# How To Make a Greedy Heuristic for the Asymmetric Traveling Salesman Problem Competitive

Boris Goldengorin[*] and Gerold Jäger[†]

**Abstract**

It is widely confirmed by many computational experiments that a greedy type heuristics for the Traveling Salesman Problem (TSP) produces rather poor solutions except for the Euclidean TSP. The selection of arcs to be included by a greedy heuristic is usually done on the base of cost values. We propose to use upper tolerances of an optimal solution to one of the relaxed Asymmetric TSP (ATSP) to guide the selection of an arc to be included in the final greedy solution. Even though it needs time to calculate tolerances, our computational experiments for the wide range of ATSP instances show that tolerance based greedy heuristics is much more accurate and faster than previously reported greedy type algorithms.

## 1 Introduction

Perhaps the most famous classic combinatorial optimization problem is called the *Traveling Salesman Problem* (TSP) ([20]). It has been given this picturesque name because it can be described in terms of a salesperson who must travel to a number of cities during one tour. Starting from his (her) home city, the salesperson wishes to determine which route to follow to visit each city exactly once before returning to his home city so as to minimize the total length tour. The length (cost) of traveling from city $i$ to city $j$ is denoted by $c(i,j)$. These costs are called *symmetric* if $c(i,j) = c(j,i)$ for each pair of cities $i$ and $j$, and *asymmetric* otherwise. A TSP instance is defined by all non-diagonal entries of the $n \times n$ matrix $C = ||c(i,j)||$. There have been a number of applications of TSP that have nothing to do with salesman. For example, when a truck leaves a distribution center to deliver goods to a number of locations, the problem

[*]Corresponding author. Department of Econometrics and Operations Research, University of Groningen, P.O. Box 800, 9700 AV Groningen, The Netherlands; Fax. +31-50-363-3720, E-mail: B.Goldengorin@rug.nl and Department of Applied Mathematics, Khmelnitsky National University, Ukraine

[†]Computer Science Institute, Martin-Luther-University Halle-Wittenberg, Von-Seckendorff-Platz 1, 06120 Halle (Saale), Germany, E-mail: jaegerg@informatik.uni-halle.de

of determining the shortest route for doing this is a TSP. Another example involves the manufacture of printed circuit board, the problem of finding the most efficient drilling sequence is a TSP. More applications to Genome Sequencing, Starlight Interferometer Program, DNA Universal Strings, Touring Airports, Designing Sonet Rings, Power Cables, etc are indicated at ([19]).

Computer scientists have found that the ATSP is among certain type of problems, called *NP-hard* problems which are especially intractable, because the time it takes to solve the most difficult examples of an NP-hard problem seems to grow exponentially as the amount of input data increases. Surprisingly, powerful algorithms based on the branch-and-cut (b-n-c) ([21]) and branch-and-bound (b-n-b) ([5]) approaches have succeeded in solving to optimality certain huge TSPs with many hundreds (or even thousands) of cities. The fact that the TSP is a typical NP-hard optimization problem means, that solving instances with a large number of cities is very time consuming if not impossible. For example, the solution of the 15112-city TSP was accomplished in several phases in 2000/2001, and used a total of 22.6 years of computer time, adjusted to a 500 MHz, EV6 Alpha processor ([18]). A *heuristic* is a solution strategy that produces an answer without any formal guarantee as to the quality ([20]). Heuristics clearly are necessary for NP-hard problems if we expect to solve them in reasonable amounts of computer time. Heuristics also are useful in speeding up the convergence of exact optimization algorithms, typically by providing good starting solutions. A *metaheuristic* is a general solution method that provides both a general structure and strategy guidelines for developing a specific heuristic method to fit a particular kind of problem.

Heuristics can be classified into several categories, namely *construction, improvement, partitioning and decomposition, and specialized heuristics* ([25]). A construction heuristic for the TSP build a tour without an attempt to improve the tour once constructed. Construction heuristics are the fastest heuristics among all of the above mentioned categories of heuristics ([6],[7]). They can be used to create approximate solutions for the TSP when the computing time is restricted, to provide good initial solutions for exact b-n-c or b-n-b algorithms ([21],[5]).

The Greedy Algorithm (GR) is one of the fastest construction type heuristic in combinatorial optimization. Computational experiments show that the GR is a popular choice for tour construction heuristics, work at acceptable level for the Euclidean TSP, but produce very poor results for the general Symmetric TSP (STSP) and Asymmetric TSP (ATSP) ([6]-[8]). The experimental results for the asymmetric TSP presented in [7] led its authors to the conclusion that the greedy algorithm "might be said to self-destruct" and that it should not be used even as "a general- purpose starting tour generator".

A numerical measure for evaluation of heuristics that compares heuristics according to their so called domination number is suggested in ([9]). The *domination number* of a heuristic $\mathcal{A}$ for the TSP is the maximum integer $d(n)$ such that, for every instance $\mathcal{I}$ of the TSP on $n$ cities, $\mathcal{A}$ produces a tour $h$ which is not worse than at least $d(n)$ tours in $\mathcal{I}$ including $h$ itself. Theorem 2.1 in [13] (see also [14]) on the greedy algorithm for the ATSP confirms the above conclusion: for every $n > 1$ there exist instances of the asymmetric TSP with n vertices for which the greedy algorithm produces the unique worst tour. In the abstract of [1] the authors send the following message: "The practical message of this paper is that the greedy algorithm should be used with great care, since for many optimization problems its usage seems impractical even for generating a starting solution (that will be improved by a local search or another heuristic)." Note that all these negative conclusions are drawn for the cost based greedy (CBG) type heuristics.

