

# Constricting Insertion Heuristic for Traveling Salesman Problem with Neighborhoods

**Sergey Alatartsev and Marcus Augustine and Frank Ortmeier**

Computer Systems in Engineering, Otto-von-Guericke University  
D-39106 Magdeburg, Germany  
{sergey.alatartsev, marcus.augustine, frank.ortmeier}@ovgu.de

## Abstract

Sequence optimization is an important problem in many production automation scenarios involving industrial robots. Mostly, this is done by reducing it to Traveling Salesman Problem (TSP). However, in many industrial scenarios optimization potential is not only hidden in optimizing a sequence of operations but also in optimizing the individual operations themselves. From a formal point of view, this leads to the Traveling Salesman Problem with Neighborhoods (TSPN). TSPN is a generalization of TSP where areas should be visited instead of points. In this paper we propose a new method for solving TSPN efficiently. We compare the new method to the related approaches using existing test benchmarks from the literature. According to the evaluation on instances with known optimal values, our method is able to obtain a solution close to the optimum.

## Introduction

Nowadays, industrial robots are programmed by a human with very precise monosemantic instructions that can be interpreted and performed only in a predefined way. Conventional programming requires exact positions of the robot end-effector to be specified (Pan et al. 2010). With modern tool-supported offline programming approaches, it is state-of-the-art to automatically generate trajectories from CAD data. In the example in Figure 1 one can see the robotic manufacturing of a toboggan<sup>1</sup>. All necessary cuttings can be automatically derived from the CAD model (right hand side of the figure) and translated into the corresponding movements in robot axis space. It is obvious that for efficient production an “optimal” sequence of all subtasks (i.e., cuttings that have to be done) is necessary. Assuming that each cut is defined by a starting/ending point, this problem can be translated to the Traveling Salesman Problem (TSP) (Applegate et al. 2007). The objective of TSP is to obtain a minimal-cost circle tour that visits all points once. A feasible solution could be the sequence of cuttings shown in the right part of Figure 1. There already exist multiple adaptations of TSP

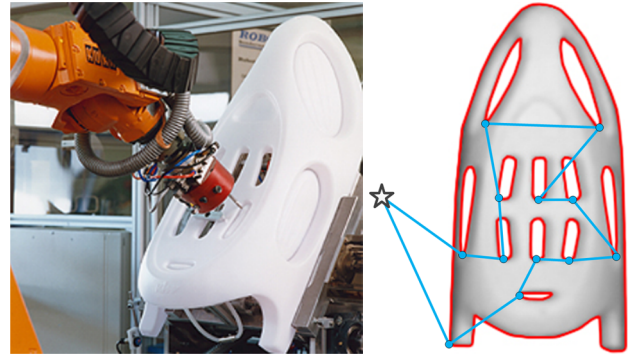


Figure 1: Cutting and deburring use case with a possible tour. The starting point is denoted with the star.

for robotics (e.g., (Baizid et al. 2010)). However, in reality many tasks do not require strong determinism and often allow a certain degree of freedom. In the example, it is not important where individual cuttings should start-end. Even a very short glance at Figure 1 shows that much shorter tours are possible if the starting-ending points of each single cutting may be moved along their contours. A formalization of this scenario may lead to a Traveling Salesman Problem with Neighborhoods (TSPN). TSPN is a generalization of TSP where points are substituted with areas and the objective is to find a minimal-cost tour through the set of regions that visits each of them once.

In this paper we propose a new method – Constricting Insertion Heuristic (CIH) – for solving TSPN problems even on large instances efficiently. Although the proposed planning approach is illustrated by its robotic application, it is not directly connected to robotics as none of the domain-specific knowledge is involved (i.e., kinematics, metrics in axis space, etc.). The main goal is to provide a general straightforward approach to solve TSPN. Therefore, CIH could be applied to any domain that can be modeled as TSPN.

In the remaining part of the paper, we will first discuss related approaches and then give a formal description of the problem as well as some important standard algorithms. Afterwards, we will explain CIH in detail. Then, an evaluation of the method and comparison with state-of-the-art al-

Copyright © 2013, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

<sup>1</sup>This is a real-world example taken from KUKA Roboter ([http://www.kuka-robotics.com/en/solutions/solutions\\_search/L\\_R131\\_Deburring\\_of\\_plastic\\_toboggans.htm](http://www.kuka-robotics.com/en/solutions/solutions_search/L_R131_Deburring_of_plastic_toboggans.htm)).

gorithms are presented. Finally, we conclude the paper and give an outlook to the future work.

## Related Work

The TSPN was introduced by Arkin and Hassin (1995). Later, it received significant attention in the domain of approximation algorithms (Mitchell 2010; Arora 2003). Furthermore, it has already been applied to the robotics domain. For example, Gentilini et al. (2011) applied this idea to a real-world use case where a robot with a hand-mounted camera had to take pictures of an object from different positions. They formulated this problem as TSPN and implemented a heuristic to speed up a Mixed-Integer Non-Linear Programming solver. This method shows good, close to optimum results. It was tested only for up to 16 regions, though in reality 50 or even more tasks are common in some industrial domains<sup>2</sup>. Our method is compared with this approach in the evaluation section.

TSP got much larger attention from researchers than TSPN. As a result, a large number of effective algorithms exists. A naive idea would be to try to use algorithms from TSP domain for TSPN. This could be done in several ways. One approach is to convert a TSPN problem to a Generalized TSP (GTSP)<sup>3</sup> by replacing the areas with a sets of points. GTSP is an extension of TSP when instead of planning between single points, the planning is done among sets of points. The objective is to find a minimum-cost path that passes through one point of each set. Oberlin et al.(2010) showed that it is possible to convert GTSP into a TSP. Though this process is possible in theory, it immensely increases the search space and becomes practically infeasible (Shi et al. 2007). Another drawback is that representing TSPN as GTSP will bring errors due to the discretization process of the areas.

Another way to apply algorithms from TSP domain to TSPN is to represent every area with one point. It transforms TSPN into two subproblems: TSP and Touring-a-sequence-of-Polygons Problem (TPP). TPP is a NP-hard problem where the goal is to find the shortest path that passes through a given sequence of areas (Dror et al. 2003).

Mennell (2009) proposed an approach for solving Close-Enough Traveling Salesman Problem (CETSP) which is a special case of TSPN. A minimal-cost tour should be found with the condition that the salesman may visit points within a certain distance (i.e., areas are disks for 2D space). A generic three-stage approach was proposed: (1) find intersections between disks, (2) represent every intersection area with a point and calculate TSP tour, (3) optimize the previously found point sequence with TPP. Multiple variations and combinations of different algorithms for these stages were investigated.

