INFORMATICA, 2013, Vol. 24, No. 3, 461–484
© 2013 *Vilnius University*

A Label Correcting Algorithm with Storing Partial Solutions to Solving the Bus Routing Problem

Jacek WIDUCH
Silesian University of Technology, Institute of Informatics
ul. Akademicka 16, 44-100 Gliwice, Poland
e-mail: jacek.widuch@polsl.pl

Received: February 2012; accepted: December 2012

Abstract. In the paper we describe the bus routing problem (BRP), which the goal is to find a route from the start stop to the final stop minimizing the time and the cost of travel and the length of the route. Additionally the time of starting travel at the start stop is given. Analysis of the problem is presented and in particular we point at properties of routes. The BRP is an example of multicriteria optimization problem (MOP), which the solution is the set of non-dominated solutions. This paper proposes a label correcting algorithm with storing partial solutions for solving the BRP. The algorithm makes possible to find all routes which belong to the set of non-dominated solutions. Apart from that the results of experimental tests are presented. Additionally the results are compared with results for the BRP where the goal is to minimize only the time and the cost of travel and the length of the route is not taken into consideration.
Key words: bus routing problem, multicriteria optimization, set of non-dominated solutions, multicriteria shortest path problem, directed weighted graph, variable weights, shortest path, loopless path, label correcting algorithm.

1. Introduction

The shortest path problem is among the most studied the graph optimization problems. Given a graph with a single weight function, the goal is to find a path with the minimal weight. It has been the subject of extensive research for many years resulting in the publication of a large number of papers. There are well known algorithms for finding the shortest path proposed by Dijkstra (1959), Bellman (1958), Ford (1956), Floyd (1962), Warshall (1962) and Johnson (1977). In many cases using a single weight function is insufficient because it does not describe precisely the studied problem. In this case, a graph with k ($k > 1$) weight functions is used and the problem is called the multicriteria shortest path (MSP) problem.

The MSP problem is an example of multicriteria optimization and it is known to be NP-complete by transformation from a 0–1 knapsack problem (Garey and Johnson, 1990; Hansen, 1980; Skriver, 2000a). Many algorithms for solving the BSP problem are known and these algorithms are classified into the following categories: label setting algorithms (Hansen, 1980; Martins, 1984; Tung and Chew, 1988), label correcting algorithms (Brumbaugh-Smith and Shier, 1989; Corley and Moon, 1985; Daellenbach and

De Kluyver, 1980; Skriver and Andersen, 2000b), k-th shortest path algorithms (Climaco and Martins, 1982), two phases algorithms (Mote *et al.*, 1991) and the others (Dell'Olmo *et al.*, 2005; Machuca *et al.*, 2009; Mandow and Pérez-de-la-Cruz, 2008a; Mandow and érez-de-la-Cruz, 2008b; Martí *et al.*, 2009; Raith and Ehrgott, 2009). All mentioned algorithms assume constant weights, i.e. the weight function has a constant value for a given arc of graph and it does not change. There are differences between problem with constant and variable weights.

In this paper the BRP, which is an example of the MSP problem with variable weights, is presented and analysed. The bus network is represented by the graph with weight functions. The weight functions determine the cost and the travel time and their values are calculated during the process of finding the solutions. For that reason, values of both weight functions are not constant for a given arc. In the paper we have made a theoretical analysis of the problem and with particular emphasis on differences between the problem with constant weights and the problem with variable weights. The algorithm for solving the BRP is also presented and it belongs to the group of label correcting algorithms with storing partial solutions. The algorithm was implemented and tested and the test results are also presented.

The paper is divided into five sections. Section 2 contains description of the BRP, i.e. a formulation of the BRP (Section 2.1) and an analysis of the BRP, in particular pointed at differences between the MSP with variable weights and constant weights (Section 2.2). The label correcting algorithm for solving the BRP is described in Section 3. In Section 4 experimental test results are reported. Conclusions are outlined in the final section.

2. The Bus Routing Problem (BRP)

2.1. *Formulation of the BRP*

The BRP is related to the choice of means of transport and finding the route of travel between the two given points. We have given the bus network consisted of n stops $s_1, \dots, s_n$. In the network buses of M bus lines numbered from 1 to M are run. Additionally, the network is divided into zones, which determines the cost of travel.

The route of the bus line consists of the sequence of stops through which the bus runs from the start stop to the final stop of the line and it is defined for each bus line. The travel between two given stops is directed. If the bus runs from stop s_i to stop s_j, it does not imply that the bus of the same line runs in the opposite direction, i.e. from s_j to s_i. The bus of a given bus line can run in both directions, but these routes can be different. The route of each bus line consists of different stops except for the start and the final stops which can be the same. If the final stop is the same as the start stop, then the bus line is called the circular line.

Let the route of i-th bus line ($i = 1, \dots, M$) consists of sequence of stops:

$$\langle s_0^i, s_1^i, \dots, s_{k-1}^i, s_k^i \rangle \tag{1}$$

where s_0^i is the start stop and s_k^i is the final stop of the line. The bus of i-th line runs between stops belonging to the route with the given frequency, i.e. it leaves the start stop s_0^i at time T_0^i, passes through stops $s_1^i, \dots, s_{k-1}^i$ at times $T_0^i + \delta_0^i, \dots, T_0^i + \delta_{k-2}^i$, respectively, and it reaches the final stop s_k^i at time $T_0^i + \delta_{k-1}^i$. The next course the bus of this line starts at time T_1^i ($T_1^i = T_0^i + \beta_0^i$) and thus it reaches stops $s_1^i, \dots, s_k^i$ at times $T_1^i + \delta_0^i, \dots, T_1^i + \delta_{k-1}^i$. The bus of i-th bus line executes p^i courses and leaves the start stop s_0^i at times $T_0^i, \dots, T_{p-1}^i$ ($T_0^i < \cdots < T_{p-1}^i$), where $T_j^i = T_0^i + \beta_{j-1}^i$ ($j = 1, \dots, p-1$). The T_0^i, $\beta_0^i, \dots, \beta_{p-2}^i$, $\delta_0^i, \dots, \delta_{k-1}^i$ ($0 < \beta_0^i < \cdots < \beta_{p-2}^i$; $0 < \delta_0^i < \cdots < \delta_{k-1}^i$) values are defined by the timetable of the bus line. We omit the times of getting on and off the bus by passengers, i.e. we assume that the time of departure from a stop equals the time of arrival to this stop.

For defined the bus network structure, the bus lines routes and the timetable, we have given the start stop s_s and the final stop s_e ($s_s \neq s_e$) between which we want to travel. The time of starting travel T_s at the start stop is given additionally. The objective is to find a route from s_s to s_e minimizing the time and the cost of travel and the length of the route. We should determine all stops belonging to the route, the bus lines which we travel between stops, stops of changes and times of departure from all stops belonging to the route.

The time of the travel depends on the chosen route and the possible stops of changes. It is the sum of the travel times between stops belonging to the route,[1] time of waiting at the start stop and times of waiting for changes. The cost of travel depends on the location of the stops in the area of zones and it is calculated as follows. A ticket for a single travel (a travel without a bus change) within the area of a single zone equals c_1 ($0 < c_1$) units, within two zones it equals c_2 ($c_1 < c_2$) units and within the confines of more than two zones it equals c_3 ($c_2 < c_3$) units.[2] For that reason the cost of travel from the start stop s_s to the final stop s_e is equal to the sum of costs of travel between the stops of bus changes. Among bus lines there are fast lines. The travel by bus of a fast line is faster than travel by a regular line which takes into account the timetable. The cost of the travel by a fast line is twice as large as the cost of the travel by a regular line. The length of the route equals the number of stops belonging to the route.

2.2. *Analysis of the BRP*

The bus network is described by the directed weighted graph $G = (V, E)$ with $|V| = n$ vertices $v_1, \dots, v_n$ and $|E| = m$ arcs $e_1, \dots, e_m$ ($e_j = (v_a, v_b)$; $v_a, v_b \in V$) (Chartrand *et al.*, 2010; Jungnickel, 1989; Wilson, 1996). The vertices represent the bus stops where the vertex v_j represents the bus stop s_j ($j = 1, \dots, n$). Thus a vertex expression with reference to the graph representing the bus network determines the bus stop of the network. The arc $e_j = (v_a, v_b)$ represents the route of travel of a specific bus line whose buses run

[1] The travel times between stops are defined by the timetable of bus lines.

[2] In a sample routes presented in the paper we assume the following costs of a ticket: $c_1 = 2.0$, $c_2 = 2.3$ and $c_3 = 2.6$ units.

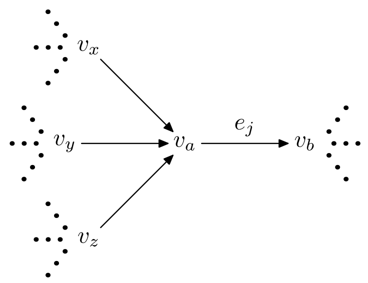

Fig. 1. A part of graph representing the bus network.

directly from a stop represented by v_a to a stop represented by v_b. Direct travel denotes that the route from v_a to v_b does not include other vertices.

$$\langle v_0^i, v_1^i, \ldots, v_{k-1}^i, v_k^i \rangle. \tag{2}$$

The route of i-th bus line descried by (1) is represented by a path (2) in the graph G from the vertex v_0^i to the vertex v_k^i, where v_0^i represents the start stop s_0^i and v_k^i represents the final stop s_k^i of the i-th bus line. The travel between two given stops is directed, thus if there exists the arc (v_a, v_b), it does not imply that there exists the arc (v_b, v_a). Between two given stops buses of many bus lines can run. For that reason the graph G representing the bus network can contain parallel arcs. Each arc $e_j = (v_a, v_b)$ has three weights: $l(e_j)$, $t(e_j)$ and $c(e_j)$.

