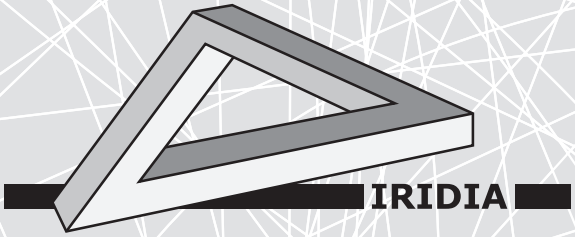

Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Estimation-based Metaheuristics for the Probabilistic Traveling Salesman Problem: A comparison to progressive approximation, the aggregation approach, and simulated annealing

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

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2009-025

October 2009

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/2009-025

Estimation-based Metaheuristics for the Probabilistic Traveling Salesman Problem: A comparison to progressive approximation, the aggregation approach, and simulated annealing

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

October 11, 2009

Abstract

The probabilistic traveling salesman problem (PTSP) is a central problem in stochastic routing. Recently, we have shown that empirical estimation is a promising approach to devise highly effective iterative improvement algorithms for the PTSP. In Balaprakash et al. (2009a), we proposed high performing estimation-based metaheuristics for the PTSP. In this technical report, we compare our estimation-based iterative improvement algorithm to the progressive approximation method and the aggregation approach. Moreover, for metaheuristics, we investigate the effectiveness of using initial solutions obtained via an exact TSP solver and the aggregation approach. Finally, we compare our estimation-based metaheuristics to simulated annealing algorithms. The results show that the estimation-based iterative improvement algorithm dominate the progressive approximation method. Using the aggregation approach, the iterative improvement algorithm reaches local optima faster on instances with very low probability values. However, the costs of local optima are significantly worse than that of our iterative improvement algorithm. The adoption of TSP optimal solutions or the solutions obtained from the aggregation approach as initial solutions in estimation-based algorithms does not give a significant benefit.

1 Introduction

The probabilistic traveling salesman problem (PTSP) is similar to the TSP with the main difference being that each node has a probability of

requiring a visit. The goal is to find a TSP tour that minimizes the expected cost of the pruned tour: this pruned tour is obtained only after knowing the nodes that require being visited and skipping the nodes that do not require being visited according to some predefined rules.

An instance of the PTSP is defined on a graph G with the following elements:

- a set $V = \{1, 2, \ldots, n\}$ of nodes;
- a set $A = \{\langle i, j\rangle : i, j \in V, i \neq j\}$ of edges, where an edge $\langle i, j\rangle$ connects the nodes i and j;
- a set $C = \{c_{ij} : \langle i, j\rangle \in A\}$ of travel costs, where c_{ij} is the cost of traversing an edge $\langle i, j\rangle$; the costs are assumed to be symmetric, that is, for all pairs of nodes i, j we have $c_{ij} = c_{ji}$;
- a set $P = \{p_i : i \in V\}$ of probabilities, where p_i specifies the probability that a node i requires being visited. The events that two distinct nodes i and j require being visited are assumed to be independent.

The probabilistic data of the PTSP 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 require being visited. A PTSP instance is called homogeneous if all probability values in the set P are the same; it is called heterogeneous, if for at least two nodes the values are different.

In Balaprakash et al. (2009a), we presented an experimental study of three high performing estimation-based metaheuristics, namely, iterated local search (ILS-EE), a memetic algorithm (MAGX-EE), and ant colony system (ACS-EE). As a control algorithm, we used an estimation-based random restart local search (RRLS-EE) algorithm. The four algorithms use `2.5-opt-EEais` (Balaprakash et al., 2009b), an effective iterative improvement algorithm for the PTSP, as a subsidiary solution improvement procedure. For a detailed exposition of the estimation-based metaheuristics, we refer the reader to Balaprakash et al. (2009a), a technical report on the estimation-based metaheuristics, which is available at

`http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2009-017r001.pdf`.

In this technical report, we compare our estimation-based algorithms to the progressive approximation method, the aggregation approach, and previously proposed simulated annealing algorithms. All algorithms discussed in this report are implemented in `C` and compiled with `gcc`, version 3.3. The implementation of ACS-EE is based on ACOTSP (Stützle, 2002). Experiments are carried out on AMD Opteron$^{\text{TM}}$244 processors running at 1.75 GHz with 1 MB L2-Cache and 2 GB RAM under Rocks Cluster GNU/Linux.

For the instance generation scheme, and the adopted parameters, please see Balaprakash et al. (2009a). The PTSP instances used for the experiments are obtained by associating a probability value to each node using a beta distribution as described by Bianchi (2006). In this scheme, the probability values of each instance are characterized by two parameters: the mean probability p_m and the percentage of maximum variance p_v: when an instance is generated with p_m and p_v, the expected value and the variance of the random variable $\boldsymbol{\omega}$ parameterized by P are p_m and $(p_v/100)\cdot p_m(1-p_m)$, respectively. For the sake of convenience, we refer to the probability level of an instance as $p = p_m(p_v\%)$.

