Fast Local Search Algorithm for Weighted Feedback Arc Set in Tournaments

Fedor V. Fomin and **Daniel Lokshtanov**
Department of Informatics,
University of Bergen, Norway.
{fedor.fomin|daniello}@ii.uib.no

Venkatesh Raman and **Saket Saurabh**
The Institute of Mathematical Sciences,
Chennai, India.
{vraman|saket}@imsc.res.in

Abstract

We present a fast local search algorithm that finds an improved solution (if there is any) in the k-exchange neighborhood of the given solution to an instance of WEIGHTED FEEDBACK ARC SET IN TOURNAMENTS. More precisely, given an arc weighted tournament T on n vertices and a feedback arc set F (a set of arcs whose deletion from T turns it into a directed acyclic graph), our algorithm decides in time $O(2^{o(k)}n \log n)$ if there is a feedback arc set of smaller weight and that differs from F in at most k arcs. To our knowledge this is the first algorithm searching the k-exchange neighborhood of an NP-complete problem that runs in (parameterized) subexponential time. Using this local search algorithm for WEIGHTED FEEDBACK ARC SET IN TOURNAMENTS, we obtain subexponential time algorithms for a local search variant of KEMENY RANKING – a problem in social choice theory and of ONE-SIDED CROSS MINIMIZATION – a problem in graph drawing.

I – Introduction

Local search algorithms, also known as neighborhood search algorithms, constitute a large class of improvement algorithms. To perform local search, a problem specific neighborhood distance function is defined on the solution space and a better solution is searched for in the neighborhood of the current solution. In particular, many local search algorithms are based on searching in the k-exchange neighborhood. This is the set of solutions that can be obtained from the current solution by exchanging at most k elements. A classical example is the TRAVELING SALESPERSON problem, where the neighborhood of a tour T can be defined as the set of all tours that differ from T in at most k edges, this is called the k-exchange neighborhood (Ahuja et al. 2002; Lin and Kernighan 1973; Papadimitriou and Steiglitz 1977).

As a rule of thumb, the chances of finding a better solution grow with k. However, for inputs of size n, a naïve brute-force search of the k-exchange neighborhood requires $n^{O(k)}$ time, which is not practical even for very small values of k. It has been generally assumed until very recently, (perhaps because of the typical algorithmic structure of local search algorithms: "Look at *all* solutions in the neighborhood of the current solution ...") that finding an improved solution in a k-exchange neighborhood, necessarily requires brute-force search of the neighborhood; therefore, verifying k-optimality requires $\Omega(n^k)$ time (see, e.g. (Aarts and Lenstra 1997) p. 339 or (Kleinberg and Tardos 2005) p. 680).

Surprisingly, in some cases the k-exchange neighborhood can be searched significantly faster. In particular, on sparse structures like planar graphs or graphs not containing some fixed graph as a minor, finding a k-local improvement for many natural problems including VERTEX COVER, ODD CYCLE TRANSVERSAL, DOMINATING SET, and r-CENTRE can be done in time $f(k) \cdot n^c$, where c is a fixed constant independent of k (Fellows et al. 2009). Similar results were also obtained for sparse instances of SATISFIABILITY (Szeider 2009). Algorithms that have time complexity of the form $f(k) \cdot n^c$ are said to be fixed parameter tractable algorithms. For an introduction to the field Parameterized Complexity, and more recent developments, see the books (Downey and Fellows 1999; Flum and Grohe 2006; Niedermeier 2006). However, all the known local search algorithms strongly exploit the sparsity of the structure, in particular, the fact that in these structures a ball of radius k induces a subgraph (or substructure) of treewidth $f(k)$, for some function k. So the techniques from (Fellows et al. 2009; Szeider 2009) do not seem to work for dense structures.

In this work, we use completely different techniques to show that fast search of the k-exchange neighborhood is also possible for dense structures. A *tournament* is a directed graph where every pair of vertices is connected by exactly one arc, and a *feedback arc set* is a set of arcs whose removal makes the graph acyclic (without any directed cycle). Feedback arc sets in tournaments are well studied, from the combinatorial (Erdös and J.W.Moon 1965; Fernandez de la Vega 1983; Jung 1970; K.B.Reid 1969; K.D.Reid and E.T.Parker 1970; Seshu and Reed 1961; Spencer 1971; 1980; Younger 1963), statistical (Slater 1961), and algorithmic (Ailon, Charikar, and Newman 2005; Alon 2006; Kenyon-Mathieu and Schudy 2007; van Zuylen 2005; van Zuylen et al. 2007) points of view. The problem has several applications - in psychology it occurs in relation to *ranking by paired comparisons*: here we wish to rank items by an objective, but we don't have access to the objective