There are various heuristic methods based on tolerances and related to a specific mathematical programming problem, for example to the well known transportation problem, available to get an initial basic feasible solution, such as northwest corner rule, best cell method, etc. Unfortunately, there is no reason to expect the basic feasible solution provided by the northwest corner rule to be particularly close to the optimal solution. Therefore, it may be worthwhile to expand a little more effort on obtaining a promising initial basic feasible solution in order to reduce the number of iterations required to reach the optimal solution. One procedure which is designed to do this is Vogel's Approximation Method (VAM) [23]. The VAM is based on the use of the "difference" associated with each row and column in the original instance $C$. A row or column "difference" is defined as the arithmetic difference between the smallest and next-to-the smallest element in that row or column. This quantity provides a measure of the proper priorities for making allocations to the respective rows and columns, since it indicates the minimum unit penalty incurred by failing to make an allocation to the smallest cell in that row or column [16]. Such a difference is also called the *regret* for that row(column) [2] because it represents the minimum penalty for not choosing the smallest cost in the row (column). The element with the smallest cost in the row (column) with the largest regret is selected in VAM for starting the transportation simplex algorithm. A similar idea applied to rows is used for the MAX-REGRET heuristic for solving the three-index assignment problem [2]. In this paper we generalize the above mentioned "differences" within the notion of upper tolerances and apply them within a framework of branch-and-bound algorithms for solving the ATSP by greedy type heuristics. Although the concept of tolerances has been applied for decades (in sensitivity analysis; see e.g. [12] and [17]), only Helsgaun's version of the Lin-Kernighan heuristic for the STSP applies tolerances; see [15]. As easy to see the VAM, MAX-REGRET and Helsgaun's version of the Lin-Kernighan heuristics are not embedded in a unified framework called a *metaheuristic* for solving different combinatorial optimization problems. Let us remind that a *metaheuristic* is a general solution method that provides both a general structure and strategy guidelines for developing a specific heuristic method to fit a particular kind of problem. All above mentioned heuristics are special cases of our metaheuristic. For example, our metaheuristic applied to the ATSP leads to a family of heuristics including our $R$-$R$TBG heuristic which is an analogy of MAX-REGRET heuristic for the three-index assignment problem [2]. To our best knowledge such a family of heuristics does not discussed in the available literature.

Currently, most of construction heuristics for the TSP including the GR delete high cost arcs or edges and save the low cost ones. A drawback of this strategy is that costs of arcs and edges are no accurate indicators whether those arcs or edges are included or excluded in a 'good' TSP solution. In ([11]) and ([26]), it is shown that *tolerances* are better indicators, and they have been successfully applied for improving the bounds and branching rules within the framework of b-n-b algorithms. A tolerance value of an edge/arc is the cost of excluding or including that edge/arc from the solution at hand. In this paper we present a tolerance based metaheuristic within a framework of branch-and-bound paradigm and apply it for solving the ATSP.

Our paper is organized as follows. In the next section we embed the GR for the ATSP in a metaheuristic based on branch-and-bound paradigm and tolerances. Here we define the Contraction Problem (CP) of finding an arc for contraction by the GR and show that an optimal solution to a natural relaxation of the Assignment Problem (AP) can be used as an extension of an optimal solution to CP. In Section 3 we briefly report on previous work related to tolerances in combinatorial optimization and discuss the computational complexities of tolerance problems for the AP and Relaxed AP. In Section 4 we describe a set of tolerance based greedy algorithms with different relaxations of the ATSP and report the results of computational experiments with them in Section 5. The conclusions and future research directions appear in Section 6.

## 2 The Greedy Algorithm within a framework of a metaheuristic

Let us remind the GR for the ATSP as it is described in ([8]). We consider the entries of the $n \times n$ matrix $C$ as costs (weights) of the corresponding simple weighted complete digraph $G = (V, E, C)$ with $V = \{1, \ldots, n\}$ vertices and $E = V \times V$ arcs such that each arc $(i, j) \in E$ is weighted by $c(i, j)$. The GR finds the lightest arc $(i, j) \in E$ and contracts it updating the costs of $C := C_p$ till $C$ consists of a pair of arcs. The contracted arcs and the pair of remaining arcs form the "Greedy" tour in $G$.

We shall address the GR in terms of the b-n-b framework for solving the ATSP. A b-n-b algorithm initially solves a *relaxation* of the original NP-hard problem. In case of the ATSP, the Assignment Problem (AP) is a common choice. The AP, in terms of the ATSP, is the problem of assigning $n$ city's outputs to $n$ city's inputs against minimum cost; an optimal solution of the AP is called a *minimum cycle cover.* In terms of the ATSP an AP feasible solution requires that each city will be visited exactly once without necessarily creating a single (*Hamiltonian*) cycle. If the minimum cycle cover at hand is a single tour, then the ATSP instance is solved; otherwise, the problem is partitioned into new subproblems by including and excluding arcs. In the course of the process, a *search tree* is generated in which all solved subproblems are listed. B-n-b algorithms comprise two major ingredients: *a branching rule* and *a lower* *bound.* The objective of branching rule is to exclude the infeasible solutions to the ATSP found for its relaxation. A lower bound is applied to fathom as many vertices in the search tree as possible. A subproblem is fathomed if its lower bound exceeds the value of the best ATSP solution found so far in the search tree. For sake of simplicity, we consider only the so called *binary branching rules* ([29]), i.e. such a branching rule which either include or exclude a selected arc from the current optimal solution to a relaxation of the ATSP.

We present the execution of the GR by a single path of the corresponding b-n-b search tree such that the lower bound is equal to the Glover et al.'s Contraction Problem (CP) ([8]) and the branching rule is defined by a lightest arc in the given ATSP instance. Since at each iteration the GR contracts a single lightest arc, i.e. creates a subproblem which includes that arc in the unknown GREEDY solution, it means that the GR discards the another subproblem in which the same arc is prohibited. Thus the GR can be represented by a single path in a search tree consisting only vertices (subproblems) related to each inclusion of an arc from the optimal solution of a relaxed ATSP. We define the CP as follows:

$$\min \{c(i,j) \,:\, i,j \in V\} = c(i_0, j_0),$$

We also use the following two simple combinatorial optimization problems related to either the ATSP or CP, namely either the Assignment Problem (AP) or the Relaxed AP (RAP), respectively. We define the AP as follows:

$$\min \left\{ a(\pi) = \sum_{i=1}^{n} c(i, \pi(i)) : \pi \in \Pi \right\} = a(\pi_0),$$

here a feasible assignment $\pi$ is a permutation which maps the rows $V$ of $C$ *onto* the set of columns $V$ of $C$ and the cost of permutation $\pi$ is $a(\pi) = \sum_{(i,j)\in\pi} c(i,j)$; $\Pi$ is the set of all permutations, and $\pi$ is called *feasible* if $a(\pi) < \infty$.

