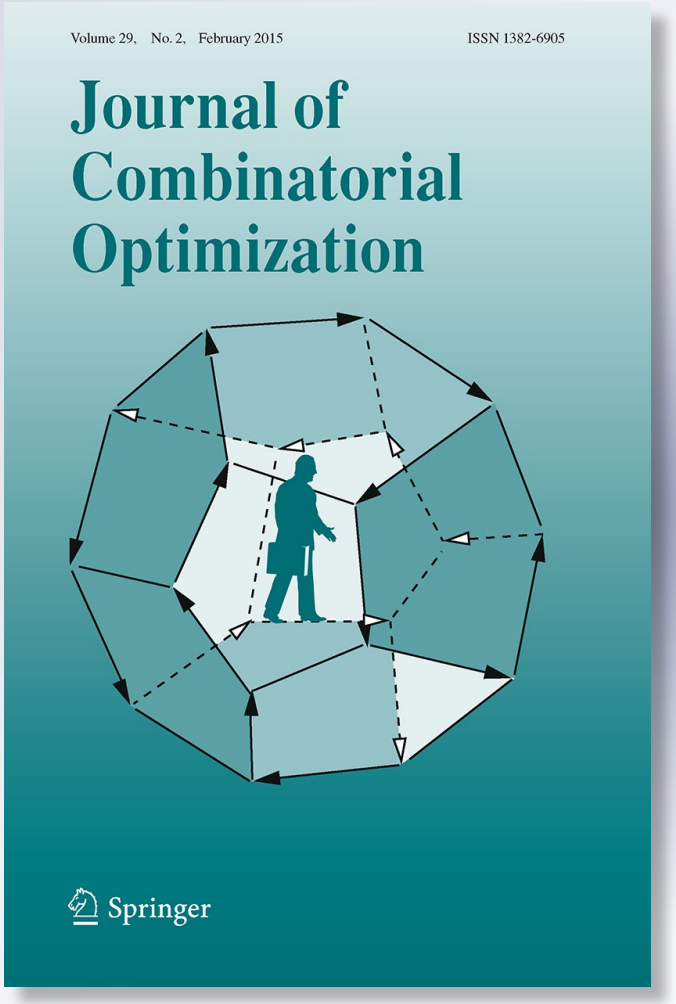
A tolerance-based heuristic approach for the weighted independent set problem

B. I. Goldengorin, D. S. Malyshev, P. M. Pardalos & V. A. Zamaraev

Journal of Combinatorial Optimization

ISSN 1382-6905
Volume 29
Number 2

J Comb Optim (2015) 29:433-450
DOI 10.1007/s10878-013-9606-z



Springer

J Comb Optim (2015) 29:433–450
DOI 10.1007/s10878-013-9606-z

A tolerance-based heuristic approach for the weighted independent set problem

B. I. Goldengorin · D. S. Malyshev · P. M. Pardalos · V. A. Zamaraev

Published online: 26 March 2013

Abstract The notion of a tolerance is a helpful tool for designing approximation and exact algorithms for solving combinatorial optimization problems. In this paper we suggest a tolerance-based polynomial heuristic algorithm for the weighted independent set problem. Several computational experiments show that our heuristics works very well on graphs of a small density.

Keywords Combinatorial optimization · Independent set problem · Heuristics · Notion of a tolerance

B. I. Goldengorin · D. S. Malyshev · P. M. Pardalos · V. A. Zamaraev
Laboratory of Algorithms and Technologies for Network Analysis, National Research University Higher School of Economics, 136 Rodionova str., Nizhny Novgorod 603093 , Russia

B. I. Goldengorin
Department of Mathematics, Subdepartment of Higher Mathematics, National Research University Higher School of Economics, 26 Shabolovka str., Moscow 119049, Russia

D. S. Malyshev · V. A. Zamaraev
Department of Applied Mathematics and Informatics, National Research University Higher School of Economics, 25/12 Bolshaya Pecherskaya str., Nizhny Novgorod 603155, Russia

D. S. Malyshev (✉) · V. A. Zamaraev
Department of Mathematical Logic and Higher Algebra, Lobachevsky State University of Nizhny Novgorod, 23 Gagarina av., Nizhny Novgorod 603950, Russia
e-mail: dsmalyshev@rambler.ru

P. M. Pardalos
Department of Industrial and System Engineering, Faculty of Engineering,
Center for Applied Optimization, University of Florida, 401 Weil Hall,
P.O. Box 116595, Gainesville, FL 32611-6595, USA

1 Introduction

After obtaining an optimal solution of a combinatorial optimization problem (COP) the next natural step may be the sensitivity analysis of the solution (Gal and Greenberg 1997). The purpose of the sensitivity analysis of a given optimal solution is to find a dependence of the solution on initial data (Greenberg 1998). There are several reasons for an interest in the sensitivity analysis. Firstly, in many cases initial data of a COP are inaccurate or have a natural uncertainty. In this case, the sensitivity analysis is needed to determine confidence both to an optimal solution and to conclusions based on it. Secondly, it may be difficult to model important characteristics of the desired optimal solution in terms of a COP. In this case, one may be interested in how the solution satisfies the properties unaccounted in a COP. Variations of exactly one element of an optimal solution are considered in the simplest sensitivity analysis. The aim of study of such perturbations is to find tolerances, which are defined as the maximal variations of a parameter (weight, cost, time, etc.) that preserve the optimality of a given solution providing the remaining data of a COP are unchanged.

The interest to tolerances is due to the fact that minimal among tolerances of elements of an optimal solution is a lower bound for the stability radius of the solution (Sotskov et al. 1995) and it serves as the basis for designing exact and approximation algorithms to solve different COPs (Goldengorin et al. 2006). Tolerances may be used for an effective enumeration of k best solutions of a COP (Murty 1968; Van der Poort et al. 1999).

