ISSN 1064-5624, Doklady Mathematics, 2013, Vol. 87, No. 3, pp. 368–371. © Pleiades Publishing, Ltd., 2013.
Original Russian Text © B.I. Gol'dengorin, D.S. Malyshev, P.M. Pardalos, 2013, published in Doklady Akademii Nauk, 2013, Vol. 450, No. 4, pp. 393–396.

COMPUTER SCIENCE

Efficient Computation of Tolerances in the Weighted Independent Set Problem for Trees

B. I. Gol'dengorin, D. S. Malyshev, and P. M. Pardalos

Presented by Academician Yu.G. Evtushenko December 19, 2012

Received December 19, 2012

DOI: 10.1134/S1064562413030186

After an optimal solution of the combinatorial optimization problem (COP) is obtained, the next natural step is to analyze its sensitivity. The purpose of the sensitivity analysis of an optimal solution of the COP is to determine the dependence of this solution on the variation of the initial data. There are several reasons for the interest in sensitivity analysis. First, in many cases, the initial data of the COP are specified inaccurately or involve natural uncertainty. In such cases, sensitivity analysis is necessary for determining confidence in both the optimal solution itself and conclusions based on this solution. Secondly, formally modeling essential properties of the required optimal solution in terms of the COP is often hard to formalize. In this situation, having an optimal solution of a simplified COP, a decision maker is interested in the extent to which the found optimal solution possesses properties not taken into account in constructing the mathematical model for the simplified problem. The purpose of studying such perturbations is to determine the so-called tolerances, which are defined as the maximum variations of individual prices (weight, time, etc.) preserving the optimality of the given solution provided that the other COP data remain unchanged.

The tolerances are of interest because the minimum tolerance of elements of an optimal solution of the problem is a lower bound for the stability radius of this optimal solution and serves as a basis on which search algorithms for solving various COPs are constructed [6]. Exceeding the maximum tolerance value ensures the instability of the given optimal solution. The first implicit algorithmic application of tolerances appeared in Vogel's method [8] for finding the solution closest to an optimal basis solution in the simplex-method for solving the transport problem and in [2] in the construction of an efficient heuristics for solving the three-index assignment problem. Tolerances have been successfully used to improve algorithms for solving not only "hard-to-solve" problems but also problems having efficient solutions, e.g., the problem about covering a graph by cycles of optimal total weight without common vertices, which is known as the linear assignment problem [1], as well as many variations and generalizations of this problem [3]. Special attention is given to effectively solvable classes of COPs, for which an optimal solution and all tolerances can be computed simultaneously. Examples of such problems are the problems of finding a minimum-weight skeleton tree [10] and a shortest path [7, 9] and the assignment problem [12].

This paper is devoted to an efficient determination of all tolerances in the weighted independent set problem (WISP) for trees on the basis of a formula for tolerances obtained in the paper and dynamic programming. Thus, for a tree with n vertices, the proposed procedure uses $O(n)$ arithmetic operations on real numbers of any length. The choice of trees as the object of study is caused by that the WISP class has low complexity and, therefore, should be useful in the development of search algorithms of branch-and-bound type for solving WISP in the general case (see, e.g., the corresponding choice for the traveling salesman problem in [5, 11]).

Laboratory of Algorithms and Technologies for Analysis of Network Structures, Higher School of Economics (National Research University) in Nizhni Novgorod, ul. Rodionova 136, Nizhni Novgorod, 603093 Russia
Department of Applied Mathematics and Computer Science, Higher School of Economics (National Research University) in Nizhni Novgorod, Bol'shaya Pecherskaya ul. 25/12, Nizhni Novgorod, 603155 Russia
Center of Applied Optimization, University of Florida, Gainesville, Florida, United States
e-mail: bgoldengorin@hse.ru, dmalishev@hse.ru, p.m.pardalos@gmail.com

1. COMBINATORIAL OPTIMIZATION PROBLEM

The combinatorial optimization problem is determined by a quadruple $\{\Gamma, c, F, f_c\}$ of parameters, where c: $\Gamma \to R$ is a weight function on a universal set Γ, F is the set of admissible solutions of the problem, and f_c is an objective function. Given a particular quadruple $\{\Gamma, c, F, f_c\}$, the corresponding COP is to find an element $S^* \in F$ at which the objective function takes an extremum (minimum or maximum) value. Such a set S^* is called an optimal solution of the problem.

