ISSN 1064-5624, Doklady Mathematics, 2014, Vol. 89, No. 2, pp. 253–256. © Pleiades Publishing, Ltd., 2014.
Original Russian Text © D.S. Malyshev, P.M. Pardalos, 2014, published in Doklady Akademii Nauk, 2014, Vol. 455, No. 5, pp. 529–532.

COMPUTER SCIENCE

Efficient Computation of Tolerances in the Weighted Independent Set Problem for Some Classes of Graphs

D. S. Malyshev[a] and P. M. Pardalos[b]

Presented by Academician Yu.G. Evtushenko October 11, 2013

Received October 11, 2013

DOI: 10.1134/S106456241402029X

After an optimal solution of a combinatorial optimization problem (COP) is obtained, the next natural step is an analysis of its sensitivity. The sensitivity analysis of an optimal solution of COP consists in determining the dependence of this solution on the variation of the initial data. The simplest sensitivity analysis studies the perturbations of only one element of an optimal solution. The purpose of the study of such perturbations is the determination of tolerances, that is, the maximum variation of an individual cost (such as weight, time, etc.) under which the optimality of a given solution is preserved, provided that the remaining COP's data are fixed.

The interest in tolerances is caused by that the minimum tolerance of elements of an optimal solution of a problem is a lower bound for the stability radius of this optimal solution and serves as a basis for developing and improving algorithms for some COPs. The first implicit algorithmic application of tolerances appeared in Vogel's method [12] for finding a solution closest to an optimal basic solution in the simplex method for solving the transportation problem. Among the cases where tolerances were successfully used we mention exact algorithms for solving the traveling salesman problem for oriented graphs [6] and the linear assignment problem [1].

Special attention is given to efficiently solvable classes of COPs, for which it is possible to simultaneously compute an optimal solution and all tolerances. Examples of such problems are the minimum spanning tree problem [14], the shortest path problem [11, 13], the assignment problem [15], and the problem on a weighted independent set problem for trees [2].

[a] *National Research University Higher School of Economics, Nizhni Novgorod Branch, Bol'shaya Pecherskaya ul. 25/12, Nizhni Novgorod, 603155 Russia*
[b] *University of Florida, 401 Weil Hall, P.O. Box 116595, Gainesville, FL 32611-6595, USA*
e-mail: dmalishev@hse.ru, dsmalyshev@rambler.ru

This paper is a continuation of [2]. It is devoted to the simultaneous determination of an optimal solution and all corresponding tolerances in the weighted independent set problem (WISP) for certain classes of graphs. For a simple graph with positive weights of vertices, this problem is to find a set of pairwise nonadjacent vertices of maximal weight in this graph. We consider bipartite and interval graphs and show that

(i) for a bipartite graph with n vertices and m edges, an optimal solution of WISP can be found in time $O(nm)$, and all tolerances can be computed in time $O(n^2)$;

(ii) for an interval graph with n vertices and m edges, an optimal solution of WISP and all tolerances can be computed in linear time $O(n + m)$.

These results not only generalize results of [2] but also are useful in developing exact and approximate algorithms for solving WISP in the general case. An algorithm presented in [2] has already been used for this purpose in [7]. Using WISP for interval graphs as a relaxation of certain NP-complete problem on sensor networks [4] is also promising.

1. COMBINATORIAL OPTIMIZATION PROBLEM AND THE TOLERANCES OF ITS ELEMENTS

A combinatorial optimization problem is determined by a quadruple $\{\Gamma, c, F, f_c\}$ of parameters, where $c: \Gamma \to \mathbb{R}_+$ is a weight function of the universal set Γ, $F \subseteq 2^\Gamma$ is the set of feasible solutions of the problem, and f_c is the objective function. Given a particular quadruple $\{\Gamma, c, F, f_c\}$, the corresponding COP consists in finding an element $S^* \in F$ at which the objective function takes its extremal (minimal or maximal) value. Such a set S^* is called an optimal solution of the problem. A combinatorial optimization problem is said to be additive if $f_c(S) = \sum_{s \in S} c(s)$ for any $S \in F$.