Now we define the RAP. A *feasible solution* $\theta$ to the RAP is a mapping $\theta$ of the rows $V$ of $C$ *into* the columns $V$ of $C$ with $a(\theta) < \infty$. We denote the set of feasible solutions to the RAP by $\Theta$. It is clear that $\Pi \subset \Theta$. The RAP is the problem of finding $a(\theta_0) = \sum_{i\in V} \min\{c(i,j) \,:\, j \in V\}$. Further we will treat a feasible solution $\theta$ as a set of $n$ arcs. As easy to see an optimal solution (arc) $(i_0, j_0)$ to the CP is included in an optimal solution $\theta_0$ to the RAP, i.e. $(i_0, j_0) \in \theta_0$. So, one may to consider $\theta_0$ as an extension of an optimal solution to the CP. If the set of optimal solutions (arcs) to the CP has more than one arc and these arcs are located in pairwise distinct rows of $C$, then at most $n$ arcs will be included in an optimal solution to the RAP. The relationship between the optimal values of CP and RAP is described in the following lemma.

**Lemma 1** *For any $n > 1$ we have that $c(i_0, j_0) \le a(\theta_0)/n$.*

**Proof.** $a(\theta_0) = \sum_{i\in V} \min\{c(i,j) \,:\, j \in V\} \ge \sum_{i\in V} c(i_0, j_0) = nc(i_0, j_0)$. ■

For sake of completeness we define the ATSP as follows. A feasible solution $\pi$ to the AP is called a *cyclic permutation* and denoted by $h$ if the set of its arcs represents a Hamiltonian cycle in $G$. The whole set of Hamiltonian cycles in $G$ is denoted by $\mathcal{H}$. Thus, the ATSP is the problem of finding

$$\min\{a(h) \,:\, h \in \mathcal{H}\} = a(h_0).$$

It is clear that $\mathcal{H} \subset \Pi \subset \Theta$ implies that $nc(i_0, j_0) \leq a(\theta_0) \leq a(\pi_0) \leq a(h_0)$. Note, that the computational complexities for finding an optimal solution to the CP, RAP, and AP are $O(n^2)$, $O(n^2)$, and $O(n^3)$, respectively.

## 3 Tolerances for the RAP and AP

The second concept we have built on is the *tolerance problem* for a relaxed ATSP. The tolerance problem for the RAP is the problem of finding for each arc $e = (i, j) \in E$ the maximum decrease $l(e)$ and the maximum increase $u(e)$ of the arc cost $c(e)$ preserving the optimality of $\theta_0$ under the assumption that the costs of all other arcs remain unchanged. The values $l(e)$ and $u(e)$ are called the *lower* and the *upper tolerances*, respectively, of arc $e$ with respect to the optimal solution $\theta_0$ and the function of arc costs $c$. In the following portion we consider a combinatorial minimization problem defined on the ground set $\mathcal{E}$:

$$\min\{a(S) \,:\, S \in D\} = a(S^*),$$

with an additive objective function $a(S) = \sum_{e \in S} c(e)$, the set of feasible solutions $D \subset 2^{\mathcal{E}}$, set of optimal solutions $D^*$ such that $S^* \in D^*$. For each $e \in \mathcal{E}$, $D^*_+(e)$ and $D^*_-(e)$ are the sets of optimal solutions containing $e$ and not containing $e$ such that $D^* = D^*_+(e) \cup D^*_-(e)$ and $D^*_+(e) \cup D^*_-(e) = \emptyset$. If $D = \Pi$ then we have the AP, and if $D = \Theta$ then we have the RAP. As shown [11],[10] for each $e \in S^*$ the upper tolerance $u_{S^* \in D}(e) = a(S) - a(S^*)$ for each $S \in D^*_-(e)$, and means the upper tolerance of an element $e \in S^*$ with respect to the set of feasible solutions $D$. Similarly, the lower tolerance $l_{S^* \in D}(e) = a(S) - a(S^*)$ for each $S \in D^*_+(e)$, and means the lower tolerance of an element $e \notin S^*$ with respect to the set of feasible solutions $D$. If one excludes an element $e$ from the optimal solution $S^*$, then the objective value of the new problem will be $a(S^*) + u_{S^* \in D}(e)$. The same holds for the lower tolerance if the element $e \in \mathcal{E} \setminus S^*$ is included. So a tolerance-based algorithm knows the cost of including or excluding elements before it selects the element either to include or to exclude. Moreover, based on the upper tolerances one may describe the set of optimal solutions $D^*$ to RAP as follows ([11],[10]):

$(i)$ if $u(e) > 0$ for all $e \in S^*$, then $|D^*| = 1$;
$(ii)$ if $u(e) > 0$ for $e \in R \subset S^*$ and $u(e) = 0$ for $e \in S^* \setminus R$, then $|D^*| > 1$ and $\cap D^* = R$.
$(iii)$ if $u(e) = 0$ for all $e \in S^*$, then $|D^*| > 1$ and $\cap D^* = \emptyset$.

