.2. We denote the variants that adopt the default parameter values as `ACS-EE(d)` `MMAS-EE(d)`, `RAS-EE(d)`, and `BWAS-EE(d)`.

The results are shown in Table 4. For most probability levels, `ACS-EE(d)` is better than other variants: the average cost of `ACS-EE(d)` is between 0.4% and 3.9% and between 0.1% and 3.9% less than that of other variants for 100 and 1000 CPU seconds, respectively. Most of the differences that have been observed are statistically significant according to the t-test.

6.2 Comparison between the variants with tuned and default values

The parameter values of each variant are tuned separately for 100 and 1000 CPU seconds in the same way as described in Section 4 using Iterative F-Race. The selected values are shown in Tables 5. We denote the variants that use the fine tuned parameter values as `ACS-EE(t)`, `MMAS-EE(t)`, `RAS-EE(t)`, and `BWAS-EE(t)`.

The results from Table 6 show that, as expected, the adoption of tuned parameter values allows each variant to achieve much better results. `MMAS-EE(t)`, `RAS-EE(t)`, and `BWAS-EE(t)` profit much more from tuning than `ACS-EE(t)` does. For 100 CPU seconds, the observed improvements are very large and are up to 8.6%. For 1000 CPU seconds, the improvement is up to 3.5%.

Table 4: Comparison of the average cost obtained by `ACS-EE(d)`, `MMAS-EE(d)`, `RAS-EE(d)`, and `BWAS-EE(d)`, on the clustered instances of size 1000 for 100 and 1000 seconds. See footnote 1 on page 8 for an explanation of the contents and the typographic conventions adopted.

	100 CPU seconds					
	`ACS-EE(d)` vs. `MMAS-EE(d)`		`ACS-EE(d)` vs. `RAS-EE(d)`		`ACS-EE(d)` vs. `BWAS-EE(d)`	
p	Difference	95% CI	Difference	95% CI	Difference	95% CI
0.050	*+2.669*	[+0.613, +4.726]	+1.435	[−0.582, +3.451]	+0.748	[−1.386, +2.883]
0.075	**−2.838**	[−3.421, −2.255]	**−3.417**	[−4.030, −2.803]	**−3.144**	[−3.649, −2.638]
0.100	**−3.905**	[−4.362, −3.448]	**−1.642**	[−2.049, −1.235]	**−1.057**	[−1.729, −0.385]
0.150	**−0.696**	[−0.882, −0.511]	**−0.605**	[−0.838, −0.373]	−0.117	[−0.289, +0.055]
0.175	**−1.034**	[−1.238, −0.830]	**−0.988**	[−1.176, −0.801]	**−0.417**	[−0.623, −0.212]
0.200	**−1.162**	[−1.357, −0.968]	**−1.081**	[−1.257, −0.904]	**−0.523**	[−0.688, −0.358]
0.300	**−2.247**	[−2.451, −2.043]	**−2.015**	[−2.164, −1.866]	**−1.175**	[−1.374, −0.975]
0.400	**−3.111**	[−3.323, −2.898]	**−2.883**	[−3.068, −2.699]	**−1.586**	[−1.773, −1.399]
0.500	**−3.319**	[−3.528, −3.110]	**−3.293**	[−3.477, −3.109]	**−1.725**	[−1.925, −1.525]

	1000 CPU seconds					
	`ACS-EE(d)` vs. `MMAS-EE(d)`		`ACS-EE(d)` vs. `RAS-EE(d)`		`ACS-EE(d)` vs. `BWAS-EE(d)`	
p	Difference	95% CI	Difference	95% CI	Difference	95% CI
0.050	**−1.893**	[−2.222, −1.563]	**−1.076**	[−1.427, −0.725]	**−0.905**	[−1.176, −0.635]
0.075	**−1.803**	[−2.008, −1.599]	**−1.371**	[−1.517, −1.224]	**−0.827**	[−1.010, −0.643]
0.100	**−0.746**	[−0.846, −0.646]	**−0.700**	[−0.793, −0.608]	**−0.378**	[−0.464, −0.293]
0.150	**−0.790**	[−0.873, −0.708]	**−1.262**	[−1.363, −1.160]	**−0.566**	[−0.655, −0.477]
0.175	**−0.925**	[−1.032, −0.818]	**−1.738**	[−1.831, −1.644]	**−0.553**	[−0.637, −0.468]
0.200	**−0.956**	[−1.057, −0.856]	**−2.120**	[−2.236, −2.004]	**−0.451**	[−0.544, −0.358]
0.300	**−0.618**	[−0.730, −0.506]	**−3.189**	[−3.307, −3.070]	**−0.275**	[−0.379, −0.172]
0.400	**−0.230**	[−0.327, −0.134]	**−3.608**	[−3.769, −3.447]	**−0.292**	[−0.402, −0.181]
0.500	**−0.139**	[−0.246, −0.032]	**−3.900**	[−4.031, −3.768]	**−0.406**	[−0.501, −0.310]

