Computing Kemeny Rankings, Parameterized by the Average KT-Distance

Nadja Betzler, Michael R. Fellows, Jiong Guo, Rolf Niedermeier, and Frances A. Rosamond

Abstract

The computation of Kemeny rankings is central to many applications in the context of rank aggregation. Unfortunately, the problem is NP-hard. Extending our previous work [AAIM 2008], we show that the Kemeny score of an election can be computed efficiently whenever the *average* pairwise distance between two input votes is not too large. In other words, KEMENY SCORE is fixed-parameter tractable with respect to the parameter "average pairwise Kendall-Tau distance d_a". We describe a fixed-parameter algorithm with running time $O(16^{\lceil d_a \rceil} \cdot \text{poly})$.

1 Introduction

Aggregating inconsistent information has many applications ranging from voting scenarios to meta search engines and fighting spam [1, 4, 5, 7]. In some sense, one deals with *consensus problems* where one wants to find a solution to various "input demands" such that these demands are met as well as possible. Naturally, contradicting demands cannot be fulfilled at the same time. Hence, the consensus solution has to provide a balance between opposing requirements. The concept of *Kemeny consensus* (or Kemeny ranking) is among the most important research topics in this context. In this paper, extending our previous work [3], we study new algorithmic approaches based on parameterized complexity analysis [6, 9, 13] for efficiently computing optimal Kemeny consensus solutions in practically relevant special cases.

Kemeny's voting scheme can be described as follows. An *election* (V, C) consists of a set V of n votes and a set C of m candidates. A vote is a *preference list* of the candidates. For instance, in the case of three candidates a, b, c, the order $c > b > a$ would mean that candidate c is the best-liked and candidate a is the least-liked for this voter. A "Kemeny consensus" is a preference list that is "closest" to the preference lists of the voters. For each pair of votes v, w, the so-called *Kendall-Tau distance* (*KT-distance* for short) between v and w, also known as the number of inversions between two permutations, is defined as

$$\text{KT-dist}(v, w) = \sum_{\{c,d\} \subseteq C} d_{v,w}(c, d),$$

where the sum is taken over all unordered pairs $\{c, d\}$ of candidates, and $d_{v,w}(c, d)$ is 0 if v and w rank c and d in the same order, and 1 otherwise. Using divide-and-conquer, the KT-distance can be computed in $O(m \cdot \log m)$ time [12]. The *score* of a preference list l with respect to an election (V, C) is defined as $\sum_{v \in V} \text{KT-dist}(l, v)$. A preference list l with the minimum score is called a *Kemeny consensus* of (V, C) and its score $\sum_{v \in V} \text{KT-dist}(l, v)$ is the *Kemeny score* of (V, C), denoted as K-score(V, C). The underlying decision problem is as follows:

KEMENY SCORE
Input: An election (V, C) and a positive integer k.
Question: Is K-score$(V, C) \leq k$?

Known results. We summarize the state of the art concerning the computational complexity of KEMENY SCORE. Bartholdi et al. [2] showed that KEMENY SCORE is NP-complete, and it remains so even when restricted to instances with only four votes [7, 8]. Given the computational hardness of KEMENY SCORE on the one side and its practical relevance on the other side, polynomial-time approximation algorithms have been studied. The Kemeny score can be approximated to a factor of 8/5 by a deterministic algorithm [15] and to a factor of 11/7 by a randomized algorithm [1]. Recently, a polynomial-time approximation scheme (PTAS) has been developed [11]. However, the running time is completely impractical and the result is only of theoretical interest. Conitzer, Davenport, and Kalagnanam [5, 4] performed computational studies for the efficient exact computation of a Kemeny consensus, using heuristic approaches such as greedy and branch-and-bound. Hemaspaandra et al. [10] provided further, exact classifications of the classical computational complexity of Kemeny elections. Very recently, we initiated a parameterized complexity study based on various problem parameterizations [3]. We obtained fixed-parameter tractability results for the parameters "score", "number of candidates", "maximum KT-distance between two input votes", and "maximum position range of a candidate".[1] For more details, see Section 2

New results. Our main result is that KEMENY SCORE can be solved in $O(16^{\lceil d_a \rceil} \cdot \text{poly}(n,m))$ time, where d_a denotes the average KT-distance between the pairs of input votes. This represents a significant improvement of the previous algorithm for the maximum KT-distance between pairs of input votes, which has running time $O((3d_{\max}+1)! \cdot \text{poly}(n,m))$. Clearly, $d_a \leq d_{\max}$.