function, but only to pairwise comparisons of the objects in question. An example for this setting is measuring people's preferences for food. The weighted generalization of the problem, WEIGHTED FEEDBACK ARC SET IN TOURNAMENTS is applied in *rank aggregation*: Here we are given several rankings of a set of objects, and we wish to produce a single ranking that, on average, is as consistent as possible with the given ones, according to some chosen measure of consistency. This problem has been studied in the context of voting (Borda 1781; Condorcet 1785), machine learning (Cohen, Schapire, and Singer 1997), and search engine ranking (Dwork et al. 2001a; 2001b). A natural consistency measure for rank aggregation is the number of pairs that occur in different order in the two rankings. This leads to *Kemeney-Young rank aggregation* (Kemeny 1959; Kemeny and Snell 1962), a special case of WEIGHTED FEEDBACK ARC SET IN TOURNAMENTS. The problem of finding a feedback arc set of minimum size in an unweighted tournament is NP-hard (Alon 2006). However, even the weighted version of the problem admits a polynomial time approximation scheme (Kenyon-Mathieu and Schudy 2007) and it was shown to be fixed parameter tractable (Raman and Saurabh 2006). In this paper we consider a local search variant of the following problem:

k-WEIGHTED FEEDBACK ARC SET IN TOURNAMENTS (k-FAST)
Instance: A tournament $T = (V, A)$, a weight function $w : A \to \mathbb{N}$, where $\mathbb{N}$ is a set of positive integers and an integer k.
Question: Is there an arc set $S \subseteq A$ such that $\sum_{e \in S} w(e) \leq k$ and $T \setminus S$ is acyclic?

To define the local search variant of the problem we need some definitions. We use $\mathscr{S}$ to denote the set of feasible solutions to a problem P. For edge/arc subset (or vertex subset) problems, $\mathscr{S}$ is a collection of subsets of edges/arcs and a natural neighborhood function is obtained by exchanging k elements of the current solution. The neighborhood function in which we are interested is called *k-exchange neighborhoods* (k-**ExN**). We elaborate this further. Let $w : A \to \mathbb{N}$ be a weight function. Then the cost function $w : \mathscr{S} \to \mathbb{N}$ is defined as $\sum_{v \in s} w(v)$ for all $s \in \mathscr{S}$. We say that s' is neighbor of s with respect to k-**ExN** if $|s \setminus s'| \leq k$ and $|s' \setminus s| \leq k$. Let $\mathcal{N}_k(s)$ denote the set of neighbors of s with respect to k-**ExN**.

k-LOCAL SEARCH WEIGHTED FEEDBACK ARC SET IN TOURNAMENTS (k-LSFAST)
Instance: A tournament $T = (V, A)$, a feedback arc set $F \subseteq A$ of T, a weight function $w : A \to \mathbb{N}$ and an integer k.
Question: Does there exist a $F' \in \mathcal{N}_k(F)$ such that $w(F') < w(F)$?

For a pair of sets A and B, let $S_{out}(A, B) = A \setminus B$ and $S_{in}(A, B) = B \setminus A$. When A and B are clear from the context, we will only use S_{in} and S_{out}. Then given a feedback arc set F we are looking for $F' \in \mathcal{N}_k(F)$ such that

- $|S_{in}(F, F')| \leq k$ and $|S_{out}(F, F')| \leq k$;
- $w(F') < w(F)$ where $F' = F \setminus S_{out} \cup S_{in}$; and
- $T \setminus F'$ is acyclic.

Alon et al. (Alon, Lokshtanov, and Saurabh 2009) introduced the method of *chromatic coding*, a variant of classical color-coding (Alon, Yuster, and Zwick 1995) and obtained the first parameterized subexponential time algorithm for k-FAST running in time $2^{O(\sqrt{k} \log k)} n^{O(1)}$. In fact this was the first non-trivial parameterized subexponential time algorithm for a problem on dense graphs. Recently Feige (Feige 2009) has obtained a faster algorithm for k-FAST running in time $2^{O(\sqrt{k})} n^{O(1)}$. In this paper we use the method of chromatic coding and show that k-LSFAST is fixed parameter tractable when parameterized by k, the number of arcs we are allowed to exchange from the current solution. In particular we obtain the following results:

- there is a randomized algorithm for k-LSFAST running in time $2^{O(\sqrt{k} \log k)} n$; and
- there is a deterministic algorithm for k-LSFAST that runs in time $2^{O(\sqrt{k} \log k)} n \log n$.

To our knowledge this is the first algorithm searching the k-exchange neighborhood of an NP-complete problem that runs in (parameterized) subexponential time. Using the local search algorithm for k-LSFAST, we obtain subexponential time algorithms for a local search variant of KEMENY RANKING – a problem in social choice theory (Conitzer, Davenport, and Kalagnanam 2006; Davenport and Kalagnanam 2004; Ephrati and Rosenschein 1993) and of ONE-SIDED CROSS MINIMIZATION – a problem in graph drawing (Eades and Wormald 1994; Dujmović et al. 2008; Dujmović, Fernau, and Kaufmann 2008).

