Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Processing Letters 94 (2005) 29–35
Information Processing Letters
www.elsevier.com/locate/ipl

A linear time algorithm for the minimum Weighted Feedback Vertex Set on diamonds

F. Carrabs [a], R. Cerulli [a,*] M. Gentili [a], G. Parlato [b],

[a] *Dipartimento di Matematica ed Informatica, Università di Salerno, 84084 Fisciano (SA), Italy*
[b] *Dipartimento di Informatica ed Applicazioni, Università di Salerno, 84081 Baronissi (SA), Italy*

Received 29 June 2004; received in revised form 26 November 2004

Available online 16 January 2005

Communicated by F. Meyer auf der Heide

Abstract

Given an undirected and vertex weighted graph G, the Weighted Feedback Vertex Problem (WFVP) consists in finding a subset $F \subseteq V$ of vertices of minimum weight such that each cycle in G contains at least one vertex in F. The WFVP on general graphs is known to be NP-hard. In this paper we introduce a new class of graphs, namely the *diamond* graphs, and give a linear time algorithm to solve WFVP on it.

Keywords: Graph algorithms; Weighted Feedback Vertex Set; Dynamic programming; Diamond

1. Introduction

Given an undirected graph $G = (V, E)$, a *Feedback Vertex Set* (FVS) of G is a subset $F \subseteq V$ of vertices such that each cycle in G contains at least one vertex in F, i.e., the subgraph G' induced by the set $V \setminus F$ of vertices is acyclic. The *Feedback Vertex Problem* (FVP) consists in finding an FVS of minimum cardinality. When with each vertex v of G is associated a weight $w(v)$ we have a *vertex weighted graph*. The Weighted Feedback Vertex Problem (WFVP) on a weighted graph G consists in finding an FVS of minimum weight, where the weight of the set is the sum of the weights of its elements. Both FVP and WFVP have application in several areas of computer science such as circuit testing, deadlock resolution, placement of converters in optical networks, combinatorial cut design. The FVP on general graph is known to be NP-hard [5]. For the WFVP the best known approximation algorithm has approximation ratio 2 (see, for example, [2,1]). This problem becomes polynomial when addressed on interval graphs [7], co-comparability

* Corresponding author. Tel.: +39 089 963326, fax: +39 089 963303.
E-mail addresses: fcarrabs@unisa.it (F. Carrabs), raffaele@unisa.it (R. Cerulli), mgentili@unisa.it (M. Gentili), gparlato@unisa.it (G. Parlato).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2004.12.008

graphs [3], permutation graphs [6], convex bipartite graphs [3].

In this paper we introduce a new class of graphs, namely the *diamond* graphs, and give a linear time algorithm to solve WFVP on it.

The sequel of the paper is organized as follows. Section 2 describes the class of diamond graphs. Section 3 contains the description of our linear time algorithm based on dynamic programming to optimally solve WFVP on diamonds. In Section 4 we will discuss how this polynomial result can be used to improve an approximated solution on a general graph, that is the object of our further research.

2. The class of diamond graphs

In this section we describe formally the class of diamond graphs. First we introduce the needed notation (for any additional definition and notation we refer to [4]).

Let $G = (V, E, w)$ be an undirected and vertex weighted graph, where V is the set of vertices, E is the set of edges, and, $w(v)$ is a positive weight associated with each vertex $v \in V$. Given a subset $X \subseteq V$ of vertices, we define its weight $W(X)$ as the sum of the weights of its elements, i.e., $W(X) = \sum_{v \in X} w(v)$. If $X = \emptyset$ then $W(X) = 0$. We denote by $G \setminus X$ the subgraph of G induced by the set of vertices $V \setminus X$. A *tree* is an acyclic and connected graph. Given a rooted tree T, we denote by C_u the set of children of vertex u in T. We define the *height* $h(u)$ of a vertex u in T, recursively as follows. If u is a leaf then $h(u) = 0$, otherwise $h(u) = \max_{x \in C_u}\{h(x)\} + 1$. We define the height $h(T)$ of the tree to be equal to the height of its root. Given a vertex u of T, the *subtree* T_u rooted in u is the subgraph of T induced by the set of vertices constituted by u and its descendants in T.