We present the empirical results of the comparative study as follows: In Sections 2 and 3, we benchmark our estimation-based iterative improvement algorithm called `2.5-opt-EEais` against the progressive approximation method (Tang and Miller-Hooks, 2004) and the aggregation approach (Campbell, 2006). Although these two approaches do not belong to the class of metaheuristics, they are considered to be viable alternatives to tackle the PTSP (Campbell and Thomas, 2008). In Section 4, we present the results of seeding metaheuristics with solutions obtained via the aggregation approach and the TSP optimal solution. In Section 5, we present the comparison between the estimation-based metaheuristics and simulated annealing algorithms.

2 Comparison to the progressive approximation method

First, we focus on the progressive approximation method. This algorithm is an iterative improvement algorithm that uses an approximated version of the exact closed-form equation at each iteration. The algorithm is run in an iterative way, in which the approximation is made more precise with an increase in the number of iterations. This results in very rough cost computations in the initial iterations but as the search progresses, the cost computation becomes more accurate. In order to perform an unbiased comparison, we use four algorithms `2.5-opt-EEais`, `2-opt-EEais`, `2.5-opt(PA)`, `2-opt(PA)`: `2-opt-EEais` is similar to `2.5-opt-EEais` except that it adopts the 2-exchange neighborhood. `2-opt(PA)` and `2.5-opt(PA)` are obtained from `2-opt-EEais` and `2.5-opt-EEais`, by replacing the estimation-based evaluation procedure with the progressive approximation method, respectively. This setting allows us to study the effectiveness of the estimation-based evaluation procedure without any bias to the adopted neighborhood structure. The adoption of `2-opt(PA)` is ascribed to the fact that the progressive approximation method is investigated in an iterative improvement algorithm that adopts the 2-exchange neighborhood (Tang and Miller-Hooks, 2004). We generated heterogeneous uniform instances with 100 nodes as described

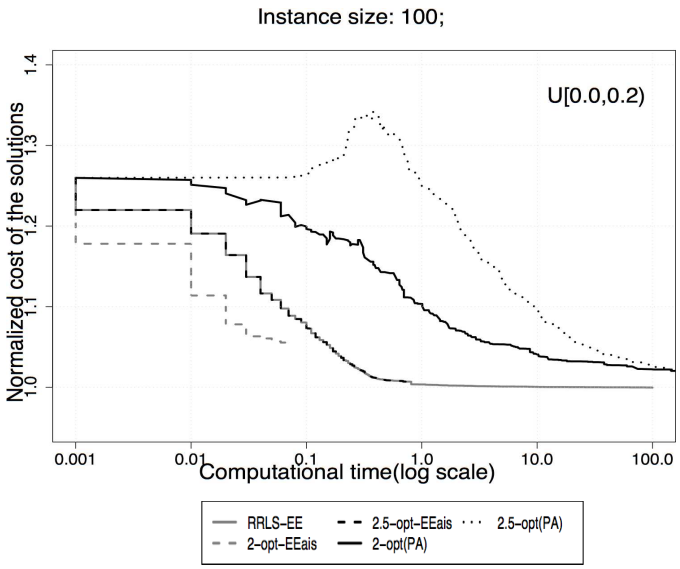

Figure 1: Experimental results on instance size 100. The plots represent the development of the solution cost over time for `2.5-opt-EEais`, `2-opt-EEais`, `2-opt(PA)`, and `2.5-opt(PA)`. The obtained solution costs of the algorithms are normalized by the final solution cost reached by RRLS-EE, which is run for 100 CPU seconds. The normalization is performed on an instance-by-instance basis for 30 instances; the normalized solution cost is then aggregated.

in Tang and Miller-Hooks (2004): The probability p_i of a node i requiring a visit is generated according to a uniform distribution U$[lb, ub)$, with respect to a lower bound lb, with $0 < lb < 1$, and an upper bound ub, with $lb \leq ub \leq 1$. The initial approximation parameter value is set to 2 and it is increased by 2 for each iteration until the value reaches 75% of n, where n is the instance size. This parameter setting differs from the original setting of Tang and Miller-Hooks (2004) because we noted that `2.5-opt(PA)` requires large values to be able to reach high quality local optima.

