On Reversal and Transposition Medians

Martin Bader

Ulmer Informatik-Berichte

Nr. 2009-04
März 2009
On Reversal and Transposition Medians

Martin Bader
University of Ulm, Institute of Theoretical Computer Science, 89069 Ulm, Germany
Email: martin.bader@uni-ulm.de

Abstract. During the last years, the genomes of more and more species have been sequenced, providing data for phylogenetic reconstruction based on genome rearrangement measures. A main task in all phylogenetic reconstruction algorithms is to solve the median of three problem. Although this problem is NP-hard even for the simplest distance measures, there are exact algorithms for the breakpoint median and the reversal median that are fast enough for practical use. In this paper, we extend this approach to the transposition median as well as to the weighted reversal and transposition median. Although there is no exact polynomial algorithm known even for the pairwise distances, we will show that it is in most cases possible to solve these problems exactly within reasonable time by using a branch and bound algorithm.

1 Introduction

Due to the increasing amount of sequenced genomes, the problem of reconstructing phylogenetic trees based on this data is of great interest in computational biology. In the context of genome rearrangements, a genome is usually represented as a permutation of $(1,\ldots,n)$, where each element represents a gene, i.e. the permutation represents the shuffled ordering of the genes on the genome. Additionally, the strandedness of the genes can be taken into account by giving each element an orientation. In the multiple genome rearrangement problem, one searches for a phylogenetic tree describing the most “plausible” rearrangement scenario for multiple genomes. Formally, given $k$ genomes and a distance measure $d$, find a tree $T$ with the $k$ genomes as leaf nodes and assign ancestral genomes to internal nodes of $T$ such that the tree is optimal w.r.t. $d$, i.e. the sum of rearrangement distances over all edges of the tree is minimal. If we set $k = 3$, i.e. we search for an ancestor such that the sum of the distances from this ancestor to three given genomes is minimized, we speak of the median problem. All of the actual state-of-the-art algorithms for solving the multiple genome rearrangement problem rely on algorithms for solving the median problem. Unfortunately, this problem is NP-hard even for the simplest rearrangement measures, namely the breakpoint distance and the reversal distance [12,8]. Currently, the most interesting distance measures are:

- The reversal distance between two genomes is the minimum number of reversals required to transform one genome into the other. It can be computed in
linear time [1]. The reversal median problem has been proven to be NP-hard [8]. The currently best software tools to solve the multiple genome rearrangement problem based on this distance measure are GRAPPA [11], MGR [7], amGRP [6], and phylo [2]. All of them rely on Caprara’s median solver [8] (GRAPPA can alternatively use Siepel’s median solver [13]).

– The transposition distance between two genomes is the minimum number of transpositions required to transform one genome into the other. So far, it is not clear whether it is in $P$ or not, and the currently best approximation algorithm has an approximation ratio of 1.375 [10]. An exact branch and bound algorithm is described in [9]. To the best of our knowledge, the only program that solves the multiple genome rearrangement problem based on this distance measure is GRAPPA-TP [14], which uses an extension of Siepel’s median solver [13] and solves pairwise distances by a fast heuristic.

– The weighted reversal and transposition distance between two genomes is the minimum weight of a sequence consisting of reversals and transpositions that transforms one genome into the other, where reversals and transpositions are weighted differently. Again, it is not clear whether it is in $P$ or not, but there exist a 1.5-approximation algorithm that covers each weight ratio from 1:1 to 1:2 (reversals:transpositions) [3]. As far as we know, the only program that solves the multiple genome rearrangement problem based on this distance measure is phylo [2], which uses a preliminary version of the median solver that we present in this paper.

In this paper, we will show how one can solve the transposition median as well as the weighted reversal and transposition median by extending Caprara’s median solver. In order to do this, we need to calculate pairwise distances between genomes, which can be either done approximately using the algorithm devised in [3], or exactly using a new branch and bound algorithm presented in this paper. Experimental results show that the approximation rate of the first method is very good in practice, and that even the exact algorithm runs in feasible time for practical use. In Section 2 we give basic definitions. The algorithm to calculate exact pairwise distances is described in Section 3, the algorithm to solve the median problem is described in Section 4. The experimental results and a comparison with GRAPPA-TP, which was kindly provided by Jijun Tang, can be found in Section 5. Section 6 summarizes the method and the results.

### 2 Preliminaries

A signed permutation $\pi = (\pi_1 \ldots \pi_n)$ is a permutation of $(1 \ldots n)$, where each element $\pi$ has an orientation (indicated by $\pi_i^+$ or $\pi_i^-$. We will use the term “permutation” as short hand for signed permutation. The permutation $id = (1 \ldots n)$ is called the identity permutation of size $n$. A segment of a permutation $\pi$ is a consecutive sequence of elements in $\pi$. A reversal is an operation that inverts the order of the elements of a segment in a permutation. Additionally, the orientation of every element in the segment is flipped. A transposition is an
operation that cuts a segment out of a permutation, and reinserts it at another position in the permutation. If we additionally apply a reversal on this segment, we speak of an inverted transposition. The weight of an operation \( op \) is denoted by \( w(op) \), and the weight of a sequence of operations is the sum of the weights of the operations in the sequence. In the following, reversals have weight \( w_r \), whereas transpositions and inverted transpositions have weight \( w_t \), and we assume that \( w_r \leq w_t \leq 2w_r \) (otherwise optimal sequences would have an unrealistic strong bias either towards reversals or transpositions). The problem of sorting by weighted reversals and transpositions is defined as follows. Given two permutations \( \pi^1, \pi^2 \), find a sequence of reversals and transpositions of minimum weight that transforms \( \pi^1 \) into \( \pi^2 \). This minimum weight is called the weighted reversal and transposition distance (wRTD) \( d_w(\pi^1, \pi^2) \). If we restrict the set of operations to transpositions only, the problem is called sorting by transpositions, and the corresponding distance is called the transposition distance (TD) \( d_t(\pi^1, \pi^2) \).

The edge set \( M \) can be obtained as follows. First, we replace in each permutation \( \pi \) by Caprara \[8\] and is a generalization of the breakpoint graph defined in \[4\]. For \( \rho, \pi \) and TMP, we will use the transposition median problem \((\rho, \pi^1, \pi^2) \) and the transposition distance \((TD) d_t(\pi^1, \pi^2) \). This minimum weight is called the transposition median problem (TMP) and is defined analogously. For solving wRTMP and TMP, we will use the multiple breakpoint graph, which has been introduced by Caprara \[8\] and is a generalization of the breakpoint graph defined in \[4\]. For permutations \( \pi^1, \ldots, \pi^n \), the MB graph \( G = (V, E) \) is a multigraph with node set \( V = \{-1, +1, -2, +2, \ldots, -n, +n\} \) (where \( n \) is the size of the permutations). The edge set can be obtained as follows. First, we replace in each permutation \( \pi^k \) (\( 1 \leq k \leq q \)) all elements with positive orientation \( \pi^k \) by \( -x + x \) and all elements with negative orientation \( \pi^k \) by \( +x - x \). Then, each permutation \( \pi^k \) induces the edge set \( M^k = \{(i, j)| i \neq j \) and \( \pi^k \) contains the adjacent values \( i \) and \( j \}\}. i.e., the edge set \( M^k \) corresponds to the adjacencies in \( \pi^k \). The edge set \( E \) of the MB graph \( G \) is the union of these edge sets, i.e. \( E = \bigcup_{k=1}^q M^k \). As each node is connected to exactly one edge in each edge set \( M^k \), the graphs \( G_{i,j} = (V, M^i \cup M^j) \) (with \( 1 \leq i, j \leq q \)) decompose into cycles with alternating edges from the edge sets \( M^i \) and \( M^j \). A cycle is called an odd cycle if its number of edges divided by \( 2 \) is an odd number, otherwise it is called an even cycle. Let \( c_{odd}(\pi^i, \pi^j) \) denote the number of odd cycles in \( G_{i,j} \), and let \( c_{even}(\pi^i, \pi^j) \) denote the number of even cycles in \( G_{i,j} \). The score \( \sigma \) between two permutations \( \pi^1 \) and \( \pi^2 \) is defined by

\[
\sigma(\pi^1, \pi^2) = c_{odd}(\pi^1, \pi^2) + (2 - \frac{2w_r}{w_t})c_{even}(\pi^1, \pi^2).
\]

The following theorems show how we can use this score to obtain lower and upper bounds for the wRTD.