The weight $l(e_j)$ equals the line number the bus runs from v_a to v_b and it takes values from range $1, \ldots, M$. The weight $t(e_j)$ takes positive values, it is not constant and it equals $t(e_j) = T_b - T_a$, where T_b is the time of arrival to v_b and T_a is the time of arrival to v_a. Thus it is a sum of the travel time from v_a to v_b and the possible time of waiting at v_a. The travel time from v_a to v_b is defined by the timetable and it is constant. The value of weight $t(e_j)$ is determined by the time of waiting at v_a which is variable. Let us consider a part of the bus network presented in Fig. 1. There are many ways of travel to v_a – in the first case we arrive from v_x and the time of arrival to v_a is T_a^x, in the second case we arrive from v_y at T_a^y and in the last case the time of arrival is T_a^z and we arrive from v_z. Let us assume that the travel time from v_a to v_b equals t_{ab} and the time of departure from v_a to v_b is the same in all cases and it equals T_a, but (3) is satisfied. The times of waiting at v_a are equal to: $\Delta t_a^x = T_a - T_a^x$, $\Delta t_a^y = T_a - T_a^y$ and $\Delta t_a^z = T_a - T_a^z$. Since (3) is satisfied, so $\Delta t_a^x \neq \Delta t_a^y \neq \Delta t_a^z$ and the weight $t(e_j)$ in each case takes different values: $t'(e_j) = \Delta t_a^x + t_{ab}$, $t''(e_j) = \Delta t_a^y + t_{ab}$, $t'''(e_j) = \Delta t_a^z + t_{ab}$. Therefore the weight $t(e_j)$ is not constant.

$$T_a^x \neq T_a^y \neq T_a^z. \tag{3}$$

The weight $c(e_j)$ takes non-negative values and it equals $c(e_j) = c_b - c_a$, where c_b equals the cost of travel to v_b and c_a equals the cost of travel to v_a.[3] The values c_a and c_b

[3]We assume the travel by a bus of regular line, otherwise the value $c(e_j)$ must be multiplied by 2.

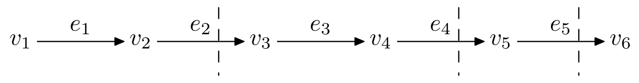

Fig. 2. The route of a bus line. The zone borders are denoted by the dashed lines.

Table 1
The values $c(e_1), \dots, c(e_5)$ depending on the start vertex v_s.

v_s	$c(e_1)$	$c(e_2)$	$c(e_3)$	$c(e_4)$	$c(e_5)$
v_1	c_1	$c_2 - c_1$	0	$c_3 - c_2$	0
v_2	–	c_2	0	$c_3 - c_2$	0
v_3	–	–	c_1	$c_2 - c_1$	$c_3 - c_2$
v_4	–	–	–	c_2	$c_3 - c_2$
v_5	–	–	–	–	c_2

depend on the location of v_a and v_b in the area of zones and they also depend on a possible change at v_a. At first, let us consider the case when v_a and v_b are located in the same zone. If we change at v_a then $c(e_j) = c_1$ or $c(e_j) = 0$ otherwise, because we do not cross a zone border while running from v_a to v_b and it does not increase the cost of travel. In the second case, v_a and v_b are located in different zones. The weight equals $c(e_j) = c_2$ if we change at v_a, otherwise the value $c(e_j)$ depends on the number of zone borders that have been crossed since the last change while travelling to v_a. If we did not cross any zone border, then $c(e_j) = c_2 - c_1$. It equals $c(e_j) = c_3 - c_2$ if we have crossed a single zone border and if we have crossed two or more zone borders it equals $c(e_j) = 0$. So, we proved that weight $c(e_j)$ is variable.

A determination of the value of weight $c(e_j)$ we illustrate by an example of running to the final vertex $v_e = v_6$ using the bus of line whose the route is shown in Fig. 2. The values $c(e_1), \dots, c(e_5)$ depend on the start vertex v_s from which we start the travel and they are presented in Table 1.

The bus route from the start stop s_s to the final stop s_e is represented by the path p_{v_s,v_e} (4) from the start vertex v_s to the final vertex v_e in the graph G. In addition, the times of departures $T_0, T_1, \dots, T_{k-1}$, where T_j is a time of departure from v_j $(j = 0, \dots, k-1;$ $v_j \in p_{v_s,v_e})$, are stored. Thus a path expression with reference to the graph G determines the bus route in the bus network. The final solution is called the path p_{v_s,v_e} and the path p_{v_s,v_i} $(v_i \neq v_e)$ is called the partial solution. The path $p'_{v_i,v_j} = sub_{p_{v_s,v_e}}(v_i, v_j)$ is a sub-path of p_{v_s,v_e} and it contains a subsequence of vertices and arcs from v_i to v_j belonging to p_{v_s,v_e}.

$$p_{v_s,v_e} = \langle v_0 = v_s, e_1, \dots, v_{k-1}, e_k,\ v_k = v_e \rangle. \tag{4}$$

The length of the path p_{v_s,v_e} is denoted by $D(p_{v_s,v_e})$ and it equals the number of vertices belonging to the path. The path p_{v_s,v_e} described by (4) has two weights T and C, defined by (5) and (6) and its length equals $D(p_{v_s,v_e}) = k + 1$. These weights represent the time and the cost of travel from v_s to v_e and they are equal to the sum of corresponding weights of arcs belonging to the path p_{v_s,v_e}. According to the formulation of the BRP presented in Section 2.1, the goal of the BRP is to find the path from v_s to v_e in the graph G minimizing (5), (6) and $D(p_{v_s,v_e})$ simultaneously.

$$T(p_{v_s,v_e}) = \sum_{j=1}^{k} t(e_j), \tag{5}$$

$$C(p_{v_s,v_e}) = \sum_{j=1}^{k} c(e_j). \tag{6}$$

The BRP is one of the problems in MOP, where we have given k $(k > 1)$ criterion functions f_i $(i = 1, \ldots, k)$ which can be minimized or maximized. In the first case the objective is to find a solution for which the value of the function is minimal and in the second case the function of determined solution should takes the maximum value. In most cases there is no a single solution for which all the criterion functions take an optimum value, i.e. the minimum or the maximum. For that reason the solution of the MOP is a set of solutions called the set of non-dominated (Pareto optimal) solutions (Coello Coello *et al.*, 2002; Ehrgott, 1999; Ehrgott and Gandibleux, 2000; Korhonen, 1992; Pareto, 1896; Roy and Vincke, 1981; Ulungu and Teghem, 1994; Voorneveld, 2003).

Definition 1. Let there be given k $(k < 1)$ minimized criterion functions $f_1, \ldots, f_k$ and two solutions X and Y. The X solution is said to dominate the Y solution if $\forall i \in \{1, \ldots, k\}$: $f_i(X) \leqslant f_i(Y)$ and $\exists j \in \{1, \ldots, k\}$: $f_j(X) < f_j(Y)$.

The BRP solving is reduced to solving the MSP problem between the start and the final vertices in a graph with variable weights. The solution of the problem consists of set of paths in the graph G formed the set of non-dominated solutions. The weights defined by (5) and (6) and the length of the path are the criterion functions and all are minimized.

Let us analyse properties of paths belonging to the set of non-dominated solutions. The set of non-dominated solutions can contain many paths with the same values of the weights (5) and (6) and the same length, according to Definition 1 these paths are non-dominated solutions. Let there be given two paths p'_{v_s,v_e} and p''_{v_s,v_e}, where (7) and (8) are satisfied. The paths p'_{v_s,v_e} and p''_{v_s,v_e} have one of the following properties:

- p'_{v_s,v_e} and p''_{v_s,v_e} differ from each other in vertices and arcs belonging to these paths,
- p'_{v_s,v_e} and p''_{v_s,v_e} contain the same sequence of vertices but they differ from each other in arcs belonging to these paths,
- p'_{v_s,v_e} and p''_{v_s,v_e} contain the same sequence of vertices and arcs, differ from each other only in the times of departure from all vertices belonging to these paths.

$$T(p'_{v_s,v_e}) = T(p''_{v_s,v_e}), \tag{7}$$

$$C(p'_{v_s,v_e}) = C(p''_{v_s,v_e}). \tag{8}$$

Let us consider the path p'_{v_s,v_e} containing a cycle $p_{v_i,v_i} = sub_{p'_{v_s,v_e}}(v_i, v_i)$, where v_i is the start and the final vertex of the cycle (Fig. 3a). If there exists the path p'_{v_s,v_e}, then