As shown in ([11],[26],[15]) the average percentage of common arcs in corresponding AP and ATSP optimal solutions varies between 40% and 80%, and we claim that the average percentage of common arcs in corresponding RAP and ATSP optimal solutions varies between 20% and 50%. Moreover, an arc with the largest upper tolerance from an optimal AP solution will appear in an optimal ATSP solution in average 10 times more often than a lightest arc from the optimal AP solution. Similarly, an arc with the largest upper tolerance from an optimal RAP solution will appear in an optimal ATSP solution in average 15 times more often than a lightest arc from the optimal RAP solution. These results confirm that predictions based on upper tolerances are clearly better than predictions based on costs ([11],[26],[15]). Hence, our branching rule for the tolerance based algorithms will use an arc with the largest upper tolerance w.r.t. either the RAP or the AP.

It is important to mention that the time complexity of the tolerance problem for a Polynomially Solvable (PS) problem (for example, either the RAP or AP) defined on the set of arcs $E$ ([3],[27]) is at most $O(|E|g(n)|)$, assuming that the time complexity of PS problem is equal to $g(n)$. Hence, the time complexities of solving the tolerance problems for RAP and AP are at most $O(n^4)$ and $O(n^5)$, respectively, because the time complexities of solving the RAP and AP are $O(n^2)$ and $O(n^3)$, respectively.

Recently, Volgenant ([28]) has suggested an $O(n^3)$ algorithm for solving the tolerance problem for AP. Let us show that the time complexity of tolerance problem for the RAP is $O(n^2)$. For finding the optimal value $a(\theta_0)$ of RAP we should find in each $i$-th row of $C$ its smallest value $\min\{c(i,j) : j \in V\} = c[i, j_1(i)]$ and for computing the upper tolerance $u[i, j_1(i)]$ of the arc $[i, j_1(i)]$ it is enough to know a pair of smallest values in the same row $i$, i.e. $c[i, j_1(i)]$ and $c[i, j_2(i)] = \min\{c(i,j) : j \in V\backslash\{j_1(i)\}\}$. Thus, $u[i, j_1(i)] = c[i, j_2(i)] - c[i, j_1(i)]$ will be computed in $O(n)$ time with $O(1)$ space complexity. For computing all lower tolerances for entries in $i$-th row it is enough to reduce each entry of the $i$-th row by the smallest value $c[i, j_1(i)]$, i.e. $l(i,j) = c(i,j) - c[i, j_1(i)]$ for $j \in V \setminus \{j_1(i)\}$ and $l[i, j_1(i)] = \infty$. Again, such a reduction can be done in $O(n)$ time and the result can be stored in $i$-th row of $C$. Now we are able to solve the RAP and compute all tolerances in $O(n^2)$ time and $O(n)$ space complexities.

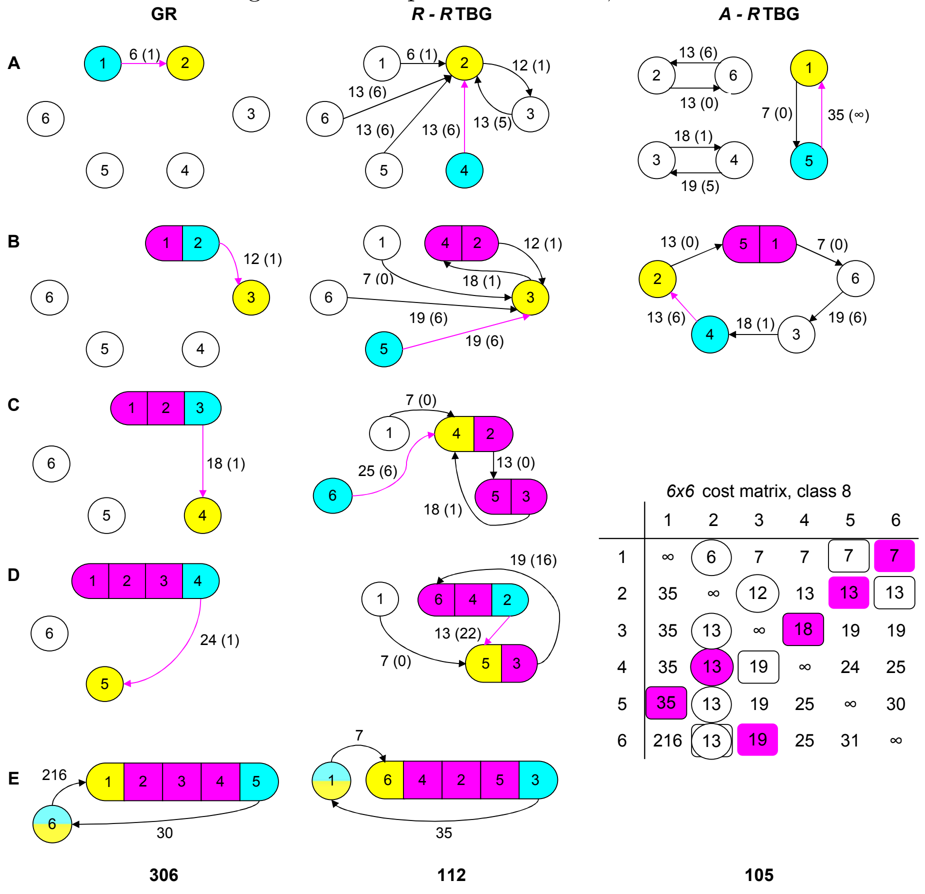
Note that the AP can be solved in $O(n^3)$ time and computing all upper tolerances for an optimal AP solution w.r.t. the AP needs also $O(n^3)$ time, but an optimal AP solution is a feasible RAP solution. Hence, with purpose to decrease the computational complexity of upper tolerances for an optimal AP solution from $O(n^3)$ time to $O(n^2)$ time, we have decided to compute the upper tolerances for an optimal AP solution w.r.t. the RAP as follows. Let $\pi_0 = \{[i, j_k(i)] : i = 1, \ldots, n\}$ be a set of arcs $[i, j_k(i)]$ from an optimal AP solution. Then we define the upper tolerance $u_{\pi_0 \in \Theta}[i, j_k(i)]$ of an arc $[i, j_k(i)]$ from an optimal AP solution w.r.t. the RAP as follows: $u_{\pi_0 \in \Theta}[i, j_k(i)] = c[i, j_{k+1}(i)] - c[i, j_k(i)]$, for $k = 1, \ldots, n-1$; $u_{\pi_0 \in \Theta}[i, j_n(i)] = c[i, j_n(i)] - c[i, j_{n-1}(i)]$, for $k = n$. Here, for each row $i$ of $C$ we have ordered all its entries in a non-decreasing order, i.e. $c[i, j_1(i)] \leq c[i, j_2(i)] \leq \ldots \leq c[i, j_{n-1}(i)] \leq c[i, j_{n-1}(i)]$.

