Tolerance-based greedy algorithms for the traveling salesman problem

Diptesh Ghosh⋆, Boris Goldengorin⋆⋆, Gregory Gutin⋆⋆⋆, and Gerold Jäger†

Abstract. In this paper we introduce three greedy algorithms for the traveling salesman problem. These algorithms are unique in that they use arc tolerances, rather than arc weights, to decide whether or not to include an arc in a solution. We report extensive computational experiments on benchmark instances that clearly demonstrate that our tolerance-based algorithms outperform their weight-based counterpart. Along with other papers dealing with the Assignment Problem, this paper indicates that the potential for using tolerance-based algorithms for various optimization problems is high and motivates further investigation of the approach. We recommend one of our algorithms as a significantly better alternative to the weight-based greedy, which is often used to produce initial TSP tours.

1   Introduction

In this paper we propose several algorithms for the traveling salesman problem (TSP). In a TSP instance of size (or dimension) n, we are given a weighted complete digraph $D = (V, A, C)$ where V is the set of n vertices, A the set of arcs between vertices in V, and $C = [c(i, j)]$ is the $n \times n$-matrix of arc weights, and we are required to find a Hamilton cycle (called a *tour*) such that the sum of the weights of the arcs in the tour is as small as possible. A TSP instance is called a *symmetric TSP* (STSP) instance if for each pair of vertices i and j, $c(i, j) = c(j, i)$; and an *asymmetric TSP* (ATSP) instance otherwise. Also, a TSP instance is defined by the weight matrix C.

Most algorithms for solving the TSP make use of the arc weights to decide whether or not to include an arc in the solution that they finally output. For example, the weight-based greedy algorithm and its variations are popular heuristics to produce initial tours for local search and other improvement heuristics (see, e.g., [3]). However, as pointed out in [8] and [18], arc *tolerances* are better indicators than arc weights for generating good tours.

⋆ P&QM Area, IIM Ahmedabad, India.
⋆⋆ Faculty of Economic Sciences, University of Groningen, The Netherlands and Department of Applied Mathematics, Khmelnitsky National University, Ukraine.
⋆⋆⋆ Department of Computer Science, Royal Holloway University of London, Egham, Surrey TW20 OEX, UK and Department of Computer Science, University of Haifa, Israel.
† Computer Science Institute, University of Halle-Wittenberg, Germany.

The arc tolerance (see e.g., [6], [7], [16]) is the maximum amount by which the weight of an arc that is in (not in) an optimal tour can be increased (respectively, decreased) while keeping other arc weights unchanged for the tour to remain optimal. Among currently known algorithms for the TSP, only Helsgaun's version of the Lin-Kernighan heuristic for the STSP (see [13]) explicitly applies tolerances in algorithm design. Implicit applications of tolerances in algorithm design are found in the Vogel's method for the Transportation Problem and in the MAX-REGRET heuristic for solving the Three-Index Assignment Problem (see [1]).

Greedy algorithms for TSP are often used in computational practice (cf. [14]) to produce tours that are further improved by local search or other algorithms. To the best of our knowledge the concept of tolerances has not been applied to the design of greedy algorithms for the TSP prior to this paper. Our aim is to initiate and motivate research on the use of tolerances for decision making in fast TSP heuristics. The algorithms that we propose might therefore not be the best of breed, but they clearly show superiority of tolerance-based algorithms over their arc-weights counterparts. We also show the superiority of tolerance based algorithms using domination analysis (see [9], [10]) for our simplest tolerance-based R-R-GREEDY heuristic applied to the Assignment Problem (AP). We prove that the domination number of R-R-GREEDY for the AP is equal to 2^{n-1}, while it is just 1 for the weight-based greedy (W-GREEDY). This means that while an assignment produced by R-R-GREEDY is *always* at least as good as 2^{n-1} assignments, there is a family of $n \times n$ AP-instances for each n for which W-GREEDY finds an assignment, which is *unique* worst possible, and, thus, in this respect W-GREEDY is no better than a heuristic producing a random solution. We believe that our results indicate high potential of tolerance-based algorithms for various optimization problems and motivate further investigation of the approach.

In the next section, we develop concepts that will help us to describe the algorithms that we introduce for the TSP in Section 3. Our tolerance-based greedy algorithms are described in Section 3, where we also prove the above-mentioned domination result. We report computational experience with our greedy algorithms in Section 4. We conclude the paper in Section 5 with a summary of our main contributions and suggestions for future research.

2 Some relevant concepts

2.1 The Relaxed Assignment Problem

The Assignment Problem (AP) is a well-known relaxation of the TSP, which is used more often for the ATSP than for the STSP. Let $D = (V, A, C)$ be a bipartite digraph with bipartition $V = V_1 \cup V_2$, $|V_1| = |V_2| = n$, and such that $A = V_1 \times V_2$. The AP is defined as the problem to find n arcs (s_i, t_i), $1 \leq i \leq n$ of minimum total weight such that $s_i \neq s_j$ and $t_i \neq t_j$ for every $1 \leq i \neq j \leq n$, i.e., the AP is the problem to find a minimum weight perfect matching. Notice that if V_1 and V_2 are two copies of the vertex set of an TSP instance, where the

arc weights of the bipartite directed graph correspond to the arc weights of the TSP and the weight of every arc from one copy of a vertex to the other is set to ∞, then the AP solution can be interpreted as a collection of cycles (called subtours) for the instance.

