Revista de Informática Teórica e Aplicada - RITA - ISSN 2175-2745
Vol. 25, Num. 1 (2018) 48-66

RESEARCH ARTICLE

A Multi-objective Version of the Lin-Kernighan Heuristic for the Traveling Salesman Problem

Uma Versão Multiobjetivo da Heurística Lin-Kernighan para o Problema do Caixeiro Viajante

Emerson B. de Carvalho[1], Elizabeth F. G. Goldbarg[2]* and Marco C. Goldbarg[2]

Abstract: The Lin and Kernighan's algorithm (LK) for the single objective Traveling Salesman Problem (TSP) is one of the most efficient heuristics for the symmetric case. Although many algorithms for the TSP were extended to the multi-objective version of the problem (MTSP), the LK was still not fully explored. Works that applied the LK for the MTSP were driven to weighted sum versions of the problem. We investigate the LK from a Pareto dominance perspective. The multi-objective LK was implemented within two local search schemes and applied to 2 to 4-objective instances. The results showed that the proposed algorithmic variants obtained better results than a state-of-the-art algorithm.
Keywords: multi-objective traveling salesman — multi-objective lin and kernighan — local search

Resumo: O algoritmo de Lin e Kernighan (LK) para o Problema do Caixeiro Viajante mono-objetivo (PCV) é uma das heurísticas mais eficientes para o caso simétrico. Embora muitos algoritmos para o PCV tenham sido estendidos para a versão multiobjetivo do problema (PCVM), o LK ainda não foi plenamente explorado. Trabalhos que aplicaram o LK ao PCVM foram direcionados a versões do problema de somas ponderadas. Nós investigamos o LK sob a perspectiva da dominância de Pareto. O LK multiobjetivo foi implementado em dois esquemas de busca local e aplicado a instâncias de 2 a 4 objetivos. Os resultados mostraram que as variantes algorítmicas propostas obtiveram melhores resultados que um algoritmo do estado-da-arte.
Palavras-Chave: caixeiro viajante multiobjetivo — lin e kernighan multiobjetivo — busca local

[1]Programa de Pós-graduacão em Sistemas e Computação, Federal University of Rio Grande do Norte, Brazil
[2]Departamento de Informática e Matemática Aplicada, Federal University of Rio Grande do Norte, Brazil
*__Corresponding author__: beth@dimap.ufrn.br
DOI: http://dx.doi.org/10.22456/2175-2745.76452 • **Received:** 13/09/2017 • **Accepted:** 15/12/2017

1. Introduction

The Lin and Kernighan's algorithm (LK) [1] is one of the most effective heuristics proposed for the single objective Traveling Salesman Problem (TSP). The TSP is a well-known NP-hard combinatorial optimization problem that models a variety of practical applications and has been a testbed for several algorithmic ideas. A book devoted to the TSP was presented by [2] in which several LK implementations were presented. The most effective LK implementation was presented in [3] and another popular implementation is part of an exact algorithm proposed in [4]. Besides, the LK has been used as part of several metaheuristics, some of them investigated in an experimental study presented in [5].

The multi-objective TSP (MTSP) is a generalization of the single objective problem which also models several important real world applications. As its single objective counterpart, many algorithmic approaches were proposed to the MTSP, most of them heuristics. Some of those algorithms are adaptations of effective methods originally designed for the TSP. Although the LK is an effective heuristic for the TSP, it is still not fully explored in the multi-objective context. In [6] [7] [8], a linear aggregation of the objectives was applied to the problem and the LK was the local search method used to find good approximations of supported solutions. As far as we are concerned no multi-objective versions of the LK were investigated to the MTSP.

We present a multi-objective version of the LK algorithm, named *MLK*, and investigate its potential in comparison with other local search methods. As local search algorithms are widely used within metaheuristics, the approach proposed in this study can be used within other heuristic methods and has the potential to improve their results. In this study, the MLK is investigated in the Pareto Local Search (PLS) frameworks proposed in [9] and [10]. As a result, two variants of the MLK were implemented and investigated in a computational experi-

ment that comprised 34 instances with 2, 3 and 4 objectives and sizes ranging from 100 to 1000.

This study is organized into five sections. The MTSP is presented in section 2. The MLK and the variants are introduced in section 3. The computational experiments are reported in section 4. Finally, conclusions and other remarks are presented in section 5.

2. The Multi-objective Traveling Salesman Problem

A multi-objective optimization problem consists in the simultaneous optimization of p objective functions $z_k(\rho)$, $k = 1,\ldots,p$, subject to some constraints on the decision vector $\rho \in P$, where P is a subset of $\Re^n$, the real coordinate space of n dimensions . It is usually assumed that there does not exist any $\rho \in P$, such that all functions z_k attain their minima at ρ [11]. The MTSP is defined as in [12]. Given an undirected edge-weighted complete graph $G = (V,E,W)$, where $V = \{1,...,n\}$ is the set of vertices, $E = \{(i,j)|i,j \in V, i \neq j\}$ is the set of edges and W a set of vectors $(w^1_{ij},...,w^p_{ij})$ each of which assigned to an edge $(i,j) \in E$, $p > 1$, the MTSP consists in finding a minimal spanning cycle of G according to equation 1. The term "minimal" refers to Pareto optimality. The p weights assigned to each edge of G correspond to different cost factors, usually conflicting.

$$z_k(\rho) = \sum_{i=1}^{n-1} w^k_{\rho(i),\rho(i+1)} + w^k_{\rho(n),\rho(1)}, \quad k = 1,...,p \qquad (1)$$

The set of Pareto optimal solutions is determined by dominance relations between the objective vectors. Let $\rho = (\rho_1,...,\rho_n)$ be a solution of the MSTP and $z_k(\rho)$, computed with equation 1, be its objective vector, also called objective point, a solution ρ' is said to weakly dominate ρ, denoted by $\rho' \preceq \rho$, if $z_k(\rho') \leq z_k(\rho)$, $\forall k, k = 1,...,p$. ρ' is said to dominate ρ, denoted by $\rho' \prec \rho$, if $z_k(\rho') \leq z_k(\rho)$, $\forall k, k = 1,...,p$, and $\exists k$ such that $z_k(\rho') < z_k(\rho)$. Solutions ρ and ρ' are said incomparable, denoted by $\rho' || \rho$, when neither $\rho' \prec \rho$ nor $\rho \prec \rho'$. Given the objective space of the MTSP, the set of incomparable solutions is the Pareto optimal set, also called set of efficient solutions.

The number of efficient solutions for the MTSP is expected to be exponential on n. Since the single objective problem belongs to NP-hard [13], the multi-objective version is also in this class. Besides, it implies solution of a possibly exponential number of NP-hard problems to determine the set of efficient solutions [14].

2.1 Related work

Exact methods for the MTSP include dynamic programming [15] and the ε-constraint method with branch-and-cut [16]. The main part of the literature dedicated to the MTSP refers to heuristic approaches such as local search [17, 18, 19, 12, 9, 20], evolutionary algorithms [21, 22, 6, 23, 24], ant colony [25] and estimation distribution algorithms [26].

Local search methods are among the most effective for the MTSP, and most works propose algorithms that use the 2 and 3-opt methods. One of the first local search approaches for this problem was proposed in [20]. It was called Two-Phase Local Search (TPLS). Two TPLS variants, the Double Two-Phase Local Search (D-TPLS) and the Pareto Double Two-Phase Local Search (PD-TPLS), were also introduced in [20]. The TPLS consists of two phases. First, the method creates a solution optimizing one objective. The solution generated in the first phase is the starting point for the second phase in which a sequence of scalarizations of the objective function vector gives rise to several new solutions. The method filters the nondominated solutions and returns them. In the D-TPLS variant, one solution for each objective is generated in the first phase. Different implementations of these algorithms were investigated in an experimental study presented in [27] where the 2 and 3-opt local searches were examined within an iterated local search framework. A 2-opt local search was presented for a special case of the MTSP, called TSP(1,2), in [17]. In this MTSP special case, the weights of the edges are values in the set $\{(1,1),(1,2),(2,1),(2,2)\}$. Lust and Teghem [12] presented a revision of meta-heuristic methods and proposed a two-phase technique. First, their technique finds a good approximation of supported efficient solutions by aggregating the objectives and applying the LK heuristic presented in [4]. In the second phase, the solution constructed in the first phase was submitted to the 2-opt local search proposed in [10]. The convergence of the original PLS algorithm was investigated in [18] where a technique that discretizes the objective space dynamically was also proposed. This technique was used to improve the convergence of the PLS algorithm.

Populational methods were also proposed to the MTSP, some of them included local search techniques. Memetic algorithms were proposed in [24] and [6]. The memetic algorithm proposed in [24] is composed of a Random-Keys Genetic Algorithm (RKGA) [28] and 2-opt local search. In the local search phase, the objectives were aggregated by the weighted Tchebycheff function and the weighted sum [29]. A Pareto memetic algorithm with path-relinking and tabu search based on the LK algorithm was proposed in [6]. In that algorithm, a random weight vector was used to aggregate the objectives. A Pareto memetic algorithm with path-relinking, whose the main idea is to use a vector with random weight at each iteration that is used to aggregate the objectives. A tournament selection is made to choose two solutions, then the Path Relinking, using the 2-opt, is applied and the two best solutions found are used in the recombination, in the end, the local search is applied. The local search is divided into two stages, the first use the 2-opt local search ignoring the common edges that are in both parent and son, and the second is used a tabu search using the Lin Kernighan algorithm. The NSGA-II [30] and the MOEA/D [31] were compared in [23]. In that work, a 2-opt local search was included in the

MOEA/D which was compared with the algorithm without the local search. Multiple crossover and mutation operators with a dynamic selection scheme were investigated in [21] for the MTSP. A framework named multi-objective ant colony optimization based on decomposition (MoACO/D) was proposed in [25]. In that approach, the main problem was decomposed in sub-problems, and the objectives were aggregated with the Tchebycheff function. The ant colony was also divided into sub-colonies, each of which optimized a sub-problem. Finally, the Multi-objective Estimation of Distribution Algorithm Based on Decomposition, MEDA/D, was proposed in [26]. That algorithm joined the MOEA/D with estimation distribution algorithms. The idea was to decompose the problem in sub-problems, each of them associated with a probabilistic method. The work presented in [26] also investigated the degree of conflict between the objectives of the MTSP and proposed a method to generate instances for which the degree of conflict can be measured.

3. Multi-objective Lin Kernighan

The LK heuristic, proposed by Lin and Kernighan [1], is one of the most efficient methods to generate high-quality solutions for the Euclidean TSP. Starting from a Hamiltonian cycle, i.e., a solution for the TSP, the main idea is to iteratively replace edges from the cycle by others that are not in the cycle. A gain function is calculated at each edge replacement. The procedure is repeated while the accumulated gain is positive and the exchange of edges is possible.

