Effective Heuristics for Large TSP Instances

Gerold Jäger

Christian Albrechts University Kiel
Computer Science Institute
Germany

German Research Foundation Grant
Tolerance Based Algorithms for Solving the
Traveling Salesman Problem

May 19, 2010
1 The Theory of Tolerances

- Motivation
- Relaxed Assignment Problem
- Definitions
- Theoretical Results
- Extensions
Traveling Salesman Problem

- Definition
- Importance
- Helsgaun’s Heuristic for the Symmetric TSP
- Greedy Heuristic for the Asymmetric TSP
Overview

3 Tolerance Based Greedy Heuristic for the Asymmetric TSP
   - Motivation
   - Description
   - Experimental Results

4 Pseudo Backbone Contraction Heuristic for the Symmetric TSP
   - Description
   - General Remarks
   - Experimental Results
   - Application on World-TSP Instance
The Theory of Tolerances

Joint work with

Boris Goldengorin, Paul Molitor.
Motivation

- Analysis of optimal solutions of combinatorial optimization problems
  - Restriction to minimization problems

- Questions:
  - How does the optimality of a given solution depend on the input data?
  - How stable are elements in an optimal solution?
  - How stable are elements outside of an optimal solution?

- Theory of tolerances
- Branch of sensitivity analysis
Applications

- Input data may be inexact or uncertain.
  
  How reliable is an optimal solution and conclusions thereof?

- Rather significant considerations may not have been built into the model due to the difficulty of formulating them

  Having solved the simplified model, how well does the optimal solution fit in with the other considerations?
Relaxed Assignment Problem

- **Given**: Complete directed graph \( G = (V, E), |V| = n \), cost function \( c : E \to \mathbb{R} \).

- **Relaxed Assignment Problem (RAP)**:
  - **Find**: set of \( n \) arcs so that
  - each vertex has exactly one out-arc
  - the overall costs of all arcs is minimized

- **Example**:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>−</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>35</td>
<td>−</td>
<td>12</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>35</td>
<td>13</td>
<td>−</td>
<td>18</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>35</td>
<td>13</td>
<td>19</td>
<td>−</td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>13</td>
<td>19</td>
<td>25</td>
<td>−</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>216</td>
<td>13</td>
<td>19</td>
<td>25</td>
<td>31</td>
<td>−</td>
</tr>
</tbody>
</table>

- **Solution arcs**: \((1, 2), (2, 3), (3, 2), (4, 2), (5, 2), (6, 2)\)
Combinatorial Minimization Problem

Combinatorial Minimization Problem consists of following parameters:

- finite ground set $\mathcal{E} = \{e_1, e_2, \ldots, e_k\}$
- cost function $c : \mathcal{E} \rightarrow \mathbb{R}$
- set of feasible solutions $D$, where $\forall S \in D : S \subseteq \mathcal{E}$
- minimization function $f_c : D \rightarrow \mathbb{R}$ of the form

\[
\forall S \in D : f_c(S) = \sum_{e \in S} c(e)
\]

- Problem: Find a feasible solution $S \in D$ with minimum costs.

- Examples of combinatorial minimization problems:
  - Traveling Salesman Problem
  - Minimum Spanning Tree Problem
  - Assignment Problem
  - Relaxed Assignment Problem
**Upper Tolerance**

- **Given:** Comb. minim. problem, optimal solution $S^* \in D$ and $e \in \mathcal{E}$.
  - Fix the costs $c(\bar{e})$ of all elements $\bar{e} \in \mathcal{E} \setminus \{e\}$.
  - Vary the costs $c(e)$ of $e \in \mathcal{E}$.

- **Case 1:** $e \in S^*$
  - **Upper Tolerance** of $e$: Supremum by which the costs of $e$ can be increased so that $S^*$ remains optimal.