Elbassioni et al. (2009) proposed a method for GTSP (they refer to this problem as a Discrete TSPN) where areas are sorted by their diameter. Then, sequentially starting

<sup>2</sup>(Pan et al. 2012) showed an example with 500 goals in welding applications for two robots to perform (i.e., 250 goals for each).

<sup>3</sup>Often also referred to as One-of-a-Set TSP, Group TSP, Multiple Choice TSP or Covering Salesman Problem.

from the area with the smallest diameter an inner area point is picked up. The algorithm selects a new point as close as possible to already chosen points. After all areas are represented with inner points, a TSP tour is calculated. In this approach a certain optimization on point allocation is done first and then the TSP tour is calculated. It was assumed that the proposed method can be also applied for continuous TSPN (we refer to this problem simply as TSPN) under the weak assumption that the new closest point can be efficiently found in the infinite set of points.

In this paper, we develop a new idea to solve TSPN, which – in contrast to other approaches – simultaneously solves the sequencing (TSP) and allocation of points inside the areas (TPP).

## Formal Foundations

This section gives a brief introduction to the formalization of the relevant problems. It also explains general ideas of existing algorithms that are used as a foundation for our method.

### Definition of the Problems

In the following, we restrict ourselves to 2D space (i.e.,  $\mathbb{R}^2$ ) and use the Euclidean distance function  $d(p, q)$  to denote the cost of moving between two points  $p, q \in \mathbb{R}^2$ . Of course, higher dimensional spaces as well as other distance functions are possible.

**TSP:** The well-known Traveling Salesman Problem (TSP) is formalized as follows:

Given a weighted undirected graph  $G = (V, E)$ , where  $V$  is a set of  $n$  vertices and  $E$  is a set of edges. The objective is to find a minimal-cost cyclic tour  $T = (v_1, \dots, v_{n+1})$  that visits all vertices only once and  $v_1 = v_{n+1}$ .

**TPP:** Another related problem is the Touring-a-sequence-of-Polygons Problem (TPP). There are two types of TPP: floating and fixed. In the fixed TPP start and end points have to be defined and their positions are fixed. In the floating TPP start and end points are no longer required. The floating TPP is relevant to this paper and simply referred to as TPP. It is defined as follows:

Given a sequence of  $n$  polygons  $A = (A_1, \dots, A_n)$ , find a minimal-cost cyclic tour  $T = (p_1, \dots, p_{n+1})$ , such that it visits  $A_i$  in the point  $p_i$  and  $p_1 = p_{n+1}$ .

Note that for TPP an ordering of the polygons is required as an input data. This order is not changed during tour calculation.

**TSPN:** The Traveling Salesman Problem with Neighborhoods is formalized as follows:

Given a set of  $n$  polygons  $A = \{A_1, \dots, A_n\}$ , find a minimal-cost cyclic tour  $T = (p_1, \dots, p_{n+1})$ , such that it visits  $A_i$  in the point  $p_i$  and  $p_1 = p_{n+1}$ .

In contrast to TPP, TSPN requires a set of polygons instead of their sequence as input data.

A comparison of input and output parameters between all three problems is provided in Table 1.

| Problem | Input             | Optimize              | Output             |
|---------|-------------------|-----------------------|--------------------|
| TSP     | set of points     | sequence              | sequence of points |
| TPP     | sequence of areas | location of points    | sequence of points |
| TSPN    | set of areas      | location and sequence | sequence of points |

Table 1: The differences between TSP, TPP and TSPN.

### Involved Sub-Algorithms

Before presenting our algorithm for TSPN, we will explain the TSP and TPP heuristics which are involved in the method. Johnson et al. (1997) classified TSP heuristics by splitting them into tour-construction heuristics and tour-improvement heuristics. Tour-construction heuristics construct a feasible solution by following certain rules. Tour-improvement heuristics start with a feasible tour and try to reduce its cost by manipulating tour edges. The process stops when a stopping condition is met, e.g., no more improvement is possible or a maximum number of iteration is reached.

One of the most well-known algorithms for tour-construction is Insertion Heuristic (IH) (Hassin and Keinan 2008). The general structure is represented in Algorithm 1.

---

#### Algorithm 1: General Insertion Heuristic algorithm

---

**Input:** Weighted graph  $G = \{V, E\}$   
**Output:** Tour  $T = (v_1, \dots, v_n)$

Initialize sub-tour  $T$  with strategy  $S1$ ;  
**while**  $T$  is a partial tour **do**  
    Choose vertex  $v \notin T$  with strategy  $S2$ ;  
    Add  $v$  to the tour  $T$  with strategy  $S3$ ;  
**end**  
**return**  $T$ ;

---

$S1$  denotes a strategy to construct an initial tour. Often an initial tour is either a triangle tour (i.e., a tour consisting of three points) or a tour that follows the points that form the convex hull border of  $V$ .

$S2$  is a strategy to choose a point  $v$  that is not yet in the tour  $T$ .  $S3$  is a way how to choose a position in  $T$  where point  $v$  should be inserted. The strategies  $S2$  and  $S3$  are repeated while  $T$  is a partial tour, i.e., not all the points of  $V$  are in the tour.

After a tour is obtained, it is possible to apply tour-improvement heuristics, e.g., 2-Opt or 3-Opt (Helsgaun 2000). These are local search algorithms that are certain cases of the  $k$ -Opt algorithm. Basic idea is to delete  $k$  edges from the tour and reconnect it in all possible ways. The goal is to find the relocations that might decrease the tour cost.

As a TPP solver, so called Rubber-band algorithm (RBA) proposed by Pan et al. (2010), is applied. The general structure of RBA is presented in Algorithm 2. The basic idea of RBA is to construct a feasible tour  $T = (p_1, \dots, p_n)$  by allocating points inside the areas  $p_i \in A_i$  and then iteratively improve it. The improvement is obtained sequentially

for every area  $A_i$ , where a new point  $new\_p_i \in \partial A_i$  in the border of the area is computed in a way that the distance to its neighbors in the tour  $p_{i-1}$  and  $p_{i+1}$  is minimized. It is expressed in the line 4 in Algorithm 2. One iteration of the improvement cycle is finished when this procedure is performed for all areas. The RBA stops when a maximum number of iterations has been performed or a desired accuracy  $\varepsilon$  is reached, i.e., the difference between tour lengths on iteration  $j$  and  $j + 1$  is less than  $\varepsilon$ .