Let T be a solution. Every iteration, the LK heuristic searches for two sets of edges: X, edges in the cycle, and Y, edges not in the cycle, $|X| = |Y| = r$, such that, the replacement of X by Y improves the cycle. The r value can vary from iteration to iteration. There are several implementations of the LK heuristic and each of them has its method to choose the edges of X and Y. A revision of those methods was presented by [32]. An edge that belongs to X or Y must satisfy constraints i–iv.

1. *Sequential exchange*: Edges x_i and y_i must share a vertex, as well as edges y_i and x_{i+1}.
2. *Viability*: x_i, x_{i+1}, y_i and y_{i+1} must be chosen so that a cycle results from the inclusion of y_i and y_{i+1} after the removal of x_i and x_{i+1}.
3. *Positive gain*: Let $g_i = w(x_i) - w(y_i)$ be the gain computed for the solution cost after exchanging edges x_i and y_i, then $G_i = g_1 + g_2 + ... + g_i$ is the accumulated gain on i iterations. G_i must be positive.
4. *Disjoint sets*: Sets X and Y must be disjoint, i.e., $X \cap Y = \emptyset$.

A detailed explanation of the LK heuristic was presented by [3].

The multi-objective version of the LK algorithm, named MLK, differs from the original algorithm on the gain function and the output. To evaluate the solutions produced in the MLK, the gain is evaluated for each objective. The algorithm continues executing if the gain is positive for, at least, one objective. The output is a set of solutions mutually incomparable, i.e., a nondominated set of solutions. The MLK pseudocode is divided into two procedures presented in Algorithms 1 and 2.

Algorithm 1 Start(s,α,p)

Require: Cycle s, the limit for the recursion depth α, the number of objectives p
Ensure: A set of mutually incomparable solutions, PS
1: $PS \leftarrow \emptyset$
2: **for** each edge $(u,v) \in s$ **do**
3: $s' \leftarrow s - (u,v)$
4: $G_i \leftarrow 0$ for $i = 1, ..., p$
5: $R \leftarrow \emptyset$
6: $ImprovePath(s', 1, \alpha, p, R, G, PS)$
7: **end for**
8: **return** PS

Algorithm 1 receives a cycle (solution), s, the limit for the recursion depth, α, and the number of objectives, p. The PS set is initialized (step 1). Every iteration of the main loop (steps 2-7), an edge is removed from s resulting in a path s' (step 3). The cumulative gain, G_i, of each objective is initialized (step 4). The set of restricted vertices, R, is initialized (step 5) and s' is submitted to the improvement procedure presented in Algorithm 2 (step 6). Algorithm 1 returns the PS set (step 8).

The improvement procedure has seven input parameters: the Hamiltonian path, s', the recursion depth, d, the limit for the recursion depth, α, the number of objectives, p, the set of restricted vertices, R, the cumulative gain, G and the PS set. The path, s', is improved satisfying i-iv. Recursive calls are executed up to α times. R is a set of vertices which are related to the iv constraint. If edge $y = (u,v)$ is considered for removal from s', and $u \in R$, y cannot be removed since an edge that started at u was withdrawn previously from the original path, and y was inserted.

Three procedures are called in Algorithm 2: *PS.insert*, *invertPath* and *cycle*. The *PS.insert*() function updates the PS set with the solution passed as the second input parameter. If the new solution is not dominated by any solution of PS, it is added to PS. Solutions dominated by the new solution are discarded. *PS.insert*() returns *True* if PS was updated, otherwise it returns *False*. The $invertPath(v,a)$ function, invert the direction of the path from v to a. The procedure amounts to remove edge (u,v) and insert edge (u,a) in s'. The $cycle()$ function adds an edge to s', from the last to the first vertex of s' and returns a cycle.

If the recursion depth is less than α, Algorithm 2, searches for the best edge exchange, i.e., the one that maximizes the gain in any objective. Otherwise, the algorithm looks for a path such that an accumulated gain in some objective is

reached.

All edges exchanges are analyzed. If a new cycle created from s' is not dominated by s and not dominated by any solution of PS, it is added to PS and a recursive call for the *ImprovePath* procedure is made with the new cycle as the input parameter.

Algorithm 2 ImprovePath(s',d,α,p,R, G, PS)

Require: Path, s', recursion depth, d, limit for d, α, number of objectives, p, set of restricted vertices, R, list of cummulative gains, G, set of nondominated solutions, PS
Ensure: Set of nondominated solutions, PS.

if $d < \alpha$ **then**
  for every edge $(u,v) \in s'$ such that $u \notin R$ **do**
    let a be the last vertex in s'
    for $i = 1, ..., p$ **do**
      $g_i \leftarrow w^i_{uv} - w^i_{ua}$; $G'_i \leftarrow G_i + g_i$
      if $G'_i > 0$ **then**
        $paux \leftarrow s'$; $paux.invertPath(v,a)$
        if not $(s'.cycle() \prec paux.cycle())$ **then**
          if $PS.insert(PS, paux.cycle())$ == True **then**
            $ImprovePath(paux, d+1, \alpha, R \cup \{u\}, G')$
          end if
        end if
      end if
    end for
  end for
else
  let a be the last vertex in s'
  Find (u,v) which maximizes $g_i = w(u,v) - w(u,a)$ among all objectives
  if $g_i > 0$ **then**
    $paux \leftarrow s'$; $paux.invertPath(v,a)$;
    $PS.insert(PS, paux.cycle())$
  end if
end if

The MLK was implemented on the *twoppls* framework as proposed by [7]. The *twoppls* is a two-phase algorithm. In the first phase, the algorithm generates an approximation of the supported efficient solutions, by solving weighted sum single-objective problems. In the second phase, the solutions provided by the first phase are submitted to the Pareto local search (PLS).

For the sake of making this paper self-contained, we give a brief introduction to the PLS which is a local search method introduced by [9] to multi-objective problems. It is a method based on Pareto dominance relations and does not require objectives aggregation or numerical parameters [7]. Algorithm 3 illustrates the PLS framework.

In the original proposal of the PLS by [9] the algorithm started from a randomly generated tour, which was added to the *NS* archive. Lust and Teghem [7] proposed to start from a set of nondominated tours. The solutions are labeled as not

Algorithm 3 PLS(NS)

Require: Initial set of nondominated solutions, NS
Ensure: A set of nondominated solutions, NS

for each $s \in NS$ **do**
  $explored(s) \leftarrow false$
end for
while $\exists s \in NS$ such that $explored(s) = false$ **do**
  $s \leftarrow$ random_not_explored_solution_from(NS)
  for each $s' \in Neighbor(s)$ **do**
    if $s' \not\prec s$ **then**
      $explored(s') \leftarrow false$
      update(s',NS)
    end if
  end for
  $explored(s) \leftarrow true$
end while
return NS

explored (step 2). A not explored solution is picked randomly from *NS* (step 4) and its neighborhood is explored (steps 6-11). Once a neighbour solution s', which is nondominated in relation to s, is found (step 7), s' is labeled not explored (step 8) and added to *NS*. After the neighborhood of s is examined, s is flagged as explored and the algorithm continues at step 4. The algorithm stops when all solutions in *NS* are flagged as explored.

The solutions of *NS* were first generated with the Nearest Neighbor heuristics. Then, they were submitted to the LK local search proposed by [4], called LKC, within a dichotomous scheme (DS) as proposed by [33]. Since the DS was proposed for bi-objective problems, an adaptation was necessary to deal with three and four-objective problems. For problems with more than two objectives, DS was executed for all combinations of two objectives. For example, for three-objective instances the DS was executed three times, each of them for a combination of two objectives.

The MLK was implemented in the *Neighbor*( ) procedure. We implemented two MLK versions as proposed by [9] (PLSP) and by [10] (PLSA). The two PLS variants are called *di_lk_p* and *di_lk_a*, respectively.

4. Computational Experiments

The *di_lk_p* and the *di_lk_a* were compared to the best version of the Two-Phase Pareto Local Search proposed by [7], called *twoppls*, which applied 2-opt local search within the PLSA in the second phase. The three algorithms started from the same set of nondominated solutions as explained in Section 3.

The algorithms were implemented in the C++11 programming language, and the experiments were executed on a PC with an Intel Core i5-2400 3.10GHz x 4 processor and 4 GB of RAM, which ran Ubuntu 13.10 64 bits. Each algorithm was executed 30 times independently for each instance. We present the assessment methodology in section 4.1. The results for 2-objective instances are presented in section 4.2

where, besides comparison among the results of the proposed approaches, a comparison with state-of-the-art algorithms is presented. Finally, results for 3 and 4-objective instances are presented in section 4.3.

4.1 Methodology

The data set for the experiments consisted of 35 instances with 2, 3 and 4 objectives, with n from 100 to 1000. The 2-objective instances used in the experiments are available at ⟨https://eden.dei.uc.pt/~paquete/tsp/⟩ and ⟨https://sites.google.com/site/thibautlust/research/multiobjective-tsp⟩.

The 2-objective instances were used in the experiments reported in [34], [12], [7], [27] and [35]. The 3 and 4-objective instances were created combining mono-objective TSP instances available at the TSPLIB library [36]. Their creation followed the same methodology used to create the 2-objective instances. For example, the 3-objective instance named kroABC100 was created from the 3 mono-objective instances named kroA100, kroB100 and kroC100.

Table 1 shows the instances of the computational experiments. The columns show the instance's name, the number of objectives and the number of cities.

Two quality indicators were used to assess the approximation sets generated by each algorithm: the hypervolume (H) [37] and the epsilon additive ($\varepsilon+$) [38]. The hypervolume indicator is the only single set quality measure that is known to be strictly monotonic about Pareto dominance [39]. Also, to evaluate algorithms concerning different quality indicators is a sound approach since different decision-maker preferences can be considered [40]. However, we have to provide the considered indicators are Pareto compliant. This is the case of the hypervolume and the epsilon indicators as shown by [38].

A reference point and a reference set are necessary to calculate the hypervolume and the epsilon additive indicator, respectively. For 2-objective instances with 100 vertices, the reference set was the Pareto front obtained by the exact method proposed in [16] and published in ⟨https://sites.google.com/site/kflorios/motsp⟩. The reference point was the *nadir* point obtained from the Pareto front. For the 3 and 4-objective instances and the 2-objective instances with more than 100 cities, the reference set was generated with the nondominated points obtained from the union of the approximation sets generated by all algorithms. The reference point consisted of the worst value of each objective of the points in the reference set.

The Kruskal-Wallis statistical test at a significance level of 0.05 was applied to check differences in the results. If a significant difference was observed, a Kruskal-Wallis test for multiple comparisons was applied to verify which variants differ from each other. The one with the best median was considered the best approach.

As explained in Section3, the PLS algorithm stops when all solutions in *NS* are labeled as explored. This was the stopping criterion used for 2-objective instances up to 200 nodes. Due to the limitation of the computational resources, we opted to establish a processing time limit for the remaining instances. The limit was 3000s. The α value was empirically set to 2.