Table 5: Fine tuned parameters values

100 CPU seconds

algorithm	parameters	range	selected value		
			Class-I	Class-II	Class-III
ACS-EE	m	$[3, 15]$	5	4	11
	β	$[0.0, 5.0]$	3.3	0.16	1.0
	ρ	$[0.001, 1.0]$	0.75	0.84	1.0
	q_0	$[0.0, 1.0]$	0.84	1.0	0.99
MMAS-EE	m	$[3, 15]$	5	4	15
	α	$[0.001, 1.5]$	1.4	1.3	0.99
	β	$[0.0, 5.0]$	3.2	0.97	2.1
	ρ	$[0.001, 1.0]$	1.0	1.0	0.97
RAS-EE	m	$[3, 15]$	3	3	6
	α	$[0.001, 1.5]$	0.33	1.1	0.71
	β	$[0.0, 5.0]$	5.0	2.6	2.1
	ρ	$[0.001, 1.0]$	1.0	0.94	0.83
	w	$[1, 10]$	1	1	1
BWAS-EE	m	$[3, 15]$	3	4	4
	α	$[0.001, 1.5]$	0.99	1.4	0.89
	β	$[0.0, 5.0]$	3.1	2.9	2.3
	ρ	$[0.001, 1.0]$	0.95	0.97	0.66

1000 CPU seconds

algorithm	parameters	range	selected value		
			Class-I	Class-II	Class-III
ACS-EE	m	$[3, 15]$	4	3	5
	β	$[0.0, 5.0]$	0.05	0.85	3.7
	ρ	$[0.001, 1.0]$	0.67	0.079	0.82
	q_0	$[0.0, 1.0]$	0.99	0.99	0.96
MMAS-EE	m	$[3, 15]$	8	15	6
	α	$[0.001, 1.5]$	1.5	1.2	1.1
	β	$[0.0, 5.0]$	1.6	1.9	0.95
	ρ	$[0.001, 1.0]$	0.99	0.98	0.62
RAS-EE	m	$[3, 15]$	10	6	11
	α	$[0.001, 1.5]$	1.2	1.5	1.4
	β	$[0.0, 5.0]$	0.85	2.1	2.7
	ρ	$[0.001, 1.0]$	1.0	0.57	0.37
	w	$[1, 10]$	1	1	1
BWAS-EE	m	$[3, 15]$	5	10	6
	α	$[0.001, 1.5]$	0.6	1.1	0.9
	β	$[0.0, 5.0]$	2.7	0.09	2.4
	ρ	$[0.001, 1.0]$	0.99	0.85	0.27

Table 6: Comparison of the average cost obtained by the four variants with tuned values to ones with default values on the clustered instances of size 1000 for 100 and 1000 seconds. See footnote 1 on page 8 for an explanation of the contents and the typographic conventions adopted.