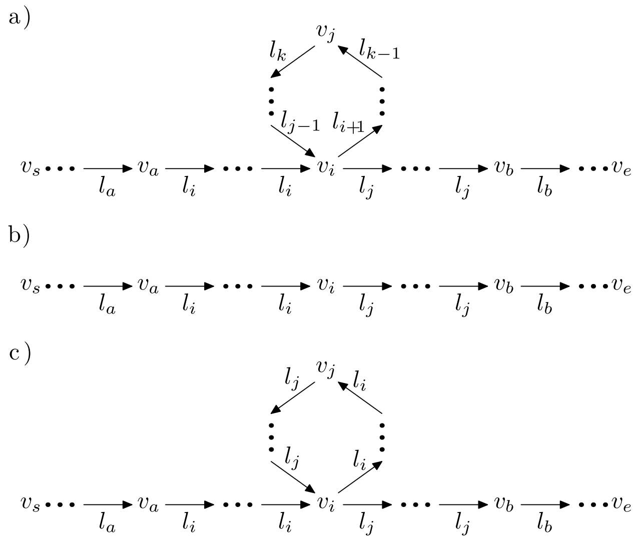

Fig. 3. A part of sample paths from the start vertex v_s to the final vertex v_e. The arcs are described by the bus lines which the buses run between vertices.

there exists the path p''_{v_s,v_e} (Fig. 3b) which has the same sequence of vertices and arcs like p'_{v_s,v_e} and it is devoid of the cycle p_{v_i,v_i}, i.e.:

$$sub_{p'_{v_s,v_e}}(v_s, v_i) = sub_{p''_{v_s,v_e}}(v_s, v_i), \tag{9}$$

$$sub_{p'_{v_s,v_e}}(v_i, v_e) = sub_{p''_{v_s,v_e}}(v_i, v_e). \tag{10}$$

It has been shown that $T(p'_{v_s,v_e}) > T(p''_{v_s,v_e})$ and $C(p'_{v_s,v_e}) > C(p''_{v_s,v_e})$ if weights of arcs are constant and non-negative, and at least one is positive (Henig, 1985; Tung and Chew, 1988; Tung and Chew, 1992). In this case the set of non-dominated solutions contains only loopless paths. The graph G representing the bus network has variable weights and for this case, it can be formulated Lemma 1.

Lemma 1. *Let there be given the path p'_{v_s,v_e} containing the cycle p_{v_i,v_i} and the path p''_{v_s,v_e} which has the same sequence of vertices and arcs like p'_{v_s,v_e} and which is loopless. If weights of arcs are variable and take non-negative values then $T(p'_{v_s,v_e}) \geqslant T(p''_{v_s,v_e})$ and $C(p'_{v_s,v_e}) \geqslant C(p''_{v_s,v_e})$.*

Proof. A property of the route of the bus line indicates that we change at v_i in p''_{v_s,v_e}. The weights c and t of arcs do not take negative values and it follows that: $T(p'_{v_s,v_e}) \geqslant T(p''_{v_s,v_e})$ and $C(p'_{v_s,v_e}) \geqslant C(p''_{v_s,v_e})$. The goal of the proof is to prove when conditions (7) and (8) are satisfied (both conditions are not satisfied in a graph with constant weights). The condition (9) holds, thus times of departure from $v_s, \ldots, v_{i-1}$ are identical in p'_{v_s,v_e} and p''_{v_s,v_e} and it indicates that $T(sub_{p'_{v_s,v_e}}(v_s, v_i)) = T(sub_{p''_{v_s,v_e}}(v_s, v_i))$. If the time of waiting for a change at v_i in p''_{v_s,v_e} equals the time of making the cycle p_{v_i,v_i} in p'_{v_s,v_e}, then times of departure from vertices $v_i, \dots, v_{e-1}$ in both paths are identical and (7) is satisfied. When $\delta t'' = \delta t' + T(p_{v_i,v_i})$, where $\delta t'$ and $\delta t''$ are the sums of times of waiting for change at the vertices in p'_{v_s,v_e} and p''_{v_s,v_e}, then (7) is satisfied, too.

A necessary condition for fulfilment (8) is a single change at v_j in the cycle p_{v_i,v_i} and passing v_i without a change (Fig. 3c). For path p'_{v_s,v_e} and p''_{v_s,v_e} there hold conditions:

$$C\big(sub_{p'_{v_s,v_e}}(v_s, v_a)\big) = C\big(sub_{p''_{v_s,v_e}}(v_s, v_a)\big),$$
$$C\big(sub_{p'_{v_s,v_e}}(v_b, v_e)\big) = C\big(sub_{p''_{v_s,v_e}}(v_b, v_e)\big).$$

If all vertices belonging to the cycle p_{v_i,v_i} are located in the same zone, then conditions (11) and (12) are satisfied and (8) is satisfied, too.

$$C\big(sub_{p'_{v_s,v_e}}(v_a, v_j)\big) = C\big(sub_{p''_{v_s,v_e}}(v_a, v_i)\big), \tag{11}$$
$$C\big(sub_{p'_{v_s,v_e}}(v_j, v_b)\big) = C\big(sub_{p''_{v_s,v_e}}(v_i, v_b)\big). \tag{12}$$

If the vertices belonging to the cycle p_{v_i,v_i} are located in different zones, then (11) is satisfied when:

- we do not cross two or more zones when we run from v_a to v_i and we do not cross any zone when we run from v_i to v_j in p_{v_i,v_i}, or
- we cross two or more zones when we run from v_a to v_i.

The condition (12) is satisfied in similar cases, i.e.:

- we do not cross any zone when we run from v_j to v_i in p_{v_i,v_i}, if we do not cross two or more zones when we run from v_i to v_b, or
- we cross two or more zones when we run from v_i to v_b.

□

To illustrate Lemma 1 let us consider a sample routes from $v_s = 1$ to $v_e = 4$, where the time of starting travel equals $T_s = 8{:}00$. The route $p'_{1,4}$ without a cycle is presented in Tables 2 and 3 the route $p''_{1,4}$ containing a cycle is presented. All vertices belonging to these paths are located in the same zone and (13), (14) are satisfied. In the route $p'_{1,4}$ we change at vertex 3 and the time of waiting for a change equals 45 minutes and it equals the time of making the cycle $3 \to 5 \to 3$ in the route $p''_{1,4}$. In addition, (15) and (16) occur and for that reason the cost of making the cycle does not increase the total cost of travel.

$$T\big(p'_{1,4}\big) = T\big(p''_{1,4}\big) = 65 \text{ minutes}, \tag{13}$$
$$C(p'_{1,4}) = C\big(p''_{1,4}\big) = 6.0 \text{ units}, \tag{14}$$
$$C\big(sub_{p'_{1,4}}(2,3)\big) = C\big(sub_{p''_{1,4}}(2,5)\big) = 2.0 \text{ units}, \tag{15}$$
$$C\big(sub_{p'_{1,4}}(3,4)\big) = C\big(sub_{p''_{1,4}}(5,4)\big) = 2.0 \text{ units}. \tag{16}$$

The routes $p'_{1,4}$ and $p''_{1,4}$ show that weights of arcs representing the time and the cost of travel are variable. Let us consider the arc $(3,4)$. In the route $p'_{1,4}$ weights of this arc

Table 2
A timetable of the route from $v_s = 1$ to $v_e = 4$ without a cycle, where the time of starting travel equals $T_s = 8{:}00$.

Vertex/zone	Arrival time	Departure time	Cost of travel	Line
1/1	8:00	8:05	0.0	1
2/1	8:08	8:12	2.0	2
3/1	8:15	9:00	4.0	3
4/1	9:05		6.0	

Table 3
A timetable of the route from $v_s = 1$ to $v_e = 4$ containing a cycle $3 \to 5 \to 3$, where the time of starting travel equals $T_s = 8{:}00$.

Vertex/zone	Arrival time	Departure time	Cost of travel	Line
1/1	8:00	8:05	0.0	1
2/1	8:08	8:12	2.0	2
3/1	8:15	8:15	4.0	2
5/1	8:20	8:55	4.0	3
3/1	9:00	9:00	6.0	3
4/1	9:05		6.0	

equal $t(3,4) = 50$ minutes and $c(3,4) = 2.0$ units, but in the route $p''_{1,4}$ these weights equal $t(3,4) = 5$ minutes and $c(3,4) = 0.0$ units.

It should be pointed out that the path p'_{v_s,v_e} defined by Lemma 1 for which (7) and (8) are satisfied is dominated by the path p''_{v_s,v_e} due to $D(p'_{v_s,v_e}) > D(p''_{v_s,v_e})$. The last property of routes belonging to the set of non-dominated solutions is determined by Lemma 2.

Lemma 2. *Let p_{v_s,v_e} and p'_{v_s,v_e} be paths which consist of the same sequence of vertices and arcs but differ from each other in times of departure from vertices belonging to these paths. Both paths are non-dominated solutions if we change once at least in these paths.*

Proof. If we do not a change in p_{v_s,v_e} and p'_{v_s,v_e}, it follows that departure time from v_s are different in these paths and departure times from all vertices belonging to these paths are different, too. For that reason $T(p_{v_s,v_e}) \neq T(p'_{v_s,v_e})$ and one of them is a dominated solution.

