Netw Spat Econ (2013) 13:67–106
DOI 10.1007/s11067-012-9173-3

A Branch and Bound Algorithm for Bi-level Discrete Network Design Problem

Hamid Farvaresh · Mohammad Mehdi Sepehri

Published online: 17 June 2012

Abstract Discrete network design problem (DNDP) is generally formulated as a bi-level programming. Because of non-convexity of bi-level formulation of DNDP which stems from the equilibrium conditions, finding global optimal solutions are very demanding. In this paper, a new branch and bound algorithm being able to find exact solution of the problem is presented. A lower bound for the upper-level objective and its computation method are developed. Numerical experiments show that our algorithm is superior to previous algorithms in terms of both computation time and solution quality. The conducted experiments indicate that in most cases the first incumbent solution which is obtained within a few seconds is superior to the final solution of some of previous algorithms.

Keywords Network design problem · Bi-level programming · Branch and bound · Outer approximation

1 Introduction

The network design problem (NDP) concerns with modifying a transportation network (infrastructure) configuration by adding new links or improving existing ones, so that certain social welfare objectives (e.g. total travel time over the network) are maximized. How to locate new links and how to increase the capacity of existing links are motivating problems. The overall objective is to minimize the total system costs under limited budget, while accounting for the route choice behavior of network

H. Farvaresh
Department of Industrial Engineering, University of Kurdistan, Sanandaj, Iran

M. M. Sepehri (✉)
Department of Industrial Engineering, Tarbiat Modares University, Tehran, Iran
e-mail: mehdi.sepehri@modares.ac.ir

Mohammad. Mehdi. Sepehri
e-mail: mehdi.sepehri@gmail.com

users. NDP can be roughly classified into three categories: the discrete network design problem (DNDP) dealing with adding new links or road-way segments to an existing road network; the continuous network design problem (CNDP) dealing with the optimal capacity expansion of a subset of existing links; and the mixed network design problem (MNDP) combining both CNDP and DNDP in a network (Yang and Bell 1998). The importance of the NDP problem has been highlighted in both academic and practical literature. From a practical view, NDP becomes more important as increasing populations coupled with economic growth produce travel demand exceeding the existing capacity of transportation infrastructures. Meanwhile, resources available to enhance capacity remain limited. Therefore, enhancing capacity needs a great deal of investments that could be hardly met in developing countries. Accordingly, it is urgent to study how to efficiently allocate limited investment to improve transport efficiency and maximize the social and economic benefits.

From an optimization point of view, the NDP can be viewed as a hierarchical decision making problem that includes system planners in the upper-level and users in the lower-level. System planners try to influence users' choices by adding or expanding some links to minimize total system costs. Total costs of the system are affected by decision variables of both system planners and users (LeBlanc and Boyce 1986). The partition of the control over the decision variables between two hierarchical levels requires the formulation of the NDP as a bi-level programming problem (BLPP). In the resulted BLPP the system planner makes decisions about network configuration to improve the performance of the system, and the network users make choices about the routes of their travel in response to the upper-level decision. Since users are assumed to make their choices to maximize their individual utility functions, their choices do not necessarily align, and often conflict, with the choices that are optimal for the system planners. Lower-level problem is generally described by a user equilibrium model.

Previous studies give practical evidence implying that even small and experimental scale NDPs are difficult to solve (Gao et al. 2005; Magnanti and Wong 1984). There is also theoretical evidence supporting these observations since the BLPP is NP-hard even in its linear form. Because of the intrinsic complexity of NDP in the form of bi-level formulation, the problem has been recognized as one of the most difficult ones, yet challenging problems for global optimality in transportation. This often leads to computation times too high for practical purposes. Due to the practical importance of NDP, many heuristic/meta-heuristic algorithms to tackle it have been employed. Meta-heuristic algorithms have enjoyed a considerable popularity in the last decades (Poorzahedy and Abulghasemi 2005; Poorzahedy and Rouhani 2007). The guarantee of finding an optimal solution in heuristic/meta-heuristic algorithms is sacrificed for the sake of a good solution in a significantly reduced amount of time.

Although good progress and algorithms have been achieved during past decades, currently the most promising solution methods dealing with realistic scale problems are heuristics/meta-heuristics which do not guarantee the optimality or near-optimality of the solution. On the other hand, some global optimization approaches have been proposed based on converting the bi-level form to a single-level one and exploiting decomposition methods (Gao et al. 2005; LeBlanc 1975). These approaches need a lot of computation time.

This paper proposes a new branch and bound algorithm for the bi-level DNDP being able to provide an optimal solution for the real-scale network design problems. To cope with explicit or implicit path enumeration, the link-node network representation with multicommodity flows is employed. The main motivation is presenting an algorithm being able to find a global optimal solution for medium to large-scale DNDP while guarding against Braess' paradox.

The remainder of this paper is organized as follows: Section 2 reviews the literature on network design problems. The proposed formulation is then described in Section 3. In Section 4, the numerical results on four test problems are reported. Finally, concluding remarks are offered in Section 5.

2 Literature review

LeBlanc (1975) made a pioneer study and proposed a branch-and-bound algorithm for solving DNDP. The bounding step was designed so that the Braess' paradox could not occur. This algorithm is relatively inefficient and becomes computationally prohibitive, even for small networks, in cases where there is a relatively large number of new links to add to the network. Our algorithm could be considered as an improvement of LeBlanc's B&B algorithm whose most building blocks including bounding, branching, fathoming, first incumbent, and some other minor parts have been improved. Poorzahedy and Turnquist (1982) developed a branch and bound algorithm for solving the integer programming model of the DNDP by replacing the objective function with another well-behaved function, which makes the problem a convex nonlinear mixed-integer program. They argued that since a strong linear relationship between the system and user optimal objective function exists, the system optimal objective function could be approximately replaced with the user optimal one. However, in situations where severe traffic congestion exists in the network, a significant gap appears between the mentioned functions. In addition, in mathematical programming, even in the presence of exact linear relationships between two objective functions, there is no guarantee that both objective functions lead to the same solution even if their objective values are very close to each other.

Steinbrink (1974a), Boyce (1984), Freisz (1985), and Magnanti and Wong (1984) conducted surveys on the modeling and algorithmic development for mathematical programming based NDP. Specifically, Steenbrink (1974a) reviewed the branch-and-bound techniques for solving DNDP. He presented an introduction to modeling DNDP and discussed techniques for the traffic assignment and Braess' paradox. He proposed a method to approximate user equilibrium flows by the system optimal flows using an iterative decomposition algorithm. In their comprehensive survey paper, Magnanti and Wong (1984) presented a unified framework of modeling DNDP and described a number of algorithms including Benders' decomposition and some of its accelerated versions, Lagrange relaxation, and dual ascent procedures as successful algorithms in providing a solution for the special cases of DNDP. As a more recent review, Yang and Bell (1998) presented a review analysis of the models and algorithms based on bi-level formulation for NDP in which MNDP was also mentioned. They classified the models based on the characteristics of the lower-level problem (deterministic or stochastic user equilibrium), the type of the decision variables at the upper-level, the objective function of the upper-level, and the type of the solution algorithm.

As previously mentioned, the bi-level programming program is very difficult to solve. Thus, designing efficient algorithms for NDP in the bi-level formulation is recognized as one of the most challenging problems in transportation (Meng et al. 2001). Therefore, many researchers used some simplifying assumptions and developed algorithms to the simplified problem. The rationale for this is mathematical tractability at the expense of realism of assumptions which, in turn, limits the applicability of the algorithms. Some of the simplifying assumptions are as follows (Poorzahedy and Rouhani 2007): (a) replacing user equilibrium flows by system equilibrium flows in the lower-level and obtaining a convex mixed integer nonlinear programming (Dantzig et al. 1979); (b) using linear functions for user travel time and make the problem less complex by assuming constant link travel time function (Boyce et al. 1973; Holmberg and Hellstrand 1998). This assumption is more suitable for intercity transportation networks owing to the fact that intercity roads are rarely congested. By assuming no congestion, there is no difference between SO and UE flows. Therefore, the objective function of the lower-level becomes like the upper-level one and this, in turn, leads to a single-level problem which is much easier to solve; (c) constraint relaxation, mostly the integrality of decision variables which may lead to an impractical solution in case of DNDP (Abdulaal and LeBlanc 1979; Dantzig et al. 1979; Steenbrink 1974b), and (d) aggregation of the network at different levels by link and node extraction procedure to make network simpler (Haghani and Daskin 1983).

Up to date, studies have mostly focused on the developing heuristics/meta-heuristics algorithm. Within the last two decades, meta-heuristic algorithms for optimization problems generally, and NDP specially, have been increasingly exploited. Meta-heuristics such as simulated annealing (Friesz et al. 1992; Lee and Yang 1994) and ant colony (Poorzahedy and Abulghasemi 2005; Poorzahedy and Rouhani 2007), have been previously used to solve NDP under deterministic conditions. Xiong and Schneider (1995) combined genetic algorithm with neural networks to solve NDP with special constraints such as mutually exclusive projects. Drezner and Salhi (2002) compared the performance of heuristics/meta-heuristics, such as descent algorithm, tabu search, simulated annealing, and genetic algorithm, for one-way and two-way network design problems. They found that genetic algorithm is superior in finding the best solution, while needing longer computation time to achieve such solutions. Karoonsoontawong and Waller (2006) used genetic algorithm, simulated annealing, and random search to solve CNDP when it is modeled as a linear bi-level programming with a lower-level of dynamic user-optimal traffic assignment model. Their study shows that genetic algorithm outperforms other algorithms. Lin et al. (2011) employed the cell transmission model to formulate CNDP as a bi-level linear program. They proposed a heuristic algorithm based on Dantzig-Wolf decomposition to solve the resulted problem. Their algorithm can obtain a local optimal solution for large-scale CNDP. However, the scope of their work was limited to single destination network.

Gao et al. (2005) presented an algorithm by using the support function concept (Floudas 1995) which in turn helps to express the relationship between user flows (the lower-level) and the new additional links in the existing urban network (the upper-level). They compared the performance of the proposed algorithm with LeBlanc's Branch and Bound (B&B) (LeBlanc 1975) on the well-known Sioux-Falls test problem with 5 projects. It should be noted that although LeBlanc's B&B is not efficient in solving real scale problems due to its loose lower bound, it guards against Braess' paradox. However, we show by a counterexample that the algorithm proposed by Gao et al. (2005) cannot guarantee the optimality of the solution. It especially does not guard against Braess' paradox. More specifically, their algorithm offers solutions, which really make system-level objective worse although the algorithm converges to a specious solution with a better objective. Zhang et al. (2009) formulated DNDP as a mathematical program with complementarity constraints where UE was represented in the form of a variational inequality problem. They proposed an active set algorithm to solve the problem. Conducted numerical experiments on three networks showed the effectiveness of their algorithm. However, they have not provided any comparative results with previous algorithms. Chung et al. (2011) formulated a single-level CNDP as a robust optimization problem explicitly incorporating traffic dynamics and demand uncertainty. They used cell transmission model and a box uncertainty set for modeling uncertain demands to have a tractable linear programming based CNDP. They considered a single-destination network.

Wang and Lo (2010) formulated CNDP as a single-level optimization problem. The main deficiency of their work is that the path set between all OD pairs should be known in advance and explicitly imported to the model. This feature prohibits employing their formulation to DNDP because adding new links changes the path set of many OD pairs. In addition, enumerating all paths between all OD pairs in a graph is an intractable problem by itself. They also used piece-wise linear regression to linearize the nonlinear travel time. Their experimental results show that because of the adopted linearization scheme, the presented formulation is not efficient in dealing even with small to medium CNDP problems. Farvaresh and Sepehri (2011) transformed the bi-level DNDP into a single-level MILP using representing UE condition as KKT conditions and employing an efficient linearization scheme to linearizing non-linear terms. They generate two sets of valid inequality which significantly strengthen the formulation. The performance of their formulation having generated valid inequalities when it solved by CPLEX on small networks, even with a relatively large number of new links, is acceptable; but it cannot deal with real problems. Luathep et al. (2011) formulated MNDP as a mathematical programming with equilibrium constraints and proposed a global optimization algorithm. They transformed the problem into a MILP problem using piecewise-linear approximation. They proposed a global optimization algorithm based on a cutting constraint method to solve the resulted MILP problem. Numerical results show the superior efficiency of the proposed method in comparison with the previous algorithms. An interesting result of their work is that solving MNDP on a given budget level most likely makes the system-level objective better than solving DNDP or CNDP separately on the same budget level. As they demonstrated, solving DNDP, CNDP, and MNDP for Sioux Falls network takes around 20, 27, and 35 min. Nevertheless, it seems that their algorithm is not efficient in dealing with large-scale real NDP problems, because it requires one relatively large MILP to be solved in each iteration.

Kim and Kim (2006) proposed a DNDP model in which total travel time was replaced by a comprehensive social cost. Their suggestion for such a social cost function seems rational since the upper-level decision maker (mostly government) should consider all issues of social costs. However, neglecting the Braess' paradox is a practical limitation of their model. They used a genetic algorithm to solve their model and presented some numerical results based on a small network. Like Poorzahedy and Rouhani (2007), a small network was used to evaluate the performance of the suggested algorithm. Both studies compared the their best-found solutions with the global one obtained by full enumeration. Nevertheless, evaluation of the computational efficiency and the solution quality of a meta-heuristic algorithm by a test problem, whose all feasible solutions can be easily enumerated, is not without its pitfalls. Xu et al. (2009) employed genetic algorithm and simulated annealing to solve CNDP. They showed that simulated annealing algorithm outperforms genetic algorithm in solution quality in a shorter time. Sun et al. (2009) introduced an immune clone annealing algorithm, which is designed by combining annealing tactic of the simulated annealing and the immune clone algorithm to solve the MNDP problem. They provide some numerical results on a small network problem. Zhang and Gao (2009) transformed the bi-level MNDP to a single-level MNDP and proposed a locally convergent algorithm using penalty function method. Miandoabchi et al. (2011) formulated a bi-modal multi-objective DNDP considering the concurrent urban road and bus network design. They developed a hybrid genetic algorithm and a hybrid clonal selection algorithm to solve the resulted problem.

Most of the heuristics/meta-heuristics based studies used the well-known Sioux-Falls test problem to evaluate the computational aspects of their algorithms. However, due to the small number of projects (5 or 10 projects) which implies that the solution space of the NDP is small, using such a simple problem could not be a good examination. Because it is more likely a stochastic search method finds the optimal solution in a small solution space. It should be noted that due to budget constraint all subsets of projects do not need to be enumerated. In fact, meta-heuristics with well-tuned parameters probably enumerate all feasible solutions in such problems.

3 Problem formulation

In this section, used notations and definitions are presented, and following that the bi-level formulation of DNDP is introduced.

The following is a summary of the notes and symbols used in this paper.