An example of such a COP is the weighted independent set problem. This problem for an ordinary graph with weighted vertices consists in fining a set of pairwise nonadjacent vertices of maximum weight in this graph. Recall that any set of pairwise nonadjacent vertices is usually said to be independent. In the weighted independent set problem, Γ is the vertex set of the input graph, F is the family of independent sets in the input graph, c is a nonnegative weight function of vertices, and $f_c(S)$ is the weight of an independent set S, which equals the sum $\sum_{s \in S} c(s)$.

Suppose given a maximum COP, its optimal solution S^*, and elements x and y. We shall refer to this problem as the old problem. From the old problem we form two new problems. The first new problem is obtained from the old one by decreasing the weight of the element x by a nonnegative number w. The second problem is obtained from the old one by increasing the weight of the element y by some nonnegative number. Consider the old problem. The lower tolerance of the element x with respect to the given optimal solution S^* (denoted by $l_{S^*}(x)$) is defined as the supremum of those numbers w for which the set S^* remains an optimal solution for the first new problem, provided that all other data of the old problem remain unchanged. The meaning of the lower tolerance is that its minimum value over all elements of the optimal solution is an estimate of the stability radius of the optimal solution of the old problem. In other words, this problem has no admissible solution S' satisfying the inequalities $f_c(S^*) > f_c(S') > f_c(S^*) - l$, where l is the minimum lower tolerance among the lower tolerances of all elements of S^*. The notion of an upper tolerance is introduced by analogy with the notion of a lower tolerance. 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 for which the set S^* remains an optimal solution for the second new problem, provided that all other data of the old problem remain unchanged. The meaning of the upper tolerance is similar to that of the lower tolerance. Namely, the old problem has no admissible solution S'' satisfying the inequality $f_c(S^*) > f_c(S'') > f_c(S^*) - u$, where u is the minimum tolerance of elements not belonging to S^*.

2. A FORMULA FOR CALCULATING TOLERANCES IN THE WEIGHTED INDEPENDENT SET PROBLEM

In what follows, we always assume that $\{\Gamma, c, F, f_c\}$ is the weighted independent set problem. Let $F_-(x)$ denote the family of independent sets not containing the vertex x. By $F_+(y)$ we denote the family of independent sets containing the vertex y. The family of independent sets of maximum weight among the elements of $F_-(x)$ and $F_+(y)$ is denoted by $F_-^*(x)$ and $F_+^*(y)$, respectively. The set F^* is the family of all optimal solutions of the problem $\{\Gamma, c, F, f_c\}$. For a subset $F' \subseteq F$, by $f_c(F')$ we denote the maximum weight of an independent set belonging to F'. The following lemma is valid.

Lemma 1. *If $S^* \in F^*$, $x \in S^*$, and $y \notin S^*$, then*

(a) $l_{S^*}(x) = f_c(F^*) - f_c(F_-^*(x))$;

(b) $u_{S^*}(y) = f_c(F^*) - f_c(F_+^*(y))$.

It follows from Lemma 1 that the lower and upper tolerances are invariants of the set of optimal solutions in the sense that their values do not depend on the chosen optimal solution.

3. CALCULATION OF TOLERANCES IN THE WEIGHTED INDEPENDENT SET PROBLEM FOR TREES

According to lemma 1, for any $S^* \in F^*$, $x \in S^*$, and $y \notin S^*$, we have $l_{S^*}(x) = f_c(F^*) - f_c(F_-^*(x))$ and $u_{S^*}(y) = f_c(F^*) - f_c(F_+^*(y))$. Thus, if the value $f_c(F^*)$ is known, then, to calculate the lower and upper tolerances of the vertices x and y, it suffices to calculate $f_c(F_-^*(x))$ and $f_c(F_+^*(y))$. At the same time, it is desirable to efficiently use the information accumulated during the calculation of $f_c(F^*)$ for finding $f_c(F_-^*(x))$ and $f_c(F_+^*(y))$ (rather than, say, recalculate the optimal independent set for the graph without or with x or y). For trees, such a procedure is indeed possible. It is based on the application of the direct and the inverse "move" of dynamic programming.