1.1 Preliminaries

Let the *position* of a candidate c in a vote v, denoted by $v(c)$, be the number of candidates that are better than c in v. That is, the leftmost (and best) candidate in v has position 0 and the rightmost has position $m-1$. For an election (V,C) and a candidate $c \in C$, the *average position* $p_a(c)$ of c is defined as

$$p_a(c) := \frac{1}{n} \cdot \sum_{v \in V} v(c).$$

For an election (V,C) the average KT-distance d_a is defined as

$$d_a := \frac{1}{n(n-1)} \cdot \sum_{\{u,v\} \in V, u \neq v} \text{KT-dist}(u,v).$$

Note that an equivalent definition is given by

$$d_a := \frac{1}{n(n-1)} \cdot \sum_{a,b \in C} \#v(a > b) \cdot \#v(b > a),$$

where for two candidates a and b the number of input votes in which a is ranked better than b is denoted by $\#v(a > b)$. This definition is useful if the input is provided by the outcomes of the pairwise elections of the candidates including the margins of victory.

We briefly introduce the relevant notions of parameterized complexity theory [6, 9, 13]. Parameterized algorithmics aims at a multivariate complexity analysis of problems. This is done by studying relevant problem parameters and their influence on the computational complexity of problems. The hope lies in accepting the seemingly inevitable combinatorial

[1]The parameterization by position range has only been discussed in the long version of [3].

explosion for NP-hard problems, but confining it to the parameter. Hence, the decisive question is whether a given parameterized problem is *fixed-parameter tractable (FPT)* with respect to the parameter, often denoted k. In other words, for an input instance I together with the parameter k, we ask for the existence of a solving algorithm with running time $f(k) \cdot \text{poly}(|I|)$ for some computable function f.

2 Parameterizations of the Kemeny Score Problem

In recent work [3], we initiated a parameterized complexity study of KEMENY SCORE. In this section, we review the considered parameterizations and results.

An election can be interpreted as having (at least) two "dimensions", the set of votes and the set of candidates. Thus, the "number n of votes" and the "number m of candidates" lead to natural parameterizations. Fixed-parameter tractability with respect to the parameter "number of votes" would imply P=NP since KEMENY SCORE is already NP-complete for instances with four votes [7]. In contrast, concerning m as a parameter, there is a trivial algorithm that tests all $m!$ orderings of the candidates for a Kemeny consensus. Using a dynamic programming approach, we were able to lower the combinatorial explosion in m; more specifically, we provided an exact algorithm running in $O(2^m \cdot m^2 n)$ time [3].[2]

A common parameterization in parameterized algorithmics is the size of the solution of a problem, motivating the consideration of "Kemeny score k" as a parameter. Using the Kemeny score as a parameter, a preprocessing procedure (that is, a "problem kernelization") together with a search tree approach led to a fixed-parameter algorithm that runs in $O(1.53^k + m^2 n)$ time. The drawback of this parameterization is that the Kemeny score can become large for many instances.

Finally, we turned our attention to two structural parameterizations: the "maximum range of candidate positions" and the "maximum KT-distance".[3] For an election (V, C), the maximum range r of candidate positions is defined as

$$r := \max_{v,w \in V, c \in C} \{|v(c) - w(c)|\}.$$

For this parameterization, we developed a dynamic programming algorithm that is based on the observation that we can "decompose" the input votes into two parts. The resulting running time is $O((3r+1)! \cdot r \log r \cdot mn)$. Further, the maximum KT-distance $d_{\max}$ is defined as

$$d_{\max} := \max_{v,w \in V} \text{KT-dist}(v, w).$$

We showed that for every election we have $r \leq d_{\max}$. Thus, our results for the maximum range of candidate positions also hold for the maximum distance. The parameterization by $d_{\max}$ was the main reason to study the parameterization by the maximum range of candidate positions and, further, motivated us to consider the average distance as parameter in this work.