100 CPU seconds

	ACS-EE(t) vs. ACS-EE(d)		MMAS-EE(t) vs. MMAS-EE(d)		RAS-EE(t) vs. RAS-EE(d)		BWAS-EE(t) vs. BWAS-EE(d)	
p	Difference	95% CI	Difference	95% CI	Difference	95% CI	Difference	95% CI
0.050	**−8.051**	[−9.708, −6.394]	**−5.030**	[−6.240, −3.819]	**−8.629**	[−10.088, −7.170]	**−7.874**	[−9.468, −6.281]
0.075	**−2.177**	[−2.727, −1.626]	**−5.838**	[−6.372, −5.305]	**−6.998**	[−7.582, −6.413]	**−5.706**	[−6.130, −5.282]
0.100	−0.208	[−0.486, +0.070]	**−4.813**	[−5.248, −4.378]	**−2.795**	[−3.143, −2.448]	**−1.741**	[−2.480, −1.001]
0.150	**−1.060**	[−1.278, −0.842]	**−1.751**	[−1.924, −1.579]	**−1.754**	[−1.950, −1.558]	**−1.134**	[−1.292, −0.977]
0.175	**−0.972**	[−1.170, −0.774]	**−2.062**	[−2.238, −1.885]	**−2.046**	[−2.170, −1.921]	**−1.263**	[−1.406, −1.119]
0.200	**−1.136**	[−1.287, −0.986]	**−2.268**	[−2.467, −2.069]	**−2.227**	[−2.423, −2.031]	**−1.479**	[−1.637, −1.321]
0.300	**−0.663**	[−0.824, −0.501]	**−2.779**	[−2.968, −2.591]	**−2.654**	[−2.826, −2.482]	**−1.577**	[−1.766, −1.388]
0.400	**−0.443**	[−0.612, −0.274]	**−3.282**	[−3.486, −3.077]	**−3.242**	[−3.435, −3.050]	**−1.601**	[−1.804, −1.399]
0.500	−0.053	[−0.178, +0.071]	**−3.192**	[−3.393, −2.991]	**−3.099**	[−3.297, −2.900]	**−1.157**	[−1.364, −0.949]

1000 CPU seconds

	ACS-EE(t) vs. ACS-EE(d)		MMAS-EE(t) vs. MMAS-EE(d)		RAS-EE(t) vs. RAS-EE(d)		BWAS-EE(t) vs. BWAS-EE(d)	
p	Difference	95% CI	Difference	95% CI	Difference	95% CI	Difference	95% CI
0.050	**−1.138**	[−1.423, −0.852]	**−2.641**	[−2.944, −2.338]	**−2.819**	[−3.062, −2.577]	**−1.654**	[−1.902, −1.405]
0.075	**−0.297**	[−0.360, −0.234]	**−2.077**	[−2.283, −1.871]	**−1.670**	[−1.812, −1.528]	**−1.094**	[−1.299, −0.888]
0.100	**−0.216**	[−0.260, −0.173]	**−0.949**	[−1.055, −0.843]	**−0.918**	[−1.016, −0.820]	**−0.582**	[−0.671, −0.494]
0.150	**−0.155**	[−0.223, −0.088]	**−0.941**	[−1.040, −0.842]	**−1.436**	[−1.532, −1.340]	**−0.681**	[−0.778, −0.585]
0.175	−0.038	[−0.111, +0.035]	**−1.056**	[−1.159, −0.954]	**−1.834**	[−1.921, −1.746]	**−0.610**	[−0.692, −0.529]
0.200	**−0.087**	[−0.170, −0.004]	**−1.067**	[−1.153, −0.980]	**−2.134**	[−2.257, −2.012]	**−0.440**	[−0.536, −0.345]
0.300	*+0.186*	[+0.077, +0.295]	**−0.721**	[−0.819, −0.624]	**−3.188**	[−3.294, −3.083]	**−0.106**	[−0.210, −0.001]
0.400	+0.073	[−0.016, +0.162]	**−0.228**	[−0.319, −0.138]	**−3.393**	[−3.570, −3.217]	−0.066	[−0.168, +0.036]
0.500	+0.104	[−0.001, +0.210]	−0.089	[−0.184, +0.007]	**−3.531**	[−3.663, −3.399]	−0.104	[−0.225, +0.017]

6.3 Comparison between the variants with tuned parameter values

The computational results of the four variants that adopt the tuned parameter values are given in Table 7. For the absolute values, we refer the reader to Balaprakash et al. (2008). From the results, we cannot identify a clear winner among the considered variants. For 100 CPU seconds, with a confidence level of 95%, under the current experimental setting, we can state that should ever the expected cost of the solutions found by `MMAS-EE(t)`, `RAS-EE(t)`, and `BWAS-EE(t)` be larger than those found by `ACS-EE(t)`, the difference would be at most 1.4%, 2.7% and 1.2%, respectively. For 1000 CPU seconds, the aforementioned differences would be at most 0.3%, 0.8% and 0.1%, respectively. There are few exceptions, where the differences are significant but rather small: for 100 CPU seconds, the maximum observed difference is less than 1% (except for $p = 0.050$ and $p = 0.075$, the average cost of `RAS-EE(t)` is 2.0% and 1.5% less than that of `ACS-EE(t)`, respectively) and for 1000 CPU seconds it is less than 0.7%.