**Theorem 1.** \([3, 5]\) A lower bound \( lb_w(\pi^1, \pi^2) \) for the weighted reversal and transposition distance \( d_w(\pi^1, \pi^2) \) can be defined as follows.

\[
d_w(\pi^1, \pi^2) \geq lb_w(\pi^1, \pi^2), \quad \text{where } lb_w(\pi^1, \pi^2) := (n - \sigma(\pi^1, \pi^2)) \frac{w_t}{2}.
\]

A lower bound \( lb_t(\pi^1, \pi^2) \) for the transposition distance \( d_t(\pi^1, \pi^2) \) can be defined as follows.

\[
d_t(\pi^1, \pi^2) \geq lb_t(\pi^1, \pi^2), \quad \text{where } lb_t(\pi^1, \pi^2) := (n - c_{odd}(\pi^1, \pi^2)) \frac{w_t}{2}.
\]
Note that if we set $w_t = 2w_r$, the lower bounds for both distances are equal. This will later simplify the description of the algorithms, as we will only use the lower bound for the wRTD.

**Theorem 2.** [3, 10] An upper bound $ub_w(\pi^i, \pi^j)$ for the weighted reversal and transposition distance $d_w(\pi^i, \pi^j)$ can be defined as follows.

$$d_w(\pi^i, \pi^j) \leq ub_w(\pi^i, \pi^j), \text{ where } lb_w(\pi^i, \pi^j) := 1.5 lb_w(\pi^i, \pi^j)$$

An upper bound $ub_t(\pi^i, \pi^j)$ for the transposition distance $d_t(\pi^i, \pi^j)$ can be defined as follows.

$$d_t(\pi^i, \pi^j) \leq ub_t(\pi^i, \pi^j), \text{ where } ub_t(\pi^i, \pi^j) := 1.375 lb_t(\pi^i, \pi^j)$$

### 3 Calculating pairwise distances

As exact polynomial algorithms are neither known for the TD nor for the wRTD, we introduce a branch and bound algorithm for the pairwise distances. The main idea of the algorithm is straightforward. W.l.o.g., the task is to find an optimal sorting sequence between a permutation $\pi$ and the identity permutation $id$ of the same size. We create a set $S$ that contains triples $(\tilde{\pi}, d'(\pi, \tilde{\pi}), lb(\tilde{\pi}, id))$, where $\tilde{\pi}$ is a permutation, $d'(\pi, \tilde{\pi})$ is the sum of the weights of all operations that have been performed on the path from $\pi$ to $\tilde{\pi}$, and $lb(\tilde{\pi}, id)$ is the lower bound for the remaining distance towards $id$ according to Theorem 1. Initially, we set $S = \{(\pi, 0, lb(\pi, id))\}$. In each step, we select the triple $(\tilde{\pi}, d'(\pi, \tilde{\pi}), lb(\tilde{\pi}, id))$ from $S$ where $d'(\pi, \tilde{\pi}) + lb(\tilde{\pi}, id)$ is minimized, and remove it from $S$. If $lb(\tilde{\pi}, id) = 0$, then $\tilde{\pi} = id$ and $d'(\pi, \tilde{\pi}) = d(\pi, id)$, i.e. we have found an optimal solution and the algorithm aborts. The sequence of operations can be reconstructed by a traceback. Otherwise, for each operation $op$, we add the triple $(op \tilde{\pi}, d'(\pi, \tilde{\pi}) + w(op), lb(op \tilde{\pi}, id))$ to $S$, i.e. we add all possible predecessors of $\tilde{\pi}$ to $S$. We now continue by again selecting the best triple.

So far, the algorithm is just an ordinary branch and bound algorithm, and does not perform very well in practice. Thus, we improve the algorithm by a duplicate elimination. Because there are usually different optimal sequences to reach an intermediate permutation, this permutation would be stored several times, and in the worst case the number of duplicates of a permutation can be exponential in the distance to the origin permutation. Therefore, we first check if we already have generated a permutation before creating a new triple containing this permutation. Searching for a possible duplicate can be done quite efficiently by hashing techniques. We can further reduce the number of elements in $S$ by working on the minimal permutations, which have been defined in [9] as follows. Given a permutation $\tilde{\pi}$, we obtain the minimal permutation $gl(\tilde{\pi})$ by ‘gluing’ all the adjacencies together, i.e. we replace each segment of elements that is identical in $\tilde{\pi}$ and $id$ by a single element. As an example, the permutations $\tilde{\pi} = (1 \ 2 \ 4 \ 3)$ and $\hat{\pi} = (1 \ 3 \ 4 \ 2)$ have both the same minimal permutation $\tilde{\pi} = (1 \ 2 \ 3)$. 
The following lemma ensures that it is sufficient to search for an optimal sorting sequence between $gl(\tilde{\pi})$ and $id'$ to obtain an optimal sorting sequence between $\tilde{\pi}$ and $id$, where $id'$ is the identity permutation of same size as $gl(\tilde{\pi})$.

Lemma 1. [9] Let $\pi$ be a permutation and $gl(\pi)$ be its minimal permutation. Let $id$ be the identity permutation of same size as $\pi$, and let $id'$ be the identity permutation of same size as $gl(\pi)$. Then, an optimal sorting sequence between $gl(\pi)$ and $id'$ can easily be transformed into an optimal sorting sequence between $\pi$ and $id$. Both sorting sequences have the same weight, i.e. $d(\pi, id) = d(\tilde{\pi}, id')$.

Note that the original lemma in [9] only considered the TD. However, the proof for the wRTD works analogously, thus this lemma holds for the TD as well as for the wRTD. While Christie used this proof only to show that one never has to split adjacencies, we will also use it for duplicate elimination. In the example above, $\tilde{\pi}$ would be considered to be a duplicate of $\hat{\pi}$. In fact, we even do not store the original permutations but only their minimal permutations, resulting in a further space improvement.

4 The median solver

Our median solver is an extension of Caprara's reversal median solver [8]. While Caprara's algorithm solves instances of the Cycle Median Problem (CMP) and reestimates the distances using the reversal distance, we extend the CMP to the weighted Cycle Median Problem and reestimate the distances using the TD or the wRTD.