In our previous work [3], we also extended some of our findings to two generalizations of KEMENY SCORE; in one case allowing ties and in the other case dealing with incomplete information. For more details, we refer to there [3].

[2]In a different context, this result has been independently achieved by Raman et al. [14].

[3]In the conference version of [3] we only dealt with the maximum KT-distance as parameter whereas in the full version (invited for submission to a special issue of *Theoretical Computer Science*) we discussed both structural parameterizations as we do here.

3 Parameter "Average KT-Distance"

In this section, we further extend the range of parameterizations studied so far by giving a fixed-parameter algorithm with respect to the parameter "average KT-distance". We start with showing how the average KT-distance can be used to upper-bound the range of positions that a candidate can take in any optimal Kemeny consensus. Based on this crucial observation, we then state the algorithm.

3.1 A Crucial Observation

Our fixed-parameter tractability result with respect to the average KT-distance of the input is based on the following lemma.

Lemma 1. *Let d_a be the average KT-distance of an election (V, C) and $d := \lceil d_a \rceil$. Then, in every optimal Kemeny consensus l, for every candidate $c \in C$ with respect to its average position $p_a(c)$ we have $p_a(c) - d < l(c) < p_a(c) + d$.*

Proof. The proof is by contradiction and consists of two claims: First, we show that we can find a vote with Kemeny score less than $d \cdot n$, that is, the Kemeny score of the instance is upper-bounded by $d \cdot n$. Second, we show that in every Kemeny consensus every candidate is in the claimed range. More specifically, we prove that every consensus in which the position of a candidate is not in a "range d of its average position" has a Kemeny score greater than $d \cdot n$, a contradiction to the first claim.

Claim 1: K-score$(V, C) < d \cdot n$.

Proof of Claim 1: To prove Claim 1, we show that there is a vote $v \in V$ with $\sum_{w \in V}$ KT-dist$(v, w) < d \cdot n$, implying this upper bound for an optimal Kemeny consensus as well. By definition,

$$d_a = \frac{1}{n(n-1)} \cdot \sum_{\{v,w\} \in V, v \neq w} \text{KT-dist}(v, w) \tag{1}$$

$$\Rightarrow \exists v \in V \text{ with } d_a \geq \frac{1}{n(n-1)} \cdot n \cdot \sum_{w \in V} \text{KT-dist}(v, w) = \frac{1}{n-1} \cdot \sum_{w \in V} \text{KT-dist}(v, w) \tag{2}$$

$$\Rightarrow \exists v \in V \text{ with } d_a \cdot n > \sum_{w \in V} \text{KT-dist}(v, w). \tag{3}$$

Since we have $d = \lceil d_a \rceil$, Claim 1 follows directly from Inequality (3).

The next claim shows the given bound on the range of possible candidates positions.

Claim 2: In every optimal Kemeny consensus l, every candidate $c \in C$ fulfills $p_a(c) - d < l(c) < p_a(c) + d$.

Proof of Claim 2: We start by showing that, for every candidate $c \in C$ we have

$$\text{K-score}(V, C) \geq \sum_{v \in V} |l(c) - v(c)|. \tag{4}$$

Note that, for every candidate $c \in C$, for two votes v, w we must have KT-dist$(v, w) \geq |v(c) - w(c)|$. Without loss of generality, assume that $v(c) > w(c)$. Then, there must be at least $v(c) - w(c)$ candidates that have a smaller position than c in v and that have a greater position than c in w. Further, each of these candidates increases the value of KT-dist(v, w)

by one. Based on this, Inequality (4) directly follows as, by definition, K-score$(V, C) = \sum_{v \in V} \text{KT-dist}(v, l)$.