- **Example:** RAP

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>35</td>
<td></td>
<td>12</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>35</td>
<td>13</td>
<td></td>
<td>18</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>35</td>
<td>13</td>
<td>19</td>
<td></td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>13</td>
<td>19</td>
<td>25</td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>216</td>
<td>13</td>
<td>19</td>
<td>25</td>
<td>31</td>
<td></td>
</tr>
</tbody>
</table>

- **Solution arcs:** $(1, 2), (2, 3), (3, 2), (4, 2), (5, 2), (6, 2)$

  ☞ **Upper tolerances** in brackets.
Lower Tolerance

Case 2: \( e \not\in S^* \)

Lower Tolerance of \( e \): Supremum by which the costs of \( e \) can be decreased so that \( S^* \) remains optimal.

Example: RAP

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>–</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>35</td>
<td>–</td>
<td>12</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>35</td>
<td>13</td>
<td>–</td>
<td>18</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>35</td>
<td>13</td>
<td>19</td>
<td>–</td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>13</td>
<td>19</td>
<td>25</td>
<td>–</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>216</td>
<td>13</td>
<td>19</td>
<td>25</td>
<td>31</td>
<td>–</td>
</tr>
</tbody>
</table>

Solution arcs: (1, 2), (2, 3), (3, 2), (4, 2), (5, 2), (6, 2)

It holds:

\[ \ell((1, 4)) = 1 \]
\[ \ell((4, 5)) = 11 \]
Backbone

- **Given**: Combinatorial minimization problem
- **Backbone**: Element contained in each optimal solution
  - Essential component of many heuristics and exact algorithms
  - compare **Pseudo Backbone Contraction Heuristic** for the TSP
Upper and lower tolerances are well-defined, i.e., do not depend on the chosen optimal solution.

Backbones are exactly the elements with positive upper tolerance.

Elements, which are contained in all feasible solutions, are exactly the elements with upper tolerance $+\infty$.

Elements, which are contained in no feasible solution, are exactly the elements with lower tolerance $+\infty$.

Let no feasible solution be subset of another feasible solution. Then the minimal upper tolerance over all single elements equals the minimal lower tolerance over all single elements.
Computation of Upper Tolerance

Let $e \in \mathcal{E}$ be contained in at least one optimal solution.

$q \triangleq$ cost of an optimal solution.

$q_e \triangleq$ cost of an optimal solution with the additional condition that $e$ is not contained in it.

It holds:

$$u(e) = q_e - q$$
Further components of theory of tolerances:

1. Instead of additive changes consider multiplicative changes

2. Instead of tolerances of single elements consider set tolerances

3. Instead of minimization function

\[ \forall S \in D : f_c(S) = \sum_{e \in S} c(e) \]
consider minimization functions

\[ \forall S \in D : f_c(S) = \prod_{e \in S} c(e) \]
\[ \forall S \in D : f_c(S) = \prod_{e \in S} c(e), \text{ where } \forall e \in E : c(e) \in \mathbb{R}^+ \]
\[ \forall S \in D : f_c(S) = \max_{e \in S} \{c(e)\} \]
Definition

- **Given**: Complete graph \( G = (V, E) \), \(|V| = n\), cost function \( c : E \rightarrow \mathbb{R} \).

- **Traveling Salesman Problem (TSP)**:
  Find: tour \( (v_1, v_2, \ldots, v_n, v_1) \) with minimum costs

\[
c(v_n, v_1) + \sum_{i=1}^{n-1} c(v_i, v_{i+1})
\]

- **Symmetric TSP (STSP)**:
  Version of TSP, where it holds:

\[
c(v, w) = c(w, v) \quad \forall v, w \in V \quad (1)
\]

- **Asymmetric TSP (ATSP)**:
  Version of TSP, where (1) does not necessarily hold.
Importance

- **Easy** to understand.
- **Hard** to solve: $\mathcal{NP}$-hard. [Karp, 1972]
- Gap between **few theoretical results** and **outstanding practical results**
- Important techniques in combinatorial optimization first have been created for the TSP, e.g.,
  - Integer Programming
  - Branch-and-Cut
  - Local Search
- Many **important applications**:
  - public transport
  - tour planning
  - design of microchips
  - genome sequencing
Example: Shortest tour through 15,112 cities in Germany
Helsgaun’s Heuristic for the Symmetric TSP

- Main ideas: [Lin, Kernighan, 1971]
- Consists of two classical components:
  - **Construction** part:
    - builds a starting tour without attempting to improve it
    - usually very fast
  - **Improvement** part:
    - improves a given starting tour and receive a better one
    - typically slower than construction part
    - mostly local search techniques
- Application to ATSP possible:
  - Transformation of ATSP instance to STSP instance of doubled size
Helsgaun’s Heuristic for the Symmetric TSP

**Construction** part:
1. Start with an arbitrary vertex.
2. In each step go to the nearest non-visited vertex.
3. If all vertices are visited, return to the starting point.
4. Receive a **starting tour**.