4.2 Results for 2-objective instances

Table 2 shows the results for 2-objective instances. The first column shows the names of the instances. There are three columns related to each algorithm. They show the median results for the H and the $\varepsilon+$ indicators, and the average processing time in seconds.

The statistical test pointed out significant differences between each algorithmic version proposed in this paper and the *twoppls* for the H and the $\varepsilon+$ results concerning the 2-objective instances with 100 vertices. These differences are illustrated in Figures 1-2 which show the boxplots of the H indicator and Figures 5-6 which show the boxplots of the $\varepsilon+$ indicator for the 2-objective instances with 100 vertices. The number on top of the box of Figures 1-4 is the value that must be added to each number in the y-axis to get the real H value. The closer to 1 the value of the H indicator, the better is the result obtained by the algorithms. The smaller the $\varepsilon+$ indicator, the better.

The first part of Table 2 shows that the MLK variants proposed in this study presented better results than the *twoppls* for 2-objective instances up to 100 nodes. The statistical tests indicated the better performance of the *di_lk_p* and the *di_lk_a* than the *twoppls* for those instances concerning the H indicator. An exception is the randEF100 instance for which the algorithms were statistically equivalent. Figures 1 and 2 show that the differences between the boxplots of the randAB100, randCD100 and randEF100 instances were not so remarkable as for the remaining 2-objective instances with 100 vertices. However, the statistical tests pointed out significant differences for these instances too. In general, the *di_lk_p* obtained the best results of the $\varepsilon+$ indicator for 2-objective instances up to 100 nodes. An exception is the clusAB100 instances for which the *di_lk_a* presented the best result. The statistical tests indicated significant differences for all pairwise comparisons regarding the $\varepsilon+$ indicator.

The second part of Table 2 shows the 2-objective instances with 150-1000 vertices. These results for the H and the $\varepsilon+$ indicators are illustrated in Figures 4 and 8, respectively. The statistical tests indicated the better performance of the *di_lk_p* and the *di_lk_a* than the *twoppls* for the kroAB150 and kroAB200 instances regarding the H indicator. For the remaining instances shown in the second part of Table 2, the algorithms finished their execution due to the processing time limit. Significant differences between the methods on the H indicator were not indicated by the statistical tests for the kroAB500 instance. The *di_lk_a* and the *twoppls* were statistically better than the *di_lk_p* on the other instances for which significant differences were not pointed out by the statistical tests for the *di_lk_a* and the *twoppls*. Except for the kroAB150 instance, the *di_lk_a* presented the best values of the $\varepsilon+$ indicator. The statistical test indicated significant differences in the results of the three algorithms tested concerning the $\varepsilon+$ indicator.

Table 1. Instances

Instance	#objective	#cities	Instance	#objective	#cities
kroAB100	2	100	kroABD100	3	100
kroAC100	2	100	kroACD100	3	100
kroAD100	2	100	kroBCD100	3	100
kroBC100	2	100	euclABC100	3	100
kroBD100	2	100	euclDEF100	3	100
kroCD100	2	100	randABC100	3	100
clusAB100	2	100	randDEF100	3	100
euclAB100	2	100	kroABCD100	4	100
euclCD100	2	100	euclABCD100	4	100
euclEF100	2	100	randABCD100	4	100
randAB100	2	100	kroAB150	2	150
randCD100	2	100	kroAB200	2	200
randEF100	2	100	kroAB300	2	300
mixdGG100	2	100	kroAB400	2	400
mixdHH100	2	100	kroAB500	2	500
mixdII100	2	100	kroAB750	2	750
kroABC100	3	100	kroAB1000	2	1000

Table 2. 2-objective instances

	di_lk_a			di_lk_p			twoppls		
	H	$\varepsilon+$	Time (s)	H	$\varepsilon+$	Time (s)	H	$\varepsilon+$	Time (s)
kroAB100	**0.69271**	206.967	74.722	**0.69271**	**182.700**	123.670	0.69263	246.133	28.768
kroAC100	**0.69711**	160.267	52.512	**0.69711**	**146.000**	98.632	0.69706	203.900	20.633
kroAD100	**0.66269**	163.300	47.292	**0.66269**	**161.067**	95.311	0.66263	238.133	16.974
kroBC100	**0.67246**	113.033	58.869	**0.67246**	**110.567**	109.723	0.67241	152.000	21.207
kroBD100	**0.65971**	170.367	54.922	**0.65971**	**158.433**	108.467	0.65966	198.533	19.868
kroCD100	**0.67762**	185.433	42.359	**0.67762**	**176.100**	71.148	0.67757	256.800	16.410
euclAB100	**0.62394**	199.533	32.330	**0.62394**	**178.467**	53.856	0.62385	266.267	11.471
euclCD100	**0.62691**	234.267	45.631	**0.62691**	**222.267**	86.319	0.62683	281.400	17.484
euclEF100	**0.61571**	226.300	51.082	0.61570	**205.000**	96.354	0.61563	247.567	18.160
randAB100	**0.84881**	629.600	10.583	0.84879	**608.167**	12.822	0.84876	704.633	2.187
randCD100	**0.84969**	578.033	10.232	**0.84969**	**570.400**	11.645	0.84965	599.500	2.166
randEF100	**0.85514**	554.200	11.465	**0.85514**	**512.600**	13.414	0.85507	595.367	2.471
mixdGG100	0.72392	405.467	20.779	**0.72394**	**384.967**	31.972	0.72382	440.400	5.753
mixdHH100	0.73265	341.967	25.812	**0.73266**	**323.433**	38.083	0.73258	402.433	6.663
mixdII100	0.73711	480.633	22.003	**0.73712**	**475.867**	34.531	0.73702	534.100	5.857
clusAB100	**0.70700**	**152.833**	84.718	**0.70700**	174.300	135.144	0.70693	272.033	30.326
kroAB150	**0.72665**	137.000	195.659	**0.72665**	**131.500**	501.826	0.72662	185.000	129.501
kroAB200	**0.75794**	**134.500**	709.272	**0.75794**	**134.500**	2207.165	0.75791	191.000	566.113
kroAB300	**0.78864**	**246.000**	3000.000	0.78773	2562.000	3000.000	0.78863	305.000	3000.000
kroAB400	**0.80654**	**637.000**	3000.000	0.80569	2908.500	3000.000	0.80652	893.000	3000.000
kroAB500	**0.82396**	**1082.000**	3000.000	0.82316	3277.000	3000.000	**0.82396**	1254.000	3000.000
kroAB750	**0.85330**	**2529.000**	3000.000	0.85275	4094.500	3000.000	0.85326	2931.000	3000.000
kroAB1000	**0.86769**	**3186.000**	3000.000	0.86725	4784.500	3000.000	0.86764	3612.500	3000.000

Table 2 shows that the *twoppls* presented the lowest average processing times for 2-objective instances up to 200 nodes, i.e., instances for which the algorithms did not stop due to processing time limit. Runtime results are illustrated in Figures 9-3. The stop criterion for those instances was to have explored all solutions in *NS*. The proposed algorithms were able to include more solutions in *NS* than the *twoppls*. Thus, their processing times were also higher than the latter algorithm.

4.3 Results for 3 and 4-objective instances

The results for the 3 and 4-objective instances are shown in Table 3 and Figures 13-12. For the 3 and 4-objective instances, the algorithms stopped due to the processing time limit, so we omitted the processing times in Table 3.

The *di_lk_p* variant presented the best results of the H and the $\varepsilon+$ indicators for the whole set of 3 and 4-objective instances. Figures 13-12 show that the *di_lk_p* variant behaviour was significantly different from the other two methods. The statistical test confirmed this fact. The *di_lk_a* variant pre-

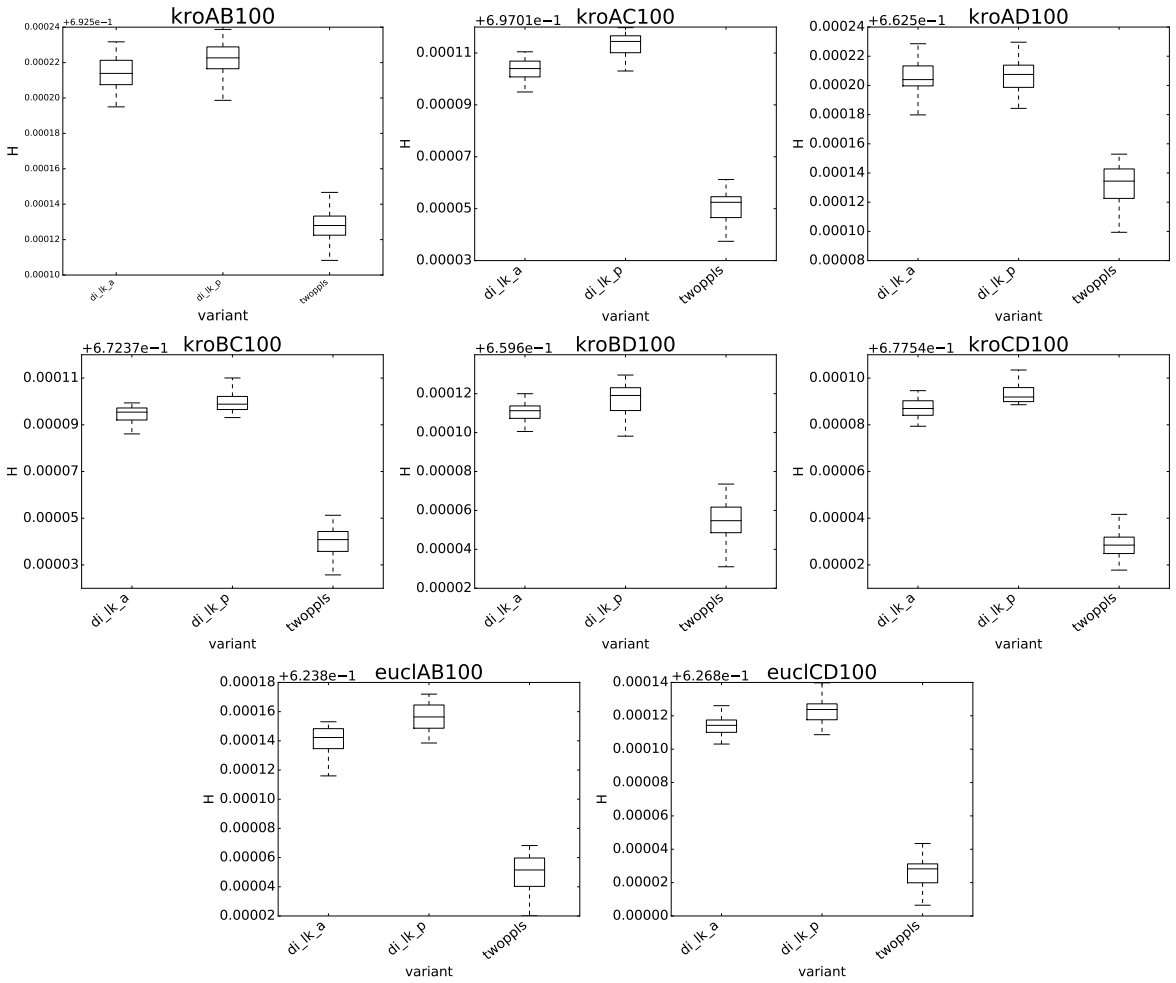

Figure 1. Boxplot for the H indicator: 2-objective instances with 100 cities (part 1)