---

#### Algorithm 2: General Rubber-band algorithm structure

---

**Input:** Sequence of areas  $A = (A_1, \dots, A_n)$ , accuracy  $\varepsilon$   
**Output:** Tour  $T = (p_1, \dots, p_n)$

1 Construct a sequence  $T = (p_1, \dots, p_n)$  so that  $p_i \in A_i$ ;  
2 **while** Desired accuracy  $\varepsilon$  is not reached **do**  
3     **foreach**  $p_i \in T$  **do**  
4         Find  $new\_p_i \in \partial A_i$ , such that  
            $d(p_{i-1}, new\_p_i) + d(new\_p_i, p_{i+1}) =$   
            $min(d(p_{i-1}, p_i) + d(p_i, p_{i+1})), p_i \in \partial A_i$ ;  
5          $p_i \leftarrow new\_p_i$ ;  
6     **end**  
7 **end**  
8 **return**  $T$ ;

---

Following the TPP definition, solution of the problem is a tour consisting of  $n + 1$  points. Note that Algorithm 2 returns a tour that consists of  $n$  points. It does not contradict the definition, as it is easy to modify the tour by extending it with a point  $p_{n+1}$  that is equal to  $p_1$ . Therefore, both tours are assumed to be correct.

### Constricting Insertion Heuristic

This section presents our algorithm for solving TSPN: Constricting Insertion Heuristic (CIH). CIH considers TSPN as two sub-problems: TSP and TPP. In contrast to the existing approaches, CIH solves TSP and TPP simultaneously. In this context, constricting means that during the search for the optimal sequence with Insertion Heuristic, points are constricted to each other with RBA to optimize their location in the areas with respect to obtained sequence.

**Modification of RBA** As previously described, RBA takes two parameters as an input: a sequence of areas  $A$  and an accuracy value  $\varepsilon$ . In the algorithm an optimal sequence of points  $T = (p_1, \dots, p_n)$  is calculated by optimizing allocation of the points in the areas. We introduce mRBA which is a small modification of RBA where a sequence of points  $T$  is included as additional parameter in input, i.e., line 1 in Algorithm 2 is omitted.

Searching for  $new\_p_i$  is a geometrical task and could be performed with any optimization technique. In this paper we use the one-dimensional Golden Section Search optimization method (Press et al. 2007). It searches in the interval from 0 to 360 degrees, and for every angle to check, the point on the border of the figure is calculated. Optimization is done until accuracy  $\mu$  is reached. Therefore, mRBA requires four input parameters:  $A, P, \varepsilon$  and  $\mu$ .

**Modification of Insertion Heuristic** Constricting Insertion Heuristic is obtained from Insertion Heuristic by modifying the strategies in Algorithm 1. Specialized implementation of the insertion strategies makes CIH capable to solve TSPN efficiently. CIH is shown in detail in Algorithm 3.

---

**Algorithm 3:** Constricting Insertion Heuristic

---

**Input:** Set of areas  $A = \{A_1, \dots, A_n\}$ , desired accuracies  $\varepsilon, \mu$

**Output:** Tour  $T = (p_1, \dots, p_n)$

```

1 Construct a set  $P = \{p_1, \dots, p_n\}$  so that  $p_i \in A_i$ ;
2  $T \leftarrow \text{ConvexHullBorderTour}(P)$ ;
3  $R \leftarrow (P - \text{ConvexHullBorderTour}(P))$ ;
4 if  $R = \emptyset$  then
5    $T \leftarrow \text{mRBA}(A, T, \varepsilon, \mu)$ ;
6   return  $T$ ;
7 end
8 while  $R \neq \emptyset$  do
9    $p_{temp} \leftarrow \underset{p_q}{\operatorname{argmin}}(d(p_q, t_j))$ , where  $p_q \in R, t_j \in T$ ,
    $q \in [1, \text{Count}(R)], j \in [1, \text{Count}(T)]$ ;
10   $L \leftarrow \infty$ ;
11   $T_{temp} \leftarrow T$ ;
12  for  $i=1$  to  $\text{Count}(T)$  do
13     $\text{Insert}(T_{temp}, p_{temp}, i)$ ;
14     $T_{temp} \leftarrow \text{mRBA}(A, T_{temp}, \varepsilon, \mu)$ ;
15    if  $\text{Length}(T_{temp}) < L$  then
16       $L \leftarrow \text{Length}(T_{temp})$ ;
17       $\text{insert\_index} \leftarrow i$ ;
18    end
19     $\text{Remove}(T_{temp}, p_{temp})$ ;
20  end
21   $\text{Insert}(T, p_{temp}, \text{insert\_index})$ ;
22   $\text{Remove}(R, p_{temp})$ ;
23   $T \leftarrow \text{mRBA}(A, T, \varepsilon, \mu)$ ;
24 end
25 return  $T$ ;

```

---

Several functions are involved during calculation. Here,  $\text{ConvexHullBorderTour}(P)$  is a function that returns a tour consisting of points that form the *border* of the convex hull of the set  $P$ . One example of such output is illustrated on part 2 in Figure 2. Function  $\text{Length}(T)$  takes a sequence of points  $T$  and returns the tour length. Function  $\text{Count}(R)$  returns the number of elements in the set  $R$ . Function  $\text{Insert}(T, p, i)$  inserts point  $p$  to the tour  $T$  at position after the element with index  $i$ . Function  $\text{Remove}(T, p)$  removes point  $p$  from the tour  $T$ .

The algorithm takes a set of areas  $A$  as an input. In the line 1 of Algorithm 3, set  $P$  is constructed in a way that point  $p_i$  belongs to area  $A_i$ <sup>4</sup>. In the lines 2 and 3 in Algorithm 3 a set of points  $P$  is split into two subsets:  $T$  and  $R$ .  $T$  is a tour that follows the points that form a convex hull border of  $P$ .  $R$  consists of all the remaining points from  $P$  that are not in the border of convex hull.

<sup>4</sup>In the following test instances,  $p_i$  is a geometrical center of the ellipses.