$G(N, A)$	a connected transportation network, with N and A being the sets of nodes and links, respectively. Here, $N=O \cup D \cup T$ where O, D and T are the set of all origin, destination, and transition nodes respectively. $A = A_1 \cup A_2$ where A_1 is the set of all existing links and A_2 is the set of all proposed new links (projects) which the model decides to be built or not
(i, j)	link designation $(i, j) \in A$
A_i^+	set of links originating from node $i \in N$
A_i^-	set of links pointing to node $i \in N$
(r, s)	origin–destination (OD) pairs where $r \in O$ and $s \in D$ are origin and destination nodes, respectively

W	the set of all OD pairs where $W \subseteq N \times N$
d_r^s	the demand between OD pair $(r, s) \in W$ which is assumed to be nonnegative constant
x_a	the link flow on link $a \in A$
$\boldsymbol{x}$	vector of x_a's with dimention equal to $\|A\|$, $\boldsymbol{x} = (\ldots, x_a, \ldots)T$
x_a^s	flow along link $a \in A$ with destination $s \in D$
y_a	a binary decision variable related to project link $a \in A_2$, taking value 1 or 0 depending on acceptance or rejection of project link $a \in A_2$
$\boldsymbol{y}$	vector of y_a's with dimension equal to $\|A_2\|$, $\boldsymbol{y} = (\ldots, y_a, \ldots)^T$
$t_a(x_a)$	the travel cost on link $a \in A$ defined as a positive and continuous function of link flow x_a which is often representing average travel time on link $a \in A$
g_a	cost of establishing link $a \in A_2$
B	total available budget

Based on the above notations, the DNDP problem under the deterministic user equilibrium can be formulated as the following bi-level programming model:

$$\min_{y} Z^U = \sum_{a \in A} x_a t_a(x_a), \tag{1}$$

$$s.t. \quad \sum_{a \in A_2} y_a g_a \le B, \tag{2}$$

$$y_a \in \{0, 1\}, \quad \forall a \in A_2 \tag{3}$$

$\boldsymbol{x}$ is the solution of user optimal equilibrium (UE problem):

$$\min_{x} Z^L = \sum_{a \in A} \int_0^{x_a} t_a(\theta) d\theta \tag{4}$$

$$s.t. \quad x_a \le M y_a, \quad \forall a \in A_2 \tag{5}$$

$$\sum_{a \in A_i^+} x_a^s - \sum_{a \in A_i^-} x_a^s = q_i^s, \quad \forall i \in N, \forall s \in D \tag{6}$$

$$x_a = \sum_{s \in D} x_a^s, \quad \forall a \in A \tag{7}$$

$$x_a^s \ge 0, \quad \forall a \in A, \forall s \in D \tag{8}$$

$$q_i^s = \begin{cases} -\sum_{r \in O} d_r^s & i = s \\ d_r^s & i = r \quad \forall i \in N, r \in O, s \in D \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Where Z^U and Z^L in (1) and (4) are the upper and lower-level objective functions, respectively. The upper-level decision maker decides which links should be established considering the response of users to resulted changes in the form of user optimum equilibrium. Constraint (2) indicates budget limitation; constraints (4)–(9) assert UE conditions; constraints (5) prohibit flow on non-constructed new links; constraints (6) are flow conservation and assure that the demands of all commodities will be satisfied and the balance of receive and delivery is held in each node; constraints (7) are flow aggregators so that the total flow on each link is the sum of the flows of all commodities on that link; and Constraints (3) and (8) show the positiveness and binary mode of decision variables. In constraint (5), M is an arbitrarily large positive number. Formulation (1)–(9) is based on the link-node flows such that the lower-level problem is formulated as a multicommodity network flow problem in a link-node representation. A commodity is associated with each destination node $s \in D$, and the demand of commodity s in node i is represented by d_i^s .

As can be seen, in the multicommodity formulation the path information is not needed. It should be noted that, based on the definitions and assumptions stated before, the optimality conditions of the link-node formulation is equivalent to the UE condition (Patriksson 1994). The lower-level problem is a convex mathematical program which, because of the strict convexity of the objective function with respect to the link flows, has a unique optimal solution in terms of link flows (Sheffi 1985).

4 Branch and bound (B&B) for DNDP

The general B&B method for DNDP of the form (1)–(9) is based on the key ideas of branching (separating), bounding, and fathoming which are explained in the following (For more information on B&B terminology and technical details, the interested readers are referred to (Floudas and Pardalos 2009; Ibaraki 1987):

4.1 Branching

Let formulation (1)–(9) be denoted as (P) and let its set of feasible solutions be denoted as FS(P). A set of sub-problems (SP_1), (SP_2), …, (SP_n) of (P) is defined as a separation of (P) if (a) A feasible solution of any of the sub-problems (SP_1), (SP_2), …, (SPn) is a feasible solution of (P); and (b) Every feasible solution of (P) is a feasible solution of exactly one of the sub-problems. In fact, conditions (a) and (b) imply that the feasible solutions of sub-problems FS(SP_1), FS(SP_2),…, FS(SP_n) are a partition of the feasible solutions of problem (P). In B&B terminology, the original problem (P) is called a root (parent) node problem while sub-problems are called the child node problems (Floudas 1995). The relationships of the parent and child nodes in DNDP could be represented by a binary tree. In this binary tree, each node corresponds to a sub-problem; and from each non-leaf node two new nodes (sub-problem) are produced by branching on one of the binary decision variables y_a, $a \in A_2$; that is, in the right child node y_a=1, while in the left child node y_a=0. The original problem P could be considered as sub-problem (SP_0) in root node. Sub-problem (SP_n) based on its local position in the tree is a restricted version of the original problem (P):

$$\min_y Z^U = \sum_{a\in A} x_a t_a(x_a), \tag{10a}$$

$$s.t. \qquad \sum_{a\in A_2\backslash \Lambda_n} y_a g_a \le B_n \tag{10b}$$

$$(SP_n): \qquad y_a \in \{0,1\}, \qquad \forall a \in A_2 \backslash \Phi_n \tag{10c}$$

$$y_a \textit{ is fixed}, \qquad \forall a \in \Phi_n \tag{10d}$$

(5), (6), (7), (8), and (9)

where Φ_n is the set of fixed variables y_a from root node to node n along with their values; B_n is the budget available at node n which is obtained from subtracting the summation of the cost of constructed projects from root node to node n. That is,

$$B_n = B - \sum_{a \in \Phi_n} y_a g_a \tag{11}$$

4.2 Bounding

A key issue in every B&B is its bounding scheme. Due to the Braess' paradox bounding in bi-level formulated DNDP has its special difficulties. More specifically, in node n, for example, one cannot set all unfixed variables y_a, $\forall a \in A_2 \backslash \Phi_n$ equal to one and find user equilibrium flows on the resulted network, because there is no guarantee that the system level objective is improved by adding new links to the network. Therefore, to avoid invalid lower-bounds, the bounding scheme should be carefully designed. The following lemma is used to compute a lower bound in node n of B&B search tree:

Lemma For any node n in the B&B tree, the optimal solution of sub-problem (SP_n) is greater than or equal to the optimal value of (SP'_n) , which gives the least possible congestion at the system optimal flows:

$$\min_{y,x} \sum_{a \in A} x_a t_a(x_a), \tag{12a}$$

$$s.t: \quad \sum_{a \in A_2 \backslash \Phi_n} y_a g_a \le B_n \tag{12b}$$

$$(SP'_n): \qquad y_a \in \{0,1\}, \qquad \forall a \in A_2 \backslash \Phi_n \tag{12c}$$

$$y_a \textit{ is fixed}, \qquad \forall a \in \Phi_n \tag{12d}$$

(5), (6), (7), (8), and (9)

Proof See (LeBlanc 1975).

Sub-problem (SP'_n) is a single-level mixed integer non-linear programming (MINLP). In other words, it is a network design problem with system optimal flows. LeBlanc (1975) did not solve sub-problem (SP'_n) and in turn, used a more relaxed problem for bounding in which all unfixed variables are set equal to 1. In fact, fixing all unfixed variables at 1 and computing system optimal flows underestimate the upper-level objective with a high gap. This leads to inefficient lower bounds. More details and computational implications of his work will be presented in the next section. Sub-problem (SP'_n) can be optimally solved by an algorithm which will be outlined in the Section 4.5. In each node, the optimal value of corresponding sub-problem's objective function should be obtained. As will be discussed later, in the presented B&B, solving only one of two resulted sub-problems (left and right child nodes) is needed, because the solution of one of the two resulted child nodes' sub-problems equals to the solution of the parent node's sub-problem. This property significantly decreases the computation efforts in B&B search algorithm. Based on the bound of a node, in order to decide to expand or to stop expanding of the node, some tests are done in the fathoming step.

4.3 Fathoming

The fathoming criteria are checked in each resulted node. In fact, in each node n it is checked if the feasible solution of the sub-problem (SP_n) can contain an optimal solution of the problem (P). If the response to this question is no, then the node is fathomed; and if the response is yes, then the solution should be found and the node is fathomed. If no certain decision is made, then the node must be considered as a candidate to more expansion in the continuation of the searching B&B tree. More specifically, in each node n these criteria are evaluated: (a) Is there any unfixed variable? If not, then the user optimal flows of the resulted network are computed; the upper-level objective is determined, and the node is fathomed; (b) Is there enough budget to construct more projects, or equally to set more $y_a = 1, \forall a \in A_2 \backslash \Phi_n$? If not, then all unfixed variables are fixed at 0, and user optimal flows of the resulted network are computed; the upper-level objective is determined, and the node is fathomed; and (c) Is the lower bound larger than or equal to the current incumbent? If yes, the node is fathomed. The criterion (b) is satisfied if $B_n < min\{g_a \mid a \in A_2 \backslash \Phi_n\}$. In cases (a) and (b) where a new solution with a better upper-level objective is obtained, it is kept as incumbent; and the search tree is scanned and all nodes having lower bounds higher than or equal to the resulted incumbent are fathomed.

The pseudo-code of the proposed B&B algorithm is depicted in Fig. 1.

4.4 Algorithm description

The key steps of the algorithm are shortly explained in the following.

The proposed algorithm starts from the root node where no decision has been made on building or not building new projects. The algorithm iteratively selects a node from the list of active nodes based on the node selection strategy. There are three main strategies to select a node from the active node list L: depth first search, breathe first search, and best first search. There is a vector of decision variables associated with each active node n indicating the status of the all projects from the root node to node n. The elements of this vector, each of which corresponds to one project, take three values 1, 0, and −1 which, in turn, indicate fixed at 1 (build), fixed at 0 (not to build), and unfixed (no decision made yet), respectively. The status of decision variables in each node is shown by a column

```
// Initialization:
    $n \leftarrow 1$, $L \leftarrow \{n\}$, $B(n) \leftarrow$ B, $A(:, n) \leftarrow -1$
    solve $(SP'_n)$ ; $\boldsymbol{y}^{so}(:, n) \leftarrow \hat{\boldsymbol{y}}$ , $LB(n) \leftarrow Z^U$
    compute UE flows $\boldsymbol{x}^*$, $\boldsymbol{y}^* \leftarrow \hat{\boldsymbol{y}}$, $UB \leftarrow F(\boldsymbol{x}^*, \boldsymbol{y}^*)$
While $L \neq \emptyset$ do
      // Node selection and branching:
      parent node $\leftarrow$ select a node from $L$ based on node selection strategy
      $u \leftarrow$ select a variable from unfixed variables of parent node based on variable selection strategy

      $L \leftarrow L$ - {parent node}          //delete selected node from L

      // Right Child Node:
          $n \leftarrow$ size$(L) + 1$
          $A(:, n) \leftarrow A(:, parent\ node)$, $A(u, n) \leftarrow 1$
          $B(n) \leftarrow B(parent\ node) - g_u$        // update budget level

          $L \leftarrow L \cup \{n\}$                    // add new node to list L

          If $A(u, n) \geq 0, \forall\, a \in A_2$ Then
                  $\hat{\boldsymbol{y}} \leftarrow A(:, n)$
                  compute UE flows $\boldsymbol{x}^*$
                  If $F(\boldsymbol{x}^*, \hat{\boldsymbol{y}}) < UB$ Then
                        $\boldsymbol{y}^* \leftarrow \hat{\boldsymbol{y}}$, $UB \leftarrow F(\boldsymbol{x}^*, \boldsymbol{y}^*)$
                        fathom all nodes of $L$ with $LB \geq UB$
                        fathom node n
                  End if
          Elseif $B(n) \leq \min\{g_a \mid a \in A_2 \wedge A(u, n) = -1\}$ Then
                  $A(u, n) \leftarrow 0,\ \forall\, a \in \{a \mid a \in A_2 \wedge A(u, n) = -1\}$
                  $\hat{\boldsymbol{y}} \leftarrow A(:, n)$
                  compute UE flows $\boldsymbol{x}^*$
                  If $F(\boldsymbol{x}^*, \hat{\boldsymbol{y}}) < UB$ Then
                        $\boldsymbol{y}^* \leftarrow \hat{\boldsymbol{y}}$, $UB \leftarrow F(\boldsymbol{x}^*, \boldsymbol{y}^*)$
                        fathom all nodes from $L$ with $LB \geq UB$
                        fathom node n
                  End if
          Else
               If $\boldsymbol{y}^{so}(u, parent\ node) = 1$ Then
                    $LB(n) \leftarrow LB(parent\ node)$
               Else
                    solve $(SP'_n)$ , $\boldsymbol{y}^{so}(:, n) \leftarrow \hat{\boldsymbol{y}}$, $LB(n) \leftarrow Z^U$
                    if $LB(n) \geq UB$ Then fathom node n
               End if
          End if
      // Left Child Node:
          $n \leftarrow$ size$(L) + 1$
          $A(:, n) \leftarrow A(:, parent\ node)$, $A(u, n) \leftarrow 0$
          $B(n) \leftarrow B(parent\ node)$             // update budget level

          $L \leftarrow L \cup \{n\}$                    // add new node to list L

          If $A(u, n) \geq 0, \forall\, a \in A_2$ Then
                  $\hat{\boldsymbol{y}} \leftarrow A(:, n)$
                  compute UE flows $\boldsymbol{x}^*$
                  If $F(x^*, \hat{y}) < UB$ Then
                        $\boldsymbol{y}^* \leftarrow \hat{\boldsymbol{y}}$, $UB \leftarrow F(\boldsymbol{x}^*, \boldsymbol{y}^*)$
                        fathom all nodes from $L$ with $LB \geq UB$
                        fathom node n
                  End if
          Else
                 If $\boldsymbol{y}^{so}(u, parent\ node) = 0$ Then
                      $LB(n) \leftarrow LB(parent\ node)$
                 Else solve $(SP'_n)$ , $\boldsymbol{y}^{so}(:, n) \leftarrow \hat{\boldsymbol{y}}$, $LB(n) \leftarrow Z^U$
                      If $LB(n) \geq UB$ Then fathom node n
          End if
End while
```

Fig. 1 Pseudo-code of proposed branch and bound for DNDP

vector of size $|A_2|$ in column n of matrix $\boldsymbol{\Lambda}$. Matrix $\boldsymbol{\Lambda}$ is a matrix of dimension $|A_2|\times N_{max}$ where N_{max} is the maximum number of active nodes in the B&B search tree. It could be approximately set to a fixed number and in cases where more active nodes appear in the tree, the matrix is rescaled. In our conducted experiments, it was set to 100, and it was not violated by the algorithm. It does not seem to be so memory restrictive in today's computers.