Table 3. 3 and 4-objective instances

	di_lk_a		di_lk_p		twoppls	
	H	$\varepsilon+$	H	$\varepsilon+$	H	$\varepsilon+$
kroABC100	0.33891	75872.000	**0.47907**	**12657.000**	0.33373	75578.000
kroABD100	0.33314	66939.000	**0.45554**	**13982.000**	0.32614	68208.000
kroACD100	0.32378	72832.500	**0.46313**	**17258.500**	0.31310	75125.000
kroBCD100	0.28959	74859.000	**0.44978**	**14577.000**	0.28940	72450.000
euclABC100	0.27847	65970.000	**0.40272**	**9783.500**	0.27434	66283.500
euclDEF100	0.27006	65125.000	**0.38930**	**13938.000**	0.26662	65468.000
randABC100	0.52029	108014.000	**0.68768**	**14841.000**	0.48704	115526.000
randDEF100	0.52879	103908.000	**0.69054**	**14560.500**	0.50563	108298.500
kroABCD100	0.09401	72409.000	**0.18964**	**19559.000**	0.09966	70882.000
euclABCD100	0.07112	59540.000	**0.14506**	**13314.000**	0.07245	62002.000
randABCD100	0.16953	112977.000	**0.32910**	**29491.000**	0.15236	117227.000

sented better results than the *twoppls* for 9 and 8 instances from the 11 instances tested. The statistical test pointed out significant differences for each pairwise comparison.

5. Conclusions

This study presented a new adaptation of the Lin and Kernighan heuristic, called MLK, for the multi-objective TSP, and its implementation within the PLS framework. We presented two MLK variants which were compared to an algorithm from the literature. The algorithms were applied to 34 instances from 2

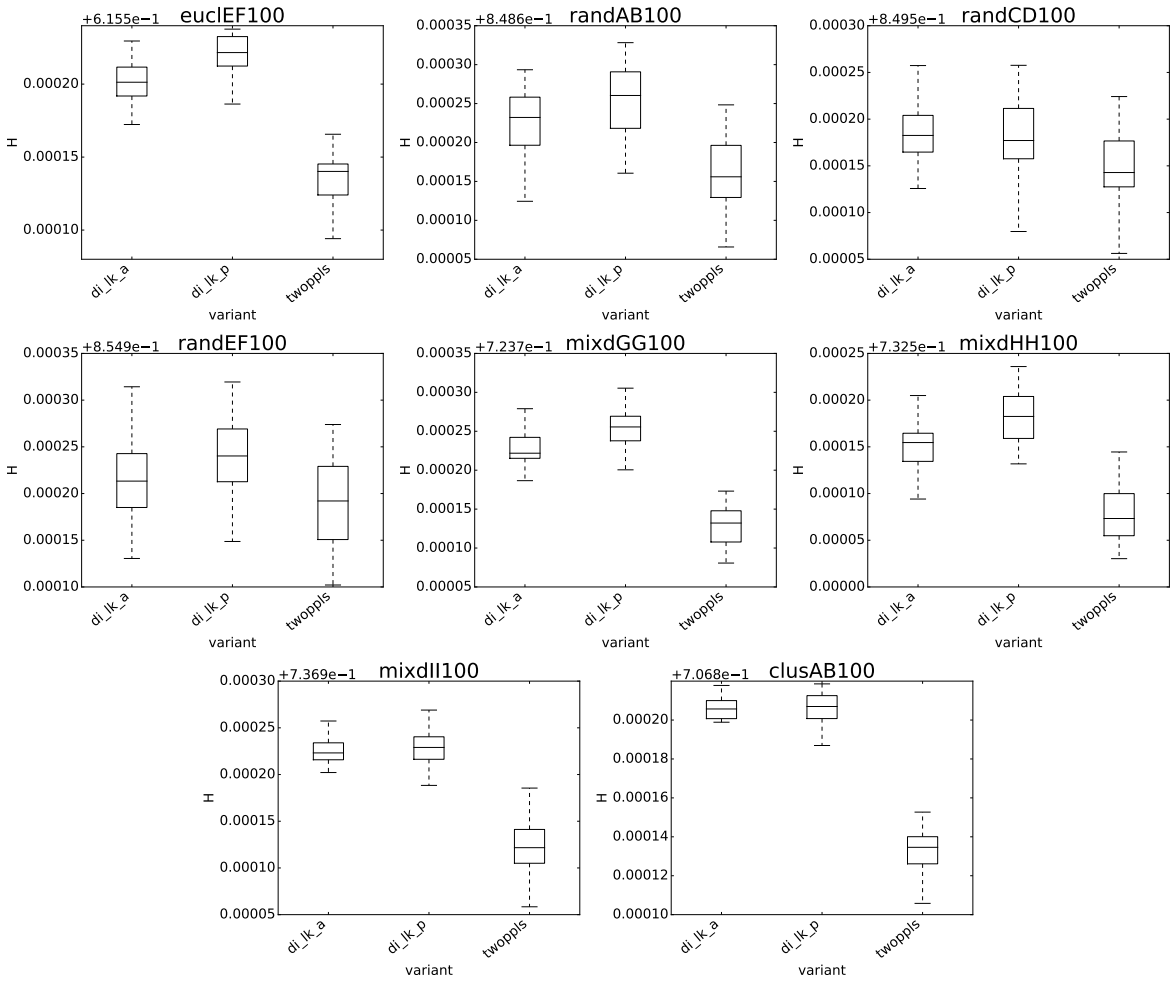

Figure 2. Boxplot for the H indicator: 2-objective instances with 100 cities (part 2)

to 4 objectives. The results showed that, in general, the MLK variants obtained better results than the algorithm from the literature.

In future works, we will investigate the MLK heuristic within metaheuristic algorithms such as evolutionary approaches.

Acknowledgements

The research reported in this paper was partially supported by CAPES, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, and CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico, Brazil, under grants 308062/2014-0 and 302387/2016-1.

Authors Contributions

The authors contributed equally to this research.

References

[1] LIN, S.; KERNIGHAN, B. W. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, v. 21, n. 2, p. 498–516, 1973.

[2] GUTIN, G.; PUNNEN, A. P. *The Traveling Salesman Problem and Its Variations*. 1. ed. Berlin, Germany: Springer Science & Business Media, 2007. v. 12. (Combinatorial Optimization, v. 12).

[3] HELSGAUN, K. An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, v. 126, n. 1, p. 106 – 130, 2000.

[4] APPLEGATE, D.; COOK, W.; ROHE, A. Chained lin-kernighan for large traveling salesman problems. *INFORMS Journal on Computing*, v. 15, n. 1, p. 82–92, 2003.

[5] MARINAKIS, Y.; MARINAKI, M.; DOUNIAS, G. Honey bees mating optimization algorithm for the euclidean traveling salesman problem. *Information Sciences*, v. 181, n. 20, p. 4684–4698, 2011.

[6] JASZKIEWICZ, A.; ZIELNIEWICZ, P. Pareto memetic algorithm with path relinking for bi-objective traveling salesperson problem. *European Journal of Operational Research*, v. 193, n. 3, p. 885–890, 2009.

