Efficient Algorithms for k-Disjoint Paths Problems on DAGs*

Rudolf Fleischer, Qi Ge, Jian Li, and Hong Zhu**

Shanghai Key Laboratory of Intelligent Information Processing,
Department of Computer Science and Engineering,
Fudan University, Shanghai, China
{rudolf,qge,lijian83,hzhu}@fudan.edu.cn

Abstract. Given an acyclic directed graph and two distinct nodes s and t, we consider the problem of finding k disjoint paths from s to t satisfying some objective. We consider four objectives, `MinMax`, `Balanced`, `MinSum-MinMin`, and `MinSum-MinMax`. We use the algorithm by Perl-Shiloach and labelling and scaling techniques to devise an FPTAS for the first three objectives. For the forth one, we propose a general and efficient polynomial-time algorithm.

1 Introduction

In communication networks, one way of providing reliable communication is to find several disjoint paths, either node disjoint or edge disjoint. The advantage is that, if some links are broken, there are still other routing paths.

Different objectives may be used to measure the quality (or usefulness) of the disjoint paths. For example, we may require that the total weight of the disjoint paths to be minimized, the so-called *MinSum objective*. This problem can be solved in polynomial time by standard network flow methods [1,9]. However, for many other objectives, the problems are hard to solve. Li *et al.* [7] proposed the MinMax objective and showed that the problem is strong NP-complete. Yang *et al.* [11] proposed the MinMin objective and proved that the problem is also strong NP-complete.

For acyclic directed graphs (DAGs), the problem seems to be easier. In this paper, we focus on finding disjoint paths on DAGs. We propose efficient algorithms for four different objectives that are practically motivated, `MinMax` k-`DP`, `Balanced` k-`DP`, `MinSum-MinMax` k-`DP`, and `MinSum-MinMin` k-`DP`.

Let $G = (V, E)$ be a directed graph, s and t two distinct nodes in V, and $\mathcal{F} : E \mapsto \mathbb{N}$ a positive integral weight function on the edges. Li *et al.* [7] had

* This work is supported by National Natural Science Fund (grants #60573025, #60496321, #60373021) and Shanghai Science and Technology Development Fund (grant #03JC14014).

** The order of authors follows the international standard of alphabetic order of the last name. In China, where first-authorship is the only important aspect of a publication, the order of authors should be Qi Ge, Jian Li, Rudolf Fleischer, Hong Zhu.

M.-Y. Kao and X.-Y. Li (Eds.): AAIM 2007, LNCS 4508, pp. 134–143, 2007.

proposed the `MinMax` k-`DP` problem, where we want to find k disjoint paths $P_1, \ldots, P_k$ from s to t such that the cost of the most expensive path is minimized, i.e., $\max_{1 \le i \le k} \mathcal{F}(P_i)$ is minimized, where $\mathcal{F}(P_i) = \sum_{e \in P_i} \mathcal{F}(e)$ is the weight of P_i. They proved that the problem is strong NP-complete, for directed and undirected graphs, and for edge-disjoint and node-disjoint paths. On DAGs, the problem is NP-complete but has a pseudo-polynomial-time algorithm. We will give an FPTAS for this problem on DAGs.

A variant of this problem is the `Balanced` k-`DP` problem where we want to find k disjoint paths $P_1, \ldots, P_k$ from s to t such that the costs of the cheapest and most expensive path are close together, i.e., $\max_{1 \le i \le k} \mathcal{F}(P_i) / \min_{1 \le i \le k} \mathcal{F}(P_i)$ is minimized. We can show by reduction from the Hamiltonian Path problem that this problem is strong NP-complete for directed and undirected graphs, and for edge-disjoint and node-disjoint paths. For DAGs, the problem is NP-complete by reduction from the Partition problem. We will give an FPTAS for this problem on DAGs.

In the `MinSum-MinMax` k-`DP` problem we want to find k disjoint $s - t$ paths $P_1, \ldots, P_k$ such that $\sum_{1 \le i \le k} \mathcal{F}(P_i)$ is minimized. Among all such paths, we want to find those minimizing $\max_{1 \le i \le k} \mathcal{F}(P_i)$. Note that we can show by reduction from the Disjoint Paths problem (see [3]) that this problem is strong NP-complete for directed graphs, and for edge-disjoint and node-disjoint paths. For undirected graphs and DAGs, the problem is NP-complete by reduction from the Partition problem. We will give an FPTAS for this problem on DAGs.

Similarly, in the `MinSum-MinMin` k-`DP` problem we want to minimize $\min_{1 \le i \le k} \mathcal{F}(P_i)$ among all k disjoint paths of minimum total length. This problem was proposed by Yang *et al.* [12]. They showed that, for $k = 2$, the problem is strong NP-complete for directed graphs and has a polynomial-time algorithm for DAGs. The latter algorithm reduces the `MinSum-MinMin 2DP` problem to the `Normalized` α^+-`MinSum 2DP` problem [13] which can be solved in polynomial time. However, the algorithm uses many expensive arithmetic operations like multiplications and divisions, and it is not very intuitive. Moreover, it cannot be generalized to arbitrary constant k. We will propose a more efficient algorithm for arbitrary constant k.