Now we introduce the class of *Diamond* graphs.

Definition 1. A weighted *diamond* $D_{r,z} = (V, E, w)$ with apices r and z ($r, z \in V$), is an undirected and vertex weighted graph where (i) each $v \in V$ is included in at least one simple path between r and z and (ii) $D_{r,z} \setminus \{z\}$ is a tree.

We refer to the two vertices r and z of a diamond $D_{r,z}$ as, respectively, the *upper* and *lower* apex of $D_{r,z}$, and, to the subgraph $D_{r,z} \setminus \{z\}$ as the tree T_r rooted in r associated with $D_{r,z}$. Consider Fig. 1(a) as an example. We have the diamond $D_{1,10}$ with upper apex $r = 1$ and lower apex $z = 10$. Note that by deleting vertex z we obtain the tree $T_1 = D_{1,10} \setminus \{z\}$.

We denote by $D_{u,z} = (V_u, E_u, w)$ the subdiamond of $D_{r,z}$, with apices u and z, induced by the vertices of T_u and vertex z. We define the *height* $h(D_{u,z})$ of the diamond $D_{u,z}$ to be equal to the height of the associated tree T_u.

Finally, observe that given a diamond $D_{r,z}$, we can see it as composed by the upper apex r and the subdiamonds $D_{r_i,z}$, for each $r_i \in C_r$, where all of them have the common lower apex z. Consider again

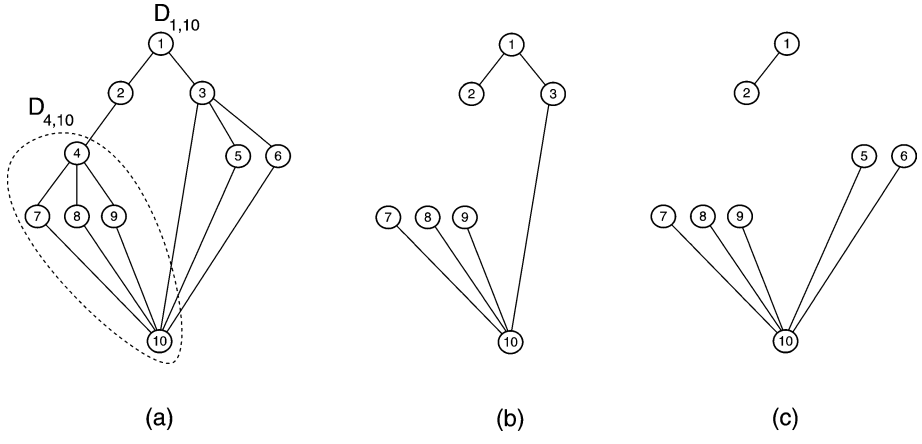


Fig. 1. (a) A diamond with apices $r = 1$ and $z = 10$. (b) The acyclic subgraph $D_{1,10} \setminus \hat{F}$ with a single path between vertices 1 and 10 when $\hat{F} = \{4, 5, 6\}$ and (c) the subgraph $D_{1,10} \setminus \hat{F}$, with $\hat{F} = \{3, 4\}$ without any path between vertices 1 and 10.

Fig. 1(a). The subgraph composed by the vertex set $\{4, 7, 8, 9, 10\}$ is a diamond with apices $r = 4$ and $z = 10$ ($D_{4,10}$), whose associated tree is the subtree T_4 of T_1. The height of T_4 is $h(T_4) = h(4) = 1$ that is also the height of $D_{4,10}$. Moreover, we can see $D_{1,10}$ as composed by vertex 1 and the two subdiamonds $D_{2,10}$ and $D_{3,10}$.

3. The dynamic programming algorithm

To simplify notation, in the rest of the section we denote a diamond $D_{r,z}$ just as D_r, since all considered diamonds have the same lower apex z.