Our algorithm is based on the ideas of the k-FAST algorithm of Alon et al (Alon, Lokshtanov, and Saurabh 2009), but differs significantly from it in a crucial step. The algorithm presented in (Alon, Lokshtanov, and Saurabh 2009) starts by preprocessing the instance and obtains an equivalent instance with at most $O(k^2)$ vertices in polynomial time. That is, given a tournament T and a positive integer k, in polynomial time the preprocessing algorithm either concludes that T does not have a feedback arc set of weight at most k or finds a new tournament T' with $O(k^2)$ vertices and $k' \leq k$ such that the original tournament T has a feedback arc set of weight at most k, if and only if T' has a feedback arc set of weight at most k'. This preprocessing allows them to assume that the instance where they actually apply the subexponential time algorithm is of size $O(k^2)$ only, which is integral to their time analysis. For k-LSFAST, and in fact for most local search problems, one can show using standard techniques (Bodlaender et al. 2008; Fortnow and Santhanam 2008) that such a pre-processing step would imply that coNP is a subset of NP/poly, an event which is deemed unlikely in complexity theory. Since we can not use polynomial time pre-processing, we resort to a more sophisticated dynamic programming strategy instead.

II – Preliminaries

For an arc weighted tournament we define the weight function $w^* : V \times V \to \mathbb{R}$ such that $w^*(u, v) = w(uv)$ if $uv \in A$

1. Let $t = 4\sqrt{k}$. Color the vertices of T uniformly at random with colors from $\{1, \ldots, t\}$.
2. Let A_c be the set of arcs whose endpoints have different colors. Find a minimum weighted feedback arc set $F' \in \mathcal{N}_k(F)$ such that S_{in} and S_{out} are contained in A_c.

Figure 1: Outline of the algorithm for k-LSFAST.

and 0 otherwise. Given a directed graph $D = (V, A)$ and a set F of arcs in A, define $D\{F\}$ to be the directed graph obtained from D by reversing all arcs of F. In our arguments we will need the following characterization of minimal feedback arc sets in directed graphs.

Proposition 1 [(Gallai 1968)] *Let $D = (V, A)$ be a directed graph and F be a subset of A. Then F is a minimal feedback arc set of D if and only if F is a minimal set of arcs such that $D\{F\}$ is a directed acyclic graph.*

Given a minimal feedback arc set F of a tournament T, the ordering σ corresponding to F is the unique topological ordering of $T\{F\}$. Conversely, given an ordering σ of the vertices of T, the feedback arc set F corresponding to σ is the set of arcs whose endpoint appears before their startpoint in σ. The cost of an arc set F is $\sum_{e \in F} w(e)$ and the cost of a vertex ordering σ is the cost of the feedback arc set corresponding to σ.

For a pair of integer vectors $\hat{p} = [p_1, \ldots, p_t]$, $\hat{q} = [q_1, \ldots, q_t]$ we say that $\hat{p} \leq \hat{q}$ if $p_i \leq q_i$ for all i. The t-sized vector $\hat{e}$ is $[1, 1, \ldots, 1]$, $\hat{0}$ is $[0, 0, \ldots, 0]$ and $\hat{e}_i$ is the t-sized vector with all entries 0 except for the i'th which is 1. For any positive integer m let $[m] = \{1, 2, \ldots, m\}$. For a given subset $V' \subseteq V$ of a digraph $D = (V, A)$ by $D[V']$ we denote the induced subgraph on V'. For simplicity whenever we say $\log n$ or $\sqrt{n}$ we mean $\lceil \log n \rceil$ and $\lceil \sqrt{n} \rceil$.

III - Main Algorithm

In this section we give our algorithm, argue about its correctness and analyze its time complexity. We start with the description of our algorithm.

Algorithm Description

Our algorithm consists of two steps. In the first step we randomly color the vertices of our graph with $t = 4\sqrt{k}$ colors, and define the arc set A_c to be the set of arcs whose endpoints have different colors. In the next step of the algorithm we find a minimum weighted feedback arc set $F' \in \mathcal{N}_k(F)$ such that S_{in} and S_{out} are contained in A_c. A summary of the algorithm is given in Figure 1.

To analyze the first step the algorithm we use the following lemma of Alon et al. (Alon, Lokshtanov, and Saurabh 2009).

Lemma 1 [(Alon, Lokshtanov, and Saurabh 2009)] *If a graph G with q edges is colored randomly with $\sqrt{8q}$ colors then the probability that G is properly colored is at least $(2e)^{-\sqrt{q/8}}$.*

Lemma 1 implies that if we randomly color the vertices of T with $t = \sqrt{8 \cdot 2k} = 4\sqrt{k}$ colors then the probability that the subgraph $T' = (V, S_{in} \cup S_{out})$ is properly colored, or equivalently that S_{in} and S_{out} are subsets of A_c is at least $(2e)^{-\sqrt{2k/8}} = 2^{-c\sqrt{k}}$ for some fixed constant c, as T' has at most $2k$ arcs. Next we show how to implement the second step of our algorithm using dynamic programming over the colored instance.