Due to space constraints we state all algorithms for $k = 2$. We shortly sketch how to generalize them to arbitrary constant $k \ge 2$.

2 Preliminaries

2.1 The Perl-Shiloach Algorithm

In this subsection, we introduce the algorithm PSA to find k node-disjoint paths on a DAG by Perl-Shiloach [8], which is a key subroutine in our algorithms.

In the *Disjoint Paths Problem (*`DPP`*)* we are given a directed graph $G = (V, E)$ and k pairs of distinct nodes $(s_1, t_1), \ldots, (s_k, t_k)$. We want to find k node- or edge-disjoint paths $P_1, \ldots, P_k$, where P_i is a path from s_i to t_i, for $1 \le i \le k$. The decision version of this problem, for node-disjoint and edge-disjoint paths, was shown to be NP-complete by Fortune *et al.* [3], even if $k = 2$. For DAGs, Perl and Shiloach gave a polynomial time algorithm, PSA [8].

PSA is actually a reduction from DPP to the *Connectivity problem.* Given a DAG $G = (V, E)$ and k pairs of distinct nodes $(s_1, t_1), \ldots, (s_k, t_k)$, let $v_1, \ldots, v_n$ be a topological order of V, i.e., there are only edges from nodes with lower indices to nodes with higher indices. We construct a graph $G_k = (V_k, E_k)$ as follows:

$$V_k = \{\langle j_1, \ldots, j_k \rangle \mid 1 \leq j_i \leq n \text{ for } 1 \leq i \leq k, j_i \neq j_l \text{ for } 1 \leq i \neq l \leq k\}, \quad (1)$$

$$E_k = \bigcup_{d=1}^{k} \{(\langle j_1, \ldots, j_{d-1}, j_d, j_{d+1}, \ldots, j_k \rangle, \langle j_1, \ldots, j_{d-1}, j'_d, j_{d+1}, \ldots, j_k \rangle) \mid (v_{j_d}, v_{j'_d}) \in E \text{ and } j_d = \min_{1 \leq l \leq k} j_l\}. \quad (2)$$

For simplicity, we will only describe the algorithms for the case of $k = 2$. For $k \geq 2$, see [2]. To find two disjoint paths from $s = v_1$ to $t = v_n$, we add two nodes $\langle s, s \rangle = \langle 1, 1 \rangle$ and $\langle t, t \rangle = \langle n, n \rangle$ to V_2 to obtain graph G'_2:

$$V'_2 = \{\langle i, j \rangle \mid 1 \leq i, j \leq n \text{ and } i \neq j\} \cup \{\langle s, s \rangle, \langle t, t \rangle\}, \quad (3)$$

$$E'_2 = \{(\langle i, j \rangle, \langle i, k \rangle) \mid (v_j, v_k) \in E, j < i\} \cup \{(\langle i, j \rangle, \langle k, j \rangle) \mid (v_i, v_k) \in E, i < j\}. \quad (4)$$

We call the edges in the first set of Eq. (4) *horizontal* edges and the edges in the second set *vertical* edges.

Lemma 1 [8]. *There are two node disjoint paths P_1, P_2 from s to t in G if and only if there is a directed path P from $\langle s, s \rangle$ to $\langle t, t \rangle$ in G'_2, and P_1 (P_2) consists of the horizontal (vertical) edges of P.* □

2.2 Edge Disjoint Versus Node Disjoint Paths

We can transform the acyclic edge disjoint case to the acyclic node disjoint case using the method by Li *et al.* [7]. Given a DAG G and two distinct nodes s, t, add new nodes u, v and edges $(u, s), (t, v)$. Form the directed line graph (see [4]), and let s', t' denote the nodes corresponding to $(u, s), (t, v)$ respectively. Replace each node w (except s', t') in the line graph with two nodes w_1, w_2 and an edge (w_1, w_2) such that all edges into (out of) w are now into w_1 (out of w_2). The weight of (w_1, w_2) is the weight of the edge in G corresponding to w. Other edges have weight 0. This weight-preserving transformation gives a one-to-one correspondence between edge disjoint paths in the original graph and node disjoint paths in the new graph. Thus, for all the problems investigated in this paper, we only give algorithms for the node disjoint case.

3 The `MinMax 2DP` Problem

Our FPTAS for `MinMax 2DP` on DAGs is based on the pseudo-polynomial-time algorithm by Li *et al.* [7] and the weight scaling technique (cf. [5,6,10]).

