Doubly-Rooted Stem-and-Cycle Ejection Chain Algorithm for the Asymmetric Traveling Salesman Problem

César Rego[a*], Dorabela Gamboa[b], Fred Glover[c]

a School of Business Administration, University of Mississippi, University, MS 38677, USA. crego@bus.olemiss.edu

b CIICESI, Escola Superior de Tecnologia e Gestão de Felgueiras, Instituto Politécnico do Porto, Apt. 205, 4610-156, Felgueiras, Portugal. dgamboa@estgf.ipp.pt

c Leeds School of Business, University of Colorado, Boulder, CO 80309-0419, USA. fred.glover@colorado.edu

Latest Revision: December 2015

Abstract – Ejection chain methods, which include the classical Lin-Kernighan (LK) procedure and the Stem-and-Cycle (S&C) reference structure, have been the source of the currently leading algorithms for large scale symmetric traveling salesman problems (STSP). Although these methods proved highly effective in generating large neighborhoods for symmetric instances, their potential application to the asymmetric setting of the problem (ATSP) introduces new challenges that require special consideration. This paper extends our studies on the single-rooted S&C to examine the more advanced doubly-rooted (DR) reference structure. The DR structure, which is allied both to metaheuristics and network optimization, allows more complex network-related (alternating) paths to transition from one tour to another, and offers special advantages for the ATSP. Computational experiments on an extensive testbed exhibits superior performance for the DR neighborhood over its LK counterpart for the ATSP. We additionally show that a straightforward implementation of a DR ejection chain algorithm outperforms the best local search algorithms and obtains solutions comparable to those obtained by the currently most advanced special-purpose algorithms for the ATSP, while requiring dramatically reduced computation time.

Keywords: traveling salesman problem, ejection chains, local search, heuristics, variable-depth neighborhoods, combinatorial optimization

* Corresponding author

1. Introduction

The classical Traveling Salesman Problem (TSP) may be described in the setting of a collection of cities having specified distances between them. The objective is to determine the shortest tour that starts from an arbitrary city, visits each remaining city exactly once, and then returns to the origin. In graph theory, the problem can be defined on a graph $G = (V, A)$, where $V = \{v_1, \ldots, v_n\}$ is a set of n vertices (nodes) and $A = \{(v_i, v_j) \mid v_i, v_j \in V,\ i \neq j\}$ is a set of arcs, together with a non-negative cost (or distance) matrix $C = [c(v_i, v_j)]$ associated with A. The problem is called the symmetric TSP (STSP) if $c(v_i, v_j) = c(v_j, v_i)$ for all $(v_i, v_j) \in A$, and the asymmetric TSP (ATSP) otherwise. Elements of A are often called edges (rather than arcs) in the symmetric case and may be denoted by $\{v_i, v_j\}$ rather than (v_i, v_j) since they are unordered rather than ordered pairs. The STSP (ATSP) consists in determining the Hamiltonian cycle (circuit), often simply called a *tour,* of minimum cost.

The version of the STSP in which distances satisfy the triangle inequality ($c(v_i, v_j) + c(v_j, v_k) \geq c(v_i, v_k)$ for all distinct $v_i, v_j, v_k \in V$) is the most studied special case of the problem, notably including the particular instance where V is a set of points in a 2-dimensional plane and $c(v_i, v_j)$ is the Euclidian distance between v_i and v_j. The ATSP is more general than the STSP and likewise embraces a wide range of applications particularly arising in scheduling optimization in manufacturing [5] and in vehicle routing in distribution and transportation networks [11, 21, 35]. These basic applications are significantly expanded by the variety of complex real-world vehicle routing problems (encompassing time-windows and other hard constraints and multiple vehicles) that can be solved by first re-casting them as an ATSP using polynomial-time transformations [23, 24, 36].

The ATSP is also much more difficult to solve than the STSP for both exact and approximation algorithms [2, 34]. Perhaps due to this difficulty the research on TSP has been mostly focused on the STSP, and algorithm developments are much less advanced for the ATSP. Nevertheless, remarkable progress in local search algorithms for the ATSP has come about by drawing on generalizations of the most powerful neighborhood search methods for the STSP, specifically the ejection chain methods represented by the classical Lin-Kernighan (LK) procedure [25] and the Stem-and-Cycle (S&C) reference structure [12]. (For an extensive coverage of these methods we refer the reader to Rego et al. [30].)

In the general context of combinatorial optimization, ejection chains are constructions to create variable-depth neighborhoods efficiently for local search procedures. The underlying technique consists of decomposing a very large neighborhood into a sequence of component neighborhood structures that can be evaluated in polynomial time. Each component neighborhood structure in the sequence does not usually correspond to a feasible solution but constitutes a reference structure that permits a feasible solution to be obtained efficiently. The S&C is a fundamental structure in a number of other reference structures used in the creation of ejection chain methods. (For algorithm designs and implementations of the S&C reference structure see Pesch and Glover [27], Rego [29], and Gamboa, Rego and Glover [9, 10].)

This paper explores a generalization of the S&C reference structure for traveling salesman problems called the Doubly-Rooted (DR) S&C (Glover [13]) that has special advantages for the ATSP. These neighborhoods exhibit a special property called *combinatorial leverage*, which enables solutions dominating exponential alternatives to be obtained with polynomial effort. Excellent studies on the domination analysis of these neighborhoods for the ATSP can be found in Punnen and Kabadi [28] and Gutin and Yeo [15].

A key contribution of this paper may be viewed in the context of two earlier publications that have deservedly received wide acclaim. The first concerns the original proposal of the now-famous Lin-Kernighan (LK) method for the Symmetric TSP [25]. The second, which is yet more germane, is a straightforward adaptation of the LK algorithm to the Asymmetric TSP by Kanellakis and Papadimitriou [22]. In the latter publication, the authors base the relevance of their contribution on results they obtain for ATSP instances ranging in size from 30 to 90 nodes and suggest that the algorithm is very suitable for the solution of very large instances of the ATSP. This specialized LK procedure (referred in the literature by the KP method) has been a mainstay of ATSP references for over three decades. An enhanced implementation of the method that includes advanced data structures and various implementation techniques to accelerate the search has been undertaken by Cirasella et al. [6] to address instances up to 1000 nodes or more. The Cirasella et al. paper is particularly devoted to showing the potential of this specialized LK variant to solve instances that are currently considered large-scale in the asymmetric setting.

Our main objective here is to provide a comparative study of the DR neighborhood structure and the generalized LK neighborhood underscored in the KP method [22] and

more recently used in the current state-of-the-art local search algorithm for the ATSP by Cirasella et al. [6]. Additionally, in order to place our developments in a broader perspective we extend our computational analysis to include comparisons with the latest metaheuristic advances for the ATSP involving Memetic Algorithms [3], Ant Colony Optimization [1, 37], Extremal Optimization [4], Genetic Algorithms [26], and hybrids of these methods [7, 39] all aimed at extending local search with advanced search guidance.

The remainder of this paper is organized as follows. Section 2 presents the doubly-rooted ejection chain method and describes the algorithm. Section 3 derives a comparative analysis of the DR with LK neighborhood structures and other state-of-the-art algorithms for the ATSP. Section 4 summarizes our findings and provides directions for further research.

2. The Doubly-Rooted Stem-and-Cycle Algorithm

The general structure of the algorithm can be briefly described as follows. Starting from an initial tour, the algorithm attempts to improve the current solution iteratively by means of a network (or graph) related subpath ejection chain method, which generates moves coordinated by a doubly-rooted stem-and-cycle reference structure.

2.1. The Doubly-Rooted Reference Structure

The doubly-rooted reference structure generalizes the single-rooted stem-and-cycle reference structure in two basic forms: a *bicycle* in which the roots are connected by a single path, joining two cycles and a *tricycle* in which the two roots are connected by three paths, thereby generating three cycles. Our terminology may be understood by reference to Figure 1, which illustrates the basic S&C reference structure with root r and tip node t and the two forms of the DR structure with root nodes denoted by r_1 and r_2.