For a given wRTMP instance with permutations $\pi^1, \ldots, \pi^q$, and an arbitrary permutation $\rho$, define $\gamma(\rho) := \sum_{k=1}^{q} \sigma(\rho, \pi^k)$. The weighted Cycle Median Problem (wCMP) is defined as follows. Given a set of $q$ permutations $\pi^1, \ldots, \pi^q$, find a permutation $\tau$ such that $qn - \gamma(\tau)$ is minimized. In the following, let $\rho^*$ be the solution of a given wRTMP and let $\delta^* := \delta(\rho^*) = \sum_{i=1}^{q} d_w(\pi^i, \rho^*)$ be its solution value. Let $\tau^*$ be the solution of the associated wCPM and let $qn - \gamma^* := qn - \gamma(\tau^*)$ be its solution value. The following lemma shows the relation between a wRTMP instance and the associated wCMP instance.

Lemma 2. Given a wRTMP instance with solution value $\delta^*$ and the associated wCMP instance with solution value $qn - \gamma^*$,

$$\frac{w_r}{2} (qn - \gamma^*) \leq \delta^* \leq \frac{3w_t}{4} (qn - \gamma^*)$$

Proof. Using the bounds given in Theorems 1 and 2, we get

$$\frac{w_r}{2} (qn - \gamma(\tau^*)) \leq \frac{w_r}{2} (qn - \gamma(\rho^*)) = \sum_{k=1}^{q} lb(\pi^k, \rho^*) \leq \delta^* \leq \sum_{k=1}^{q} d(\pi^k, \tau^*) \leq 1.5 \sum_{k=1}^{q} lb(\pi^k, \tau^*) = \frac{3w_t}{4} (qn - \gamma^*)$$

Note that this proof also holds for the TD if we set $w_r = 1$, $w_t = 2$, and restrict the search space of the wCMP to permutations where all elements have positive orientation. In this case, $\gamma(\tau) = \sum_{k=1}^{q} c_{odd}(\pi^k, \tau)$, i.e. an optimal solution of
the wCPM maximizes the number of odd cycles. In most cases, $\delta^*$ is very close to the lower bound. This motivates the idea to solve a wRTMP instance by solving the associated wCMP instance and then check whether the solution of the wCMP instance is also a solution of the wRTMP instance. We will now address the problem of solving a wCMP instance. As we will use a branch and bound algorithm that successively extends a partial solution until we have a complete solution, we must extend the MB graph such that we can use it to obtain strong lower bounds for partial solutions. A graph $(V, E)$ is weighted if each edge $e \in E$ has an integer weight $w(e)$. Given a weighted graph $G = (V, E)$ with node set $V = \{-1, 1, -2, 2, \ldots, -n, n\}$, a weighted matching $M$ is a set of edges in $G$ such that each node in $V$ is incident to at most one edge in $M$ and each edge in $M$ has an odd weight (restricting the weights to be odd will simplify later proofs). A weighted matching $M$ is called *permutation matching* if $H \cup M$ defines a Hamiltonian cycle on $G$, i.e. a cycle that visits each node in $V$ exactly once.

**Lemma 3.** [4] There is a one-to-one correspondence between signed permutations and permutation matchings where each edge has weight 1.

In other words, we can transform each permutation matching into a permutation by ignoring the weights. On the other hand, we can reduce the search space to permutation matchings. Interpreting the MB graph as the special case of a weighted graph (where each weight is set to 1) leads to the following formulation of the wCMP. Given a node set $V$ with $|V| = 2n$ and $q$ permutation matchings $M^1, \ldots, M^q$, find a permutation matching $M^\tau$ with edge weights 1 that minimizes $\sum_{k=1}^{q} (n - \sigma(M^\tau, M^k))$. Note that we do not restrict the weights of the edges of the given permutation matches. While all edges in the initial problem have weight 1, the branch and bound algorithm will create partial solutions where also other edge weights are possible.

**Lemma 4.** The weighted cycle distance $n - \sigma(S, T)$ on permutation matchings is a metric.

**Proof.** 1. Positive definiteness: $n - \sigma(S, S) = 0$, because the graph decomposes into $n$ odd cycles. For permutation matchings $S, T$ with $S \neq T$, there must be at least one cycle with at least four edges, thus the overall number of
Lemma 5. Given a wCMP instance associated with weighted matchings \( M^1, \ldots, M^q \) and solution \( M^* \), we have

\[
\frac{q}{2} \sum_{k=1}^{q} (n - \sigma(M^*, M^k)) \geq \frac{qn}{2} - \sum_{k=1}^{q-1} \sum_{l=k+1}^{q} \frac{\sigma(M^k, M^l)}{q-1}
\]

Proof. Using the triangle inequality given in Lemma 4, we get

\[
\frac{qn}{2} - \sum_{k=1}^{q-1} \sum_{l=k+1}^{q} \frac{\sigma(M^k, M^l)}{q-1} = \frac{1}{q-1} \sum_{k=1}^{q-1} \sum_{l=k+1}^{q} (n - \sigma(M^k, M^l)) \leq \sum_{k=1}^{q} (n - \sigma(M^*, M^k))
\]

In order to describe partial solutions, we must introduce the contraction of an edge. Given a weighted graph \( G = (V, E) \) with \( |V| = 2n \) and \( E = \bigcup_{k=1}^{q} M^k \), where each \( M^k \) is a permutation matching, the contraction of an edge \( e = (v_i, v_j) \) is an operation that modifies \( G \) as follows. The nodes \( v_i, v_j \) are removed from \( V \). Each permutation matching \( M^k \) is transformed into \( M^k/e \) by the following...
rules. If \( e \in M^k \), remove \( e \) from \( M^k \), i.e. \( M^k/e = M^k \setminus \{e\} \). Otherwise, let \((a, v_i)\) and \((b, v_j)\) be the two edges incident to \( v_i \) and \( v_j \) in \( M^k \). Remove these edges and add a new edge \((a, b)\), i.e. \( M^k/e = M^k \setminus \{(a, v_i), (b, v_j)\} \cup \{(a, b)\} \). The weight of the new edge \((a, b)\) will be set to \( w(a, b) := w(a, v_i) + w(b, v_j) + 1 \). Note that this is also an odd number, as \( w(a, v_i) \) and \( w(b, v_j) \) are odd. Analogously, the base matching \( H \) will be replaced by \( H/e \).

**Lemma 6.** [8] Given two perfect matchings \( M \) and \( L \) of \( V \) and an edge \((v_i, v_j)\) \( \in M \), \( M \cup L \) defines a Hamiltonian cycle of \( V \) if and only if \((M/e) \cup (L/e)\) defines a Hamiltonian cycle of \( V \setminus \{v_i, v_j\} \).

**Lemma 7.** Let \( M^1, \ldots, M^q \) be a wCMP instance, let \( M^\tau \) be a permutation matching with edge weights 1, and let \( e \in M^\tau \) be an edge. Then,

\[
\sum_{k=1}^{q} (n - \sigma(M^\tau, M^k)) = q - \sum_{k=1}^{q} \sigma(M^k, \{e\}) + \sum_{k=1}^{q} (n - 1 - \sigma(M^\tau/e, M^k/e))
\]

**Proof.** A cycle in \( M^\tau \cup M^k \) is either absorbed by the contraction step, or it corresponds to a cycle in \( M^\tau/e \cup M^k/e \) of the same length. In the first case, the absorbed cycle is equivalent to the cycle in \( M^k \cup \{e\} \), and the sum of the scores of the absorbed cycles is \( \sum_{k=1}^{q} \sigma(M^k, \{e\}) \). As there are no new cycles in \( M^\tau/e \cup M^k/e \), we get \( \sum_{i=1}^{q} \sigma(M^\tau, M^k) = -\sum_{k=1}^{q} \sigma(M^k, \{e\}) + \sum_{k=1}^{q} (n - \sigma(M^\tau/e, M^k/e)) = q - \sum_{k=1}^{q} \sigma(M^k, \{e\}) + \sum_{k=1}^{q} (n - 1 - \sigma(M^\tau/e, M^k/e)) \).