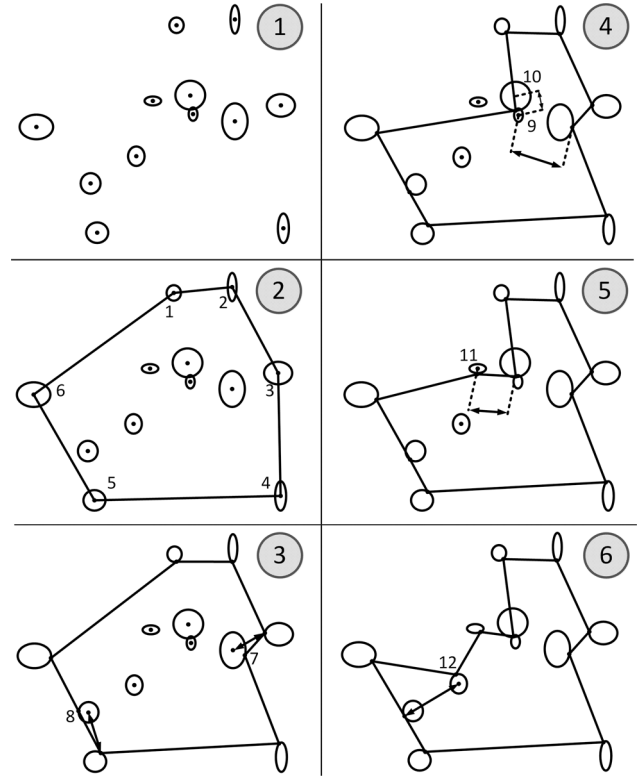


Figure 2: Workflow of CIH on test instance “tspn2DE12.2”

It could be the case that all points from  $P$  belong to the border of convex hull. This check is performed in line 4 of Algorithm 3. In that case  $T$  is already the desired sequence of areas to visit and mRBA is applied to find the optimal point allocation within the obtained sequence.

In case if not all points from  $P$  belong to the border of the convex hull set, points from the remainder  $R$  are selected one by one and inserted in the tour  $T$ . Line 9 in Algorithm 3 reflects the strategy  $S2$  where the point  $p_{temp}$  is chosen so that it is the closest point to one of the points from  $T$ .

Strategy  $S3$  (lines 10-23) in Algorithm 3 is a sequential insertion of  $p_{temp}$  to all possible positions within tour  $T$  performing mRBA algorithm every time and measuring the tour distance. After all combinations are checked,  $p_{temp}$  is inserted to the position that gives a minimal increase of the tour length and mRBA is performed. Afterwards,  $p_{temp}$  is deleted from the remainder set  $R$ . The algorithm stops when all points from  $R$  are inserted to  $T$ . The obtained tour  $T$  is the desired tour.

**Example of CIH Workflow** In the following, a test instance of TSPN with 12 ellipses is solved by CIH.

CIH is a tour construction heuristic and starts from any arbitrary sub-tour. However, it is more efficient to take points that form the border of convex hull as an initial tour. Therefore, 6 points are added to the tour on part 2 of Figure 2. If all points are inserted, their optimal sequence is found and only their locations within the areas should be optimized by

mRBA. However, in this example 6 points are left. Therefore, new points 7–12 are added one by one to the tour so that a point which is the nearest to the tour is picked up. The nearest distance is denoted with an arrow in Figure 2. For example, on part 3 point 7 is picked up as it is the nearest to the point 3 in the existing tour in part 2. Further, point 8 is added to the tour, because it is the nearest point to the point 5, which is already in the tour. The parts 2–6 in Figure 2 show how the algorithm adds new points to the tour one after the other.

Note that some iterations are combined at one part of Figure 2 (e.g. points 7 and 8 in part 3) as the picture was not changed visually.

CIH stops when there is no point left outside of the tour. In this example, an optimal solution was obtained as it is illustrated on part 6 of Figure 2. An animated process of CIH is presented online (Alatartsev, Augustine, and Ortmeier 2012).

**Solution Improvement** Obviously, TSP tour-improvement heuristics could improve TSPN solution by changing the sequence. However, this improvement may cause such a case that the location of the points in the areas becomes not optimal in regard to the new obtained sequence. Therefore, mRBA is applied afterwards.

Within this paper 2-Opt and 3-Opt algorithms are used as tour-improvement methods for TSP and mRBA for TPP. We denote a combination of 2-Opt and mRBA by *2-Impr.* and a combination of 3-Opt and mRBA by *3-Impr.* respectively.

## Evaluation

This section presents an evaluation of CIH on three sets of instances. The first set of instances (with up to 16 areas) was used to compare CIH with the optimal values. The second test is an evaluation of the algorithm on “stretched” areas (up to 70 areas). Finally, the results of CIH for CETSP instances with a large number of disks (up to 595 areas) are shown.

**Evaluation of CIH on Test Instances with Known Optimum** For the evaluation of CIH on the test instances with apriori known optimum values, instances developed by (Gentilini, Margot, and Shimada 2011) are used. The tests are available on-line<sup>5</sup> with a precise description. The test “*tspn2DE7\_N*” is decoded as a *2D* test with 7 ellipses. *N* can be “1” or “2” that reflects the box size circumscribing the ellipse. Ellipses with the box size “1” are larger than with box size “2”. The results of the evaluation are presented in Table 2.

Using these instances, we compared CIH with the optimal values (calculated by (Gentilini, Margot, and Shimada 2011)) and two different approaches. One algorithm was proposed by Gentilini et al.(2011) that improves Mixed-Integer NonLinear Program (MINLP) solver by introducing a heuristic. In the following, we will refer to this approach as Heuristic in Solver (HiS).

Another conception to compare with is the idea proposed by Mennell et al. (2009) where, first of all, every area is represented with a point and then sequentially TSP and TPP are solved. The original approach called LK-SOCP uses the Lin-Kernighan (LK) heuristic for solving TSP and second-order cone program (SOCP) for TPP. In the remaining part we refer to this method as TSP→TPP. We are not aiming at replication of all nuances of the original approach but rather on a comparison of the concept of applying methods for TSP and TPP. In other words, we check whether TSP and TPP should be solved sequentially one after the other or in parallel as in our method. To make an unbiased comparison between CIH and TSP→TPP, the same TPP solver as in the CIH is used. As a TSP solver Nearest Neighbor (NN) search improved with 3-Opt method is applied.

Within these tests, accuracy  $\mu$  for point retrieval at the border of an area and accuracy  $\varepsilon$  for RBA are set to 0.01. It is possible to tune these parameters in order to significantly speed up the calculations. For example, a solution with the cost value 393.054 for the test “*tspn2DE16\_1*” was found by CIH with  $\varepsilon = 0.01$  and  $\mu = 0.01$  within 44.67 *ms*. However,  $\varepsilon = 0.1$  and  $\mu = 0.1$  lead to a result with the cost value 393.058 within 30.82 *ms*.