In both the S&C structure and the DR structure there are exactly three nodes linked to each root; however special cases may occur leading to degenerate forms of these structures. If the tip and root nodes coincide in the S&C then only two nodes are linked to the root and the structure corresponds to a feasible tour as shown in Figure 2(a). Likewise, if the two roots coincide in the DR structure, then four nodes are linked to that root originating a bicycle structure with the two cycles connected by the root node as shown in Figure 2(b).

The nodes adjacent to the roots are called *subroots* and are divided into two classes: *cycle subroots* and *non-cycle subroots*, where the latter are those that lie on the path between the two roots of a bicycle. Cycle and non-cycle subroot nodes are indicated by grey and black shading, respectively. We do not highlight the non-cycle subroot in the S&C as it has no special use in this structure.

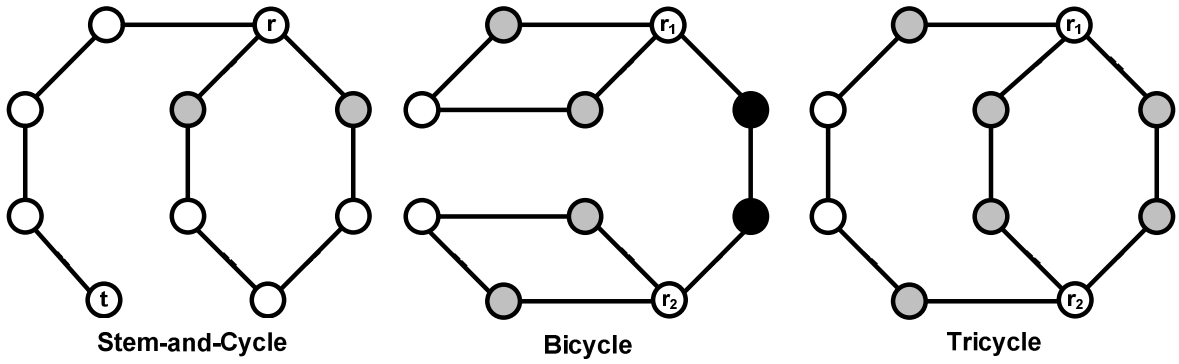

Figure 1. Single-rooted and doubly-rooted S&C reference structures.

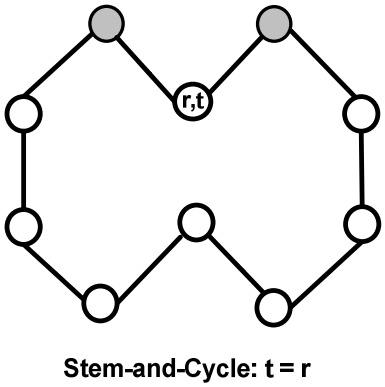

Figure 2(a). Degenerate single-rooted structure.

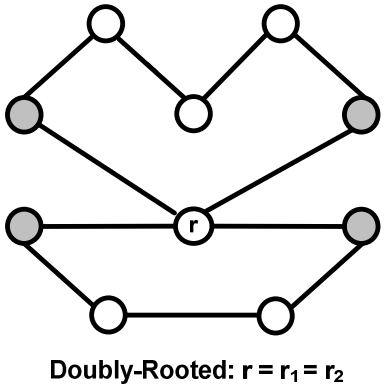

Figure 2(b). Degenerate doubly-rooted structure.

In an ejection chain process we distinguish *ejection moves* that transform one reference structure into another of the same type from *trial moves* that create a valid solution structure (i.e., a TSP tour) from a reference structure. Consequently, the structure obtained with a trial move is called a *trial solution*. From this point on we focus on the DR structure and refer to the S&C whenever relevant for the explanation of the doubly rooted ejection chain process. The interpretation and uses of these structures will become clear in the process.

Ejection Moves

The rules to transition between structures are given by two types of ejection moves.

Cycle subroot ejection move: Select a cycle subroot *s* and an associated root *r*. Add an arbitrary new edge (*s*, *j*) (not in the current structure) and delete the edge (*s*, *r*). After the step, *j* becomes a root (and *r* is no longer a root unless the two roots coincided before the step).

Non-cycle subroot ejection move: Select a non-cycle subroot *s* and an associated root *r*. Add a new edge (*s*, *j*) such that *j* lies on the cycle in common with *r*, and delete the edge (*s*, *r*). Node *j* becomes a new root (and *r* is no longer a root).

These moves are exactly the same for both the symmetric and asymmetric settings, though in the asymmetric case, the added and deleted arcs must be directed the same relative to the subroot *s* (i.e., the added and deleted pair is either (*s*, *j*) and (*s*, *r*) or (*j*, *s*) and (*r*, *s*)). Figure 3 depicts the application of cycle and non-cycle ejection moves to the DR structures of Figure 1. In the figure, grey lines denote edges that are added by the move and dotted lines denote edges to be deleted.

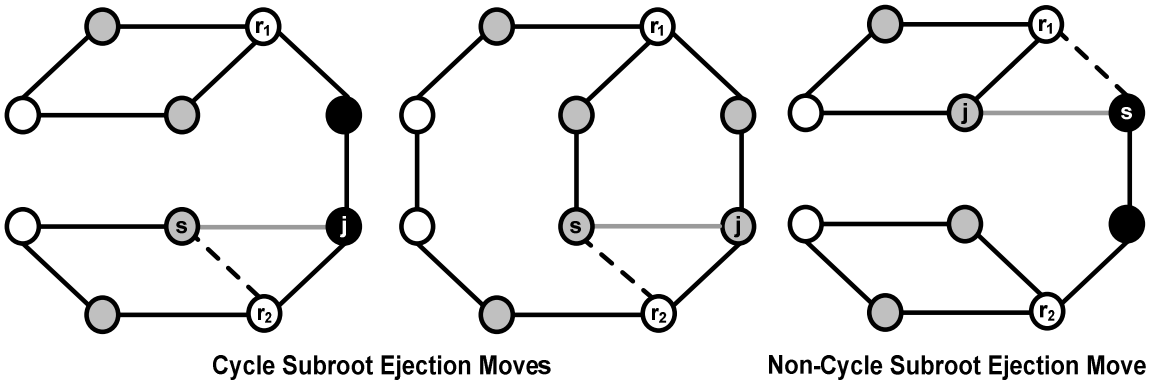

Figure 3. Examples of cycle and non-cycle ejection moves.

Trial Solutions

The trial solutions available to the doubly-rooted structure are those generated by the union of the trial solutions available to the single-rooted stem-and-cycle (S&C) structure obtained by deleting any edge linking a root node to a cycle subroot. Such a subroot becomes the *tip* of the S&C, while the (root) node that remains with three incident edges becomes the S&C root. Trial moves are created by linking the tip to one of the subroots and deleting the arc that unions the subroot to the root, giving rise to two possible trial

solutions associated with each subroot. Figure 4 illustrates the creation of a possible trial solution associated with a single-rooted S&C structure obtained from the bicycle and tricycle doubly-rooted S&C of Figure 1. In the diagrams, nodes t_1 and t_2 specify the tip nodes associated with the two S&C structures from which the trial solutions are obtained.

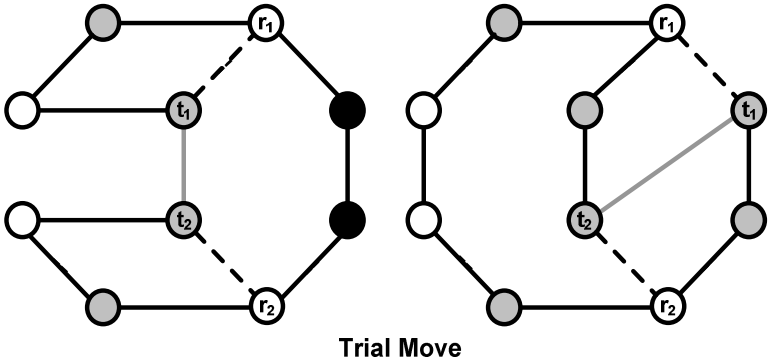

Figure 4. Examples of trial moves on bicycle and tricycle structures.