First, the sub-problem (SP_0') ,(SP_n') at root node ($n=0$), is solved, and this leads to the lowest lower-bound for the original problem P because all sub-problems (SP_n') , $n\neq 0$, are the restricted versions of sub-problem (SP_0') . The optimal solution of sub-problem (SP_0') which is indicated by $\hat{\boldsymbol{y}}$ in pseudo-code is stored in the column n of matrix $\boldsymbol{y}^{so}$. Matrix $\boldsymbol{y}^{so}$, like matrix $\boldsymbol{\Lambda}$, is of dimension $|A_2|\times N_{max}$. It is obvious that the optimal solution of sub-problem (SP_0') is also a feasible solution for the original problem P. Therefore, if the user optimal flows of the resulted network are computed and put into the objective function of problem P, a solution is obtained in the root node. This solution (incumbent in the root node) is an upper-bound (UB) for problem P. In the pseudo-code, the current incumbent, its associated solution, and user optimal flows are stored in UB and $\boldsymbol{y}^*$, $\boldsymbol{x}^*$, respectively. In most cases, the gap between the resulted incumbent and the optimal solution of problem P is small and, hence, the incumbent in root node is very useful in the fathoming step.

In each iteration, one node from active nodes is selected; and in the selected node, one unfixed variable is fixed, or simply the decision on building a project is separated. This leads to two new active nodes, namely, right and left child nodes. Once a node is created, it is checked to see if all variables are fixed (the created node is a leaf node) or not. In the case of arriving at a leaf node, user optimal flows are computed, and the corresponding upper-level objective is compared with the current UB. If the upper-level objective is lower than UB, then UB and $\boldsymbol{y}^*$ are updated. If the node is not a leaf node, then the bounding step is performed. Since the lower bound of one of the child nodes equals to that of the parent node, only one lower bound is computed in each iteration. This is because the value of the branching variable in the solution of the sub-problem corresponding to the parent node is either 0 or 1. In the former case, it does not need to compute the lower bound for the left child node, and in the latter case this holds for the right child node. This property improves computation costs significantly. Once a right child node is added to the search tree, the level of available budget at that node is checked. If the available budget does not suffice to build any other project, all unfixed variables are fixed at 0, and user optimal flows are computed, and the corresponding upper-level objective is compared with the current UB. If the upper-level objective is lower than UB, UB and $\boldsymbol{y}^*$ are updated, and the node is fathomed. Since the available budget of the left child node is the same as that of the parent node, the available budget is not checked when a left child node is added to the search tree.

4.5 Computing lower bounds

The method of computing the lower bounds in the search tree is the main concern of any B&B algorithm. In each iteration, two new nodes are added to the search tree, but only one lower bound needs to be computed through solving problem (SP′). For the ease of reference, let rewrite the formulation of (SP'_n) in node n as follows:

$$\min_{y,x} f(x,y), \tag{13a}$$

$$s.t. \quad g(x,y) \leq 0, \tag{13b}$$

$$(SP'_n): \qquad x \in \mathrm{X}, \tag{13c}$$

$$y \in \mathrm{Y}. \tag{13d}$$

where,

$$f(x,y) = \sum_{a \in A} x_a t_a(x_a), \tag{14}$$

$$g(x,y) = x_a - M y_a, \forall a \in A_2, \tag{15}$$

$$\mathrm{X} = \{x_a = \sum_{s \in D} x_a^s, \forall a \in A; \sum_{a \in A_i^+} x_a^s - \sum_{a \in A_i^-} x_a^s = d_i^s, \forall i \in N, \forall s \in D; x_a^s \geq 0, \forall a \in A, \forall s \in D\} \subseteq \mathrm{R}^{|\mathrm{A}|}, \tag{16}$$

$$\mathrm{Y} = \{\sum_{a \in A_2 \backslash \Lambda_n} y_a g_a \leq B_n; y_a \in \{0,1\}, \forall a \in A_2 \backslash \Phi_n; y_a \text{ is fixed}, \forall a \in \Phi_n\} = \{1,0\}^{|A_2|}, \tag{17}$$

Problem (SP'_n) is a MINLP formulation such that: (a) objective function is convex and differentiable, (b) constraints are linear, X is a compact convex set, and Y is a finite set of binary elements and (c) for any fixed y, the resulted NLP, which is a system optimal equilibrium problem, has an optimal solution. Therefore, this problem can be optimally solved by algorithms like generalized Benders' decomposition (GBD) (Floudas 1995; Geoffrion 1972), and outer approximation (OA) (Duran and Grossmann 1986; Fletcher and Leyffer 1994). The key idea underlying these algorithms is that in each iteration they generate an upper-bound and a lower-bound on the MINLP optimal solution. They alternate between solving a nonlinear programming (sub-problem) and solving a mixed integer linear programming (master-problem). The sub-problems of both methods for (SP'_n) are similar, and the major difference between them lies in the different derivations of the master problem. Both algorithms were experimentally tested; however, since OA had superior performance, only the OA related formulation is presented. Problem (SP'_n) can be written as:

$$\min_{y} \inf_{x} f(x,y), \tag{18a}$$

$$s.t. \quad g(x,y) \leq 0, \tag{18b}$$

$$x \in \mathrm{X}, \tag{18c}$$

$$y \in \mathrm{Y}. \tag{18d}$$

Since the inner problem is bounded (it is a parametric in y system optimal equilibrium), infimum could be replaced by minimum. Let

$$V = \{y \in \mathrm{Y} | \text{there exists}\, x \in \mathrm{X}\, \text{such that}\, \mathrm{g}(x,y) \leq 0\}. \tag{19}$$

For any $y \in V$, Let $\upsilon(y)$ be the optimal value of the following nonlinear sub-problem:

$$\min_x f(x,y), \tag{20a}$$

$$NLP(y): \qquad s.t.\mathrm{g}(x,y) \leq 0, \tag{20b}$$

$$x \in X. \tag{20c}$$

$NLP(y)$ is the system optimal equilibrium flow problem for the network corresponding to the $y \in V$. In fact, $NLP(y)$ is a restricted version of problem $(SP_n^{'})$ because only a specific feasible y out of all $y \in V$ has been chosen. Therefore, the optimal solution of the $NLP(y)$ is a feasible solution for problem $(SP_n^{'})$ and more importantly, $\upsilon(y)$ is an upper bound to the optimal value of problem $(SP_n^{'})$.

Then problem $(SP_n^{'})$ is equivalent to the following problem:

$$\min_y \upsilon(y), \tag{21a}$$

$$\text{s.t.} y \in \mathrm{Y} \cap V \tag{21b}$$

Note that $\mathrm{Y} \cap V$ represents the projection of the feasible space of problem $(SP_n^{'})$ onto the y-space. Problem (21a, 21b) is difficult to solve because both V and $\upsilon(y)$ are known implicitly (Duan and Xiaoling 2006; Floudas 1995). To cope with this difficulty, Duran and Grossmann (1986) used the outer linearization of $\upsilon(y)$ and a particular representation of V. Then based on Duran and Grossmann's (1986) arguments for the general formulation of MINLP, it is concluded that problem $(SP_n^{'})$ is equal to the following problem:

$$\min_{(x,y)\in\Omega} f(x,y) = \min_{y^k\in V} \min_{x\in X} \{f(x,y) | g(x,y^k) \leq 0\} \tag{22}$$

$$= \min_{y^k\in V} \min f(x^k,y^k) + \nabla^T f(x^k,y^k) \begin{pmatrix} x - x^k \\ 0 \end{pmatrix}, \tag{23a}$$

$$\text{s.t.} \quad g(x^k,y^k) + \nabla^T g(x^k,y^k) \begin{pmatrix} x - x^k \\ 0 \end{pmatrix} \leq 0, \tag{23b}$$

$$\mathrm{x} \in \mathrm{X}. \tag{23c}$$

$$= \min_{y^k \in V} \min \eta \tag{24a}$$

$$\text{s.t.} \quad \eta \geq f(x^k, y^k) + \nabla^T f(x^k, y^k) \begin{pmatrix} x - x^k \\ 0 \end{pmatrix}, \tag{24b}$$

$$0 \geq g(x^k, y^k) + \nabla^T g(x^k, y^k) \begin{pmatrix} x - x^k \\ 0 \end{pmatrix}, \tag{24c}$$

$$\mathrm{x} \in \mathrm{X}, \tag{24d}$$

$$\eta \in R^1. \tag{24e}$$

In view of the fact that $g(x, y)$ is linear, constraints (24c) is simplified to $0 \geq g(x, y^k)$. Now, let

$$K^* = \{k | y^k \in V \text{ and } x^k \text{ are system optimal equilibrium flows corresponding to } NLP(y^k)\}. \tag{25}$$

Then the following master problem is equivalent to problem $(SP_n^{'})$ (Duran and Grossmann 1986).

$$\min \eta \tag{26a}$$

$$\text{s.t.} \ \eta \geq f(x^k, y^k) + \nabla^T f(x^k, y^k) \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix}, \ k \in K, \tag{26b}$$

$$0 \geq g(x^k, y), \ k \in K^*, \tag{26c}$$

$$M - OA : y \in V, \tag{26d}$$

(24d) and (24e).

Note that since the function $f(x, y)$ is convex and $g(x, y)$ is linear, the linearizations in *M-OA* correspond to outer approximations of the nonlinear feasible space of problem $(SP_n^{'})$. It is assumed that the current transportation network is a connected network and, hence, for any $y^k \in V$ there exist system optimal equilibrium flows x^k, or equally, $NLP(y^k)$ sub-problem is always feasible. Therefore, constraint $y^k \in V$ can be replaced by $y^k \in \mathrm{Y}$. Nevertheless, if $NLP(y^k)$ sub-problem is infeasible, the algorithm can solve the problem by adding one infeasible cut to the master problem. In addition, since solving the master problem *M-OA* requires enumerating all feasible binary variables y^k, another relaxed version of *M-OA* is considered in which the solution of previous *NLP*(y^k) sub-problems, k=1,2, …,K, is employed and thus:

$$\min \eta \tag{27a}$$

$$\text{s.t. } \eta \geq f(x^k, y^k) + \nabla^T f(x^k, y^k) \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix}, \; k \in K, \tag{27b}$$

$$RM - OA: \qquad 0 \geq g(x^k, y), \; k \in K, \tag{27c}$$

(26d), (24d) and (24e).

Since problem *RM-OA* is a relaxed version of *M-OA*, the solution of problem *RM-OA* is a lower-bound for problem $(SP_n^{'})$. Furthermore, since function linearizations are accumulated as iterations k proceed, the sequence of the lower-bounds resulting from solving the master problem *RM-OA* is non-decreasing. Adding the following integer cut can improve convergence time because it prevents master problem from choosing the previous 0–1 values examined at the previous K iteration (Duran and Grossmann 1986).

$$\sum_{a \in B^k} y_a - \sum_{a \in N^k} y_a \leq \left| B^k \right| - 1, \; k = 1, \ldots, K. \tag{28}$$

where $B^k = \left\{ a | y_a^k = 1 \right\}$ and $N^k = \left\{ a | y_a^k = 0 \right\}$ }.

As the iterations proceed, two sequences of updated upper-bounds and lower-bounds are obtained, which are shown to be non-increasing and non-decreasing, respectively. Duran and Grossmann (1986), have shown that these two sequences converge in a finite number of iterations. The pseudo-code of OA based algorithm to solve problem $(SP_n^{'})$ is presented in Fig. 2.

When the network is not small, solving *NLP*(y^k) sub-problems and computing system optimal equilibrium flows x^k at iteration k using off-the-shelf NLP optimization softwares is very difficult. This is not prohibitive because there are efficient algorithms to compute system optimal flows for large networks (e.g. Frank-Wolf (LeBlanc et al. 1975) and Bush-based algorithms like DBA (Nie 2010)). In all conducted experiments, Frank-Wolf was used to solve *NLP* sub-problems.

Remark 1 In each node of the B&B search tree, variables have been partitioned in two disjoint sub-sets, namely fixed variables and unfixed variables. While the search proceeds to deeper nodes, the number of unfixed variables decreases, and number of fixed variables increases. This leads to a simpler master problem of $(SP_n^{'})$. Therefore, solving problem $(SP_n^{'})$ by OA is more likely to be easier in deeper nodes of B&B search tree -in order to obtain the lower-bound- than it is in shallow nodes of the tree. In other words, while the search visits deeper nodes, the computation time of bounding potentially decreases or at least does not increase.

```
// Initialization:
A feasible $y^0$ is given.
Select a small positive number $\varepsilon$ as convergence tolerance.
  $k \leftarrow 0$, $UBD = ¥$,  $LBD = -¥$.
While $(UBD - LBD) \geq \varepsilon$ do
  Solve $NLP(y^k)$ and compute system optimal flows $x^k$
  $\upsilon(y^k) \leftarrow f(x^k, y^k)$
    if $\upsilon(y^k) < UBD$ Then
    $UBD \leftarrow \upsilon(y^k)$
    $y^* \leftarrow y^k, x^* \leftarrow x^k$
    End if
  Solve RM-OA, store the solution in $y^{k+1}$, $LBD \leftarrow \eta$
  $k \leftarrow k+1$
End while
```

Fig. 2 Pseudo-code of OA based algorithm to solve problem (SP'_n) in bounding step

4.6 Variable selection for branching

Variable selection for branching on the selected node can have a significant influence on the performance of the B&B algorithm. Because robust techniques for identifying branching variables have not been established yet, a common way of choosing a branching variable is using heuristics. In the proposed algorithm, the bounding step provides useful information to guide the choice of a high-impact branching variable. A heuristic impact index is assigned to unfixed variables based on the solution of problem (SP'_n) in the node n. This index reflects branching priority of unfixed variables. First, unfixed variables are sorted based on the solution of problem (SP'_n) such that variables having value 1 are put at the higher level comparing to the variables having value 0. Then, variables are sorted decreasingly based on the total travel cost of their associated links, namely, $x_a t(x_a)$. Therefore, the branching variable is one which has value 1 in the solution of (SP'_n) and has the highest associated total travel cost in the resulted network.

5 Numerical results

In this section, the validity of the proposed algorithm is evaluated, and the results are compared with those of the previous works. To this end, four test problems are presented. The first test problem is a small network with a low number of new links. The first test problem is a case in which Braess' paradox may occur; the second one is the well-known Sioux-Falls network with 5 new links; the third problem is a small network with a relatively large number of proposed new links; and the fourth one is a relatively large network with a moderate number of new links. The details of the tested problems are reported in Table 1.