In this section, we propose a linear time algorithm to solve the WFVP on a diamond based on dynamic programming. We will introduce two new problems on diamonds (the *Path* problem and the *NoPath* problem) and show how to obtain a minimum weighted FVS on D_r by the optimum solutions of these two problems. We then prove they have an optimal substructure property that allows us to optimally solve them by means of dynamic programming.

In the sequel of the paper we will consider a diamond D_r with upper apex r and lower apex z. To better clarify the role of the two new problems in our resolution algorithm, we state now the following observation.

Observation 1. By definition of diamond, the set $F = \{z\}$ is an FVS of D_r. Hence, an optimum solution F^* of WFVP on D_r is such that, either it is composed of the single vertex z (i.e., $F^* = \{z\}$) or, it does not contain vertex z and it is such that $W(F^*) \leqslant w(z)$. Therefore, the WFVP on a diamond can be solved, first by looking for the minimum FVS, say $\hat{F}$, that does not contain vertex z, and then, by choosing the minimum weight set between $\hat{F}$ and $\{z\}$.

Let $\hat{F} \subseteq V \setminus \{z\}$ be an FVS on D_r. Note that the subgraph $D_r \setminus \hat{F}$ either contains a single simple path between r and z or it does not contain any of such a path, as it is stated by the following proposition.

Proposition 1. *Given a diamond D_r with apices r and z, if $\hat{F} \subseteq V \setminus \{z\}$ is an FVS of D_r, then there exists at most one path between r and z in $D_r \setminus \hat{F}$.*

Proof. By contradiction let us suppose that there exist two distinct paths p_1 and p_2 between r and z in $D_r \setminus \hat{F}$. Let $v \neq z$ be the last vertex of the common longest subpath of p_1 and p_2 starting from r. Then the subpaths from v to z of p_1 and p_2, joined together, form a cycle in $D_r \setminus \hat{F}$, but this is a contradiction since $D_r \setminus \hat{F}$ must be acyclic being $\hat{F}$ an FVS of D_r. □

For example, consider again the diamond in Fig. 1. The set $\hat{F} = \{4, 5, 6\}$ is an FVS that does not contain vertex z and such that there exists the simple path $P = \{1, 3, 10\}$ connecting r and z in $D_r \setminus \hat{F}$ (see Fig. 1(b)). The set $\hat{F} = \{3, 4\}$ is an FVS that does not contain z and such that there is no path connecting r and z in $D_r \setminus \hat{F}$ (see Fig. 1(c)).

Now, we are ready to define two new problems that will be useful to solve WFVP on D_r.

Path problem. Given a diamond D_r, find a subset $F_r^+ \subseteq V \setminus \{z\}$ of minimum weight such that (i) $D_r \setminus F_r^+$ does not contain cycles, and (ii) there exists exactly a single path in $D_r \setminus F_r^+$ between r and z.

NoPath problem. Given a diamond D_r, find a subset $F_r^- \subseteq V \setminus \{z\}$ of minimum weight such that (i) $D_r \setminus F_r^-$ does not contain cycles, and (ii) there does not exist a path in $D_r \setminus F_r^-$ between r and z.

From the above observations it follows that the optimum solution F^* of WFVP on D_r is such that:

$$W(F^*) = \min\{w(z), W(F_r^+), W(F_r^-)\}$$

and, therefore we have either $F^* = \{z\}$ or $F^* = F_r^+$ or $F^* = F_r^-$.

Hence, we are interested now in solving both *Path* and *NoPath* problems.