Let these paths consist of sequence of vertices and arcs:

$$p_{v_s,v_e} = p'_{v_s,v_e} = \langle v_s, \dots, e_j, v_j, e_k, \dots, e_k, v_k, e_i, \dots, v_e \rangle$$

where v_j and v_k are vertices where we change. Let the arrival times at v_j and v_k are equal T_{aj}, T_{ak}, and T_{dj}, T_{dk} be departure times from v_j and v_k, and the time of waiting for a change at v_k equals $\delta t_k = T_{dk} - T_{ak}$. If it is possible to leave v_j at time $T'_{dj} > T_{dj}$ in p'_{v_s,v_e}, where $T'_{dj} - T_{dj} \leqslant \delta t_k$, then it is possible to leave v_k at time T_{dk}, and it does not increase the total time of travel from v_s to v_e in the path p'_{v_s,v_e}, i.e., $T(p_{v_s,v_e}) = T(p'_{v_s,v_e})$. □

Table 4
Timetables of the routes from $v_s = 6$ to $v_e = 10$, where the time of starting travel equals $T_s = 8{:}00$.

Vertex	Arrival times			Departure times			Line
6	8:00	8:00	8:00	8:10	8:25	8:40	4
7	8:15	8:30	8:45	8:15	8:30	8:45	4
8	8:25	8:40	8:55	9:00	9:00	9:00	5
9	9:05	9:05	9:05	9:05	9:05	9:05	5
10	9:10	9:10	9:10				

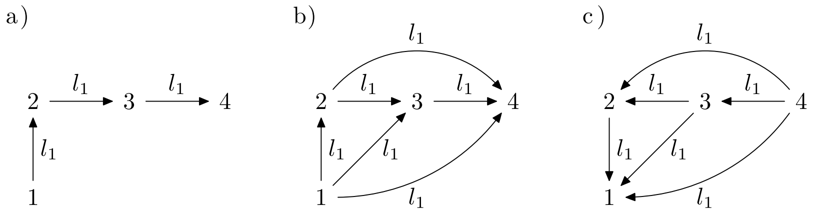

Fig. 4. The route of the bus line l_1 represented by the graphs G, G_a and G_{ar}.

Lemma 2 is illustrated in Table 4, where a sample routes from $v_s = 6$ to $v_e = 10$ are presented. There are three routes with the same time and cost of travel which contain the same sequence of vertices and arcs, but departure times from vertex $v_s = 6$ are different.

3. The Algorithm to Solving the BRP

In this section we present an algorithm for finding all routes from the start vertex v_s to the final vertex v_e, belonging to the set of non-dominated solutions. The algorithm belongs to a group of label correcting algorithms with storing partial solutions and current elimination of dominated solutions. The algorithm is presented in the form of the procedure FindRoutes (Fig. 5). The bus network is represented by the graph $G = (V, E)$ and additionally it is represented by other graphs: $G_a = (V, E_a)$, $G_{ar} = (V, E_{ar})$. We create the graph G_a by adding into the set E additional arcs for each bus line according to the following principles. Let the route of i-th bus line is described by (2). For each vertices v^i_j $(j = 0, \dots, k-2)$ and v^i_u $(u = j+2, \dots, k)$ we add arc (v^i_j, v^i_u). For example, let us consider the bus network represented by the graph G where buses of the bus line l_1 are run (Fig. 4a). The graph G_a (Fig. 4b) is obtained by adding arcs $(1, 3)$, $(1, 4)$ and $(2, 4)$ into the set E. The graph G_{ar} is created by reversing all arcs of E_a to the opposite direction (Fig. 4c).

During the process of finding the solutions the following data structures are used: S, PS and Q. The S structure is a list of final solutions formed as the set of non-dominated solutions. Each solution represents the path from v_s to v_e, where the time of starting travel

```
Input:   $G, G_a, G_{ar}$   –  graphs representing the bus network
         $v_s, v_e$         –  vertices representing the start and the final stops
         $T_s$              –  the time of starting travel at the start stop
Output:  $S$                –  list of computed routes
procedure FindRoutes($G, G_a, G_{ar}, v_s, v_e, T_s, S$)
    $P_t \leftarrow$ PathMinT($G_a, v_s, v_e, T_s$);   $\overline{t}^e_m \leftarrow$ MinT($G_{ar}, v_e$);
    $P_c \leftarrow$ PathMinC($G_a, v_s, v_e, T_s$);   $\overline{c}^e_m \leftarrow$ MinC($G_{ar}, v_e$);
    $P_d \leftarrow$ PathMinD($G_a, v_s, v_e, T_s$);   $\overline{d}^e_m \leftarrow$ MinD($G_{ar}, v_e$);
    for all $v_i \in V$ do
        $PS[v_i] \Leftarrow \emptyset$;
    end for
    $PS[v_s] \Leftarrow (0, 0, 0, 1, (0, T_s, \textbf{null}))$;   Push($Q, v_s$);
    while $Q \neq \emptyset$ do
        $v_i \leftarrow$ Pop($Q$);
        for all $(v_i, v_j, l_j) \in out(v_i); v_j \neq v_s$ do
            for all $P \in$ not analysed solutions in $PS[v_i]$ do
                if not BelongsToPath($P, v_j$) and $P.l_j \neq l_j$ then
                    $T_i \leftarrow$ time of departure from $v_i$;
                    $d_{sj} \leftarrow$ length of the route from $v_s$ to $v_j$;
                    $t_{sj}$, $c_{sj} \leftarrow$ the time and the cost of travel from $v_s$ to $v_j$;
                    if NotDom($t_{sj}, c_{sj}, d_{sj}, P_t, P_c, P_d, \overline{t}^e_m, \overline{c}^e_m, \overline{d}^e_m$) then
                        $P_j \leftarrow$ CheckSolution($t_{sj}, c_{sj}, l_j, d_{sj}, PS[v_j]$);
                        if $P_j \neq$ null then   {exists solution $P_j$ in the list $PS[v_j]$}
                            Push($P_j.LP, (v_i, T_i, \&P)$);
                        else
                            $P' \leftarrow (l_j, t_{sj}, c_{sj}, d_{sj}, (v_i, T_i, \&P))$;
                            if $v_j = v_e$ then
                                $PS[v_j] \Leftarrow$ AddToFinalList($PS[v_j], P'$);
                            else
                                $PS[v_j] \Leftarrow$ AddToList($PS[v_j], P'$);   Push($Q, v_j$);
                            end if
                        end if
                    end if
                end if
            end for
        end for
    end while
    $S \Leftarrow$ CreateFullPaths($PS[v_e], v_e$);   $S \Leftarrow$ PathsDifferTimes($S, G$);
end procedure
```

Fig. 5. The algorithm for finding all routes from the start vertex v_s to the final vertex v_e belonging to the set of non-dominated solutions.

at v_s equals T_s. For each vertex $v_j \in V$ the $PS[v_j]$ structure contains a list of partial solutions from v_s to v_j and for v_e it contains a list of final solutions and it constitute the set of non-dominated solutions. The partial and the final solutions are described by a record $(l_j, t_{sj}, c_{sj}, d_{sj}, LP)$, where:

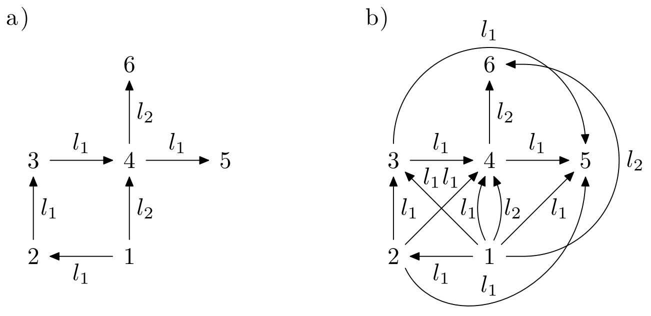

Fig. 6. A sample bus network represented by the graphs G and G_a.

- l_j – the bus line by which we travel to v_j from a vertex which precedes v_j in the path,
- t_{sj}, c_{sj} – the time and the cost of travel from v_s to v_j (for v_s they equal 0),
- d_{sj} – the length of the path from v_s to v_j,
- LP – a list of records $(v_i, T_i, *P_i)$, where:
  - v_i – the vertex which precedes the vertex v_j in the path,
  - T_i – time of departure from v_i towards v_j (for v_s it equals the time of starting travel T_s),
  - $*P_i$ – a pointer to the partial solution for the vertex v_i.

The Q structure is a queue containing vertices for which the partial solutions have been determined.

The initial part of the algorithm (lines 2–8) contains finding the paths of the minimal time and the minimal cost of travel and the minimal length from v_s to v_e in the graph G_a. These paths are stored in P_t, P_c and P_d, and they are computed by the PathMinT (line 2), the PathMinC (line 3) and the PathMinD procedures (line 4), where the first two procedures implement the Dijkstra algorithm and the last implements Breadth-First-Search method (Jungnickel, 1989). We can prove that it is not possible to find the path of the minimal cost of travel in the graph G using shortest paths algorithms like the Dijkstra or the Bellman–Ford. Let us consider the graph G representing the bus network where we want to determine the path of the minimal cost of travel from $v_s = 1$ to $v_e = 5$ (Fig. 6a). If we assume that all vertices belong to the same zone except for vertices 2 and 3, then $1 \to 4 \to 5$ is the path of the minimal cost of travel determined in the graph G by the Dijkstra or the Bellman–Ford algorithm and it has cost 4.0 units. This path is incorrect because the path of minimal cost of travel has form $1 \to 2 \to 3 \to 4 \to 5$ and its cost equals 2.6 units. This path is determined by the Dijkstra or the Bellman–Ford algorithm in the graph G_a.