The computational results are given in Figure 1 and Table 1. From the plot, we can observe that `2.5-opt-EEais` is approximately 1.5 orders of magnitude faster than `2-opt(PA)` and `2.5-opt(PA)` to reach local optima. `2.5-opt-EEais` obtains an average solution cost which is up to 2.56% and 1.31% less than that of `2-opt(PA)` and `2.5-opt(PA)`. Also note that `2.5-opt(PA)` moves to worse solutions during initial iterations. This is ascribed to the fact that the progressive approximation is not effective for node-insertion moves when the approximation parameter takes small values. Although `2-opt-EEais` is faster than all other algorithms in reaching local optima, the obtained solution cost is rather poor.

Table 1: Comparison of the average cost obtained by `2.5-opt-EEais`, `2-opt-EEais`, `2-opt(PA)` and `2.5-opt(PA)` on uniform instances with 100 nodes. For a given comparison A vs. B, the table reports the observed relative difference d between the two algorithms A and B and the 95% confidence interval CI obtained through the t-test. If the value is positive, algorithm A obtained an average cost that is larger than the one obtained by algorithm B. In this case, the value is typeset in italics if it is significantly different from zero according to the t-test at a confidence level of 95%. If the value is negative, algorithm A obtained an average cost that is smaller than the one obtained by algorithm B. In this case, the value is typeset in boldface if it is significantly different from zero according to the t-test, at a confidence level of 95%.

	`2.5-opt-EEais` vs. `2-opt-EEais`		`2.5-opt-EEais` vs. `2-opt(PA)`		`2.5-opt-EEais` vs. `2.5-opt(PA)`	
p	d	CI	d	CI	d	CI
U(0.0,0.2]	$\mathbf{-4.50}$	$[-5.52, -3.47]$	$\mathbf{-1.32}$	$[-2.05, -0.59]$	$\mathbf{-1.31}$	$[-2.03, -0.59]$
U(0.0,0.5]	$\mathbf{-3.86}$	$[-5.22, -2.50]$	$\mathbf{-2.38}$	$[-3.69, -1.08]$	$+0.46$	$[-0.44, +1.35]$
U(0.0,1.0]	$\mathbf{-2.68}$	$[-3.80, -1.56]$	$\mathbf{-2.56}$	$[-3.92, -1.21]$	-1.05	$[-2.68, +0.57]$

3 Comparison to the aggregation approach

Now, we investigate the effectiveness of using the aggregation approach for the PTSP instances with low node probability values. The main idea of this approach is to reduce the size of a given instance by grouping the nodes that are close to each other. We implemented an aggregation-based iterative improvement algorithm that comprises the following steps: the nodes of a given instance are grouped into regions such that the total expected probability in each region is no more than 0.5; then each region is considered as a node; a heuristic is used to obtain an initial solution of the aggregated instance and `2.5-opt-EEais` is applied on this initial solution; the final full solution is obtained by connecting the nodes within each region using a heuristic. We used two algorithms `2.5-opt-EEais(Agg-NN)` and `2.5-opt-EEais(Agg-SF)` for the comparison. While `2.5-opt-EEais(Agg-NN)` adopts the nearest neighbor heuristic to generate an initial solution for `2.5-opt-EEais` and for connecting the nodes within each region, `2.5-opt-EEais(Agg-SF)` adopts the space filling curve heuristic as in Campbell (2006) for the same tasks. We use homogeneous instances generated from the three TSPLIB instances `dsj1000`, `ali535`, and `gr666`. These TSP instances are used by Campbell (2006) to show the effectiveness of the aggregation approach. Since the aggregation approach is proposed for instances with low average probability values, we consider the following values for p_m: $\{0.050, 0.075, 0.100\}$ (Class I) and $\{0.150, 0.175, 0.200\}$ (Class II). Each algorithm is run 10 times on each instance.

Table 2 shows the observed relative difference in the solution cost between the algorithms on the `gr666` instances. `2.5-opt-EEais` obtains an