An integer programming formulation of the AP on an ATSP instance defined on a complete digraph $G = (V, A, C)$ (where $|V| = n$, and $C = [c(i, j)]$) using variables x_{ij}, $i, j \in V$ such that $x_{ij} = 1$ when (i, j) is included in the solution and 0 otherwise, is given below.

$$\text{Minimize} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c(i, j) x_{ij}$$

$$\text{Subject to} \quad \sum_{j=1}^{n} x_{ij} = 1 \quad i \in \{1, \dots, n\} \tag{1}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad j \in \{1, \dots, n\} \tag{2}$$

$$x_{ij} \in \{0, 1\} \quad i, j \in \{1, \dots, n\}$$

The Relaxed Assignment Problem (RAP) is a relaxation of the AP in which constraint set (2) is removed from the earlier formulation. Thus, instead of an one-to-one matching in case of the AP, in the RAP the first copy of V maps into the second copy of V. Note that a feasible solution to the AP is also a feasible solution to the RAP, but not vice versa. Also note that a solution to the RAP may not consist exclusively of cycles.

2.2 Determining tolerances for AP and RAP

Extending the informal definition of tolerances in the introductory section, the upper (lower) tolerance of an arc that is included in (excluded from) an optimal solution to the AP is the maximum amount by which the weight of the arc can be increased (respectively, reduced) while keeping other arc weights unchanged, such that the current optimal solution to the AP remains optimal. Tolerances for arcs of the RAP can be defined analogously.

Computing arc tolerances for the AP involves revising the arc weight to a suitably high value if the arc is a part of the optimal solution, and a suitably low value if it is not (see [6]), and re-solving the AP. The AP can be solved in $\mathcal{O}(n^3)$ time, and using a shortest path based approach, all arc tolerances can also be computed in $\mathcal{O}(n^3)$ time (see [19]).

Computing arc tolerances for the RAP, on the other hand, is a more tractable problem. An optimal solution to the RAP can be characterized as a collection of arcs, one from each vertex in the graph, such that the weight of the arc is the smallest among those of all arcs from that vertex. Therefore, for each arc that belongs to an optimal solution to the RAP, its upper tolerance is the excess of the weight of the second smallest weight out-arc from the same vertex over the weight of that arc. If the arc is not in an optimal solution, then its lower

tolerance would be the excess of the weight of the arc over that of the smallest weight out-arc from the same vertex. Obtaining all tolerances therefore requires finding the weights of the two least weight entries in each row of the cost matrix, and then performing a simple subtraction operation once for each arc. Both jobs can be achieved in $\mathcal{O}(n^2)$ time so that the overall complexity of determining all arc tolerances for the RAP is $\mathcal{O}(n^2)$ time.

2.3   The contraction procedure and a greedy algorithm

The (path) contraction procedure (see, e.g., [4]) is a method of updating a digraph once a directed path is removed from it and replaced by a single vertex. Consider a digraph $D = (V, A, C)$ with $C = [c(i, j)]$ and a directed path $P = v_1 v_2 \cdots v_k$ in it. The contraction procedure for marking the path (and replacing it by a vertex p) replaces D by a digraph D_p. The vertex set of D_p is $V_p = V \cup \{p\} - \{v_1, \ldots, v_k\}$. The arc set of D_p includes all arcs (i, j) from D where $i, j \notin P$. In addition for all vertices i in V_p except p, it introduces and includes arcs (i, p) with weight $c(i, v_1)$ and (p, i) with weight $c(v_k, i)$ (where c, i, v_1, and v_k are defined for digraph D). In this paper we only need a special case of the path contraction procedure, namely contracting only a single arc (say a) from a digraph D. We use a shorthand notation $CP(a, D)$ for this procedure.

Given the contraction procedure, a generic greedy algorithm can be defined as follows:

A generic greedy algorithm
Input: A weighted complete digraph $D = (V, A, C)$.
Output: A tour T.
Step 1: $G \leftarrow D$, $T \leftarrow \emptyset$.
Step 2: While G consists of at least three vertices, using a suitable myopic procedure, choose an arc (say $a = (u, v)$), that does not create a cycle, to include in the tour.
(For example, if arcs (1,2) and (2,3) are already contracted, the contraction of (3,1) would create a cycle.)
Set $T \leftarrow T \cup \{a\}$, $G \leftarrow CP(a, G)$.
Step 3: Set $T \leftarrow T \cup \{(v_1, v_2), (v_2, v_1)\}$ and output T.

This algorithm is generic since the myopic arc selection procedure used in Step 2 has not been defined. Typically greedy algorithms employ myopic procedures based on arc weights, choosing the least weight arc as the one to contract. Therefore, as a benchmark for tolerance-based algorithms that we present in the next section, we define the following variant.

W-GREEDY algorithm: At each iteration of the generic greedy algorithm, in Step 2, the myopic procedure chooses the least weight arc. This arc is chosen for contraction (i.e., inclusion in the tour).

3 Tolerance-based greedy algorithms

Since exploratory computational experiments (see, e.g., [18]) show that, given an optimal AP solution to an TSP instance, the 'probability' of the arc with the largest upper tolerance for the AP solution being in an optimal TSP solution is much higher than the 'probability' of the smallest weight arc being in an optimal TSP solution, it is interesting to create myopic procedures for the generic greedy algorithm developed in Section 2.3 which use tolerances instead of arc weights to choose arcs. In this section, we introduce the following three variants of such myopic procedures, leading to three greedy algorithms.

R-R-GREEDY algorithm: At Step 2 of each iteration of the generic greedy algorithm, the myopic procedure generates an optimal RAP solution on the digraph. Then the upper tolerances of each arc included in the solution are generated. The arc in the optimal RAP solution with the largest upper tolerance is chosen for contraction (i.e., inclusion in the tour).

A-R-GREEDY algorithm: At each iteration, the myopic procedure generates an optimal AP solution and an optimal RAP solution on the digraph. For each arc in the AP solution *and* in the RAP solution, the upper tolerance (w.r.t. the RAP) is computed, and for each arc in the AP solution but *not* in the RAP solution, the lower tolerance (w.r.t. the RAP) is computed, and multiplied with -1. The values thus obtained are sorted, and the arc with the largest value is chosen for contraction.