As both CIH and TSP→TPP could be understood as tour construction heuristics, for both of them tour-improvement heuristic *3-Impr.* is applied during these test runs. However, on these test instances, improvement algorithm has no effect for TSP→TPP. Therefore, we do not include this column into the results table.

For the evaluation we used an Intel Core 2 Quad CPU, 2.83 Ghz with 8 Gb RAM, running Windows Vista. The computational time of HiS was calculated by Gentilini et al. (2011) with different hardware and software – Intel Xeon, 3.33 GHz CPU with 12 GB of RAM, running Fedora. Regardless of hardware and software differences, CIH is faster than HiS at average in 37 times<sup>6</sup>. Though this number should not be understood as unbiased comparison, it shows that significant improvement in computational time has been achieved.

We found out that errors produced by TSP→TPP are not caused by the TPP solver (except “*tspn2DE9\_2*”) but rather by a bad representation of the areas, since the tours produced by TSP solver are optimal. TSP→TPP solved 13 instances out of 24 to optimality. HiS solved 15 tests out of 24 to optimality with an average error of 0.15%. Though the underlying principal of CIH is greediness, in practice this method provides good results. CIH turns out to be optimal in 20 tests out of 24. Applying *3-Impr.* allowed us to decrease the average error from 0.33% to 0.20%.

**Evaluation of CIH on Test Instances with “Stretched” Ellipses** As previous test instances were limited to 16 areas, we developed a set of instances up to 60 areas. Tests are available on-line (Alatartsev, Augustine, and Ortmeier 2012). In this test we show the efficiency of the CIH for the ellipses that have different ratio between their axis radii, i.e., stretched along one of the axis.

<sup>5</sup>STSPN Instances: <http://wpweb2.tepper.cmu.edu/fmargot/ampl.html>

<sup>6</sup>The average time for all instances in this test for HiS is 650.42 *ms* and the time for CIH is 17.26 *ms*.

| Instance    | Optimal value | HiS      |         | TSP→TPP  |         | CIH      |         | CIH (3-Impr.) |         |
|-------------|---------------|----------|---------|----------|---------|----------|---------|---------------|---------|
|             |               | error(%) | $t(ms)$ | error(%) | $t(ms)$ | error(%) | $t(ms)$ | error(%)      | $t(ms)$ |
| tspn2DE5_1  | 191.255       | 0.00     | 140     | 0.00     | 0.24    | 0.00     | 1.39    | 0.00          | 1.73    |
| tspn2DE5_2  | 219.307       | 0.00     | 130     | 0.00     | 0.23    | 0.00     | 0.93    | 0.00          | 1.09    |
| tspn2DE6_1  | 202.995       | 0.00     | 240     | 0.00     | 0.41    | 0.00     | 1.49    | 0.00          | 1.82    |
| tspn2DE6_2  | 248.860       | 0.00     | 180     | 0.00     | 0.38    | 0.00     | 1.45    | 0.00          | 1.7     |
| tspn2DE7_1  | 201.492       | 0.00     | 300     | 0.00     | 0.63    | 0.00     | 6.46    | 0.00          | 6.05    |
| tspn2DE7_2  | 239.788       | 0.00     | 250     | 0.98     | 0.61    | 0.00     | 2.86    | 0.00          | 2.89    |
| tspn2DE8_1  | 190.243       | 0.00     | 370     | 0.07     | 0.57    | 0.28     | 0.46    | 0.07          | 0.79    |
| tspn2DE8_2  | 229.150       | 0.01     | 400     | 0.00     | 0.87    | 0.00     | 5.34    | 0.00          | 5.71    |
| tspn2DE9_1  | 259.290       | 0.00     | 400     | 4.23     | 1.34    | 0.00     | 8.59    | 0.00          | 9.14    |
| tspn2DE9_2  | 262.815       | 0.00     | 410     | 2.05     | 1.16    | 0.00     | 7.04    | 0.00          | 9.05    |
| tspn2DE10_1 | 225.126       | 0.00     | 410     | 0.15     | 1.30    | 0.00     | 8.5     | 0.00          | 11.12   |
| tspn2DE10_2 | 273.192       | 0.21     | 350     | 0.21     | 1.55    | 0.00     | 8.82    | 0.00          | 8.61    |
| tspn2DE11_1 | 247.886       | 0.75     | 630     | 0.69     | 2.19    | 0.00     | 12.21   | 0.00          | 16.07   |
| tspn2DE11_2 | 258.003       | 0.00     | 390     | 0.00     | 2.20    | 0.00     | 12.36   | 0.00          | 12.28   |
| tspn2DE12_1 | 265.858       | 0.00     | 550     | 0.00     | 2.38    | 0.00     | 14.9    | 0.00          | 19.61   |
| tspn2DE12_2 | 312.493       | 0.50     | 860     | 2.62     | 2.21    | 0.00     | 19.16   | 0.00          | 18.99   |
| tspn2DE13_1 | 278.876       | 0.00     | 1150    | 0.00     | 4.70    | 0.00     | 24      | 0.00          | 26.19   |
| tspn2DE13_2 | 324.271       | 0.20     | 490     | 0.20     | 4.67    | 0.00     | 22.78   | 0.00          | 24.62   |
| tspn2DE14_1 | 310.794       | 0.00     | 950     | 0.00     | 12.45   | 0.00     | 37.66   | 0.00          | 44.47   |
| tspn2DE14_2 | 270.638       | 0.56     | 690     | 0.26     | 12.38   | 0.04     | 26.93   | 0.04          | 29.3    |
| tspn2DE15_1 | 289.716       | 0.22     | 1080    | 0.00     | 4.69    | 0.00     | 45.2    | 0.00          | 48.69   |
| tspn2DE15_2 | 293.357       | 0.01     | 1200    | 0.02     | 7.80    | 1.36     | 47.31   | 1.36          | 49.66   |
| tspn2DE16_1 | 369.945       | 1.09     | 2840    | 0.00     | 26.90   | 6.24     | 44.67   | 3.38          | 48.8    |
| tspn2DE16_2 | 295.130       | 0.00     | 1200    | 0.00     | 10.73   | 0.00     | 53.75   | 0.00          | 58.34   |
| Average:    |               | 0.15     | 650.42  | 0.48     | 4.27    | 0.33     | 17.26   | 0.20          | 19.03   |