First, we use PSA to construct the graph G'_2 as in Eqs. (3) and (4). Then `MinMax 2DP` is equivalent to finding a directed path P from $\langle s,s\rangle$ to $\langle t,t\rangle$ in G'_2 minimizing $\max\{\mathcal{F}(P_H),\mathcal{F}(P_V)\}$, where P_H (P_V) denotes the horizontal (vertical) edges of P.

The pseudo-polynomial-time algorithm uses a standard labeling method. If there is a directed path P from $\langle s,s\rangle$ to a node $\langle i,j\rangle \in V'_2$, we label it by $(X,Y,Pred)$, where X (Y) is a positive integer denoting the total weight of all horizontal (vertical) edges in P and $Pred$ is the index of the predecessor of $\langle i,j\rangle$ in P. We compute for each node in topological order a set of labels for all the paths from $\langle s,s\rangle$ to that node. When the algorithm terminates, the label in the label set of $\langle t,t\rangle$ minimizing $\max\{X,Y\}$ is the solution.

Unfortunately, the number of labels lab^2 for one node may be exponentially large, where $lab=(n-1)\cdot\max_{e\in E}\mathcal{F}(e)$. In order to obtain a polynomial-time algorithm, we must somehow compress the label set. We use the scaling technique known from the Subset Sum FPTAS.

We store the labels of each node $\langle i,j\rangle$ in a 2-dimensional array $L_{i,j}[1\ldots\ell, 1\ldots\ell]$, where $\ell=\lfloor\log_{1+\delta} lab\rfloor+1$. Label $(X,Y,Pred)$ will be stored in $L[\lfloor\log_{1+\delta} X\rfloor+1, \lfloor\log_{1+\delta} Y\rfloor+1]$. Each cell of $L_{i,j}$ will store at most one label. We use a set $I_{i,j}$ to keep track of all the entries of $L_{i,j}$ that actually store a label. Then, for each node in the topological order, we compute the label set of the node. If a new label should be stored in an array cell which already contains another label, then we discard the new label. We let $\mathcal{F}_{i,j}$ denote the cost of edge (v_i,v_j) in E. The third component $Pred$ of the label $(X,Y,Pred)$ is now of the form $(\langle i',j'\rangle,(a,b))$ and is used to reconstruct the path P corresponding to the label, where $\langle i',j'\rangle$ is the index of the predecessor of $\langle i,j\rangle$ in P and (a,b) is the index of the cell of $L_{i',j'}$ from which $(X,Y,Pred)$ is computed.

The subroutine LABELSCALING computes for each node a scaled set of labels, while the main algorithm FPTAS-DAG-MinMax-2DP returns the approximate solution.

LABELSCALING$(G=(V,E),s,t,\mathcal{F},\delta)$

Construct $G'_2=(V'_2,E'_2)$ as in Eqs. (3) and (4);
Initialize matrices $L_{i,j}[k,l]=NULL$, **for** $1\le i,j\le n$ and
        $1\le k,l\le\lfloor\log_{1+\delta} lab\rfloor+1$;
$I_{i,j}=\emptyset$, **for** $1\le i,j\le n$;
$I_{s,s}=\{(1,1)\}$;
$L_{s,s}[1,1]=(0,0,NULL)$;
for $i\leftarrow 1$ to n **do**
    for $j\leftarrow 1$ to n **do**
        for each $(\langle i,k\rangle,\langle i,j\rangle)\in E'$
            for each $(a,b)\in I_{i,k}$
                let $(X,Y,Pred)$ be the label in $L_{i,k}[a,b]$;

11 let $c = \lfloor \log_{1+\delta}(X + \mathcal{F}_{k,j}) \rfloor + 1$;
12 **if** $L_{i,j}[c, b] == NULL$
13 then $L_{i,j}[c, b] = (X + \mathcal{F}_{k,j}, Y, (\langle i, k\rangle, (a, b)))$;
14 $I_{i,j} = I_{i,j} \cup \{(c, b)\}$;
15 **for** each $(\langle k, j\rangle, \langle i, j\rangle) \in E'_2$
16 **for** each $(a, b) \in I_{k,j}$
17 let $(X, Y, Pred)$ be the label in $L_{k,j}[a, b]$;
18 let $d = \lfloor \log_{1+\delta}(Y + \mathcal{F}_{k,i}) \rfloor + 1$;
19 **if** $L_{i,j}[a, d] == NULL$
20 then $L_{i,j}[a, d] = (X, Y + \mathcal{F}_{k,i}, (\langle k, j\rangle, (a, b)))$;
21 $I_{i,j} = I_{i,j} \cup \{(a, d)\}$;

FPTAS-DAG-MinMax-2DP($G = (V, E), s, t, \mathcal{F}, \epsilon$)