All experiments were carried out on a HP Pavilion dv3 with 4 GB RAM and a 2.26 GHz Core2Duo CPU. The main body of B&B was coded in MATLAB. CPLEX 12.2 was used to solve MIP master problems in the lower bounding step. The parallel optimization mode of CPLEX has been set to deterministic mode. In all conducted

Table 1 Detail of test problems

Test problem	$\|N\|$	$\|A\|$	$\|A_1\|$	$\|A_2\|$	one/two-way new projects	OD pairs
1	8	14	11	3	1	3
2	24	76	66	10	2	528
3	16	42	17	25	1	2
4	100	317	287	30	2	817

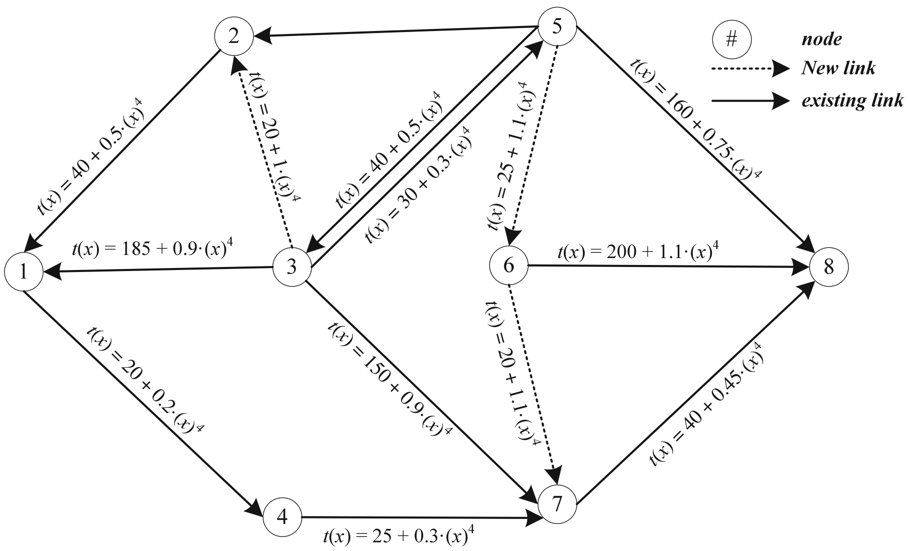
experiments, Frank-Wolf algorithm coded in C++ language with convergence rate 0.01 or maximum 20 iterations was used to solve *NLP* sub-problems. Hereafter, we abbreviate our proposed algorithm as FS-B&B. In all experiments, a column vector of size $|A_2|$ with zero elements is given as initial solution to G-GBD.

5.1 Comparative results

To validate the efficiency of the proposed algorithm, the performance of the proposed algorithm is compared with those of the previous ones proposed for DNDP including LeBlanc's B&B (LeBlanc 1975) (hereafter abbreviated as L-B&B), Gao et al.'s GBD based algorithm (Gao et al. 2005) (hereafter abbreviated as G-GBD), and Poorzahedy and Abulghasemi (2005) and Poorzahedy and Rouhani's (2007) ant system based meta-heuristics (hereafter abbreviated as PAR-AS). LeBlanc's B&B is an exact algorithm guarding against Braess' paradox. G-GBD algorithm is an exact algorithm (as claimed in the published paper) which uses GBD main concepts as well as the relationship between user and system optimal flows. However, we experimentally show that the algorithm proposed by Gao et al. (2005) cannot guarantee the optimality of the solution. It especially does not guard against the Braess' paradox. More specifically, their algorithm offers solutions, which really make the system-level objective worse, although the algorithm converges to a specious solution with a better objective. PAR-AS is an ant system based meta-heuristics which was introduced in (Poorzahedy and Abulghasemi 2005) and was later improved in (Poorzahedy and Rouhani 2007) through being hybridized with genetic algorithm, simulated annealing, and tabu search. We only report the best results obtained from the base ant system in their earlier work (Poorzahedy and Abulghasemi 2005) and its improvement 1 in their later work (Poorzahedy and Rouhani 2007).

Test problem 1 The network of the first test problem is shown in Fig. 3. This problem is a modified network of the one which has been reported in (LeBlanc 1975). The travel time function of each link is depicted close to the link. There are three new links (3, 2), (5, 6), and (6, 7) with construction cost 8, 5, and 4 units of money, respectively. The decision variables corresponding to these new links are denoted by vector $(y_1, y_2, y_3)^T$. There are three OD pairs (1, 7), (5, 1), (3, 8) with demand 2, 5, and 5, respectively. The problem is solved in two budget scenarios 9 and 17 units of money.

The detailed results of all four mentioned algorithms on test problem 1 are presented in Table 2. Column 1 and 2 show the ratio of available budget to the total



node
New link
existing link
$t(x) = 40 + 0.5\cdot(x)^4$
$t(x) = 20 + 1\cdot(x)^4$
$t(x) = 40 + 0.5\cdot(x)^4$
$t(x) = 30 + 0.3\cdot(x)^4$
$t(x) = 25 + 1.1\cdot(x)^4$
$t(x) = 160 + 0.75\cdot(x)^4$
$t(x) = 185 + 0.9\cdot(x)^4$
$t(x) = 200 + 1.1\cdot(x)^4$
$t(x) = 20 + 0.2\cdot(x)^4$
$t(x) = 150 + 0.9\cdot(x)^4$
$t(x) = 20 + 1.1\cdot(x)^4$
$t(x) = 40 + 0.45\cdot(x)^4$
$t(x) = 25 + 0.3\cdot(x)^4$

Fig. 3 Test network 1. A hypothetical network for Braess' paradox

cost needed to build all projects (B/C), and the absolute value of available budget, respectively. Column 3 contains the optimal value of the system-level objective of each scenario. Column 4 includes the names of the algorithms. Column 5 and 6 show the objective value corresponding to each algorithm and the final solution they provide, respectively. As shown in Table 1, all algorithms except G-GBD converge to the optimal solution in both budget scenarios. The solution $(y_1, y_2, y_3)^T$=(0, 0, 0)^T is given to G-GBD as the initial solution. G-GBD converges to solutions with the system level objective 2554.964 and 2514.028 in budget levels 9 and 17, respectively. In budget level 9 the new link (3, 2) is suggested to be established, and in budget level

Table 2 Results for test problem 1: Braess' paradox

B/C	Budget	optimal objective value	Algorithm	Objective Value	Solution $(y_1, y_2, y_3)^T$
0.53	9	2686.516	L-B&B	2686.516	$(0, 0, 0)^T$
			PAR-AS	2686.516	$(0, 0, 0)^T$
			G-GBD	2895.925	$(1, 0, 0)^T$
			FS-B&B	2686.516	$(0, 0, 0)^T$
1	17	2686.516	L-B&B	2686.516	$(0, 0, 0)^T$
			PAR-AS	2686.516	$(0, 0, 0)^T$
			G-GBD	2946.683	$(1, 1, 1)^T$
			FS-B&B	2686.516	$(0, 0, 0)^T$

"L-B&B" = LeBlanc B&B, "G-GBD" = Gao et al. Generalized Benders Decomposition,

"FS-B&B" = The proposed B&B "PAR-AS" = Poorzahedy, Abulghasemi and Rouhani's Ant System

17 all the three new links (3, 2), (5, 6) and (6, 7) are suggested to be established as well. However, if the user equilibrium flows and the associated system-level objective of the resulted network are computed, it is found that the true value of the system-level objective equals 2895.925 and 2946.683 for budget levels 9 and 17, respectively. In fact, the algorithm converges to a point different form the true objective value. But the main fault of G-GBD is that it suggests the worst solution $(y_1, y_2, y_3)^T=(1, 1, 1)^T$, because the optimal solution is $(y_1, y_2, y_3)^T=(0, 0, 0)^T$ with system-level objective value 2686.516. As a matter of fact, the algorithm falls in the trap of Braess' paradox. This shows that G-GBD may converge to a point whose value is not the true system-level objective. By inspecting the generated bounds in each iteration, one can find that the provided upper bounds are not valid and this, in turn, makes the algorithm converge to a non-optimal solution. Therefore, the G-GBD algorithm should be considered as a local optimal algorithm unless it is explicitly assumed that conditions like the Braess' paradox do not happen. There is no guarantee for PAR-AS to guard against the Braess' paradox, although in this specific case it found the optimal solution.

Test problem 2 Sioux-Falls network is a well-known test problem for both DNDP and CNDP. We use almost the same data as reported in (LeBlanc 1975).[1] Here, we assume that all the proposed projects are new links, and that their associated head and tail nodes are not directly connected. This test problem is somewhat different from that of (LeBlanc 1975) where the proposed projects are links which should be upgraded to a specific level. Briefly, in this test problem $|N|=24$, $|A|=76$, $|A_1|=66$, $|A_2|=10$, and the available budget $B=3000,000$. It is assumed that the proposed projects are two-way streets and, hence, the total binary variables decreased to 5 variables each of which corresponds to a new two-way street. Due to the network's low number of new projects, all the above-mentioned algorithms obtain the optimal solution. Set A_2 and cost of projects are assumed to be as follows:

$$A_2 = \{\mathrm{P}_1 = \{(6,8),(8,6)\}, \mathrm{P}_2 = \{(7,8),(8,7)\}, \mathrm{P}_3 = \{(9,10),(10,9)\},$$

$$\mathrm{P}_4 = \{(10,16),(16,10)\}, \mathrm{P}_5 = \{(13,24),(24,13)\}\},$$

$$g_1 = 650000; g_2 = 1000000; g_3 = 625000; g_4 = 1200000; g_5 = 850000.$$

The detailed results of employing all algorithms are presented in Table 3. Column 1 and 2 are the same as those of Table 2. Column 3–5 show the algorithm abbreviation, objective value, and the solution provided by each algorithm, respectively. Column 6 shows either the total iteration to converge (for G-GBD) or the total number of lower bound computations (for FS-B&B and L-B&B). These two indices are the main indicators of computational costs of the mentioned algorithms. No index is reported for PAR-AS because, firstly, there is no guarantee to obtain an optimal solution, and secondly, the number of runs or ants is usually chosen by the user. Column 7 shows the total CPU time of algorithms to terminate. As can be seen from Table 3, all algorithms successfully obtain the optimal solution in a reasonable time. In fact, there is no significant difference between the computational costs of G-GBD, PAR-AS, and FS-B&B in any of the budget levels. Algorithm L-B&B needs more

[1] Sioux-Falls network data are those that available on-line at: http://www.bgu.ac.il/bargera/tntp/.

Table 3 Results for test problem 2: Sioux-Falls network

B/C	Budget	Algorithm	Objective Value	Solution	Iteration/ bounding	CPU time (s)
0.46	2×10^6	L-B&B	1.5839×10^7	3, 5	12	11
		PAR-AS	1.5839×10^7, (8/10)[a]	3, 5	–	8[b]
		G-GBD	1.5839×10^7	3, 5	9	7
		FS-B&B	1.5839×10^7	3, 5	2	7
0.63	3×10^6	L-B&B	1.1320×10^7	3, 4, 5	13	13
		PAR-AS	1.1320×10^7, (10/10)[a]	3, 4, 5	–	7[b]
		G-GBD	1.1320×10^7	3, 4, 5	10	7
		FS-B&B	1.1320×10^7	3, 4, 5	3	6
0.92	4×10^6	L-B&B	9.4208×10^6	1, 3, 4, 5	5	4
		PAR-AS	9.4208×10^6, (10/10)[a]	1, 3, 4, 5	–	7[b]
		G-GBD	9.4208×10^6	1, 3, 4, 5	11	9
		FS-B&B	9.4208×10^6	1, 3, 4, 5	6	9

"L-B&B" = LeBlanc B&B, "G-GBD" = Gao et al. Generalized Benders Decomposition,

"FS-B&B" = The proposed B&B "PAR-AS" = Poorzahedy, Abulghasemi and Rouhani's Ant System

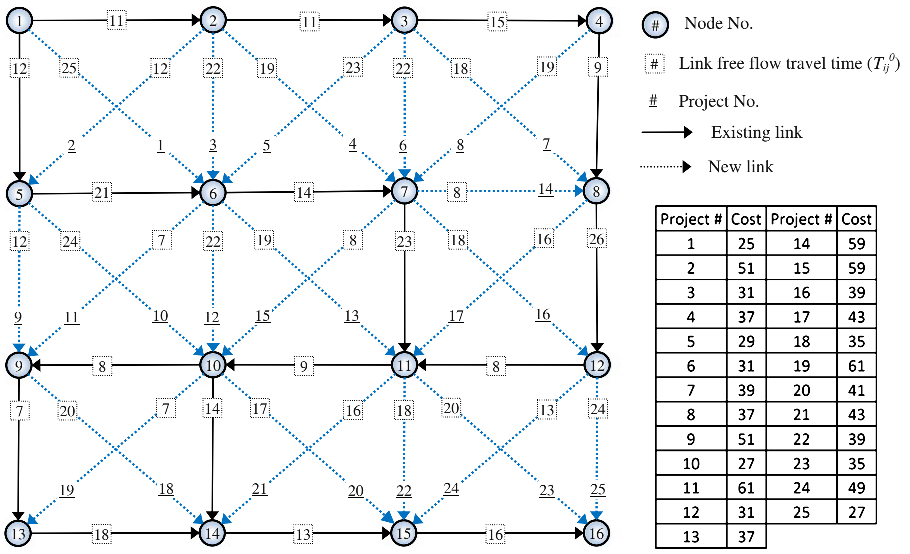
[a] Frequency of finding optimal solution in 10 runs

[b] Average time for each run

time in low budget levels and less time in high budget levels to find the optimal solution or optimal confirmation of the found solution. As a result, reviewing previous studies reveals that the performance of the developed algorithms has been evaluated by small to medium-scale networks having a low number of projects being the candidate to be added to the network. Since the computational efficiency of these algorithms is mostly affected by the number of candidate projects, such test problems could not be a good examination to evaluate the performance of algorithms dealing with DNDP. Therefore, we devise a simple test problem with a relatively high number of new projects and a large network with a medium number of new projects to evaluate and compare the computational efficiency of the proposed and the previous algorithms. These networks are presented in test problems 3 and 4.

Test problem 3 This problem has 16 nodes, 17 existing links, and 25 new links (see Fig. 4.). The link travel time function is assumed as $t(x_a) = T_a^0 + B_a x_a^4$, where T_a^0 and B_a are free flow travel time and constant of link *a*, respectively. The values of T_a^0 are depicted on links, and the values of B_a are assumed to be $10^{-6}\times T_{ij}^0$. There are two OD pairs (1, 16) and (4, 13) each of which with demand 100.

In this test problem, there are totally more than 33×10^6 solutions in the solution space when budget constraint is relaxed; This number of points is much larger than 32 solutions in case of 5 new projects and 1024 solutions in case of having 10 new projects. The performance of previous algorithms and our proposed algorithm on this test problem are compared. Table 4 shows the results of employing G-GBD, PAR-AS, L-B&B, and FS-B&B on test problem 3. The results reported for PAR-AS algorithm



Project #	Cost	Project #	Cost
1	25	14	59
2	51	15	59
3	31	16	39
4	37	17	43
5	29	18	35
6	31	19	61
7	39	20	41
8	37	21	43
9	51	22	39
10	27	23	35
11	61	24	49
12	31	25	27
13	37		

Fig. 4 Test network 3

are the best obtained results from the base ant system and its improvement 1. Column 1 and 2 are the same as those of Table 2. Column 3 shows algorithms by their abbreviations; and column 4 and 5 show their corresponding objective values as well as the relative deviation of the resulted objective from optimal objective values (relative deviation from optimal) in each scenario, correspondingly. The frequency of finding the optimal solution by PAR-AS is shown next to its associate objective value in column 4. Column 6 indicates the solution of each method in each scenario in terms of selected projects to be built. Column 7 reflects the consumed budget which is a portion of the available budget. This column was deliberately inserted to show how each algorithm effectively consumes the available budget. Column 8 shows the total CPU time for each method to solve related DNDP in each scenario. A limit of 10 h of CPU time is imposed for L-B&B algorithm in each budget level. The total number of nodes that L-B&B and FS-B&B need to compute lower bound, and the total number of iterations that G-GBD needs to converge are the main factors of their computational efficiency. Therefore, the number of computed lower bounds (L-B&B and FS-B&B) or iterations (G-GBD) is shown in column 9. The best value for the metric of each column in each budget scenario on the condition that the optimal solution is obtained in the shortest time, is printed in bold.