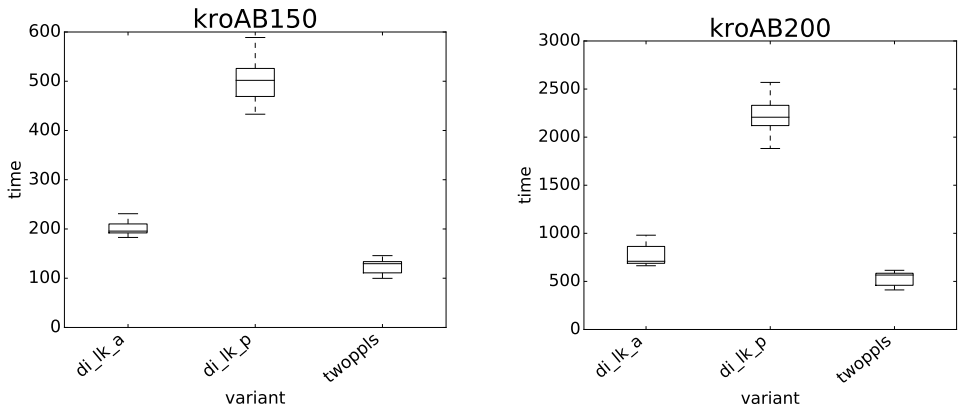

Figure 3. Boxplot for the processing times: kroAB150 and kroAB200

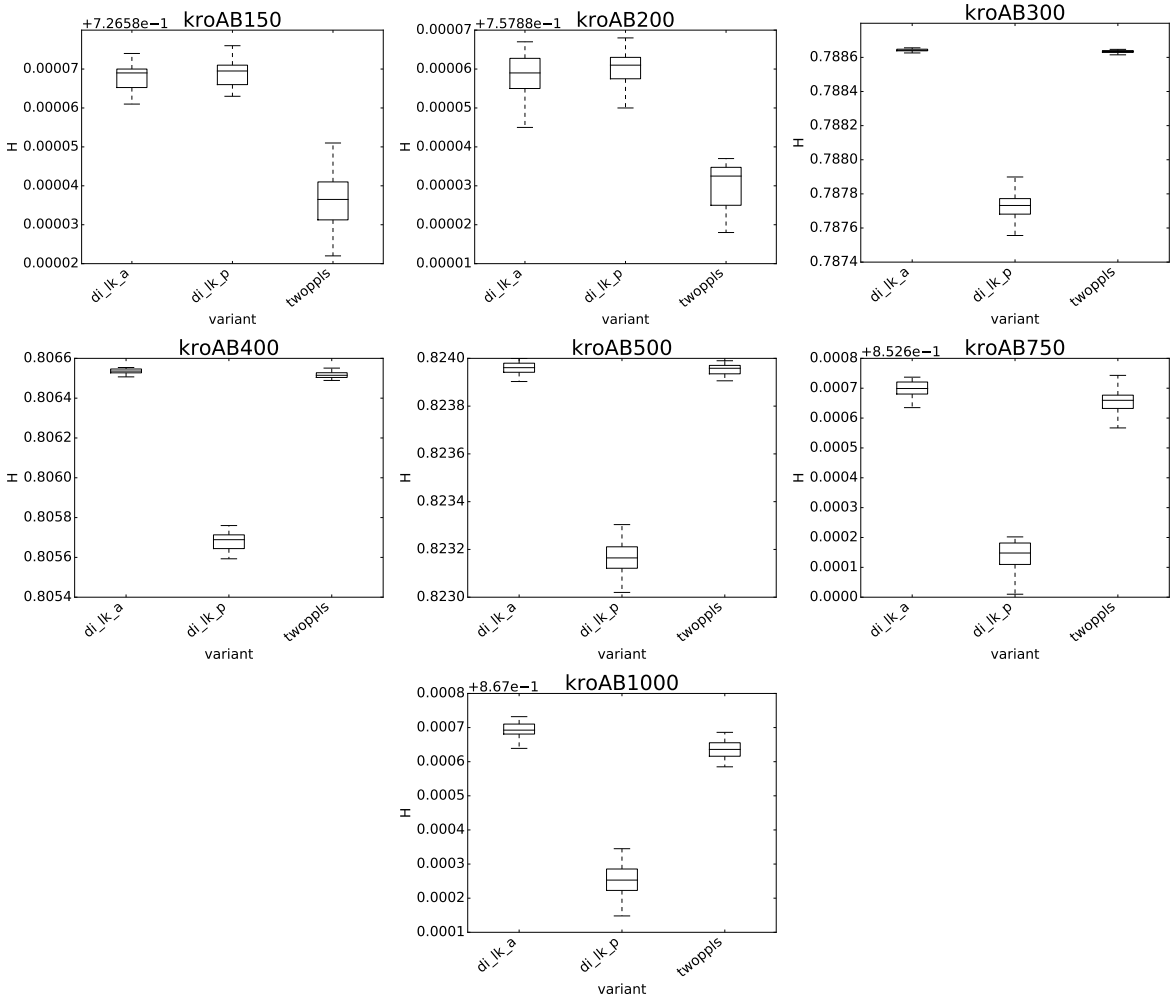

Figure 4. Boxplot for the H indicator: 2-objective instances with 150 to 1000 cities

[7] LUST, T.; TEGHEM, J. Two-phase pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics*, v. 16, n. 3, p. 475–510, 2010.

[8] LUST, T.; JASZKIEWICZ, A. Speed-up techniques for solving large-scale biobjective tsp. *Computers & Operations Research*, v. 37, n. 3, p. 521–533, 2010.

[9] PAQUETE, L.; CHIARANDINI, M.; STÜTZLE, T. Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In: GANDIBLEUX, X. et al. (Ed.). *Metaheuristics for Multiobjective Optimisation*. 1. ed. Berlin, Germany: Springer, 2004, (Lecture Notes in Economics and

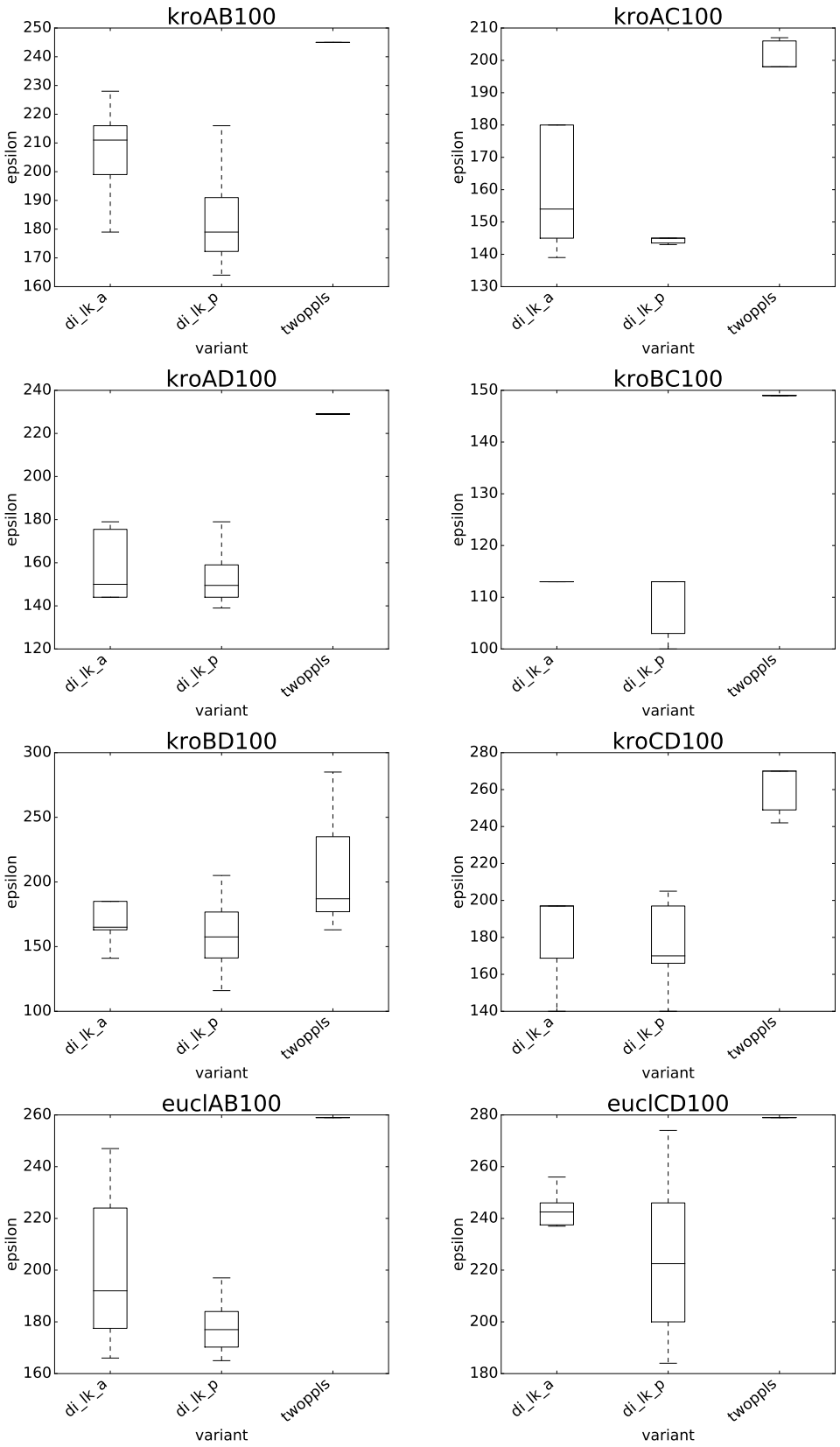

Figure 5. Boxplot for the $\varepsilon+$ indicator: 2-objective instances with 100 cities (part 1)

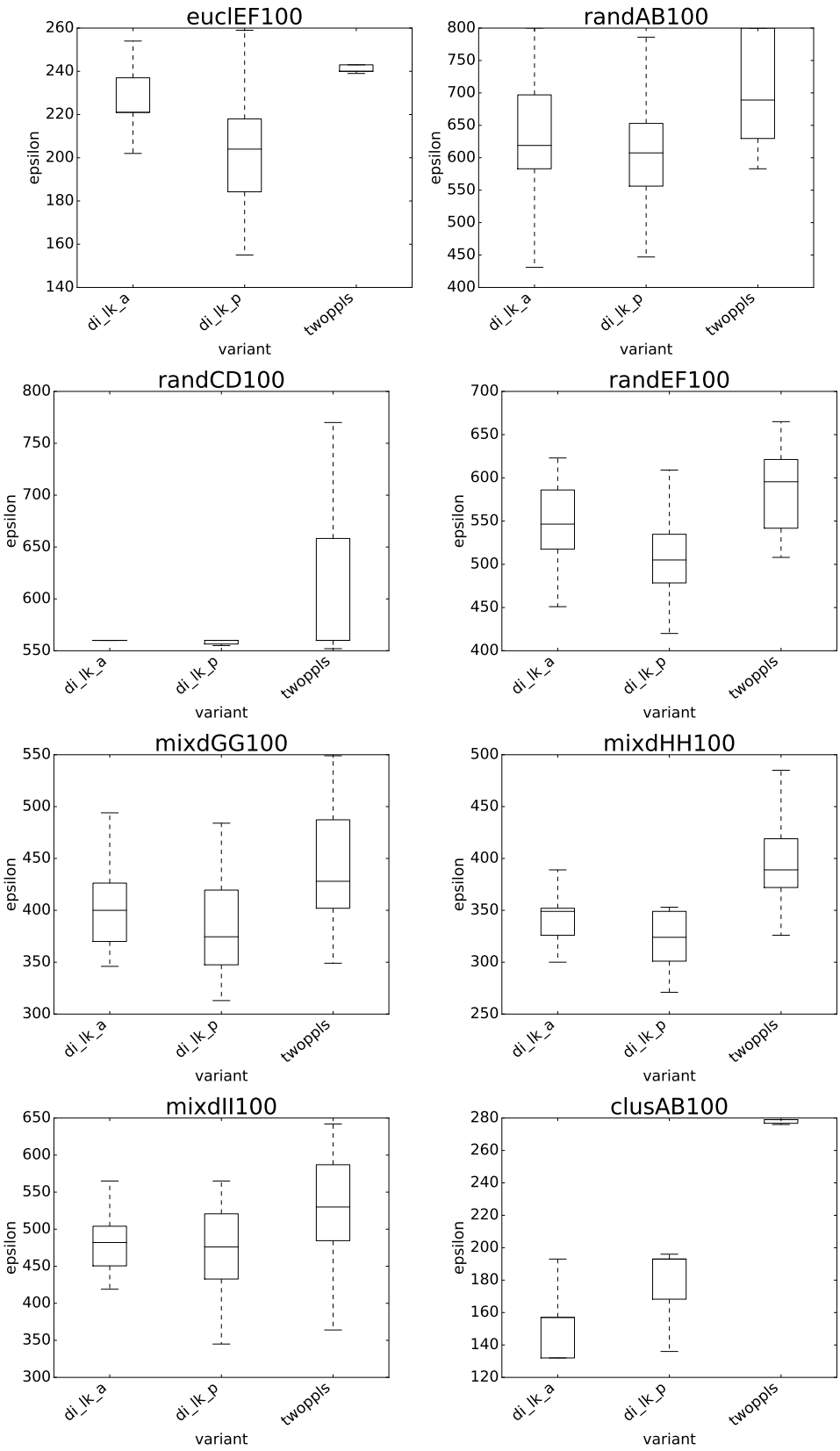

Figure 6. Boxplot for the $\varepsilon+$ indicator: 2-objective instances with 100 cities (part 2)