For the first time, tolerances were implicitly used in the Vogel's method (Reinfeld and Vogel 1965) for finding the closest solution to an optimal basic one in the simplex method. Also, tolerances were used in Balas and Saltzman (1991) under the term "maximum regret" (max-regret) for constructing an effective heuristics for the three-index assignment problem. Tolerances are used in the Helsgaun's improvement of the Lin-Kernighan's heuristics (Helsgaun 2000, 2009) for finding suboptimal solutions of the traveling salesman problem for simple graphs, in exact and approximation algorithms for the traveling salesman problem for digraphs and its variations (Ernst et al. 2010; Germs et al. 2012; Ghosh et al. 2007; Jager and Molitor 2008; Turkensteen et al. 2008). Tolerances have been successfully used to improve algorithms not only for intractable problems, but also for effectively (polynomially) solvable ones, such as the problem of covering vertices of a graph by disjoint cycles of minimum (maximum) total weight, also known as the linear assignment problem (Boyko et al. 2006) as well as for its many variations and generalizations (Bekker et al. 2005). A special attention is paid to polynomially solvable classes of COPs, for which an optimal solution and all tolerances can be computed simultaneously. Among these problems are the minimum weight spanning tree problem (Tarjan 1982), the shortest path problem (Ramaswamy et al. 2005; Shier and Witzgall 1980) and the assignment problem (Volgenant 2006).

In the present article we develop polynomial-time heuristic for the weighted independent set problem based on the concept of a tolerance and experimentally investigate its efficiency. The paper is organized as follows. In Sect. 1 we prove some auxiliary results about tolerances. In Sect. 2 we show that all tolerances for the weighted independent set problem for trees can be computed in linear time in the number of vertices. In Sect. 3 we describe the proposed heuristics. Finally, in Sect. 4 we present an

experimental study of our heuristics, which shows that for graphs of a low density the heuristic give solutions that are very close to optimal.

2 Definitions and notation

A *combinatorial optimization problem* (COP, for short) is defined by a quadruple of parameters $\{\Gamma, c, F, f_c\}$, where $c : \Gamma \to R$ is the *cost* (*weight*) *function* of elements of the *ground set* Γ; $F \subseteq 2^{\Gamma}$ is the *set of feasible solutions*; f_c is the *objective function*, defined on F. For a given concrete (individual) quadruple the corresponding COP is to determine an element $S^* \in F$ with the extremal (minimum or maximum) value of the objective function. The set S^* is said to be an *optimal solution* of the problem. A COP $\{\Gamma, c, F, f_c\}$ is an *additive problem*, if for any $S \in F$ we have $f_c(S) = \sum_{s \in S} c(s)$.

There are many COPs, for which tolerances appeared useful not only for the stability radius analysis (Sotskov et al. 1995), but also for development of new (exact and approximation) algorithms for their solution. For example, the traveling salesman problem is such a problem, i.e. to find a *hamiltonian cycle* (a circuit, that visits each vertex exactly once) of a given graph with positive lengths of edges, which has the minimal sum of lengths of its involved edges. For this problem Γ is the set of edges of an instance graph; F is the family of all sets of edges, constituting a hamiltonian cycle; c is a positive length function (a passing cost function) of edges; $f_c(S)$ is the length of a hamiltonian cycle $S \in F$, equal to the sum $\sum_{e \in S} c(e)$.

Another example of a COP is the *weighted independent set problem* (WIS, for short). This problem is to find a subset of pair-wise nonadjacent vertices of the maximum total weight in a graph with positive weights of vertices. Let us recall that any such a subset (i.e., a subset of pair-wise nonadjacent vertices) is referred to as *independent*. For this problem Γ is the set of vertices of a given graph; F is the set of its independent sets; c is a positive weight function; $f_c(S)$ is the weight of an independent set S, equal to $\sum_{s \in S} c(s)$.

Let us consider an additive COP of the max $-type$ (i.e., a maximization problem), let S^* be an optimal solution, $x \in S^*$ and $y \notin S^*$ (i.e., $y \in \Gamma \setminus S^*$). We call this problem *old*. We will create two *new* problems on the basis of the old one. The first of them is obtained from the old problem by a reduction of $c(x)$ by a nonnegative number. The second one is obtained from the old by increasing $c(y)$ by some nonnegative number. Let us return to the old problem. The *lower tolerance of x with respect to S^** (denoted by $l_{S^*}(x)$) is the supremum of those nonnegative numbers w, by which $c(x)$ can be decreased such that S^* is still an optimal solution for the first new problem under keeping all other data of the old problem unchanged. The *upper tolerance of y with respect to S^** (denoted by $u_{S^*}(y)$) is the supremum of those nonnegative numbers w, by which $c(y)$ can be increased such that S^* remains an optimal solution for the second new problem (by keeping all other data of the old COP unchanged). A significance of the lower tolerance of x is that it is an estimation of the stability radius for the optimal solution S^* of the old problem within its feasible solutions, containing x. In other words, there is no feasible solution S' of the old COP, containing x, such that

the inequality $f_c(S^*) > f_c(S^{'}) > f_c(S^*) - l_{S^*}(x)$ holds. A significance of an upper tolerance is analogous to a lower tolerance. Namely, there is no a feasible solution S'' of the old COP, such that $y \notin S^*$ and the inequality $f_c(S^*) > f_c(S^{''}) > f_c(S^*) - u_{S^*}(y)$ is true. It implies that $l = \min\limits_{x \in S^*} l_{S^*}(x)$ and $u = \min\limits_{y \notin S^*} u_{S^*}(y)$ are estimations for the stability radius of S^* among *all* feasible solutions of the old COP. Indeed, there is no solution $S^{'''} \in F$, such that $f_c(S^*) > f_c(S^{'''}) > f_c(S^*) - l$ and $f_c(S^*) > f_c(S^{'''}) > f_c(S^*) - u$ simultaneously hold.

In the above definitions of tolerances the supremums are attainable, so the word "supremum" can be replaced by the word "maximum". This is due to the finiteness of F.

3 A formula for tolerances with respect to the WIS

In what follows we will assume that $\{\Gamma, c, F, f_c\}$ is the WIS. By $F_-(x)$ we will denote the set of all independent sets, which do not contain a vertex x. The set of all independent sets, containing a vertex y, will be denoted by $F_+(y)$. The families of all independent sets, having the largest weight among elements of $F_-(x)$ and $F_+(y)$, will be denoted by $F^*_-(x)$ and $F^*_+(y)$, respectively. The set F^* contains all optimal solutions of the considering COP. For a subset $F' \subseteq F$ we denote by $f_c(F')$ the largest weight among weights of independent sets, belonging to F'.

Lemma 1 *Let $S^* \in F^*$, $x \in S^*$ and $y \notin S^*$. Then:*

(a) $l_{S^*}(x) = f_c(F^*) - f_c(F^*_-(x))$,
(b) $u_{S^*}(y) = f_c(F^*) - f_c(F^*_+(y))$.