In fact, the trial solutions that result by transforming a cycle subroot *s* into a tip *t*, for each such *s* associated with a given root *r*, are the same as the trial solutions similarly produced from the subroots of the other root, and hence attention can be restricted to only one of the two sets of subroots for this purpose. Thus, each cycle subroot of a given root produces two trial solutions. The enriched pool of such trial solutions, together with an enriched set of moves for transitioning from one reference structure to the next, provide the potential advantage of the doubly-rooted structure over the single-rooted stem-and-cycle structure. For the asymmetric case the advantage is provided by the fact that a subset of these possibilities is capable of preserving tour orientation, which gives rise to exactly one possible trial solution in a bicycle structure and two trial solutions in a tricycle. These orientation preserving moves are shown in Figures 5(a) and 5(b).

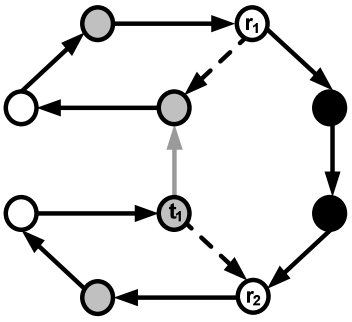

Figure 5(a). Trial move on a bicycle structure.

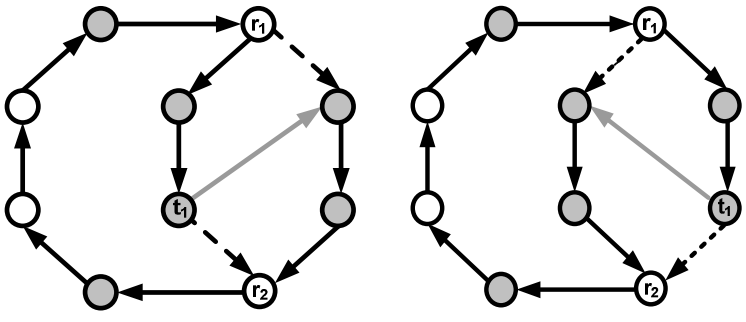

Figure 5(b). Trial moves on a tricycle structure.

2.2 The Ejection Chain Procedure

In the design of stem-and-cycle ejection chain procedures we impose what we call *legitimacy restrictions* in order to achieve two main purposes: (1) to prevent the method from visiting solutions already inspected during the ejection chain process; (2) to generate special forms of alternating paths which have proved useful in several classical graph theory problems. For the first purpose it is sufficient to stipulate that no deleted edge is added back during the construction of the chain. The second purpose deserves some additional attention.

Alternating Path Considerations

In classical alternating path methods in graph theory, and in neighborhood search processes related to them, the customary approach is to restrict the edges qualifying for deletion to be edges of the starting solution. Methods that use this approach, which include the classical LK procedure, may be viewed as *static alternating path methods*, because new edges introduced during the construction of the chain are exempt from deletion. However, certain neighboring solutions cannot be obtained except by generating alternating paths in which previously added path edges are also candidates to be dropped. Thus, in contrast to classical approaches, this produces a *dynamic alternating path method*. In fact, the dynamic alternating paths provided by single- and doubly-rooted S&C structures provide the ability to reach any TSP tour by a succession of moves starting from any other tour, in contrast to the LK neighborhood that can fail to find certain types of improved tours even if they are close to the current tour, as it will be discussed later. As proved in Glover [13], this ability to reach any possible tour is retained by adding a simple "non-reversal" condition, which prevents an edge from being deleted if is inserted immediately after deleting another edge that was previously inserted. These restrictions define the legitimacy conditions for the S&C algorithm described in Rego [29], and are also incorporated into the present doubly-rooted S&C algorithm.

A general design of the doubly-rooted stem-and-cycle neighborhood search procedure can be described as in Figure 6, where we define a legitimate neighborhood for a node v_i, denoted by $LN(v_i)$, as the subset of nodes of G that do not violate the legitimacy restrictions identified above.

Step 0. Initialization

(a) Initialize a legitimate neighborhood for all nodes.
(b) Denote the starting solution by S
(c) Select the initial root nodes v_{r_1} and v_{r_2}
(d) Create the initial DR structure:
Select a cycle subroot node for each root v_{r_1} and v_{r_2} and initialize the values of all other subroot nodes.
(e) Set $k = 0$ and $L =$ maximum number of levels of an ejection chain.

Step 1. Generate the ejection chain

(a) Ejection Move:
For each subroot v_{s_k} compute the value of the ejection move for each node $v_j \in LN(v_{s_k})$ as follows: $E_k = c(v_{s_k}, v_j) - c(v_{s_k}, v_{r_k})$, where v_{r_k} is the root node associated with v_{s_k}. If v_{s_k} is a non-cycle subroot, then the ejection move is only possible if $v_j \in CY_k$, where CY_k is the cycle in common with v_{r_k}.
(b) Select the node v_{j*} that yields the minimum E_k value and keep track of the subroot node v_{s_k} considered for the move.
(c) Trial Move:
Consider one of the root nodes v_r, and the three possible S&C structures that can be obtained by deleting an edge $(v_{r,}v_{s_i})$, where $v_{s_i}(i = 1,2,3)$ represents each subroot associated with v_r;
Compute the value of the trial moves associated with each cycle subroot $v_{cs_i}(i = 1,2)$ of each S&C structure and choose one that minimizes $T_k = c(v_t, v_{cs_i}) - c(v_{cs_i}, v_r)$, where v_t represents the tip node of considered S&C structure. The solution cost change is given by $\Delta_k = E_k + T_k$.
(d) Keep track of the level k^* that produces the best trial tour so far and record the subroot node involved in the trial move.
(e) Update LN.
(f) Set $k = k + 1$ and set $v_{r_i} = v_{j*}$, where v_{r_i} is the root affected by the move.
(g) If $k < L$ and LN is not empty return to Step 1. Otherwise go to Step 2.

Step 2. Perform the compound move

(a) Apply to S each ejection move considered in the ejection chain up to the level k^*.
(b) Complete the update of S by executing the trial move for the level k^*.

Figure 6. An iteration of the doubly-rooted stem-and-cycle procedure.

Complexity

The complexity of the doubly-rooted stem-and-cycle (DR) ejection chain procedure is determined as follows. For selecting an ejection move $2(n-3)$ operations may be considered for each subroot, hence a maximum of $6\times2(n-3)$ operations in total. Six additional operations are necessary to evaluate all possible trial moves associated with that ejection move (i.e. two operations for each of the three subroots considered for trial move evaluation). Hence, one level of a DR ejection chain may be performed in $O(n)$ time. According to the legitimacy restriction the number of levels of an ejection chain is

bounded by $n(n-1)/2-n$; that is, all the edges may be deleted only once, except the number of the edges corresponding to the cardinality of a solution. Thus, the overall complexity of an ejection chain evaluation may reach $O(n^3)$. However, since the best trial solution is usually found at a relatively lower level, this effort can be notably reduced. The theoretical justification for this remark may be found in the theorem of Glover [12] which proves that, by using a subpath ejection method like the one used by our algorithm, subject to the restrictions that we consider as legitimate, it is always possible to generate any neighboring solution that differs from the starting solution by m edges by adding less than $2m$ edges. Since two neighboring solutions can differ at most by n edges, then $2n$ may define an upper bound on the number of levels of an ejection chain and therefore, L is considered a user-supplied parameter. Hence, the real worst case complexity for one iteration of the algorithm may be considered as $O(n^2)$.

2.3 The Ejection Chain Algorithm

As stated earlier the LK procedure relies on a static alternating path construction, which constitutes a limitation of the method in relation to more general ejection chain methods that have the ability to generate neighborhoods produced by dynamic alternating paths. It is well known that the paths generated by the LK neighborhood are unable to reach some tours that differ only by 4 edges from the current tour, which can otherwise be obtained by the so-called *double-bridge* neighborhood, originally suggested as a supplement of the basic LK procedure and likewise considered in the KP variant of the method [22] for the ATSP. An interesting theoretical analysis provided in Funke, Grünert and Irnich [8] shows that even a generalization of the LK approach that incorporates generalized alternating paths cannot reach solutions accessible to the S&C neighborhood.