1 $\delta = (1 + \epsilon)^{\frac{1}{2n-2}} - 1$;
2 LABELSCALING($G = (V, E), s, t, \mathcal{F}, \delta$)
3 **for** each index (a, b) in $I_{t,t}$
4 let $L_{t,t}[a, b] = (X, Y, Pred)$;
5 find the (a^*, b^*) minimizing $\max\{X, Y\}$;
6 Reconstruct the two disjoint paths from the third components of the labels;

Lemma 2. *Let $\langle i, j\rangle$ be a node in V'_2 such that there is a directed path P from $\langle s, s\rangle$ to $\langle i, j\rangle$. Let $X = \mathcal{F}(P_H)$ and $Y = \mathcal{F}(P_V)$. When the algorithm LABELSCALING terminates, then there exists an index pair (a, b) such that the label $(\tilde{X}, \tilde{Y}, Pred)$ in $L_{i,j}[a, b]$ satisfies $X/(1 + \delta)^{i+j-2} \leq \tilde{X} \leq (1 + \delta)^{i+j-2} X$ and $Y/(1 + \delta)^{i+j-2} \leq \tilde{Y} \leq (1 + \delta)^{i+j-2} Y$.*

Proof. By induction on $i + j$. If $i + j = 2$, the node is $\langle 1, 1\rangle = \langle s, s\rangle$, and $L_{s,s}[1, 1] = (0, 0, NULL)$ is the correct result.

Let $i+j = l$ and $3 \leq l \leq 2n$. Suppose there is a path P from $\langle s, s\rangle$ to $\langle i, j\rangle$ with total weight of the horizontal (vertical) edges X (Y). Without loss of generality, assume the predecessor of $\langle i, j\rangle$ in P is $\langle i, j'\rangle$; the vertical case has a similar proof. Let the path from $\langle s, s\rangle$ to $\langle i, j'\rangle$ in P be P^+, and let X^+ (Y^+) be the total weight of the horizontal (vertical) edges of P^+. Then, $X = X^+ + \mathcal{F}_{j',j}$ and $Y = Y^+$. Since $j' < j$, $i + j' < i + j$, by the induction assumption there exists a label $(\tilde{X}^+, \tilde{Y}^+, Pred)$ such that $X^+/(1+\delta)^{i+j'-2} \leq \tilde{X}^+ \leq (1+\delta)^{i+j'-2} X^+$ and $Y^+/(1+\delta)^{i+j'-2} \leq \tilde{Y}^+ \leq (1+\delta)^{i+j'-2} Y^+$. So, when LABELSCALING computes the label set of the node $\langle i, j\rangle$ and enters steps 8 and 9, it will compute a new label $(\tilde{X}^+ + \mathcal{F}_{j',j}, \tilde{Y}^+, (\langle i, j'\rangle, (a, b)))$.

If the algorithm enters steps 12 to 14, the label $(\tilde{X}^+ + \mathcal{F}_{j',j}, \tilde{Y}^+, (\langle i, j'\rangle, (a, b)))$ will be stored. Let $\tilde{X} = \tilde{X}^+ + \mathcal{F}+_{j',j}, \tilde{Y} = \tilde{Y}^+$. Then, $\tilde{X} = \tilde{X}^+ + \mathcal{F}_{j',j} \leq (1 + \delta)^{i+j'-2} X^+ + \mathcal{F}_{j',j} \leq (1 + \delta)^{i+j'-2}(X^+ + \mathcal{F}_{j',j}) \leq (1 + \delta)^{i+j'-2} X \leq (1 + \delta)^{i+j-2} X$, and $\tilde{Y} = \tilde{Y}^+ \leq (1 + \delta)^{i+j'-2} Y^+ \leq (1 + \delta)^{i+j-2} Y$. Similarly, we have $\tilde{X} \geq X/(1 + \delta)^{i+j-2}$ and $\tilde{Y} \geq Y/(1 + \delta)^{i+j-2}$.

If the algorithm skips steps 12 to 14, then there exists a label $(\tilde{X}, \tilde{Y}, Pred)$. By the fact that $\lfloor \log_{1+\delta} \tilde{X} \rfloor + 1 = \lfloor \log_{1+\delta}(\tilde{X}^+ + \mathcal{F}_{j',j}) \rfloor + 1$ and $\lfloor \log_{1+\delta} \tilde{Y} \rfloor + 1 = \lfloor \log_{1+\delta} \tilde{Y}^+ \rfloor + 1$, we have $\tilde{X} < (1 + \delta)(\tilde{X}^+ + \mathcal{F}_{j',j}) \leq (1 + \delta)((1 + \delta)^{i+j'-2} X^+ +$

$\mathcal{F}_{j',j}) \leq (1+\delta)^{i+j-2}(X^+ + \mathcal{F}_{j',j}) \leq (1+\delta)^{i+j-2}X$, and $\tilde{Y} \leq (1+\delta)\tilde{Y}^+ \leq (1+\delta)^{i+j-2}Y^+ = (1+\delta)^{i+j-2}Y$. Similarly, we have $\tilde{X} \geq X/(1+\delta)^{i+j-2}$ and $\tilde{Y} \geq Y/(1+\delta)^{i+j-2}$. □

Theorem 1. *The algorithm FPTAS-DAG-MinMax-2DP is an FPTAS for the* `MinMax 2DP` *problem on DAGs.*

Proof. First, it is easy to verify that the time complexity of the algorithm is $O(n^3(\log_{1+\delta} lab)^2) = O(n^5\epsilon^{-2}(\ln lab)^2)$, where $lab = O(n \cdot \max_{e\in E} \mathcal{F}(e))$.