Examining the PAR-AS by the provided simple test problem reveals that unlike the reported results on small test problems of previous studies (Poorzahedy and Abulghasemi 2005; Poorzahedy and Rouhani 2007), the algorithm is not able to find the optimal solution in most budget levels. The algorithm was carried out over 50 runs in each budget level. It only finds the optimal solution in budget levels 100 in 4 runs out of 50 runs. Furthermore, in cases where the budget level is moderately tight, and by implication, where there are enough combinations of projects satisfying budget constraint, the gap between the solutions of PAR-AS and the optimal solution is significant. G-GBD finds the optimal solution in budget levels 100, 200, 300, and

Table 4 The results of the proposed algorithm on test problem 3 in comparison with the previous study

B/C	Budget	Algorithm	Objective Value	RDO (%)	Solution	Consumed budget	Iteration/ bounding	CPU time (s)
0.10	100	L-B&B	4.3677×10^{7}	0	8,10,25	91	1413	1186
		PAR-AS	4.3677×10^{7}, (4/50)[a]	0	8,10,25	91	–	189[b]
		G-GBD	4.3677×10^{7}	0	8,10,25	91	31	14
		FS-B&B	4.3677×10^{7}	0	8,10,25	91	3	3
0.20	200	L-B&B	2.0619×10^{7}	0	8,10,15,19	184	8015	7054
		PAR-AS	2.1744×10^{7}, (0/50)[a]	5.46	4,8,10,15,23	195	–	365[b]
		G-GBD	2.0619×10^{7}	0	8,15,19,23	192	148	111
		FS-B&B	2.0619×10^{7}	0	8,15,19,23	192	4	6
0.29	300	L-B&B	6.0258×10^{6}	0	1,8,10,15,17,19,23	287	21346	20473
		PAR-AS	1.0370×10^{7}, (0/50)[a]	72.09	3,8,13,15,16,19,23	299	–	494[b]
		G-GBD	6.0258×10^{6}	0	1,8,10,15,17,19,23	287	237	259
		FS-B&B	6.0258×10^{6}	0	1,8,10,15,17,19,23	287	7	22
0.39	400	L-B&B	3.0109×10^{6}	0	1,4,8,10,15,16,17,19,23,25	390	11603	11344
		PAR-AS	4.0109×10^{6}, (0/50)[a]	33.21	1,4,8,10,15,16,19,22,23,25	386	–	648[b]
		G-GBD	3.0109×10^{6}	0	1,4,8,10,15,16,17,19,23,25	390	453	930
		FS-B&B	3.0109×10^{6}	0	1,4,8,10,15,16,17,19,23,25	390	10	41
0.49	500	L-B&B	2.6103×10^{6}	0	1,4,6,8,10,13,15,16,17,19,22,23,25	497	4641	4163
		PAR-AS	3.6465×10^{6}, (0/50)[a]	39.69	1,5,8,9,10,12,15,16,17,18,19,23,25	499	–	726[b]
		G-GBD	2.9360×10^{6}	12.48	1,4,6,8,13,15,16,17,19,21,23,25	474	437	720
		FS-B&B	2.6103×10^{6}	0	1,4,6,8,10,13,15,16,17,19,22,23,25	497	13	42
0.59	600	L-B&B	2.5638×10^{6}	0	1,3,4,6,7,8,10,12,13,15,16,17,19,20,23,25	600	1176	1110
		PAR-AS	2.7581×10^{6}, (0/50)[a]	7.58	1,3,4,6,8,9,10,12,15,16,17,18,19,23,25	569	–	792[b]
		G-GBD	2.6644×10^{6}	3.93	1,3,4,6,7,8,10,12,13,15,16,17,18,19,23,25	594	275	218
		FS-B&B	2.5638×10^{6}	0	1,3,4,6,7,8,10,12,13,15,16,17,19,20,23,25	600	16	13

Table 4 (continued)

B/C	Budget	Algorithm	Objective Value	RDO (%)	Solution	Consumed budget	Iteration/ bounding	CPU time (s)
0.69	700	L-B&B	2.5524×10^6	0	1,3,4,6,7,8,9,10,12,13,15,16,17,18,19,20,23,25	686	62	53
		PAR-AS	2.5688×10^6, (0/50)[a]	0.64	1,4,6,7,8,9,10,12,13,14,15,16,17,19,20,23,25	679	–	787[b]
		G-GBD	2.5956×10^6	1.69	1,3,4,6,8,9,10,13,14,15,16,17,19,20,21,23,25	683	94	36
		FS-B&B	2.5524×10^6	0	1,3,4,6,7,8,9,10,12,13,15,16,17,18,19,20,23,25	686	18	8

"L-B&B" = LeBlanc B&B, "G-GBD" = Gao et al. Generalized Benders Decomposition, "B/C" = Budget/Cost

"FS-B&B" = The proposed B&B "PAR-AS" = Poorzahedy, Abulghasemi and Rouhani's Ant System "RDO" = relative deviation from optimal

[a] Frequency of finding optimal solution in 50 runs

[b] Average time for each run

400 in a time much longer than that of FS-H&B. It is worthy to note that in all four budget scenarios, the convergent point of G-GBD is different from the system-level objective, although the solution in terms of the selected projects was optimal. Hence, to compute the objective value, the user optimal flows corresponding to the resulted network is obtained and put in the system-level objective. This algorithm cannot converge to the optimal solution in the next three budget levels. As mentioned in test problem 1, G-GBD does not guarantee to obtain the optimal solution due to its invalid upper-bound. In budget levels 500, 600, and 700, there relative deviation of G-GBD objective value from the optimal solution is 12.48 %, 3.93 %, and 1.69 %, respectively.

L-B&B can find global solutions in all cases in a CPU time much longer than that of FS-B&B. In budget level 700 where the budget constraint is not so restrictive, L-B&B terminates in a reasonable time. However, it is evident that in most practical cases, the budget constraint is very restrictive. In cases with moderate budget level L-B&B finds the optimal solution within 3 to 6 h. The number of computed bounds in L-B&B is high. This is due to the significant gap between the lower bounds of predecessor nodes and the incumbent solution. In fact, the lower bounds are not effective. This has also been previously pointed by (Poorzahedy and Turnquist 1982). In contrast, FS-B&B, due to its tight lower bounds, explores fewer nodes and computes fewer lower-bounds.

As Table 4 shows, FS-B&B finds the optimal solution of the test problem 3 in all cases in the shortest CPU time. Briefly, it outperforms all other algorithms in both computational cost and solution quality. The time FS-B&B needs to compute lower bounds varies around 0.5 to 2 s in different budget scenarios. The total number of computed lower-bounds is small. This is partly due to effective branching variable selection and node selection, and partly due to relatively tight lower bounds in predecessor nodes.

Based on the above comparisons, test problem 3 reveals some drawbacks of the existing algorithms, including disability of finding optimal or near optimal solution in many cases in a reasonable time. This simple test problem, the key feature of which is having enough number of new projects to challenge enumerative algorithms, exhibits the shortcomings of the previous algorithms.

Test problem 4 This test problem is designed so that the network has enough nodes, links, and OD pairs to challenge the proposed algorithm (Fig. 5). We arranged 4 copies of Sioux-Falls network in a 4-cell layout and connected some adjacent nodes. We also added some new nodes and links to the network. The parameters of the links are similar to those of Sioux-Falls network. The results of employing all above-mentioned algorithms have been presented in Table 5. Columns of Table 5 are same as those of Table 4. This problem is challenging for all algorithms due to the number of links and OD pairs.

The results show that G-GBD does not find the optimal solution in any budget level. Since the user optimal flows are very different from the system optimal ones, the computed Lagrangian multipliers from the lower-level problem are not valid for the upper-level problem and this, in turn, leads G-GBD to generate invalid upper-bounds. It should be noted that computing optimal Lagrangian multipliers in the lower-level problem has its own computational costs. G-GBD does not converge

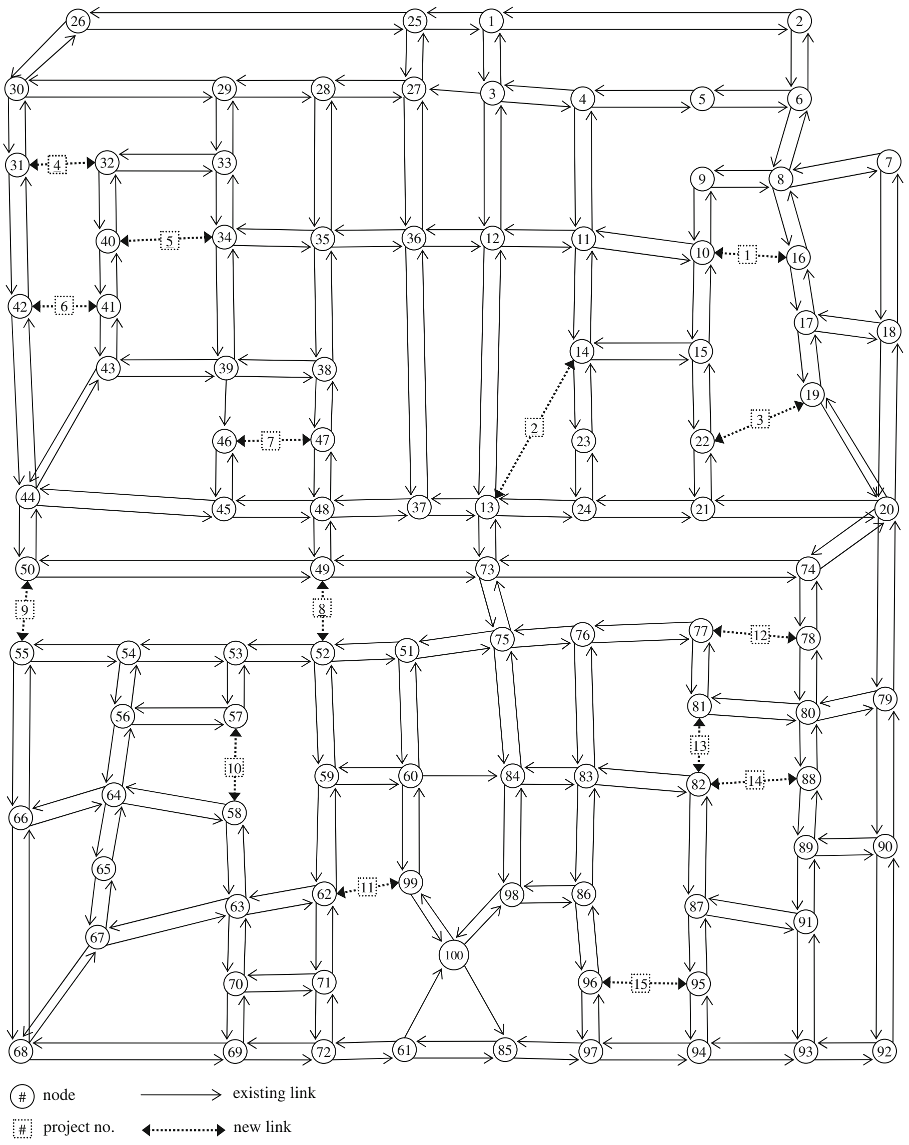


Fig. 5 Test network 4

even in 3000 iterations in budget levels 500 and 600. The gap between the objective value resulted from G-GBD and the optimal one, considering the system-level costs, is significant. It is worthy to note that even 1 % improvement in system-level costs in the network for real cases is remarkable. Algorithm PAR-AS provides the worst quality solutions. By considering solutions of PAR-AS in test problem 3 and 4 over various budget levels, it is implied that this algorithm tends to select more projects than other algorithms. Comparing the performance of L-B&B with G-GBD and PAR-AS indicates that this algorithm outperforms both other algorithms in terms of CPU time and solution quality. In test problem 4 and in the budget level 700, L-B&B has

Table 5 The results of the proposed algorithm on test problem 4 in comparison with the previous study

B/C	Budget	Algorithm	Objective Value	RDO (%)	Solution	Consumed budget	Iteration/ bounding	CPU time (s)
0.10	100	L-B&B	3.1652×10^6	0	6,11	94	95	106
		PAR-AS	3.1652×10^6, (2/50)[a]	0	6,11	94	–	267[b]
		G-GBD	3.1814×10^6	0.51	2	97	15	118
		FS-B&B	3.1652×10^6	0	6,11	94	13	89
0.20	200	L-B&B	2.9175×10^6	0	6,9	163	264	293
		PAR-AS	2.9698×10^6, (0/50)[a]	1.79	8,14	168	–	267[b]
		G-GBD	2.9594×10^6	1.44	8,12	176	233	1934
		FS-B&B	2.9175×10^6	0	6,9	163	9	98
0.29	300	L-B&B	2.8500×10^6	0	6,8,9	267	782	887
		PAR-AS	2.9029×10^7, (0/50)[a]	1.85	3,5,9,14	286	–	268[b]
		G-GBD	2.8992×10^6	1.73	2,8,12	273	1081	9296
		FS-B&B	2.8500×10^6	0	6,8,9	267	29	329
0.39	400	L-B&B	2.7842×10^6	0	2,6,8,9	364	938	1116
		PAR-AS	2.9028×10^6, (0/50)[a]	4.26	3,4,8,14,15	373	–	267[b]
		G-GBD	2.8055×10^6	0.76	2,8,9,12	385	2649	23311
		FS-B&B	2.7842×10^6	0	2,6,8,9	364	10	112
0.49	500	L-B&B	2.7442×10^6	0	1,2,6,8,9,10	488	1003	1194
		PAR-AS	2.8369×10^6, (0/50)[a]	3.38	3,5,6,7,8,10,11,14	490	–	266[b]
		G-GBD	2.8332×10^6	3.24	1,4,5,7,8,10,13	465	3000[c]	27270
		FS-B&B	2.7442×10^6	0	1,2,6,8,9,10	488	51	564
0.59	600	L-B&B	2.7112×10^6	0	1,2,6,8,9,10,11,15	600	887	1047
		PAR-AS	2.7658×10^6, (0/50)[a]	2.01	2,3,8,10,12,13,14,15	591	–	270[b]
		G-GBD	2.8200×10^6	4.01	2,3,7,9,11,12,13,14	553	3000[c]	27390
		FS-B&B	2.7112×10^6	0	1,2,6,8,9,10,11,15	600	84	769
0.69	700	L-B&B	2.6876×10^6	0	1,2,6,8,9,10,11,12,15	672	728	881
		PAR-AS	2.7112×10^6, (0/50)[a]	1.06	1,2,3,6,8,9,12,13,14	671	–	267[b]
		G-GBD	2.7160×10^6	0.66	1,2,5,8,9,10,12,14,15	694	2437	22323
		FS-B&B	2.6876×10^6	0	1,2,6,8,9,10,11,12,15	672	207	1840

"L-B&B" = LeBlanc B&B, "G-GBD" = Gao et al.'s Generalized Benders Decomposition, "B/C" = Budget/Cost

"FS-B&B" = The proposed B&B "PAR-AS" = Poorzahedy, Abulghasemi and Rouhani's Ant System "RDO" = relative deviation from optimal

[a] Frequency of finding optimal solution in 50 runs

[b] Average time for each run

[c] G-GBD did not converge in 3000 iterations

the best performance in terms of CPU time. Nonetheless, high budget levels are not the main cases where practitioners need decision support tools for network design purposes.