Table 2: Evaluation of CIH on TSPN instances from 5 to 16 ellipses

The test instances are calculated according to the following principle. At first, centers of ellipses  $(x_i, y_i)$  are calculated as a random integer number laying in the interval  $[0, 100]$ . The coefficient of elongation  $CE$  is calculated as uniformly distributed float number in the interval of  $[A, B]$ , where  $A$  and  $B$  are real positive numbers. For the generation of tests for this section, intervals  $[1, 1]$ ,  $[1, 5]$  and  $[1, 10]$  were used. The radius along  $X$ -axis is calculated as  $R_x = 100/N \times 2 \times Rand$ , where  $N$  is a desired number of areas in the test and  $Rand$  is a random real number that lies in  $[0.1, 1]$ . The radius along the  $Y$ -axis is calculated as  $R_y = R_x \times CE$ . With a probability of 0.5,  $R_x$  and  $R_y$  are exchanged. This method allows to generate test instances with ellipses of different elongation along one of the axis. The name of the instance “60\_1\_5” should be understood as test with 60 ellipses with one of the axis radius stretched from 1 to 5 times in comparison to another axis radius.

On these test instances we compare CIH and its improvement CIH (3-Impr.) with TSP→TPP and its improved version TSP→TPP (3-Impr.). The best obtained value is the minimal value among results of the four analyzed approaches. In these tests, accuracies were set as follows:  $\varepsilon = 0.01$  and  $\mu = 0.01$ . The results of the evaluation are presented in Table 3.

In the following instances, CIH (3-Impr.) obtained 13 best values out of 18 instances. The average error to best obtained value is 0.45%. TSP→TPP (3-Impr.) reached 6 best values. The average error is increased with ellipses getting stretched along one of the axes. For all tests with parameter “1\_1” the average error is 0.44%, for “1\_5” it is 2.10% and

for “1\_10” it is 6.26%. This clearly reflects the intuitive idea that simple area representation experiences difficulties with “more complicated” areas.

Although both CIH (3-Impr.) and TSP→TPP call 3-Opt once, at a certain area size (after 40, 30, 50 areas for “1\_1”, “1\_5”, “1\_10” respectively) CIH (3-Impr.) unexpectedly finishes computation faster. This can be explained by the fact that in TSP→TPP the major time is spent on 3-Opt calculation as the initial tour is far from being optimal. On the contrary, in CIH (3-Impr.) the initial tour obtained by CIH is close to optimum. Therefore, 3-Opt makes less exchanges and requires less time.

A sequential application of the method for solving TSP and then TPP (TSP→TPP) gives worse results than solving both at a time (CIH). The reason is that during the TSP calculation the areas are represented as points (e.g., ellipse geometrical centers in the test instances) and information about the overall shape is ignored. The obtained tour could be optimal, but only with regard to the chosen points. Applying TPP afterwards will improve the solution by allocating points inside the areas. But at no point in time does the algorithm consider the effects of the area shapes on the sequence in which the areas are visited. Application of both methods at a time (CIH) on the other hand means that during the calculation of the TSP we also coherently optimize the point locations inside those areas. Therefore, it allows for the consideration of certain information about the area shapes into the process of TSP planning. Thus, the final sequence for CIH is better than for TSP→TPP, even if the same algorithms for TSP and TPP are applied in both strategies.

| Instance | Best obtained value | TSP→TPP  |         | TSP→TPP (3- <i>Impr.</i> ) |         | CIH      |         | CIH (3- <i>Impr.</i> ) |         |       |
|----------|---------------------|----------|---------|----------------------------|---------|----------|---------|------------------------|---------|-------|
|          |                     | error(%) | $t(ms)$ | error(%)                   | $t(ms)$ | error(%) | $t(ms)$ | error(%)               | $t(ms)$ |       |
| 1        | 20_1.1              | 320.720  | 0.00    | 17                         | 0.00    | 28       | 1.81    | 89                     | 1.81    | 102   |
| 2        | 20_1.5              | 313.497  | 0.00    | 20                         | 0.00    | 32       | 3.11    | 101                    | 3.11    | 106   |
| 3        | 20_1.10             | 276.793  | 7.50    | 56                         | 5.03    | 72       | 0.00    | 182                    | 0.00    | 193   |
| 4        | 30_1.1              | 383.578  | 0.06    | 283                        | 0.06    | 385      | 1.46    | 363                    | 0.00    | 408   |
| 5        | 30_1.5              | 316.922  | 10.69   | 545                        | 8.46    | 764      | 0.00    | 443                    | 0.00    | 500   |
| 6        | 30_1.10             | 321.188  | 6.50    | 493                        | 3.63    | 592      | 0.00    | 654                    | 0.00    | 709   |
| 7        | 40_1.1              | 421.339  | 0.83    | 1868                       | 0.83    | 2036     | 2.41    | 625                    | 0.00    | 1001  |
| 8        | 40_1.5              | 368.802  | 3.97    | 1571                       | 0.00    | 1956     | 0.00    | 1140                   | 0.00    | 1303  |
| 9        | 40_1.10             | 312.353  | 15.30   | 1112                       | 15.22   | 1482     | 0.75    | 1211                   | 0.00    | 1393  |
| 10       | 50_1.1              | 438.182  | 0.24    | 1948                       | 0.00    | 2389     | 4.27    | 1595                   | 0.15    | 2872  |
| 11       | 50_1.5              | 457.114  | 0.88    | 2839                       | 0.00    | 3848     | 2.12    | 1904                   | 2.12    | 2337  |
| 12       | 50_1.10             | 397.472  | 12.49   | 4004                       | 8.38    | 4763     | 3.34    | 2182                   | 0.00    | 3559  |
| 13       | 60_1.1              | 563.603  | 0.00    | 9653                       | 0.00    | 12424    | 7.99    | 2355                   | 0.86    | 6854  |
| 14       | 60_1.5              | 563.438  | 4.15    | 10966                      | 3.43    | 14018    | 0.38    | 2320                   | 0.00    | 3750  |
| 15       | 60_1.10             | 499.973  | 7.61    | 10231                      | 5.16    | 12077    | 3.60    | 2621                   | 0.00    | 6132  |
| 16       | 70_1.1              | 622.098  | 1.76    | 26472                      | 1.76    | 29201    | 3.39    | 3326                   | 0.00    | 16915 |
| 17       | 70_1.5              | 587.004  | 0.74    | 30646                      | 0.74    | 37051    | 3.43    | 3921                   | 0.00    | 17393 |
| 18       | 70_1.10             | 509.905  | 1.85    | 24631                      | 0.11    | 29740    | 0.02    | 4713                   | 0.00    | 6333  |
| Average: |                     |          | 4.14    | 7075                       | 2.93    | 8492     | 2.12    | 1652                   | 0.45    | 3992  |

Table 3: Evaluation of CIH on TSPN instances from 20 to 70 ellipses

### Evaluation of CIH on Test Instances for CETSP

CETSP is a special case of TSPN where arbitrary areas are substituted with disks. This simplification allows building several specific efficient algorithms that are able to solve CETSP fast. Nevertheless, we provide an evaluation of CIH on test instances developed by Mennell et al.(2009) for CETSP and available online <sup>7</sup>. The results are presented in Table 4.