Solving a Colored Instance

The second part of our algorithm takes a t-colored tournament $T = (V_1 \cup V_2 \cup \ldots \cup V_t, A)$ and a feedback arc set F of T as input, and produces a weighted feedback arc set $F' \in \mathcal{N}_k(F)$ such that S_{in} and S_{out} are contained in A_c and $w(F')$ is minimized. Now, $S_{in} \cup S_{out} \subseteq A_c$ implies that $F \cap (A \setminus A_c) = F' \cap (A \setminus A_c)$ and hence $T\{F'\}[V_i] = T\{F\}[V_i]$ must be an acyclic tournament for every i. Let $\sigma_{old} = v_1 v_2 \ldots v_n$ be the ordering of V corresponding to the given feedback arc set F of T and let $\sigma_{new} = u_1 u_2 \ldots u_n$ be the ordering of V corresponding to the feedback arc set F' of T. Let $n_i = |V_i|$ for every i and let $\hat{n}$ be the vector $[n_1, n_2 \ldots, n_t]$. For every color class V_i of T, let $v_1^i v_2^i \ldots v_{n_i}^i$ be the order in which the vertices of V_i appear according to σ_{old}. Observe that since $T\{F'\}[V_i] = T\{F\}[V_i]$ and every acyclic tournament has a unique topological ordering we have that the vertices of every color class V_i of T appear in the same order in σ_{old} and σ_{new}, that is, $v_1^i v_2^i \ldots v_{n_i}^i$. We exploit this fact to give a dynamic programming algorithm for the problem.

Lemma 2 *Given a t-colored tournament T and a feedback arc set F, we can find a minimum weight feedback arc set $F' \in \mathcal{N}_k(F)$ such that $S_{in}, S_{out} \subseteq A_c$ in $O(n^2 t (2k+1)^{t+1})$ time and $O((2k+1)^{t+1} n)$ space.*

Proof: For an integer $x \geq 1$ define $S_x^i = \{v_1^i \ldots, v_x^i\}$ and $S_0 = S_0^i = \emptyset$. Given an integer vector $\hat{p}$ of length t in which the i'th entry is between 0 and n_i, let $S(\hat{p}) = S_{p_1}^1 \cup S_{p_2}^2 \ldots \cup S_{p_t}^t$, $T(\hat{p})$ be $T[S(\hat{p})]$ and $A(\hat{p})$ be the arc set of $T(\hat{p})$. For a feedback arc set F of T we set $F(\hat{p}) = F \cap A(\hat{p})$. Observe that for any ordering $\sigma = v_1 v_2 \ldots v_n$ of V corresponding to the feedback arc set F' of T such that $F' \setminus A_c = F \setminus A_c$ and for any integer x there exists a $\hat{p}$ such that $\{v_1 \ldots, v_x\} = S(\hat{p})$.

Fixing the tournament T and a feedback arc set F of T we define LSFAS$(\hat{p}, k_1, k_2)$, to be the weight of a minimum weight feedback arc set F' of $T(\hat{p})$ such that (a) $|F(\hat{p}) \setminus F'| \leq k_1$, (b) $|F' \setminus F(\hat{p})| \leq k_2$ and (c) $F' \setminus A_c = F(\hat{p}) \setminus A_c$. If $k_1 < 0$ or $k_2 < 0$ then LSFAS$(\hat{p}, k_1, k_2)$ is set to ∞. We call a feedback arc set F' satisfying these conditions a *(k_1, k_2)-constrained feedback arc set of $T(\hat{p})$*. We proceed to prove that the following recurrence holds for

$\textsc{LsFas}(\hat{p}, k_1, k_2)$.

$$\textsc{LsFas}(\hat{p}, k_1, k_2) = \min_i \Big\{ \textsc{LsFas}\Big(\hat{p} - \hat{e}_i, k_1', k_2'\Big) + \sum_{u \in S(\hat{p})} w^*(v^i_{p_i}, u) \Big\} \quad (1)$$

Here the minimum is taken over i, such that $p_i > 0$, $k_1' = k_1 - |\{uv^i_{p_i} \in F(\hat{p})\}|$ and $k_2' = k_2 - |\{v^i_{p_i} u \in A(\hat{p}) \setminus F(\hat{p})\}|$. First we prove that the left hand side is at most the right hand side. Let i be the integer that minimizes the right hand side. Taking the ordering of $S(\hat{p} - \hat{e}_i)$ corresponding to an optimal (k_1', k_2')-constrained feedback arc set of $T(\hat{p} - \hat{e}_i)$ and appending it with $v^i_{\hat{p}_i}$ gives an ordering σ of $T(\hat{p})$ corresponding to a (k_1, k_2)-constrained feedback arc set of $T(\hat{p})$ with cost at most $\textsc{LsFas}(\hat{p} - \hat{e}_i, k_1', k_2') + \sum_{u \in S(\hat{p})} w^*(v^i_{p_i}, u)$.