By combining Lemmas 6 and 7, we get the following

**Corollary 1.** Given a wCMP instance \( M^1, \ldots, M^q \) with solution \( M^\tau \) and an edge \( e \in M^\tau \), then \( M^\tau \setminus \{e\} \) is a solution of the wCMP instance \( M^1/e, \ldots, M^q/e \).

We are now ready to describe our branch and bound algorithm for wCMP. A partial solution consists of a matching that is not necessarily perfect, and the lower bound of a partial solution \( M \) can be calculated by contracting all edges in \( M \) and calculating the lower bound of the contracted graph, using the formulas described in Lemmas 5 and 7. In each step, we select the partial solution \( M \) with the currently least lower bound and expand it as follows. Let \( V' \) be the nodes of \( V \) such that for all \( v_i \in V', v_j \in V : (v_i, v_j) \not\in M \), and let \( v_a \) be a fixed node in \( V' \). Then, we create new partial solutions \( M' \) by setting \( M' = M \cup (v_a, v_b) \) for all \( v_b \in V', v_b \neq v_a \). Partial solutions \( M' \) that cannot be expanded to a permutation matching (i.e. \( M' \cup H \) contains a cycle that is not a Hamiltonian cycle) can be directly discarded, for all other partial solutions we calculate the lower bounds. The algorithm has found an optimal solution for the wCMP when the partial solution with the least lower bound is a perfect matching.

The algorithm can easily be extended such that it can solve the wRTMP or the TMP by adding the following step. Whenever the best partial solution \( M^\tau \) is a perfect matching, create the corresponding permutation \( \pi^\tau \) and test if \( \sum_{k=1}^{q} d_w(\pi^k, \pi^\tau) \) is equal to the lower bound (of course one has to take \( d_t \) instead of \( d_w \) if one wants to solve the TMP). In this case, we have found an optimal solution. Otherwise, we increase the lower bound for \( M^\tau \) and reinsert it.
into the set of partial solutions. We can get a further speed-up of the pairwise
distance algorithm by providing an upper bound (remember that we only want
to test if the sum of the pairwise distances is equal to the lower bound, thus
we can abort the pairwise distance algorithms if the currently best results are
above this bound). If one wants to solve the TMP, partial solutions are further
restricted to matchings where all edges are of the form \((+i, -j)\), because other
edges correspond to a change in orientation in the permutation and therefore
these permutations cannot be sorted by transpositions only.

5 Experimental results

We tested our algorithm on artificial datasets with 37 and 100 markers, reflect-
ing the size of mitochondrial and chloroplast genomes. We created datasets
by, starting from the identity permutation, creating three different sequences of
operations to get the input genomes. The weight of the edges is uniformly dis-
tributed in \([0.5r, 1.5r]\), where \(r\) is the expected weight of an edge, varying from 2
to 15. For the data sets to test the transposition median solver, \(w_t\) was set to 1.
For the data sets to test the weighted reversal and transposition median solver,
\(w_r\) was set to 1, and we created different data sets for \(w_t = 1, w_t = 1.5,\) and
\(w_t = 2\). When creating the data sets, the probability of performing a transpo-
sition reflects the weight of \(w_t\), i.e. the expected ratio of reversals to transpositions
is \(w_t : w_r\). For each combination of these parameters, we created 10 data sets.
All tests were performed on a standard 3.16 GHz PC, the running time for each
test case was limited to one hour, and RAM was limited to 4GB.

The experiments showed that the size of the datasets has only little influence on
the results. Up to an expected edge length of \(r = 8\), we could solve all test cases
exactly, most of them even in less than one second. For higher distances, the run-
ing times increased. However, we could still solve 9 instances of the TMP with
\(r = 15\) and \(n = 37\) with an average running time of 6:21 min, and 6 instances of
the TMP with \(r = 15\) and \(n = 100\) with an average running time of 16:27 min.

For the instances of the wRTMP, the running times depend on the used weight
ratio. While setting \(w_r : w_t\) to 1 : 2 allowed us to solve all test cases within a
few seconds, the running times increased for the other weight ratio. Moreover,
we had to prune the heap due to the memory limit in some cases, which means
that there is a slight chance that we missed the optimal solution. Nevertheless,
we were still able to solve all test cases, except for a few test cases with \(n = 100,\)
\(w_r : w_t = 1 : 1,\) and \(r > 13\). Using the approximation algorithm instead of the
exact algorithm for the pairwise distances resulted in better running times at
almost the same accuracy. In most cases, we found a solution of same weight
as with the exact algorithm, and the gap between the weights of the solutions
never exceeded 1.

A comparison with GRAPPA-TP on the instances of the TMP shows that GRAPPA-TP
is slightly less accurate than our median solver, but its main drawback is the
speed. For \(r = 7\), its average running time was 3:36 min (\(n = 37\)) respectively
6:41 min (\(n = 100\)), while our algorithm solved these test cases within less than
one second. Increasing the edge lengths further decreased the number of solved test cases. For \( n = 37 \), none of the test cases with \( r \geq 14 \) could be solved within one hour. For \( n = 100 \), none of the test cases with \( r \geq 11 \) could be solved within one hour.

A more detailed view of the test results can be found in Appendix A. In the tables, we list the number of solved test cases and the average running time of the approximation algorithm and the exact algorithm for each combination of parameters, as well as the average gap and the maximum gap between the solution of the approximation algorithm and the exact algorithm. Of course, the gaps can only be computed for test cases which have been solved by both algorithms, and the average running times only consider test cases that could be solved within the time limit. The column “proven exact” indicates the number of test cases where we can assure that the exact algorithm did not miss an optimal solution due to heap pruning. Note that a solution there we cannot assure this might still be exact. In fact, as we only prune the currently worst solutions, the probability of missing an optimal solution is rather low.

6 Conclusion

We presented an extension of Caprara’s median solver that can solve instances of the TMP and the wRTMP. The method has been tested on artificial datasets, showing that is possible to solve the wRTMP and the TMP exactly in many cases. A comparison with GRAPPA–TP on TMPs shows that our algorithm brings a speed improvement of several orders of magnitude.