The relaxation of constraint set (2) in the formulation of AP to generate RAP was arbitrary. One could easily come up with another relaxation of the AP (let us call it RAP1) in which constraint set (1) is relaxed instead of the set (2). The third algorithm implements a myopic procedure that uses both the RAP and RAP1 relaxations.

A-RC-GREEDY algorithm: Optimal solutions are generated for AP as well as for RAP and RAP1. The myopic procedure described in the A-R-GREEDY algorithm is carried out twice, once with the optimal solutions to AP and RAP, and the second time with the optimal solutions to AP and RAP1. In the second case, the tolerances are computed with respect to the RAP1 relaxation. Of the two candidates that emerge from the two procedures, the one which has a larger value is chosen for contraction.

Note that for A-R-GREEDY and A-RC-GREEDY we only approximate the upper tolerances for the AP. The reason is, that in practice, solving an AP in $\mathcal{O}(n^3)$ time by the Hungarian algorithm and then computing approximate tolerances in $\mathcal{O}(n^2)$ time is much faster than using the Hungarian algorithm and then Volgenant's method (see [19]) for exactly computing the tolerances in $\mathcal{O}(n^3)$, even though both methods have an overall $\mathcal{O}(n^3)$ time complexity.

The greedy algorithms described above can be speeded up considerably using book-keeping techniques. For example, in R-R-GREEDY, if in an iteration, the end vertex of the contracted arc does not contain a smallest or a second smallest

weight arc from any of the vertices, then in the next iteration, both the RAP solution and the upper tolerances remain unchanged. Even otherwise, the changes in the RAP solution and upper tolerance at the next iteration involve only those vertices from which the smallest or second smallest weight arcs were directed to the end vertex of the contracted arc in the previous iteration. Furthermore, in the A-R-GREEDY and A-RC-GREEDY algorithms, if the arc contracted does not belong to a subtour with two arcs only, the optimal AP solution before and after the contraction operation differ only by the arc contracted.

Our extensive computational experiments with the W-GREEDY and R-R-GREEDY applied to a wide set of the AP instances with $n \geq 100$ (see [2] for a description of the instances) show that the quality of R-R-GREEDY solutions is at least 10 times better than the quality of W-GREEDY solutions and these results are further supported by domination analysis (for all undefined definitions in domination analysis and graph theory we refer to [9], [10]). The domination number of a heuristic $\mathcal{H}$ for a combinatorial optimization problem $\mathcal{P}$ is the maximum number, taken over all instances of size n, of solutions that are not better than the solution found by $\mathcal{H}$ for any instance of size n. The domination number of W-GREEDY for the AP equals 1 [9], i.e., for every n there are instances of AP for which W-GREEDY finds the *unique* worst solution.

The following theorem indicates that the domination number of R-R-GREEDY for the AP is exponential.

Let $K_{n,n}$ be a complete bipartite graph with partite sets $V = \{1, 2, \ldots, n\}$ and $V' = \{1', 2', \ldots, n'\}$. The weight of an edge ij' is denoted by c_{ij}. Recall that the *Assignment Problem (AP)* is the problem of finding an assignment (i.e., a perfect matching) of $K_{n,n}$ of total minimum weight.

Theorem 1. *For the AP, R-R-GREEDY has domination number 2^{n-1}.*

Proof. Let AP-R-R-GREEDY be the algorithm R-R-GREEDY applied to the AP. If — like for the TSP — all diagonal elements of the cost matrix are ∞, the algorithm is the same as R-R-GREEDY except that in the generic greedy algorithm the myopic procedure may also choose an arc that creates a cycle.

In the general case the algorithm works as follows. Set $W = W' = M = \emptyset$. While $V \neq W$ do the following: For each $i \in V - W$, compute two lightest edges ij' and ik', where $j', k' \in V' - W'$, and the difference $\Delta_i = |c_{ij} - c_{ik}|$. For $i \in V - W$ with maximum Δ_i choose the lightest edge ij', where $j' \in V' - W'$, and add ij' to M, i to W and j' to W'.

Consider an instance $\mathcal{I}$ of AP on $K_{n,n}$ such that the number of assignments that are not better than the assignment found by AP-R-R-GREEDY equals the domination number of AP-R-R-GREEDY. We may assume that AP-R-R-GREEDY constructs the assignment $M = \{11', 22', \ldots, nn'\}$ and the edges in M are chosen such that ii' is picked before jj' if and only if $i < j$. Then $11'$ and $1j'$ for some $j \neq 1$ are lightest edges of $K_{n,n}$ subject to the condition that the start vertex is 1, and it holds $c_{11} \leq c_{1j}$. Since adding a constant to the weight of every edge with the first vertex 1 will not change the solution found by AP-R-R-GREEDY, we may assume that $c_{1j} = 0$ and $c_{11} = \Delta_1 \leq 0$. Since

$c_{1k} \geq 0$ for each $k \neq 1$ and taking larger c_{1k} with $k \neq 1$ may only increase the number of assignments of weight at least $c(M) := \sum_{i=1}^{n} c_{i,i'}$, we may assume that $c_{1k} = 0$ for each $k \neq 1$.

Let $22'$ and $2s'$, $s > 2$, be the lightest edges with the first vertex 2 with the possible exception of $21'$. As above we may assume that $c_{22} = \Delta_2 \leq 0$ and $c_{2k} = 0$ for each $k > 2$. Since $11'$ was chosen by AP-R-R-GREEDY before $22'$, $c_{21} \geq \Delta_1 + \Delta_2$ and we may assume that $c_{21} = \Delta_1 + \Delta_2$. Analogously, for $t \geq 3$, we may assume that $c_{ts} = 0$ for each $s > t$ and $c_{tk} = \Delta_k + \Delta_{k+1} + \ldots + \Delta_t$ for each $k < t$. Notice that $c_{tt} = \Delta_t \leq 0$ for $t < n$. (However, we have no restriction on the sign of $c_{nn} = \Delta_n$ as we do not have any choice when picking the edge nn'. We may assume that $\Delta_n \geq 0$ to decrease the number of assignments of weight larger or equal to $c(M)$.)