Algorithm FS-B&B solves test problem 4 in 6 budget levels having the ratio B/C smaller than 0.6 with the best performance. The algorithm particularly works well in budget levels ranging from small to medium (with a B/C ratio smaller than 0.4). An interesting note about FS-B&B is its good quality first incumbent solution which is obtained in the root node. The first incumbent solution in all budget levels for test problem 4 and its relative deviation with the solution of FS-B&B (optimal solution), G-GBD, and PAR-AS is presented in Table 6. Relative deviation of first incumbent with a given algorithm X is calculated by:

$$\begin{aligned}&\textit{relative deviation}\\&=\frac{[(\text{objective value of first incumbent})-(\text{objective value of algorithm X})]}{\text{objective value of algorithm X}}\end{aligned} \tag{29}$$

Reviewing the results shows that except budget level 100, the first incumbent of FS-B&B was better than the solution of PAR-AS with a significant deviation. It was also better than the solution of G-GBD in 5 budget levels out of 7 budget levels with a considerable gap, and was the same as G-GBD's solution in the 2 remaining budget levels. Therefore, the first incumbent solution of FS-B&B can be a very good heuristic solution for DNDP. This seems worthy when the CPU time of obtaining the first incumbent is considered. The CPU time of obtaining the first incumbent in all budget scenarios of test problem 3 and 4 was shorter than 3 and 10 s, respectively.

Table 6 The relative deviation of the first incumbent of FS-B&B from the final solution of other algorithms

Budget	First Incumbent objective value	Relative deviation from the objective value of		
		FS-B&B	PAR-AS	G-GBD
100	3181379	−0.51 %	−0.51 %	0.00 %
200	2919748	−0.08 %	1.71 %	1.36 %
300	2862726	−0.44 %	1.40 %	1.27 %
400	2805472	−0.76 %	3.47 %	0.00 %
500	2770601	−0.95 %	2.39 %	2.26 %
600	2721557	−0.38 %	1.62 %	3.62 %
700	2689721	−0.08 %	0.98 %	0.58 %

"FS-B&B" = The proposed B&B

"PAR-AS" = Poorzahedy, Abulghasemi and Rouhani's Ant System

"G-GBD" = Gao et al.'s Generalized Benders Decomposition, Another important point regarding the proposed algorithm is that most of its total CPU time is spent on proving that the found incumbent solution is optimal. In other words, a common situation in the proposed algorithm is spending a large amount of computing time to improve the solution from a near-optimal to an optimal one. This is also true about lower bounds such that during the search most of the nodes have a tight lower bound which is close to the incumbent solution. If, for example, a maximum deviation of 0.2 % from the optimum solution is tolerable for the upper-level decision maker, the CPU time of algorithm termination in budget scenarios 6 and 7 drops from 769 and 1840 to 541 and 1283 s, respectively, while the optimal solutions remain unchanged.

The reported results for FS-B&B on all test problems are the best solutions obtained from the three before mentioned node selection strategies. In all test problems, the best first search as the node selection strategy was the most effective in terms of the number of both explored nodes and the total computed lower-bounds. In other words, choosing nodes with the lowest lower bound in each iteration leads to a node selection strategy with lowest computational cost. In other words, it is more likely to have the optimal solution in nodes with the smallest system objective under the system optimal flows. This is supported by (Poorzahedy and Turnquist, 1982) work where they found that system-level objective under the user and system optimal flows has a high correlation in about 500 different experimental networks.

The effectiveness of the proposed heuristic for finding branching variable against random branching variable selection was evaluated. In test problems 2–4 the average computation time saved by using the proposed heuristic against random variable selection was around 28 %, 39 %, and 47 %, respectively. Consequently, it is concluded that the proposed heuristic for branching variable selection has a significant effect on the total computation time of the algorithm.

5.2 Effects of congestion level on the performance of the algorithm

The computational cost of the proposed algorithm is strongly affected by the total number of computed lower bounds. The number of computed lower bounds is proportionate to the number of explored nodes of the B&B tree. The size of B&B tree is significantly affected by the quality of the lower bounds and the incumbent solution. The quality of the lower bounds and incumbent solution, however, is related to the level of congestion. To examine the effect of congestion, the algorithm is evaluated in different scenarios of congestion. To do that, Sioux-Falls network with 15 new one-way links (15 projects) is considered. It should be noted that we consider more new links in the network to explore more aspects of the algorithm, although these new links may not be possible to be added to the real Sioux-Falls network. The solution space of the corresponding DNDP consists of 2^{15} points when the budget constraint is neglected.

We used the demand matrix of Sioux-Fall network (Poorzahedy and Rouhani 2007) as the basis. To study the congestion effect, we consider multiple demand scenarios. These scenarios results from multiplying the basis demand matrix by a certain factor (Demand Factor). Demand factor varies in the range [0.25, 2]. Besides, available budget varies from 0.1 to 0.6 of the total cost of all proposed projects.

Table 7 Performance of the algorithm in different demand and budget scenarios

Demand Factor	B/C	CPU time (s)	Bounding	Optimal Objective	V/C ratio				Demand Factor	B/C	CPU time (s)	Bounding	Optimal Objective	V/C ratio			
					Min	Max	Mean	StDev						Min	Max	Mean	StDev
0.25	0.1	0.42	1	307	0.05	1.21	0.45	0.31	1.25	0.1	8.61	7	24,246	0.43	3.45	1.96	0.70
0.25	0.2	0.27	1	305					1.25	0.2	51.10	46	23,192				
0.25	0.3	0.23	1	305					1.25	0.3	72.70	47	22,167				
0.25	0.4	0.22	1	305					1.25	0.4	136.95	121	22,021				
0.25	0.5	0.22	1	305					1.25	0.5	151.56	149	21,300				
0.25	0.6	0.22	1	305					1.25	0.6	133.65	139	20,913				
0.5	0.1	0.57	1	2,029	0.09	1.58	0.82	0.41	1.5	0.1	11.38	7	52,949	0.54	4.11	2.35	0.82
0.5	0.2	0.26	1	2,020					1.5	0.2	43.52	8	45,644				
0.5	0.3	0.26	1	2,020					1.5	0.3	86.99	19	43,335				
0.5	0.4	0.25	1	2,020					1.5	0.4	159.93	62	42,143				
0.5	0.5	0.27	1	2,020					1.5	0.5	225.39	143	41,551				
0.5	0.6	0.25	1	2,020					1.5	0.6	161.13	95	39,388				
0.75	0.1	4.85	5	6,986	0.14	2.12	1.19	0.52	1.75	0.1	12.22	8	119,462	0.65	4.79	2.74	0.95
0.75	0.2	4.89	7	6,705					1.75	0.2	43.21	16	99,823				
0.75	0.3	12.05	22	6,620					1.75	0.3	76.52	7	88,036				
0.75	0.4	35.03	54	6,576					1.75	0.4	90.85	12	82,838				
0.75	0.5	13.47	29	6,453					1.75	0.5	244.50	81	82,222				
0.75	0.6	46.16	80	6,427					1.75	0.6	113.28	26	76,757				
1	0.1	6.15	5	13,254	0.25	2.78	1.57	0.61	2	0.1	14.32	8	353,628	0.75	5.47	3.14	1.08
1	0.2	5.05	7	12,754					2	0.2	32.69	9	257,685				
1	0.3	10.74	13	12,396					2	0.3	29.92	5	220,715				
1	0.4	52.50	56	12,351					2	0.4	75.63	6	193,430				
1	0.5	41.55	49	12,048					2	0.5	161.06	32	192,947				
1	0.6	60.24	91	11,976					2	0.6	122.46	24	171,769				

"B/C" = Budget/Cost, "Min" = Minimum, "Max" = Maximum, "StDev" = Standard deviation

Table 7 shows the result of applying the algorithm on all generated scenarios. Column 1 shows demand factor. Column 2 shows B/C ratio indicating the available budget over total cost of all projects. Columns 3, 4, and 5 show computation time, the number of solved lower bounding problems, and the optimal system-level objective (vehicle-hour) of each scenario, respectively. Columns 6–9 show basic statistics minimum, maximum, mean and standard deviation of volume/capacity of links of the network corresponding to each demand factor. These columns indicate some aspects of congestion level over the network. For example, in case of demand factor 1, the minimum and maximum of v/c ratio of all links are 0.25 and 2.78, respectively; while the mean and standard deviation of v/c ratio of all links are 1.75 and 0.61, respectively. These indexes show that in scenarios corresponding to demand factor 1 the level of congestion is high (level of service is low). On the other hand, these statistics show the low level of congestion in the case of demand factor 0.25.

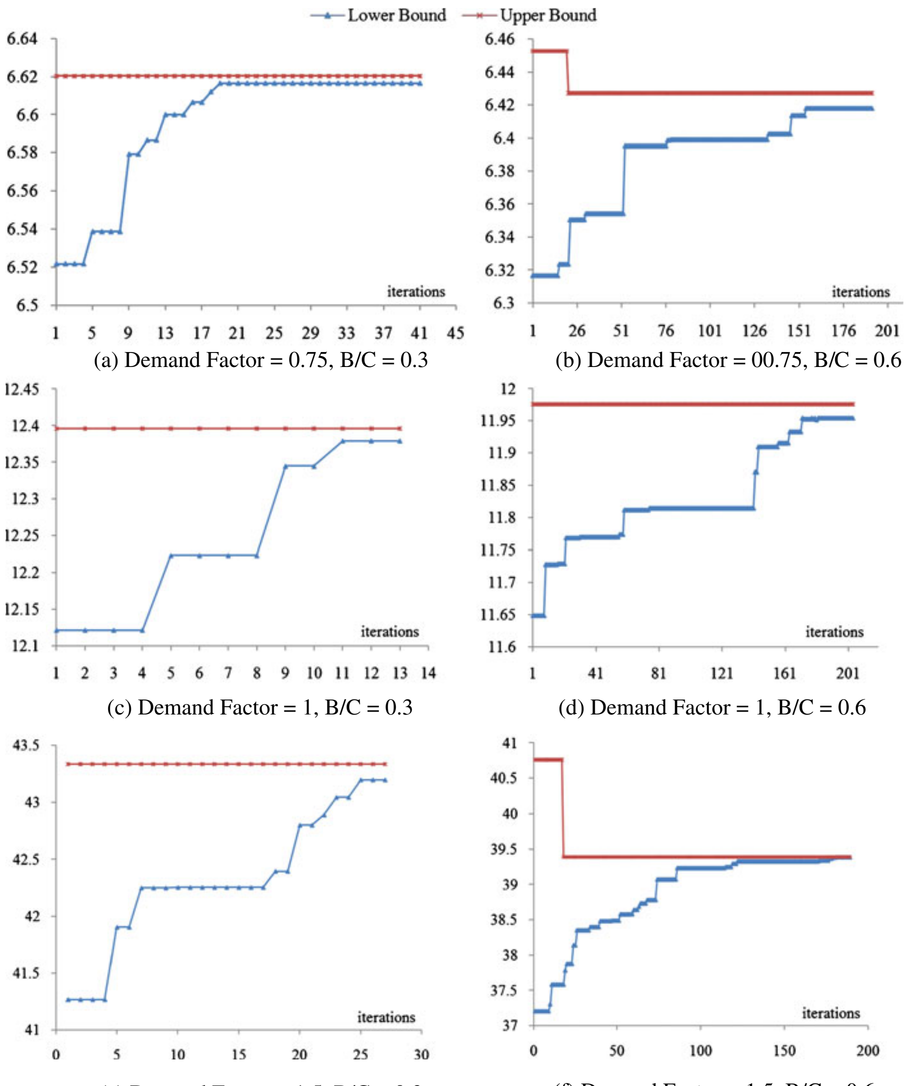
By reviewing the results of Table 7 some conclusions are drawn. The performance of the algorithm is significantly related to the level of congestion. In the low to medium congestion level (demand factor 0.25 and 0.5) algorithm converges to the optimal solution rapidly; while it converges to the optimal solution more slowly in the high congestion levels (demand factor 1 and 1.5). In cases where the congestion level is extremely high (demand factor 1.75 and 2) the performance of the algorithm in terms of both the computation time and the solved lower bounding problems is better than cases of high congestion level. In other words, in the networks with low to medium congestion, the quality of lower bounds is good. In highly congested networks, the quality of lower bounds gets worse, and it get improved again in severely congested networks. The level of available budget affects the convergence time of the algorithm in all demand scenarios negatively. Nevertheless, it seems that the impact of congestion on the performance of the algorithm is higher than the available budget. However, to scrutinize this feature of the algorithm more test problems and conditions should be examined.

Figure 6 shows the variations of the lower and upper bounds as the algorithm proceeds. In most cases, the algorithm finds the optimal solution in root node. In fact, most of its search is devoted to prove the optimality of the incumbent solution through updating the lower bound.

6 Conclusion

In this paper, a new branch and bound algorithm was proposed for the bi-level discrete network design problem. The computational performance and the solution quality of the proposed algorithm were evaluated on four test problems and compared with previous relevant algorithms. The results showed that the proposed algorithm outperforms previous algorithms in both CPU time and solution quality. It was experimentally shown that previous simple test problems are not good examinations to evaluate the performance of the existing algorithms. Therefore, two new test problems were devised to shed light on some drawbacks of the previous algorithms.

The main features of the proposed algorithm are: (a) it is complete, i.e. it can provide the optimal solution if there is any, although its time complexity may be high



(a) Demand Factor = 0.75, B/C = 0.3
(b) Demand Factor = 00.75, B/C = 0.6
(c) Demand Factor = 1, B/C = 0.3
(d) Demand Factor = 1, B/C = 0.6
(e) Demand Factor = 1.5, B/C = 0.3
(f) Demand Factor = 1.5, B/C = 0.6

Fig. 6 Upper and lower bound convergence in the different scenarios of demand and available budget

in large-scale problems. As a characteristic feature, unlike some previous algorithms, it guards against the Braess' paradox; (b) it benefits from good strategies of node selection and branching variable selection; (c) it benefits from a good lower-bounding scheme, although it imposes extra computational costs to the algorithm. However, a part of these computational costs is compensated by exploiting the lower-bounding step through extracting effective guides to node as well as branching variable selection.

The computational cost of the proposed algorithm is strongly affected by the total number of computed lower bounds. Further studies can focus on different parts of the lower bounding step. In the present work, a good algorithm was presented to compute lower bound based on outer approximation. The algorithm is composed of two parts, namely, master problem and sub-problem which are iteratively solved until they converge. Sub-problems are efficiently solved by today's traffic assignment algorithms, but master problems may be difficult in cases having a high number of new candidate links in a large network (e.g. more than 100 new links in a network with more than 1000 links). Employing Bush-based algorithms, like OBA (Bar-Gera, 2002) and DBA (Nie 2010) for solving sub-problems can lead to faster convergence, especially for large urban networks due to their superiority in memory usage and computation time and also producing highly precise UE solutions. In the current study, CPLEX solver was employed to solve master problems. Considering the structure of the problem having a lot of multicommodity network flow constraints, it is promising to develop specialized algorithms to solve the master problems more efficiently. This can be studied in future works. The quality of lower bounds seems good. By considering the fact that generally no explicit relationship has yet been found between system and user optimal flows (Karakostas et al. 2011), lower bounds are not expected to be improved significantly. Node selection and branching variable selection strategies could be improved. In this research best first strategy had the best performance comparing to depth first and breathe first search strategies. Yet, evaluating a combination of these search strategies needs more investigation. The branching variable selection is also a candidate to be more studied in order to adopt more effective variable selection strategy. In the current setting of the algorithm half of the lower bounds of the total explored nodes need to be computed because of the relationship between child and parent nodes of the B&B search tree. However, the problems which are solved to compute lower bounds seem to have the potential to reduce the computed lower bounds through finding better relationships among downstream and upstream nodes. Finding such relations among the solutions of lower bounding problems in different locations of the B&B search tree can be an area which merits further research.