References


A Detailed experimental results

<table>
<thead>
<tr>
<th>r</th>
<th>solved app</th>
<th>avg gap</th>
<th>max gap</th>
<th>avg time</th>
<th>solved exact</th>
<th>proven exact</th>
<th>avg time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:06</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:06</td>
<td>10</td>
<td>9</td>
<td>1:16</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:02</td>
<td>10</td>
<td>10</td>
<td>0:04</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:15</td>
<td>10</td>
<td>9</td>
<td>2:22</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0.22</td>
<td>1</td>
<td>1:17</td>
<td>9</td>
<td>5</td>
<td>6:00</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>2:12</td>
<td>8</td>
<td>3</td>
<td>1:54</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>0.11</td>
<td>1</td>
<td>2:18</td>
<td>9</td>
<td>2</td>
<td>7:22</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>0.11</td>
<td>1</td>
<td>1:59</td>
<td>8</td>
<td>3</td>
<td>4:17</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>0.13</td>
<td>1</td>
<td>4:37</td>
<td>8</td>
<td>0</td>
<td>15:32</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>0.11</td>
<td>1</td>
<td>2:57</td>
<td>9</td>
<td>3</td>
<td>6:21</td>
</tr>
</tbody>
</table>

*Table 1. n = 37, transposition distance*
<table>
<thead>
<tr>
<th>r</th>
<th>solved app</th>
<th>avg gap</th>
<th>max gap</th>
<th>avg time</th>
<th>solved exact</th>
<th>proven exact</th>
<th>avg time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>9</td>
<td>9</td>
<td>0:00</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:04</td>
<td>10</td>
<td>10</td>
<td>0:04</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0:25</td>
<td>9</td>
<td>8</td>
<td>0:30</td>
</tr>
<tr>
<td>13</td>
<td>9</td>
<td>0.17</td>
<td>1</td>
<td>7:50</td>
<td>6</td>
<td>4</td>
<td>2:35</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>0.14</td>
<td>1</td>
<td>2:54</td>
<td>7</td>
<td>5</td>
<td>0:19</td>
</tr>
<tr>
<td>15</td>
<td>7</td>
<td>0.17</td>
<td>1</td>
<td>7:24</td>
<td>6</td>
<td>3</td>
<td>16:27</td>
</tr>
</tbody>
</table>

Table 2. $n = 100$, transposition distance

<table>
<thead>
<tr>
<th>r</th>
<th>solved app</th>
<th>avg gap</th>
<th>max gap</th>
<th>avg time</th>
<th>solved exact</th>
<th>proven exact</th>
<th>avg time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>0.1</td>
<td>1</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:01</td>
<td>10</td>
<td>10</td>
<td>0:01</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1:20</td>
<td>10</td>
<td>8</td>
<td>1:24</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>0.1</td>
<td>1</td>
<td>0:06</td>
<td>10</td>
<td>10</td>
<td>0:03</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0.2</td>
<td>1</td>
<td>2:48</td>
<td>10</td>
<td>6</td>
<td>3:01</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>4:19</td>
<td>10</td>
<td>2</td>
<td>6:17</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>0.2</td>
<td>1</td>
<td>5:23</td>
<td>10</td>
<td>2</td>
<td>6:04</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>0.3</td>
<td>1</td>
<td>9:38</td>
<td>10</td>
<td>1</td>
<td>11:05</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>0.2</td>
<td>1</td>
<td>9:28</td>
<td>10</td>
<td>0</td>
<td>13:18</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>0.3</td>
<td>1</td>
<td>10:49</td>
<td>10</td>
<td>1</td>
<td>16:21</td>
</tr>
</tbody>
</table>

Table 3. $n = 37$, $w_r = 1$, $w_t = 1$
<table>
<thead>
<tr>
<th>r</th>
<th>solved app</th>
<th>avg gap</th>
<th>max gap</th>
<th>avg time</th>
<th>solved exact</th>
<th>proven exact</th>
<th>avg time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:05</td>
<td>10</td>
<td>10</td>
<td>0:05</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>1:01</td>
<td>10</td>
<td>7</td>
<td>1:22</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:38</td>
<td>10</td>
<td>8</td>
<td>0:52</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>4:21</td>
<td>10</td>
<td>7</td>
<td>4:49</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:53</td>
<td>10</td>
<td>7</td>
<td>0:58</td>
</tr>
<tr>
<td>13</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>2:09</td>
<td>8</td>
<td>6</td>
<td>2:39</td>
</tr>
<tr>
<td>14</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>16:46</td>
<td>7</td>
<td>4</td>
<td>19:33</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>13:40</td>
<td>3</td>
<td>2</td>
<td>1:30</td>
</tr>
</tbody>
</table>

Table 4. \( n = 100, w_r = 1, w_t = 1 \)

<table>
<thead>
<tr>
<th>r</th>
<th>solved app</th>
<th>avg gap</th>
<th>max gap</th>
<th>avg time</th>
<th>solved exact</th>
<th>proven exact</th>
<th>avg time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>0.05</td>
<td>0.5</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>0.05</td>
<td>0.5</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>0.05</td>
<td>0.5</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>0.05</td>
<td>0.5</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0.15</td>
<td>0.5</td>
<td>0:06</td>
<td>10</td>
<td>9</td>
<td>0:01</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:10</td>
<td>10</td>
<td>9</td>
<td>0:11</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>0.1</td>
<td>0.5</td>
<td>0:05</td>
<td>10</td>
<td>8</td>
<td>0:05</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:56</td>
<td>10</td>
<td>8</td>
<td>1:16</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>0.1</td>
<td>0.5</td>
<td>1:09</td>
<td>10</td>
<td>7</td>
<td>1:33</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>0.05</td>
<td>0.5</td>
<td>0:09</td>
<td>10</td>
<td>8</td>
<td>0:09</td>
</tr>
</tbody>
</table>

Table 5. \( n = 37, w_r = 1, w_t = 1.5 \)
<table>
<thead>
<tr>
<th>( r )</th>
<th>solved app</th>
<th>avg gap</th>
<th>max gap</th>
<th>avg time</th>
<th>solved exact</th>
<th>proven exact</th>
<th>avg time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
</tbody>
</table>

Table 6. \( n = 100, w_r = 1, w_t = 1.5 \)

<table>
<thead>
<tr>
<th>( r )</th>
<th>solved app</th>
<th>avg gap</th>
<th>max gap</th>
<th>avg time</th>
<th>solved exact</th>
<th>proven exact</th>
<th>avg time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
</tbody>
</table>

Table 7. \( n = 37, w_r = 1, w_t = 2 \)
<table>
<thead>
<tr>
<th>r</th>
<th>solved app</th>
<th>avg gap</th>
<th>max gap</th>
<th>avg time</th>
<th>solved exact</th>
<th>proven exact</th>
<th>avg time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>0.1</td>
<td>1</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>0.1</td>
<td>1</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>10</td>
<td>10</td>
<td>0:00</td>
</tr>
</tbody>
</table>

*Table 8. n = 100, \( w_r = 1, w_t = 2 \)*

<table>
<thead>
<tr>
<th>r</th>
<th>solved</th>
<th>avg gap</th>
<th>max gap</th>
<th>avg time</th>
<th>r</th>
<th>solved</th>
<th>avg gap</th>
<th>max gap</th>
<th>avg time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
<td>2</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:00</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>0.2</td>
<td>2</td>
<td>0:04</td>
<td>3</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:05</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>0.3</td>
<td>2</td>
<td>0:16</td>
<td>4</td>
<td>10</td>
<td>0.1</td>
<td>1</td>
<td>0:29</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>0.22</td>
<td>1</td>
<td>0:22</td>
<td>5</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0:34</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>0.13</td>
<td>1</td>
<td>2:57</td>
<td>6</td>
<td>10</td>
<td>0.4</td>
<td>2</td>
<td>2:06</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>0.5</td>
<td>2</td>
<td>3:36</td>
<td>7</td>
<td>10</td>
<td>0.2</td>
<td>1</td>
<td>6:41</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>13:25</td>
<td>8</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>12:35</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>0.33</td>
<td>1</td>
<td>4:23</td>
<td>9</td>
<td>7</td>
<td>0.43</td>
<td>2</td>
<td>19:08</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>0.2</td>
<td>3</td>
<td>0:21</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>16:16</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>7:54</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>12</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>21:34</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>39:55</td>
<td>13</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>39:55</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>14</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>15</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 9. GRAPPA-TP, n = 37 (left) and n = 100 (right).*
List of technical reports published by the University of Ulm
Some of them are available by FTP from ftp.informatik.uni-ulm.de
Reports marked with * are out of print