The WISP for a tree is solved by considering this problem for subtrees rooted at all possible vertices and reducing the problem for a subtree rooted at the vertex x to the WISP for subtrees rooted at the sons of x. This approach was proposed in [4]. For a COP $\{\Gamma, c, F, f_c\}$ being the WISP for a tree T rooted at a vertex r, consider all subtrees T_x, $x \in V(T)$, where T_x denotes the subtree of T rooted at x. Let us introduce the following variables: $s(x)$ is an optimal independent subset of the subtree T_x and $s_{\text{in}}(x)$ and $s_{\text{out}}(x)$ are, respectively, independent subsets of T_x having maximum weight among all independent sets of T_x containing and not containing the vertex x. The sets $s(x)$, $s_{\text{in}}(x)$, and $s_{\text{out}}(x)$ have weights $w(x)$, $w_{\text{in}}(x)$, and $w_{\text{out}}(x)$, respectively. Note that $w(x) = \max(w_{\text{in}}(x), w_{\text{out}}(x))$; moreover, $s(x) = s_{\text{in}}(x)$ if $w_{\text{in}}(x) \ge w_{\text{out}}(x)$ and $s(x) = s_{\text{out}}(x)$ otherwise. It is easy to write recursive equations for weights: if x is a leaf of the tree, then $w_{\text{in}}(x) = c(x)$ and $w_{\text{out}}(x) = 0$, and if the vertex x is not a leaf, then $w_{\text{out}}(x) = \sum\limits_{y \in \cosh(x)} w(x)$ and $w_{\text{in}}(x) = c(x) + \sum\limits_{y \in \cosh(x)} w_{\text{out}}(y)$, where $\cosh(x)$ denotes the set of immediate descendants of the vertex x in the input tree. By analogy, recursive equations for the sets $s(x)$, $s_{\text{in}}(x)$, and $s_{\text{out}}(x)$ are derived.

After obtaining $w_{\text{in}}(x)$, $w_{\text{out}}(x)$, $w(x)$, $s_{\text{in}}(x)$, $s_{\text{out}}(x)$, and $s(x)$ for each $x \in V(T)$ (by a direct "move" of dynamic programming), we calculate all tolerances by means of an inverse "move." For this purpose, we introduce the following variables: $W_{\text{in}}(x) = f_c(F_+^*(x))$, $W_{\text{out}}(x) = f_c(F_-^*(x))$, $S_{\text{in}}(x) \in F_+^*(x)$, and $S_{\text{out}}(x) \in F_-^*(x)$. Clearly, if x is a root of T, then $W_{\text{in}}(x) = w_{\text{in}}(r)$ and $W_{\text{out}}(x) = w_{\text{out}}(r)$. Note that if $p(x)$ is a parent of a nonroot vertex x, then $W_{\text{out}}(p(x)) - w(x) = W_{\text{in}}(x) - w_{\text{in}}(x)$, and $W_{\text{out}}(x) - w_{\text{out}}(x) = \max(W_{\text{in}}(p(x)) - w_{\text{out}}(x), W_{\text{out}}(p(x)) - w(x))$. These equalities are easy to rewrite as recursive equations. By analogy, equalities for $S_{\text{in}}(x)$ and $S_{\text{out}}(x)$ are written. It follows from Lemma 1 that, for any $x \in S^* = s(r)$, we have $l_{S^*}(x) = w(r) - W_{\text{out}}(x)$ and, for any $y \notin S^*$, we have $u_{S^*}(y) = w(r) - W_{\text{in}}(y)$.

The pseudocode for the solution of the WISP for a tree T with root r and the subsequent calculation of all tolerances is given below.