To prove that the right hand side is at most the left hand side, let σ be an ordering of $T(\hat{p})$ corresponding to a (k_1, k_2)-constrained feedback arc set F' of $T(\hat{p})$ and let $v \in V_i$ be the last vertex of this ordering. Then $v = v^i_{p_i}$ and σ restricted to $V(T(\hat{p} - \hat{e}_i))$ is an ordering of $T(\hat{p} - \hat{e}_i)$ corresponding to the feedback arc set $F' \cap A(\hat{p} - \hat{e}_i)$. Now, $|F(\hat{p}) \setminus F'| \leq k_1$ and v is incident to $|\{uv^i_{p_i} \in F(\hat{p})\}|$ edges of $F(\hat{p}) \setminus F'$. Similarly $|F' \setminus F(\hat{p})| \leq k_2$ and v is incident to $|\{v^i_{p_i} u \in A(\hat{p}) \setminus F(\hat{p})\}|$ edges of $F' \setminus F(\hat{p})$. Thus $F' \setminus \{v\}$ is a (k_1', k_2')-constrained feedback arc set of $T(\hat{p} - \hat{e}_i)$. The total weight of the edges with startpoint in v and endpoint in $V(T(\hat{p} - \hat{e}_i))$ is exactly $\sum_{u \in V(T(\hat{p}))} w^*(v^i_{\hat{p}_i}, u)$. Thus the cost of σ is at least the value of the right hand side of the inequality, completing the proof.

Recurrence 1 naturally leads to a memoization based algorithm for the problem. We compute $\textsc{LsFas}(\hat{p}, k, k)$ by applying Recurrence 1, returning ∞ whenever $k_1 < 0$ or $k_2 < 0$ and storing the output of every recursive call except the ones with $k_1 < 0$ or $k_2 < 0$ in a table. If at a later stage of the algorithm a recursive call is made with the same parameters we return the corresponding table entry instead of recomputing the function. To bound the running time it is sufficient to bound the size of the memoization table and the running time required to compute each entry from the other ones.

The number of possible values for $\hat{p}$ is less than n^t and hence there are at most $n^t k^2$ table entries. For each entry it takes $O(nt)$ time to compute it giving a $O(t \cdot n^{t+1} k^2)$ time bound for the algorithm. In order to obtain a better running time bound, we show that only a small subset of the possible values of $\hat{p}$ will ever be visited by a recursive call of the algorithm.

For a vector $\hat{p}$ define $\delta^+(\hat{p}) = \{uv \in A \mid u \in S(\hat{p}), v \notin S(\hat{p})\}$ and $\delta^-(\hat{p}) = \{uv \in A \mid u \notin S(\hat{p}), v \in S(\hat{p})\}$. The crucial observation is that if $\textsc{LsFas}(\hat{p}, k_1, k_2)$ is visited by a recursive call, then $|F \cap \delta^+(\hat{p})| + k_1 \leq k$ and $|\delta^-(\hat{p}) \setminus F| + k_2 \leq k$. To see that the inequalities hold, let F' be a (k_1, k_2)-constrained feedback arc set of $T(\hat{p})$ that corresponds to an ordering that takes all the vertices of $S(\hat{p})$ before the vertices not in $S(\hat{p})$. Then $\delta^+(\hat{p}) \cap F \subseteq F \setminus F'$ and $\delta^-(\hat{p}) \setminus F \subseteq F' \setminus F$. Thus, to bound the size of the memoization table it is sufficient to upper bound the number of *legal* vectors, that is $\hat{p}$ that satisfy $|\delta^+(\hat{p}) \cap F| + |\delta^-(\hat{p}) \setminus F| \leq 2k$.

Let $\sigma_{old} = v_1 \dots v_n$ be the ordering of V corresponding to F, $\hat{p}$ be a legal vector and $S = S(\hat{p})$. Now, let x be the largest integer such that $v_x \in S$. We prove that for every $y < x - 2k$, v_y must be in S as well. Suppose not, then $v_x v_y \in (\delta^+(\hat{p}) \cap F) \cup (\delta^-(\hat{p}) \setminus F)$. Consider a vertex v_z with $y < z < x$. If $v_z \in S$ then $v_z v_y \in (\delta^+(\hat{p}) \cap F) \cup (\delta^-(\hat{p}) \setminus F)$ and if $v_z \notin S$ then $v_x v_z \in (\delta^+(\hat{p}) \cap F) \cup (\delta^-(\hat{p}) \setminus F)$. Since there are at least $2k$ choices for z this implies that $|(\delta^+(\hat{p}) \cap F) \cup (\delta^-(\hat{p}) \setminus F)| > 2k$, contradicting that $\hat{p}$ is legal.