The KP method for the ATSP starts with a type of LK search based on sequences of special 3-opt moves (with segment reordering) rather than 2-opt moves (with segment reversals) used in the original LK for the STSP. When the LK search fails to improve the solution, the method searches for an improving 4-opt *double-bridge* move (with no reversals). Then KP returns to LK search and iterates in this manner until neither of the searches improves the tour.

In order to assess the effectiveness of the doubly-rooted S&C neighborhood structure compared to the KP variant, we have adopted for our implementation a similar alternating strategy between the ejection chain search and the 4-opt double-bridge neighborhood. We

should stress that our 4-opt search is made separate from the basic ejection chain process only for the sake of comparisons since the doubly-rooted S&C neighborhood can also generate double-bridge moves. We should also note that the 4-opt search used in our DR algorithm corresponds to the efficient $O(n^2)$ procedure of Glover [14] used in its recent implementations [6] analyzed here, as opposed to the potentially $O(n^4)$ procedure considered in the original KP algorithm [22]. Our implementation of the 4-opt double-bridge procedure adopts the compact version devised by Johnson [17].

3. Computational Analysis

Our tests reported here employ a straightforward implementation of an ejection chain algorithm based on the DR structure for the ATSP. Comparisons are established with the KP variant of the LK approach proposed by Kanellakis and Papadimitriou [22] for the ATSP. The KP implementation analyzed in this paper is due to Johnson and McGeoch and described in Cirasella et al. [6], though in the latter, the authors only report results for a more advanced implementation of the algorithm based on the so-called Iterated LK framework, denoted herein by iKP. The computational study of the basic KP is reported in [18].

A number of attempts have been made to solve ATSPs using state-of-the-art algorithms for the STSP by first transforming the ATSP into an equivalent STSP instance. To address this approach the currently most effective LK variant for the STSP due to Helsgaun [16], herein denoted by H-LK, is also included in our analysis. Results for this advanced H-LK algorithm are also reported in [18].

After completing comparisons with all LK variants whose results have been reported for the ATSP, we complete our analysis by establishing comparisons with the latest and most advanced metaheuristic algorithms for the ATSP.

3.1. Comparison with Lin-Kernighan Variants

Table 1 reports the computational results for all ATSP instances in the TSPLIB [32] testbed. The first two columns give the problem name and its size. The next columns show for each algorithm the relative percentage deviation from the optimal solution value, and the running time (in seconds) necessary to find those solutions. (Dashed cells indicate that results are not available for the corresponding problems and algorithms, and the optimal solutions found are highlighted in boldface figures.) The summary at the

bottom of Table 1 provides the average percent deviations from optimality, the average running times and the number of optimal solutions found over the set of problems tested by the corresponding algorithm.

The results for the KP, iKP, and H-LK algorithms are averages over 5 or more (independent) runs for each instance (cf. [6], pp. 14) obtained by choosing a different city to initiate the nearest neighbor heuristic used to construct a starting tour (cf. [6], pp. 21). Runs were performed using a Silicon Graphics Power Challenge machine with 31 196 MHz MIPS R10000 processors, 1 MB 2nd level caches and 7.6 GB of main memory shared by all processors. The implementation is a serial one; therefore if no implicit parallelism is employed by the compiler the algorithm presumably uses only one of the processors available. For comparison purposes, results for the DR algorithm are obtained in a similar manner except that only 5 runs were used for each instance and all runs were carried out using an Intel Core Duo 2.66 GHz processor with 3 GB RAM.

In order to accurately compare computational times between algorithms, we provide normalized running times derived from runs of the standard benchmark code available in the DIMACS Implementation Challenge website [20]. Using the Hungarian method benchmark code as suggested in [18] for the case of the ATSP, we encountered relative factors of 1.500, 3.5714, and 4.6193 for n = 100, 316 and 1000, respectively; hence we found 3.2 a reasonable compromise for the actual factors of the two machines.

Before discussing the results, we point out a number of important considerations. The only performance metric used in the literature for all LK variants consists of averages computed over a very small number of runs, which makes comparison between algorithms very difficult. Clearly, a sample size of 5 runs is not sufficiently large for the assessment of a randomized algorithm to be statistically significant, but may be acceptable if not much variability is found in sufficiently large experiments. Otherwise, there is a good deal of uncertainty concerning how representative the averages may be, including the degree to which they may have been if different cities had been chosen to initiate the algorithm, and raising doubt about the quality of the best solution found over the multiple runs.

		Percentage Excess over Optimal					Normalized Running Time				
Number of Runs		5+	5+	5+	5	5	5+	5+	5+	5	5
Instance	n	KP	iKP	H-LK	DR Best Run	DR Average	KP	iKP	H-LK	DR Best Run	DR Average
atex1	16	—	0.44	—	**0.00**	**0.00**	—	0.07	—	0.00	0.00
atex3	32	—	0.03	—	**0.00**	**0.00**	—	1.69	—	0.00	0.04
atex4	48	—	**0.00**	—	**0.00**	0.99	—	2.22	—	0.29	0.69
atex5	72	—	0.15	—	0.08	1.10	—	7.77	—	1.57	2.18
atex8	600	4.25	0.99	0.82	1.55	2.99	3.38	2112.70	123.0	1233.63	898.85
big702	702	2.10	0.21	0.41	0.58	0.77	6.04	1136.52	119.0	579.20	600.74
br17	17	—	**0.00**	—	**0.00**	**0.00**	—	0.11	—	0.00	0.00
code198	198	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.54	114.24	6.5	0.03	0.04
code253	253	0.10	**0.00**	0.28	**0.00**	**0.00**	1.09	267.14	24.5	0.00	0.88
dc112	112	0.39	0.31	0.28	0.19	0.30	15.47	904.31	4.5	4.93	4.95
dc126	126	0.65	0.64	0.54	0.08	0.14	22.69	1533.84	5.5	13.25	10.68
dc134	134	0.57	0.55	0.02	0.43	0.55	13.43	714.98	6.5	2.59	5.73
dc176	176	0.67	0.59	0.49	0.61	0.79	20.48	1448.29	105.0	36.00	28.78
dc188	188	0.59	0.52	0.13	**0.00**	0.24	12.98	674.15	28.5	1.44	27.03
dc563	563	0.79	0.78	0.12	1.29	1.37	111.95	4379.74	2231.5	429.06	352.58
dc849	849	0.62	0.61	0.23	0.64	0.66	114.80	5666.96	2180.0	9.98	263.96
dc895	895	0.60	0.58	0.25	0.57	0.65	144.43	7214.80	22077.0	3408.35	2399.71
dc932	932	0.26	0.27	0.26	0.10	0.13	119.17	4611.56	72373.0	2699.78	2139.80
atex1	53	—	0.01	—	**0.00**	0.99	—	7.99		0.26	0.59
atex3	70	—	0.02	—	**0.00**	0.33	—	4.83		0.90	1.64
atex4	101	3.11	0.13	**0.00**	**0.00**	0.27	0.04	0.62	1.0	5.34	1.40
atex5	111	4.04	0.12	**0.00**	**0.00**	0.18	0.04	1.00	1.0	3.20	2.54
atex8	121	3.12	0.33	**0.00**	**0.00**	0.06	0.05	1.70	1.0	1.60	3.84
big702	131	2.16	0.27	**0.00**	**0.00**	0.16	0.06	1.52	1.0	0.74	3.95
br17	141	3.15	0.21	**0.00**	**0.00**	0.26	0.06	1.23	1.5	5.86	10.73
code198	151	4.43	0.40	**0.00**	**0.00**	1.96	0.07	1.18	1.0	23.78	11.88
code253	161	5.89	0.24	**0.00**	**0.00**	1.45	0.07	1.43	2.0	4.96	6.73
dc112	171	4.44	0.45	**0.00**	**0.00**	0.70	0.09	1.52	2.5	13.82	15.12
dc126	34	—	3.02	—	**0.00**	0.82	—	0.11	—	0.00	0.10
dc134	36	—	0.03	—	**0.00**	0.08	—	0.11	—	0.06	0.12
dc176	39	—	**0.00**	—	**0.00**	0.08	—	0.12	—	0.03	0.10
ftv44	45	—	1.30	—	**0.00**	0.35	—	0.12	—	0.03	0.22
ftv47	48	—	0.49	—	**0.00**	0.14	—	0.17	—	0.06	0.16
ftv55	56	—	**0.00**	—	**0.00**	**0.00**	—	0.25	—	0.10	0.15
ftv64	65	—	0.33	—	**0.00**	0.16	—	0.37	—	0.06	0.13
ftv70	71	—	0.08	—	**0.00**	0.65	—	0.30	—	0.16	0.76
ftv90	91	—	0.08	—	**0.00**	**0.00**	—	0.47	—	0.54	0.96
kro124p	100	—	1.28	—	**0.00**	0.60	—	0.34	—	4.03	4.86
p43	43	—	0.01	—	**0.00**	0.01	—	23.81	—	0.10	0.22
rbg323	323	0.78	0.30	**0.00**	**0.00**	0.05	3.71	414.53	46.5	24.32	40.72
rbg358	358	1.50	0.79	**0.00**	**0.00**	0.03	3.33	342.26	120.5	31.30	74.93
rbg403	403	0.22	0.11	**0.00**	**0.00**	**0.00**	9.00	743.99	221.5	9.38	33.92
rbg443	443	0.11	0.09	**0.00**	**0.00**	0.03	11.74	908.12	154.0	2.88	14.22
ry48p	48	—	1.92	—	**0.00**	0.61	—	0.16	—	0.86	0.44
td100.1	101	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.20	39.90	0.5	0.48	1.13
td1000.20	1001	0.01	**0.00**	**0.00**	**0.00**	**0.00**	7.29	720.57	140.5	438.30	814.89
td316.10	317	**0.00**	**0.00**	0.11	**0.00**	**0.00**	3.87	1950.07	24.5	2.82	4.98
Average 28 Instances		1.59	0.34	0.14	0.22	0.49	22.36	1282.46	3571.75	320.96	277.67
Average 47 Instances		—	0.40	—	0.13	0.44	—	765.10	—	191.41	165.70
Optimal 28 Instances		3	5	15	18	6					
Optimal 47 Instances		—	9	—	36	11					