Suppose we are given a maximization additive COP, an optimal solution S^* of this problem, and elements $x \in S^*$ and $y \in \Gamma \backslash S^*$. We denote this problem by Π. We associate this problem with two other problems, Π_1 and Π_2. The problem Π_1 is obtained from Π by decreasing the weight of x by a nonnegative number w_1. The problem Π_2 is obtained from Π by increasing the weight of the element y by some nonnegative number w_2.

Let us return to the problem Π. The lower tolerance of the element x with respect to a given optimal solution S^* (denoted by $l_{S^*}(x)$) is defined as the supremum of those numbers w_1 for which the set S^* remains an optimal solution of Π_1, provided that all other data of Π are intact. The upper tolerance of the element y with respect to the optimal solution S^* (denoted by $u_{S^*}(y)$) is defined as the supremum of those numbers w_2 for which the set S^* remains an optimal solution of Π_2, provided that the other data of Π are intact. The meaning of the notions of lower and upper tolerances is that their minimal values are estimates for the stability radius of the solution S^* of the problem Π. More precisely, the problem Π has no solutions $S', S'' \in F$ such that $f_c(S^*) > f_c(S') > f_c(S^*) - l$ or $f_c(S^*) > f_c(S'') > f_c(S^*) - u$, where $l = \min\limits_{x \in S^*} l_{S^*}(x)$ and $u = \min\limits_{y \notin S^*} u_{S^*}(y)$.

In the study of the problem Π, we use the following notation:

$$F_-(x) = \{S \in F: x \notin S\},$$

$$F_+(y) = \{S \in F: y \in S\};$$

F^* is the set of optimal solutions of the problem Π;

$$F^*_-(x) = F_-(x) \cap F^*, \quad F^*_+(y) = F_+(y) \cap F^*;$$

$f_c(F') = \max\limits_{S \in F'} f_c(S)$, where F' is a subset of F.

The following lemma is valid (see, e.g., [3] or [10]).

Lemma 1. *Suppose that $S^* \in F^*$, $x \in S^*$, and $y \notin S^*$. Then $l_{S^*}(x) = f_c(F^*) - f_c(F^*_-(x))$ and $u_{S^*}(y) = f_c(F^*) - f_c(F^*_+(y))$.*

2. CASES OF JOINT EFFICIENT COMPUTATION OF AN OPTIMAL SOLUTION OF COP AND ALL TOLERANCES

By Lemma 1, if $f_c(F^*)$ is known, then, to compute the lower and upper tolerances of x and y, it suffices to find $f_c(F^*_-(x))$ and $f_c(F^*_+(y))$. However, it is expedient to use the information accumulated during the computation of $f_c(F^*)$ to determine $f_c(F^*_-(x))$ and $f_c(F^*_+(y))$ (rather than, say, run the algorithm again for data without/with x/y). This is one of the ideas used in computing all tolerances with the complexity not exceeding (or slightly exceeding) that of computing an optimal solution.

In [13], it was shown that, for a given edge-weighted graph on n vertices and some optimal solutions of the maximum flow and shortest path problems, the tolerances of all edges can be computed in time $O(n^2)$. Tarjan showed in [14] that, for the minimum spanning tree problem and a given optimal solution, all tolerances can be computed in time $O(m\alpha(n, m))$ (where $\alpha(n, m)$ is a very slowly growing the inverse Ackermann function). Of particular interest are the cases where both an optimal solution and all tolerances with respect to this solution can be calculated in linear time. In [5], the shortest path and minimum spanning tree problems were considered; it was shown that, for a planar graph on n vertices, an optimal solution and all tolerances for these problems can be calculated in time $O(n)$. This time is also sufficient for computing an optimal solution and all tolerances in WISP for an n-vertex tree [2].

In this paper, we show that, in the cases of bipartite and interval graphs, all tolerances in WISP can be computed in polynomial time not exceeding the time required for finding an optimal solution.