To uniquely determine $\hat{p}$ it is enough to uniquely determine S. If x is given then $v_x \in S$, for all $y > x$, $v_y \notin S$ and the argument in the above paragraph implies $\{v_1, \dots, v_{x-2k-1}\} \subset S$. Thus, to specify S it is sufficient to specify x and $S \cap C$ where $C = \{v_{x-2k}, \dots v_{x-1}\}$. Let $C_j = C \cap V_j = \{v^j_{a_j}, v^j_{a_j+1}, \dots v^j_{b_j}\}$. Then $S \cap C = (S \cap C_1) \cup (S \cap C_2) \dots \cup (S \cap C_t)$ and for each j there are at most $2k+1$ possible choices of $S \cap C_j$ since $S \cap C_j$ is either $\emptyset$ or $\{v^j_{a_j}, v^j_{a_j+1}, \dots v^j_{b'_j}\}$ for some $b'_j \leq b_j$. Thus there are $n(2k+1)^{t-1}$ choices for S and hence at most $n(2k+1)^{t-1}k^2 \leq n(2k+1)^{t+1}$ entries in the memoization table. Since each recursive step takes $O(nt)$ time the total running time bound becomes $n(2k+1)^{t+1} \cdot nt = O((2k+1)^{t+1} n^2 t)$. □

In fact, the algorithm provided in Lemma 2 can be made to run slightly faster by pre-computing the value of $\sum_{u \in S(\hat{p})} w^*(v^i_{\hat{p}_i}, u))$, k_1' and k_2' for every $\hat{p}$ and i using dynamic programming, and storing it in a table. This would let us reduce the time to compute a table entry using Recurrence 1 from $O(nt)$ to $O(t)$ yielding an algorithm that runs in time $O(n \cdot t \cdot (2k+1)^{t+2})$ time. This gives us the following theorem.

Theorem 1 *k-LSFAST can be solved in expected time $2^{O(\sqrt{k} \log k)} n$ and $2^{O(\sqrt{k} \log k)} n$ space.*

Proof: Our algorithm proceeds by repeating the steps described in Figure 1. We use Lemma 2 to find a minimum weight arc set $F' \in \mathcal{N}_k(F)$ such that $S_{in}, S_{out} \subseteq A_c$ and check if $w(F') < w(F)$. If $w(F') < w(F)$ then we return true else we repeat the steps described in Figure 1. The correctness of the algorithm follows from Lemma 2. Combining Lemmata 1 and 2 yield an expected running time of $O((2e)^{\sqrt{k/4}}) \cdot O(4\sqrt{k} \cdot (2k+1)^{1+4\sqrt{k}} n) \leq 2^{O(\sqrt{k} \log k)} n$ for finding a $F' \in \mathcal{N}_k(F)$ such that $w(F') < w(F)$ if one exists. The space required by the algorithm is $n \cdot (2k+1)^{4\sqrt{k}+1} \leq 2^{O(\sqrt{k} \log k)} n$. □

The algorithm described in Figure 1 can be derandomized using *universal coloring families* introduced in (Alon, Lokshtanov, and Saurabh 2009). For integers m, k and r, a family $\mathcal{F}$ of functions from $[m]$ to $[r]$ is called a universal (m, k, r)-coloring family if for any graph G on the set of vertices $[m]$ with at most k edges, there exists an $f \in \mathcal{F}$ which is a proper vertex coloring of G. An explicit construction of a $(n, 2k, O(\sqrt{k}))$-coloring family can replace the

randomized coloring step in the algorithm for k-LSFAST. Towards this we utilize the following result.

Proposition 2 [(Alon, Lokshtanov, and Saurabh 2009)] *For any $n > 10k^2$ there exists an explicit universal $(n, k, O(\sqrt{k}))$-coloring family $\mathcal{F}$ of size $|\mathcal{F}| \leq 2^{O(\sqrt{k}\log k)} \log n$.*

Finally, combining the algorithm from Theorem 1 with the universal coloring families given by Proposition 2 yields a deterministic subexponential time algorithm for k-LSFAST.

Theorem 2 *k-LSFAST can be solved in time $2^{O(\sqrt{k}\log k)} n \log n$.*

Other Applications

Our subexponential algorithm for k-LSFAST is useful in obtaining subexponential time algorithm for local search variants of various other problems arising in different domains. The first one is related to KEMENY RANKING that arises in social choice theory. Preference lists are typical examples of elements in social science surveys and voting systems. In several such cases we wish to combine various lists into one which reflects the opinion of the surveyed group as much as possible. The Kemeny aggregation problem was introduced by Kemeny (Kemeny 1959; Kemeny and Snell 1962) to abstract out problem of combining preference lists into one. Given a set of m permutations (called *votes*) over a set of n alternatives (called *candidates*), the k-KEMENY-OPTIMAL AGGREGATION (k-KOA) problem asks for a permutation of candidates, called an *optimal aggregation*, such that the sum of the τ-distance from the votes is at most k. The τ-distances between two permutation π_1 and π_2 is the number of pairs of candidates that are ordered differently in the two permutations and is denoted by $\tau(\pi_1, \pi_2)$. Let $L(\pi)$ denote the set of pairs of candidates (a, b) such that there exist a vote π_j such that (a, b) is ordered differently in π and π_j.