To simplify the proof of Claim 2, in the following, we shift the positions in l such that $l(c) = 0$. Accordingly, we shift the positions in all votes in V, that is, for every $v \in V$ and every $a \in C$, we decrease $v(a)$ by the original value of $l(c)$. Clearly, shifting all positions does not affect the relative differences of positions between two candidates. Then, let the set of votes in which c has a nonnegative position be V^+ and let V^- denote the remaining set of votes, that is, $V^- := V \backslash V^+$.

Now, we show that if candidate c is placed outside of the given range in an optimal Kemeny consensus l, then K-score$(V, C) > d \cdot n$. The proof is by contradiction. We distinguish two cases:

Case 1: $l(c) \geq p_a(c) + d$.
As $l(c) = 0$, in this case $p_a(c)$ becomes negative. Then,

$$0 \geq p_a(c) + d \Leftrightarrow -p_a(c) \geq d.$$

It follows that $|p_a(c)| \geq d$. The following shows that Claim 2 holds for this case.

$$\sum_{v \in V} |l(c) - v(c)| = \sum_{v \in V} |v(c)| = \sum_{v \in V^+} |v(c)| + \sum_{v \in V^-} |v(c)|. \tag{5}$$

Next, replace the term $\sum_{v \in V^-} |v(c)|$ in (5) by an equivalent term that depends on $|p_a(c)|$ and $\sum_{v \in V^+} |v(c)|$. For this, use the following, derived from the definition of $p_a(c)$:

$$\begin{aligned} n \cdot p_a(c) &= \sum_{v \in V^+} |v(c)| - \sum_{v \in V^-} |v(c)| \\ \Leftrightarrow \sum_{v \in V^-} |v(c)| &= n \cdot (-p_a(c)) + \sum_{v \in V^+} |v(c)| = n \cdot |p_a(c)| + \sum_{v \in V^+} |v(c)|. \end{aligned}$$

The replacement results in

$$\sum_{v \in V} |l(c) - v(c)| = 2 \cdot \sum_{v \in V^+} |v(c)| + n \cdot |p_a(c)| \geq n \cdot |p_a(c)| \geq n \cdot d.$$

This says that K-score$(V, C) \geq n \cdot d$, a contradiction to Claim 1.

Case 2: $l(c) \leq p_a(c) - d$.
Since $l(c) = 0$, the condition is equivalent to $0 \leq p_a(c) - d \Leftrightarrow d \leq p_a(c)$, and we have that $p_a(c)$ is nonnegative. Now, we show that Claim 2 also holds for this case.

$$\begin{aligned} \sum_{v \in V} |l(c) - v(c)| = \sum_{v \in V} |v(c)| &= \sum_{v \in V^+} |v(c)| + \sum_{v \in V^-} |v(c)| \\ &\geq \sum_{v \in V^+} v(c) + \sum_{v \in V^-} v(c) = p_a(c) \cdot n \geq d \cdot n. \end{aligned}$$

Thus, also in this case K-score$(V, C) \geq n \cdot d$, a contradiction to Claim 1. □

Based on Lemma 1, for every position we can define the set of candidates that can take this position in an optimal Kemeny consensus. The subsequent definition will be useful for the formulation of the algorithm.

Definition 1. Let (V, C) be an election. For $i \in \{0, \ldots, m-1\}$, let P_i denote the set of candidates that can assume the position i in an optimal Kemeny consensus, that is, $P_i := \{c \in C \mid p_a(c) - d < i < p_a(c) + d\}$.

Based on Lemma 1, we can easily show the following.

Lemma 2. *For every position i, the size of P_i is at most $4d$.*

Proof. The proof is by contradiction. Assume that there is a position i with $|P_i| > 4d$. Due to Lemma 1, for every candidate $c \in P_i$ the positions which c may assume in an optimal Kemeny consensus can differ by at most $2d-1$. This is true because, otherwise, candidate c could not be in the given range around its average position. Then, in a Kemeny consensus, each of the at least $4d+1$ candidates must hold a position that differs at most by $2d-1$ from position i. As there are only $4d-1$ such positions ($2d-1$ on the left and $2d-1$ on the right of i), one obtains a contradiction. □

3.2 Basic Idea of the Algorithm

In Subsection 3.4, we will present a dynamic programming algorithm for Kemeny Score. It exploits the fact that every candidate can only appear in a fixed range of positions in an optimal Kemeny consensus.[4] The algorithm "generates" a Kemeny consensus from the left to the right. It tries out all possibilities for ordering the candidates locally and then combines these local solutions to yield a Kemeny consensus.