3.1. Optimal substructure and recursion rules

In this section, we conjunctly characterize the structure of an optimal solution for both *Path* and *NoPath* problems. We recall that a problem has the optimal substructure property if any optimal solution to the problem contains within it optimal solutions to subproblems [4]. We will see that both *Path* and *NoPath* problems, defined on D_r, have an optimal substructure property by considering as subproblems those of determining the solution to *Path* and *NoPath*

on the subdiamonds D_u, for each $u \in V \setminus \{z\}$. Let us denote by F_u^+ and F_u^- the optimal solutions of *Path* and *NoPath* problem, respectively, on D_u. Now, we prove that the optimal solution F_u^+ (F_u^-) contains, for each $u_i \in C_u$, either $F_{u_i}^+$ or $F_{u_i}^-$.

Proposition 2. *Given the optimum solution F_u^+ on D_u, then each set $F_{u_i} = F_u^+ \cap V_{u_i}$, $u_i \in C_u$, is an optimum solution to either Path problem or NoPath problem on D_{u_i}.*

Proof. By feasibility of F_u^+, it follows that $z \notin F_{u_i}$ and $D_{u_i} \setminus F_{u_i}$ is acyclic. Then, the set F_{u_i} respects condition (i) for both *Path* problem and *NoPath* problem on the subdiamond D_{u_i}. Now, two cases, based on the fact that either there exists a path between u_i and z on $D_{u_i} \setminus F_{u_i}$ or not, may occur. If there is such a path, then F_{u_i} is the optimum solution set for *Path* problem on D_{u_i}, i.e., $F_{u_i} = F_{u_i}^+$. Indeed, if there were another feasible set F'_{u_i} for *Path* problem on D_{u_i} such that $W(F'_{u_i}) < W(F_{u_i})$, we could obtain the set $F'^+_u = (F_u^+ \setminus F_{u_i}) \cup F'_{u_i}$ that is feasible for *Path* problem on D_u and such that $W(F'^+_u) < W(F_u^+)$ obtaining a contradiction. If there does not exit any path between u_i and z on $D_{u_i} \setminus F_{u_i}$, then by applying a similar reasoning it follows that $F_{u_i} = F_{u_i}^-$. □

The same considerations can be made for the optimum solution F_u^- of the *NoPath* problem as stated by the following proposition.

Proposition 3. *Given the optimum solution F_u^- on D_u, then each set $F_{u_i} = F_u^- \cap V_{u_i}$, $u_i \in C_u$, is an optimum solution to either Path problem or NoPath problem on D_{u_i}.*

Following Lemmas 1 and 2, we describe how to build the weights of sets F_u^+ and F_u^- on D_u, by considering the value of optimum solution of *Path* and *NoPath* problems on D_x, for each $x \in C_u$. In the next section we will use these values to construct the optimum sets F_r^+ and F_r^-.

Lemma 1.

Case A: $h(u) = 0$.
$W(F_u^+) = 0$.
Case B: $h(u) > 0$ *and* $(u, z) \in E$.
$W(F_u^+) = \sum_{x \in C_u} W(F_x^-)$.
Case C: $h(u) > 0$ *and* $(u, z) \notin E$.
$W(F_u^+) = \min_{x \in C_u}\{W(F_x^+) + \sum_{y \in C_u \setminus \{x\}} W(F_y^-)\}$.

Proof. *Case* A is easily verified since $F_u^+ = \emptyset$.

Case B: if $(u, z) \in E$, since $u \notin F_u^+$, then the path connecting u and z in $D_u \setminus F_u^+$ is composed by the single edge (u, z). Thus, to avoid cycles in $D_u \setminus F_u^+$, the set F_u^+ is obtained, by Propositions 1 and 2, by the union of the optimum sets F_x^-, $x \in C_u$, and we obtain $W(F_u^+) = \sum_{x \in C_u} W(F_x^-)$.

Case C: if $(u, z) \notin E$, since $u \notin F_u^+$, in order to have a path between u and z in $D_u \setminus F_u^+$ and by applying Propositions 1 and 2, the optimum set is obtained by the minimum weight union of exactly one set F_x^+, $x \in C_u$, and, F_y^-, $y \in C_u \setminus \{x\}$. Therefore, $W(F_u^+) = \min_{x \in C_u}\{W(F_u^+) + \sum_{y \in C_u \setminus \{x\}} W(F_y^-)\}$. □

Lemma 2.