4. THE WISP-ALGORITHM + CALCULATION OF ALL TOLERANCES (T, r)

{
for (in the order reverse to the width-first traversal of T)
{ **if** (x is a leaf)
{
$w_{\text{out}}(x) = 0$; $s_{\text{out}}(x) = \phi$;
$w_{\text{in}}(x) = c(x)$; $s_{\text{in}}(x) = \{x\}$;
$w(x) = c(x)$; $s(x) = \{x\}$;
}
else
{
$w_{\text{in}}(x) = \sum\limits_{y \in \cosh(x)} w_{\text{out}}(y) + c(x)$; $s_{\text{in}}(x) = \bigcup\limits_{y \in \cosh(x)} s_{\text{out}}(y) \cup \{x\}$;
$w_{\text{out}}(x) = \sum\limits_{y \in \cosh(x)} w(y)$; $s_{\text{out}}(x) = \bigcup\limits_{y \in \cosh(x)} s(y)$;
$w(x) = \max(w_{\text{in}}(x), w_{\text{out}}(x))$;
if ($w(x) > w_{\text{out}}(x)$) $s(x) = s_{\text{in}}(x)$;
else $s(x) = s_{\text{out}}(x)$;
}
}
for (in the order of the width-first traversal of the tree T)
{
if ($x = r$)
{
$W_{\text{out}}(x) = w_{\text{out}}(r)$; $S_{\text{out}}(x) = s_{\text{out}}(r)$;
$W_{\text{in}}(x) = w_{\text{in}}(r)$; $S_{\text{in}}(x) = s_{\text{in}}(r)$;
}
else
{
$W_{\text{in}}(x) = W_{\text{out}}(p(x)) - w(x) + w_{\text{in}}(x)$;
$S_{\text{in}}(x) = S_{\text{out}}(p(x)) \cup (s_{\text{in}}(x) \backslash s(x))$;
$W_{\text{out}}(x) = w_{\text{out}}(x) + \max(W_{\text{in}}(p(x)) - w_{\text{out}}(x), W_{\text{out}}(p(x)) - w(x))$;
if ($W_{\text{out}}(x) - w_{\text{out}}(x) \ge W_{\text{out}}(p(x)) - w(x)$) $S_{\text{out}}(x) = S_{\text{in}}(p(x))$;
else $S_{\text{out}}(x) = S_{\text{out}}(p(x)) \cup (s_{\text{out}}(x) \backslash s(x))$;
}
}
for ($x \in V(T)$)
{
if ($x \in s(r)$) $l_{S^*}(x) = w(r) - W_{\text{out}}(x)$;
else $u_{S^*}(x) = w(r) - W_{\text{in}}(x)$;
}
}

Note that, by means of the pseudocode presented above, the value $w(r)$ and all tolerances (without the corresponding sets) are calculated in time $O(n)$, where n is the number of vertices in the tree T. However, if the sets $s(x)$, $s_{\text{in}}(x)$, and $s_{\text{out}}(x)$ are stored in a singly or doubly connected list (which ensures the possibility of implementing union in time $O(1)$), then $s_{\text{in}}(x)$ and $s_{\text{out}}(x)$ are calculated in time $O(|\cosh(x)|)$. Therefore, an optimal independent set for the tree T is determined in time $O\Big(\sum\limits_{x \in V(T)} |\cosh(x)|\Big) = O(n)$.

It can also be shown that the total time required to compute all sets $s_{\text{in}}(x) \backslash s(x)$ and $s_{\text{out}}(x) \backslash s(x)$ is $O(n)$ (for this purpose, it is necessary to write recursive formulas for $s_{\text{out}}(x) \backslash s(x)$ and $s(x) \backslash s_{\text{out}}(x)$, store these sets in singly or doubly connected lists, and determine them in the first part of the pseudocode). Thus, the complexity of the entire pseudocode is $O(n)$.

ACKNOWLEDGMENTS

This work was supported by the Laboratory of Algorithms and Technologies for Analyzing Network Structures, Higher School of Economics (state contract no. 11.G34.31.0057). Gol'dengorin and Malyshev acknowledge the support of the Science Foundation of Higher School of Economics under the collective research program “Teacher–Students” (project no. 11-04-0008).

REFERENCES

1. V. Boiko, B. Gol’dengorin, and V. V. Kuz’menko, in *Theory of Optimal Solutions* (Nats. Akad. Nauk Ukrainy, Inst. Kibernetiki im. V.M. Glushkova, Kiev, 2006), Vol. 5, pp. 98–104 [in Russian].
2. E. Balas and M. J. Saltzman, Oper. Res. **39**, 150–161 (1991).
3. H. Bekker, E. Braad, and B. Goldengorin, Lect. Notes Comput. Sci. **3503**, 610–613 (2005).
4. C. Chen, M. Kuo, and J. Sheu, BIT Numer. Math. **28**, 353–356 (1988).
5. R. Germs, B. Goldengorin, and M. Turkensteen, Comput. Oper. Res. **39** (2), 291–298 (2012).
6. B. Goldengorin, G. Jager, and P. Molitor, J. Computer Sci. **2** (9), 716–734 (2006).
7. R. Ramaswamy, J. Orlin, and N. Chakravarti, Math. Program. **102**, 355–369 (2005).
8. N. V. Reinfeld and W. R. Vogel, *Mathematical Programming* (Prentice-Hall, Englewood Cliffs, N.J., 1958).
9. D. R. Shier and C. Witzgall, Networks **10** (4), 277–291 (1980).
10. R. Tarjan, Inform. Process. Lett. **14**, 30–33 (1982).
11. M. Turkensteen, D. Ghosh, B. Goldengorin, and G. Sierksma, Eur. J. Oper. Res. **189**, 775–788 (2008).
12. A. Volgenant, Eur. J. Oper. Res. **169**, 338–339 (2006).

Translated by O. Sipacheva