Second, we will show that the approximation ratio is $1+\epsilon$. Suppose the optimum solution is P_1, P_2. By Lemma 1, there is a path P in G_2' corresponding to P_1, P_2. Let $X = \mathcal{F}(P_H) = \mathcal{F}(P_1)$ and $Y = \mathcal{F}(P_V) = \mathcal{F}(P_2)$. By Lemma 2, there is a label $(\tilde{X}, \tilde{Y}, Pred)$ in $L_{t,t}$ such that $\tilde{X} \leq (1+\delta)^{2n-2}X = (1+\epsilon)X$ and $\tilde{Y} \leq (1+\delta)^{2n-2}Y = (1+\epsilon)Y$. Let $(X', Y', Pred')$ be the solution returned by the algorithm and $Opt = \max\{X, Y\}$. Then, $\max\{X', Y'\} \leq \max\{\tilde{X}, \tilde{Y}\} \leq \max\{(1+\epsilon)X, (1+\epsilon)Y\} = (1+\epsilon)\cdot\max\{X, Y\} = (1+\epsilon)OPT$. Thus, FPTAS-DAG-MinMax-2DP is an FPTAS for the `MinMax 2DP` problem on DAGs. □

For arbitrary constant $k \geq 2$, we can generalize the subroutine LABELSCALING by first constructing $G_k = (V_k, E_k)$ as in Eqs. (1) and (2) from G, using a k-dimensional array $L_{d_1,\ldots,d_k}[1\ldots\ell,\ldots,1\ldots\ell]$, where $\ell = \lfloor \log_{1+\delta} lab \rfloor + 1$, and then updating L the same way as in LABELSCALING. By induction on k, we can get the following result similar to Lemma 2.

Lemma 3. *Let $\langle d_1,\ldots,d_k\rangle$ be a node in V_k such that there is a directed path P from $\langle s,\ldots,s\rangle$ to $\langle d_1,\ldots,d_k\rangle$. Let $X_i = \mathcal{F}(P_{d_i})$, $1 \leq i \leq k$, where P_{d_i} denotes the edges in dimension i of P in G_k. When the algorithm LABELSCALING terminates, there exists an index pair $(l_1,\ldots,l_k)$ such that the label $(\tilde{X}_1,\ldots,\tilde{X}_k, Pred)$ in $L_{d_1,\ldots,d_k}[l_1,\ldots,l_k]$ satisfies $X_i/(1+\delta)^{d_1+\ldots+d_k-k} \leq \tilde{X}_i \leq (1+\delta)^{d_1+\ldots+d_k-k}X_i$ for $1 \leq i \leq k$.*

Then we can modify the algorithm FPTAS-DAG-MinMax-2DP by setting $\delta = (1+\epsilon)^{\frac{1}{kn-k}} - 1$ to get an FPTAS for the `MinMax k-DP` problem.

4 The `Balanced 2DP` Problem

The FPTAS for `Balanced 2DP` on DAGS is similar to the one described in Section 3, using the same subroutine LABELSCALING.

FPTAS-DAG-Balanced-2DP$(G = (V, E), s, t, \mathcal{F}, \epsilon)$

```
δ = (1 + ε)^(1/(4n−4)) − 1;
LABELSCALING(G = (V, E), s, t, F, δ)
for each index (a, b) in I_{t,t}
      let L_{t,t}[a, b] = (X, Y, Pred);
      find (a*, b*) minimizing max{X, Y}/ min{X, Y};
Reconstruct the 2 disjoint paths from the third entry of the labels;
```

Theorem 2. *FPTAS-DAG-Balanced-2DP is an FPTAS for the* `Balanced 2DP` *problem on DAGs.*

Proof. First, it can easily be seen that the time complexity of FPTAS-DAG-Balanced-2DP is $O(n^3(\log_{1+\delta} lab)^2) = O(n^5\epsilon^{-2}(\ln lab)^2)$, where $lab = O(n \cdot \max_{e\in E} \mathcal{F}(e))$.