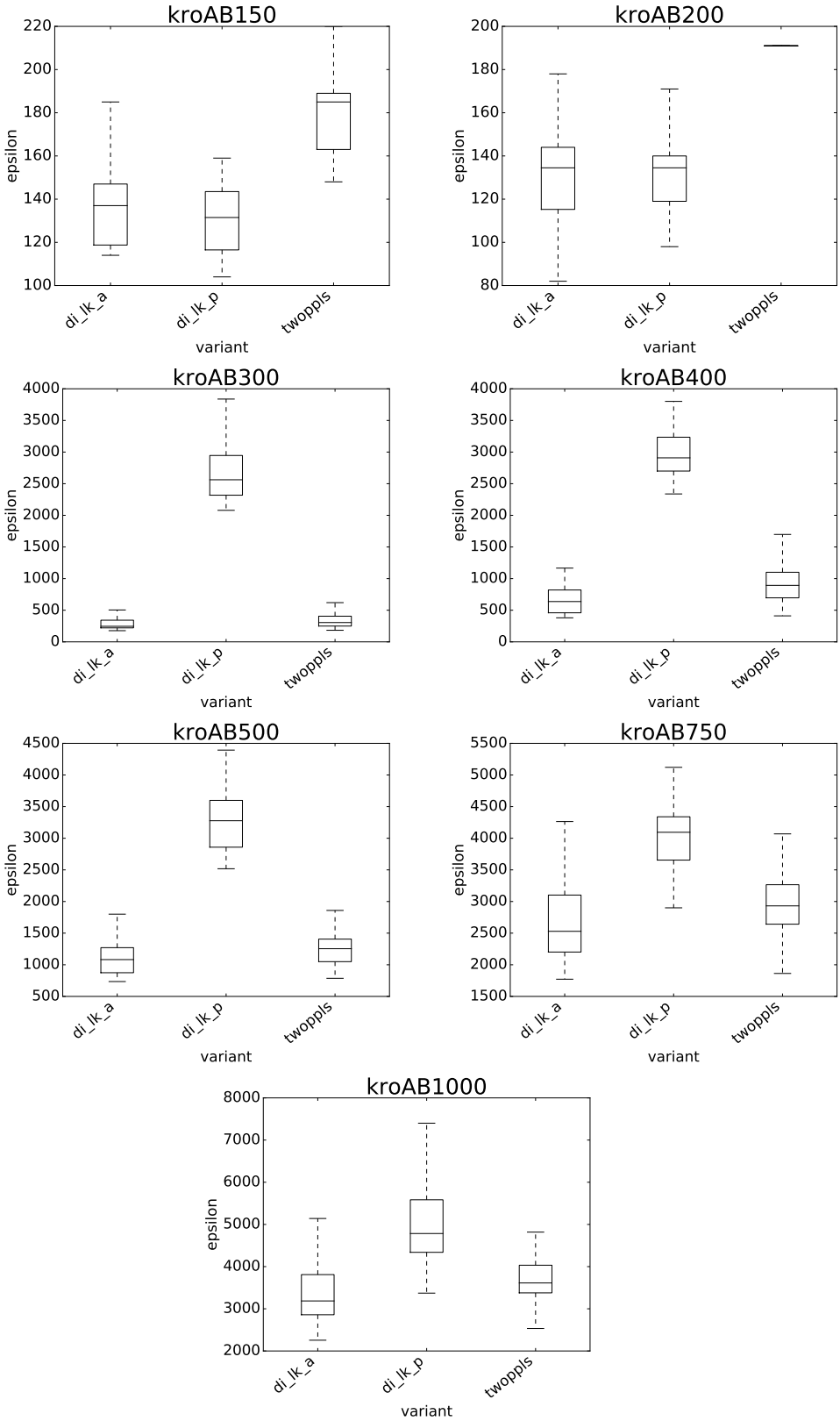

Figure 7. Boxplot for the $\varepsilon+$ indicator: 2-objective instances with 150 to 1000 cities

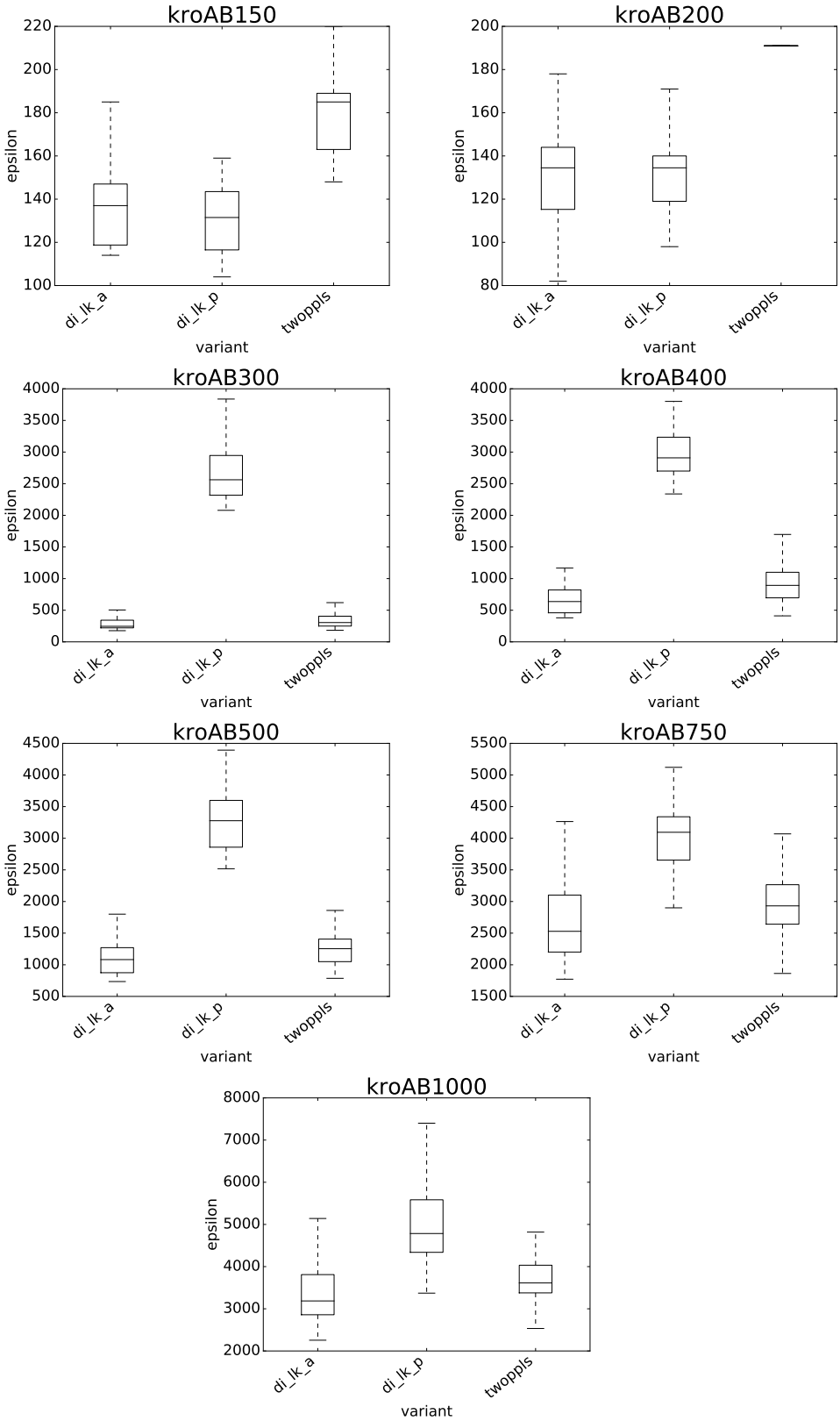

Figure 8. Boxplot for the $\varepsilon+$ indicator: 2-objective instances with 150 to 1000 cities

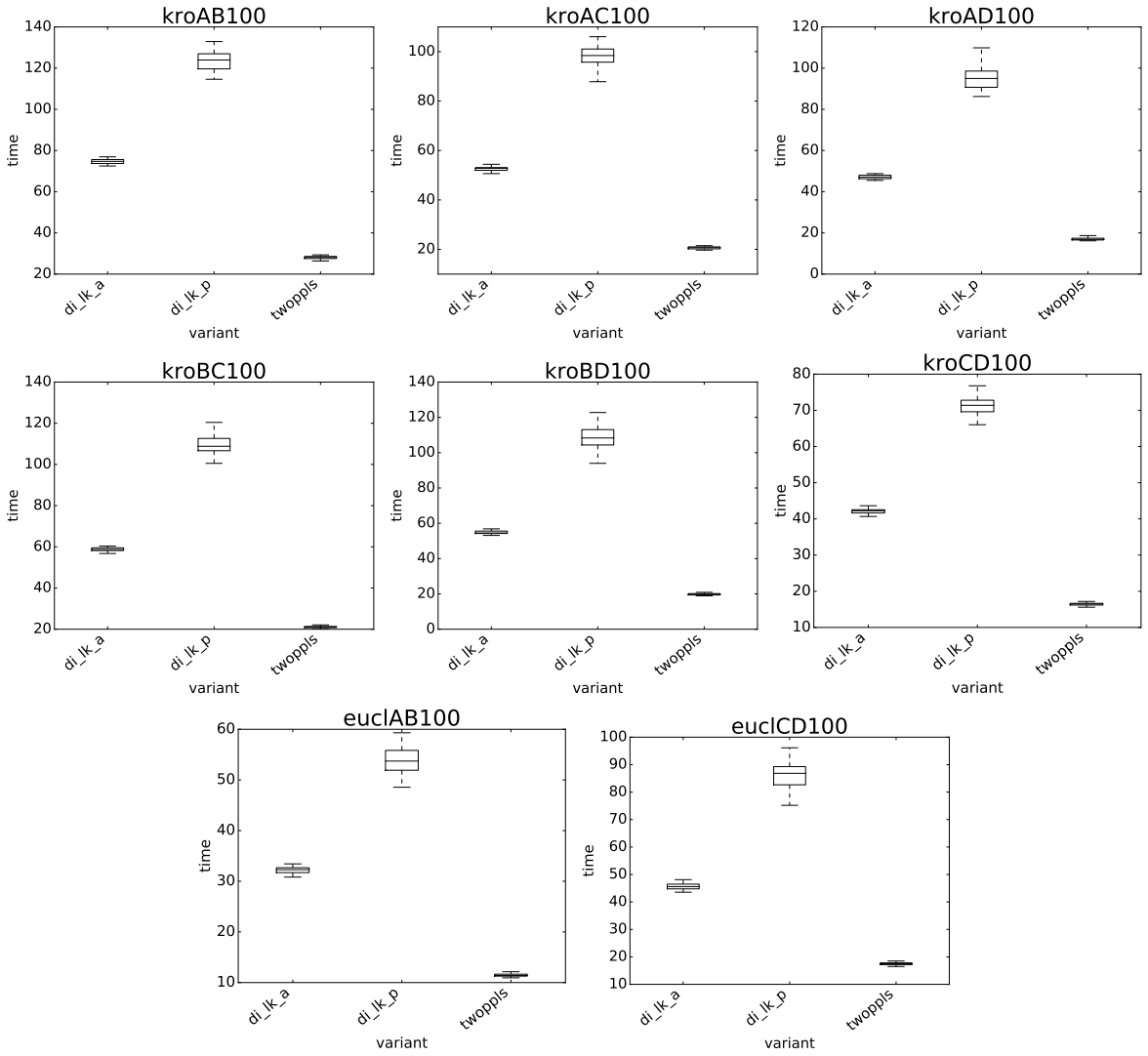

Figure 9. Boxplot for the processing times: 2-objective instances with 100 cities (part 1)