91-01  Ker-I Ko, P. Orponen, U. Schöning, O. Watanabe
Instance Complexity

91-02*  K. Gladitz, H. Fassbender, H. Vogler
Compiler-Based Implementation of Syntax-Directed Functional Programming

91-03*  Alfons Geser
Relative Termination

91-04*  J. Köbler, U. Schöning, J. Toran
Graph Isomorphism is low for PP

91-05  Johannes Köbler, Thomas Thierauf
Complexity Restricted Advice Functions

91-06*  Uwe Schöning
Recent Highlights in Structural Complexity Theory

91-07*  F. Green, J. Köbler, J. Toran
The Power of Middle Bit

91-08*  V. Arvind, Y. Han, L. Hamachandra, J. Köbler, A. Lozano, M. Mundhenk, A. Ogiwara, U. Schöning, R. Silvestri, T. Thierauf
Reductions for Sets of Low Information Content

92-01*  Vikraman Arvind, Johannes Köbler, Martin Mundhenk
On Bounded Truth-Table and Conjunctive Reductions to Sparse and Tally Sets

92-02*  Thomas Noll, Heiko Vogler
Top-down Parsing with Simulataneous Evaluation of Noncircular Attribute Grammars

92-03  Fakultät für Informatik
17. Workshop über Komplexitätstheorie, effiziente Algorithmen und Datenstrukturen

92-04*  V. Arvind, J. Köbler, M. Mundhenk
Lowness and the Complexity of Sparse and Tally Descriptions

92-05*  Johannes Köbler
Locating P/poly Optimally in the Extended Low Hierarchy

92-06*  Armin Kühnemann, Heiko Vogler
Synthesized and inherited functions - a new computational model for syntax-directed semantics

92-07*  Heinz Fassbender, Heiko Vogler
A Universal Unification Algorithm Based on Unification-Driven Leftmost Outermost Narrowing
92-08* Uwe Schöning
On Random Reductions from Sparse Sets to Tally Sets

92-09* Hermann von Hasseln, Laura Martignon
Consistency in Stochastic Network

92-10 Michael Schmitt
A Slightly Improved Upper Bound on the Size of Weights Sufficient to Represent Any Linearly Separable Boolean Function

92-11 Johannes Köbler, Seinosuke Toda
On the Power of Generalized MOD-Classes

92-12 V. Arvind, J. Köbler, M. Mundhenk
Reliable Reductions, High Sets and Low Sets

92-13 Alfons Geser
On a monotonic semantic path ordering

92-14* Joost Engelfriet, Heiko Vogler
The Translation Power of Top-Down Tree-To-Graph Transducers

93-01 Alfred Lupper, Konrad Froitzheim
AppleTalk Link Access Protocol basierend auf dem Abstract Personal Communications Manager

The COCOON Object Model

93-03 Thomas Thierauf, Seinosuke Toda, Osamu Watanabe
On Sets Bounded Truth-Table Reducible to P-selective Sets

93-04 Jin-Yi Cai, Frederic Green, Thomas Thierauf
On the Correlation of Symmetric Functions

93-05 K.Kuhn, M.Reichert, M. Nathe, T. Beuter, C. Heinlein, P. Dadam
A Conceptual Approach to an Open Hospital Information System

93-06 Klaus Gaßner
Rechnerunterstützung für die konzeptuelle Modellierung

93-07 Ullrich Keßler, Peter Dadam
Towards Customizable, Flexible Storage Structures for Complex Objects

94-01 Michael Schmitt
On the Complexity of Consistency Problems for Neurons with Binary Weights

94-02 Armin Kühnemann, Heiko Vogler
A Pumping Lemma for Output Languages of Attributed Tree Transducers

94-03 Harry Buhrman, Jim Kadin, Thomas Thierauf
On Functions Computable with Nonadaptive Queries to NP

94-04 Heinz Faßbender, Heiko Vogler, Andrea Wedel
Implementation of a Deterministic Partial E-Unification Algorithm for Macro Tree Transducers
94-05  V. Arvind, J. Köbler, R. Schuler  
On Helping and Interactive Proof Systems

94-06  Christian Kalus, Peter Dadam  
Incorporating record subtyping into a relational data model

94-07  Markus Tresch, Marc H. Scholl  
A Classification of Multi-Database Languages

94-08  Friedrich von Henke, Harald Rueß  
Arbeitstreffen Typtheorie: Zusammenfassung der Beiträge

Construction and Deduction Methods for the Formal Development of Software

94-10  Axel Dold  
Formalisierung schematischer Algorithmen

94-11  Johannes Köbler, Osamu Watanabe  
New Collapse Consequences of NP Having Small Circuits

94-12  Rainer Schuler  
On Average Polynomial Time

94-13  Rainer Schuler, Osamu Watanabe  
Towards Average-Case Complexity Analysis of NP Optimization Problems

94-14  Wolfram Schulte, Ton Vullinghs  
Linking Reactive Software to the X-Window System

94-15  Alfred Lupper  
Namensverwaltung und Adressierung in Distributed Shared Memory-Systemen

94-16  Robert Regn  
Verteilte Unix-Betriebssysteme

94-17  Helmuth Partsch  
Again on Recognition and Parsing of Context-Free Grammars: Two Exercises in Transformational Programming

94-18  Helmuth Partsch  
Transformational Development of Data-Parallel Algorithms: an Example

95-01  Oleg Verbitsky  
On the Largest Common Subgraph Problem

95-02  Uwe Schöning  
Complexity of Presburger Arithmetic with Fixed Quantifier Dimension

95-03  Harry Buhrman, Thomas Thierauf  
The Complexity of Generating and Checking Proofs of Membership

95-04  Rainer Schuler, Tomoyuki Yamakami  
Structural Average Case Complexity

95-05  Klaus Achatz, Wolfram Schulte  
Architecture Independent Massive Parallelization of Divide-And-Conquer Algorithms
95-06  **Christoph Karg, Rainer Schuler**
Structure in Average Case Complexity

95-07  **P. Dadam, K. Kuhn, M. Reichert, T. Beuter, M. Nathe**
ADEPT: Ein integrierender Ansatz zur Entwicklung flexibler, zuverlässiger
kooperierender Assistenten in klinischen Anwendungsumgebungen

95-08  **Jürgen Kehrer, Peter Schulthess**
Aufbereitung von gescannten Röntgenbildern zur filmlosen Diagnostik

95-09  **Hans-Jörg Burtschick, Wolfgang Lindner**
On Sets Turing Reducible to P-Selective Sets

95-10  **Boris Hartmann**
Berücksichtigung lokaler Randbedingung bei globaler Zieloptimierung mit neuronalen
Netzen am Beispiel Truck Backer-Upper

95-12  **Klaus Achatz, Wolfram Schulte**
Massive Parallelization of Divide-and-Conquer Algorithms over Powerlists

95-13  **Andrea Mößle, Heiko Vogler**
Efficient Call-by-value Evaluation Strategy of Primitive Recursive Program Schemes

95-14  **Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß**
A Generic Specification for Verifying Peephole Optimizations