3. BIPARTITE AND INTERVAL GRAPHS

A graph G is said to be bipartite if the set $V(G)$ can be divided into two parts X and Y (called the parts of G) so that $E(G) \subseteq X \times Y$. A graph G is said to be interval if each of its vertices can be associated with an interval of the real line so that two vertices of G are adjacent if and only if the corresponding intervals intersect. This set of intervals is called an interval representation of G.

4. APPLICATION OF FLOW ALGORITHMS TO COMPUTING TOLERANCES IN WISP FOR BIPARTITE GRAPHS

A reduction of WISP for bipartite graphs to searching for a maximal flow in a network is well known. Let us describe this reduction. Suppose we are given a bipartite graph G with parts X and Y and a weight function c: $X \cup Y \to \mathbb{R}_+$. We construct an edge-weighted graph G' as follows. We augment G by two vertices s and t; s is joined by edges with all vertices of the part X, and t is joined with all vertices of Y. Each edge $e = (x, y)$ of the graph G' is assigned with a weight $c'(e)$ by the rule: if $x = s$, then $c'(e) = c(y)$; if $y = t$, then $c'(e) = c(x)$; otherwise, $c'(e) = \infty$ (infinity can be replaced by the sum $\sum\limits_{v \in X \cup Y} c(v)$). Recall that an (s, t)-cut of the graph G' is any partition of its vertex set into parts S and T such that $s \in S$ and $t \in T$. The weight of this cut is defined as $\sum\limits_{v \in S, u \in T} c'(v, u)$. It is well known that if (S, T) is a cut of G' with minimum weight, then $(A \cap S) \cup (B \cap T)$ is an independent set of G with maximum weight. By the Ford–Fulkerson theorem, a cut of G' with minimum weight can be

found by transforming this graph into a network (by introducing a natural orientation of the edges of G', with respect to which s is a source, t is a sink, and the capacities of arcs equal the corresponding weights) and applying any flow algorithm. At present, the best known algorithm for computing a maximal flow in a network with n vertices and m arcs is the Orlin algorithm [9] with the complexity $O(nm)$. Therefore, for a bipartite graph with n vertices and m edges, WISP can be solved in time $O(nm)$, and, according to a result of [13], the tolerances with respect to the found optimal solution can be computed in time $O(n^2)$.

5. COMPUTATION OF TOLERANCES IN WISP FOR INTERVAL GRAPHS IN LINEAR TIME

In this section, we consider WISP $\{\Gamma, c, F, f_c\}$ for an interval graph G with an interval representation $I(G)$. We assume that the vertices $v_1, v_2, \ldots, v_n$ of G are linearly ordered so that $i \le j \Leftrightarrow a_i \le a_j$, where a_i is the coordinate of the left endpoint of the interval from $I(G)$ corresponding to the vertex v_i. Given the graph G, the representation $I(G)$ and this ordering can be computed in time linear with respect to the sum of the numbers of its vertices and edges (this is a corollary of Algorithm 9 in [8]). We propose to determine an optimal solution of WISP for the graph G and the tolerances of all vertices with respect to this solution by the method of dynamical programming. For this purpose, for each $1 \le i \le n$, we introduce the notation

(i) $j_i' = \min(\{j: (v_j, v_i) \in E(G)\} \cup \{i\}) - 1$;

(ii) $j_i'' = \max(\{j: (v_j, v_i) \in E(G)\} \cup \{i\}) + 1$;

(iii) G_i' (G_i'') is the graph obtained from G by deleting the vertices $v_{i+1}, v_{i+2}, \ldots, v_n$ (respectively, $v_1, v_2, \ldots, v_{i-1}$);

(iv) $S'(i)$ and $S''(i)$ are optimal independent sets of the graphs G_i' and G_i'', respectively;

(v) $S_{\text{in}}'(i)$ and $S_{\text{out}}'(i)$ ($S_{\text{in}}''(i)$ and $S_{\text{out}}''(i)$) are independent sets in the graph G_i' (respectively, in the graph G_i'') which have maximum weight among all independent sets containing/not containing the vertex v_i;

(vi) $W'(i)$, $W_{\text{in}}'(i)$, $W_{\text{out}}'(i)$, $W''(i)$, $W_{\text{in}}''(i)$, and $W_{\text{out}}''(i)$ are the weights of the sets $S'(i)$, $S_{\text{in}}'(i)$, $S_{\text{out}}'(i)$, $S''(i)$, $S_{\text{in}}''(i)$, and $S_{\text{out}}''(i)$, respectively.