The Dijkstra algorithm is also used in the procedures MinT (line 2) and MinC (line 3) for determining minimal costs and times of travel from all vertices to the final vertex v_e. Determined values are stored in arrays $\overline{c}^e_m$ and $\overline{t}^e_m$, where $\overline{c}^e_m[v_j]$ and $\overline{t}^e_m[v_j]$ stores adequately the minimal cost and time of travel from v_j to v_e. The last array $\overline{d}^e_m[v_j]$ stores minimal lengths of paths from all vertices to the final vertex v_e. It is determined by the

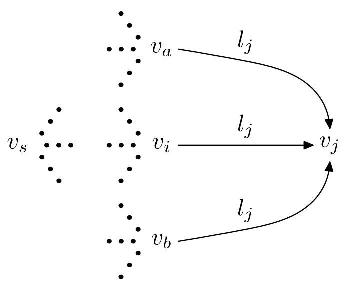

Fig. 7. The vertices which precede the vertex v_j in the route.

procedure MinD (line 4), which implements Breadth-First-Search method. For each vertex $v_i \neq v_s$ the $PS[v_i]$ structure is initialised by the empty list (lines 5–7). The $PS[v_s]$ structure of the start vertex v_s is initialised by a record representing an initial solution from which the algorithm starts determining all solutions and v_s is inserted into the queue (line 8).

In the **while**-loop (lines 9–33) all non-dominated paths are computed by visiting the vertices of the graph G_a using a modified Breadth-First-Search method. We do not compute paths which differ from each other only in times of departure from vertices, because these paths are computed (line 34) based on the paths computed in the **while**-loop.

A single iteration contains following operations. The first vertex v_i is taken from the queue Q (line 10) and all its outgoing arcs (v_i, v_j) are analysed (lines 11–32). For arc (v_i, v_j) we try to extend each the solution P belonging the list $PS[v_i]$ and create a new solution for the vertex v_j (lines 12–31). The new solution represents a path from v_s to v_j where v_j is preceded by v_i in the path. During the process of finding the solutions the graph G_a containing additional arcs is used and it allows to examine all reachable vertices from v_i by the bus of the bus line l_j. For that reason only vertices where we change are added to the path. Thus, we extend the solution P if all conditions are satisfied:

- the solution P has not been analysed yet (checking in the line 12),
- the vertex v_j does not belong to the path represented by the solution P (it is checked by the procedure BelongsToPath in the line 13),
- the bus is changed at v_i (checking in the line 13).

If all mentioned conditions are satisfied, the time of departure from v_i, the length of the route, the time and the cost of travel from v_s to v_j are determined for the extended solution (lines 14–16). In the next, the procedure NotDom is used to determine if it is possible to obtain a non-dominated final solution from the extended solution (line 17). When the estimation is positive, we try to add the extended solution to the list $PS[v_j]$.

The vertex v_j can be reached from many vertices by the bus of line l_j (Fig. 7). In the Fig. 7, there are three ways to travel to v_j and each of the cases represents a different path from v_s to v_j. For that reason, there may exist many paths with the same length, time and cost of travel where we run to v_j by bus of the same bus line. In this case the list $PS[v_j]$ contains only a single record which represents all these paths. It reduces the number of stored and analysed solutions. Thus, before creating a new record representing the path we check using the procedure CheckSolution if the record P_j, where the time and the cost of travel are equal t_{sj} and c_{sj}, the length of the route equals d_{sj} and where we run to v_j by bus of the line l_j, exists in the list $PS[v_j]$ (line 18). If P_j exists we only add a record $(v_i, T_i, \&P)$ to the list $P_j \cdot LP$ stored in P_j (line 20). Otherwise, a new record P' is created (line 22) and it is added to $PS[v_j]$ (lines 24, 26). If P' is created for the final vertex v_e, there is used the procedure AddToFinalList (line 24) which add P' to the list $PS[v_j]$ and removes dominated solutions. Otherwise, there is used the procedure AddToList and v_j is added to the queue Q (line 26). We finish the computation in the **while**–loop when the queue Q is empty and it means that we have searched a space of solutions.

In the last part of the algorithm we use the procedure CreateFullPaths to determine full routes for all computed solutions represented by records stored in the list $PS[v_e]$ (line 34). The full route is determined based on the values stored in the record, i.e. the vertex v_i which precedes the vertex v_e in the path and the pointer $*P_i$ to the partial solution for v_i. In the last step of the algorithm the paths, that differ from each other only in times of departure from all vertices belonging to the path, are determined using the procedure PathsDifferTimes. In the procedure all paths belonging to the S list are analysed (line 34). Let us consider the path $P \in S$ containing a sequence vertices of changes: $v_0 = v_s, v_1, \ldots, v_{k-1}, v_k = v_e$, where times of departure from these vertices are equal respectively: $T_0, T_1, \ldots, T_{k-1}$. A new solution P' is created and added to the list S if one can leave v_i $(0 \leqslant i < k-1)$ at any time later than T_i and arrive to v_e at the same time as in P. The solution P' differs from P only in times of departure from vertices belonging to the path, and as a consequence differs in times of waiting for changes, but total times of waiting in both paths are equal.

The extended solution represents a path from v_s to v_j, its length equals d_{sj}, the time and the cost of travel equal t_{sj} and c_{sj} and it is estimated if it is possible to obtain a non-dominated final solution from it. The estimation is made on the basis of determined paths P_t, P_c, P_d and values $\overline{c}^e_m[v_j]$, $\overline{t}^e_m[v_j]$, $\overline{d}^e_m[v_j]$ using the procedure NotDom (line 17). Let t_{max} equals the maximal time, c_{max} equals the maximal cost of travel and d_{max} equals the maximal length from among paths P_t, P_c and P_d, what is described by (17)–(19).

$$t_{\max} = \max\{T(P_t), T(P_c), T(P_d)\}, \tag{17}$$

$$c_{\max} = \max\{C(P_t), C(P_c), C(P_d)\}, \tag{18}$$

$$d_{\max} = \max\{D(P_t), D(P_c), D(P_d)\}. \tag{19}$$

The minimal time of travel from v_j to v_e equals $\overline{t}^e_m[v_j]$, for that reason the value of the time of travel t_{se} the final solution obtained from the extended solution is defined by (20). In the same way are defined the cost of travel c_{se} and the length of the path d_{se} (21), (22).

$$t_{se} \geqslant t_{sj} + \overline{t}^e_m[v_j], \tag{20}$$

$$c_{se} \geqslant c_{sj} + \overline{c}^e_m[v_j], \tag{21}$$

$$d_{se} \geqslant d_{sj} + \overline{d}^e_m[v_j]. \tag{22}$$

It is not possible to obtain a non-dominated final solution from the extended solution if one of conditions (23)–(25) is satisfied. These conditions are checked in the procedure NotDom. Additionally, there is checked if paths P_t, P_c, P_d dominate the final solution obtained from the extended solution. It is checked by comparing values $t_{sj}+\overline{t}^e_m[v_j]$, $c_{sj}+\overline{c}^e_m[v_j]$ and $d_{sj}+\overline{d}^e_m[v_j]$ with the time and the cost of travel and the length of paths P_t, P_c, P_d using Definition 1.

$$t_{sj}+\overline{t}^e_m[v_j] > t_{\max} \;\wedge\; c_{sj}+\overline{c}^e_m[v_j] \geqslant c_{\max} \;\wedge\; d_{sj}+\overline{d}^e_m[v_j] \geqslant d_{\max}, \tag{23}$$

$$t_{sj}+\overline{t}^e_m[v_j] \geqslant t_{\max} \;\wedge\; c_{sj}+\overline{c}^e_m[v_j] > c_{\max} \;\wedge\; d_{sj}+\overline{d}^e_m[v_j] \geqslant d_{\max}, \tag{24}$$

$$t_{sj}+\overline{t}^e_m[v_j] \geqslant t_{\max} \;\wedge\; c_{sj}+\overline{c}^e_m[v_j] \geqslant c_{\max} \;\wedge\; d_{sj}+\overline{d}^e_m[v_j] > d_{\max}. \tag{25}$$

The procedure AddToList adds the record P' representing the path p'_{v_s,v_j} to the list $PS[v_j]$ (line 26) and estimates it based on already existing records. It is possible to determine whether the final solution p'_{v_s,v_e} obtained from P' is dominated. Let P'' is already existing record belonging to $PS[v_j]$ and it represents the path p''_{v_s,v_j}. If (26) is satisfied, it is possible to obtain the final solution p''_{v_s,v_e} from p''_{v_s,v_j}, where the time of departure from v_j is the same as in p'_{v_s,v_e} and it follows that the time of travel meets (27).

$$T(p'_{v_s,v_j}) = T(p''_{v_s,v_j}), \tag{26}$$

$$T(p'_{v_s,v_e}) = T(p''_{v_s,v_e}). \tag{27}$$

The relationships between the lengths of the paths p'_{v_s,v_j} and p''_{v_s,v_j} and the lengths of obtained the final solutions p'_{v_s,v_e} and p''_{v_s,v_e} are described by (28)–(30).