Proof We will prove only (a), since (b) can be proved similarly. If $c(x)$ is decreased and weights of all other vertices remain unchanged, then the weights of all independent sets, containing the vertex x, are decreased by the same value as the weight of x. The weights of all independent sets, which are not containing x, do not change. Hence, S^* remains an optimal solution until a subtrahend for the x's weight will be no more than the difference between $f_c(F^*)$ and an optimal value of f_c on $F_-(x)$ (i.e., $f_c(F^*_-(x))$). Therefore, $l_{S^*}(x) = f_c(F^*) - f_c(F^*_-(x))$.

Remark 1 The lower and upper tolerances are invariants of optimal solutions in the sense that their values are independent of a chosen optimal solution, if there are at least two such solutions. This is an immediate consequence of Lemma 1.

Remark 2 An optimal independent set S^* is the unique optimal solution if and only if all lower tolerances with respect to S^* are positive and all upper tolerances with respect to S^* are also positive. The necessity is a trivial corollary of Lemma 1 and the sufficiency is easy to prove by contradiction.

Remark 3 Assume a graph has at least two optimal independent sets. The lower tolerances of elements in the nonempty intersection of all optimal solutions are strictly positive and they are equal to zeros for elements outside this intersection, if such exist. On the other hand, the upper tolerances of vertices in the union of all optimal solutions are equal to zeros and they are strictly positive for vertices outside this union, if such

exists. In order to show this, let $S' = \bigcap_{S \in F^*} S$ and $S'' = \bigcup_{S \in F^*} S$. For any $y_1 \in S''$ and for any $y_2 \in \Gamma \setminus S''$ we have $f_c(F^*) = f_c(F^*_+(y_1))$ and $f_c(F^*) > f_c(F^*_+(y_2))$. Hence, by Lemma 1, for any optimal solution S^*_1, which contains y_1, we have $u_{S^*_1}(y_1) = 0$ and $u_{S^*_1}(y_2) > 0$. For any $x_1 \in S'$ and for any $x_2 \in \Gamma \setminus S'$ we have $f_c(F^*) > f_c(F^*_-(x_1))$ and $f_c(F^*) = f_c(F^*_-(x_2))$. Therefore, by Lemma 1, for any optimal solution S^*_2, which contains x_2, we have $l_{S^*_2}(x_1) > 0$ and $l_{S^*_2}(x_2) = 0$.

Remark 4 Remark 2 is useful for formulations of rules for branching in branch-and-bound algorithms. Indeed, branchings on elements with strictly positive lower tolerances protect from a possibility of branchings on elements outside an optimal solution, and, therefore, reduces a chance of useless walks in finding a global optimum.

4 Linear-time algorithm for computing tolerances for the WIS on trees

By Lemma 1, for any $S^* \in F^*, x \in S^*$ and $y \notin S^*$ the relations $l_{S^*}(x) = f_c(F^*) - f_c(F^*_-(x))$ and $u_{S^*}(y) = f_c(F^*) - f_c(F^*_+(y))$ hold. Therefore, when $f_c(F^*)$ is known, then for a computation of the upper and the lower tolerances of x and y we need only to compute $f_c(F^*_-(x))$ and $f_c(F^*_+(y))$. At the same time, information accumulated during the computation of $f_c(F^*)$, would be desirable to use effectively for determining $f_c(F^*_-(x))$ and $f_c(F^*_+(y))$ (and, say, not to recompute optimal independent sets for the graphs without/with x/y). For trees such a procedure is really possible. It is based on the application of breadth-first and reverse-back orders in dynamic programming.

The WIST (i.e., the WIS for trees) can be solved by considering the problem for subtrees (of a given tree) with roots at all vertices and by reducing the problem for a subtree with root x to the WIST for subtrees with roots at children of x (Chen et al. 1988). Let $\{\Gamma, c, F, f_c\}$ be the weighted independent set problem for a tree T with root at a vertex $root$. Let $x \in V(T)$ and T_x be the subtree of T with root at x. In order to describe the reverse-back part we introduce the following variables: $set(x)$ is an optimal independent set of T_x; $set_{in}(x)$ and $set_{out}(x)$ are independent sets of T_x with the largest weight among all independent sets of T_x which contain/do not contain x, respectively. The sets $set(x), set_{in}(x), set_{out}(x)$ have weights $weight(x), weight_{in}(x), weight_{out}(x)$, correspondingly. Note that

$$\begin{aligned} weight(x) &= \max(weight_{in}(x), weight_{out}(x)), \\ set(x) &= set_{in}(x), \quad \text{if } weight_{in}(x) \geq weight_{out}(x), \quad \text{and } set(x) = set_{out}(x), \\ &\quad \text{otherwise.} \end{aligned}$$

Now we can easily write the recurrence relations for $weight_{in}(x)$ and $weight_{out}(x)$: if x is a leaf, then

$$weight_{in}(x) = c(x), \quad weight_{out}(x) = 0,$$

if x is not a leaf, then

$$weight_{out}(x) = \sum_{y \in children(x)} weight(y),$$
$$weight_{in}(x) = c(x) + \sum_{y \in children(x)} weight_{out}(y),$$

where $children(x)$ is the set of all immediate descendants of x in the initial tree. By analogy, we have

$$set_{in}(x) = \bigcup_{y \in children(x)} set_{out}(y) \cup \{x\},$$
$$set_{out}(x) = \bigcup_{y \in children(x)} set(y).$$

Once we know $weight_{in}(x)$, $weight_{out}(x)$, $weight(x)$, $set_{in}(x)$, $set_{out}(x)$, $set(x)$ for every $x \in V(T)$, we can compute all tolerances by means of the breadth-first part of dynamic programming. To this end we introduce the following variables: $Weight_{in}(x) = f_c(F^*_+(x))$, $Weight_{out}(x) = f_c(F^*_-(x))$. Clearly, if x is the root of T, then $Weight_{in}(x) = weight_{in}(root)$, and $Weight_{out}(x) = weight_{out}(root)$. Note that if $parent(x)$ is the parent of a nonroot vertex x, then

$$\begin{aligned} Weight_{out}(parent(x)) - weight(x) &= Weight_{in}(x) - weight_{in}(x), \\ Weight_{out}(x) - weight_{out}(x) &= \max(Weight_{in}(parent(x)) \\ &\quad - weight_{out}(x), Weight_{out}(parent(x)) \\ &\quad - weight(x)). \end{aligned}$$

These relations can be easily rewritten in a form of recurrence relations. From Lemma 1 it follows that for any $x \in S^* = set(root)$ the relation $l_{S^*}(x) = weight(root) - Weight_{out}(x)$ holds and for any $y \notin S^*$ the equality $u_{S^*}(y) = weight(root) - Weight_{in}(y)$ is true.