Case A: $h(u) = 0$.
$W(F_u^-) = w(u)$.
Case B: $h(u) > 0$ *and* $(u, z) \in E$.
$W(F_u^-) = w(u) + \sum_{x \in C_u} \min\{W(F_x^-), W(F_u^+)\}$.
Case C: $h(u) > 0$ *and* $(u, z) \notin E$.

$$W(F_u^-) = \min\left\{ w(u) + \sum_{x \in C_u} \min\{W(F_x^-), W(F_x^+)\}, \sum_{x \in C_u} W(F_x^-) \right\}.$$

Proof. *Case* A is easily verified since $F_u^- = \{u\}$.

Case B: if $(u, z) \in E$, then $u \in F_u^-$ otherwise there would be a path between u and z in $D_u \setminus F_u^-$. Therefore, by applying Propositions 1 and 3, we select on each subdiamond D_x, $x \in C_u$, the minimum weighted set among F_x^+ and F_x^-. Then $W(F_u^-) = w(u) + \sum_{x \in C_u} \min\{W(F_x^-), W(F_x^+)\}$.

Case C: if $(u, z) \notin E$, then u can either belong to F_u^- or not. If $u \in F_u^-$, there is no path between u and z in $D_u \setminus F_u^-$, thus we have the same solution as for Case B. If $u \notin F_u^-$ then to avoid paths between the apices in $D_u \setminus F_u^-$, the optimum set is obtained by the union of the optimum solutions of the *NoPath* problem on the subdiamonds D_x for each child x of vertex u, therefore we would have $W(F_u^-) = \sum_{x \in C_u} W(F_u^-)$. Thus, for this case, $W(F_u^-) = \min\{w(u) + \sum_{x \in C_u} \min\{W(F_x^-), W(F_x^+)\}, \sum_{x \in C_u} W(F_x^-)\}$. □

According to lemmas above, the computation of the optimum weighted FVS value can be carried out by a dynamic programming algorithm. The algorithm scans all the vertices of T_r through a postorder visit starting from the root node r. At each vertex u, both weight $W(F_u^+)$ and $W(F_u^-)$, to *Path* and *NoPath* problems on D_u are computed. Once that the weights of these sets are computed for the upper apex r, the optimum set among $\{z\}$, F_r^+ and F_r^- is chosen.

The computational complexity of the algorithm, sketched above, is given by the sum of the computational complexity needed to compute $W(F_u^+)$ and $W(F_u^-)$, for each vertex u of the tree T_r. The computation involved in both cases A of Lemmas 1 and 2 takes O(1) time. For cases B it takes $O(|C_u|)$ time. Indeed, the computation of $W(F_u^+)$ is carried out by computing the sum of $|C_u|$ values, while, to compute $W(F_u^-)$, $|C_u|$ minimum operations between two values are carried out and the sum of $|C_u| + 1$ values is computed. It follows directly that the evaluation of $W(F_u^-)$, for case C of Lemma 2, takes $O(|C_u|)$ time. Finally, observe that, to compute $W(F_u^+)$ for case C of Lemma 1, we can evaluate $\min_{x\in C_u}\{S - W(F_x^-) + W(F_x^+)\}$, where $S = \sum_{x\in C_u} W(F_x^-)$, which can be accomplished in $O(|C_u|)$ time. Thus, the computational complexity of the algorithm is given by

$$\sum_{u\in V\setminus\{z\}} O\big(|C_u|\big) = O\big(|E|\big) = O\big(|V|\big),$$

where the last relation follows from the fact that for any diamond $|E| \leqslant 2|V|$. The above observations prove the following theorem.

Theorem 1. *The solution value of the weighted feedback vertex set problem on diamonds can be computed* $O(|V|)$ *time.*

The detailed dynamic programming algorithm is given in Fig. 2.