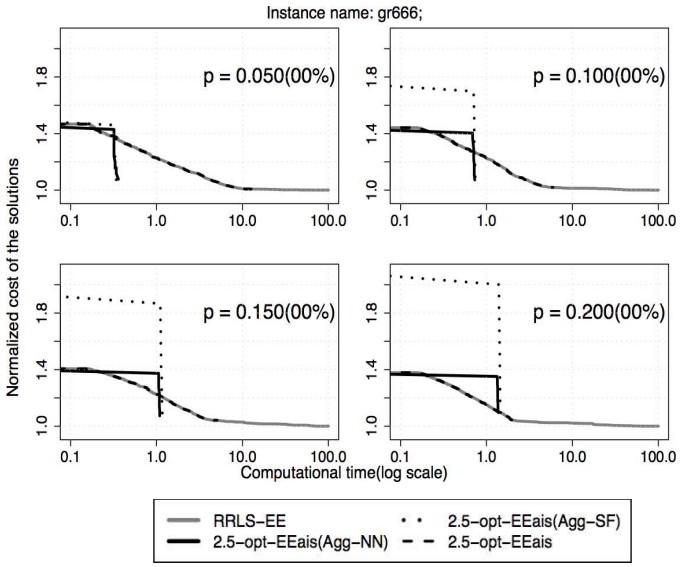

Figure 2: Experimental results on the instance `gr666`. The plots represent the development of the solution cost over time for `2.5-opt-EEais`, `2.5-opt-EEais(Agg-NN)`, and `2.5-opt-EEais(Agg-SF)`. Note that nearest neighbor and space filling curve heuristic solutions are considered as initial reference solutions for `2.5-opt-EEais(Agg-NN)` and `2.5-opt-EEais(Agg-SF)`, respectively. The obtained solution costs of the algorithms are normalized by the final solution cost reached by RRLS-EE, which is run for 100 CPU seconds. The normalization is performed on a run-by-run basis for 10 runs; the normalized solution cost is then aggregated.

average solution cost that is significantly less than that of `2.5-opt-EEais(Agg-NN)` and `2.5-opt-EEais(Agg-SF)` except for $p = 0.175$ (00%). The observed differences are up to 6.4% for `2.5-opt-EEais(Agg-NN)` and 5.11% for `2.5-opt-EEais(Agg-SF)`. However, from the run time development plots given in Figure 2, we can observe that `2.5-opt-EEais(Agg-NN)` and `2.5-opt-EEais(Agg-SF)` reach local optima in short computation times. The observed difference in computation time needed to reach local optimum between the aggregation-based algorithms and `2.5-opt-EEais` ranges from an order of magnitude to a factor of 5. We observed a trend in which the computation time needed to reach a local optimum decreases with an increase in the value of p and in instance size: While on the `ali535` and `gr666` instances, the speed advantage is observed up to $p = 0.175(00\%)$, on the `dsj1000` instances, it is observed up to $p = 0.075(00\%)$.

Table 2: Comparison of the average cost obtained by `2.5-opt-EEais`, `2.5-opt-EEais(Agg-NN)`, and `2.5-opt-EEais(Agg-SF)`, over 10 independent runs on instance `gr666`. Typographic conventions are the same as in Table 1.

	`2.5-opt-EEais` vs. `2.5-opt-EEais(Agg-NN)`		`2.5-opt-EEais` vs. `2.5-opt-EEais(Agg-SF)`	
p	d	[95% CI]	d	[95% CI]
0.050(00%)	$\mathbf{-6.48}$	$[-8.07, -4.89]$	$\mathbf{-4.22}$	$[-4.80, -3.64]$
0.075(00%)	$\mathbf{-5.83}$	$[-8.05, -3.62]$	$\mathbf{-3.89}$	$[-4.97, -2.81]$
0.100(00%)	$\mathbf{-4.94}$	$[-5.77, -4.11]$	$\mathbf{-4.97}$	$[-5.98, -3.95]$
0.150(00%)	$\mathbf{-2.93}$	$[-4.90, -0.95]$	$\mathbf{-3.56}$	$[-5.20, -1.92]$
0.175(00%)	-1.73	$[-3.74, +0.28]$	-1.86	$[-3.99, +0.28]$
0.200(00%)	$\mathbf{-5.45}$	$[-7.26, -3.64]$	$\mathbf{-5.11}$	$[-6.86, -3.36]$

4 TSP approximation and aggregation approach in estimation-based algorithms

In this section, we investigate whether the effectiveness of estimation-based algorithms can be improved by using the aggregation approach or the TSP optimal solution as initial solution under short computation time. For this study, we adopt ILS-EE, which is shown to be quite effective for short computation time (Balaprakash et al., 2009a). We compare ILS-EE that uses the nearest neighbor solution as the initial solution to ILS-EE(Agg-NN), in which `2.5-opt-EEais(Agg-NN)` is used to generate initial solution and ILS-EE(TSP-OPT) in which the TSP optimal solution is used as initial solution. We use homogeneous instances generated from the following TSPLIB instances: `lin318`, `gr666`, and `dsj1000`. The following values are considered for p_m: $\{0.050, 0.075, 0.100\}$ (Class I) and $\{0.150, 0.175, 0.200\}$ (Class II) and $\{0.300, 0.400, 0.500\}$ (Class III). We apply ILS-EE(Agg-NN) only on Class I and Class II instances. Each algorithm is run 10 times on each instance with a stopping criterion of $n/10$ CPU seconds. The adoption of $n/100$ CPU seconds as stopping criterion is not realistic because it is does not allow ILS-EE to finish at least three iterations, especially on the large instances with small p_m. For the three algorithms, we choose the parameter values that obtained the lowest average cost across the 10 tuning runs for 100 CPU seconds. The adopted values for ps, n_{db}, rst_{itr} are 2, 1, 0.084 (Class I), 1, 1, 0.37 (Class II), and 1, 1, 0.31(Class III), respectively. It should be noted that for ILS-EE(TSP-OPT), we did not include the computation time to obtain the TSP optimal solutions. Hence, the results presented here can be considered a best-case analysis to illustrate what would be the performance if we already were aware of the optimal TSP solution. In practice, the time for the TSP solver would have to be added. However, as we show below, even the best-case analysis suggests that starting from a TSP-optimal solution does not yield a significant advantage. On INTEL Xeon E5410 processors running at 2.33GHz with 6MB L2 cache and 8GB RAM under