The k-local search variant of this problem, denoted by k-LSKOA, is that given an input consisting of m votes (say V), that is m permutations $\{\pi_1, \ldots, \pi_m\}$, over n candidates (say C) and a permutation π, to check whether there is another permutation π^* such that $\sum_{i=1}^{m} \tau(\pi^*, \pi_i) < \sum_{i=1}^{m} \tau(\pi, \pi_i)$, $|L(\pi) \setminus L(\pi^*)| \leq k$ and $|L(\pi^*) \setminus L(\pi)| \leq k$. It is well known that k-KOA (Dwork et al. 2001a; 2001b) can be modelled as solving a weighted feedback arc set problem in *multi-tournaments* (tournaments where between every pair of vertices $\{u, v\}$ there is either $(u, v) \in A$ or $(v, u) \in A$ or both $(u, v), (v, u) \in A$). The modelling essentially does the following: Given an instance (C, V, k) of k-KOA, constructs a multi-tournament T such that (C, V, k) is an YES instance of k-KOA if and only if T has a feedback arc set of weight at most k. Using this transformation one can give a reduction from k-LSKOA to k-LSFAST. Lemma 2 can also be proved for multi-tournaments and hence Theorems 1 and 2 can be generalized for multi-tournaments. This together with the equivalence between k-LSKOA and k-LSFAST (in multi-tournaments) allow us to conclude the following theorem.

Theorem 3 *k-LSFKOA can be solved in time $2^{O(\sqrt{k}\log k)} n \log n$.*

Our other application stems from graph-drawing and the problem itself is called k-ONE SIDED CROSS MINIMIZATION (k-OSCM). The problem consists of a bipartite graph $G = (V_1, V_2, E)$, a permutation π of V_1, and a positive integer k and the objective is to check whether there is a permutation π_m of V_2 such that, when the vertices of V_1 are placed on a line (also called a *layer*) in the order induced by π and the vertices of V_2 are placed on a second layer (parallel to the first one) in the order induced by π_m, then drawing straight-line segment for each edge in E will introduce no more than k (pairwise) edge crossings. It has been brought to our attention by a colleague working in graph drawing that even this problem can be modelled as weighted feedback arc set in multi-tournaments, almost similar to the modelling done for k-KOA. This modelling allows us to prove that the local search variant obtained by viewing the problem as weighted feedback arc set in multi-tournaments can be solved in parameterized subexponential time.

IV – Conclusion

In this paper we obtained a parameterized subexponential time algorithm for a local search variant of the k-WEIGHTED FEEDBACK ARC SET IN TOURNAMENTS problem. This algorithm was also useful in designing subexponential time algorithm for local search variants of k-KOA and k-ONE SIDED CROSS MINIMIZATION. It is natural to ask whether the local search variant of weighted feedback arc set is fixed parameter tractable for all digraphs. Unfortunately, it can easily be shown that it is not the case unless some unlikely collapse happens in parameterized complexity theory. A natural open question is whether we can solve k-LSFAST in time $2^{O(\sqrt{k})} n^{O(1)}$ like k-FAST. Other questions which remain open are whether we can use chromatic-coding or color-coding to solve local search variant of other problems. The problems which might be feasible using this approach are the local search variants of complete version of BETWEENNESS and MINIMUM QUARTET INCONSISTENCY.

References

Aarts, E. H. L., and Lenstra, J. K. 1997. *Local Search in Combinatorial Optimization*. Princeton University Press.

Ahuja, R. K.; Ergun, Ö.; Orlin, J. B.; and Punnen, A. P. 2002. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics* 123(1-3):75–102.

Ailon, N.; Charikar, M.; and Newman, A. 2005. Aggregating inconsistent information: ranking and clustering. In *ACM Symposium on Theory of Computing (STOC)*, 684–693.

Alon, N.; Lokshtanov, D.; and Saurabh, S. 2009. Fast FAST. In *ICALP (1)*, volume 5555 of *Lecture Notes in Computer Science*, 49–58.

Alon, N.; Yuster, R.; and Zwick, U. 1995. Color-coding. *J. Assoc. Comput. Mach.* 42(4):844–856.

Alon, N. 2006. Ranking tournaments. *SIAM J. Discrete Math.* 20:137–142.

Bodlaender, H. L.; Downey, R. G.; Fellows, M. R.; and Hermelin, D. 2008. On problems without polynomial kernels. In *Proc. 35th ICALP*, volume 5125 of *Lecture Notes in Computer Science*, 563–574. Springer.

Borda, J. 1781. Mémoire sur les élections au scrutin. *Histoire de l'Académie Royale des Sciences*.

Cohen, W. W.; Schapire, R. E.; and Singer, Y. 1997. Learning to order things. In *Advances in neural information processing systems (NIPS)*, 451–457.

Condorcet, M. 1785. Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix.

Conitzer, V.; Davenport, A.; and Kalagnanam, J. 2006. Improved bounds for computing kemeny rankings. In *AAAI'06*, 620–626. AAAI Press.