An algorithm for solving the WIST and computing all tolerances is presented below.

Note that the computation of $weight(root)$ and of all tolerances (without the optimal solution $set(root)$) is carried out by means of the presented algorithm in $O(n)$ time, where n is the number of vertices in T. At the same time, if the sets $set(x)$, $set_{in}(x)$, $set_{out}(x)$ are stored in singly or doubly linked lists, then $set_{in}(x)$ and $set_{out}(x)$ can be computed in $O(|children(x)|)$ time. Therefore, an optimal independent set for T is determined in $O(\sum_{x \in V(T)} |children(x)|) = O(n)$ time and, hence, the total time of the algorithm is $O(n)$.

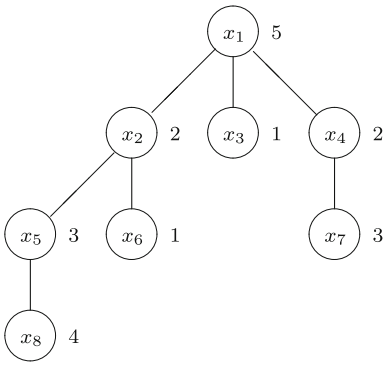
Let us demonstrate Algorithm 1 on the tree in Fig. 1. Names of vertices are inside the circles, while their weights are specified near the circles.

The breadth-first search defines the order $x_1, x_2, x_3, \ldots, x_8$ for vertices. Therefore, $x_8, x_7, x_6, \ldots, x_1$ is the reverse-back order. Let us consider the first four iterations of

Algorithm 1 WISTAndTolerances(T)

for each vertex in the reverse-back order of T **do**
  if x is a leaf **then**
    $weight_{out}(x) = 0$; $set_{out}(x) = \emptyset$;
    $weight_{in}(x) = c(x)$; $set_{in}(x) = \{x\}$;
    $weight(x) = c(x)$; $set(x) = \{x\}$;
  else
    $weight_{in}(x) = \sum_{y \in children(x)} weight_{out}(y) + c(x)$;
    $set_{in}(x) = \bigcup_{y \in children(x)} set_{out}(y) \cup \{x\}$;
    $weight_{out}(x) = \sum_{y \in children(x)} weight(y)$;
    $set_{out}(x) = \bigcup_{y \in children(x)} set(y)$;
    $weight(x) = \max(weight_{in}(x), weight_{out}(x))$;
    if $weight(x) > weight_{out}(x)$ **then**
      $set(x) = set_{in}(x)$;
    else
      $set(x) = set_{out}(x)$;
for each vertex in the breadth-first order of T **do**
  if x is a leaf **then**
    $Weight_{out}(x) = weight_{out}(x)$;
    $Weight_{in}(x) = weight_{in}(x)$;
  else
    $Weight_{in}(x) = Weight_{out}(parent(x)) - weight(x) + weight_{in}(x)$;
    $Weight_{out}(x) = weight_{out}(x) + \max(Weight_{in}(parent(x)) - weight_{out}(x), Weight_{out}(parent(x)) - weight(x))$;
for $x \in V(T)$ **do**
  if $x \in set(root)$ **then**
    $l_{S^*}(x) = weight(root) - Weight_{out}(x)$;
  else
    $u_{S^*}(x) = weight(root) - Weight_{in}(x)$;

Fig. 1 An example of the WIST



the first loop of the algorithm. Since the vertices x_8, x_7, x_6 are leaves, after the first three iterations we have

$$set_{in}(x_i) = set(x_i) = \{x_i\}, set_{out}(x_i) = \emptyset \quad \text{for any } i \in \overline{6,8},$$
$$weight_{in}(x_8) = weight(x_8) = 4, weight_{in}(x_7) = weight(x_7) = 3,$$

Table 1 The process for the computation of an optimal solution

i	$weight_{in}(x_i)$	$weight_{out}(x_i)$	$set_{in}(x_i)$	$set_{out}(x_i)$	$weight(x_i)$	$set(x_i)$
8	4	0	$\{x_8\}$	$\emptyset$	4	$\{x_8\}$
7	3	0	$\{x_7\}$	$\emptyset$	3	$\{x_7\}$
6	1	0	$\{x_6\}$	$\emptyset$	1	$\{x_6\}$
5	3	4	$\{x_5\}$	$\{x_8\}$	4	$\{x_8\}$
4	2	3	$\{x_4\}$	$\{x_7\}$	3	$\{x_7\}$
3	1	0	$\{x_3\}$	$\emptyset$	1	$\{x_3\}$
2	6	5	$\{x_2, x_8\}$	$\{x_6, x_8\}$	6	$\{x_2, x_8\}$
1	13	10	$\{x_1, x_6, x_7, x_8\}$	$\{x_2, x_3, x_7, x_8\}$	13	$\{x_1, x_6, x_7, x_8\}$

$$weight_{in}(x_6) = weight(x_6) = 1,$$
$$weight_{out}(x_8) = weight_{out}(x_7) = weight_{out}(x_6) = \emptyset.$$

At the fourth iteration we have

$$\begin{aligned}
weight_{in}(x_5) &= \sum_{y \in children(x_5)} weight_{out}(y) + c(x_5) = 0 + 3 = 3,\\
set_{in}(x_5) &= \bigcup_{y \in children(x_5)} set_{out}(y) \cup \{x_5\} = \emptyset \cup \{x_5\} = \{x_5\},\\
weight_{out}(x_5) &= \sum_{y \in children(x_5)} weight(y) = 4,\\
set_{out}(x_5) &= \bigcup_{y \in children(x_5)} set(y) = \{x_8\},\\
weight(x_5) &= \max(4, 3) = 4,
\end{aligned}$$

and as $weight_{out}(x_5)$ is equal to $weight(x_5)$, then $set(x_5) = set_{out}(x_5) = \{x_8\}$. The steps of the first loop are presented in Table 1.