In the present work we focused on DNDP with deterministic traffic assignment in the lower-level. The proposed algorithm can be easily adapted to non-deterministic traffic assignments. In this case, the lower bounding step is affected such that the sub-problems are non-deterministic user optimal traffic assignments. The computational efficiency and more customization of the algorithm to such a model should be studied in future studies. Finally, employing heuristics/meta-heuristics algorithms to have a cooperative search seems to be good practice to make the algorithm more scalable. The support to this idea is the feasible and near-optimal solution of DNDP in the root node of the proposed B&B. This is appropriate for neighborhood search to move to better solutions in a shorter time. Thus, as an area for further research, combining heuristics/meta-heuristics algorithm with the proposed global method to exploit advantages of both methods in order to provide a scalable method for large-scale DNDP is suggested.

Acknowledgments The authors would like to thank the three anonymous referees for many thoughtful and constructive comments. We also wish to thank Mr. Fouad Behzadpour for his grammatical and lexical revisions and comments. We accept full responsibility for all the remaining errors and shortcomings.

Appendix A. The network data of test problem 4

The link parameters and OD trips data of test problem 4 are presented in Tables 8 and 9, respectively. In Table 8 each link is shown in the form of *i* <=> *j*: (T, B, C, g), where *i* and *j* are tail and head nodes, T is free-flow, B is the congestion factor, C is the capacity corresponding to each link, respectively. The forth element, g, is the cost of new projects. For existing links the value of g is left blank. The symbols <=> and -> show two-way and one-way streets, respectively.

Table 8 Link parameters of test network 4

1<=>2: (6, 0.15, 25900.2, -)	34<=>39: (6, 0.15, 13512, -)	70<=>71: (4, 0.15, 5000, -)
1<=>3: (4, 0.15, 23403.47, -)	35<=>36: (6, 0.15, 4908.83, -)	71<=>72: (2, 0.15, 5078.51, -)
1<=>25: (4, 0.15, 14360.52, -)	35<=>38: (4, 0.15, 4876.51, -)	73<=>74: (6, 0.15, 25900.2, -)
2<=>6: (5, 0.15, 4958.18, -)	36<=>37: (3, 0.15, 25900.2, -)	73<=>75: (4, 0.15, 23403.47, -)
3<=>4: (4, 0.15, 17110.52, -)	37<=>48: (4, 0.15, 5091.26, -)	74<=>78: (5, 0.15, 4958.18, -)
3<=>12: (4, 0.15, 23403.47, -)	38<=>39: (5, 0.15, 5127.53, -)	75<=>76: (4, 0.15, 17110.52, -)
3->27: (4, 0.15, 14110.52, -)	38<=>47: (4, 0.15, 4924.79, -)	75<=>84: (4, 0.15, 23403.47, -)
4<=>5: (2, 0.15, 17782.79, -)	39<=>43: (3, 0.15, 14564.75, -)	76<=>77: (2, 0.15, 17782.79, -)
4<=>11: (6, 0.15, 4908.83, -)	39->46: (3, 0.15, 9599.18, -)	76<=>83: (6, 0.15, 4908.83, -)
5<=>6: (4, 0.15, 4948, -)	40<=>41: (2, 0.15, 5229.91, -)	77<=>81: (5, 0.15, 10000, -)
6<=>8: (2, 0.15, 4898.59, -)	41<=>43: (2, 0.15, 4823.95, -)	78<=>80: (2, 0.15, 4898.59, -)
7<=>8: (3, 0.15, 7841.81, -)	42<=>44: (4, 0.15, 23403.47, -)	79<=>80: (3, 0.15, 7841.81, -)
7<=>18: (2, 0.15, 23403.47, -)	43<=>44: (4, 0.15, 5002.61, -)	79<=>90: (2, 0.15, 23403.47, -)
8<=>9: (10, 0.15, 5050.19, -)	44<=>45: (6, 0.15, 5059.91, -)	80<=>81: (10, 0.15, 5050.19, -)
8<=>16: (5, 0.15, 5045.82, -)	44<=>50: (2, 0.15, 9560.18, -)	80<=>88: (5, 0.15, 5045.82, -)
9<=>10: (3, 0.15, 13915.79, -)	45<=>46: (2, 0.15, 5229.91, -)	82<=>83: (5, 0.15, 10000, -)
10<=>11: (5, 0.15, 10000, -)	45<=>48: (3, 0.15, 4885.36, -)	82<=>87: (6, 0.15, 13512, -)
10<=>15: (6, 0.15, 13512, -)	47<=>48: (2, 0.15, 5078.51, -)	83<=>84: (6, 0.15, 4908.83, -)
11<=>12: (6, 0.15, 4908.83, -)	48<=>49: (2, 0.15, 9560.18, -)	83<=>86: (4, 0.15, 4876.51, -)
11<=>14: (4, 0.15, 4876.51, -)	49<=>50: (6, 0.15, 25900.2, -)	84<=>98: (4, 0.15, 5824.79, -)
12<=>13: (3, 0.15, 25900.2, -)	49<=>73: (4, 0.15, 23403.47, -)	85<=>97: (4, 0.15, 4191.26, -)
12<=>36: (3, 0.15, 5891.26, -)	51<=>52: (4, 0.15, 17110.52, -)	86<=>96: (4, 0.15, 4991.26, -)
13<=>24: (4, 0.15, 5091.26, -)	51<=>60: (4, 0.15, 23403.47, -)	86<=>98: (6, 0.15, 4958.83, -)
13<=>37: (3, 0.15, 5091.26, -)	51<=>75: (4, 0.15, 6124.52, -)	87<=>91: (3, 0.15, 14564.75, -)
13<=>73: (2, 0.15, 9560.18, -)	52<=>53: (2, 0.15, 17782.79, -)	87<=>95: (3, 0.15, 8199.18, -)
14<=>15: (5, 0.15, 5127.53, -)	52<=>59: (6, 0.15, 4908.83, -)	88<=>89: (2, 0.15, 5229.91, -)
14<=>23: (4, 0.15, 4924.79, -)	53<=>54: (4, 0.15, 4948, -)	89<=>90: (2, 0.15, 12823.95, -)
15<=>22: (3, 0.15, 9599.18, -)	53<=>57: (5, 0.15, 10000, -)	89<=>91: (2, 0.15, 4823.95, -)
16<=>17: (2, 0.15, 5229.91, -)	54<=>55: (3, 0.15, 5248, -)	90<=>92: (4, 0.15, 23403.47, -)
17<=>18: (3, 0.15, 14424.75, -)	54<=>56: (2, 0.15, 4898.59, -)	91<=>93: (4, 0.15, 5002.61, -)
17<=>19: (2, 0.15, 4823.95, -)	55<=>66: (2, 0.15, 23403.47, -)	92<=>93: (6, 0.15, 5059.91, -)
18<=>20: (4, 0.15, 23403.47, -)	56<=>57: (10, 0.15, 5050.19, -)	93<=>94: (2, 0.15, 5229.91, -)
19<=>20: (4, 0.15, 5002.61, -)	56<=>64: (5, 0.15, 5045.82, -)	94<=>95: (4, 0.15, 5000, -)
20<=>21: (6, 0.15, 5059.91, -)	58<=>63: (6, 0.15, 13512, -)	94<=>97: (4, 0.15, 4924.79, -)

Table 8 (continued)

20<=>74: (2, 0.15, 9560.18, -)	58<=>64: (4, 0.15, 4854.92, -)	96<=>97: (4, 0.15, 5100, -)
20<=>79: (5, 0.15, 22403.47, -)	59<=>60: (6, 0.15, 9908.83, -)	98<=>100: (2, 0.15, 6824.79, -)
21<=>22: (2, 0.15, 5229.91, -)	59<=>62: (4, 0.15, 4876.51, -)	99<=>100: (2, 0.15, 6824.63, -)
21<=>24: (3, 0.15, 4885.36, -)	60->84: (4, 0.15, 6900.2, -)	100->85: (4, 0.15, 5071.26, -)
23<=>24: (2, 0.15, 5078.51, -)	60<=>99: (4, 0.15, 5824.79, -)	10<=>16: (4, 0.15, 4554.92, 65)
25<=>26: (6, 0.15, 25900.2, -)	61<=>72: (4, 0.15, 5091.26, -)	13<=>14: (2, 0.15, 23523.47, 97)
25<=>27: (4, 0.15, 23403.47, -)	61<=>85: (3, 0.15, 6724.79, -)	19<=>22: (3, 0.15, 17780.81, 58)
26<=>30: (5, 0.15, 4958.18, -)	61->100: (4, 0.15, 6524.79, -)	31<=>32: (3, 0.15, 17841.81, 78)
27<=>28: (4, 0.15, 17110.52, -)	62<=>63: (5, 0.15, 5127.53, -)	34<=>40: (4, 0.15, 14854.92, 52)
27<=>36: (4, 0.15, 23403.47, -)	62<=>71: (4, 0.15, 4924.79, -)	41<=>42: (3, 0.15, 7731.81, 51)
28<=>29: (2, 0.15, 17782.79, -)	63<=>67: (3, 0.15, 14564.75, -)	46<=>47: (3, 0.15, 15000, 59)
28<=>35: (6, 0.15, 4908.83, -)	63<=>70: (3, 0.15, 9599.18, -)	49<=>52: (3, 0.15, 13898.59, 104)
29<=>30: (4, 0.15, 4948, -)	64<=>65: (2, 0.15, 5229.91, -)	50<=>55: (4, 0.15, 12057.82, 112)
29<=>33: (5, 0.15, 10000, -)	64<=>66: (3, 0.15, 19679.9, -)	57<=>58: (3, 0.15, 13915.79, 59)
30<=>31: (2, 0.15, 4898.59, -)	65<=>67: (2, 0.15, 4823.95, -)	62<=>99: (4, 0.15, 5524.79, 43)
31<=>42: (2, 0.15, 23403.47, -)	66<=>68: (4, 0.15, 23403.47, -)	77<=>78: (2, 0.15, 17782.79, 72)
32<=>33: (10, 0.15, 5050.19, -)	67<=>68: (4, 0.15, 5002.61, -)	81<=>82: (4, 0.15, 14854.92, 48)
32<=>40: (5, 0.15, 5045.82, -)	68<=>69: (6, 0.15, 5059.91, -)	82<=>88: (4, 0.15, 6124.52, 64)
33<=>34: (5, 0.15, 5345.82, -)	69<=>70: (2, 0.15, 5229.91, -)	95<=>96: (2, 0.15, 5078.51, 69)
34<=>35: (5, 0.15, 10000, -)	69<=>72: (3, 0.15, 4885.36, -)	

Table 9 OD table. r-s:d_r^s

1-27: 20	10-54: 120	17-100: 780	27-69: 40	35-9: 120	42-54: 20	49-83: 80	56-27: 40	63-21: 120	72-52: 40	79-36: 140	85-59: 200	94-74: 20
1-28: 100	10-78: 160	18-4: 20	27-73: 20	35-19: 60	42-74: 240	49-89: 120	56-51: 40	63-24: 60	72-67: 20	79-38: 40	85-91: 60	94-91: 240
1-34: 260	10-81: 560	18-40: 100	27-85: 40	35-40: 280	42-78: 20	50-11: 280	56-70: 100	63-29: 40	72-76: 100	79-74: 40	85-96: 160	95-4: 40
1-37: 100	10-84: 400	18-45: 20	28-9: 160	35-45: 80	42-83: 320	50-16: 160	56-73: 60	63-44: 220	72-84: 40	79-78: 80	86-6: 100	95-31: 40
1-39: 100	10-85: 380	18-47: 20	28-13: 600	35-60: 200	42-88: 140	50-27: 20	56-76: 80	63-71: 200	73-5: 100	79-94: 100	86-14: 140	95-35: 260
1-56: 100	10-86: 420	18-53: 20	28-22: 40	35-63: 100	42-95: 220	50-33: 40	56-83: 80	63-78: 20	73-6: 80	79-95: 40	86-41: 140	95-36: 140
1-57: 40	10-88: 880	19-9: 80	28-27: 40	35-65: 80	43-5: 20	50-34: 120	57-16: 280	64-4: 20	73-11: 280	80-6: 80	86-47: 220	95-42: 20
1-60: 160	11-6: 80	19-23: 60	28-38: 100	35-67: 100	43-12: 100	50-55: 380	57-31: 120	64-10: 140	73-23: 100	80-10: 780	86-51: 20	95-51: 20
1-86: 60	11-37: 200	19-47: 60	28-55: 60	35-81: 120	43-13: 120	50-59: 40	57-32: 160	64-20: 80	73-38: 60	80-20: 120	86-77: 20	95-57: 100
1-95: 60	11-71: 80	19-49: 60	28-61: 80	35-84: 140	43-40: 260	50-73: 80	57-37: 120	64-29: 100	73-41: 80	80-27: 40	86-78: 20	95-63: 200
1-100: 600	11-77: 100	19-67: 160	28-75: 20	35-95: 40	43-48: 20	50-77: 40	57-58: 160	64-34: 880	73-42: 20	80-40: 440	86-82: 420	95-67: 60
3-14: 20	11-100: 780	19-73: 80	28-97: 100	36-10: 380	43-49: 100	50-91: 240	57-71: 100	64-37: 120	73-45: 20	80-41: 280	86-98: 100	95-76: 100
3-23: 20	12-6: 40	19-78: 40	28-100: 240	36-29: 15	43-51: 20	51-13: 40	58-8: 120	64-48: 60	73-56: 160	80-61: 120	87-13: 20	95-80: 60
3-33: 20	12-15: 140	19-88: 260	29-5: 40	36-45: 60	43-64: 140	51-15: 40	58-14: 140	64-50: 80	73-57: 100	80-62: 80	87-61: 140	95-96: 160
3-39: 20	12-33: 120	19-99: 20	29-20: 60	36-46: 140	43-82: 800	51-17: 40	58-41: 780	64-51: 40	73-58: 260	80-63: 120	87-65: 300	96-4: 60
3-60: 40	12-47: 140	20-5: 20	29-60: 160	36-53: 160	43-91: 160	51-21: 20	58-43: 360	64-52: 160	73-76: 100	80-77: 100	87-77: 40	96-5: 20
3-86: 20	12-64: 600	20-28: 60	29-70: 20	36-72: 180	44-8: 80	51-28: 40	58-46: 520	64-77: 600	74-41: 40	80-87: 120	87-80: 120	96-22: 480
4-24: 40	13-14: 280	20-43: 240	29-72: 60	36-77: 100	44-11: 80	51-37: 20	58-49: 260	64-88: 260	74-55: 40	80-88: 440	87-88: 240	96-45: 100
4-27: 40	13-20: 120	20-81: 120	29-80: 160	36-83: 160	44-12: 60	51-38: 20	58-51: 60	65-6: 40	74-59: 40	80-96: 40	87-94: 520	96-64: 60
4-29: 100	13-29: 40	21-6: 20	29-82: 260	36-88: 440	44-17: 120	51-40: 40	58-54: 160	65-9: 80	74-76: 40	80-99: 100	87-99: 40	96-80: 40
4-77: 100	13-32: 120	21-11: 80	29-91: 60	36-89: 280	44-20: 240	51-47: 20	58-57: 560	65-11: 80	74-87: 20	81-4: 120	87-100: 40	96-84: 100
4-81: 140	13-48: 160	21-24: 100	29-93: 20	37-3: 20	44-34: 500	51-61: 20	58-84: 60	65-28: 100	74-100: 80	81-70: 140	88-23: 60	96-94: 220
4-82: 240	13-50: 60	21-40: 120	29-95: 60	37-5: 20	44-40: 320	51-62: 20	58-98: 40	65-42: 120	75-8: 160	81-71: 100	88-27: 40	96-95: 340
4-86: 100	13-54: 60	21-41: 120	30-4: 80	37-19: 60	44-58: 180	51-64: 40	59-15: 140	65-59: 200	75-16: 180	81-73: 100	88-28: 160	97-5: 20
5-28: 100	13-81: 120	21-76: 40	30-8: 200	37-36: 260	44-80: 440	51-70: 20	59-22: 260	65-71: 120	75-34: 60	81-97: 20	88-32: 440	97-16: 120