Davenport, A., and Kalagnanam, J. 2004. A computational study of the kemeny rule for preference aggregation. In *AAAI'04*. AAAI.

Downey, R. G., and Fellows, M. R. 1999. *Parameterized complexity*. New York: Springer-Verlag.

Dujmović, V.; Fellows, M. R.; Kitching, M.; Liotta, G.; McCartin, C.; Nishimura, N.; Ragde, P.; Rosamond, F. A.; Whitesides, S.; and Wood, D. R. 2008. On the parameterized complexity of layered graph drawing. *Algorithmica* 52(2):267–292.

Dujmović, V.; Fernau, H.; and Kaufmann, M. 2008. Fixed parameter algorithms for one-sided crossing minimization revisited. *J. Disc. Algorithms* 6:313–323.

Dwork, C.; Kumar, R.; Naor, M.; ; and Sivakumar, D. 2001a. Rank aggregation revisited.

Dwork, C.; Kumar, R.; Naor, M.; and Sivakumar, D. 2001b. Rank aggregation methods for the web. In *WWW10*.

Eades, P., and Wormald, N. C. 1994. Edge crossings in drawings of bipartite graphs. *Algorithmica* 11:379–403.

Ephrati, E., and Rosenschein, J. S. 1993. Multi-agent planning as a dynamic search for social consensus. In *IJCAI'93*. AAAI.

Erdös, P., and J.W.Moon. 1965. On sets on consistent arcs in tournaments. *Canadian Mathematical Bulletin* 8:269–271.

Feige, U. 2009. Faster fast(feedback arc set in tournaments). *CoRR* abs/0911.5094.

Fellows, M.; Fomin, F. V.; Lokshtanov, D.; Rosamond, F.; Saurabh, S.; and Villanger, Y. 2009. Local search: Is brute-force avoidable? In *IJCAI*, 486–491. AAAI.

Fernandez de la Vega, W. 1983. On the maximal cardinality of a consistent set of arcs in a random tournament. *J. Combinatorial Theory, Ser. B* 35:328–332.

Flum, J., and Grohe, M. 2006. *Parameterized Complexity Theory*. Berlin: Springer-Verlag.

Fortnow, L., and Santhanam, R. 2008. Infeasibility of instance compression and succinct PCPs for NP. In *Proc. 40th STOC*, 133–142. ACM Press.

Gallai, T. 1968. On directed paths and circuits. In *Theory of Graphs (Proc. Colloq., Tihany, 1966)*. New York: Academic Press. 115–118.

Jung, H. 1970. On subgraphs without cycles in tournaments. *Combinatorial Theory and its Applications II* 675–677.

K.B.Reid. 1969. On sets of arcs containing no cycles in tournaments. *Canad. Math Bulletin* 12:261–264.

K.D.Reid, and E.T.Parker. 1970. Disproof of a conjecture of Erdös and Moser on tournaments. *J. Combin. Theory* 9:225–238.

Kemeny, J., and Snell, J. 1962. *Mathematical models in the social sciences*. Blaisdell.

Kemeny, J. 1959. Mathematics without numbers. *Daedalus* 88:571–591.

Kenyon-Mathieu, C., and Schudy, W. 2007. How to rank with few errors. In *STOC'07*, 95–103. ACM.

Kleinberg, J., and Tardos, E. 2005. *Algorithm Design*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Lin, S., and Kernighan, B. W. 1973. An effective heuristic algorithm for traveling-salesman problem. *Operations Research* 21:498–516.

Niedermeier, R. 2006. *Invitation to fixed-parameter algorithms*. Oxford University Press.

Papadimitriou, C. H., and Steiglitz, K. 1977. On the complexity of local search for the traveling salesman problem. *SIAM J. Comput.* 6(1):76–83.

Raman, V., and Saurabh, S. 2006. Parameterized algorithms for feedback set problems and their duals in tournaments. *Theoretical Computer Science* 351(3):446–458.

Seshu, S., and Reed, M. 1961. *Linear Graphs and Electrical Networks*. Addison-Wesley.

Slater, P. 1961. Inconsistencies in a schedule of paired comparisons. *Biometrika* 48:303–312.

Spencer, J. 1971. Optimal ranking of tournaments. *Networks* 1:135–138.

Spencer, J. 1980. Optimal ranking of unrankable tournaments. *Period. Math. Hungar.* 11(2):131–144.

Szeider, S. 2009. The parameterized complexity of k-flip local search for SAT and MAX SAT. In *SAT*, volume 5584 of *Lecture Notes in Computer Science*, 276–283. Springer.

van Zuylen, A.; Hegde, R.; Jain, K.; and Williamson, D. P. 2007. Deterministic pivoting algorithms for constrained ranking and clustering problems. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 405–414.

van Zuylen, A. 2005. Deterministic approximation algorithms for ranking and clusterings. Technical Report 1431, Cornell ORIE.

Younger, D. 1963. Minimum feedback arc sets for a directed graph. *IEEE Trans. Circuit Theory* 10:238–245.