More specifically, according to Lemma 2 the number of candidates that can take a position i in an optimal Kemeny consensus for any $0 \leq i \leq m-1$ is at most $4d$. Thus, for position i, we can test all possible candidates. Having chosen a candidate for position i, the remaining candidates that could also assume i must either be left or right of i in a Kemeny consensus. Thus, we test all possible two-partitionings of this subset of candidates and compute a "partial" Kemeny score for every possibility. For the computation of the partial Kemeny scores at position i we make use of the partial solutions computed for the previous position $i-1$.

3.3 Definitions for the Algorithm

To state the algorithm, we need some further definitions. For $i \in \{0, \ldots, m-1\}$, let $I(i)$ denote the set of candidates that could be "inserted" at position i *for the first time*, that is,

$$I(i) := \{c \in C \mid c \in P_i \text{ and } c \notin P_{i-1}\}.$$

Let $F(i)$ denote the set of candidates that must be "forgotten" at latest at position i, that is,

$$F(i) := \{c \in C \mid c \notin P_i \text{ and } c \in P_{i-1}\}.$$

For our algorithm, it is essential to subdivide the overall Kemeny score into *partial Kemeny scores* (pK). More precisely, for a candidate c and a subset of candidates R with $c \notin R$, we set

$$\text{pK}(c, R) := \sum_{c' \in R} \sum_{v \in V} d_v^R(c, c'),$$

where for $c \notin R$ and $c' \in R$ we have $d_v^R(c, c') := 0$ if in v we have $c' > c$, and $d_v^R(c, c') := 1$, otherwise. Intuitively, the partial Kemeny score denotes the score that is "induced" by

[4]In contrast, the previous dynamic programming algorithms from [3] for the parameters "maximum range of candidate positions" and "maximum KT-distance" rely on decomposing the input. Further, here we obtain a much better running time by using a more involved dynamic programming approach.

candidate c and the candidate subset R if the candidates of R have greater positions than c in an optimal Kemeny consensus.[5] Then, for a Kemeny consensus $l := c_0 > c_1 > \cdots > c_{m-1}$, the overall Kemeny score can be expressed by partial Kemeny scores as follows.

$$\text{K-score}(V, C) = \sum_{i=0}^{m-2} \sum_{j=i+1}^{m-1} \sum_{v \in V} d_{v,l}(c_i, c_j) \tag{6}$$

$$= \sum_{i=0}^{m-2} \sum_{c' \in R} \sum_{v \in V} d_v^R(c_i, c') \text{ for } R := \{c_j \mid i < j < m\} \tag{7}$$

$$= \sum_{i=0}^{m-2} \text{pK}(c_i, \{c_j \mid i < j < m\}). \tag{8}$$

Next, consider the three-dimensional dynamic programming table. Roughly speaking, define an entry for every position i, every candidate c that can assume i, and every candidate subset C' of $P_i \backslash \{c\}$. The entry stores the "minimum partial Kemeny score" over all possible orders of the candidates of C' under the condition that c takes position i and all candidates of C' take positions smaller than i. To define the dynamic programming table formally, we need some further notation.