Denote the number of assignments of weight at least $c(M) = \sum_{i=1}^{n} \Delta_i$ by $g(\Delta_1, \Delta_2, \ldots, \Delta_n)$. By the arguments above the domination number $d(n)$ of AP-R-R-GREEDY equals

$$\min\{g(\Delta_1, \Delta_2, \ldots, \Delta_n) : \ \Delta_1 \leq \Delta_2 \leq \cdots \leq \Delta_{n-1} \leq 0\}.$$

Let $\mathrm{Op}(i, p)$ denotes an operation that replaces in M the edges $\{ii', (i+1)(i+1)', \ldots, (i+p)(i+p)'\}$ by the edges $\{i(i+1)', (i+1)(i+2)', \ldots, (i+p-1)(i+p)', (i+p)i'\}$. Notice that $\mathrm{Op}(i, p)$ preserves the weight of the assignment. Consider the following procedure. It starts from $i := 1$. It chooses an arbitrary integer p with $0 \leq p \leq n - i$, performs $\mathrm{Op}(i, p)$, sets $i := i + p + 1$ and continues this loop if $i \leq n$.

Let $f(n)$ be the number of all possible assignments that can be obtained by the procedure. Since all these assignments are of the same weight as the assignment $\{11', 22', \ldots, nn'\}$, we conclude that $d(n) \geq f(n)$. Clearly, $f(1) = 1$ and set $f(0) = 1$. To compute $f(n)$ observe that after using $\mathrm{Op}(1, p)$ we will have $f(n - p)$ possible assignments. Thus, for each $n \geq 2$ we have $f(n) = f(n-1) + f(n-2) + \ldots + f(0)$. This implies that $f(n) = 2^{n-1}$ for $n \geq 1$.

To show that $d(n) = f(n)$ construct a complete digraph K_n^* with vertices $\{1, 2, \ldots, n\}$ and with a loop on every vertex. For arbitrary $1 \leq i, j \leq n$, the arc (i, j) of K_n^* corresponds to the edge ij' in $K_{n,n}$ and we set the weight of (i, j) equal c_{ij}. We call every arc (i, j) with $i < j$ *forward* and with $i \geq j$ *backward*. Notice that the weight of every forward arc is 0.

An assignment (i.e., perfect matching) in $K_{n,n}$ corresponds to a *cycle factor* of K_n^*, which is a collection of disjoint cycles (some of them may be loops) that cover all vertices of K_n^*. In particular, the weight of an assignment in $K_{n,n}$ equals the weight of the corresponding cycle factor in K_n^*. Notice that the weight of every forward arc is 0 and, thus, the weight of a cycle factor equals the sum of the weights of its backward arcs. We call a pair $(i, j), (i', j')$ of backward arcs *intersecting* if the intervals $[j, i]$ and $[j', i']$ of real line intersect (one of these intervals may be just a point). Observe that if a cycle factor does not have intersecting backward arcs, then its weight equals $\sum_{i=1}^{n} \Delta_i = c(M)$ and every such cycle factor corresponds to an assignment that can be obtained by the procedure above. Thus, there are exactly $f(n) = 2^{n-1}$ cycle factors without intersecting backward arcs.

Now suppose that a cycle factor F has an intersecting pair $(i, j), (i', j')$ of backward arcs. Thus, there is an integer k such that $k \in [j, i] \cap [j', i']$. By the definition of a cycle factor, $k < n$. Observe that the above arguments imply that $c(F) \leq \sum_{i=1}^{n} \Delta_i + \Delta_k \leq c(M)$. Thus, if we set $\Delta_i < 0$ for each $i < n$, we will obtain that every cycle factor with a pair of intersecting backward arcs has weight smaller than $c(M)$. Observe that a cycle factor with intersecting backward arcs corresponds to an assignment in $K_{n,n}$ that cannot be obtained by the procedure above. Therefore, $d(n) = f(n) = 2^{n-1}$. ∎

In the next section, we compare the three tolerance-based greedy algorithms introduced in this section with each other on benchmark TSP instances using the W-GREEDY algorithm to calibrate the algorithms. Since the performance of the W-GREEDY algorithm has been compared with other well-known algorithms for the TSP (see, e.g., [4]), the next section also provides an indirect comparison with those algorithms.

4   Computational experience

The four greedy algorithms mentioned in the paper were implemented in order to observe their performance on benchmark instances of the TSP. The implementations were done in C under Linux on a GenuineIntel Intel® Xeon$^{\text{TM}}$ 3.2GHz machine with 4 GB RAM. In our implementations we use the Jonker and Volgenant's (see [15]) code for solving the AP.

Out of the four algorithms, only the W-GREEDY algorithm is known in the literature (see the GR algorithm in [4] and [11]). Therefore, we report our computational results using W-GREEDY as a base. Assume that for a particular TSP instance, W-GREEDY finds a tour of length L_W, while another algorithm $\mathcal{A}$ finds a tour of length L_A. Also assume that the length of an optimal tour, or a good lower bound for it, is $L^\star$. Then for that instance we define the solution quality parameter q_A for $\mathcal{A}$ as

$$q_A = \frac{L_A - L^\star}{L_W - L^\star} \times 100$$

As the execution times for many small instances are often very small and therefore inaccurate, we measure the time for the whole class. Assume that for a whole class, W-GREEDY needs time T_W, while another algorithm $\mathcal{A}$ needs time T_A. Then for that class we define the time quality parameter τ_A for $\mathcal{A}$ as

$$\tau_A = \frac{T_A}{T_W} \times 100$$

Clearly, the smaller the average value of q_A and the value of τ_A is, the better is the algorithm. We tested the algorithms on nine classes of instances. Classes 1 through 7 were taken from [4], Class 8 is the class of GYZ instances introduced in [11] for which the domination number of the W-GREEDY algorithm for the

ATSP is 1 (see Theorem 2.1 in [11]) and Class 9 is the amalgamation of several classes of instances from [14]. The exact description of the nine classes is as follows.