96-01  **Ercüment Canver, Jan-Tecker Gayen, Adam Moik**
Formale Entwicklung der Steuerungssoftware für eine elektrisch ortsbediente Weiche
mit VSE

96-02  **Bernhard Nebel**
Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of
Using the ORD-Horn Class

96-03  **Ton Vullinghs, Wolfram Schulte, Thilo Schwinn**
An Introduction to TkGofer

96-04  **Thomas Beuter, Peter Dadam**
Anwendungsspezifische Anforderungen an Workflow-Mangement-Systeme am
Beispiel der Domäne Concurrent-Engineering

96-05  **Gerhard Schellhorn, Wolfgang Ahrendt**
Verification of a Prolog Compiler - First Steps with KIV

96-06  **Manindra Agrawal, Thomas Thierauf**
Satisfiability Problems

96-07  **Vikraman Arvind, Jacobo Torán**
A nonadaptive NC Checker for Permutation Group Intersection

96-08  **David Cyrluk, Oliver Möller, Harald Rueß**
An Efficient Decision Procedure for a Theory of Fix-Sized Bitvectors with
Composition and Extraction

96-09  **Bernd Biechele, Dietmar Ernst, Frank Houdek, Joachim Schmid, Wolfram Schulte**
Erfahrungen bei der Modellierung eingebetteter Systeme mit verschiedenen SA/RT–
Ansätzen
96-10  Falk Bartels, Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß
Formalizing Fixed-Point Theory in PVS

96-11  Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß
Mechanized Semantics of Simple Imperative Programming Constructs

96-12  Axel Dold, Friedrich W. von Henke, Holger Pfeifer, Harald Rueß
Generic Compilation Schemes for Simple Programming Constructs

96-13  Klaus Achatz, Helmut Partsch
From Descriptive Specifications to Operational ones: A Powerful Transformation
Rule, its Applications and Variants

97-01  Jochen Messner
Pattern Matching in Trace Monoids

97-02  Wolfgang Lindner, Rainer Schuler
A Small Span Theorem within P

97-03  Thomas Bauer, Peter Dadam
A Distributed Execution Environment for Large-Scale Workflow Management
Systems with Subnets and Server Migration

97-04  Christian Heinlein, Peter Dadam
Interaction Expressions - A Powerful Formalism for Describing Inter-Workflow
Dependencies

97-05  Vikraman Arvind, Johannes Köbler
On Pseudorandomness and Resource-Bounded Measure

97-06  Gerhard Partsch
Punkt-zu-Punkt- und Mehrpunkt-basierende LAN-Integrationsstrategien für den
digitalen Mobilfunkstandard DECT

97-07  Manfred Reichert, Peter Dadam
ADEPT $^\text{flex}$ - Supporting Dynamic Changes of Workflows Without Loosing Control

97-08  Hans Braxmeier, Dietmar Ernst, Andrea Mößle, Heiko Vogler
The Project NoName - A functional programming language with its development
environment

97-09  Christian Heinlein
Grundlagen von Interaktionsausdrücken

97-10  Christian Heinlein
Graphische Repräsentation von Interaktionsausdrücken

97-11  Christian Heinlein
Sprachtheoretische Semantik von Interaktionsausdrücken

97-12  Gerhard Schellhorn, Wolfgang Reif
Proving Properties of Finite Enumerations: A Problem Set for Automated Theorem
Provers
Dietmar Ernst, Frank Houdek, Wolfram Schulte, Thilo Schwinn
Experimenteller Vergleich statischer und dynamischer Softwareprüfung für eingebettete Systeme

Wolfgang Reif, Gerhard Schellhorn
Theorem Proving in Large Theories

Thomas Wennekers
Asymptotik rekurrenter neuronaler Netze mit zufälligen Kopplungen

Peter Dadam, Klaus Kuhn, Manfred Reichert
Clinical Workflows - The Killer Application for Process-oriented Information Systems?

Mohammad Ali Livani, Jörg Kaiser
EDF Consensus on CAN Bus Access in Dynamic Real-Time Applications

Johannes Köbler, Rainer Schuler
Using Efficient Average-Case Algorithms to Collapse Worst-Case Complexity Classes

Daniela Damm, Lutz Claes, Friedrich W. von Henke, Alexander Seitz, Adelinde Uhrmacher, Steffen Wolf
Ein fallbasiertes System für die Interpretation von Literatur zur Knochenheilung

Thomas Bauer, Peter Dadam
Architekturen für skalierbare Workflow-Management-Systeme - Klassifikation und Analyse

Marko Luther, Martin Strecker
A guided tour through Typelab

Heiko Neumann, Luiz Pessoa
Visual Filling-in and Surface Property Reconstruction

Ercüment Canver
Formal Verification of a Coordinated Atomic Action Based Design

Andreas Küchler
On the Correspondence between Neural Folding Architectures and Tree Automata

Heiko Neumann, Thorsten Hansen, Luiz Pessoa
Interaction of ON and OFF Pathways for Visual Contrast Measurement

Thomas Wennekers
Synfire Graphs: From Spike Patterns to Automata of Spiking Neurons

Thomas Bauer, Peter Dadam
Variable Migration von Workflows in ADEPT

Heiko Neumann, Wolfgang Sepp
Recurrent V1 – V2 Interaction in Early Visual Boundary Processing

Frank Houdek, Dietmar Ernst, Thilo Schwinn
Prüfen von C–Code und Statmate/Matlab–Spezifikationen: Ein Experiment
98-12 Gerhard Schellhorn
Proving Properties of Directed Graphs: A Problem Set for Automated Theorem Provers

98-13 Gerhard Schellhorn, Wolfgang Reif
Theorems from Compiler Verification: A Problem Set for Automated Theorem Provers

98-14 Mohammad Ali Livani
SHARE: A Transparent Mechanism for Reliable Broadcast Delivery in CAN

98-15 Mohammad Ali Livani, Jörg Kaiser
Predictable Atomic Multicast in the Controller Area Network (CAN)

99-01 Susanne Boll, Wolfgang Klas, Utz Westermann
A Comparison of Multimedia Document Models Concerning Advanced Requirements

99-02 Thomas Bauer, Peter Dadam
Verteilungsmodelle für Workflow-Management-Systeme - Klassifikation und Simulation

99-03 Uwe Schöning
On the Complexity of Constraint Satisfaction

99-04 Ercument Canver
Model-Checking zur Analyse von Message Sequence Charts über Statecharts

99-05 Johannes Köbler, Wolfgang Lindner, Rainer Schuler
Derandomizing RP if Boolean Circuits are not Learnable

99-06 Utz Westermann, Wolfgang Klas
Architecture of a DataBlade Module for the Integrated Management of Multimedia Assets

99-07 Peter Dadam, Manfred Reichert

99-08 Vikraman Arvind, Johannes Köbler
Graph Isomorphism is Low for ZPP^{NP} and other Lowness results

99-09 Thomas Bauer, Peter Dadam
Efficient Distributed Workflow Management Based on Variable Server Assignments

2000-02 Thomas Bauer, Peter Dadam
Variable Serverzuordnungen und komplexe Bearbeiterzuordnungen im Workflow-Management-System ADEPT

2000-03 Gregory Baratoff, Christian Toepfer, Heiko Neumann
Combined space-variant maps for optical flow based navigation

2000-04 Wolfgang Gehring
Ein Rahmenwerk zur Einführung von Leistungspunktsystemen
2000-05  *Susanne Boll, Christian Heinlein, Wolfgang Klas, Jochen Wandel*
Intelligent Prefetching and Buffering for Interactive Streaming of MPEG Videos