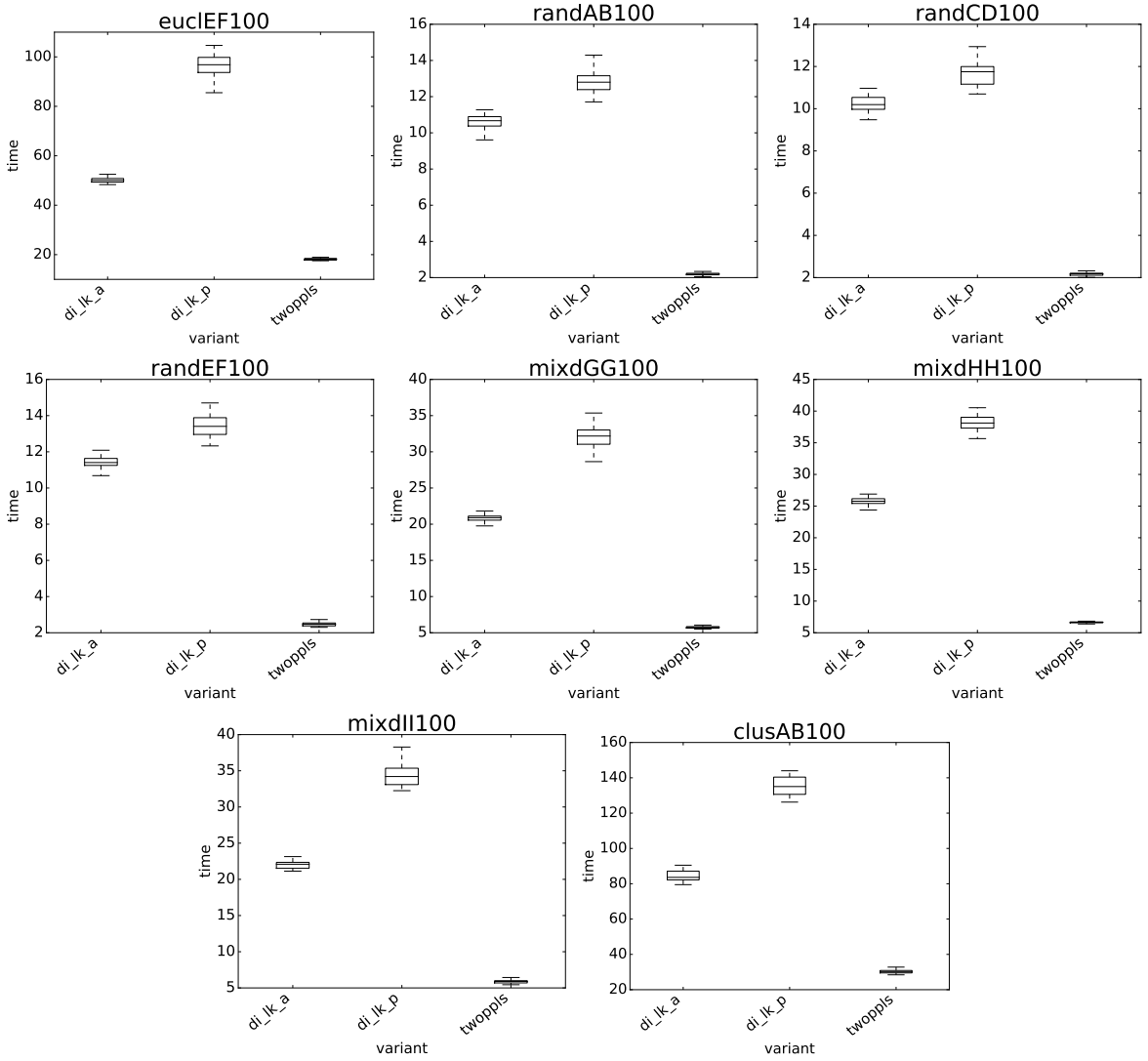

Figure 10. Boxplot for the processing times: 2-objective instances with 100 cities (part 2)

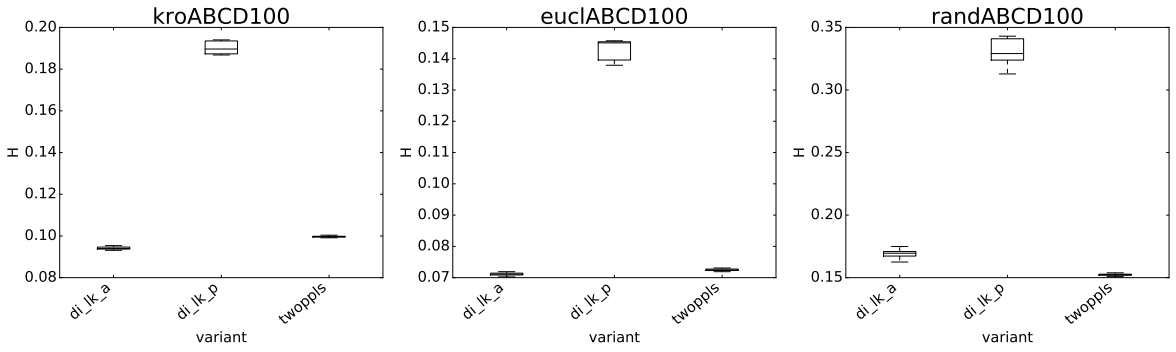

Figure 11. H indicator boxplots for 4-objective instances

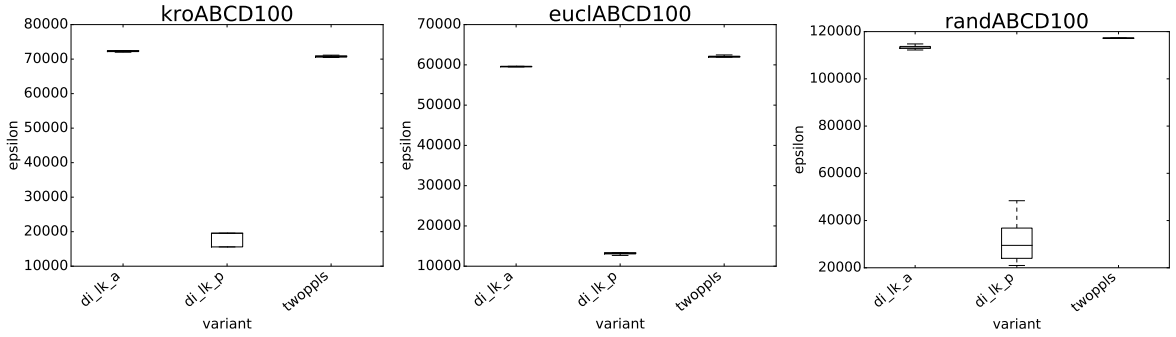

Figure 12. $\varepsilon+$ indicator boxplots for 4-objective instances

Mathematical Systems, v. 535). cap. 7, p. 177–199.

[10] ANGEL, E.; BAMPIS, E.; GOURVÉS, L. A dynasearch neighborhood for the bicriteria traveling salesman problem. In: GANDIBLEUX, X. et al. (Ed.). *Metaheuristics for Multiobjective Optimisation*. Berlin, Germany: Springer, 2004, (Lecture Notes in Economics and Mathematical Systems, v. 535). cap. 5, p. 153–176.

[11] EHRGOTT, M. Vilfredo pareto and multi-objective optimization. *Documenta Mathematica*, extra volume ISMP, n. 1, p. 447–453, 2012.

[12] LUST, T.; TEGHEM, J. The multiobjective traveling salesman problem: a survey and a new approach. In: COELLO, C. A. C.; DHAENENS, C.; JOURDAN, L. (Ed.). *Advances in Multi-objective Nature Inspired Computing*. 1. ed. Berlin, Germany: Springer, 2010, (Advances in Multi-Objective Nature Inspired Computing, v. 272). cap. 6, p. 119–141.

[13] KARP, R. M. Reducibility among combinatorial problems. In: MILLER, R. E.; THATCHER, J. W. (Ed.). *Complexity of Computer Computations*. 1. ed. New York, USA: Plenum Press, 1972, (The IBM Research Symposia Series, v. 1). cap. 8, p. 85–103.

[14] EHRGOTT, M. Approximation algorithms for combinatorial multicriteria optimization problems. *International Transactions in Operational Research*, v. 7, n. 1, p. 5–31, 2000.

[15] FISCHER, R.; RICHTER, K. Solving a multiobjective traveling salesman problem by dynamic programming. *Mathematische Operationsforschung und Statistik. Series Optimization*, v. 13, n. 2, p. 247–252, 1982.

[16] FLORIOS, K.; MAVROTAS, G. Generation of the exact pareto set in multi-objective traveling salesman and set covering problems. *Applied Mathematics and Computation*, v. 237, n. 1, p. 1–19, 2014.

[17] ANGEL, E.; BAMPIS, E.; GOURVÉS, L. Approximating the pareto curve with local search for the bicriteria tsp (1, 2) problem. *Theoretical Computer Science*, v. 310, n. 1, p. 135–146, 2004.

[18] DUBOIS-LACOSTE, J.; LÓPEZ-IBÁÑEZ, M.; STÜTZLE, T. Anytime pareto local search. *European Journal of Operational Research*, v. 243, n. 2, p. 369–385, 2015.

[19] HANSEN, M. P. Use of substitute scalarizing functions to guide a local search based heuristic: The case of motsp. *Journal of Heuristics*, v. 6, n. 3, p. 419–431, 2000.

[20] PAQUETE, L.; STüTZLE, T. A two-phase local search for the biobjective traveling salesman problem. In: FONSECA, C. M. et al. (Ed.). *Evolutionary Multi-Criterion Optimization*. Berlin, Germany: Springer, 2003. (Lecture Notes in Computer Science, v. 2632).

[21] ELAOUD, S.; TEGHEM, J.; LOUKIL, T. Multiple crossover genetic algorithm for the multiobjective traveling salesman problem. *Electronic Notes in Discrete Mathematics*, v. 36, n. 1, p. 939–946, 2010.

[22] JASZKIEWICZ, A. Genetic local search for multi-objective combinatorial optimization. *European Journal of Operational Research*, v. 137, n. 1, p. 50–71, 2002.

[23] PENG, W.; ZHANG, Q.; LI, H. Comparison between moea/d and nsga-ii on the multi-objective travelling salesman problem. In: *Multi-objective Memetic Algorithms*. 1. ed. Berlin, Germany: Springer, 2009, (Studies in Computational Intelligence, v. 171). cap. 14, p. 309–324.