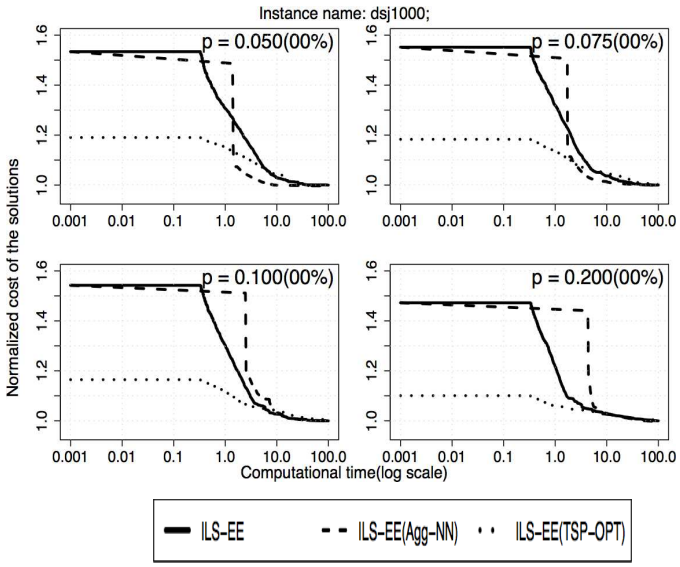

Figure 3: Experimental results on the instance `dsj1000`. The plots represent the development of the solution cost over time for ILS-EE, ILS-EE(Agg-NN), and ILS-EE(TSP-OPT). The obtained solution costs of the algorithms are normalized by the final solution cost reached by ILS-EE. The normalization is performed on a run-by-run basis for 10 runs; the normalized solution cost is then aggregated.

Rocks Cluster GNU/Linux, the Concorde TSP solver version 03.12.19 (using Qsopt as the linear programming solver), took 2.11, 17.15, and 276.86 CPU seconds to find optimum for `lin318`, `gr666`, and `dsj1000`, respectively.

The run time development plots on the `dsj1000` instances are shown in Figure 3. Table 3 reports the average cost differences between the algorithms and the bounds obtained. We could not observe any advantage of using initial solutions from the aggregation approach and using TSP optimal solutions. In spite of starting from TSP-optimal solution, which is much better than a nearest neighbor solution for the PTSP as shown in Figure 3, the average solution cost of ILS-EE(TSP-OPT) is comparable to that of ILS-EE. From the runtime development plot, we can observe that the rate at which the solution cost develops over time is rather slow. From this we can infer that the region around TSP optimal solution has a number of improving neighbor solutions but the improvement in cost is rather small. For instances with very low node probability values, the curves on the development of the solution quality over time in Figure 3 for ILS-EE(Agg-NN) are, for a small computation time interval, slightly below the ones of ILS-EE. This effect is mainly because `2.5-opt-EEais(Agg-NN)`, for these probability levels, reaches local optima faster than `2.5-opt-EEais`. This effect has already been discussed in Section 3.

Table 3: Comparison of the average cost obtained by ILS-EE, ILS-EE(Agg-NN), and ILS-EE(TSP-OPT) over 10 independent runs on instance `lin318`, `gr666`, and `dsj1000`. Typographic conventions are the same as in Table 1.