Class 1: All asymmetric instances from TSPLIB [17] (26 instances).

Class 2: All symmetric instances from TSPLIB [17] with less than 3000 vertices (99 instances).

Class 3: Asymmetric instances with $c(i, j)$ randomly and uniformly chosen from $\{0, 1, \cdots, 100000\}$ for $i \neq j$. 10 instances are generated for dimensions $100, 200, \cdots, 1000$ and three instances for dimensions $1100, 1200, \cdots, 3000$ (160 instances).

Class 4: Asymmetric instances with $c(i, j)$ randomly and uniformly chosen from $\{0, 1, \cdots, i \cdot j\}$ for $i \neq j$. 10 instances are generated for dimensions $100, 200, \cdots, 1000$ and three instances for dimensions $1100, 1200, \cdots, 3000$ (160 instances).

Class 5: Symmetric instances with $c(i, j)$ randomly and uniformly chosen from $\{0, 1, \cdots, 100000\}$ for $i < j$. 10 instances are generated for dimensions $100, 200, \cdots, 1000$ and three instances for dimensions $1100, 1200, \cdots, 3000$ (160 instances).

Class 6: Symmetric instances with $c(i, j)$ randomly and uniformly chosen from $\{0, 1, \cdots, i \cdot j\}$ for $i < j$. 10 instances are generated for dimensions $100, 200, \cdots, 1000$ and three instances for dimensions $1100, 1200, \cdots, 3000$ (160 instances).

Class 7: Sloped plane instances with given x_i, x_j, y_i, y_j randomly and uniformly chosen from $\{0, 1, \cdots, i \cdot j\}$ for $i \neq j$ and $c(i, j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - \max\{0, y_i - y_j\} + 2 \cdot \max\{0, y_j - y_i\}$ for $i \neq j$. 10 instances are generated for dimensions $100, 200, \cdots, 1000$ and three instances for dimensions $1100, 1200, \cdots, 3000$ (160 instances).

Class 8: GYZ instances (see Theorem 2.1 in [11]) in which the arc weights $c(i, j)$ are defined as

$$c(i,j) = \begin{cases} n^3 & \text{for } i = n, j = 1; \\ in & \text{for } j = i+1, i = 1, 2, \ldots, n-1; \\ n^2 - 1 & \text{for } i = 3, 4, \ldots, n-1; j = 1; \\ n \min\{i, j\} + 1 & \text{otherwise.} \end{cases}$$

One instance is generated for each $n = 5, 10, \cdots, 1000$ (200 instances).

Class 9: There are 12 problem generators from Johnson et al. [14], called *tmat, amat, shop, disc, super, crane, coin, stilt, rtilt, rect, smat,* and *tsmat.* Each of these generators yields 24 instances, 10 of dimensions 100, 10 of dimension 316, three of dimension 1000, and one of dimension 3162 (288 instances).

Note that for Classes 1 and 2 we use as $L^\star$ the known optima (see [17]), for the symmetric and almost-symmetric Classes 5, 6, and 9 the HK (Held-Karp) lower bound ([12]) and for the asymmetric Classes 3, 4, and 7 the AP lower bound.

It is clear from Table 1 that the usual weight-based greedy algorithm is comprehensively outperformed by tolerance-based greedy algorithms in terms of solution quality, although it takes much less execution time than two of the

Table 1. Performance of tolerance-based algorithms

Algorithm	Problem Class	q values mean	q values std. dev.	τ values
R-R GREEDY	1	109.18	160.76	100
	2	125.49	55.18	105.17
	3	32.61	10.95	106.63
	4	2.16	0.99	115.69
	5	35.68	11.18	116.42
	6	3.27	1.07	121.58
	7	5.92	1.9	81.16
	8	19.08	2.07	103.88
	9	64.77	40.52	65.76
A-R GREEDY	1	33.52	23.29	100
	2	69.67	29.63	3301.72
	3	9.37	8.17	131.6
	4	0.19	0.26	147.13
	5	12.38	7.16	4302.68
	6	1.36	0.58	8228.45
	7	2.78	1.65	11581.88
	8	0	0.01	92.53
	9	30.05	29.19	3480.53
A-RC GREEDY	1	22.07	17.16	125
	2	62.31	37.55	3408.62
	3	4.03	5.25	219.34
	4	0.38	0.65	240.22
	5	6.98	4.43	3428.08
	6	2.42	1.15	5075.24
	7	2.61	1.50	12301.37
	8	0	0.01	122.65
	9	25.96	29.56	3555.79

tolerance-based algorithms. It is also clear that A-RC-GREEDY, and to a lesser extent, A-R-GREEDY are greedy algorithms of choice if one desires good-quality solutions. Even the extremely simplistic R-R-GREEDY algorithm generates better quality solutions for all except two classes (Classes 1 and 2) in nearly the same time. This fact is seen most starkly in Classes 4 and 7.

An interesting observation is that AP relaxation based algorithms require very long execution times on average for instances in Classes 7 and 9. For instances in these classes, experiments show that the optimal solutions to the AP relaxation for the digraphs in several iterations have many cycles of length 2, and the arc to be contracted usually comes from one of these cycles. Consequently, in the next step of the algorithm, the AP relaxation needs to be solved again, and the tolerances recalculated, thus leading to long execution times (refer to the discussion on book-keeping techniques in Section 3).