$$D(p'_{v_s,v_j}) > D(p''_{v_s,v_j}) \quad\Rightarrow\quad D(p'_{v_s,v_e}) > D(p''_{v_s,v_e}), \tag{28}$$

$$D(p'_{v_s,v_j}) = D(p''_{v_s,v_j}) \quad\Rightarrow\quad D(p'_{v_s,v_e}) = D(p''_{v_s,v_e}), \tag{29}$$

$$D(p'_{v_s,v_j}) < D(p''_{v_s,v_j}) \quad\Rightarrow\quad D(p'_{v_s,v_e}) < D(p''_{v_s,v_e}). \tag{30}$$

If the costs of travel meet (31) then (32) is satisfied and if (28) or (29) is satisfied then P' is not added to the list $PS[v_j]$ because the final solution p'_{v_s,v_e} obtained from P' is dominated by p''_{v_s,v_e}. Otherwise, i.e. if (33) is satisfied, it is not possible to determine the relationship between $C(p'_{v_s,v_e})$ and $C(p''_{v_s,v_e})$ and P' is added to $PS[v_j]$.

$$C(p'_{v_s,v_j}) > C(p''_{v_s,v_j}), \tag{31}$$

$$C(p'_{v_s,v_e}) > C(p''_{v_s,v_e}), \tag{32}$$

$$C(p'_{v_s,v_j}) \leqslant C(p''_{v_s,v_j}). \tag{33}$$

If (34) is satisfied, then (35) is also satisfied and we do not add P' to $PS[v_j]$ only when (28) or (29) occur and additionally (36) occurs. If (28) or (29) and (36) are satisfied then p''_{v_s,v_e} dominates p'_{v_s,v_e}.

$$T\big(p'_{v_s,v_j}\big) > T\big(p''_{v_s,v_j}\big), \tag{34}$$

$$T\big(p'_{v_s,v_e}\big) \geqslant T\big(p''_{v_s,v_e}\big), \tag{35}$$

$$C\big(p'_{v_s,v_j}\big) > C\big(p''_{v_s,v_j}\big). \tag{36}$$

In the last case, (37) occurs and it follows that (38) is satisfied. In the case when it is not possible to determine the relationship between $C(p'_{v_s,v_e})$ and $C(p''_{v_s,v_e})$, we have to add P' to the list $PS[v_j]$.

$$T\big(p'_{v_s,v_j}\big) < T\big(p''_{v_s,v_j}\big), \tag{37}$$

$$T\big(p'_{v_s,v_e}\big) \leqslant T\big(p''_{v_s,v_e}\big). \tag{38}$$

To sum up, the record P' is not added to the list $PS[v_j]$ when (39) are satisfied and these conditions are used in the procedure AddToList.

$$T\big(p'_{v_s,v_j}\big) \geqslant T\big(p''_{v_s,v_j}\big) \wedge C\big(p'_{v_s,v_j}\big) > C\big(p''_{v_s,v_j}\big) \wedge D\big(p'_{v_s,v_j}\big) \geqslant D\big(p''_{v_s,v_j}\big). \tag{39}$$

If (40) and (41) are satisfied, then P' is dominated by P'' but it is added to the list $PS[v_j]$. Thus, the list $PS[v_j]$ of the vertex $v_j \neq v_e$ can contains records being dominated solutions and a non-dominated final solution for the final vertex v_e can be obtained from a dominated partial solution for $v_j \neq v_e$. This is an important property of a graph with variable weight and it is not satisfied in a graph with constant weight. It has been shown, that it is not possibly to obtain a non-dominated final solution from a dominated partial solution if weights are constant (Azevedo and Martins, 1991; Martins *et al.*, 1999; Mote *et al.*, 1991).

$$T\big(p'_{v_s,v_j}\big) > T\big(p''_{v_s,v_j}\big) \wedge C\big(p'_{v_s,v_j}\big) = C\big(p''_{v_s,v_j}\big) \wedge D\big(p'_{v_s,v_j}\big) \geqslant D\big(p''_{v_s,v_j}\big), \tag{40}$$

$$T\big(p'_{v_s,v_j}\big) \geqslant T\big(p''_{v_s,v_j}\big) \wedge C\big(p'_{v_s,v_j}\big) = C\big(p''_{v_s,v_j}\big) \wedge D\big(p'_{v_s,v_j}\big) > D\big(p''_{v_s,v_j}\big). \tag{41}$$

A time complexity of the algorithm FindRoutes depends on a number of paths belonging to the set of non-dominated solutions. The MSP problem is known to be NP-complete and it has been shown that the number of non-dominated solutions is exponential at least in the worst case which implies that any deterministic algorithm that attempts to solve it is also exponential in terms of pessimistic time complexity. The maximum number of non-dominated solutions equals g^{n-1}, where n is a number of vertices and g is a maximum out-degree of a vertex in the graph (Skriver and Andersen, 2000b). A time complexity of the algorithm is clearly dominated by execution of the **while**-loop (lines 9–33). Each of computed paths can contain at most n vertices. In the line 10 the vertex v_i is taken from the queue Q and g_i arcs are examined, where g_i is the out-degree of the vertex v_i. Thus, in the worst case it is necessary to examine $\prod_{i=1}^{n-1} d_i$ arcs and the pessimistic time complexity equals $O(d^n)$, where $g = \max_{i=1,\dots,n-1}\{d_i\}$.

Table 5
Parameters of the graphs representing the bus network.

	G	G_a, G_{ar}
Number of vertices	1211	1211
Number of arcs	5549	35 610
Max. in-degree of vertex	13	122
Max. out-degree of vertex	13	122
Min. in-degree of vertex	1	6
Min. out-degree of vertex	1	6
Max. degree of vertex	26	244
Min. degree of vertex	2	13

4. Experimental results

The procedure FindRoutes (Fig. 5) has been implemented in C++ and the test experiments were carried out on 2.66 GHz Pentium-IV computer with 1 GB of RAM, running under Windows Server 2003 Enterprise Edition. In addition we have adapted the Brumbaugh-Smith (Brumbaugh-Smith and Shier, 1989) and the Skriver (Skriver and Andersen, 2000b) algorithms to variable weights and both algorithms have been implemented in C++. In the tests we used a bus network consisting of 1211 stops divided into 26 zones where buses of 500 bus lines are run. The longest length of the route of the bus line equals 29 and the shortest length equals 6. The network is represented by graphs G, G_a, G_{ar} and their parameters are described in Table 5.

The main goal of the experimental tests was to compare the computation time using the tested algorithms and it depends on the number of computed partial solutions, therefore it was counted in the tests. The second goal of the tests was to investigate an impact of the number of criterion functions on the computation time and the number of computed partial solutions. For this purpose, implemented algorithms have been modified to solve the BRP where only two criterion functions are used, i.e. the time and the cost of travel. In addition we have modified a record stored in the list $PS[v_j]$ and the record $(l_j, t_{sj}, c_{sj}, v_i, T_i, *P_i)$ was used. The modified record does not contain the list *LP*, instead, it contains the pointer $*P_i$ to the partial solution for the vertex v_i and when the estimation is positive (line 17) we create a new record and add it to $PS[v_j]$. Thus, each path from v_s to v_j is represented by separate record in $PS[v_j]$. It increases the number of stored and analysed solutions. The goal of this modification was testing an impact of the form of the record representing the solution on the computation time and the number of computed partial solutions.

We have carried out 1371616 tests using the procedure FindRoutes in both versions and 673719 tests using the Skriver algorithm, where two criterion functions are considered, and 220425 tests for version with three criterion functions. It was not possible to carry out the same number of tests for the Skriver algorithm as to the procedure FindRoutes and the same number of tests for the both versions of Skriver algorithm due to out of memory. For the same reason, all tests have failed for the Brumbaugh-Smith algorithm, which determines all existing routes from v_s to v_e. The aim of a single test was finding all routes belonging to the set of non-dominated solutions for the given pairs of vertices v_s

Table 6
The number of all computed non-dominated solutions using the FindRoutes and the Skriver algorithms depending on the number of changes.

Number of	Criterion functions: T, C		Criterion functions: T, C, D	
changes	FindRoutes	Skriver	FindRoutes	Skriver
0	64771	61169	66060	53513
1	553605	503427	596898	221679
2	12217636	1771642	2783846	365116
3	5774482	3136625	8509761	440584
4	13311829	3328943	23268867	436417
5	27485412	1405485	57994949	269344
6	24613443	197354	112657304	197415
7	2965135	41990	170024075	108229
8	00521063	10783	275566765	62240
9	195336	1086	362479435	10745
10	65372	162	481644186	2308
11	14350		680149204	
12			754155025	
13			708535381	
14			694029918	
15			641701731	
16			573463728	
17			335616072	
18			162101575	
19			63804321	
20			13519427	
21			914334	

and v_e and the time of starting travel T_s. We present all results depending on the length of the route and the number of changes in the route.

The results of test experiments using the procedure FindRoutes and the Skriver algorithm have been presented in Table 6–11. We have shown the number of all computed non-dominated solutions during all experiments (Table 6 and Table 9), the maximal number of computed partial solutions during a single experiment (Table 7 and Table 10) and a maximal computation time in hrs:mins:secs format (Table 8 and Table 11). The results of tests using the procedure FindRoutes presented in Section 3 and the Skriver algorithm, where three criterion functions are considered, are denoted by "criterion functions: T, C, D" and "criterion functions: T, C" denotes results of tests using modified algorithms where only two criterion functions are considered. All records stored in the lists $PS[v_j]$ are treated as computed partial solutions. The results, where the partial solution is represented by a record containing the list *LP*, have been denoted by *LP* and the results, where the modified record has been used, have been denoted by $*P$.