Table 1: Comparative analysis of ejection chain algorithms.

Additional confusion arises from the fact in some instances more than 5 runs are allowed, but without explanation of this exception to the testing protocol. In [6] the authors promised to provide full details in the final paper [18] with respect to how the experiments have been conducted, but unfortunately this did not happen. Results reported for the KP algorithm are likewise limited, in this case by examining only a subset of the standard TSPLIB testbed.

To account for these difficulties and to ensure that we do not give our DR algorithm an unfair advantage, we have limited the number of our runs to exactly five for all instances. Beyond this, however, we test our algorithm on the complete testbed of 48 real-world instances of TSPLIB, and in addition to the average results over the five runs we report the solution found by the best of those runs. We argue that although the average solution quality over a specified number of runs is a relevant measure, the best solution found is a more appropriate metric of performance when a small number of runs is considered. In real applications, where time to conduct a large number of runs is not always available, one usually wants to implement the best solution found within the allotted time, or to present the decision maker with a subset of the best solutions for further consideration. Since the best solution found was not given in the experiments reported in [6] and [18] we assume for the sake of the analysis that all runs on individual instances produced solutions of similar quality (since otherwise the conclusion on the relative performance between the LK variants would have been defective by the absence of this information, especially if the optimal solution had been found in at least one of the runs). Although our DR algorithm exhibits a similar behavior on all test runs, with little variability in solution quality, it often manages to find the optimal solution in at least one of these runs for a given instance; hence, we consider it informative to report the solution and computation time of our best run.

For an appropriate comparative assessment, we also note that the KP and the DR algorithms do not belong to the same class of heuristics as the iKP and H-LK algorithms, which reapply their core algorithm multiple times and take one or more orders of magnitude longer to run on the larger instances. In particular, following the classification established in the DIMACS TSP Implementation Challenge [20] and consistent with [18, 19], the KP and DR algorithms may be considered *local search* procedures, while the iKP and H-LK types of algorithms are categorized as *iterated* (*chained* or *repeated*) *local search*. Iterated local search methods expend greater effort to explore the solution space more thoroughly and so are expected to find better solutions to compensate for their substantially increased running times. However, as our results make apparent, the

expectation of finding better solutions proves valid only in relation to the local search KP method, while our new DR algorithm obtains exceedingly high quality solutions without the expenditure of such additional effort.

We now discuss a few of the main conclusions that can be inferred from the results reported in Table 1. First, our DR algorithm discloses itself to be far more effective and robust than the KP algorithm. Out of 28 instances for which results are available for KP, in only 4 instances did the KP algorithm manage to find tours that are on average slightly better than the averages found by the DR algorithm and in only 2 instances did these averages surpass the quality of the solutions found on our best run. By contrast, in some cases the average quality of solutions found by the DR algorithm exceeded that of the KP algorithm by more than 4% (a significant amount relative to differences in TSP tours) and our best solutions exceeded these KP average solutions by almost 6%. Although the non-iterated form of KP terminates its search much earlier than DR, the solution quality of the KP approach is seriously compromised. For the ftv* instances, the KP algorithm produces solutions that on average deviate from optimality by 3.79% compared to an average deviation of only 0.63% produced by our DR algorithm. Moreover, the best of our runs finds the optimal solution for all instances of this class. For the other instances in this reduced testbed the KP algorithm produces tours of high quality, although the DR algorithm still exhibits superior performance. It is possible to predict that this relative advantage of our DR algorithm over the KP algorithm would be even more significant if KP had been tested on the remaining 19 instances of the standard testbed. These instances were omitted in [18] due to the imposed size limit of less than 100 cities, but it is noteworthy that these instances are not necessarily easy even for the advanced iKP variant of this KP algorithm. In particular, for instances as small as 34 cities (ftv33), the long running iKP procedure produces solutions that are on average more than 3% away from optimality and is unable to produce a zero gap even for the smallest of these instances (atex1) containing only 16 cities. Indeed, only in 4 out of the 19 instances did iKP manage to find a zero gap from optimality. Comparing DR with iKP on the basis of average solution quality over the multiple runs and on the whole set of 47 instances, the DR algorithm obtains the best average solution quality on 11 instances while iKP finds it on 10. Among these, a 0.00% gap from optimality is achieved on 9 instances by the iKP algorithm and on 11 instances by the DR algorithm. Overall, the iKP and DR algorithms find an equal number of 19 better solutions relative to each other and match on 9 instances. In sum, the two algorithms exhibit similar average solution quality over the whole test set. Hence, in a comparison solely on the basis of the average solution quality over the multiple runs neither of the two algorithms appears to dominate the other.

However, it is apparent that the iKP algorithm requires significantly more computational time than the DR algorithm, regardless of possible imperfections in running time normalization relative to the computers where the algorithms were run. For instance, on the rbg* problems the DR algorithm finds its best tours in a fraction of the time (10% or less) required by the iKP algorithm to find its best tours, whose quality does not match that obtained by the DR method. In some cases, as illustrated by rgb403 and rbg443, the DR algorithm requires less than 5% of the time required by the iKP algorithm. Differences in speed of a similar order of magnitude may be seen in the dc* and td* instances, where on dc126 and td316.10, for example, these relative percentages may translate to over 25 minutes for the iKP compared to less than 10 seconds for the DR algorithm to find tours of the same or better quality.