2000-06  *Wolfgang Reif, Gerhard Schellhorn, Andreas Thums*
Fehlersuche in Formalen Spezifikationen

2000-07  *Gerhard Schellhorn, Wolfgang Reif (eds.)*

2000-08  *Thomas Bauer, Manfred Reichert, Peter Dadam*
Effiziente Durchführung von Prozessmigrationen in verteilten Workflow-Management-Systemen

2000-09  *Thomas Bauer, Peter Dadam*
Vermeidung von Überlastsituationen durch Replikation von Workflow-Servern in ADEPT

2000-10  *Thomas Bauer, Manfred Reichert, Peter Dadam*
Adaptives und verteiltes Workflow-Management

2000-11  *Christian Heinlein*
Workflow and Process Synchronization with Interaction Expressions and Graphs

2001-01  *Hubert Hug, Rainer Schuler*
DNA-based parallel computation of simple arithmetic

2001-02  *Friedhelm Schwenker, Hans A. Kestler, Günther Palm*
3-D Visual Object Classification with Hierarchical Radial Basis Function Networks

2001-03  *Hans A. Kestler, Friedhelm Schwenker, Günther Palm*
RBF network classification of ECGs as a potential marker for sudden cardiac death

2001-04  *Christian Dietrich, Friedhelm Schwenker, Klaus Riede, Günther Palm*
Classification of Bioacoustic Time Series Utilizing Pulse Detection, Time and Frequency Features and Data Fusion

2002-01  *Stefanie Rinderle, Manfred Reichert, Peter Dadam*
Effiziente Verträglichkeitsprüfung und automatische Migration von Workflow-Instanzen bei der Evolution von Workflow-Schemata

2002-02  *Walter Guttmann*
Deriving an Applicative Heapsort Algorithm

2002-03  *Axel Dold, Friedrich W. von Henke, Vincent Vialard, Wolfgang Goerigk*
A Mechanically Verified Compiling Specification for a Realistic Compiler

2003-01  *Manfred Reichert, Stefanie Rinderle, Peter Dadam*
A Formal Framework for Workflow Type and Instance Changes Under Correctness Checks

2003-02  *Stefanie Rinderle, Manfred Reichert, Peter Dadam*
Supporting Workflow Schema Evolution By Efficient Compliance Checks

2003-03  *Christian Heinlein*
Safely Extending Procedure Types to Allow Nested Procedures as Values
2003-04  Stefanie Rinderle, Manfred Reichert, Peter Dadam
On Dealing With Semantically Conflicting Business Process Changes.

2003-05  Christian Heinlein
Dynamic Class Methods in Java

2003-06  Christian Heinlein
Vertical, Horizontal, and Behavioural Extensibility of Software Systems

2003-07  Christian Heinlein
Safely Extending Procedure Types to Allow Nested Procedures as Values
(Corrected Version)

2003-08  Changling Liu, Jörg Kaiser
Survey of Mobile Ad Hoc Network Routing Protocols

2004-01  Thom Frühwirth, Marc Meister (eds.)
First Workshop on Constraint Handling Rules

2004-02  Christian Heinlein
Concept and Implementation of C+++, an Extension of C++ to Support User-Defined
Operator Symbols and Control Structures

2004-03  Susanne Biundo, Thom Frühwirth, Günther Palm(eds.)
Poster Proceedings of the 27th Annual German Conference on Artificial Intelligence

2005-01  Armin Wolf, Thom Frühwirth, Marc Meister (eds.)
19th Workshop on (Constraint) Logic Programming

2005-02  Wolfgang Lindner (Hg.), Universität Ulm , Christopher Wolf (Hg.) KU Leuven
2. Krypto-Tag – Workshop über Kryptographie, Universität Ulm

2005-03  Walter Guttmann, Markus Maucher
Constrained Ordering

2006-01  Stefan Sarstedt
Model-Driven Development with ACTIVECHARTS, Tutorial

2006-02  Alexander Raschke, Ramin Tavakoli Kolagari
Ein experimenteller Vergleich zwischen einer plan-getriebenen und einer
leichtgewichtigen Entwicklungsmethode zur Spezifikation von eingebetteten
Systemen

2006-03  Jens Kohlmeyer, Alexander Raschke, Ramin Tavakoli Kolagari
Eine qualitative Untersuchung zur Produktlinien-Integration über
Organisationsgrenzen hinweg

2006-04  Thorsten Liebig
Reasoning with OWL - System Support and Insights –

2008-01  H.A. Kestler, J. Messner, A. Müller, R. Schuler
On the complexity of intersecting multiple circles for graphical display
<table>
<thead>
<tr>
<th>Year</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008-03</td>
<td>Frank Raiser</td>
<td>Semi-Automatic Generation of CHR Solvers from Global Constraint Automata</td>
</tr>
<tr>
<td>2008-04</td>
<td>Ramin Tavakoli Kolagari, Alexander Raschke, Matthias Schneiderhan, Ian Alexander</td>
<td>Entscheidungsdokumentation bei der Entwicklung innovativer Systeme für produktlinien-basierte Entwicklungsprozesse</td>
</tr>
<tr>
<td>2008-05</td>
<td>Markus Kalb, Claudia Dittrich, Peter Dadam</td>
<td>Support of Relationships Among Moving Objects on Networks</td>
</tr>
<tr>
<td>2008-06</td>
<td>Matthias Frank, Frank Kargl, Burkhard Stiller (Hg.)</td>
<td>WMAN 2008 – KuVS Fachgespräch über Mobile Ad-hoc Netzwerke</td>
</tr>
<tr>
<td>2008-07</td>
<td>M. Maucher, U. Schöning, H.A. Kestler</td>
<td>An empirical assessment of local and population based search methods with different degrees of pseudorandomness</td>
</tr>
<tr>
<td>2008-08</td>
<td>Henning Wunderlich</td>
<td>Covers have structure</td>
</tr>
<tr>
<td>2008-09</td>
<td>Karl-Heinz Niggl, Henning Wunderlich</td>
<td>Implicit characterization of FPTIME and NC revisited</td>
</tr>
<tr>
<td>2008-10</td>
<td>Henning Wunderlich</td>
<td>On span-$P^{cc}$ and related classes in structural communication complexity</td>
</tr>
<tr>
<td>2008-11</td>
<td>M. Maucher, U. Schöning, H.A. Kestler</td>
<td>On the different notions of pseudorandomness</td>
</tr>
<tr>
<td>2008-12</td>
<td>Henning Wunderlich</td>
<td>On Toda’s Theorem in structural communication complexity</td>
</tr>
<tr>
<td>2008-13</td>
<td>Manfred Reichert, Peter Dadam</td>
<td>Realizing Adaptive Process-aware Information Systems with ADEPT2</td>
</tr>
<tr>
<td>2009-01</td>
<td>Peter Dadam, Manfred Reichert</td>
<td>The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support Challenges and Achievements</td>
</tr>
</tbody>
</table>
2009-03  *Alena Hallerbach, Thomas Bauer, Manfred Reichert*  
Correct Configuration of Process Variants in Provop

2009-04  *Martin Bader*  
On Reversal and Transposition Medians
Ulmer Informatik-Berichte
ISSN 0939-5091

Herausgeber:
Universität Ulm
Fakultät für Ingenieurwissenschaften und Informatik
89069 Ulm