		ILS-EE vs. ILS-EE(Agg-NN)		ILS-EE vs. ILS-EE(TSP-OPT)	
	p	d	[95% CI]	d	[95% CI]
	$n/10$ CPU seconds				
`lin318`	0.050(00%)	+0.00	[−0.02, +0.02]	−0.01	[−0.03, +0.02]
	0.075(00%)	+0.02	[−0.02, +0.05]	+0.02	[−0.02, +0.05]
	0.100(00%)	+0.05	[−0.05, +0.15]	+0.03	[−0.07, +0.13]
	0.150(00%)	−0.05	[−0.42, +0.33]	−0.28	[−0.70, +0.13]
	0.175(00%)	+0.10	[−0.25, +0.45]	−0.09	[−0.42, +0.24]
	0.200(00%)	+0.36	[−0.03, +0.75]	+0.25	[−0.07, +0.57]
	0.300(00%)	–	–	+0.20	[−0.19, +0.59]
	0.400(00%)	–	–	−0.19	[−0.52, +0.15]
	0.500(00%)	–	–	−0.09	[−0.31, +0.13]
`gr666`	0.050(00%)	+0.12	[−0.10, +0.33]	+0.01	[−0.30, +0.33]
	0.075(00%)	+0.33	[−0.25, +0.91]	+0.08	[−0.52, +0.68]
	0.100(00%)	+0.04	[−0.13, +0.20]	+0.11	[−0.07, +0.28]
	0.150(00%)	+0.28	[−0.15, +0.71]	+0.21	[−0.36, +0.77]
	0.175(00%)	+0.29	[−0.60, +1.17]	+0.06	[−0.61, +0.73]
	0.200(00%)	−0.36	[−1.04, +0.32]	−0.37	[−0.75, +0.01]
	0.300(00%)	–	–	+0.07	[−0.59, +0.73]
	0.400(00%)	–	–	+0.06	[−0.32, +0.44]
	0.500(00%)	–	–	−0.06	[−0.56, +0.43]
`dsj1000`	0.050(00%)	+0.38	[−0.54, +1.30]	+0.33	[−0.54, +1.19]
	0.075(00%)	+0.06	[−0.09, +0.20]	−0.28	[−0.97, +0.40]
	0.100(00%)	+0.10	[−0.32, +0.53]	−0.44	[−1.30, +0.43]
	0.150(00%)	+0.03	[−0.49, +0.54]	−0.34	[−0.68, +0.01]
	0.175(00%)	*+0.53*	[+0.19, +0.88]	+0.27	[−0.08, +0.62]
	0.200(00%)	+0.26	[−0.13, +0.65]	−0.23	[−0.72, +0.26]
	0.300(00%)	–	–	−0.43	[−0.94, +0.08]
	0.400(00%)	–	–	−0.01	[−0.38, +0.35]
	0.500(00%)	–	–	−0.01	[−0.25, +0.23]

5 Comparison with estimation-based simulated annealing algorithms

Finally, we compare our estimation-based metaheuristics ILS-EE, MAGX-EE, and ACS-EE to estimation-based simulated annealing algorithms. We implemented two simulated annealing algorithms built on top of `2.5-opt-EEais`. These two algorithms use the Metropolis acceptance criterion (Metropolis et al., 1953) parameterized by a temperature parameter to decide to move from a current solution to the neighbor solution. In the first algorithm, the temperature is controlled by the sampling error of the cost estimation as in the simulated annealing algorithm of Bowler et al. (2003). We denote this algorithm SA-EE(B). In the second algorithm, the temperature parameter is controlled as in a general purpose stochastic simulated annealing algorithm of Gutjahr and Pflug (1996) and Gutjahr (2004). The initial value of the temperature parameter is set to 1000; each iteration comprises $2n$ neighbor selections and accept/reject decisions; the temperature value is reduced by 5% at each iteration until it reaches 0.0001. Instead of the adaptive sample size procedure, a sample size schedule is used to determine the number of realizations, which is set to 20 times itr, where itr is the iteration number. We denote this algorithm SA-EE(G). Note that SA-EE(G) has not been applied to the PTSP but to tackle a closely related stochastic routing problem. We use SA-EE(G) to study the effectiveness of using the sample size schedule. We allow each algorithm to run for 1000 CPU seconds. For ILS-EE, MAGX-EE, and ACS-EE, we choose the parameter values that produced the lowest average cost across the 10 tuning runs for 1000 CPU seconds in Section 4.2 of Balaprakash et al. (2009a). They are listed in Table 4.

The results on clustered instances with 1000 nodes are given in Table 5. We can see that our estimation-based algorithms completely dominate SA-EE(B) and SA-EE(G). The average cost of the solutions obtained by ILS-EE, MAGX-EE, ACS-EE are up to 7.32% lower than that of SA-EE(B) and SA-EE(G).