In sum, the DR algorithm not only outperforms the basic KP algorithm but is comparably effective and considerably more efficient than its advanced iKP variant. This relative advantage is especially relevant considering the fact that our algorithm does not take advantage of the various speed-up optimizations used in the KP and iKP algorithms. The performance of the DR algorithm is additionally noteworthy when comparisons are made relative to its best run for each instance. As shown in Table 1, the average excess over the optimum tour length achieved by this best run is 0.13%, which is significantly lower than the 0.40% achieved by iKP while consuming a much longer computation time than that required by the DR algorithm. Also, the DR algorithm finds 36 optimal solutions out of the 47 instances compared to only 9 optimal solutions found by the iKP algorithm.

Comparing with the H-LK algorithm we should first note that results on individual instances as reported in [18] are only available for the reduced testbed of larger instances; however, the same study (cf. [18], pp. 450) points out that this algorithm obtains solutions within 0.3% of optimal within a (normalized) second for the omitted instances with less than 100 cities. For these same instances the average deviation from optimal of our DR algorithm is 0.36% obtained in 0.7 (normalized) seconds, while the best of our runs finds the optimal solution for all but one of these instances (atex8) for which the solution still falls within a 1% gap from optimal. As a result, our best run produces a 0.00% gap in 0.48 (normalized) seconds on average. While both algorithms produce solutions of similar quality on average for these 19 instances, our DR algorithm appears to be more efficient, especially considering that the running time to perform the ATSP to STSP transformation required by the H-LK algorithm is not included in the times reported in [18]. For the remaining 28 instances, the H-LK algorithm produces solutions of better quality than our DR algorithm (0.14% versus 0.49% excess over optimal) on average, but

this difference in quality comes at the expense of a computation time that is more than an order of magnitude greater than that used by our DR algorithm. Finally, a comparison with our best run shows that although the excess over optimal obtained by H-LK (0.14%) is still lower than that of DR (0.21%), this difference is not significant, especially considering that the DR algorithm was capable of finding 18 optimal solutions while H-LK could only find 15 of them.

3.2. Comparisons with Advanced Metaheuristic Algorithms

We now extend our computational analysis by establishing comparisons with several of the most advanced metaheuristic algorithms in the ATSP literature. Table 2 provides results for the following additional algorithms:

- Model-Induced Max-Min Ant Colony (MIMM ACO) [1]
- Max-Min Ant Colony (MM ACO) [37]
- Extremal Optimization (EO) [4]
- Cooperative Genetic Ant System (CGA) [7]
- Memetic Algorithm (MA) [3]
- Hybrid Genetic Algorithm (HGA) [39]
- Genetic Algorithm (GA) [26]

The results shown in Table 2 are a compilation of two recent computational studies. The results for MIMM-ACO, MM-ACO, EO, and CGAS were obtained from Bai et al. [1] and refer to averages over 25 runs carried out using a Pentium(R) Dual 1.80 GHz processor with 2 GB RAM. Results for MA, HGA, and GA are from Nagata and Soler [26]. In the latter, the authors report results for their GA with population sizes of 100 and 300, referred to here by GA-100 and GA-300, respectively. The HGA and GA results are averages of 100 runs using a Pentium 1.6 GHz processor and a Xeon 2.93 GHz processor, respectively, while MA results are for 20 runs using a Pentium 1.7 GHz processor. As for our DR algorithm we now provide the actual machine times obtained using the same Intel Core Duo 2.66 GHz processor with 3 GB RAM.

		Percentage Excess over Optimal										Running Time (seconds)									
Number of Runs		5	5	25	25	25	25	20	100	100	100	5	5	25	25	25	25	20	100	100	100
Instance	*n*	DR Best Run	DR Average	MIMM ACO	MM ACO	EO	CGAS	MA	HGA	GA 100	GA 300	DR Best Run	DR Average	MIMM ACO	MM ACO	EO	CGAS	MA	HGA	GA 100	GA 300
br17	17	**0.0000**	**0.0000**					**0.00**	**0.00**	**0.0000**	**0.0000**	0.00	0.00					0.05	1.9	0.00	0.00
ft53	53	**0.0000**	0.9906	**0.00**	0.22	**0.00**	0.35	**0.00**	**0.00**	0.0060	**0.0000**	0.08	0.18	3.53	3.17	3.85	6.78	0.20	41.2	0.03	0.09
ft70	70	**0.0000**	0.3336	**0.00**	1.71	**0.00**	**0.00**	0.03	**0.00**	0.0010	**0.0000**	0.28	0.51	9.85	10.15	8.93	15.32	0.86	83.6	0.04	0.14
ftv100	101	**0.0000**	0.2685					**0.00**		0.0030	**0.0000**	1.67	0.44					0.40		0.06	0.29
ftv110	111	**0.0000**	0.1839					0.02		0.0070	**0.0000**	1.00	0.79					0.98		0.08	0.36
ftv120	121	**0.0000**	0.0554					0.14		0.0160	**0.0000**	0.50	1.20					2.00		0.1	0.37
ftv130	131	**0.0000**	0.1560					0.01		0.0100	**0.0000**	0.23	1.23					1.30		0.11	0.42
ftv140	141	**0.0000**	0.2562					0.08		0.0120	**0.0000**	1.83	3.35					2.11		0.12	0.53
ftv150	151	**0.0000**	1.9609					0.01		0.0070	**0.0000**	7.43	3.71					1.54		0.15	0.56
ftv160	161	**0.0000**	1.4536					0.02		0.0010	**0.0000**	1.55	2.10					2.04		0.17	0.65
ftv170	171	**0.0000**	0.7042	0.05	0.25	0.28	**0.00**	0.05	0.02	0.0020	**0.0000**	4.32	4.73	108.28	96.73	103.27	128.76	2.78	68.3	0.19	0.69
ftv33	34	**0.0000**	0.8243	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**		**0.0000**	**0.0000**	0.00	0.03	6.12	9.75	4.78	28.73	0.05		0.01	0.06
ftv35	36	**0.0000**	0.0815	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.0000**	**0.0000**	0.02	0.04	5.35	15.37	7.35	21.35	0.08	15.6	0.02	0.05
ftv38	39	**0.0000**	0.0784	**0.00**	**0.00**	**0.00**	**0.00**	0.10		**0.0000**	**0.0000**	0.01	0.03	8.64	10.96	7.83	29.79	0.26		0.02	0.08
ftv44	45	**0.0000**	0.3472	**0.00**	**0.00**	**0.00**	**0.00**	0.44		0.2110	0.0372	0.01	0.07	9.37	12.35	8.21	37.63	0.36		0.02	0.07
ftv47	48	**0.0000**	0.1351	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.0000**	**0.0000**	0.02	0.05	7.52	10.08	9.37	29.7	0.13	90.3	0.02	0.07
ftv55	56	**0.0000**	**0.0000**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.0000**	**0.0000**	0.03	0.05	6.38	18.63	5.06	18.41	0.16	125.6	0.03	0.1
ftv64	65	**0.0000**	0.1631	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.0000**	**0.0000**	0.02	0.04	15.37	27.65	16.42	29.25	0.24	101.2	0.04	0.17
ftv70	71	**0.0000**	0.6462	0.03	5.78	0.72	0.75	0.01	**0.00**	**0.0000**	**0.0000**	0.05	0.24	64.53	61.25	32.26	69.54	0.38	43.8	0.04	0.19
ftv90	91	**0.0000**	**0.0000**					**0.00**		0.0060	**0.0000**	0.17	0.30					0.28		0.05	0.15
kro124p	100	**0.0000**	0.5951	**0.00**	1.64	0.35	**0.00**	0.01	**0.00**	0.0010	**0.0000**	1.26	1.52	33.25	54.21	20.86	78.52	0.74	28.9	0.08	0.33
p43	43	**0.0000**	0.0071	**0.00**	0.08	0.13	**0.00**	0.01	**0.00**	0.0010	**0.0000**	0.03	0.07	8.35	9.38	5.47	7.53	0.35	4.2	0.02	0.1
rbg323	323	**0.0000**	0.0452	**0.00**	1.30	0.06	0.13	**0.00**	**0.00**	**0.0000**	**0.0000**	7.60	12.73	0.01	96.75	87.12	103.28	0.07	110.4	0.96	4.25
rbg358	358	**0.0000**	0.0344	**0.00**	0.75	**0.00**	0.35	**0.00**	**0.00**	**0.0000**	**0.0000**	9.78	23.42	0.01	75.37	69.65	96.49	0.08	58.4	1.32	5.63
rbg403	403	**0.0000**	**0.0000**	**0.00**	1.35	**0.00**	0.31	**0.00**	**0.00**	**0.0000**	**0.0000**	2.93	10.60	0.01	104.39	85.32	147.83	0.08	33.1	1.61	6.63
rbg443	443	**0.0000**	0.0294	**0.00**	1.73	**0.00**	**0.00**	**0.00**	**0.00**	**0.0000**	**0.0000**	0.90	4.44	0.01	90.65	76.14	143.76	0.09	144.2	1.7	6.74
ry48p	48	**0.0000**	0.6130	**0.00**	**0.00**	**0.00**	**0.00**	0.03	**0.00**	**0.0000**	**0.0000**	0.27	0.14	7.83	7.97	5.45	12.35	0.32	53.4	0.02	0.07
Average 16 Instances		0.0000	0.2737	0.01	0.99	0.10	0.13	0.01	0.00	0.0007	0.0000	1.72	3.67	18.02	45.45	35.77	60.59	0.41	62.76	0.38	1.58
Average 18 Instances		0.0000	0.3127	0.00	0.82	0.09	0.11	0.04	—	0.0123	0.0021	1.53	3.27	16.36	39.71	30.96	55.83	0.40	—	0.34	1.41
Average 27 Instances		0.0000	0.3690	—	—	—	—	0.04	—	0.0105	0.0014	1.56	2.67	—	—	—	—	0.66	—	0.26	1.07
Optimal 16 Instances		16	3	13	5	10	10	10	13	11	16										
Optimal 18 Instances		18	2	16	8	13	13	10	—	12	17										
Optimal 27 Instances		27	4	—	—	—	—	13	—	13	26										