3.2. Construction of F_r^+ and F_r^- sets

In this section we describe the linear time recursive procedure, Build_Solution, to build the feedback vertex set whose optimum value is found by our dynamic

Input: A weighted diamond $D_r = (V, E, w)$ with upper apex r and lower apex z.
Output: The weight $W(F^*)$ of an optimum weighted feedback vertex set.

Compute_set_values(D_r, r);
$W(F^*) = \min\{w(z), W(F_r^+), W(F_r^-)\}$;
return $W(F^*)$;

Procedure Compute_set_values(D_r, u)
/* Case A of Lemmas 1 and 2 */
If $h(u) = 0$ **then**
  $W(F_u^+) = 0$; $W(F_u^-) = w(u)$; **return**;
for each $x \in V$ such that x is a child of u in T_r **do**
  Compute_set_values(D_r, x);
/* Case B of Lemmas 1 and 2 */
If $(u, z) \in E$ **then**
  $W(F_u^+) = \sum_{x\in C_u} W(F_x^-)$;
  $W(F_u^-) = w(u) + \sum_{x\in C_u} \min\{W(F_x^-), W(F_x^+)\}$;
/* Case C of Lemmas 1 and 2 */
else

$$W(F_u^+) = \min_{x\in C_u}\left\{W(F_x^+) + \sum_{y\in C_u - \{x\}} W(F_y^-)\right\};$$

$$W(F_u^-) = \min\left\{w(u) + \sum_{x\in C_u} \min\{W(F_x^-), W(F_x^+)\}, \sum_{x\in C_u} W(F_x^-)\right\};$$

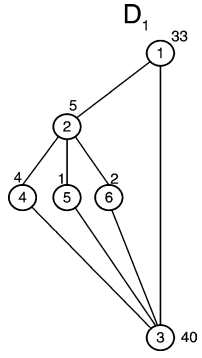
return;

Fig. 2. The dynamic programming algorithm.

Input: T_r, u, *flag* $\in \{+, -\}$
Output: Print the vertices of F_r^{flag}
Build_Solution($T_r, r,$ *flag*);

Procedure Build_Solution($T_r, u,$ *flag*)
if (*flag* $==-$ AND $I_u ==$ true)
  print(u);
if u is not a leaf
  for each child x of u **do**
    if *flag* $==+$ **then**
      Build_Solution(T_r, x, P_x^+);
    else
      Build_Solution(T_r, x, P_x^-);

Fig. 3. Build_Solution procedure.



Vertex	I_u	P_u^+	P_u^-
1	1	-	-
2	1	-	+
4	1	+	+
5	1	-	+
6	1	-	+

Fig. 4. On the left a diamond whose optimum weighted FVS is $F^* = \{2\}$ is given, and on the right, the values of the labels associated with each vertex are shown.

programming algorithm presented in the previous section. By Lemmas 1 and 2, given a vertex u of T_r, an easy way to build the optimal sets F_u^- and F_u^+ is to store the optimal sets $F_{u_i}^-$ and $F_{u_i}^+$ for each $u_i \in C_u$. However, from an implementation point of view, this choice would require both the storage of a large quantity of data and the use of complex data structures to manage sets. Now, we describe a more efficient strategy to build these optimum sets. In order to do that, we associate with each vertex u of T_r a boolean variable I_u that holds TRUE if and only if $u \in F_u^-$. Moreover, for each vertex $u \neq r$ of T_r, we associate two variables P_u^+ and P_u^- with values from $\{+, -\}$. In more detail, given $x \in C_u$, $P_x^+ = +$ ($P_x^- = +$) if $F_x^+ \subseteq F_u^+$ ($F_x^+ \subseteq F_u^-$) and $P_x^+ = -$ ($P_x^- = -$) if $F_x^- \subseteq F_u^+$ ($F_x^- \subseteq F_u^-$).

These variables are set during the execution of the dynamic programming algorithm and are used by the procedure Build_Solution shown in Fig. 3. This procedure takes as input the tree T_r, a vertex $u \in T_r$ and a *flag* $\in \{+, -\}$, and returns either the optimum set F_u^+ or F_u^- according to the value of flag.