Next, we prove that the approximation ratio is $1+\epsilon$. Suppose the optimum solution is P_1, P_2. By Lemma 1, there is a path P in G'_2 corresponding to P_1, P_2. Let $X = \mathcal{F}(P_H) = \mathcal{F}(P_1)$ and $Y = \mathcal{F}(P_V) = \mathcal{F}(P_2)$. By Lemma 2, there is a label $(\tilde{X}, \tilde{Y}, Pred)$ in $L_{t,t}$ such that $X/\sqrt{1+\epsilon} = X/(1+\delta)^{2n-2}X \le \tilde{X} \le (1+\delta)^{2n-2}X = X\sqrt{1+\epsilon}$ and $Y/\sqrt{1+\epsilon} = Y/(1+\delta)^{2n-2}Y \le \tilde{Y} \le (1+\delta)^{2n-2}Y = Y\sqrt{1+\epsilon}$. Let $(X', Y', Pred')$ be the solution returned by the algorithm and $Opt = \max\{X,Y\}/\min\{X,Y\} = \max\{X/Y, Y/X\}$. Then, $\max\{X'/Y', Y'/X'\} \le \max\{\tilde{X}/\tilde{Y}, \tilde{Y}/\tilde{X}\} \le \max\{(1+\epsilon)X/Y, (1+\epsilon)Y/X\} = (1+\epsilon)\cdot\max\{X/Y, Y/X\} = (1+\epsilon)OPT$. Thus, FPTAS-DAG-Balanced-2DP is an FPTAS for the `Balanced 2DP` problem on DAGs. $\square$

We can generalize the algorithm FPTAS-DAG-Balanced-2DP for arbitrary constant $k \ge 2$ by setting $\delta = (1+\epsilon)^{\frac{1}{2kn-2k}} - 1$ and then using a similar way to update L as in FPTAS-DAG-Balanced-2DP. The correctness proof follows from Lemma 3 and a generalization of Theorem 2.

5 The `MinSum-MinMax 2DP` Problem

In this section, we present an FPTAS for the `MinSum-MinMax 2DP` problem on DAGs. The difference between this problem and the `MinMax 2DP` problem is that in this problem we should find two disjoint paths with `MinMax` objective among the set of two disjoint paths whose total weight is minimized.

In the pseudo-polynomial-time algorithm for the `MinMax 2DP` problem on DAGs, for each node $\langle i,j\rangle$ in G'_2, if there is a path from $\langle s,s\rangle$ to $\langle i,j\rangle$, then we will compute a label to store the information of this path. Now, for the `MinSum-MinMax 2DP` problem, instead of keeping information for all paths, we only store the information of the shortest paths. This can be done by scanning each node in topological order, and then computing for each node a set of labels corresponding to the shortest paths. We then can use the scaling method to convert the pseudo-polynomial-time algorithm to an FPTAS.

SHORTESTLABELSCALING$(G = (V,E), s, t, \mathcal{F}, \delta)$

```
Construct G'_2 = (V'_2, E'_2) as in Eqs. (3) and (4);
 Initialize  matrices L_{i,j}[k,l] = NULL, for 1 ≤ i,j ≤ n and
          1 ≤ k,l ≤ ⌊log_{1+δ} lab⌋ + 1;
I_{i,j} = ∅, for 1 ≤ i,j ≤ n;
I_{s,s} = {(1,1)};
L_{s,s}[1,1] = (0,0,NULL);
for i ← 1 to n do
    for j ← 1 to n do
```

$currentmin = +\infty$;
for each $(\langle i,k\rangle, \langle i,j\rangle) \in E'_2$
for each $(a,b) \in I_{i,k}$
let $(X, Y, Pred)$ be the label in $L_{i,k}[a,b]$;
if $X + \mathcal{F}_{k,j} + Y < currentmin$
then $I_{i,j} = \emptyset$;
set all entries of $L_{i,j}$ to $NULL$;
$currentmin = X + \mathcal{F}_{k,j} + Y$;
if $X + \mathcal{F}_{k,j} + Y == currentmin$
let $c = \lfloor \log_{1+\delta}(X + \mathcal{F}_{k,j}) \rfloor + 1$;
if $L_{i,j}[c,b] == NULL$
then $L_{i,j}[c,b] = (X + \mathcal{F}_{k,j}, Y, (\langle i,k\rangle, (a,b)))$;
$I_{i,j} = I_{i,j} \cup \{(c,b)\}$;
for each $(\langle k,j\rangle, \langle i,j\rangle) \in E'_2$
for each $(a,b) \in I_{k,j}$
let $(X, Y, Pred)$ be the label in $L_{k,j}[a,b]$;
if $X + Y + \mathcal{F}_{k,i} < currentmin$
then $I_{i,j} = \emptyset$;
set all entries of $L_{i,j}$ to $NULL$;
$currentmin = X + Y + \mathcal{F}_{k,i}$;
if $X + Y + \mathcal{F}_{k,i} == currentmin$
let $d = \lfloor \log_{1+\delta}(Y + \mathcal{F}_{k,i}) \rfloor + 1$;
if $L_{i,j}[a,d] == NULL$
then $L_{i,j}[a,d] = (X, Y + \mathcal{F}_{k,i}, (\langle k,j\rangle, (a,b)))$;
$I_{i,j} = I_{i,j} \cup \{(a,d)\}$;