After the first loop we have $S^* = \{x_1, x_6, x_7, x_8\}$ and $f_c(S^*) = 13$. $F^*_-(x)$ and $F^*_+(y)$ are computed in the second loop of the algorithm. After the first two iterations of this loop we have

$$\begin{aligned}
&Weight_{in}(x_1) = weight_{in}(x_1) = 13,\ Weight_{out}(x_1) = weight_{out}(x_1) = 10,\\
&Weight_{in}(x_2) = Weight_{out}(parent(x_2)) - weight(x_2) + weight_{in}(x_2)\\
&\qquad\qquad = 10 - 6 + 6 = 10,\\
&Weight_{out}(x_2) = weight_{out}(x_2) + \max(Weight_{in}(parent(x_2)) - weight_{out}(x_2),\\
&Weight_{out}(parent(x_2)) - weight(x_2)) = 5 + \max(13 - 5, 10 - 6) = 13.
\end{aligned}$$

The steps of the second loop are presented in Table 2.

Table 2 Results of the second loop

i	$Weight_{in}(x_i)$	$Weight_{out}(x_i)$
1	13	10
2	10	13
3	10	13
4	9	13
5	12	13
6	13	12
7	13	10
8	13	12

Finally, tolerances are computed in the third loop:

$$\begin{aligned}
l_{S^*}(x_1) &= weight(root) - Weight_{out}(x_1) = l_{S^*}(x_7)\\
&= weight(root) - Weight_{out}(x_7) = 3,\\
l_{S^*}(x_6) &= weight(root) - Weight_{out}(x_6) = l_{S^*}(x_8)\\
&= weight(root) - Weight_{out}(x_8) = 1,\\
u_{S^*}(x_2) &= weight(root) - Weight_{in}(x_2) = u_{S^*}(x_3)\\
&= weight(root) - Weight_{in}(x_3) = 3,\\
u_{S^*}(x_4) &= weight(root) - Weight_{in}(x_4) = 4,\\
u_{S^*}(x_5) &= weight(root) - Weight_{in}(x_5) = 1.
\end{aligned}$$

5 A heuristic for the weighted independent set problem

Let us recall that an independent set with the largest weight is the unique optimal solution, if and only if the lower tolerances of all its vertices are positive and the upper tolerances of all vertices outside the solution are also positive. Let us also recall that under the condition of non-uniqueness of an optimal solution the lower tolerances are positive only for the vertices, which are not in the intersection of all optimal independent sets, and the upper tolerances are positive only for the vertices that do not belong to the union of such sets. In general, if there exist at least two optimal solutions of a COP, then information only about values of all tolerances (with respect to some unknown optimal solution) allows to exclude the elements which are outside any optimal solution (and to reveal the elements belonging to every optimal solution). This fact is employed in exact and approximation algorithms for solving some COPs.

Note that tolerances might be used for an acceleration of exact enumeration algorithms (for example, branch-and-bound algorithms) for solving NP-complete problems and for a fast extraction of information about the structure of optimal and sub-optimal solutions of polynomial-time solvable problems (see Ernst et al. 2010; Germs et al. 2012; Ghosh et al. 2007; Jager and Molitor 2008; Turkensteen et al. 2008; Boyko et al. 2006; Bekker et al. 2005). An application of the latter idea appears useful, because

an intractable problem is often "replaced" by some polynomial-time relaxation and a choice of such a relaxation is determined by a "similarity" (with respect to some qualitative or quantitative measures) of optimal solutions of both problems. Similar arguments can be used for selecting a branching element. For instance, a branching can be done on an element x with the largest value of the tolerance with respect to an optimal solution S_1^* of the relaxation. The reason for the choice of such an element is that the solution S_1^* is the most sensible (in terms of values of the objective function) with respect to an elemination of x from S_1^* (if $x \in S_1^*$) or an addition of x to S_1^* (if $x \notin S_1^*$). At the same time, the similarity between S_1^* and an unknown optimal solution S_2^* of the initial problem lead to an idea that S_2^* is substantially sensitive with respect to an addition/removal of x. Thereby, one can claim with a high reliability that *either x belongs to some optimal solutions of the initial problem and its relaxation, or it does not belong to any pair of such solutions*.

One might develop heuristic algorithms for solving intractable COPs based on determining an element of a polynomial-time relaxation with the largest tolerance (with respect to an optimal solution S^* of the relaxation) and an inclusion/exclusion of this element in/from a heuristic solution of the initial problem depending on its membership of S^*. This scheme will be implemented further.

We offer the WIST as a polynomial-time relaxation of the WIS. The choice of such a relaxation is based in a variety of reasons. Firstly, any connected graph contains at least one spanning tree. Since this tree contains all vertices of the graph, then the weight of its optimal independent set is a majorant for weights of all independent sets of the graph. Secondly, an optimal solution of the WIST and values of all tolerances with respect to this solution can be calculated in linear time (due to Algorithm 1).

In order to describe the algorithm let us introduce some notations. Let H be a given graph and G be an induced subgraph of H. By $Set_h(G)$ we will denote a procedure to compute a heuristic solution for G with respect to the suggested approach. By $G \setminus \{x\}$ and $G \setminus N(x)$ we will denote the induced subgraphs obtained from G by removals of x or its neighbourhood, respectively. The mentioned algorithm is presented below.

Algorithm 2 WISHeuristics(H)

1: **return** $Set_h(H)$;

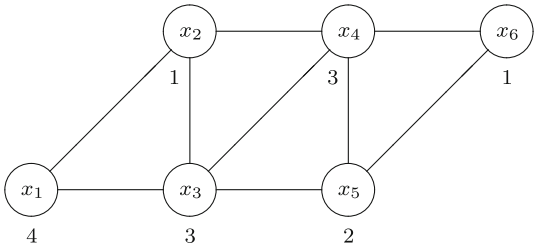
Let us estimate the computational complexity of this algorithm. Let H has n vertices and m edges. We will assume that all graphs in Algorithm 3 are represented by sorted lists of edges. Connected components of a graph can be computed in $O(m+n)$ time. Denote by $time(n, m)$ the time needed to compute a spanning tree of a connected graph with n vertices and m edges. Recall that Algorithm 1 simultaneously computes an optimal solution S^* and all tolerances with respect to this solution in $O(n)$ time. Suppose that a set of vertices returned by $Set_h(G)$ is stored in a linked list. Thus, one call of $Set_h(G)$ runs in $O(n + m + time(n, m))$ time. There are at most n calls of $Set_h(G)$, therefore the complexity of Algorithm 3 is $O(n(n + m + time(n, m)))$.