## 4 A tolerance based metaheuristic

Let us explain our enhancements of tolerance based greedy algorithms compared to the cost based GR in the framework of b-n-b algorithms. Our first enhancement is based on an improved lower bound for the ATSP (either the RAP or AP) compared to the CP. Our second enhancement is based on an improved branching rule which uses the largest upper tolerance of an arc from an optimal solution to either the RAP or AP instead of a lightest arc from an optimal solution to the CP. Note that by using the RAP instead of CP we may expect the quality improvement of a RAP based greedy solutions without essential increasing of the computing times. Since the time complexity of AP is $O(n^3)$, the question whether we may reach further improvements by the AP based greedy algorithm without essential increment of computing times will be answered by means of computational experiments.

Now we are ready to present two Tolerance Based Greedy (TBG) algorithms, namely the $X$-$Y$TBG algorithms with $X, Y \in \{R, A\}$ in the same framework as the original GR. Here the first letter "X" stands for a relaxation of the ATSP and the second letter "Y" stands for the tolerance problem of one of the relaxations of ATSP. Our first algorithm is the RAP Tolerance Based Greedy ($R$-$R$TBG) algorithm. The $R$-$R$TBG algorithm recursively finds an arc $(i, j) \in E$ with the largest upper tolerance from an optimal RAP solution, and contracts it updating the costs of $C := C_p$ till $C$ consists of a pair of arcs. The index $p$ in $C_p$ stands for a new vertex $p$ found after each contraction of the arc $(i, j)$. The contracted arcs and the pair of remaining arcs form the "$R$-$R$TBG" tour in $G$. The $R$-$R$TBG algorithm runs in $O(n^3)$ time since a single contraction needs $O(n^2)$ time. Note that if at the current iteration after a single contraction the deleted column does not include either a lightest out-arc or a corresponding upper tolerance, then at the next iteration of $R$-$R$TBG algorithm we preserve all lightest out-arcs and the corresponding upper tolerances. Otherwise, at the next iteration the rows containing either a lightest out-arc or a corresponding upper tolerance will be updated.

Figure 1: Example: 6 × 6 matrix, class 8.



|   | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | ∞ | 6 | 7 | 7 | 7 | 7 |
| 2 | 35 | ∞ | 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 | ∞ |

For example (see the $R$-$R$TBG algorithm in Fig. 1), after first contraction on the iteration **A** we delete the second column from the $6 \times 6$ costs matrix together with the minima attained on the arcs $(1,2),(3,2),(5,2),(6,2)$, and preserve the minimum attained on the arc $(2,3)$ in the third column. Hence, on the next iteration **B** the $R$-$R$TBG algorithm updates the minima and upper tolerances in rows $1,(4,2),5,6$ and preserves the minimum $c[3,(4,2)] = 12$ and its upper tolerance $u[3,(4,2)] = 1$. Here $p = (4,2)$ is a new vertex representing the contracted arc $(4,2)$. This book-keeping technique (for preserving either a lightest out-arc or a corresponding upper tolerance) incorporated in the $R$-$R$TBG algorithm explains why in average we have reduced the PC times of $R$-$R$TBG compared to GR (see Table 1).

Our second algorithm is the $A$-$R$TBG. The choice of the AP as the ATSP relaxation can be motivated as follows. The out-degree of each vertex in a graph representing an optimal RAP solution is equal to one and the in-degree of each vertex can be an arbitrary number between 0 and $n-1$ such that the sum of in-degrees of all vertices is equal to $n$ (see iteration **A** of the $R$-$R$TBG in Fig. 1). The in-degree and out-degree for each vertex in the graph representing an optimal AP solution are equal to one (see iteration **A** of the $A$-$R$TBG in Fig. 1). Hence, the structural distinctions between the graphs of a feasible ATSP tour and a feasible RAP solution are worse compared to the structural distinctions between the graphs of a feasible ATSP tour and a feasible AP solution (see Fig. 1). Since the AP can be solved in $O(n^3)$ time and computing all upper tolerances to an optimal AP solution needs also $O(n^3)$ time, in the AP with RAP Tolerance Based Greedy ($A$-$R$TBG) algorithm we have decided to use an optimal AP solution as a better approximation of the unknown optimal ATSP solution. Note that each optimal AP solution is a feasible RAP solution. Therefore, with purpose to decrease the computational complexity of upper tolerances for an optimal AP solution from $O(n^3)$ time to $O(n^2)$ time, we have decided to compute the upper tolerances of RAP for the optimal AP solution instead of upper tolerances for the optimal AP solution itself.

The $A$-$R$TBG algorithm recursively finds an arc $(i, j)$ in an optimal AP solution with the largest upper tolerance from the tolerance RAP, and contracts it updating the costs of $C := C_p$ till $C$ consists of either a hamiltonian cycle or a pair of arcs. The contracted arcs and the pair of remaining arcs form the "$A$-$R$TBG" tour in $G$. The $A$-$R$TBG algorithm runs in $O(n^4)$ time, since a single contraction needs $O(n^3)$ time. A slight modification of the $A$-$R$TBG algorithm based on upper and lower tolerances is denoted by $A$-$R_1$TBG algorithm and contracts an arc with the largest tolerance chosen from all tolerances as follows. Each lower tolerance is used with a negative sign and each upper tolerance with a positive sign. Hence, in the $A$-$R_1$TBG algorithm all involved tolerances have finite numbers.