5 Summary and Future Research Directions

In this paper, we examine in detail the idea of using arc tolerances instead of arc weights as a basis for making algorithmic decisions on whether or not to include an arc in an optimal solution. Such methods have only been studied in passing in the literature (see [13]) and deserve more attention. In order to evaluate the usefulness of the concept, three tolerance-based greedy algorithms are proposed (see Section 3) for the traveling salesman problem. Two of these (A-R-GREEDY and A-RC-GREEDY) are based on an AP relaxation of the original problem, while the third one (R-R-GREEDY) is based on a new relaxation of the AP relaxation itself. With the purpose of investigating the usefulness of the relaxed AP (RAP), we made extensive computational experiments with our R-R-GREEDY heuristic applied to the AP (not reported here in detail due to the space limitation) and studied the worst case performance using domination analysis. The computational results for the TSP show that the R-R-GREEDY outperforms a weight-based greedy (W-GREEDY) in quality at least 10 times on average, while for AP the corresponding domination numbers for R-R-GREEDY and W-GREEDY are 2^{n-1} and 1, respectively.

Our experiments show that the quality of solutions produced by tolerance-based greedy algorithms are overall significantly better than those found by the arc weight-based greedy algorithm. Unfortunately, A-R-GREEDY and A-RC-GREEDY are often slower than W-GREEDY, but R-R-GREEDY, being superior to W-GREEDY in quality, is nearly as fast as W-GREEDY. Overall, the simplest tolerance-based greedy, R-R-GREEDY, is the best algorithm for solving the STSP, while the A-RC-GREEDY algorithm could be suggested for the ATSP. It is worth mentioning that the construction heuristics in [4] (see from Table 1 in [4]) have the following average excesses (taken over seven families of instances) over the length of an optimal tour or lower bound: GR= 580.35%, RI= 710.95%, KSP= 135.08%, GKS= 98.09%, RPC= 102.02%, COP= 23.01%. Computational experiments reported in [5] for our algorithms give R-R-GREEDY= 67.14%, A-R-GREEDY=34.75%, and A-RC-GREEDY= 29.19%. It is not difficult to obtain an upper bound $2(n-3)!$ for the domination number of ATSP-R-R-GREEDY and it would be interesting to find a non-trivial lower bound. Another question is whether a 1-tree-based relaxation of the traveling salesman problem would generate tolerance-based greedy algorithms that are better for the STSP. Also it would be interesting to see if the success of tolerance-based algorithms on the TSP can be extended to other combinatorial optimization problems.

References

1. E. Balas, M.J. Saltzman. An algorithm for the three-index assignment problem. Operations Research **39**, 150–161, 1991.
2. M. Dell'Amico, P. Toth. Algorithms and codes for dense assignment problems: the state of the art. Discrete Applied Mathematics **100**, 17–48, 2000.
3. D. Gamboa, C. Rego, F. Glover. Implementation analysis of efficient heuristic algorithms for the traveling salesman problem. Computers & Operations Research **33**, 1154–1172, 2006.

4. F. Glover, G. Gutin, A. Yeo, A. Zverovich. Construction heuristics for the asymmetric TSP. European Journal of Operational Research **129**, 555–568, 2001.
5. B. Goldengorin, G. Jäger. How To Make a Greedy Heuristic for the Asymmetric Traveling Salesman Problem Competitive. SOM Research Report 05A11, University of Groningen, The Netherlands, 2005
   (http://som.eldoc.ub.rug.nl/reports/themeA/2005/05A11).
6. B. Goldengorin, G. Jäger, P. Molitor. Some Basics on Tolerances. To appear in the Proceedings of AAIM 2006, Lecture Notes in Computer Science, Springer.
7. B. Goldengorin, G. Sierksma. Combinatorial optimization tolerances calculated in linear time. SOM Research Report 03A30, University of Groningen, The Netherlands, 2003 (http://som.eldoc.ub.rug.nl/reports/themeA/2003/03A30/).
8. B. Goldengorin, G. Sierksma, M. Turkensteen. Tolerance based algorithms for the ATSP. Graph-Theoretic Concepts in Computer Science. 30th International Workshop, WG 2004, Bad Honnef, Germany, June 21-23, 2004. J. Hromkovic, M. Nagl, B. Westfechtel (eds.), Lecture Notes in Computer Science **3353**, 222–234, 2004.
9. G. Gutin, A. Yeo. Domination analysis of combinatorial optimization algorithms and problems. Graph Theory, Combinatorics and Algorithms: Interdisciplinary Applications. M. Golumbic, I. Hartman (eds.). Springer, 2005.
10. G. Gutin, A. Yeo and A. Zverovich. Exponential Neighborhoods and Domination Analysis for the TSP. Chapter 6 in: The Traveling Salesman Problem and Its Variations. G. Gutin, A.P. Punnen (eds.). Kluwer, Dordrecht, 223–256, 2002.
11. G. Gutin, A. Yeo, A. Zverovich. Traveling salesman should not be greedy: domination analysis of greedy type heuristics for the TSP. Discrete Applied Mathematics **117**, 81–86, 2002.
12. M. Held, R. Karp. The Traveling-Salesman Problem and Minimum Spanning Trees. Operations Research. **18**, 1138–1162, 1970.
13. K. Helsgaun. An effective implementation of the Lin-Kernighan traveling salesman heuristic. European Journal of Operational Research **126**, 106–130, 2000.
14. D.S. Johnson, G. Gutin, L.A. McGeoch, A. Yeo, W. Zhang, A. Zverovich. Experimental analysis of heuristics for the ATSP. Chapter 10 in: The Traveling Salesman Problem and Its Variations. G. Gutin, A.P. Punnen (eds.). Kluwer, Dordrecht, 445–489, 2002.
15. R. Jonker, A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. Computing **38**, 325-340, 1987.
16. M. Libura. Sensitivity analysis for minimum Hamiltonian path and traveling salesman problems. Discrete Applied Mathematics **30**, 197–211, 1991.
17. G. Reinelt. TSPLIB – a traveling salesman problem library. ORSA Journal of Computing **3**, 376–384, 1991.
18. M. Turkensteen, D. Ghosh, B. Goldengorin, G. Sierksma. Tolerance-based branch and bound algorithms. A EURO conference for young OR researchers and practitioners, ORP3 2005, 6 – 10 September 2005, Valencia, Spain. Proceedings Edited by C. Maroto et al., ESMAP S.L., 171–182, 2005.
19. A. Volgenant. An addendum on sensitivity analysis of the optimal assignment. European Journal of Operational Research **169**, 338–339, 2006.