**Improvement** part:
5. For $k \leq n$ apply a $k$-OPT step, i.e.:
   - Replace tour edges by non-tour edges, such that
     - the edges are still a tour
     - the tour is better than the last one

Example of a 2-OPT step

6. Repeat step 5 as long as improving steps can be found.
Optimizations:

1. Choose $k$ small.
2. For a given vertex consider only the $s$ best neighboring edges, the so-called candidate system.
   Helsgaun’s main improvement:
   For each vertex do not consider the $s$ shortest neighboring edges, but the $s$ neighboring edges with smallest lower tolerance with respect to the minimum spanning tree.
3. Apply $t$ (nearly) independent runs of the heuristic.

The larger the algorithm parameters $k$, $s$ and $t$ are, the slower, but more effective (regarding quality) is Helsgaun’s Heuristic.
Greedy Heuristic for the Asymmetric TSP

- Fast Construction Heuristic for the ATSP: [Glover, Gutin, Yeo, Zverovich, 2001]

1. Determine the arc with smallest costs in the graph.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>35</td>
<td></td>
<td>12</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>35</td>
<td>13</td>
<td></td>
<td>18</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>35</td>
<td>13</td>
<td>19</td>
<td></td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>13</td>
<td>19</td>
<td>25</td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>216</td>
<td>13</td>
<td>19</td>
<td>25</td>
<td>31</td>
<td></td>
</tr>
</tbody>
</table>

2. Contract both vertices of this arc to a new vertex.
Greedy Heuristic for the Asymmetric TSP

1. Repeat 1. and 2., until we have only two vertices.

\[
\begin{array}{c|ccc|}
(1,2,3) & 4 & 5 & 6 \\
\hline
(1,2,3) & - & 18 & 19 \quad 19 \\
4 & 35 & - & 24 \quad 25 \\
5 & 35 & 25 & - \quad 30 \\
6 & 216 & 25 & 31 \quad - \\
\end{array}
\]
Connect these two vertices to a cycle.

Replace all vertices by its contracted paths.

Receive the output tour.

**Here:** Output tour \((1, 2, 3, 4, 5, 6, 1)\) with costs **306**.

(worst possible tour)
Joint work with

Diptesh Ghosh, Boris Goldengorin, Gregory Gutin.
Motivation

- **Greedy Heuristic:** The arc with smallest cost is chosen.
- **Problem:** Some vertices may have large costs to all other vertices.
- **Idea:** For a vertex the difference of the second-smallest cost to the smallest cost is more important than the value of the cost itself.

Choose the contracted arc based not on the smallest cost,
but on the largest upper tolerance to the Relaxed Assignment Problem.
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>35</td>
<td></td>
<td>12</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>35</td>
<td>13</td>
<td></td>
<td>18</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>35</td>
<td>13</td>
<td>19</td>
<td></td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>13</td>
<td>19</td>
<td>25</td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>216</td>
<td>13</td>
<td>19</td>
<td>25</td>
<td>31</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>3</th>
<th>(4,2)</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>35</td>
<td></td>
<td>18</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>(4,2)</td>
<td>35</td>
<td>12</td>
<td></td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>19</td>
<td>25</td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>216</td>
<td>19</td>
<td>25</td>
<td>31</td>
<td></td>
</tr>
</tbody>
</table>
Here: Output Tour \((1, 6, 4, 2, 5, 3, 1)\) with costs 112. (instead of 306)
Experimental Results

Experiments:

- Original Greedy Heuristic for the ATSP $\leadsto$ ORIG-GR
- Tolerance Based Greedy Heuristic for the ATSP $\leadsto$ TOL-GR

for 9 different instance classes:

- symmetric and asymmetric instances
- random and real instances
- instances whose cost values strongly and weakly differ regarding