In both cases, i.e. solving the BRP where two and three criterion functions are used, the maximal number of partial solutions was computed using the Skriver algorithm. In the first case, it is about 419 times more than the maximal number of computed partial solutions by the procedure FindRoutes, where the modified record has been used and it is about 5295 times more if the record containing the list *LP* was used. In the second case, these values are equal respectively to 315 and 1468. The Skriver algorithm does

Table 7
The maximal number of computed partial solutions in a single experiment using the FindRoutes and the Skriver algorithms depending on the number of changes.

Number of changes	Criterion functions: T, C			Criterion functions: T, C, D		
	FindRoutes		Skriver	FindRoutes		Skriver
	$*P$	LP		$*P$	LP	
0	1491	794	18957108	1073	814	43614321
1	7582	2143	3011897	7015	2884	44384030
2	19113	4217	47126308	19231	8233	44389232
3	56434	5919	51243332	64751	16505	44389232
4	71417	9211	52400859	74420	20146	44389232
5	105666	9708	52546673	111912	26447	44384030
6	107024	9924	52546673	140716	30240	44384030
7	107024	9924	52546673	140716	30240	44244484
8	107024	9924	51630198	140716	30240	44384030
9	95361	9924	50734519	140716	30240	42006160
10	105666	9924	694076	140716	30240	35940676
11	73423	9924		140716	30240	
12	61973	8233		140716	30240	
13	24838	6281		140716	30240	
14				140716	30240	
15				140716	29533	
16				140716	28261	
17				140385	26346	
18				137378	26511	
19				140716	26511	
20				140716	24793	
21				57428	21288	

not apply conditions for estimation of the partial solutions that are applied by the procedure FindRoutes. Therefore, a larger space of solutions by the Skriver algorithm than by the procedure FindRoutes is searched and the Skriver algorithm computes a larger number of partial solutions. For that reason, the Skriver algorithm has a larger memory complexity and it was not possible to carry out the same number of experiments as to the procedure FindRoutes. The computation time depends on a searched space of solutions by the algorithm, therefore the computation time is larger for the Skriver algorithm.

With the increase of the length of the route and the number of changes, a searched space of solutions grows. For that reason, the number of computed partial solutions is growing with the increase of the number of changes and the length of the route. In the first case, i.e. in the BRP where two criterion functions are used, using the record with the list LP in the procedure FindRoutes decreases about 10 times the number of computed partial solutions and it decreases the computation time. In the second case, the number of computed solutions is decreased 5 times.

Additionally, there was tested an influence of the number of criterion functions on the computation time and the number of computed partial solutions. Let us consider solution P_1 and P_2 for which $T(P_1) > T(P_2)$, $C(P_1) > C(P_2)$ and $D(P_1) < D(P_2)$ are satisfied. In the first case, when only two criterion functions T and C are taken into account then P_2 dominates P_1. In the second case, when in addition the third criterion function D is taken

Table 8
The maximal computation time in a single experiment using the FindRoutes and the Skriver algorithms depending on the number of changes, the unit of the time is [hrs:mins:secs].

Number of changes	Criterion functions: T, C			Criterion functions: T, C, D		
	FindRoutes		Skriver	FindRoutes		Skriver
	$*P$	LP		$*P$	LP	
0	0:00:00	0:00:00	0:02:35	0:00:00	0:00:00	0:41:30
1	0:00:00	0:00:00	0:00:14	0:00:02	0:00:01	1:19:48
2	0:01:00	0:00:00	0:59:49	0:00:15	0:00:07	1:26:29
3	0:00:00	0:00:00	3:26:19	0:01:53	0:00:57	1:26:29
4	0:00:00	0:00:00	5:44:07	0:02:24	0:00:59	1:26:29
5	0:00:00	0:00:00	5:44:07	0:02:33	0:00:59	1:26:29
6	0:00:32	0:00:09	5:44:07	0:02:43	0:00:59	1:26:29
7	0:00:32	0:00:09	3:51:26	0:02:43	0:00:59	1:02:09
8	0:00:06	0:00:04	3:51:26	0:02:43	0:00:59	1:09:44
9	0:00:06	0:00:04	3:51:26	0:02:43	0:00:59	0:46:02
10	0:00:00	0:00:00	0:00:04	0:02:43	0:00:59	0:18:28
11	0:00:00	0:00:00		0:02:43	0:00:59	
12	0:00:00	0:00:00		0:02:43	0:00:59	
13	0:00:00	0:00:00		0:02:43	0:00:59	
14				0:02:43	0:00:59	
15				0:02:43	0:00:59	
16				0:02:43	0:00:59	
17				0:02:32	0:00:59	
18				0:02:24	0:00:57	
19				0:02:24	0:00:57	
20				0:02:11	0:00:51	
21				0:01:52	0:00:51	

Table 9
The number of all computed non-dominated solutions using the FindRoutes and the Skriver algorithms depending on the length of the route.

Length of the route	Criterion functions: T, C		Criterion functions: T, C, D	
	FindRoutes	Skriver	FindRoutes	Skriver
1–5	153204	139158	173746	110206
6–10	517938	461066	1731971	552477
11–15	2217636	841915	7420803	919132
16–20	1740571	1265327	25055825	563973
21–25	2756929	1607326	85607310	21801
26–30	4075188	1649887	358860843	1
31–35	5503521	1539983	931244636	
36–40	7258514	1276342	2126130693	
41–45	9772586	905702	2165955723	
46–50	11845671	461975	399262866	
51–55	13029210	199495	15438041	
56–60	9471339	71627	4577491	
61–65	6262071	30175	1720131	
66–70	3096544	7763	355638	
71–75	1006427	890	43505	
76–80	249627	14	3434	
81–85	26952		206	
86–90	2226			

Table 10
The maximal number of computed partial solutions in a single experiment using the FindRoutes and the Skriver algorithms depending on the length of the route.

Length of the route	Criterion functions: T, C			Criterion functions: T, C, D		
	FindRoutes		Skriver	FindRoutes		Skriver
	$*P$	LP		$*P$	LP	
1–5	1097	701	90169	764	611	341385
6–10	4229	1210	1496675	2681	1835	33140590
11–15	8801	2118	26578616	6544	3401	44169271
16–20	21273	3589	26578616	18174	7496	44389232
21–25	33918	5071	50138560	24757	10451	44389232
26–30	48616	7644	52400859	47360	18770	2050394
31–35	70023	9211	52546673	89688	24000	
36–40	88899	9708	52546673	137378	27774	
41–45	92482	9708	52546673	140716	30240	
46–50	95361	9924	52546673	140716	30240	
51–55	98962	9924	52546673	140716	30240	
56–60	107024	9924	52546673	140716	30240	
61–65	107024	9924	52546673	140716	30240	
66–70	107024	9924	49217090	140716	30240	
71–75	107024	9924	49217090	134626	30240	
76–80	84174	9924	6982760	83661	27959	
81–85	81987	9924		79517	24352	
86–90	55883	7405				

Table 11
The maximal computation time in a single experiment using the FindRoutes and the Skriver algorithms depending on the length of the route, the unit of the time is [hrs:mins:secs].

Length of the route	Criterion functions: T, C			Criterion functions: T, C, D		
	FindRoutes		Skriver	FindRoutes		Skriver
	$*P$	LP		$*P$	LP	
1–5	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:11
6–10	0:00:00	0:00:00	0:00:06	0:00:00	0:00:00	0:14:57
11–15	0:01:00	0:00:00	0:04:50	0:00:01	0:00:00	0:57:26
16–20	0:00:00	0:00:00	0:04:50	0:00:07	0:00:01	1:26:29
21–25	0:00:00	0:00:00	3:14:12	0:00:20	0:00:08	1:26:29
26–30	0:00:00	0:00:00	3:26:19	0:02:31	0:00:54	
31–35	0:00:00	0:00:00	3:45:59	0:02:31	0:00:59	
36–40	0:00:06	0:00:03	4:39:16	0:02:31	0:00:59	
41–45	0:00:32	0:00:09	5:44:07	0:02:43	0:00:59	
46–50	0:00:32	0:00:09	5:44:07	0:02:43	0:00:59	
51–55	0:00:32	0:00:09	4:39:16	0:02:43	0:00:59	
56–60	0:00:32	0:00:09	3:51:26	0:02:43	0:00:59	
61–65	0:00:32	0:00:09	3:51:26	0:02:43	0:00:59	
66–70	0:00:01	0:00:01	3:51:26	0:02:24	0:00:59	
71–75	0:00:01	0:00:01	3:51:26	0:02:32	0:00:54	
76–80	0:00:00	0:00:00	0:01:57	0:02:06	0:00:43	
81–85	0:00:00	0:00:00		0:00:08	0:00:07	
86–90	0:00:00	0:00:00				

into account then P_1 and P_2 are non-dominated solutions. For that reason, the number of computed partial solution in the second case is greater than in the first and it increases the computation time. This was the reason why the different number of tests was carried out by the Skriver algorithm in both cases. It was not possible to carry out the same number of tests due to out of memory. Adding the third criterion function increases about 3 times the number of computed partial solutions by the procedure FindRoutes where the record with the list *LP* is used. For the second version of the procedure it increases the number of computed partial solutions about 1.5 times.