The distinctions in behavior of the GR, $R$-$R$TBG, and $A$-$R$TBG by means of the $6 \times 6$ numerical example taken from class 8 are illustrated in Fig. 1. The numbers $1, 2, ..., 6$ on the left side of the vertical line and above the horizontal line in the $6 \times 6$ costs matrix are the numbers of cities. The same numbers are indicated as cycled numbers for vertices (cities). An optimal RAP (respectively, AP, ATSP) solution is indicated in the $6 \times 6$ costs matrix by cycled (respectively, squared, red) entries. A blue(yellow) vertex is an out-(in-) vertex before contraction and a red arc is an arc chosen for contraction. Each red arc after contraction is oriented from left to right and represented by two red neighboring vertices. If an end of a contracted sequence of arcs (path) is chosen for contraction, then the corresponding end became either blue or yellow. The numbers $x(y)$ along each arc are the weight $x$ (respectively, upper tolerance $y$) of the arc for GR, $R$-$R$TBG, and $A$-$R$TBG, where the upper tolerance $y$ is computed w.r.t. the RAP. A bold letter on the left side corresponds to the current iteration number for each algorithm. The GR, $R$-$R$TBG, $A$-$R$TBG algorithms output greedy solutions with values 306, 112, 105 (optimal value), respectively. Fig. 1 shows that the $A$-$R$TBG algorithm needs less iterations (contractions) than the $A$-$R$TBG algorithm and returns a greedy solution at the second iteration (**B**) since all depicted arcs are arcs from an optimal AP solution which is a Hamiltonian cycle.

## 5 Computational experiments

The algorithms were implemented in C under Linux and tested on an AMD Opteron(tm) Processor 242 1.6 GHz machine with 4 GB of RAM. We test all four greedy algorithms GR, $R$-$R$TBG, $A$-$R$TBG, and $A$-$R_1$TBG on the following 8 classes of instances. The classes from 1 to 7 are exactly the classes from ([8]) and class 8 is the class of GYZ instances introduced in ([13]) for which the domination number of GR algorithm for the ATSP is 1 (see Theorem 2.1 in [13]). The exact description of the 8 classes is as follows.

1 *All* asymmetric instances from TSPLIB ([24]) (26 instances).

2 *All* symmetric instances from TSPLIB with dimension smaller than 3000 (99 instances).

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

4 Asymmetric instances with $c(i, j)$ randomly and uniformly chosen from $\{0, 1, \cdots, i \cdot j\}$ for $i \neq j$, 10 for each dimension $100, 200, \cdots$, 1000 and 3 for each dimension $1100, 1200, \cdots, 3000$ (160 instances).

5 Symmetric instances with $c(i, j)$ randomly and uniformly chosen from $\{0, 1, \cdots, 100000\}$ for $i < j$, 10 for each dimension $100, 200, \cdots$, 1000 and 3 for each dimension $1100, 1200, \cdots, 3000$ (160 instances).

6 Symmetric instances with $c(i, j)$ randomly and uniformly chosen from $\{0, 1, \cdots, i \cdot j\}$ for $i < j$, 10 for each dimension $100, 200, \cdots$, 1000 and 3 for each dimension $1100, 1200, \cdots, 3000$ (160 instances).

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 for each dimension $100, 200, \cdots$, 1000 and 3 for each dimension $1100, 1200, \cdots$, 3000 (160 instances).

8

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

for each dimension $5, 10, \cdots$, 1000 (200 instances).

Table 1 gives for all classes the average excess of all algorithms above the optima (the TSPLIB classes 1 and 2 are with known optima [24]), the AP lower bound (for the asymmetric classes 3, 4, 7, and 8) or the HK (Held-Karp) lower bound ([15]) (for the symmetric classes 5 and 6) over all instances of this class. Additionally, it gives the average execution times over all instances of the classes.

Table 1: Average excess over optimum, AP or HK, average time

| | Cl. 1 (26) | | Cl. 2 (99) | | Cl. 3 (160) | | Cl. 4 (160) | |
|---|---|---|---|---|---|---|---|---|
| | Opt. (%) | Time (sec.) | Opt. (%) | Time (sec.) | AP (%) | Time (sec.) | AP (%) | Time (sec.) |
| GR | 30.45 | 0.04 | 118.20 | 4.14 | 287.31 | 21.27 | 1135.64 | 17.84 |
| $R$-$R$TBG | 23.28 | 0.04 | 107.31 | 2.53 | 85.75 | 12.81 | 22.00 | 11.80 |
| $A$-$R$TBG | 7.17 | 0.31 | 96.64 | 27.03 | 22.29 | 149.01 | 2.60 | 228.48 |
| $A$-$R_1$TBG | 10.95 | 0.35 | 75.42 | 29.68 | 25.67 | 179.88 | 1.86 | 246.28 |

| | Cl. 5 (160) | | Cl. 6 (160) | | Cl. 7 (160) | | Cl. 8 (200) | |
|---|---|---|---|---|---|---|---|---|
| | HK (%) | Time (sec.) | HK (%) | Time (sec.) | AP (%) | Time (sec.) | AP (%) | Time (sec.) |
| GR | 233.12 | 19.61 | 924.53 | 18.66 | 2439.86 | 18.67 | 481.97 | 1.07 |
| $R$-$R$TBG | 84.28 | 12.56 | 27.90 | 12.12 | 119.49 | 11.56 | 93.64 | 1.28 |
| $A$-$R$TBG | 38.69 | 165.05 | 14.77 | 271.94 | 61.12 | 901.74 | 0 | 55.79 |
| $A$-$R_1$TBG | 28.89 | 201.99 | 11.42 | 294.86 | 50.10 | 909.37 | 0 | 56.14 |