Table 2: Comparative analysis with metaheuristic algorithms.

We begin by addressing the MM-ACO, MIMM-ACO, EO and CGAS algorithms as they are all tested on the same instances and computer system. Among these, MIMM-ACO seems to be the best performing algorithm finding the lowest average deviation from optimal and the largest number of optimal solutions while using the smallest computation time. Establishing direct comparisons with the remaining algorithms is not clear cut since the former are tested on a smaller set of instances. For example, the average deviation from optimal of MM-ACO can exceed 5% for some instances as in the case of ftv70. Even though the testbed used by HGA differs in a few instances and the results for this latter algorithm are averages over 100 runs as opposed to 25 runs used by the former algorithms, the performance of HGA appears to be on a par with that of MIMM-ACO. On the other hand, MA and GA are substantially faster in achieving solutions of similar quality to those found by the best performing of these algorithms, with GA obtaining lower percentage excess over optimal and a greater number of optimal solutions when the population size is increased from 100 to 300. However, considering that the GA algorithm runs on a more powerful computer than MA it is also apparent that GA with a population of 300 solutions has higher running times than MA.

In order to appropriately compare the potential advantage of the metaheuristic approaches analyzed here relative to the local search methods examined in the previous section, a number of preliminary remarks should be made. As it can be seen from Tables 1 and 2, the computational studies on local search algorithms for the ATSP involve instances of size slightly over 1000 cities while those on the metaheuristic algorithms only report results for a subset of these instances containing less than 450 cities. An interesting observation that can be derived from the results in Table 1 is that the instances overlooked by the metaheuristic algorithms comprise those where iKP had more trouble in finding its best solutions, which seems to suggest that those instances are particularly challenging. Within the range of the problem sizes tested by the metaheuristic algorithms those instances include dc126, dc176, and td316.10, on which iKP required over 25 minutes, as opposed to less than 3 minutes on average required by this method to find a 0.57% optimality gap on the instances in Table 2. A similar analysis on the instances larger than 500 cities (also omitted in the present tests) shows that these instances impose a significant computational burden on the algorithms tested in Table 1. Specifically, for these larger instances the DR, iKP, and H-LK algorithms required on average around 20 minutes, 1 hour, and 4 hours respectively, to find solutions that deviate from optimality by less than 1% on average. By developing sophisticated search strategies such as those underscored by the metaheuristics analyzed here one would aim to challenge state-of-the-art local search approaches in solving larger and more difficult

ATSP instances. Therefore, the absence of results for these more ambitious algorithm designs on the complete testbed of Table 1 somewhat limits the scope of our computational analysis. However, it is clear that our DR local search algorithm competes favorably with these advanced metaheuristic algorithms. Although not shown in our table, the MA [3] and GA [26] original references report high success rates in finding optimal solutions in the respective 20 runs and 100 runs performed by these algorithms on each test instance. By contrast, our DR algorithm succeeds in finding the optimal solution for all test instances while executing only 5 runs. Taking into account the differences between processors, GA and DR exhibit similar running times while MA appears to be slightly faster. MIMM-ACO, MM-ACO, EO, CGAS, and HGA show high variability of running times across problems and are relatively slower algorithms, though still displaying high levels of effectiveness on their reduced testbed.

4. Conclusion

We report a comparative study of two major neighborhood structures for the ATSP, the KP specialized variant of the LK procedure suggested by Kanellakis and Papadimitriou [22], and the DR ejection chain method proposed in Glover [13]. For a meaningful analysis, we report results for a doubly-rooted ejection chain algorithm that follows the basic design described in [13] and establish comparisons with the implementations reported in Cirasella et al. [6]. Given the considerable advances in local search algorithms for the STSP relative to those available for the ATSP and the fact that any ATSP can be transformed into an equivalent STSP, attempts have been made to solve ATSPs using algorithms originally designed for the STSP. In order to include this type of solution approach in our analysis, we also compared our DR local search algorithm with the currently most effective LK variant for the STSP due to Helsgaun [16], which for the sake of achieving higher performance in solving ATSP instances has been further enhanced with an adaptive iterated procedure.

Computational testing carried out on a standard ATSP test bank demonstrates that our DR method significantly outperforms the enhanced LK extension, establishing the merit of our contribution relative to both the original KP publication and the currently most advanced extension of this work for the ATSP. Not only does the DR approach dominate the local search instance of the LK method with respect to solution quality, but it generates solutions whose quality matches that obtained by the advanced Iterated LK procedures, while requiring significantly less computation time. Moreover, when comparisons are made with respect to solutions found by the best run on each instance

as opposed to the average of those runs, our DR algorithm finds optimal solutions in a fraction of the time required by other algorithms to find solutions that are often of inferior quality.

It is appropriate to emphasize that our DR method allows for a relatively straightforward algorithmic implementation. This contrasts with recent studies of other algorithms, which rely heavily on metaheuristic search guidance to achieve higher performance on this more difficult asymmetric variant of the TSP. However, a comparative analysis with the latest and most advanced of these methods, embodying genetic algorithms, ant colony optimization and hybrids of these methods with local search, shows that our DR local search algorithm is highly effective compared with the best of these methods. While executing no more than five runs on each instance our DR algorithm finds all optimal solutions in less than two seconds on average.

The promising performance of our doubly-rooted ejection chain method in solving hard ATSP instances suggests the merit of introducing more advanced metaheuristic guidance into our approach. For example, the recent findings that population-based methods do a good job of identifying arcs belonging to optimal solutions invite consideration of embedding our algorithm within a population-based framework. As originally suggested in Rego and Glover [31], approaches using adaptive memory constructs such as featured in scatter search and path relinking (e.g., [33, 38, 40]) appear particularly worth exploring in this regard.

Acknowledgements

This work was partially supported by FEDER Funds through the “Programa Operacional Factores de Competitividade - COMPETE” program and by National Funds through FCT “Fundação para a Ciência e a Tecnologia” under the project: PTDC/EGE-GES/121660/2010.

References

[1] J. Bai, G.-K. Yang, Y.-W. Chen, L.-S. Hu, and C.-C. Pan, A Model Induced Max-Min Ant Colony Optimization for Asymmetric Traveling Salesman Problem, Applied Soft Computing 13 (2013), 1365-1375.