The number of criterion functions increases the number of non-dominated solutions. In the first case, i.e. in the BRP where two criterion functions are used, the maximal number of non-dominated solutions in the single test was 461440 but only 385 solutions differ from each other in the path, other solutions were the same path and differ from each other only in the times of departure. In the second case, these values are equal 3162693 and 66 respectively. Thus adding the length of the route as the third criterion function increases about 7 times the number of non-dominated solutions.

5. Conclusions

In this paper we considered the bus routing problem (BRP), in particular, we analysed the problem and properties of a route. The goal of the problem is to find a route from the given start stop to the given final stop minimizing the time and the cost of travel and the length of the route, where the time of starting travel at the start stop is given additionally. The BRP is an example of the multicriteria shortest path (MSP) problem with variable weights which the solution is the set of non-dominated solutions. The MSP problem has already been considered in the literature and several algorithms have been proposed, however, these algorithms assume constant weights. We have analysed differences between the MSP problem with variable and constant weights. Additionally, the results of solving the BRP where the goal is to minimize only the time and the cost of travel are presented.

In the paper we proposed a label correcting algorithm with storing partial solutions for solving the BRP. The algorithm makes it possible to find all routes belonging to the set of non-dominated solutions. As shown, the MSP is known to be NP-complete and in the worst case, the number of non-dominated solutions grows exponentially with the n value. The pessimistic time complexity of the algorithm is determined by the maximal number of non-dominated solutions and it equals $O(d^n)$, where n equals a number of vertices in the graph representing the bus network and g equals a maximum out-degree of a vertex in the graph.

On the basis of the results of experimental tests we found that presented algorithm exhibits a reasonable execution time for the bus network containing about 1200 stops. Additionally, the results demonstrate that the number of non-dominated solutions is not exponential in practice. In the algorithm we take into consideration only three criterion functions representing the time and the cost of travel and the length of the route but it is possible to add other criterion functions.

References

Azevedo, J.A., Martins, E.Q.V. (1991). An algorithm for the multiobjective shortest path problem on acyclic networks. *Investigação Operacional*, 11(1), 52–69.

Bellman, R. (1958). On a routing problem. *Quarterly of Applied Mathematics*, 16(1), 87–90.

Brumbaugh-Smith, J., Shier, D. (1989). An empirical investigation of some bicriterion shortest path algorithms. *European Journal of Operational Research*, 43(2), 216–224.

Chartrand, G., Lesniak, L., Zhang, P. (2010). *Graphs & Digraphs*. 5th edition, Chapman & Hall/CRC, Boca Raton.

Climaco, J.C. Martins, E.Q.V. (1982). A bicriterion shortest path algorithm. *European Journal of Operational Research*, 11(4), 399–404.

Coello Coello, C.A. van Veldhuizen, D.A. Lamont, G.B. (2002). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic, New York.

Corley, H.W., Moon, I.D. (1985). Shortest paths in networks with vector weights. *Journal of Optimization Theory and Application*, 46(1), 79–86.

Daellenbach, H.G., De Kluyver, C.A. (1980). Note on multiple objective dynamic programming. *Journal of the Operational Research Society*, 31(7), 591–594.

Dell'Olmo, P., Gentili, M., Scozzari, A. (2005). On finding dissimilar Pareto-optimal paths. *European Journal of Operational Research*, 162(1), 70–82.

Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. *Numerical Mathematics*, 1, 269–271.

Ehrgott, M. (1999). Multicriteria optimization. *Vorlesungsskripte, Sonstiges*. Department of Mathematics, University of Kaiserslautern, Germany.

Ehrgott, M., Gandibleux, X. (1999). A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 22(4), 425–460.

Floyd, R. (1962). Algorithm 97: Shortest path. *Communications of the ACM*, 5(6), 345.

Ford, L.R. (1956). *Network Flow Theory*. Report P-923, The Rand Corporation, Santa Monica.

Garey, M., Johnson, D. (1990). *Computers and Intractibility: A Guide to the Theory of NP-Completeness*. Freeman, New York.

Hansen, P. (1980). *Bicriterion path problems*. Multiple Criteria Decision Making: Theory and Application, Springer, Berlin, pp. 109–127.

Henig, M.I. (1985). The shortest path problem with two objective functions. *European Journal of Operational Research*, 25(2), 281–291.

Johnson, D.B. (1977). Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM*, 24(1), 1–13.

Jungnickel, D. (1999). *Graphs, Networks and Algorithms*. 2nd edition, Springer, Berlin.

Korhonen, P. (1992). Multiple criteria decision support – a review. *European Journal of Operational Research*, 63(3), 361–375.

Machuca, E., Mandow, L., Pérez de la Cruz, J.L. (2009). An evaluation of heuristic functions for bicriterion shortest path problems. In: *Proceedings of the XIV Portuguese Conference on Artificial Inteligence (EPIA 2009)*, Universidade de Aveiro, Portugal, pp. 205–216.

Mandow, L., Pérez-de-la-Cruz, J.L. (2008a). Frontier search for bicriterion shortest path problems. In: *Proceeding of the 2008 Conference on ECAI 2008 18th European Conference on Artificial Intelligence*, pp. 480–484.

Mandow, L., Pérez-de-la-Cruz, J.L. (2008b). Path recovery in frontier search for multiobjective shortest path problems. *Journal of Intelligent Manufacturing*, 21(1), 89–99.

Martí, R., González Velarde, J.L., Duarte, A. (2009). Heuristics for the bi-objective path dissimilarity problem. *Computers & Operations Research*, 36(11), 2905–2912.

Martins, E.Q.V. (1984). On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2), 236–245.

Martins, E.Q.V., Pascoal, M.M.B., Rasteiro, D.M.L.D., Santos. J.L.E. (1999). The optimal path problem. *Investigação Operacional*, 19, 43–60.

Mote, J., Murthy, I., Olson, D.L. (1991). A parametric approach to solving bicriterion shortest path problems. *European Journal of Operational Research*, 53(1), 81–82.

Pareto, V. (1896). *Course d'Economie Politique*. Rouge, Lausanne.

Raith, A., Ehrgott, M. (2009). A comparison of solution strategies for biobjective shortest path problems. *Journal Computers and Operations Research*, 36(4), 1299–1331.

Roy, B., Vincke, P. (1981). Multicriteria analysis: survey and new directions. *European Journal of Operational Research*, 8(3), 207–218.

Skriver, A.J.V., Andersen, K.A. (2000a). A classification of bicriteria shortest path (BSP) algorithms. *Asia-Pacific Journal of Operational Research*, 17, 199–212.

Skriver, A.J.V., Andersen, K.A. (2000b). A label correcting approach for solving bicriterion shortest-path problems. *Computers & Operations Research*, 27(6), 507–524.

Tung, C.T., Chew, K.L. (1988). A bicriterion Pareto-optimal path algorithm. *Asia-Pacific Journal of Operational Research*, 5, 166–172.

Tung, C.T., Chew, K.L. (1992). A bicriterion Pareto-optimal path algorithm. *European Journal of Operational Research*, 62(2), 203–209.

Ulungu, E.L., Teghem, J. (1994). Multi-objective combinatorial optimization problems: a survey. *Journal of Multi-Criteria Decision Analysis*, 3(2), 83–104.

Voorneveld, M. (2003). Characterization of Pareto dominance. *Operations Research Letters*, 31(1), 7–11.

Warshall, S. (1962). A theorem on Boolean matrices. *Journal of the ACM*, 9(1), 11–12.

Wilson, R.J. (1996). *Introduction to Graph Theory*. 4th edition, Addison-Wesley, Reading.

J. Widuch was awarded the doctor of engineering sciences degree at the Silesian University of Technology, Institute of Informatics in 2008. At present works as an assistant professor in the Institute of Informatics of Silesian University of Technology. His main research interest is connected with multicriteria optimization algorithms in transportation problems and parallel computing. He is a member of Upper Silesian Regional Committee of Polish Olympiad in Informatics.

Autobusų maršrutų sudarymo algoritmas pagrįstas žymeklio koregavimu ir dalinių sprendinių įsiminimu

Jacek WIDUCH

Straipsnyje aprašytas autobusų maršrutų sudarymo (BRP) algoritmas, kurio tikslas surasti maršrutą nuo pradinės iki galinės stotelės minimizuojant laiką, kelionės kainą ir ilgį. Kelionės pradžios laikas pradinėje stotelėje yra duotas. Išanalizavus uždavinį pateiktos maršrutų savybės. BRP yra daugiakriterės optimizacijos uždavinys, kurio sprendiniais yra nedominuojami sprendiniai. Šiame straipsnyje pasiūlytas žymeklio koregavimo algoritmas išsaugantis dalinius sprendinius. Algoritmu randami visi maršrutai priklausantys nedominuojamų sprendinių aibei. Taip pat pateikti eksperimentinio testavimo rezultatai. Gauti rezultatai palyginti su BRP uždavinio sprendimo rezultatais, kai minimizuojamas tik laikas, ar tik kaina, ar tik maršruto ilgis.