Now it is necessary to fix a spanning tree in Algorithm 3. We propose to use the maximum spanning tree of $G(V, E)$, where the weight function $l : E \longrightarrow \mathbb{R}_+$ is

Algorithm 3 $Set_h(G)$

if G is not connected **then**
  Determine all connected components $G_1, G_2, \ldots, G_k$ of the graph G;
  return $\bigcup_{i=1}^{k} Set_h(G_i)$;
else
  if G is a tree **then**
    Compute an optimal solution S^* of the WIS for G (using Algorithm 1);
    return S^*;
  else
    Find a spanning tree T of the graph G;
    Compute an optimal solution S^* of the WIST for T and values of all tolerances with respect to S^* (using Algorithm 1);
    Determine $z = \arg\max(\{l_{S^*}(x) : x \in S^*\} \cup \{u_{S^*}(y) : y \notin S^*\})$;
    if $z \in S^*$ **then**
      return $(Set_h(G \setminus \{N(z)\}) \cup \{z\})$;
    else
      return $Set_h(G \setminus \{z\})$;

Fig. 2 An example of the WIS



defined as $l(e) = \max(c(a), c(b))$. This choice is based on the intuition that vertices with larger weights are included into an exact optimal solution of the WIS for G with a higher probability. Therefore, we want to choose a spanning tree of the graph G such that neighbourhoods of vertices with large weights in the tree were similar to neighbourhoods of the same vertices in the graph. Such a similarity is provided with the Kruskal's algorithm for solving the maximum spanning tree problem. Recall that at the first phase of the Kruskal's algorithm edges are sorted and at the second phase feasible edges are selected (Kruskal 1956). The former phase can be done in $O(mlog(m))$ time and the latter one in $O(m\alpha(m, n))$ time, where $\alpha(m, n)$ is the inverse Ackermann's function (Cormen et al. 2001). In order to represent all graphs, arising in Algorithm 3, by sorted lists of edges it is enough to sort only edges of H and store the sorted edges in a list. Therefore, edges sorting in subsequent calls of the Kruskal's algorithm can be omitted and, hence, $time(n, m) \in O(m\alpha(m, n))$. Considering the above details, we conclude that the execution time of Algorithm 3 is $O(n(n + m\alpha(n, m)) + mlog(m)) = O(nm\alpha(n, m))$.

Let us demonstrate Algorithm 2 on the graph depicted below (see Fig. 2).

Edges of some optimal spanning tree are highlighted to make them thicker.

The graph (see Fig. 3) has many optimal trees. Let us fix one of them. An optimal solution S^* for the corresponding WIST is unique and it coincides with the set $\{x_1, x_4, x_5\}$. Hence, $l_{S^*}(x_1) = 3, l_{S^*}(x_4) = 2, l_{S^*}(x_5) = 2$ and $u_{S^*}(x_2) = 3$,

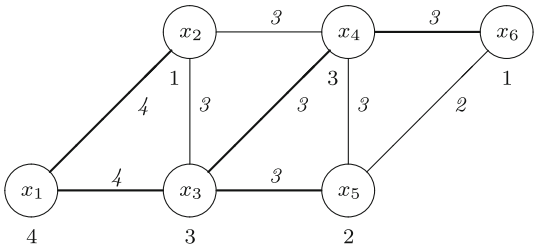


Fig. 3 The first graph with an optimal spanning tree

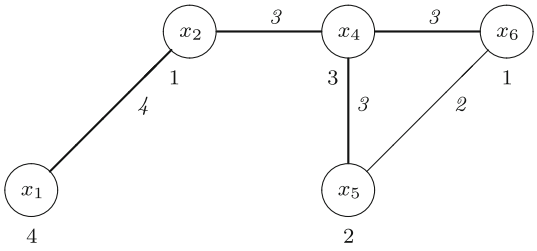


Fig. 4 The second graph with an optimal spanning tree

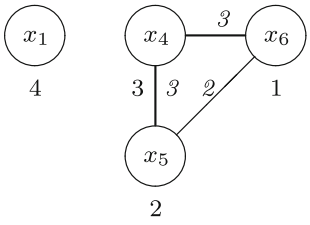


Fig. 5 The third graph with an optimal spanning tree

$u_{S^*}(x_3) = 4, u_{S^*}(x_6) = 2$. So, x_3 is the unique vertex with the maximal tolerance. Since $x_3 \notin S^*$, this vertex should be removed. We have the graph in Fig. 4.

The last graph has the unique optimal spanning tree. One of its optimal independent sets is $\{x_1, x_5, x_6\}$. For this set S^* we have $l_{S^*}(x_1) = 3, l_{S^*}(x_5) = 0, l_{S^*}(x_6) = 0$ and $u_{S^*}(x_2) = 3, u_{S^*}(x_4) = 0$. The vertex x_2 has the maximal tolerance with respect to S^* and it does not belong to S^*, so it should be deleted. The result is shown below (see Fig. 5).

The graph has two connected components, one of them is the one-vertex graph. This vertex is included to the solution. For the second connected component an optimal tree is unique and one of its optimal independent sets is $\{x_4\}$. All tolerances with respect to this solution are equal to zeros. Hence, x_4 is the vertex with the maximal tolerance and it is included to the solution. One can easily verify that the found solution $\{x_1, x_4\}$ is also the unique optimal independent set of the initial graph.

6 Computational experiments

Let us describe conditions of our experiments and analyze their results. Two numerical parameters characterizing a quality of our heuristic are considered. We are interested in values of such parameters and we are not interested in time of their computation.

Table 3 Results for Erdős-Renyi graphs with 50 vertices

p	Density	Approximation ratio in %	Dominance ratio in %
0.05	0.048	100	100
0.1	0.096	100	100
0.15	0.152	98.9	99.99
0.2	0.198	83.3	99.35
0.25	0.254	89.2	99.45
0.3	0.302	81	97.18
0.35	0.34	76.6	91.91
0.4	0.406	69.5	87.69
0.45	0.446	73.7	89.8
0.5	0.494	97.8	99.72
0.55	0.543	66.2	89.15
0.6	0.598	74.3	90.94
0.65	0.652	57.4	55.67
0.7	0.696	50.8	49.69
0.75	0.744	63.5	82.57
0.8	0.802	63	79.04
0.85	0.848	52.2	60.57
0.9	0.91	77.9	96.55
0.95	0.952	66.9	78.38

The first of the mentioned characteristics is the *approximation ratio*, which is defined as the fraction of the weight of an obtained heuristic solution to the weight of an optimal solution. The second one is the *dominance ratio* that is the fraction of the number of independent sets, which weights do not exceed weight of a heuristic solution, to the number of all maximal independent sets. An exact optimal solution is found by the simple greedy branching (in a dense case) or by the branch-and-bound algorithm with the lower bound provided by our heuristic (in a sparse case), all independent sets are enumerated by means of the Bron-Kerbosch's algorithm (Bron and Kerbosch 1973). Recall that the *density* of a graph with n vertices is the fraction of the number of its edges to $\frac{n(n-1)}{2}$ (i.e., the number of edges in the complete n-vertex graph).