In these scenarios accuracy  $\mu$  is 0.01. As these tests have large instances with up to 595 areas, some speed up techniques were involved: (1)  $p_{temp}$  is taken randomly in line 9 in Algorithm 3, (2) accuracy  $\varepsilon$  in line 14 in Algorithm 3 is 0.5 (it is 0.01 in all other mRBA calls).

During these tests 2-*Impr.* is applied as a tour-improvement heuristic. Here, the TSP improvement heuristic 3-Opt was substituted with the more simple 2-Opt, as it requires less computational time. For example, for the instance “bubbles5” with 251 areas CIH found a solution within 19.72% from best known value for 167.5s. 2-*Impr.* provided 18.82% from the best known value in 179.25s. 3-*Impr.* achieved 16.78% but required significantly more time – 2975.2 s.

Using the same test set, (Mennell 2009) evaluated 11 heuristics for CETSP. Extending this list with CIH (2-*Impr.*) and sort ascending by the average error we obtain the place of our heuristic. Although these heuristics were developed especially for CETSP, they take into account that areas are represented only as disks. This information allows to construct very effective methods. In contrast, CIH was developed for TSPN and capable to solve arbitrary shapes. Note, that the comparison will not be unbiased, as methods oriented on different problems are compared. Even so for “Bubbles1-3”, CIH (2-*Impr.*) took the 3rd place out of 12.

It turns out that CIH (2-*Impr.*) efficiency decreases when number of areas is more than 100, e.g., in the overall test “Bubbles1-9”, CIH (2-*Impr.*) took the 11th position. For the “concentricCircle” test, CIH (2-*Impr.*) took the 7th place out of 12 and for the “team” test, CIH (2-*Impr.*) took the 8th place out of 12.

Although CIH is a heuristic for TSPN, it is still capable to solve CETSP instances even better than some specialized CETSP heuristics.

**Evaluation of Precision Parameters Influence** CIH efficiency depends on the two parameters  $\varepsilon$  and  $\mu$ . Obviously, with decreasing of  $\varepsilon$  and  $\mu$  the time to solve the instance increases. Results are presented in Figure 3. The evaluation was held on the benchmarks from Table 2. Surprisingly, this caused a very insignificant error increase<sup>8</sup>. Dependence of the average error is shown in Figure 4. The error grows slowly from 0.33% for  $\varepsilon=0.01$  and  $\mu=0.01$  up to 0.4% when  $\varepsilon=30$  and  $\mu=30$ . This reflects that the algorithm is very stable on small instances even if the precision values are very inaccurate. Precision parameters have weak influence to the maximum error on small instances. It was kept unchanged (6.24% for “*tspn2DE16\_1*”) for all combinations of  $\varepsilon$  and  $\mu$  (except the case:  $\varepsilon=30$  and  $\mu=30$ ).

## Conclusion

The Traveling Salesman Problem with Neighborhoods was covered in this paper. Its importance was shown on a real world use case. We proposed a new method – Constricting Insertion Heuristic – that splits TSPN into the two sub-problems TSP and TPP and solves them in parallel.

CIH was evaluated on three different test instances. The results of the test with a small number of areas showed that

<sup>8</sup>We applied the mRBA algorithm after CIH to eliminate the error accumulation of the point location on the borders, which appears with large precision values.

<sup>7</sup><http://www.minlp.org/library/problem/index.php?i=65&lib=MINLP>

| Instance           | N of areas | Best obtained value | CIH      |         | CIH (2-Impr.) |         |
|--------------------|------------|---------------------|----------|---------|---------------|---------|
|                    |            |                     | error(%) | t(s)    | error(%)      | t(s)    |
| concentricCircles1 | 17         | 53.158              | 0.00     | 0.01    | 0.00          | 0.02    |
| concentricCircles2 | 37         | 153.132             | 5.32     | 0.38    | 5.32          | 0.40    |
| concentricCircles3 | 61         | 271.076             | 4.15     | 1.88    | 4.15          | 1.92    |
| concentricCircles4 | 105        | 454.457             | 4.71     | 9.23    | 3.13          | 9.56    |
| concentricCircles5 | 149        | 645.381             | 6.26     | 28.02   | 5.35          | 29.13   |
| Average            |            |                     | 4.09     | 7.90    | 3.59          | 8.21    |
| bubbles1           | 37         | 349.135             | 0.05     | 0.56    | 0.05          | 0.57    |
| bubbles2           | 77         | 428.279             | 0.85     | 5.20    | 0.84          | 5.34    |
| bubbles3           | 127        | 530.733             | 0.33     | 22.41   | 0.33          | 23.50   |
| bubbles4           | 185        | 829.888             | 10.87    | 67.50   | 10.51         | 188.40  |
| bubbles5           | 251        | 1062.335            | 19.72    | 167.45  | 18.82         | 179.25  |
| bubbles6           | 325        | 1383.139            | 13.87    | 355.69  | 10.46         | 409.91  |
| bubbles7           | 408        | 1720.214            | 19.52    | 693.41  | 15.64         | 819.52  |
| bubbles8           | 497        | 2101.373            | 20.47    | 1263.88 | 17.74         | 1826.49 |
| bubbles9           | 595        | 2426.274            | 27.18    | 2147.98 | 23.04         | 2803.94 |
| Average            |            |                     | 12.54    | 524.90  | 10.83         | 695.21  |
| team1_100          | 100        | 307.337             | 2.61     | 10.62   | 2.61          | 10.71   |
| team2_200          | 200        | 246.683             | 1.22     | 93.39   | 1.22          | 86.39   |
| team3_300          | 300        | 466.241             | 12.20    | 258.50  | 12.20         | 836.56  |
| team4_400          | 400        | 680.211             | 8.02     | 606.50  | 7.77          | 635.70  |
| team5_499          | 499        | 702.823             | 11.50    | 1068.40 | 10.60         | 1418.90 |
| team6_500          | 500        | 225.216             | 0.19     | 1168.12 | 0.17          | 1230.18 |
| Average            |            |                     | 5.96     | 534.25  | 5.76          | 703.07  |