Table 4: Fine tuned parameter values for each algorithm in 1000 CPU seconds. This table gives the parameters considered for tuning, the range of each parameter given to Iterative F-Race, and the chosen values for each instance class. For an explanation of the parameters, see Section 3.2 in Balaprakash et al. (2009a).

algorithm	parameters	range	selected value		
			Class-I	Class-II	Class-III
ILS-EE	ps	$[1, 90]$	1	1	1
	n_{db}	$[1, 50]$	1	1	1
	rst_{itr}	$[0, 1]$	0.089	0.34	0.69
MAGX-EE	*pop_size*	[3, 15]	4	3	8
	off_frac	$[0.1, 1.0]$	0.24	0.67	0.16
	ps	$[1, 90]$	9	2	1
	n_{db}	$[1, 50]$	1	1	1
	p_n	$[0.1, 1.0]$	0.33	0.11	0.29
	p_c	$[0.1, 1.0]$	0.92	0.97	0.72
ACS-EE	m	$[3, 15]$	10	8	11
	q_0	$[0.0, 1.0]$	1.0	1.0	0.97
	β	$[0.0, 5.0]$	0.18	0.00	0.83
	ρ	$[0.001, 1.0]$	0.31	0.30	0.61

Table 5: Comparison of the average cost obtained by ILS-EE, MAGX-EE, ACS-EE, SA-EE(B), and SA-EE(G) on clustered instances with 1000 nodes for 1000 CPU seconds. Typographic conventions are the same as in Table 1.

	ILS-EE vs. SA-EE(B)		MAGX-EE vs. SA-EE(B)		ACS-EE vs. SA-EE(B)	
p	d	CI	d	CI	d	CI
0.050(00%)	**−2.73**	[−4.55, −0.90]	**−2.72**	[−4.54, −0.89]	**−2.66**	[−4.48, −0.84]
0.050(83%)	**−0.07**	[−0.09, −0.05]	**−0.07**	[−0.09, −0.05]	**−0.07**	[−0.09, −0.05]
0.075(00%)	**−1.91**	[−2.19, −1.63]	**−1.84**	[−2.09, −1.59]	**−1.77**	[−2.06, −1.48]
0.075(83%)	**−0.14**	[−0.22, −0.06]	**−0.14**	[−0.22, −0.07]	**−0.13**	[−0.21, −0.06]
0.100(00%)	**−2.69**	[−3.75, −1.63]	**−2.62**	[−3.71, −1.54]	**−2.61**	[−3.65, −1.57]
0.100(83%)	**−0.22**	[−0.29, −0.15]	**−0.22**	[−0.29, −0.14]	**−0.21**	[−0.27, −0.14]
0.150(00%)	**−3.61**	[−4.11, −3.11]	**−3.48**	[−3.93, −3.03]	**−3.53**	[−4.06, −2.99]
0.150(83%)	**−1.10**	[−1.47, −0.74]	**−0.83**	[−1.22, −0.45]	**−0.74**	[−1.04, −0.45]
0.175(00%)	**−3.88**	[−4.34, −3.41]	**−3.79**	[−4.21, −3.37]	**−3.62**	[−4.09, −3.14]
0.175(83%)	**−1.53**	[−2.20, −0.85]	**−1.40**	[−2.10, −0.71]	**−1.15**	[−2.10, −0.20]
0.200(00%)	**−4.47**	[−5.03, −3.92]	**−4.29**	[−4.84, −3.73]	**−4.21**	[−4.86, −3.56]
0.200(83%)	**−1.66**	[−1.90, −1.42]	**−1.76**	[−2.14, −1.38]	**−1.55**	[−1.94, −1.17]
0.300(00%)	**−2.98**	[−3.50, −2.46]	**−2.98**	[−3.55, −2.42]	**−2.95**	[−3.54, −2.36]
0.300(83%)	**−3.21**	[−3.88, −2.54]	**−2.89**	[−3.56, −2.22]	**−2.64**	[−3.31, −1.96]
0.400(00%)	**−3.12**	[−3.69, −2.56]	**−2.87**	[−3.47, −2.26]	**−3.05**	[−3.62, −2.49]
0.400(83%)	**−4.39**	[−5.46, −3.31]	**−4.13**	[−5.19, −3.07]	**−4.18**	[−5.35, −3.02]
0.500(00%)	**−4.62**	[−5.34, −3.91]	**−4.40**	[−5.13, −3.68]	**−4.81**	[−5.54, −4.08]
0.500(83%)	**−6.58**	[−7.68, −5.48]	**−6.39**	[−7.62, −5.16]	**−6.44**	[−7.72, −5.16]