We also set $W'(0) = W_{\text{in}}'(0) = W_{\text{out}}'(0) = W''(n+1) = W_{\text{in}}''(n+1) = W_{\text{out}}''(n+1) = 0$ and $S'(0) = S_{\text{in}}'(0) = S_{\text{out}}'(0) = S''(n+1) = S_{\text{in}}''(n+1) = S_{\text{out}}''(n+1) = \phi$.

The graph G being interval leads to the recursive equations $W_{\text{out}}'(i) = W'(i-1)$, $W_{\text{in}}'(i) = c(v_i) + W'(j_i')$, $S_{\text{out}}'(i) = S'(i-1)$, and $S_{\text{in}}'(i) = S'(j_i') \cup \{v_i\}$. Clearly, $W'(i) = \max(W_{\text{in}}'(i), W_{\text{out}}'(i))$; moreover, $S'(i) = S_{\text{out}}'(i)$ if $W'(i) \ge W_{\text{in}}'(i)$ and $S'(i) = S_{\text{in}}'(i)$ otherwise. Recursive equations for $W''(i)$, $W_{\text{in}}''(i)$, $W_{\text{out}}''(i)$, $S''(i)$, $S_{\text{in}}''(i)$, and $S_{\text{out}}''(i)$ are similar.

It follows from Lemma 1 and G being interval that, for any vertex $v_i \notin S^* = S'(n)$, we have $u_{S^*}(v_i) = W'(n) - (W_{\text{in}}'(i) + W''(j_i''))$, and $S_{\text{in}}'(i) \cup S''(j_i'') \in F_+^*(v_i)$. It also follows from G being interval that, for any vertex $v_i \in S^*$, the value $f_c(F_-^*(v_i))$ equals

$$\max\Big(\max_{k \in \overline{j_i'+1, j_i''-1} \setminus \{i\}} (W_{\text{in}}'(k) + W''(j_k'')), W'(j_i') + W''(j_i'')\Big).$$

An element of the set $F_-^*(v_i)$ is defined accordingly. Thus, for any vertex $v_i \in S^*$, the lower tolerance $l_{S^*}(v_i)$ equals $W'(n) - \max\Big(\max_{k \in \overline{j_i'+1, j_i''-1} \setminus \{i\}} (W_{\text{in}}'(k) + W''(j_k'')), W'(j_i') + W''(j_i'')\Big)$.

A pseudocode of an algorithm solving WISP for the graph G and subsequently computing all tolerances is given later on. We skip computing the values of $S''(i)$, $S_{\text{in}}''(i)$, and $S_{\text{out}}''(i)$, which are not needed for computing the tolerances. If it is necessary to determine an element of $F_-^*(v_i)$ for $v_i \in S^*$ (an element of $F_+^*(v_i)$ for $v_i \notin S^*$), the algorithm can be supplemented in a natural way without the loss of the linear complexity.

Algorithm 1.

{

$W'(0) = 0$; $W_{\text{in}}'(0) = 0$; $W_{\text{out}}'(0) = 0$; $W''(n+1) = 0$; $W_{\text{in}}''(n+1) = 0$; $W_{\text{out}}''(n+1) = 0$;

$S'(0) = \phi$; $S_{\text{in}}'(0) = \phi$; $S_{\text{out}}'(0) = \phi$;

for ($i \in 1, \ldots, n$)

{

$j_i' = \min(\{j: (v_j, v_i) \in E(G)\} \cup \{i\}) - 1$;

$j_i'' = \max(\{j: (v_j, v_i) \in E(G)\} \cup \{i\}) + 1$;