We consider random graphs which are produced by different random graph models. The first one is the Erdős-Renyi model. In this model graphs are generated on the set of vertices $\{1, 2, \ldots, n\}$ and for each pair of distinct vertices i and j an edge (i, j) belongs to the graph with a fixed probability p. Graphs with n vertices $1, 2, 3, \ldots, n$ and m edges are produced by the second model. Here a pair (i, j) of distinct vertices is chosen uniformly at random among all feasible pairs, it is added as an edge and removed from a set of feasible pairs. The process is repeated m times. As the third model we consider power-law graphs, i.e. graphs for which the fraction of the number of vertices of degree d to the number of all vertices is asymptotically distributed proportionally to $d^{-\beta}$, where $\beta > 1$. To generate such graphs we use a tool from the Boost Graph Library (www.boost.org/doc/libs/1_-41_-0/boost/graph/plod_-generator.hpp).

Table 4 Results for Erdős-Renyi graphs with 100 vertices

β	Density	Approximation ratio in %	Dominance ratio in %
0.05	0.048	99.9	–
0.1	0.096	97.1	–
0.15	0.148	93.4	–
0.2	0.196	91.6	97.8
0.25	0.256	88.9	99.9
0.3	0.308	67.5	90.7
0.35	0.346	56.2	57.6
0.4	0.398	82.5	99.3
0.45	0.449	37.7	26
0.5	0.496	51.2	61.3
0.55	0.54	37.8	23.3
0.6	0.594	45.5	38.3
0.65	0.656	76.6	95.9
0.7	0.698	37.9	28.5
0.75	0.744	35.9	32.5
0.8	0.805	53.1	56.7
0.85	0.854	28.1	27.57
0.9	0.9	32.9	25.6
0.95	0.953	45.2	40.8

Table 5 Results for the second model graphs with different number of vertices

n	Dens./appr. ratio in % for $m = \frac{3}{2}n$	Dens./appr. ratio in % for $m = [nlog_2(n)]$	Dens./appr. ratio in % for $m = n^{\frac{3}{2}}$
50	0.061/100	0.23/97.7	0.288/95.7
60	0.051/98.9	0.2/83.7	0.262/95.1
70	0.043/100	0.178/100	0.242/95
80	0.038/100	0.16/77.7	0.226/75.5
90	0.034/100	0.146/78.8	0.213/73.1
100	0.03/100	0.134/90.2	0.202/66.6
110	0.028/100	0.124/88.1	0.192/88.8
120	0.025/98.3	0.116/82.2	0.184/67
130	0.023/100	0.109/77.3	0.177/73
140	0.022/99.8	0.103/92.4	0.17/80.9
150	0.02/100	0.097/88.5	0.162/70.5

The first computational experiment is for Erdős-Renyi graphs with 50 and 100 vertices and the values of p are chosen from 0.05 to 0.95 with the step 0.05. Weights of all vertices are uniformly generated within the range $[0, 100n]$. The expectation of the density for Erdős-Renyi graphs is equal to p (Tables 3, 4).

Table 6 Results for power-law graphs with 50 vertices

β	Density	Approximation ratio in %	Dominance ratio in %
1.1	0.361	98.67	99.87
1.2	0.304	91.25	96.94
1.3	0.248	97.49	99.74
1.4	0.204	95.16	99.57
1.5	0.169	100	100
1.6	0.153	100	100
1.7	0.120	100	100
1.8	0.107	100	100
1.9	0.091	100	100
2.0	0.076	100	100
2.1	0.066	100	100
2.2	0.060	99.99	99.31
2.3	0.053	99.99	98.42
2.4	0.046	99.9	93.49
2.5	0.039	100	100
2.6	0.037	100	100
2.7	0.034	100	100
2.8	0.031	100	100
2.9	0.031	100	100
3.0	0.029	100	100
3.1	0.028	100	100
3.2	0.028	100	100
3.3	0.025	100	100
3.4	0.025	100	100
3.5	0.023	100	100
3.6	0.023	100	100
3.7	0.022	100	100
3.8	0.022	100	100
3.9	0.022	100	100
4.0	0.02	100	100

A dash means that we did not wait until an end of the computations, because the Bron-Kerbosch's algorithm took too much time. So, the first experiment shows that for Erdős-Renyi graphs our heuristic *stably* works better for graphs of a small density. This phenomena seems to be independent of a type of graphs and it depends only on a growth of the number of edges. Let us make sure of that by conducting an experiment for graphs of the second model with various functions of a growth of the number of edges. Weights of vertices are distributed as in the first experiment (Table 5).

We can observe from the second series of experiments that if the number of edges grows not quickly, the heuristic algorithm steadily works well enough and this stability

Table 7 Results for power law-graphs with different β and number of vertices

β	Dens./appr. ratio in % for $n = 100$	Dens./appr. ratio in % for $n = 300$	Dens./appr. ratio in % for $n = 500$
2.0	0.075/99.78	0.069/99.09	0.068/100
2.1	0.061/100	0.058/99.89	0.057/99.41
2.2	0.053/99.85	0.049/99.39	0.048/99.6
2.3	0.047/99.56	0.042/99.28	0.041/100
2.4	0.04/100	0.036/100	0.034/99.45
2.5	0.035/100	0.03/99.39	0.03/99.59
2.6	0.032/100	0.026/99.8	0.025/99.69
2.7	0.028/100	0.022/100	0.022/99.57
2.8	0.025/100	0.02/99.53	0.019/99.89
2.9	0.023/100	0.017/99.98	0.016/99.73
3.0	0.02/100	0.015/99.66	0.014/99.87

Table 8 Results for power-law graphs with $\beta = 2.5$

n	Density	Approximation ratio in %
100	0.036	100
200	0.031	100
300	0.03	100
400	0.03	99.71
500	0.03	99.75
600	0.029	99.78
700	0.029	99.69
800	0.029	99.78
900	0.029	99.78
1,000	0.029	99.62

is almost independent of n. On the other hand, a growth of the relative number of edges in graphs leads to downgrading the heuristic indicators and a strong their dispersion.