From the absolute values reported in Balaprakash et al. (2008), we observed that for 1000 CPU seconds all the variants obtain average solution costs that are smaller than that of 100 CPU seconds; the improvements for 10 times increase of the computation time are in the range of 0.4% to 2.1% except for $p = 0.050$, where the improvements are between 2.9% to 4.5%.

In order to further assess the solution costs achieved by the variants for a very long computation time, we allowed the variants to run for 10000 CPU seconds as suggested by Bianchi and Gambardella (2007) and Bianchi (2006). The parameter values of each variant are the same as that of 1000 CPU seconds. The results are shown in Table 7. In spite of few significant differences between the variants, the general trend is similar to that of shorter computation times: there is no clear winner among the considered variants.

7 Conclusion and future work

The main contribution of the paper is the development and the empirical analysis of new state-of-the-art ACO algorithms for the PTSP. We used the current best performing ACO algorithm `pACS+1-shift` as a starting point. We showed that the adoption of the state-of-the-art iterative improvement algorithm `2.5-opt-EEais` allows `pACS` to obtain a significant improvement in the solution cost. To develop a complete estimation-based ACS, we adopted an estimation-based approach to evaluate the solution costs. Finally, we customized $\mathcal{MAX}$–$\mathcal{MIN}$ ant system, rank-based ant system, and best-worst ant system to solve the PTSP. We showed that all of them can be used to effectively tackle the PTSP provided that their parameter values are fine tuned. In a nutshell, we showed that the proposed estimation-based approach is an effective alternative to the analytical computation techniques when applying ACO and local search to the PTSP. Note that this conclusion contradicts the previous results reported

Table 7: Comparison of the average cost obtained by `ACS-EE(t)`, `MMAS-EE(t)`, `RAS-EE(t)`, and `BWAS-EE(t)`, on the clustered instances of size 1000 for 100, 1000, and 10000 seconds. See footnote 1 on page 8 for an explanation of the contents and the typographic conventions adopted.

100 CPU seconds

	`ACS-EE(t)` vs. `MMAS-EE(t)`		`ACS-EE(t)` vs. `RAS-EE(t)`		`ACS-EE(t)` vs. `BWAS-EE(t)`	
p	Difference	95% CI	Difference	95% CI	Difference	95% CI
0.050	−0.597	[−1.498, +0.303]	*+2.076*	[+1.392, +2.759]	+0.555	[−0.155, +1.264]
0.075	*+0.940*	[+0.425, +1.455]	*+1.590*	[+1.184, +1.996]	+0.481	[−0.023, +0.985]
0.100	*+0.744*	[+0.515, +0.972]	*+0.976*	[+0.809, +1.143]	*+0.487*	[+0.297, +0.676]
0.150	+0.002	[−0.156, +0.160]	+0.096	[−0.050, +0.243]	−0.042	[−0.213, +0.128]
0.175	+0.067	[−0.062, +0.196]	+0.097	[−0.025, +0.218]	−0.125	[−0.261, +0.012]
0.200	−0.018	[−0.181, +0.145]	+0.023	[−0.130, +0.175]	**−0.177**	[−0.307, −0.048]
0.300	−0.119	[−0.262, +0.024]	−0.011	[−0.166, +0.144]	**−0.256**	[−0.415, −0.098]
0.400	**−0.267**	[−0.414, −0.119]	−0.074	[−0.229, +0.082]	**−0.428**	[−0.573, −0.282]
0.500	**−0.185**	[−0.332, −0.037]	**−0.254**	[−0.406, −0.102]	**−0.628**	[−0.757, −0.500]