Appendix

Examples

Example 1. Consider a TSP instance of size 6, with the arc weight matrix shown below.

	1	**2**	**3**	**4**	**5**	**6**
1	∞	6	7	7	7	7
2	7	∞	12	13	13	13
3	35	13	∞	18	19	19
4	35	13	19	∞	24	25
5	35	13	19	25	∞	30
6	216	13	19	25	31	∞

The optimal AP solution to this instance is given by $\{(1,6), (6,2), (2,1), (3,4), (4,5), 5,3)\}$ with length 88 units. This solution contains two subtours, 1–6–2–1 and 3–4–5–3. (Of course this solution is not a unique optimal AP solution.) The optimal RAP solution to this instance contains the arc set $\{(1,2), (2,1), (3,2), (4,2), (5,2), (6,2)\}$ with total objective value of 65 units. Note that in the optimal RAP solution there is only one subtour 1–2–1, and it does not span all vertices.

Example 2. **AP tolerances:** In Example 1, for the optimal AP solution, the arc (1,6) is in the solution. The upper tolerance for this arc is computed by increasing its weight so that it does not remain in an optimal AP solution. Check that this can be done by increasing its weight by more than 1 unit. After this, a new optimal AP solution $\{(1,5), (5,3), (3,4), (4,6), (6,2), (2,1)\}$ is obtained which has an objective value of 89. Hence the upper tolerance for arc (1,6) is 1.

The arc (1,3) is not in the optimal AP solution. In order to compute its lower tolerance, we need to find out by how much its weight needs to be reduced in order for it to enter into an optimal AP solution. The best solution including the arc (1,3) is $\{(1,3), (3,6), (6,4), (4,5), (5,2), (2,1)\}$ with objective value of 95; which means that the weight of the arc (1,3) needs to be reduced by more than $95-88 = 7$ for it to enter an optimal AP solution. Hence the lower tolerance of arc (1,3) is 7.
RAP tolerances: Consider the optimal RAP solution in Example 1. The arc (1,2) is in the solution. If the arc is to be forced out of any optimal RAP solution, then its weight needs to be increased by more than 1 unit. After such an increase, the solution $\{(1,3), (2,1), (3,2), (4,2), (5,2), (6,2)\}$ with objective value 66 becomes optimal. Hence the upper tolerance of the arc (1,3) is 1.

Finally consider the arc (1,6) which is not in the optimal RAP solution. It would enter into an optimal RAP solution only if its weight is reduced by more than 1 unit. After such a reduction, the solution $\{(1,6), (2,1), (3,2), (4,2), (5,2), (6,2)\}$ becomes optimal. Hence the lower tolerance for arc (1,6) is 1.

Example 3. The development of the solution finally output by each of the four algorithms through their iterations is shown in Figures 1 and 2. A black (grey) vertex is an out- (in-) vertex before contraction. Each light grey arc after the contraction is directed from left to right and represented by light grey neighboring vertices. If a begin (end) of a contracted sequence of arcs (path) is chosen for contraction, then it becomes grey (black). The numbers $x(y)$ along each arc are the weight x (respectively, upper/lower tolerance y) of the arc for the four algorithms, computed w.r.t. the RAP. A bold letter on the left side corresponds to the current iteration number for each algorithm. The W-GREEDY, R-R-GREEDY, A-R-GREEDY, and A-RC-GREEDY algorithms output greedy solutions with values 306, 90, 89, 88 (optimal value), respectively. Figure 2 shows that the A-R-GREEDY algorithm returns a greedy solution at the fourth iteration ($\mathbf{D}$) and the A-RC-GREEDY algorithm at the third iteration ($\mathbf{C}$), since all depicted arcs are from an optimal AP solution which is a Hamiltonian cycle.