}

for $(i \in \overline{1, n})$
{
$W'_{\text{out}}(i) = W'(i-1)$; $W'_{\text{in}}(i) = c(v_i) + W'(j'_i)$;
$S'_{\text{out}}(i) = S'(i-1)$; $S'_{\text{in}}(i) = S'(j'_i) \cup \{v_i\}$;
$W''_{\text{out}}(n-i+1) = W''(n-i+2)$; $W''_{\text{in}}(n-i+1) = c(v_{n-i+1}) + W''(j''_{n-i+1})$;
$W'(i) = \max(W'_{\text{in}}(i), W''_{\text{out}}(i))$; $W''(n-i+1) = \max(W''_{\text{in}}(n-i+1), W''_{\text{out}}(n-i+1))$;
if $(W'(i) > W'_{\text{out}}(i))$ $S'(i) = S'_{\text{in}}(i)$;
else $S'(i) = S'_{\text{out}}(i)$;
}
for $\{v_i \in S'(n)\}$
{
$l_{S^*}(v_i) = W'(n) - \max(\max_{k \in \overline{j'_i+1, j''_i-1} \setminus \{i\}} (W'_{\text{in}}(k) + W''(j''_k)), W'(j'_i) + W''(j''_i))$;
}
for $(v_i \in S'(n))$ $\{u_{S^*}(v_i) = W'(n) - W'_{\text{in}}(i) - W''(j''_i)\}$;

We store the sets $S'(i)$, $S'_{\text{in}}(i)$, $S'_{\text{out}}(i)$ in simply connected lists; this ensures the possibility of joining such lists in time $O(1)$. It is easy to show that, when this storage strategy is used, the computational complexity of the pseudocode presented above is $O(|V(G)| + |E(G)|)$.

ACKNOWLEDGMENTS

This work was supported by the Laboratory of Algorithms and Technologies for Networks Analysis, Higher School of Economics (National Research University), contract no. 11.G34.31.0057.

REFERENCES

1. V. Boiko, B. Gol'dengorin, and V. Kuz'menko, in *The Theory of Optimal Solutions* (Nats. Akad. Nauk Ukraini, Inst. Kibernetiki im. V.M. Glushkova, Kiev, 2006), Vol. 5, pp. 98–104 [in Ukranian].
2. B. I. Goldengorin, D. S. Malyshev, and P. M. Pardalos, Dokl. Math. **87**, 368–371 (2013).
3. V. K. Leont'ev and E. N. Gordeev, Kibernetika, No. 5, 82–89 (1986).
4. V. Boginski, C. Commander, P. Pardalos, and Y. Ye, *Sensors: Theory, Algorithms, and Applications* (Springer-Verlag, Berlin, 2012).
5. H. Booth and J. Westbrook, Algorithmica **11**, 341–352 (1994).
6. R. Germs, B. Goldengorin, and M. Turkensteen, Comput. Oper. Res. **39**, 291–298 (2012).
7. B. Goldengorin, D. Malyshev, P. Pardalos, and V. Zamaraev, J. Combin. Opt. (2013).
8. M. Habib, R. McConnell, C. Paul, and L. Viennot, Theor. Comput. Sci. **234**, 59–84 (2000).
9. J. Orlin, in *Proceedings of Symposium on Theory of Computing, Palo ALto, Calif., U.S.A., 2013* (Palo ALto, 2013), pp. 765–774.
10. M. Libura, Discrete Appl. Math. **30**, 197–211 (1991).
11. R. Ramaswamy, J. Orlin, and N. Chakravarti, Math. Program. **102**, 355–369 (2005).
12. N. Reinfeld and W. Vogel, *Mathematical Programming* (Prentice-Hall, Englewood Cliffs, N. J., 1958).
13. D. Shier and C. Witzgall, Networks **10**, 277–291 (1980).
14. R. Tarjan, Inform. Process. Lett. **14**, 30–33 (1982).
15. A. Volgenant, Europ. J. Oper. Res. **169**, 338–339 (2006).

Translated by O. Sipacheva