1000 CPU seconds

	`ACS-EE(t)` vs. `MMAS-EE(t)`		`ACS-EE(t)` vs. `RAS-EE(t)`		`ACS-EE(t)` vs. `BWAS-EE(t)`	
p	Difference	95% CI	Difference	95% CI	Difference	95% CI
0.050	**−0.378**	[−0.592, −0.164]	*+0.635*	[+0.445, +0.826]	**−0.385**	[−0.601, −0.169]
0.075	−0.019	[−0.042, +0.005]	+0.007	[−0.051, +0.064]	**−0.028**	[−0.055, −0.000]
0.100	−0.011	[−0.045, +0.022]	+0.003	[−0.038, +0.045]	−0.012	[−0.039, +0.016]
0.150	−0.003	[−0.060, +0.053]	+0.021	[−0.037, +0.080]	−0.039	[−0.087, +0.008]
0.175	*+0.095*	[+0.020, +0.170]	+0.060	[−0.019, +0.140]	+0.021	[−0.051, +0.092]
0.200	+0.025	[−0.046, +0.096]	−0.072	[−0.161, +0.018]	**−0.098**	[−0.174, −0.021]
0.300	*+0.291*	[+0.212, +0.369]	*+0.186*	[+0.106, +0.266]	+0.016	[−0.078, +0.110]
0.400	+0.071	[−0.017, +0.159]	**−0.149**	[−0.248, −0.051]	**−0.153**	[−0.234, −0.072]
0.500	+0.054	[−0.047, +0.155]	**−0.278**	[−0.378, −0.178]	**−0.198**	[−0.294, −0.102]

10000 CPU seconds

	`ACS-EE(t)` vs. `MMAS-EE(t)`		`ACS-EE(t)` vs. `RAS-EE(t)`		`ACS-EE(t)` vs. `BWAS-EE(t)`	
p	Difference	95% CI	Difference	95% CI	Difference	95% CI
0.050	**−0.165**	[−0.258, −0.071]	*+0.155*	[+0.102, +0.208]	*+0.087*	[+0.046, +0.128]
0.075	**−0.014**	[−0.027, −0.000]	+0.016	[−0.002, +0.034]	*+0.019*	[+0.007, +0.031]
0.100	**−0.028**	[−0.049, −0.007]	−0.044	[−0.113, +0.025]	−0.018	[−0.044, +0.007]
0.150	−0.035	[−0.078, +0.007]	+0.020	[−0.010, +0.050]	−0.050	[−0.114, +0.015]
0.175	−0.071	[−0.147, +0.006]	−0.001	[−0.072, +0.070]	−0.040	[−0.120, +0.040]
0.200	+0.003	[−0.084, +0.091]	−0.053	[−0.140, +0.034]	−0.013	[−0.121, +0.096]
0.300	+0.001	[−0.068, +0.070]	**−0.148**	[−0.239, −0.058]	**−0.094**	[−0.170, −0.018]
0.400	*+0.094*	[+0.012, +0.175]	**−0.334**	[−0.471, −0.197]	−0.034	[−0.127, +0.059]
0.500	+0.016	[−0.086, +0.119]	**−0.483**	[−0.618, −0.348]	−0.058	[−0.135, +0.018]

in the ACO literature for the PTSP.

Our next step in future work is the customization of effective TSP-specific SLS methods such as iterated local search, memetic algorithm and comparing them with the proposed estimation-based ACO algorithms. Further research effort will be devoted to design estimation-based SLS methods to solve stochastic vehicle routing problems. Another promising research direction is to investigate the application of estimation-based SLS methods to multi-objective stochastic routing problems.

Acknowledgments

The authors thank Leonora Bianchi for providing the source code of `pACS+1-shift`. This research has been supported by COMP^{2}SYS, a Marie Curie Training Site funded by the Commission of the European Community. The authors acknowledge support from F.R.S.-FNRS, the fund for scientific research of the French Community of Belgium.

References

D. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. Concorde—a code for solving traveling salesman problems, 2001. URL `http://www.math.princeton.edu/tsp/concorde.html`.

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, 2007a. Springer-Verlag.

P. Balaprakash, M. Birattari, T. Stützle, and M. Dorigo. Adaptive sample size and importance sampling in estimation-based local search for the probabilistic traveling salesman problem: A complete analysis. Technical Report TR/IRIDIA/2007-015, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2007b. URL `http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2007-015r002.pdf`.

P. Balaprakash, M. Birattari, T. Stützle, Z. Yuan, and M. Dorigo. Estimation-based ant colony optimization and local search for the probabilistic traveling salesman problem. IRIDIA Supplementary page, 2008. URL `http://iridia.ulb.ac.be/supp/IridiaSupp2008-018/`.