FPTAS-DAG-MinSum-MinMax-2DP($G = (V,E), s, t, \mathcal{F}, \epsilon$)

$\delta = (1+\epsilon)^{\frac{1}{2n-2}} - 1$;
SHORTESTLABELSCALING($G = (V,E), s, t, \mathcal{F}, \delta$);
for each index (a,b) in $I_{t,t}$
let $L_{t,t}[a,b] = (X, Y, Pred)$;
find (a^*, b^*) minimizing $\max\{X, Y\}$;
Reconstruct the two disjoint paths from the third entry of the labels;

Lemma 4. *Let $\langle i,j\rangle$ be a node in V'_2 and P a shortest path from $\langle s,s\rangle$ to $\langle i,j\rangle$. Let $X = \mathcal{F}(P_H)$ and $Y = \mathcal{F}(P_V)$. When the algorithm SHORTEST LABELSCALING terminates, then there exists an index pair (a,b) such that the label $(\tilde{X}, \tilde{Y}, Pred)$ in $L_{i,j}[a,b]$ satisfies $X/(1+\delta)^{i+j-2} \leq \tilde{X} \leq (1+\delta)^{i+j-2}X$ and $Y/(1+\delta)^{i+j-2} \leq \tilde{Y} \leq (1+\delta)^{i+j-2}Y$.*

Proof. First, a simple induction on the topological order of the nodes shows that when SHORTESTLABELSCALING terminates, the labels of each node correspond to shortest paths to the nodes.

Second, similar to the proof of Lemma 2, we can show there is a label $(\tilde{X}, \tilde{Y}, Pred)$ satisfying $X/(1+\delta)^{i+j-2} \leq \tilde{X} \leq (1+\delta)^{i+j-2}X$ and $Y/(1+\delta)^{i+j-2} \leq \tilde{Y} \leq (1+\delta)^{i+j-2}Y$. □

From Lemma 4 and an analysis similar to the proof of Theorem 1, we obtain the following result.

Theorem 3. *FPTAS-DAG-MinSum-MinMax-2DP is an FPTAS for the* `MinSum-MinMax 2DP` *problem on DAGs.* □

We can generalize the subroutine SHORTESTLABELSCALING for arbitrary constant $k \geq 2$ in a similar way as in the generalization of subroutine LABELSCALING in Section 3, and thus give an FPTAS for the the `MinSum-MinMax k-DP` problem on DAGs.

6 The `MinSum-MinMin 2DP` Problem

In this section, we present an efficient polynomial-time algorithm for the `MinSum-MinMin 2DP` problem on DAGs.

We introduce some notions. Let $\mathbb{Z}^2 = \{(X,Y) \mid X,Y \in \mathbb{Z}\}$ be the set of all pairs of positive integers. We define the relationship '<' on $\mathbb{Z}^2$ as: $(X,Y) < (X',Y')$ if and only if $X < X'$ or ($X = X'$ and $Y < Y'$). The operation '+' on two elements of $\mathbb{Z}^2$ is defined as $(X,Y) + (X',Y') = (X+X', Y+Y')$.

We first use PSA to transform the given graph G into G'_2, but with different edge weights than before. We let the weight of an edge in G'_2 be an element of $\mathbb{Z}^2$. Let $\mathcal{F}' : E'_2 \mapsto \mathbb{N}$ be the new weight function on G'_2. For a horizontal edge $(\langle i,j\rangle, \langle i,j'\rangle)$, $\mathcal{F}'((\langle v_i,v_j\rangle, \langle v_i, v_{j'}\rangle)) = (\mathcal{F}_{j,j'}, \mathcal{F}_{j,j'})$, and for a vertical edge $(\langle i,j\rangle, \langle i',j\rangle)$, $\mathcal{F}'((\langle i,j\rangle, \langle i',j\rangle)) = (\mathcal{F}_{i,i'}, 0)$. For G'_2 and weight function $\mathcal{F}'$, we compute the shortest path P from $\langle s,s\rangle$ to $\langle t,t\rangle$. It can be shown that the two disjoint paths from s to t in G corresponding to P in G' are an optimum solution to the `MinSum-MinMin 2DP` problem.