For example, by calling Build_Solution($T_r, u, +$) the set F_u^+ is returned. Note that this set (by Propositions 2 and 3) contains, for each child x of u, either F_x^+ or F_x^-, and this information is described by the value of P_x^+. Therefore, the procedure is recursively called on each child x of u with parameters T_r, x, P_x^+. A similar reasoning is applied when Build_Solution is called to build F_u^-. However, since in this case u can either belong to F_u^- or not, the value of the variable I_u has to be considered too.

Consider as an example the diamond in Fig. 4. The upper apex is $r = 1$ and the lower apex is $z = 3$, the label on each vertex denotes its weight, for example, vertex 1 has weight $w(1) = 33$. The optimum weighted FVS is $F^* = \{2\}$ whose optimum weight is $W(F^*) = \min\{w(z), W(F_1^+), W(F_1^-)\}$, where $F_1^+ = \{2\}$ and $F_1^- = \{1, 5, 6\}$. The values of the labels to build the optimum sets F_u^+ and F_u^- are given in Fig. 4.

The optimum sets F_u^- associated with the vertices 4, 5, 6, are, respectively, $F_4^- = \{4\}$, $F_5^- = \{5\}$, $F_6^- = \{6\}$, and therefore, $I_4 = I_5 = I_6 =$ true. The set $F_2^+ = \{5, 6\}$ is computed by selecting $F_4^+ = \emptyset$, $F_5^- = \{5\}$ and $F_6^- = \{6\}$, therefore, $P_4^+ = +$, $P_5^+ = -$ and $P_6^+ = -$.

It is easy to see that the time complexity of procedure Build_Solution is linear. Hence, we have the main result of this section.

Theorem 2. *The weighted feedback vertex set problem on diamonds can be solved in* $O(|V|)$ *time.*

4. Conclusion and further research

In this paper we have presented the family of diamond graphs where it is possible to solve WFVP in linear time. We described a dynamic programming algorithm to compute both the value and the vertices that compose the optimal solution in $O(n)$ time. Object of our further research is both (i) the study of the larger class of multidiamond graphs (diamonds with multi-upper and/or lower apices), and (ii) the use of our exact algorithm on diamonds to improve the approximated solution returned by existing heuristics that solve WFVP on general graphs. To better clarify

this idea, let G be a graph and F be an approximate FVS returned by a given approximation algorithm. We could improve the solution of the given set F by substituting one or more of its vertices, say $F' \subseteq F$, with a set $S \subseteq V \setminus F$ of less weight such that the resulting new set is an FVS. Consider, for example, the acyclic subgraph $G \setminus F = T$, and, assume to add to it the vertex $z \in F$. The resulting graph is, after appropriate reduction operations, either a diamond or a multidiamond and by applying our algorithm we could improve the initial FVS.

References

[1] V. Bafna, P. Berman, T. Fujito, Constant ratio approximations of the weighted feedback vertex set problem for undirected graphs, in: ISAAC95, Algorithms and Computation, in: Lecture Notes in Comput. Sci., vol. 1004, Springer-Verlag, Berlin, 1995, pp. 142–151.

[2] A. Becker, D. Geiger, Approximation algorithms for the loop cutset problem, in: Proc. 10th Conf. on Uncertainty in Artificial Intelligence, 1994, pp. 60–68.

[3] M.S. Chang, Y.D. Liang, Minimum feedback vertex sets in cocomparability graphs and convex bipartite graphs, Acta Inform. 34 (1997) 337–346.

[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, Cambridge, MA, 2001.

[5] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, CA, 1979.

[6] Y.D. Liang, On the feedback vertex set problem in permutation graphs, Inform. Process. Lett. 52 (1994) 123–129.

[7] C.L. Lu, C.Y. Tang, A linear-time algorithm for the weighted feedback vertex problem on interval graphs, Inform. Process. Lett. 61 (1997) 107–111.