[2] O. Bräysy, E. Martínez, Y. Nagata, and D. Soler, The Mixed Capacitated General Routing Problem with Turn Penalties, Expert Systems with Applications 30 (2011), 12954-12966.

[3] L. Buriol, P.M. França, and P. Moscato, A New Memetic Algorithm for the Asymmetric Traveling Salesman Problem, Journal of Heuristics 10 (2004), 483-506.

[4] Y.-W. Chen, Y.-J. Zhu, G.-K. Yang, and Y.-Z. Lu, Improved Extremal Optimization for the Asymmetric Traveling Salesman Problem, Physica A: Statistical Mechanics and its Applications 390 (2011), 4459-4465.

[5] F.F. Choobineh, E. Mohebbi, and H. Khoo, A Multi-objective Tabu Search for a Single-Machine Scheduling Problem with Sequence-dependent Setup Times, European Journal of Operational Research 175 (2006), 318-337.

[6] J. Cirasella, D.S. Johnson, L.A. McGeoch, and W. Zhang, The Asymmetric Traveling Salesman Problem: Algorithms, Instance Generators and Tests, Proc Algorithm Engineering and Experimentation Third International Workshop, ALENEX 2001, Lecture Notes in Computer Science, Vol. 2153, Springer, 2001, pp. 32-59.

[7] G. Dong, W.W. Guo, and K. Tickle, Solving the Traveling Salesman Problem using Cooperative Genetic Ant Systems, Expert Systems with Applications 39 (2012), 5006-5011.

[8] B. Funke, T. Grünert, and S. Irnich, A Note on Single Alternating Cycle Neighborhoods for the TSP, Journal of Heuristics 11 (2005), 135-146.

[9] D. Gamboa, C. Rego, and F. Glover, Data Structures and Ejection Chains for Solving Large-Scale Traveling Salesman Problems, European Journal of Operational Research 160 (2005), 154-171.

[10] D. Gamboa, C. Rego, and F. Glover, Implementation Analysis of Efficient Heuristic Algorithms for the Traveling Salesman Problem, Computers & Operations Research 33 (2006), 1154-1172.

[11] M. Gendreau, J. Nossack, and E. Pesch, Mathematical Formulations for a 1-Full-Truckload Pickup-and-Delivery Problem, European Journal of Operational Research 242 (2015), 1008-1016.

[12] F. Glover, "New Ejection Chain and Alternating Path Methods for Traveling Salesman Problems," Computer Science and Operations Research: New Developments in Their Interfaces, O. Balci, R. Sharda and S.A. Zenios (Editors), Pergamon, Oxford, 1992, pp. 449-509.

[13] F. Glover, Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems, Discrete Applied Mathematics 65 (1996), 223-253.

[14] F. Glover, Finding a Best Traveling Salesman 4-Opt Move in the Same Time as a Best 2-Opt Move, Journal of Heuristics 2 (1996), 169-179.

[15] G. Gutin and A. Yeo, Upper Bounds on ATSP Neighborhood Size, Discrete Applied Mathematics 129 (2003), 533-538.

[16] K. Helsgaun, An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic, European Journal of Operational Research 126 (2000), 106-130.

[17] D.S. Johnson, Notes on Finding Best 2-Bridge Move, Notes Provided to the Authors, 2000.

[18] D.S. Johnson, G. Gutin, L. A. McGeoch, A. Yeo, W. Zhang, and A. Zverovitch, "Experimental Analysis of Heuristics for the ATSP," The Traveling Salesman Problem and Its Variations, G. Gutin and A. Punnen (Editors), Kluwer, Boston, 2002, pp. 445-487.

[19] D.S. Johnson and L.A. McGeoch, "Experimental Analysis of Heuristics for the STSP," The Traveling Salesman Problem and Its Variations, G. Gutin and A. Punnen (Editors), Kluwer, Boston, 2002, pp. 369-443.

[20] D.S. Johnson, L.A. McGeoch, F. Glover, and C. Rego, 8th DIMACS Implementation Challenge: The Traveling Salesman Problem, 2000, webpage. http://dimacs.rutgers.edu/Challenges/TSP.

[21] H. Jula, M. Dessouky, P. Ioannou, and A. Chassiakos, Container Movement by Trucks in Metropolitan Networks: Modeling and Optimization, Transportation Research Part E: Logistics and Transportation Review 41 (2005), 235-259.

[22] P.C. Kanellakis and C.H. Papadimitriou, Local Search for the Asymmetric Traveling Salesman Problem, Operations Research 28 (1980), 1086–1099.

[23] G. Laporte, Modeling and Solving Several Classes of Arc Routing Problems as Traveling Salesman Problems, Computers & Operations Research 24 (1997), 1057-1061.

[24] G. Laporte, A Concise Guide to the Traveling Salesman Problem, Journal of the Operational Research Society 61 (2010), 35-40.
[25] S. Lin and B. Kernighan, An Effective Heuristic Algorithm for the Traveling Salesman Problem, Operations Research 21 (1973), 498-516.
[26] Y. Nagata and D. Soler, A New Genetic Algorithm for the Asymmetric Traveling Salesman Problem, Expert Systems with Applications 39 (2012), 8947-8953.
[27] E. Pesch and F. Glover, TSP Ejection Chains, Discrete Applied Mathematics 76 (1997), 165-181.
[28] A. Punnen and S. Kabadi, Domination Analysis of Some Heuristics for the Asymmetric Traveling Salesman Problem, Discrete Applied Mathematics 119 (2002), 117-128.
[29] C. Rego, Relaxed Tours and Path Ejections for the Traveling Salesman Problem, European Journal of Operational Research 106 (1998), 522-538.
[30] C. Rego, D. Gamboa, F. Glover, and C. Osterman, Traveling Salesman Problem Heuristics: Leading Methods, Implementations and Latest Advances, European Journal of Operational Research 211 (2011), 427-441.
[31] C. Rego and F. Glover, "Local Search and Metaheuristics," The Traveling Salesman Problem and Its Variations, G. Gutin and A. Punnen (Editors), Kluwer, Boston, 2002, pp. 309-368.
[32] G. Reinelt, TSPLIB - A Traveling Salesman Problem Library, ORSA Journal on Computing 3 (1991), 376-384.
[33] M.G.C. Resende, C.C. Ribeiro, F. Glover, and R. Martí, "Scatter Search and Path Relinking: Fundamentals, Advances and Applications," Handbook of Metaheuristics: International Series in Operations Research & Management Science, M. Gendreau and J.-Y. Potvin (Editors), Springer, New York, 2010, Vol. 146, Chapter 4, pp. 87-107.
[34] R. Roberti and P. Toth, Models and Algorithms for the Asymmetric Traveling Salesman Problem, EURO Journal on Transportation and Logistics 1 (2012), 113–133.
[35] A. Rodríguez and R. Ruíz, The Effect of the Asymmetry of Road Transportation Networks on the Traveling Salesman Problem, Computers and Operations Research 39 (2012), 1566-1576.
[36] D. Soler, E. Martínez, and J.C. Micó, A Transformation for the Mixed General Routing Problem with Turn Penalties, Journal of the Operational Research Society 59 (2008), 540-547.
[37] T. Stützle and H.H. Hoos, MAX-MIN Ant System, Future Generation Computer Systems 16 (2000), 889-914.
[38] Y. Wang, Z. Lu, F. Glover, and J.-K. Hao, Path Relinking for Unconstrained Binary Quadratic Programming, European Journal of Operational Research 223 (2012), 595-604.
[39] L.N. Xing, Y.W. Chen, K.W. Yang, F. Hou, X.S. Shen, and H.P. Cai, A Hybrid Approach Combining an Improved Genetic Algorithm and Optimization Strategies for the Asymmetric Traveling Salesman Problem, Engineering Applications of Artificial Intelligence 21 (2008), 1370-1380.
[40] M. Yagiura, T. Ibaraki, and F. Glover, A Path Relinking Approach with Ejection Chains for the Generalized Assignment Problem, European Journal of Operational Research 169 (2006), 548-569.