DAG-MinSum-MinMin-2DP$(G = (V,E), s, t, \mathcal{F})$

Construct $G'_2 = (V'_2, E'_2)$ as in Eq. (3) and (4);
for each $\langle i,j\rangle \in V'_2$
    let $d_{i,j} = (+\infty, +\infty)$;
        $p_{i,j} = NULL$;
$d_{s,s} = (0,0)$;
for $i \leftarrow 1$ to n **do**
    for $j \leftarrow 1$ to n **do**
        for each $(\langle i,k\rangle, \langle i,j\rangle) \in E'_2$
            if $d_{i,k} + (\mathcal{F}_{k,j}, \mathcal{F}_{k,j}) < d_{i,j}$
                then $d_{i,j} = d_{i,k} + (\mathcal{F}_{k,j}, \mathcal{F}_{k,j})$;
                    $p_{i,j} = \langle i,k\rangle$;
        for each $(\langle k,j\rangle, \langle i,j\rangle) \in E'_2$
            if $d_{k,j} + (\mathcal{F}_{k,i}, 0) < d_{i,j}$
                then $d_{i,j} = d_{k,j} + (\mathcal{F}_{k,i}, 0)$;
                    $p_{i,j} = \langle k,j\rangle$;
Reconstruct the two disjoint paths from $p_{t,t}$;

It can easily be shown that $d_{i,j}$ is the value of the shortest path from $\langle s,s\rangle$ to $\langle i,j\rangle$ with respect to the weight function $\mathcal{F}'$.

When the algorithm DAG-MinSum-MinMin-2DP terminates, for any node $\langle i,j\rangle$ in G'_2, let P be the path from $\langle s,s\rangle$ to $\langle i,j\rangle$ constructed by tracing backwards from $p_{i,j}$ to $\langle s,s\rangle$. Let $\mathcal{F}'(P) = (X,Y)$, then by the definition of $\mathcal{F}'$, we have $\mathcal{F}(P) = X$ and $\mathcal{F}(P_H) = Y$. Since (X, Y) is minimized, X is also minimized, and for any path P' from $\langle s, s\rangle$ to $\langle i, j\rangle$ such that $\mathcal{F}(P') = X$, we have $\mathcal{F}(P_H) \leq \mathcal{F}(P'_H)$. We also have $Y \leq X - Y$, that is, $\mathcal{F}(P_H) \leq \mathcal{F}(P_V)$. Suppose for contradiction, $\mathcal{F}(P_H) > \mathcal{F}(P_V)$, then by the symmetry of the construction of G'_2, there is another path P' that $P_H = P'_V$ and $P_V = P'_H$. Thus, $\mathcal{F}(P') = X$ and $\mathcal{F}(P'_V) = \mathcal{F}(P_H) = Y > \mathcal{P_V} = \mathcal{F}(P'_H)$, contradicting the fact that (X, Y) is minimal.

The above result is also true for $\langle t, t\rangle$. This proves the correctness of the algorithm. The running time of the algorithm is $O(|E'|) = O(n^3)$.

We note that our algorithm can easily be generalized to the case of $k > 2$, in contrast to the algorithm by Yang *et al.* [12]. When $k > 2$, we construct G'_k as in Eqs. (1) and (2), and again set the weight of each edge in G'_k to be an element of $\mathbb{Z}^2$. The first integer of the weight is the sum of the weights of all k paths, and the second integer is the weight of the minimum weight path. Then, we use a standard shortest path algorithm to compute the shortest path in G' under the new weight function.

References

1. P. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows, Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, 1993.
2. J. Bang-Jensen and G. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer, 2000.
3. S. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homemorphism problem. *Theoretical Computer Science*, 10:111–121, 1980.
4. F. Harary. *Graph Theory*. Addison-Wesley, Reading, MA, 1972.
5. R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17:237–260, 1992.
6. O. H. Ibarra and C. E. Kim. Fast approximation for the knapsack and sum of subset problems. *JACM*, 22:463–468, 1975.
7. C. L. Li, S. T. McCormick, and D. Simchi-Levi. The complexity of finding two disjoint paths with Min-Max objective function. *Discrete Applied Mathematics*, 26(1):105–115, 1990.
8. Y. Perl and Y. Shiloach. Finding two disjoint paths between two pairs of vertices in a graph. *J. ACM*, 25(1):1–9, 1978.
9. J. W. Suurballe and R. Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14:325–336, 1984.
10. G. Tsaggouris and C. Zaroliagis. Improved FPTAS for Multiobjective Shortest Paths with Applications. CTI Technical Report TR 2005/07/03, July 2005.
11. B. Yang, S. Q. Zheng, and S. Katukam. Finding two disjoint paths in a network with Min-Min objective function. In *Proc. 15th IASTED International Conference on Parallel and Distributed Computing and Systems*, pp. 75–80, 2003.
12. B. Yang, S. Q. Zheng, and E. Lu. Finding two disjoint paths in a network with MinSum-MinMin objective function. Technical Report, Dept. of Computer Science, Univ. of Texas at Dallas, April, 2005.
13. B. Yang, S. Q. Zheng, and E. Lu. Finding two disjoint paths in a network with normalized $\alpha^+ - MIN - SUM$ objective function. In *Proc. 16th International Symp. on Algorithms and Computation*, pp. 954–963, 2005.