J. L. Bentley. Fast algorithms for geometric traveling salesman problems. *ORSA Journal on Computing*, 4(4):387–411, 1992.

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

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, July 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 Operational Research*, 176(1):131–144, 2007.

L. Bianchi and L. M. Gambardella. Ant colony optimization and local search based on exact and estimated objective values for the probabilistic traveling salesman problem. Technical Report IDSIA-06-07, IDSIA, USI-SUPSI, Manno, Switzerland, June 2007.

L. Bianchi, L. Gambardella, and M. Dorigo. Solving the homogeneous probabilistic travelling salesman problem by the ACO metaheuristic. In M. Dorigo, G. Di Caro, and M. Sampels, editors, *ANTS '02: Proceedings of the Third International Workshop on Ant Algorithms*, volume 2463 of *LNCS*, pages 176–187, Berlin, Germany, 2002a. Springer-Verlag.

L. Bianchi, L. M. Gambardella, and M. Dorigo. An ant colony optimization approach to the probabilistic traveling salesman problem. In J. J. Guervós, P. Adamidis, H. Beyer, J. L. Martín, and H. Schwefel, editors, *In Proceedings of the 7th International Conference on Parallel Problem Solving From Nature, PPSN VII*, volume 2439 of *LNCS*, pages 883–892, London, UK, 2002b. Springer-Verlag.

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*, 162:206–219, 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, and M. Dorigo. The ACO/F-RACE algorithm for combinatorial optimization under uncertainty. In K. F. Doerner, M. Gendreau, P. Greistorfer, W. J. Gutjahr, R. F. Hartl, and M. Reimann, editors, *Metaheuristics - Progress in Complex Systems Optimization*, Operations Research/Computer Science Interfaces Series, pages 189–203, Berlin, Germany, 2006. Springer-Verlag.

M. Birattari, P. Balaprakash, T. Stützle, and M. Dorigo. Estimation-based local search for stochastic combinatorial optimization using delta evaluations: A case study in the probabilistic traveling salesman problem. *INFORMS Journal on Computing*, (in press), 2008.

J. Branke and M. Guntsch. Solving the probabilistic TSP with ant colony optimization. *Journal of Mathematical Modelling and Algorithms*, 3(4):403–425, 2004.

B. Bullnheimer, R. F. Hartl, and C. Strauss. A new rank based version of the ant system: A computational study. *Central European Journal for Operations Research and Economics*, 7(1):25–38, 1999.

O. Cordón, I. F. de Viana, and F. Herrera. Analysis of the best-worst ant system and its variants on the TSP. *Mathware and Soft Computing*, 9(2–3):177–192, 2002.

G. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

M. Dorigo and M. Birattari. Swarm intelligence. *Scholarpedia*, 2(9):1462, 2007.

M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.

R. A. Fisher. *Statistical Methods for Research Workers*. Oliver and Boyd, London, UK, 1925.

M. Gendreau, G. Laporte, and R. Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88:3–12, 1996.

W. J. Gutjahr. A converging ACO algorithm for stochastic combinatorial optimization. In A. Albrecht and K. Steinhofl, editors, *Stochastic Algorithms: Foundations and Applications*, volume 2827 of *LNCS*, pages 10–25, Berlin, Germany, 2003. Springer-Verlag.

W. J. 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.

H. Hoos and T. Stützle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, San Francisco, CA, 2005.

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/`.

G. Laporte, F. Louveaux, and H. Mercure. A priori optimization of the probabilistic traveling salesman problem. *Operations Research*, 42:543–549, 1994.

O. Martin, S. W. Otto, and E. W. Felten. Large-step Markov chains for the traveling salesman problem. *Complex Systems*, 5(3):299–326, 1991.

T. Stützle. ACOTSP: A software package of various ant colony optimization algorithms applied to the symmetric traveling salesman problem, 2002. URL `http://www.aco-metaheuristic.org/aco-code/`.

T. Stützle and H. Hoos. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. *Future Generation Computer Systems*, 16(8):889–914, 2000.

J. W. Tukey. Comparing individual means in the analysis of variance. *Biometrics*, 5(2):99–114, 1949.