  - time
  - quality
Experimental Results

- Results:
  - TOL-GR only slightly larger time
  - TOL-GR considerably better quality

- Example class:
  - 160 asymmetric instances of sizes 100 to 3000 with $c_{i,j}$ randomly and uniformly chosen from $\{0, 1, \ldots, i \cdot j\}$ for $1 \leq i \neq j \leq n$.
  - Average excess over best known lower bound:
    - ORIG-GR: 1135.27 %
    - TOL-GR: 22.23 %
  - Average time:
    - ORIG-GR: 0.93 seconds
    - TOL-GR: 0.99 seconds

- Conclusion:
  - TOL-GR better construction heuristic than ORIG-GR
Joint work with

Changxing Dong, Paul Molitor, Dirk Richter.
1. Using known heuristics, e.g., Helsgaun’s Heuristic, find good starting tours.
Find all common edges in these starting tours.

Such edges are called pseudo backbone edges.

3. **Contract** all edges of paths of pseudo backbone edges to one edge.

4. Create a new (reduced) instance by omitting the vertices, which lie on a path of pseudo backbone edges:

   Fix the contracted edges, i.e., force them to be in the final tour.
Apply Helsgaun’s Heuristic to the new instance.
Re-contract the tour of the new instance to a tour of the original instance.

Here: The last tour is the optimum one. Compare:
General Remarks

- Two advantages:
  1. Reduction of the set of vertices.
  2. Fixing of a part of the edges.

⇒ Helsgaun’s Heuristic can be applied with larger algorithm parameters $k$, $s$ and $t$ than for the original instance.

- The heuristic works rather good, if the starting tours are
  1. good ones
  2. not too similar
     (as otherwise the search space is restricted too strongly)
Experimental Results

Competition: TSP homepage
http://www.tsp.gatech.edu/

- Large TSP datasets from practice: for comparison of exact algorithms and heuristics.
- 74 unsolved example instances: VLSI and national instances
- For 18 of 74 instances we have set a new record.
- 10 of 18 records are still up to date.
Experimental Results

Our new records

<table>
<thead>
<tr>
<th>Date</th>
<th># Vertices</th>
<th>Date</th>
<th># Vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>24.05.2006</td>
<td>6,880</td>
<td>14.07.2008</td>
<td>28,534</td>
</tr>
<tr>
<td>22.10.2006</td>
<td>19,289</td>
<td>14.07.2008</td>
<td>39,603</td>
</tr>
<tr>
<td>20.03.2008</td>
<td>17,845</td>
<td>14.07.2008</td>
<td>56,769</td>
</tr>
<tr>
<td>28.03.2008</td>
<td>13,584</td>
<td>19.07.2008</td>
<td>104,815</td>
</tr>
<tr>
<td>01.04.2008</td>
<td>19,402</td>
<td>14.08.2008</td>
<td>52,057</td>
</tr>
<tr>
<td>27.05.2008</td>
<td>38,478</td>
<td>14.08.2008</td>
<td>238,025</td>
</tr>
<tr>
<td>23.06.2008</td>
<td>21,215</td>
<td>10.11.2008</td>
<td>29,514</td>
</tr>
<tr>
<td>30.06.2008</td>
<td>34,656</td>
<td>26.08.2009</td>
<td>32,892</td>
</tr>
</tbody>
</table>
Joint work with

Changxing Dong, Christian Ernst, Paul Molitor, Dirk Richter.
Most difficult instance of TSP homepage:
World-TSP instance with 1,904,711 cities.

(Ambitious) aim: Computation of a new record World-TSP tour.

Problem: Finding good starting tours is too difficult.
Idea: Combine Pseudo Backbone Contraction Heuristic with Partitioning.
Compute good tours in **overlapping windows**.

The overlap size is chosen half the width (height) of the window frame.

⇒ Each vertex is contained in exactly 4 windows, unless it is located near the boundary.
2. Use edges contained in all tours of 4 overlapping windows as pseudo backbone edges.

3. Apply the Pseudo Backbone Contraction Heuristic recursively.

Our current tour:
- 0.01% over the current best tour
- Found in 13 hours
Thanks for your attention!