Table 4: Evaluation of CIH on CETSP instances (up to 595 disks)

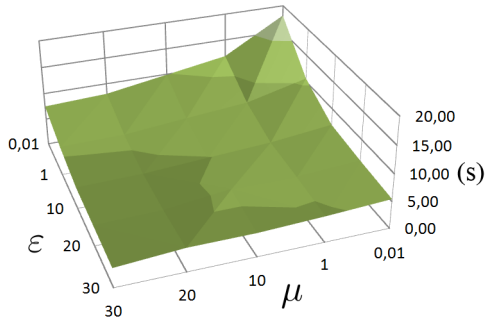


Figure 3: Dependence of the time (s) on precision parameters  $\varepsilon$  and  $\mu$

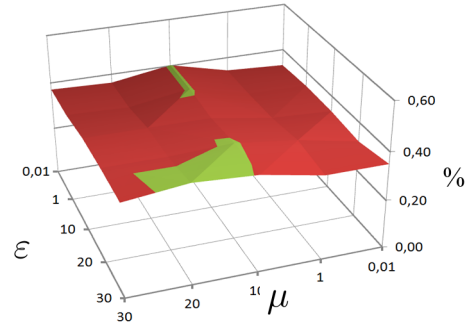


Figure 4: Dependence of the error (%) on precision parameters  $\varepsilon$  and  $\mu$

CIH is able to solve these instances close to the optimum. The evaluation on the test instances with “stretched” ellipses showed that CIH produces better solutions than TSP→TPP, when the ratio between radii of ellipses along the axis becomes larger, i.e., ellipses become more “stretched”. Even though there exist multiple specialized heuristics for CETSP, which might outperform CIH, our method is still able to solve these instances well. CIH highest efficiency was achieved in the tests with a maximum of around 100 areas.

Due to the promising results, we are motivated to continue the work on CIH in several directions. Although, CIH showed good results, it is still considered to be a construction heuristic. The problem of adaptation of tour-improvement heuristics in TSPN is of a great interest. CIH is

a general approach and may be applied to multiple different industrial use cases. One way is to make adaptations of CIH for robotics, e.g., by substituting Euclidean space with robot axis space. It will allow to obtain more optimal solutions, as information about the robot structure will be involved in the solution process. Another direction could be the adaptation to the scenario with multiple salesmen, i.e., multiple robots working on the same product at a time.

## Acknowledgments

We would like to kindly thank Vera Mersheeva (University of Klagenfurt, Austria) and Kevin Nelson (Blue Technik LLC, USA) for their detailed comments and discussions.



## References

- Alatartsev, S.; Augustine, M.; and Ortmeier, F. 2012. <http://euromover.cs.uni-magdeburg.de/cse/robotics/tspn/>.
- Applegate, D. L.; Bixby, R. E.; Chvatal, V.; and Cook, W. J. 2007. *The traveling salesman problem: a computational study*. Princeton University Press.
- Arkin, E. M., and Hassin, R. 1995. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics* 55:197–218.
- Arora, S. 2003. Approximation schemes for np-hard geometric optimization problems: A survey. *Mathematical Programming* 97.
- Baizid, K.; Chellali, R.; Yousnadj, A.; Meddahi, A.; and Bentaleb, T. 2010. Genetic algorithms based method for time optimization in robotized site. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1359–1364.
- Dror, M.; Efrat, A.; Lubiw, A.; and Mitchell, J. S. B. 2003. Touring a sequence of polygons. In *35th annual ACM symposium on Theory of Computing*, 473–482. ACM Press.
- Elbassioni, K. M.; Fishkin, A. V.; and Sitters, R. 2009. Approximation algorithms for the euclidean traveling salesman problem with discrete and continuous neighborhoods. *International Journal of Computational Geometry and Applications* 173–193.
- Gentilini, I.; Margot, F.; and Shimada, K. 2011. The traveling salesman problem with neighbourhoods: MINLP solution. *Optimization Methods and Software* 0:1–15.
- Hassin, R., and Keinan, A. 2008. Greedy heuristics with regret, with application to the cheapest insertion algorithm for the TSP. *Operations Research Letters* 36(2):243–246.
- Helsgaun, K. 2000. An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research* 126:106–130.
- Johnson, D. S., and McGeoch, L. A. 1997. *Local search in combinatorial optimization*. John Wiley and Sons, London.
- Mennell, W. 2009. *Heuristics for solving three routing problems: close-enough traveling salesman problem, close-enough vehicle routing problem, sequence-dependent team orienteering problem*. Ph.D. Dissertation, University of Maryland.
- Mitchell, J. S. 2010. A constant-factor approximation algorithm for TSP with pairwise-disjoint connected neighborhoods in the plane. In *Proceedings of the 2010 annual symposium on Computational geometry*, SoCG '10, 183–191. New York, NY, USA: ACM.
- Oberlin, P.; Rathinam, S.; and Darbha, S. 2010. Today's traveling salesman problem. *IEEE Robotics & Automation Magazine* 17(4):70–77.
- Pan, Z.; Polden, J.; Larkin, N.; Duin, S. V.; and Norrish, J. 2010. Recent progress on programming methods for industrial robots. In *Proceedings for the joint conference of ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*.
- Pan, Z.; Polden, J.; Larkin, N.; Duin, S.; and Norrish, J. 2012. Automated offline programming for robotic welding system with high degree of freedoms. In Wu, Y., ed., *Advances in Computer, Communication, Control and Automation*, volume 121 of *Lecture Notes in Electrical Engineering*. Springer Berlin Heidelberg. 685–692.
- Pan, X.; Li, F.; and Klette, R. 2010. Approximate shortest path algorithms for sequences of pairwise disjoint simple polygons. In *CCCG*, 175–178.
- Press, W. H.; Flannery, B. P.; Teukolsky, S. A.; and Vetterling, W. T. 2007. *Numerical recipes – the art of scientific computing*. Cambridge: Cambridge University Press, 3rd edition.
- Shi, X. H.; Liang, Y. C.; Lee, H. P.; Lu, C.; and Wang, Q. X. 2007. Particle swarm optimization-based algorithms for TSP and generalized TSP. *Information Processing Letters* 103(5):169–176.