Let $\Pi(C')$ denote the set of all possible orders of the candidates in C', where $C' \subseteq C$. Further, consider a Kemeny consensus in which every candidate of C' has a position smaller than every candidate in $C \backslash C'$. Then, the *minimum partial Kemeny score restricted to* C' is defined as

$$\min_{(c_1 > c_2 > \cdots > c_x) \in \Pi(C')} \left\{ \sum_{s=1}^{x} \text{pK}(c_s, \{c_j \mid s < j < m\} \cup (C \backslash C')) \right\} \text{ with } x := |C'|.$$

That is, it denotes the minimum partial Kemeny score over all orders of C'. We define an entry of the dynamic programming table T for a position i, a candidate $c \in P_i$, and a candidate subset $P_i' \subseteq P_i$ with $c \notin P_i'$. For this, we define $L := \bigcup_{j \leq i} F(j) \cup P_i'$. Then, an entry $T(i, c, P_i')$ denotes the minimum partial Kemeny score restricted to the candidates in $L \cup \{c\}$ under the assumptions that c is at position i in a Kemeny consensus, all candidates of L have positions smaller than i, and all other candidates have positions greater than i. That is, for $|L| = i - 1$, define

$$T(i, c, P_i') := \min_{(c_0 > \cdots > c_{i-1}) \in \Pi(L)} \sum_{s=0}^{i-1} \text{pK}(c_s, C \backslash \{c_j \mid j \leq s\}) + \text{pK}(c, C \backslash (L \cup \{c\})).$$

3.4 Dynamic Programming Algorithm

The algorithm is displayed in Fig. 1. It is easy to modify the algorithm such that it outputs an optimal Kemeny consensus: for every entry $T(i, c, P_i')$, one additionally has to store a candidate c' that minimizes $T(i-1, c', (P_i' \cup F(i)) \backslash \{c'\})$ in line *11*. Then, starting with a minimum entry for position $m-1$, we reconstruct a Kemeny consensus by iteratively adding the "predecessor" candidate. The asymptotic running time remains unchanged. Moreover, in several applications, it is helpful not having *one* optimal Kemeny consensus but to enumerate all of them. At the expense of an increased running time, our algorithm can be extended to provide such an enumeration by storing all possible predecessor candidates.

Lemma 3. *The algorithm in Fig. 1 correctly computes* KEMENY SCORE.

[5] By convention and somewhat counterintuitive, we say that candidate c has a greater position than candidate c' if $c' > c$ in a vote.

Input: An election (V, C) and, for every $0 \leq i < m$, the set P_i of candidates that can assume position i in an optimal Kemeny consensus.
Output: The Kemeny score of (V, C).

```
Initialization:
01 for  $i = 0, \ldots, m-1$
02    for all  $c \in P_i$
03       for all $P'_i \subseteq P_i \backslash \{c\}$
04          $T(i, c, P'_i) := +\infty$
05 for all $c \in P_0$
06    $T(0, c, \emptyset) := \mathrm{pK}(c, C \backslash \{c\})$

Update:
07 for  $i = 1, \ldots, m-1$
08    for all $c \in P_i$
09       for all $P'_i \subseteq P_i \backslash \{c\}$
10          if  $|P'_i \cup \bigcup_{j \leq i} F(j)| = i - 1$ and $T(i-1, c', (P'_i \cup F(i)) \backslash \{c'\})$ is defined then
11             $T(i, c, P'_i) = \min_{c' \in P'_i \cup F(i)} T(i-1, c', (P'_i \cup F(i)) \backslash \{c'\}) + \mathrm{pK}(c, (P_i \cup \bigcup_{i<j<m} I(j)) \backslash (P'_i \cup \{c\}))$

Output:
12 $K$-score $= \min_{c \in P_{m-1}} T(m-1, c, P_{m-1} \backslash \{c\})$
```

Figure 1: Dynamic programming algorithm for KEMENY SCORE

Proof. For the correctness, we have to show two points:

First, all table entries are well-defined, that is, for an entry $T(i, c, P'_i)$ concerning position i there must be exactly $i - 1$ candidates that have positions smaller than i. This condition is assured by line *10* of the algorithm.[6]

Second, we must ensure to find an optimal solution. Due to Equality (8), we know that the Kemeny score can be decomposed into partial Kemeny scores. Thus, it remains to show that the algorithm considers a decomposition that leads to an optimal solution. For every position the algorithm tries all candidates in P_i. According to Lemma 1, one of these candidates must be the "correct" candidate c for this position. Further, for c we can show that the algorithm tries a sufficient set of possibilities to partition all remaining candidates $C \backslash \{c\}$ such that they have either smaller or greater positions than i. More precisely, every candidate of $C \backslash \{c\}$ must be in exactly one of the following three subsets:

1. The set F of candidates that have already been forgotten, that is, $F := \bigcup_{0 \leq j \leq i} F(j)$,
2. the set of candidates that can assume position i, that is, $P_i \backslash \{c\}$, or
3. the set I of candidates that are not inserted yet, that is, $I := \bigcup_{i < j < m} I(j)$.