Table 1 provides an overview of the quality of solutions and corresponding PC times. The $R$-$R$TBG algorithm returns in average 14.86 times better quality and requires in average only 0.75 of GR PC times. The highest improvement in quality (51.62 times) is attained for class 4 (randomly and uniformly distributed asymmetric instances) by spending in average just 0.66 of PC times GR. The second best improvement for asymmetric instances is attained on class 7 in quality with the average factor 20.42 and decreasing the GR PC times in average by the factor 0.62. The worst improvement in quality (in average just 1.1 times) is attained for class 2 (all symmetric TSPLIB instances with $n < 3000$ which are not the target of this paper) by spending just 0.61 of GR PC times. Further improvements in quality for classes 1 to 7 (in average by factor 80.53 (109.12)) by spending more in average 15.05 (16.14) PC times are reached by the $A$-$R$TBG ($A$-$R_1$TBG) algorithm. Note that for class 8 the $A$-$R$TBG algorithm finds a tour with optimal length or with optimal length +1 in all cases, whereas the GR always finds the worst tour.

Table 2 shows the quality of the greedy algorithms for all examples of class 1, i.e. all 26 asymmetric TSPLIB instances. Note that both $A$-$R$TBG and $A$-$R_1$TBG have solved all asymmetric TSPLIB instances with $323 \leq n \leq 443$ exactly.

Table 2: Excess for all asymmetric TSPLIB instances (Class 1)

| | br17 | p43 | ry48p | ft53 | ft70 | ftv33 | ftv35 | ftv38 | ftv44 |
|---|---|---|---|---|---|---|---|---|---|
| Dim. | 17 | 43 | 48 | 53 | 70 | 34 | 36 | 39 | 45 |
| GR | 148.72 | 5.16 | 32.55 | 77.73 | 14.84 | 33.51 | 24.37 | 34.44 | 18.78 |
| $R$-$R$TBG | 107.69 | 2.60 | 28.87 | 25.04 | 5.78 | 20.45 | 15.61 | 21.96 | 26.78 |
| $A$-$R$TBG | 5.13 | 1.12 | 9.81 | 16.99 | 1.84 | 23.41 | 14.46 | 6.54 | 11.53 |
| $A$-$R_1$TBG | 112.82 | 0.52 | 6.98 | 15.58 | 2.50 | 17.34 | 8.15 | 13.40 | 7.87 |

| | ftv47 | ftv55 | ftv64 | ftv70 | ftv100 | ftv110 | ftv120 | ftv130 | ftv140 |
|---|---|---|---|---|---|---|---|---|---|
| Dim. | 48 | 56 | 65 | 71 | 101 | 111 | 121 | 131 | 141 |
| GR | 26.01 | 22.57 | 26.54 | 28.21 | 27.29 | 25.54 | 30.89 | 32.16 | 33.18 |
| $R$-$R$TBG | 17.12 | 7.59 | 8.81 | 20.46 | 29.64 | 27.22 | 18.65 | 18.25 | 19.38 |
| $A$-$R$TBG | 9.74 | 11.13 | 2.01 | 4.51 | 15.44 | 6.59 | 7.57 | 6.20 | 6.07 |
| $A$-$R_1$TBG | 5.80 | 10.57 | 8.48 | 19.54 | 5.26 | 3.52 | 12.19 | 9.06 | 4.01 |

| | ftv150 | ftv160 | ftv170 | kro124p | rbg323 | rbg358 | rbg403 | rbg443 |
|---|---|---|---|---|---|---|---|---|
| Dim. | 151 | 161 | 171 | 100 | 323 | 358 | 403 | 443 |
| GR | 36.92 | 37.20 | 37.06 | 21.01 | 7.16 | 8.00 | 0.65 | 1.10 |
| $R$-$R$TBG | 13.33 | 15.17 | 16.55 | 17.86 | 16.59 | 30.18 | 41.91 | 31.76 |
| $A$-$R$TBG | 4.83 | 5.29 | 4.72 | 11.53 | 0 | 0 | 0 | 0 |
| $A$-$R_1$TBG | 3.60 | 4.32 | 6.61 | 6.66 | 0 | 0 | 0 | 0 |

It is worth mentioning that the construction heuristics from Table 1 in ([8]) have the following qualities in average over all classes of instances 1 to 7: GR= 580.35%, RI= 710.95%, KSP= 135.08%, GKS= 98.09%, RPC= 102.02%, COP= 23.01% while for our algorithms $R$-$R$TBG= 67.14%, $A$-$R$TBG= 34.75%, and $A$-$R_1$TBG= 29.19%.

## 6 Conclusions and future research directions

The tolerance based b-n-b framework suggests a new metaheuristic, proves to be a powerful methodology for creating new greedy type algorithms and fast finding high quality solutions for the ATSP instances; it has the robustness and consistency required by large-scale applications.

Our experimental results for tolerance based greedy (TBG) type heuristics question the above mentioned messages. For example, our $R_1$-TBG algorithm (see Sections 4 and 5 in this paper) applied to the most popular randomly generated asymmetric instances (class 4 in our paper) returns TBG solutions the quality of which better in average by factor 610.5 compared to usual CBG solutions and took approximately 13.11 times longer than CBG greedy heuristic. Our experiments with the same $R_1$-TBG algorithm on the GYZ instances (class 8) used in the theoretical analysis (see [13]) show that we are able to solve all these instances with optimal length or with optimal length + 1, whereas the GR always finds the worst tour. Moreover, our very simple $R$-$R$TBG algorithm applied to the class 4 finds TBG solutions with better quality in average by factor 51.62 (compared to usual CBG solutions) and took approximately just 0.66 times shorter than CBG greedy heuristic.

All TBG heuristics presented in this paper have in average better quality than the following well known cost based construction heuristics: Greedy (GR), Random Insertion (RI), Karp-Steele patching (KSP), Modified Karp-Steele patching (GKS), Recursive Path Contraction (RPC), and our $A$-$R_1$TBG heuristic is competitive with the Contract or Patch (COP) heuristic (see [8]).