W - GREEDY

R - R - GREEDY

A

B

C

D

E

306

90
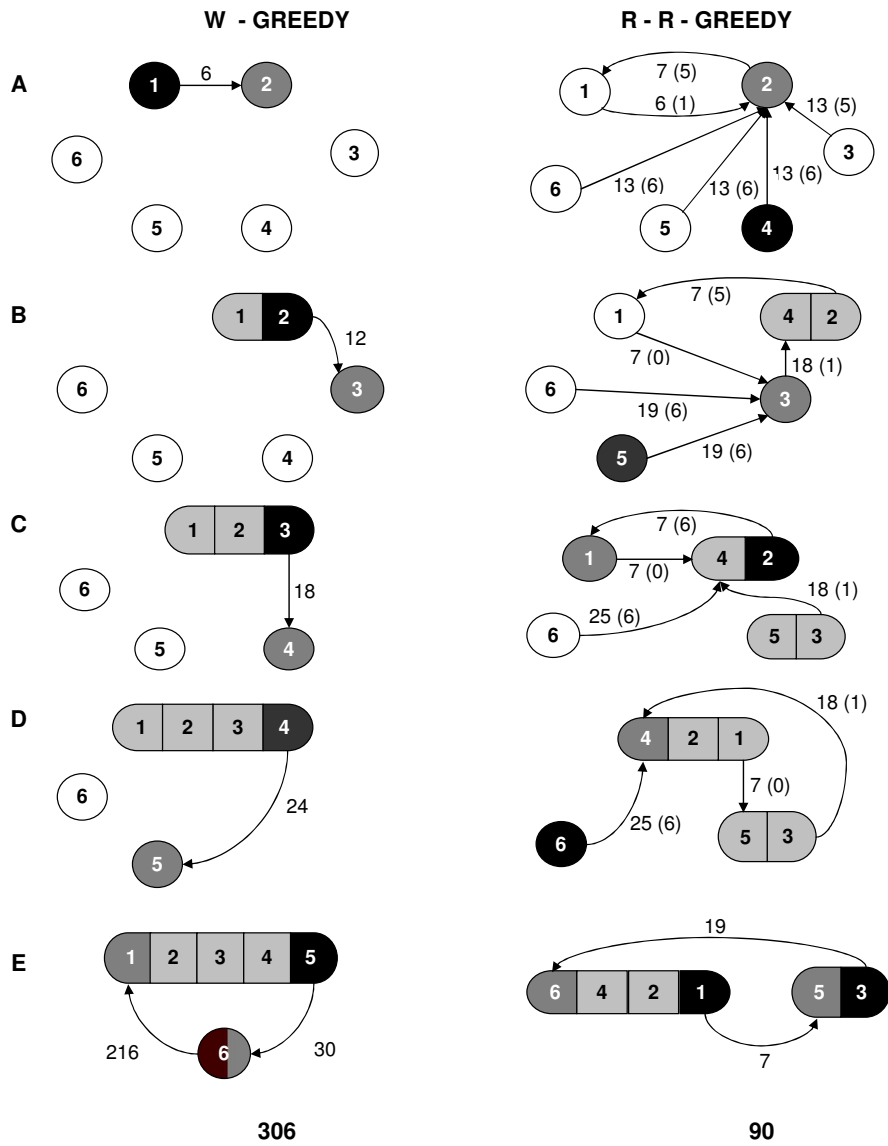
Fig. 1. Working of W-GREEDY and R-R-GREEDY
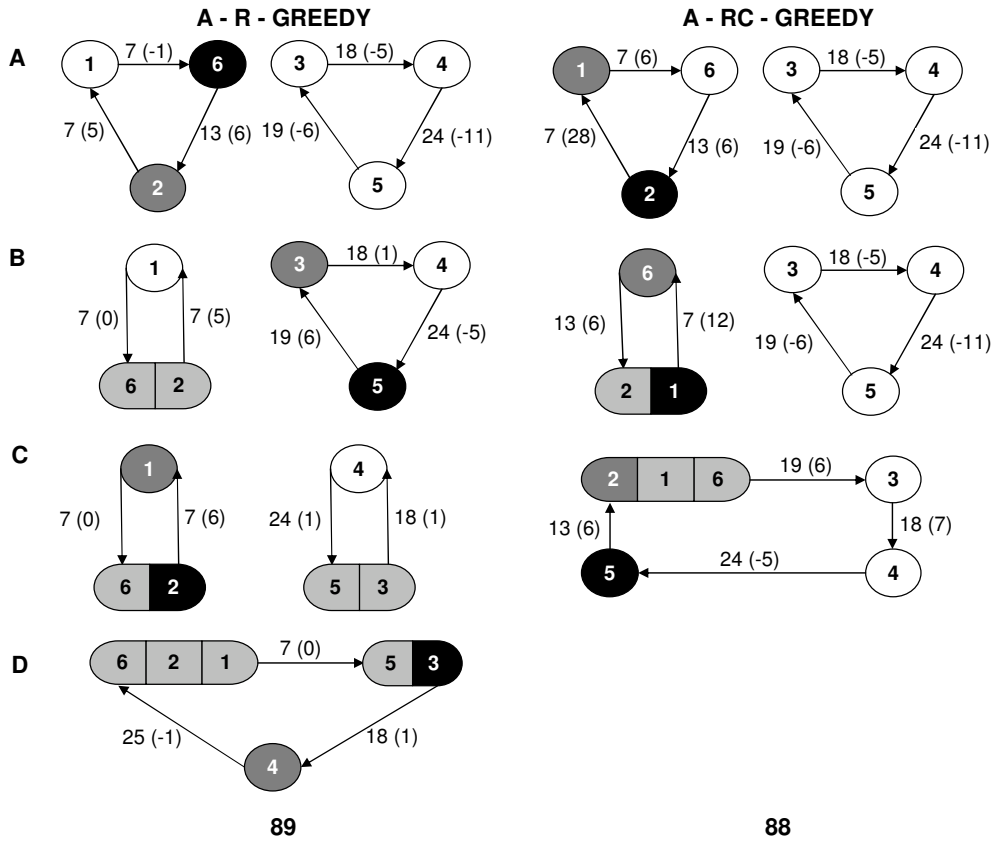
Fig. 2. Working of A-R-GREEDY and A-RC-GREEDY
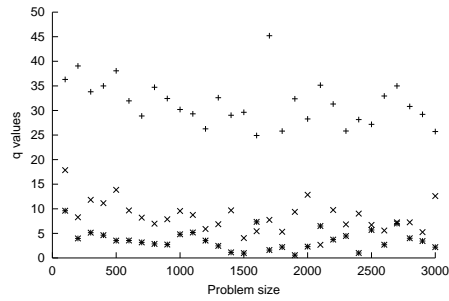
Solution quality parameter values

Class 1

Class 2

Class 3

Class 4

Class 5

Class 6

Class 7

Class 8