Due to Lemma 1 and the definition of $F(j)$, we know that a candidate of F cannot take a position greater than $i - 1$ in an optimal Kemeny consensus. Thus, it is sufficient to try only partitions in which the candidates of F have positions smaller than i. Analogously, one can argue that for all candidates in I it is sufficient to consider partitions in which they have positions greater than i. Thus, it remains to try all possibilities to partition the candidates of P_i. This is done in line *09* of the algorithm. Thus, the algorithm returns an optimal Kemeny score. ☐

Theorem 1. Kemeny Score *can be solved in* $O(n^2 \cdot m \log m + 16^d \cdot (16d^2 \cdot m + 4d \cdot m^2 \log m \cdot n))$ *time with average KT-distance* d_a *and* $d := \lceil d_a \rceil$. *The size of the dynamic programming table is* $O(16^d \cdot 4dm)$.

Proof. The dynamic programming procedure requires the set of candidates P_i for $0 \leq i < m$ as input. To determine P_i for all $0 \leq i < m$, we need the average positions of all candidates and the average KT-distance d_a of (V, C). To determine d_a, we compute the pairwise distances of all pairs of votes. As we have $O(n^2)$ pairs and the pairwise KT-distance can be computed in $O(m \log m)$ time [12], this takes $O(n^2 \cdot m \log m)$ time. The average positions of all candidates can be computed in $O(n \cdot m)$ time by iterating once over every vote and adding the position of every candidate to a counter variable for this candidate. Thus, the input for the dynamic programming algorithm can be provided in $O(n^2 \cdot m \log m)$ time.

Concerning the dynamic programming algorithm itself, due to Lemma 2, for $0 \leq i < m$, the size of P_i is upper-bounded by $4d$. Then, for the initialization as well as for the update, the algorithm iterates over m positions, $4d$ candidates, and 2^{4d} candidates subsets. Whereas the initialization in the innermost step (line *04*) can be done in constant time, in every innermost step of the update phase (line *11*) we have to look for a minimum entry and we have to compute a pK-score. To find the minimum, we have to consider all candidates of $P'_i \cup F(i)$. As $P'_i \cup F(i)$ is a subset of P_{i-1}, it can contain at most $4d$ candidates. Further, the required pK-score can be computed in $O(n \cdot m \log m)$ time. Thus, for the dynamic programming we arrive at the running time of $O(m \cdot 4d \cdot 2^{4d} \cdot (4d + n \cdot m \log m)) = O(16^d \cdot (16d^2 \cdot m + 4d \cdot m^2 \log m \cdot n))$.

[6]It can still happen that a candidate takes a position outside of the required range around its average position. Since such an entry cannot lead to an optimal solution according to Lemma 1, this does not affect the correctness of the algorithm. To improve the running time it would be convenient to "cut away" such possibilities. We defer considerations in this direction to an extended version of this paper.

Concerning the size of the dynamic programming table, there are m positions and at most $4d$ candidates that can assume a position. The number of considered subsets is bounded from above by 2^{4d}. Hence, the size of T is $O(16^d \cdot 4d \cdot m)$. $\square$

Finally, let us discuss the differences between the dynamic programming algorithm we used for the “maximum range of candidate positions” in [3] and the algorithm presented in this work. In our previous work [3], the dynamic programming table stored all possible orders of the candidates of a given subset of candidates. In this work, we eliminate the need to store all orders by using the decomposition of the Kemeny score into partial Kemeny scores. This allows us to restrict the considerations for a position to a candidate and its order relative to all other candidates. We believe that our new approach can also be used to improve the running time of the algorithm of [3].