	ILS-EE vs. SA-EE(G)		MAGX-EE vs. SA-EE(G)		ACS-EE vs. SA-EE(G)	
p	d	CI	d	CI	d	CI
0.050(00%)	**−1.18**	[−2.27, −0.08]	**−1.16**	[−2.25, −0.07]	**−1.11**	[−2.19, −0.02]
0.050(83%)	**−1.32**	[−2.13, −0.51]	**−1.32**	[−2.13, −0.52]	**−1.32**	[−2.13, −0.51]
0.075(00%)	**−1.44**	[−2.30, −0.57]	**−1.37**	[−2.22, −0.52]	**−1.30**	[−2.18, −0.41]
0.075(83%)	**−3.21**	[−4.99, −1.42]	**−3.21**	[−4.99, −1.42]	**−3.20**	[−4.98, −1.42]
0.100(00%)	**−2.02**	[−2.98, −1.05]	**−1.95**	[−2.93, −0.97]	**−1.93**	[−2.89, −0.97]
0.100(83%)	**−3.40**	[−5.49, −1.32]	**−3.40**	[−5.49, −1.31]	**−3.39**	[−5.47, −1.31]
0.150(00%)	**−3.41**	[−5.23, −1.59]	**−3.27**	[−5.02, −1.53]	**−3.32**	[−5.09, −1.55]
0.150(83%)	**−2.78**	[−4.39, −1.16]	**−2.51**	[−4.17, −0.86]	**−2.42**	[−4.02, −0.83]
0.175(00%)	**−2.89**	[−3.81, −1.98]	**−2.80**	[−3.69, −1.92]	**−2.63**	[−3.58, −1.67]
0.175(83%)	**−2.68**	[−3.76, −1.61]	**−2.56**	[−3.70, −1.42]	**−2.31**	[−3.30, −1.32]
0.200(00%)	**−3.72**	[−5.26, −2.19]	**−3.54**	[−5.06, −2.02]	**−3.46**	[−5.02, −1.90]
0.200(83%)	**−3.03**	[−3.94, −2.12]	**−3.13**	[−4.07, −2.18]	**−2.92**	[−3.98, −1.86]
0.300(00%)	**−4.25**	[−4.92, −3.58]	**−4.25**	[−5.03, −3.48]	**−4.22**	[−5.02, −3.42]
0.300(83%)	**−4.14**	[−5.44, −2.84]	**−3.83**	[−4.88, −2.78]	**−3.58**	[−4.78, −2.37]
0.400(00%)	**−5.77**	[−6.62, −4.92]	**−5.52**	[−6.45, −4.59]	**−5.70**	[−6.57, −4.83]
0.400(83%)	**−4.54**	[−5.52, −3.56]	**−4.28**	[−5.13, −3.44]	**−4.34**	[−5.40, −3.28]
0.500(00%)	**−7.13**	[−9.58, −4.68]	**−6.92**	[−9.38, −4.45]	**−7.32**	[−9.78, −4.86]
0.500(83%)	**−4.66**	[−5.41, −3.90]	**−4.46**	[−5.10, −3.82]	**−4.52**	[−5.28, −3.75]

6 Summary

Our iterative improvement algorithm `2.5-opt-EEais` completely dominates the progressive approximation method. On instances with low node probability values, `2.5-opt-EEais` that adopts the aggregation approach reachs local optima faster but the costs of the obtained local optima are worse than that of `2.5-opt-EEais` that do not use the aggregation approach. We also investigated whether the effectiveness of estimation-based algorithms can be improved by using optimal tours for the TSP or the tours returned by the aggregation approach as initial solutions under short computation time. However, we could not observe a significant improvement in obtained solution cost with the two approaches. Our estimation-based metaheuristics completely outperform the two estimation-based simulated annealing algorithms we have implemented.

Acknowledgments

The authors thank Prof. Ann Melissa Campbell for providing the source code of the aggregation approach. This research has been supported by COMP^{2}SYS, an Early Stage Training project funded by the European Commission within the Marie Curie Actions program (MEST-CT-2004-505079), and by ANTS and META-X, which are ARC projects funded by the French Community of Belgium. The authors acknowledge support from the fund for scientific research F.R.S.-FNRS of the French Community of Belgium.

References

P. Balaprakash, M. Birattari, T. Stützle, and M. Dorigo. Estimation-based metaheuristics for the probabilistic traveling salesman problem. Technical Report TR/IRIDIA/2009-017, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2009a. URL `http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2009-017r001.pdf`.

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. *European Journal of Operational Research*, 199(1):98–110, 2009b.

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.

N. E. Bowler, T. M. A. Fink, and R. C. Ball. Characterization of the probabilistic traveling salesman problem. *Physical Review E*, 68(3):036703–036710, 2003.

A. M. Campbell. Aggregation for the probabilistic traveling salesman problem. *Computers & Operations Research*, 33(9):2703–2724, 2006.

A. M. Campbell and B. W. Thomas. Challenges and advances in a priori routing. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 123–142. SV, 2008.

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.

W. J. Gutjahr and G. C. Pflug. Simulated annealing for noisy cost functions. *Journal of Global Optimization*, 8(1):1–13, 1996.

N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculation by fast computing machines. *Journal of Chemical Physics*, 21:1087—1092, 1953.

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/`.

H. Tang and E. Miller-Hooks. Approximate procedures for probabilistic traveling salesperson problem. *Transportation Research Record: Journal of the Transportation Research Board*, 1882:27–36, 2004.