Power-law graphs are important for the analysis and solution of real-world problems, because they are appropriate mathematical models of scale-free networks (see Barabasi and Reka 1999; Bollobas et al. 2001; Faloutsos et al. 1999; Kumar et al. 1999). The parameter β for such networks is typically in the range $2 < \beta < 3$. It is true for the WWW graph, the Call graph, the Math Reviews graph (the Collaboration graph), graphs of social and airline networks. Power-law graphs with typical above β are extremely sparse, because for power-law graphs with n vertices, m edges and $\beta > 2$ the asymptotical equality $m = \frac{1}{2}\frac{\zeta(\beta-1)}{\zeta(\beta)}n$ holds (Aiello et al. 2000), where $\zeta(\beta)$ is the Riemann's zeta function. Taking into account the results of the previous experiments, one could expect that our heuristic algorithm will stably work well enough on power-law graphs with typical β. Let us be convinced of it by making the corresponding

computational experiments (weights for vertices are chosen in the same way as in the previous experiments) (Tables 6, 7, 8).

We see that our expectations came true for power-law graphs.

Acknowledgments All authors are partially supported by LATNA Laboratory, NRU HSE, RF government grant, ag. 11.G34.31.0057. This study comprises research findings of the first two authors from the "Calculus of tolerances for combinatorial optimization problems: theory and algorithms" project carried out within the Higher School of Economics' 2011–2012 Academic Fund Program. This research was supported by the Federal Target Program "Research and educational specialists of innovative Russia for 2009–2012", state contract No 14.B37.21.0393.

References

Aiello W, Chung F, Lu L (2000) A random graph model for power-law graphs. Exp Math 10:53–66

Balas E, Saltzman M (1991) An algorithm for the three-index assignment problem. Oper Res 39:150–161

Barabasi A, Reka A (1999) Emergence of scaling in random networks. Science 286:509–512

Bekker H, Braad E, Goldengorin B (2005) Selecting the roots of a small system of polynomial equations by tolerance based matching. Lect Notes Comput Sci 3503:610–613

Bron C, Kerbosch J (1973) Finding all cliques of an undirected graph. Commun ACM 16:575–577

Bollobas B, Riordan R, Spencer J, Tusnady G (2001) The degree sequence of a scale-free random graph process. Random Struct Algorithms 18:279–290

Boyko V, Goldengorin B, Kuzmenko V (2006) Tolerance-based algorithms. Theory Optim Decis 5:98–104 (in Ukrainian)

Chen C, Kuo M, Sheu J (1988) An optimal time algorithm for finding a maximum weight independent set in a tree. BIT Numer Math 28:353–356

Cormen T, Leiserson C, Rivest R, Stein C (2001) Introduction of algorithms, 2nd edn. MIT Press/McGraw-Hill, Cambridge

Ernst C, Dong C, Jager G, Richter D, Molitor P (2010) Finding good tours for huge euclidean TSP instances by iterative backbone contraction. In: AAIM, pp 119–130

Faloutsos M, Faloutsos P, Faloutsos C (1999) On the power-law relationships of the internet topology. Comput Commun Rev 29:251–262

Gal T, Greenberg H (1997) Advances in sensitivity analysis and parametric programming. Kluwer Academic Publishers, New York

Germs R, Goldengorin B, Turkensteen M (2012) Lower tolerance-based branch and bound algorithms for the ATSP. Comput Oper Res 39:291–298

Ghosh D, Goldengorin B, Gutin G, Jager G (2007) Tolerance-based greedy algorithms for the traveling salesman problem. In: Bapat R, Das A, Partasaraty T, Neogy S (eds) Mathematical programming and game theory for decision making. World Scientific Publishing, Singapore

Goldengorin B, Jager G, Molitor P (2006) Tolerances applied in combinatorial optimization. J Comput Sci 2:716–734

Greenberg H (1998) An annotated bibliography for post-solution analysis in mixed integer and combinatorial optimization. In: Woodruff D (ed) Advances in computational and stochastic optimization, logic programming and heuristic search. Kluwer Academic Publishers, New York, pp 97–148

Helsgaun K (2000) An effective implementation of the Lin-Kernighan traveling salesman heuristic. Eur J Oper Res 126:106–130

Helsgaun K (2009) General k-opt submoves for the Lin-Kernighan TSP heuristic. Math Program Comput 1:119–163

Jager G, Molitor P (2008) Algorithms and experimental study for the traveling salesman problem of second order. In: COCOA, pp 211–224

Kruskal J (1956) On the shortest spanning subtree of a graph and the traveling salesman problem. Proc Am Math Soc 7(1):48–50

Kumar R, Raghavan P, Rajagopalan S, Tomkins A (1999) Trawing the Web for emerging cyber-communities. Comput Netw 7:1481–1493

Murty K (1968) An algorithm for ranking all the assignments in order of increasing cost. Oper Res 16: 682–687

Ramaswamy R, Orlin J, Chakravarti N (2005) Sensitivity analysis for shortest path problems and maximum capacity path problems in undirected graphs. Math Program 102:355–369

Reinfeld N, Vogel W (1965) Mathematical programming. Prentice Hall, Englewood Cliffs

Shier D, Witzgall C (1980) Arc tolerances in minimum-path and network flow problems. Networks 10: 277–291

Sotskov Y, Leontev V, Gordeev E (1995) Some concepts of stability analysis in combinatorial optimization. Discret Appl Math 58:169–190

Tarjan R (1982) Sensitivity analysis of minimum spanning trees and shortest path trees. Inf Process Lett 14:30–33

Turkensteen M, Ghosh D, Goldengorin B, Sierksma G (2008) Tolerance-based branch and bound algorithms for the ATSP. Eur J Oper Res 189:775–788

Van der Poort E, Libura M, Sierksma G, Van der Veen J (1999) Solving the k-best traveling salesman problem. Comput Oper Res 26:409–425

Volgenant A (2006) An addendum on sensitivity analysis of the optimal assignment. Eur J Oper Res 169:338–339