4 Conclusion

We significantly improved the running time for a natural parameterization (maximum KT-distance between two input votes) for the KEMENY SCORE problem. There have been some experimental studies [5, 4] that hinted that the Kemeny problem is easier when the votes are close to a consensus (and thus tend to have a small average distance). Our results for the average distance parameterization can be regarded as a theoretical explanation for this behavior.

As further challenges for future work, we envisage the following:

- Extend our findings to the KEMENY SCORE problem with input votes that may have ties or that may be incomplete (also see [3]).
- Extend our results to improve the running time for the parameterization by position range—we conjecture that this is not hard to do.
- Improve the running time as well as the memory consumption (which is exponential in the parameter)—we believe that significant improvements are still possible.
- Implement the algorithms for the parameters “number of candidates”, “range of position of candidates” [3], and “average KT-distance” (including some maybe heuristic improvements of the running times).
- Investigate typical values of the average KT-distance, either under some distributional assumption or for real-world data.

Acknowledgements. We are grateful to an anonymous referee of *COMSOC 2008* for constructive feedback. This work was supported by the DFG, research project DARE, GU 1023/1, Emmy Noether research group PIAF, NI 369/4, and project PALG, NI 369/8 (Nadja Betzler and Jiong Guo). Michael R. Fellows and Frances A. Rosamond were supported by the Australian Research Council. This work was done while Michael Fellows stayed in Jena as a recipient of the Humboldt Research Award of the Alexander von Humboldt foundation, Bonn, Germany.

Nadja Betzler, Jiong Guo, and Rolf Niedermeier,
Institut für Informatik,
Friedrich-Schiller-Universität Jena,
Ernst-Abbe-Platz 2,
D-07743 Jena, Germany.
Email: `(betzler,guo,niedermr)@minet.uni-jena.de`

Michael R. Fellows and Frances A. Rosamond,
PC Research Unit, Office of DVC (Research),
University of Newcastle,
Callaghan, NSW 2308, Australia.
Email: `(michael.fellows,frances.rosamond)@newcastle.edu.au`

References

[1] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. In *Proc. 37th STOC*, pages 684–693. ACM, 2005.

[2] J. Bartholdi III, C. A. Tovey, and M. A. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6:157–165, 1989.

[3] N. Betzler, M. R. Fellows, J. Guo, R. Niedermeier, and F. A. Rosamond. Fixed-parameter algorithms for Kemeny scores. In *Proc. of 4th AAIM*, volume 5034 of *LNCS*, pages 60–71. Springer, 2008. Long version submitted to *Theoretical Computer Science.*

[4] V. Conitzer, A. Davenport, and J. Kalagnanam. Improved bounds for computing Kemeny rankings. In *Proc. 21st AAAI*, pages 620–626, 2006.

[5] A. Davenport and J. Kalagnanam. A computational study of the Kemeny rule for preference aggregation. In *Proc. 19th AAAI*, pages 697–702, 2004.

[6] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, 1999.

[7] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the Web. In *Proc. of 10th WWW*, pages 613–622, 2001.

[8] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation revisited, 2001. Manuscript.

[9] J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer, 2006.

[10] E. Hemaspaandra, H. Spakowski, and J. Vogel. The complexity of Kemeny elections. *Theoretical Computer Science*, 349:382–391, 2005.

[11] C. Kenyon-Mathieu and W. Schudy. How to rank with few errors. In *Proc. 39th STOC*, pages 95–103. ACM, 2007.

[12] J. Kleinberg and E. Tardos. *Algorithm Design.* Addison Wesley, 2006.

[13] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford University Press, 2006.

[14] V. Raman and S. Saurabh. Improved fixed parameter tractable algorithms for two "edge" problems: MAXCUT and MAXDAG. *Information Processing Letters*, 104(2):65–72, 2007.

[15] A. van Zuylen and D. P. Williamson. Deterministic algorithms for rank aggregation and other ranking and clustering problems. In *Proc. 5th WAOA*, volume 4927 of *LNCS*, pages 260–273. Springer, 2007.