The simplicity of our $R$-$R$TBG algorithm shows that this algorithm can be recommended for practical usage as an online algorithm which can be used by a vehicle driver for finding high quality Hamiltonian cycles.

An interesting direction of research is to construct different classes of tolerance based heuristics (for example, construction, improvement etc.). Moreover, by using the suggested metaheuristic for presenting the GR we have opened a way for creating tolerance based heuristics for many other combinatorial optimization problems defined on the set of all permutations, for example the Linear Ordering [4], Quadratic and Multidimensional Assignment [22] problems. Finally, an open question: find the domination numbers of $X$-$Y$TBG algorithms with $X, Y \in \{R, A\}$.

## 7 Acknowledgements

This work is done when both authors have enjoyed the hospitality of Applied Mathematics Department, Khmelnitsky National University, Ukraine. We would like to thank all colleagues from this department including V. G. Kamburg, S. S. Kovalchuk, and I. V. Samigulin. The research of both authors was supported by a DFG grant, Germany and SOM Research Institute, University of Groningen, the Netherlands.

## References

[1] J. Bang-Jensen, G. Gutin, A. Yeo. When the greedy algorithm fails. Discrete Optimization **1**, 121–127, 2004.

[2] E. Balas, M.J. Saltzman. An algorithm for the three-index assignment problem. Oper. Res. **39**, 150–161, 1991.

[3] N. Chakravarti, A. P. M. Wagelmans. Calculation of stability radii for combinatorial optimization problems. Oper. Res. Lett. **23**, 1–7, 1999.

[4] S. Chanas, P. Kobylanski. A new heuristic algorithm solving the linear ordering problem. Comput. Optim. and Appl. **6**, 191–205, 1996.

[5] M. Fischetti, A. Lodi, P. Toth. Exact methods for the asymmetric traveling salesman problem. Chapter 2 in: The Traveling Salesman Problem and Its

Variations. G. Gutin, A.P. Punnen (Eds.). Kluwer, Dordrecht, 169–194, 2002.

[6] D.S. Johnson, L.A. McGeoch. Experimental analysis of heuristics for the STSP. Chapter 9 in: The Traveling Salesman Problem and Its Variations. G. Gutin, A.P. Punnen (Eds.). Kluwer, Dordrecht, 369–444, 2002.

[7] 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.

[8] F. Glover, G. Gutin, A. Yeo, A. Zverovich. Construction heuristics for the asymmetric TSP. European J. Oper. Res. **129**, 555–568, 2001.

[9] F. Glover, A. Punnen. The traveling salesman problem new solvable cases and linkages with the development of approximation algorithms. J. Oper. Res. Soc. **48**, 502–510, 1997.

[10] B. Goldengorin, G. Sierksma. Combinatorial optimization tolerances calculated in linear time. SOM Research Report 03A30, University of Groningen, Groningen, The Netherlands, 2003 (http://www.ub.rug.nl/eldoc/som/a/03A30/03a30.pdf).

[11] 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. Hromkovic J., Nagl M., Westfechtel B. (Eds.), Lecture Notes in Comput. Sci. **3353**, 222–234, 2004.

[12] H. Greenberg. An annotated bibliography for post-solution analysis in mixed integer and combinatorial optimization. In: D. L. Woodruff (Ed.). Advances in computational and stochastic optimization, logic programming, and heuristic search. Kluwer Academic Publishers, Dordrecht, 97–148, 1998.

[13] G. Gutin, A. Yeo, A. Zverovich. Traveling salesman should not be greedy: domination analysis of greedy type heuristics for the TSP. Discrete Appl. Math. **117**, 81–86, 2002.

[14] G. Gutin, A. Yeo. Anti-matroids. Oper. Res. Let. **30**, 97-99, 2002.

[15] K. Helsgaun. An effective implementation of the Lin-Kernighan traveling salesman heuristic. European J. Oper. Res. **126**, 106–130, 2000.

[16] F.S. Hillier, G.J. Liberman. Introduction to Operations Research. Holden-Day, Inc. San Francisco, 1967.

[17] M. Libura. Sensitivity analysis for minimum hamiltonian path and traveling salesman problems. Discrete Appl. Math. **30**, 197–211, 1991.

[18] http://www.tsp.gatech.edu/d15sol/

[19] http://www.tsp.gatech.edu/apps/index.html

[20] D.L. Miller and J.F. Pekny. Exact solution of large asymmetric traveling salesman problem. Science **251**, 754–761, 1991.

[21] D. Naddef. Polyhedral theory and branch-and-cut algorithms for the symmetric TSP. Chapter 2 in: The Traveling Salesman Problem and Its Variations. G. Gutin, A.P. Punnen (Eds.). Kluwer, Dordrecht, 29–116, 2002.

[22] Nonlinear Assignment Problems. Algorithms and Applications. P.M. Pardalos and L.S. Pitsoulis (Eds.). Kluwer, Dordrecht, 2000.

[23] N.V. Reinfeld and W.R. Vogel. Mathematical Programming. Prentice-Hall, Englewood Cliffs, N.J., 1958.

[24] G. Reinelt. TSPLIB – a Traveling Salesman Problem Library. ORSA J. Comput. **3**, 376–384, 1991.

[25] E.A. Silver. An overview of heuristic solution methods. J. Oper. Res. Soc. **55**, 936–956, 2004.

[26] M. Turkensteen, D. Ghosh, B. Goldengorin, G. Sierksma. Tolerance-based Search for Optimal Solutions of NP-Hard Problems. Submitted.

[27] S. Van Hoesel, A. Wagelmans. On the complexity of postoptimality analysis of 0/1 programs. Discrete Appl. Math. **91**, 251–263, 1999.

[28] A. Volgenant. An addendum on sensitivity analysis of the optimal assignment. European J. Oper. Res. (to appear).

[29] L.A. Wolsey. Integer programming. John Wiley & Sons, Inc., New York, 1998.