[24] SAMANLIOGLU, F.; FERRELL, W. G.; KURZ, M. E. A memetic random-key genetic algorithm for a symmetric multi-objective traveling salesman problem. *Computers & Industrial Engineering*, v. 55, n. 2, p. 439–449, 2008.

[25] CHENG, J. et al. Multi-objective ant colony optimization based on decomposition for bi-objective traveling salesman problems. *Soft Computing*, v. 16, n. 4, p. 597–614, 2012.

[26] ZHOU, A.; GAO, F.; ZHANG, G. A decomposition based estimation of distribution algorithm for multiobjective traveling salesman problems. *Computers & Mathematics with Applications*, v. 66, n. 10, p. 1857–1868, 2013.

[27] PAQUETE, L.; STÜTZLE, T. Design and analysis of stochastic local search for the multiobjective traveling salesman problem. *Computers & Operations Research*, v. 36, n. 9, p. 2619–2631, 2009.

[28] BEAN, J. C. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, v. 6, n. 2, p. 154–160, 1994.

[29] BOWMAN, V. J. On the relationship of the tchebycheff norm and the efficient frontier of multiple-criteria objectives. In: FANDEL, G.; TROCKEL, W. (Ed.). *Multiple Criteria Decision Making*. 1. ed. Berlin, Germany: Springer, 1976, (Lecture Notes in Economics and Mathematical Systems, v. 30). cap. 5, p. 76–86.

[30] DEB, K. et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, v. 6, n. 2, p. 182–197, 2002.

[31] ZHANG, Q.; LI, H. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, v. 11, n. 6, p. 712–731, 2007.

[32] JOHNSON, D. S.; MCGEOCH, L. A. Experimental analysis of heuristics for the stsp. In: GUTIN, G.; PUNNEN, A. P. (Ed.). *The Traveling Salesman Problem and Its Variations*. Berlin, Germany: Springer, 2007, (Combinatorial Optimization, v. 19). cap. 10, p. 369–443.

[33] HAMACHER, H. W.; RUHE, G. On spanning tree problems with multiple objectives. *Annals of Operations Research*, v. 52, n. 4, p. 209–230, 1994.

[34] LUST, T. *New Metaheuristics for Solving MOCO Problems: Application to the Knapsack Problem, the Traveling Salesman Problem and IMRT Optimization*. Tese (Doutorado) — Faculté Polytechnique de Mons, Mons, Belgium, 2010.

[35] PAQUETE, L.; STÜTZLE, T. *Stochastic Local Search Algorithms for Multiobjective Combinatorial Optimization: Methods and Analysis*. 2006.

[36] HEIDELBERG, R.-K.-U. *TSPLIB*. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

[37] ZITZLER, E.; THIELE, L. Multiobjective optimization using evolutionary algorithms — a comparative case study. In: EIBEN, A. et al. (Ed.). *Parallel Problem Solving from Nature (PPSN V)*. Berlin, Germany: Springer, 1998. (Lecture Notes in Computer Science, v. 1498).

[38] ZITZLER, E. et al. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, v. 7, n. 2, p. 117–132, 2003.

[39] BADER, J.; ZITZLER, E. Hype: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, v. 19, n. 1, p. 45–76, 2011.

[40] KNOWLES, J.; THIELE, L.; ZITZLER, E. *A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers*. Zurich, Switzerland, 2006.

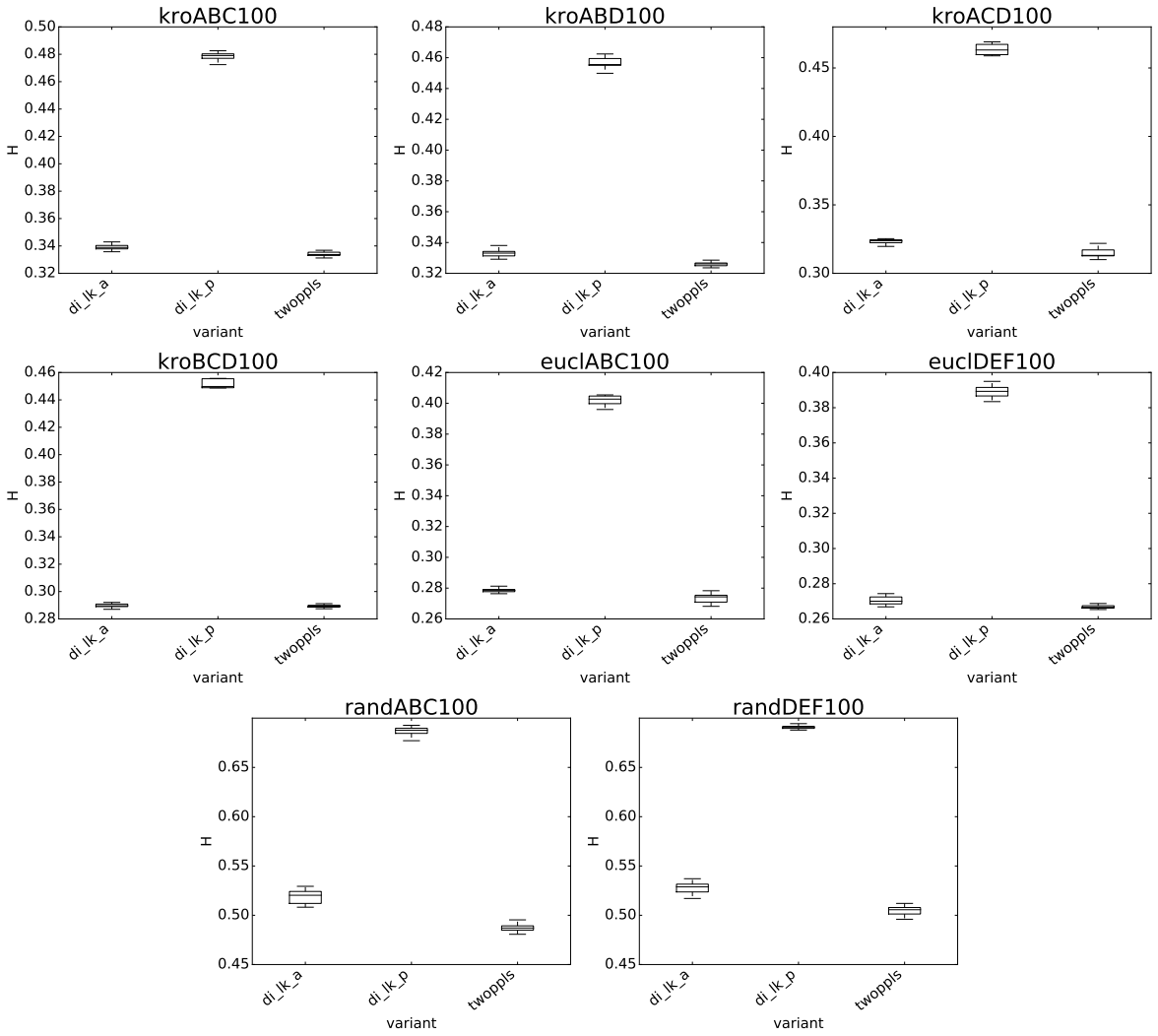

Figure 13. H indicator boxplots for 3-objective instances

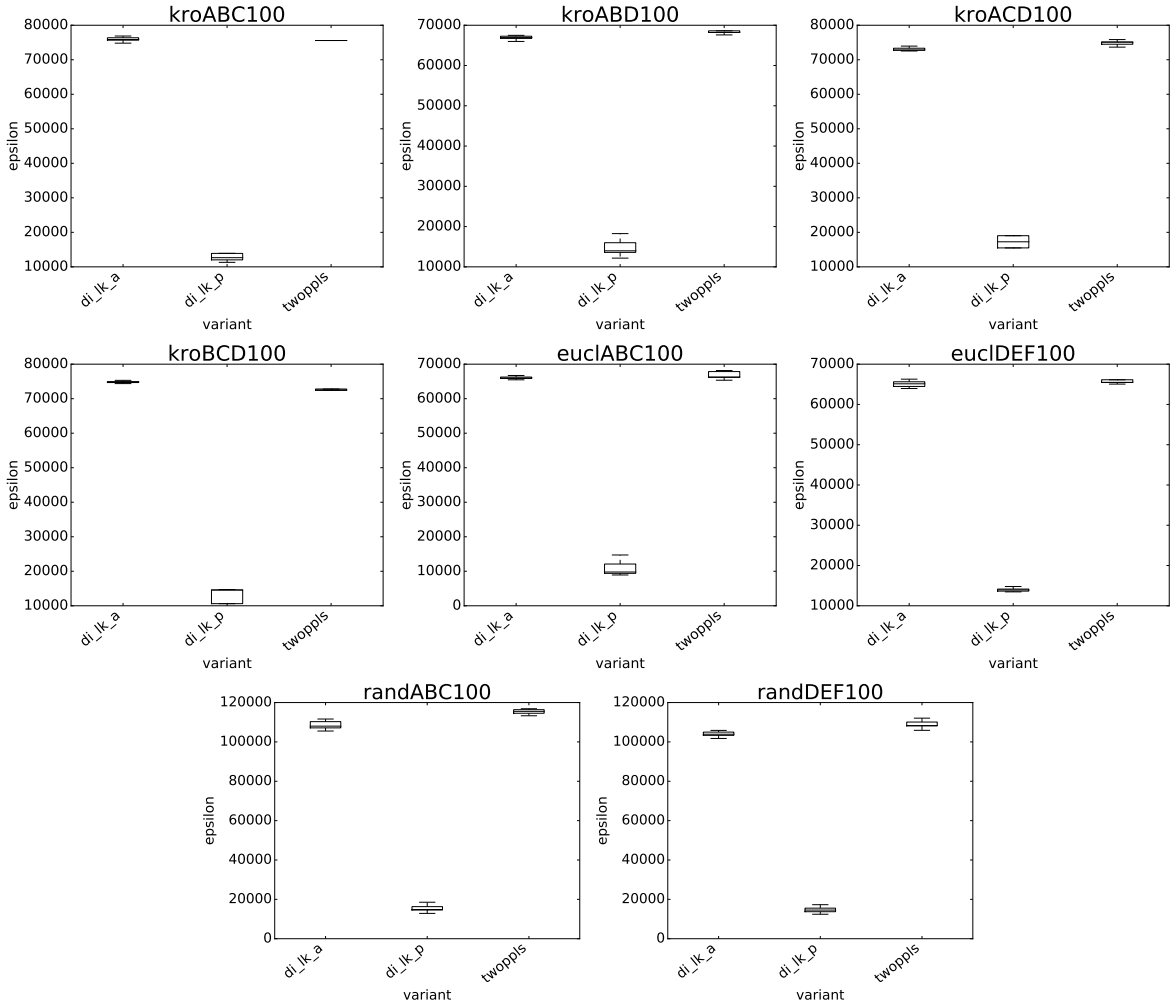

Figure 14. $\varepsilon+$ indicator boxplots for 3-objective instances