Table 9 (continued)

5-39: 40	13-86: 120	21-94: 360	30-12: 140	37-38: 120	44-84: 140	51-83: 260	59-23: 160	65-78: 60	75-38: 20	82-33: 560	88-41: 560	97-17: 120
5-53: 40	14-10: 420	22-21: 16	30-20: 100	37-45: 120	44-89: 560	51-89: 120	59-32: 160	67-44: 240	75-40: 40	82-35: 800	88-42: 100	97-34: 400
5-55: 20	14-13: 192	22-33: 140	30-28: 80	37-49: 100	44-91: 260	51-100: 20	59-42: 20	67-50: 20	75-41: 20	82-40: 880	88-45: 120	97-39: 140
5-61: 160	14-16: 140	22-47: 420	30-34: 160	37-53: 100	44-97: 40	52-4: 80	59-48: 120	67-72: 20	75-50: 20	82-56: 320	88-46: 240	97-41: 120
5-72: 20	14-20: 100	22-54: 20	30-43: 40	37-67: 180	44-98: 160	52-12: 40	59-54: 80	67-87: 520	75-53: 20	82-62: 420	88-50: 80	97-59: 280
5-85: 40	14-22: 240	22-56: 80	30-47: 20	37-70: 40	45-16: 240	52-17: 100	59-100: 60	67-96: 220	75-65: 20	82-77: 200	88-57: 280	97-64: 280
5-91: 20	14-60: 80	22-63: 220	30-77: 20	37-73: 60	45-19: 240	52-24: 20	60-5: 20	67-98: 80	75-80: 600	82-81: 560	88-67: 260	97-65: 180
5-98: 100	14-61: 120	22-64: 140	30-85: 60	37-85: 120	45-35: 80	52-49: 160	60-13: 120	69-3: 20	75-82: 60	82-84: 400	88-70: 240	97-67: 80
6-16: 180	14-74: 240	22-70: 60	31-17: 280	37-93: 60	45-44: 240	52-53: 100	60-27: 40	69-12: 140	76-12: 140	82-86: 420	88-82: 20	98-9: 140
6-33: 80	14-85: 120	22-80: 100	31-27: 20	38-17: 300	45-52: 100	52-56: 140	60-39: 140	69-29: 20	76-13: 80	82-88: 40	88-93: 120	98-11: 220
6-48: 20	14-93: 80	22-88: 240	31-32: 280	38-37: 120	45-57: 40	52-67: 40	60-47: 140	69-34: 240	76-15: 100	82-89: 780	89-11: 120	98-48: 160
6-62: 160	15-6: 40	22-89: 340	31-42: 40	38-40: 140	45-99: 40	52-71: 100	60-49: 40	69-36: 60	76-27: 40	83-11: 320	89-59: 200	98-67: 360
6-80: 160	15-12: 140	23-44: 140	31-56: 40	38-51: 600	45-100: 240	52-84: 100	60-54: 40	69-47: 140	76-37: 120	83-21: 80	89-78: 100	98-81: 140
6-87: 40	15-29: 600	23-52: 100	31-60: 40	38-85: 380	46-10: 360	53-31: 40	60-55: 140	69-55: 40	76-39: 100	83-24: 80	91-10: 520	98-87: 100
6-93: 20	15-40: 240	23-64: 140	31-74: 20	38-89: 780	46-22: 420	53-44: 20	60-83: 80	69-59: 80	76-50: 40	83-32: 160	91-14: 240	98-94: 80
6-100: 160	15-42: 40	23-71: 60	31-78: 60	39-10: 880	46-38: 240	53-65: 40	60-85: 120	69-76: 40	76-86: 100	83-33: 280	91-15: 520	98-97: 360
7-5: 40	15-43: 160	23-82: 360	31-80: 40	39-17: 560	46-48: 220	53-71: 20	60-86: 600	69-80: 40	76-95: 100	83-41: 200	91-36: 60	99-14: 220
7-20: 100	15-51: 20	24-8: 40	32-8: 160	39-19: 260	46-49: 20	53-95: 20	60-91: 80	69-84: 100	76-97: 40	83-57: 280	91-58: 360	99-22: 420
7-58: 80	15-69: 300	24-14: 80	32-12: 120	39-59: 100	46-72: 80	53-99: 140	61-4: 100	69-93: 100	77-21: 80	83-75: 60	91-65: 340	99-27: 20
7-85: 80	15-75: 700	24-15: 80	32-15: 180	40-6: 100	46-76: 20	54-13: 120	61-14: 260	69-95: 140	77-43: 20	83-80: 160	91-69: 80	99-31: 40
7-86: 40	16-5: 100	25-19: 20	32-17: 180	40-41: 560	46-86: 20	54-49: 60	61-31: 80	70-15: 80	77-51: 20	83-84: 280	91-85: 60	99-50: 40
8-13: 120	16-10: 152	25-27: 20	32-31: 160	40-50: 20	46-87: 40	54-50: 80	61-56: 120	70-16: 60	77-54: 40	83-86: 320	91-96: 20	99-67: 80
8-15: 120	16-21: 120	25-61: 20	32-53: 100	40-61: 120	46-98: 20	54-51: 60	61-77: 40	70-19: 20	77-57: 160	83-91: 80	93-22: 220	99-80: 100
8-29: 18	16-31: 280	25-85: 20	32-73: 40	40-65: 260	47-4: 40	54-52: 80	62-3: 40	70-40: 240	77-63: 40	83-93: 80	93-28: 40	99-91: 20
8-41: 280	16-48: 60	25-86: 20	33-8: 320	40-73: 60	47-13: 140	54-63: 40	62-4: 160	70-58: 520	77-78: 120	84-4: 100	93-31: 40	100-8: 40
8-56: 140	16-72: 320	25-100: 260	33-13: 380	40-87: 140	47-28: 100	54-75: 40	62-27: 20	70-61: 260	77-84: 40	84-11: 280	93-33: 60	100-13: 140

Table 9 (continued)

8-83: 160	16-83: 280	26-17: 20	33-27: 20	41-8: 60	47-34: 360	54-85: 120	62-29: 20	70-62: 240	77-98: 100	84-13: 140	93-39: 160	100-21: 100
8-95: 60	16-97: 40	26-36: 20	33-73: 20	41-15: 40	47-36: 140	55-22: 140	62-31: 40	70-67: 240	78-16: 280	84-37: 260	93-44: 240	100-28: 100
9-19: 80	16-98: 160	26-51: 40	33-88: 100	41-20: 80	47-41: 120	55-43: 80	62-36: 140	70-71: 420	78-49: 60	84-42: 40	93-53: 20	100-57: 120
9-46: 140	17-8: 280	26-58: 80	34-3: 60	41-29: 40	47-77: 20	55-46: 100	62-43: 60	71-3: 20	78-50: 80	84-52: 120	93-54: 20	100-77: 40
9-49: 100	17-15: 300	26-89: 100	34-9: 280	41-53: 100	47-89: 340	55-50: 100	62-44: 100	71-14: 20	78-64: 180	84-55: 140	93-67: 80	100-80: 160
9-77: 160	17-21: 120	26-95: 100	34-10: 780	41-54: 60	48-36: 100	55-58: 380	62-45: 80	71-31: 40	78-71: 20	84-64: 140	93-70: 360	100-83: 80
10-4: 240	17-29: 40	26-100: 120	34-21: 600	41-56: 120	49-32: 160	55-70: 100	62-49: 60	71-33: 100	78-77: 80	84-65: 120	93-80: 80	100-88: 180
10-16: 280	17-46: 340	27-22: 80	34-22: 220	41-57: 40	49-41: 80	55-82: 200	62-63: 260	71-35: 260	78-85: 40	84-95: 140	93-95: 140	100-89: 100
10-22: 520	17-50: 40	27-37: 20	34-24: 120	41-82: 380	49-52: 220	55-97: 140	62-73: 20	71-58: 360	79-11: 800	85-24: 60	94-4: 20	100-96: 20
10-44: 500	17-55: 20	27-38: 20	34-41: 780	42-5: 20	49-62: 60	55-98: 420	62-82: 140	71-89: 100	79-19: 360	85-41: 100	94-43: 240	100-99: 140
10-45: 240	17-63: 200	27-46: 20	34-53: 60	42-35: 40	49-70: 80	56-8: 320	63-4: 100	72-33: 40	79-22: 520	85-53: 40	94-45: 360	
10-50: 120	17-80: 280	27-47: 20	34-81: 80	42-45: 20	49-82: 240	56-19: 360	63-9: 180	72-44: 80	79-23: 360	85-55: 80	94-56: 100	

References

Abdulaal M, LeBlanc LJ (1979) Continous equilibrium network design models. Transp Res 13B:19–32

Bar-Gera H (2002) Origin-based algorithm for the traffic assignment problem. Transp Sci 36:398–417

Boyce DE (1984) Urban transportation netwrok equilibrium and design models: recent achievements and future prospectives. Environ Plann A 16:1445–1474

Boyce DE, Farhi A, Weischedel R (1973) Optimal network problem: a branch-and-bound algorithm. Environ Plann 5:519–533

Chung BD, Yao T, Xie C, Thorsen A (2011) Robust optimization model for a dynamic network design problem under demand uncertainty. Netw Spat Econ 11:371–389

Dantzig GB, Harvey RP, Lansdowne ZF, Robinson DW, Maier SF (1979) Formulating and solving the network design problem by decomposition. Transp Res 13B:5–17

Drezner T, Salhi S (2002) Using hybrid metaheuristics for the one-way and two-way network design problem. Nav Res Log 49:449–463

Duan L, Xiaoling S (2006) Nonlinear integer programming. Springer, New York

Duran MA, Grossmann IE (1986) An outer algorithm for a class of mixed integer nonlinear programs. Math Program 36:307

Farvaresh H, Sepehri MM (2011) A single-level mixed integer linear formulation for a bi-level discrete network design problem. Transp Res 47E:623–640

Fletcher R, Leyffer S (1994) Solving mixed integer nonlinear programs by outer approximation. Math Program 66:327

Floudas C (1995) Nonlinear and mixed-integer optimization-fundamentals and applications. Oxford University Press, New York

Floudas CA, Pardalos PM (2009) Encyclopedia of optimization, 2nd edn. Springer, New York

Friesz TL (1985) Transportation network equilibrium, design and aggregation: key developments and research opportunities. Transp Res 19A:413–427

Friesz TL, Cho HJ, Mehta NJ, Tobin RL, Anadalingam G (1992) A simulated annealing approach to the network design problem with variational inequality constraints. Transp Sci 26:18–26

Gao Z, Wu J, Sun H (2005) Solution algorithm for the bi-level discrete network design problem. Transp Res 39B:479–495

Geoffrion AM (1972) Generalized Benders decomposition. J Optimiz Theory App 10:237–260

Haghani AE, Daskin MS (1983) Network design application of an extraction algorithm for network aggregation. Transp Res Rec 944:37–46

Holmberg K, Hellstrand J (1998) Solving the uncapacitated network design problem by a Lagrangian heuristic and branch-and-bound. Oper Res 46:247–259

Ibaraki T (1987) Enumerative approaches to combinatorial optimization. Ann Oper Res 10(11)

Karakostas G, Taeyon K, Anastasios V, Xia H (2011) On the degradation of performance for traffic networks with oblivious users. Transp Res 45B: 364–371

Karoonsoontawong A, Waller ST (2006) Dynamic continuous network design problem: linear bilevel programming and metaheuristic approaches. Transp Res Rec 1964:104–117

Kim BJ, Kim W (2006) An equilibrium network design model with a social cost function for multimodal networks. Ann Regional Sci 40:473–491

Leblanc LJ (1975) An algorithm for the discrete network design problem. Transp Sci 9:183–199

LeBlanc LJ, Boyce DE (1986) A bilevel programming algorithm for exact solution of the network design problem with user-optimal flows. Transp Res 20B:259–265

Leblanc LJ, Morlok EK, Pierskalla WP (1975) An efficient approach to solving the road network equilibrium traffic assignment problem. Transp Res 9B:309–318

Lee CK, Yang KI (1994) Network design of one-way streets with simulated annealing. Pap Reg Sci 32 (2):119–134

Lin DY, Karoonsoontawong A, Waller ST (2011) Dantzig-Wolfe decomposition based heuristic scheme for bi-level dynamic network design problem. Netw Spat Econ 11:101–126

Luathep P, Sumalee A, Lam WHK, Li Z-C, Lo HK (2011) Global optimization method for mixed transportation network design problem: a mixed-integer linear programming approach. Transp Res 45B:808–827

Magnanti TL, Wong RT (1984) Network design and transportation planning: models and algorithms. Transp Sci 18:1–55

Meng Q, Yang H, Bell MGH (2001) An equivalent continuously differentiable model and a locally convergent algorithm for the continuous network design problem. Transp Res 35B:83–105

Miandoabchi E, Zanjirani FR, Dullaert W, Szeto WY (2011) Hybrid evolutionary metaheuristics for concurrent multi-objective design of urban road and public transit networks. Netw Spat Econ. doi:10.1007/s11067-011-9163-x

Nie YM (2010) A class of bush-based algorithms for the traffic assignment problem. Transp Res 44B:73–89

Patriksson M (1994) The traffic assignment problems: models and methods. VSP, Utercht

Poorzahedy H, Abulghasemi F (2005) Application of ant system to network design problem. Transportation 32:251–273

Poorzahedy H, Rouhani OM (2007) Hybrid meta-heuristic algorithms for solving network design problem. Eur J Oper Res 182:578–596

Poorzahedy H, Turnquist MA (1982) Approximate algorithms for the discrete network design problem. Transp Res 16B:45–55

Sheffi Y (1985) Urban transportation network: equilibrium analysis with mathematical programming methods. Prentice Hall, England Cliffs

Steenbrink A (1974a) Transport network optimization in the Dutch integral transportation study. Transp Res 8B:11–27

Steenbrink PA (1974b) Optimization of transport networks. John Wiley & Sons, New York

Sun Y, Song R, He S, Chen Q (2009) Mixed transportation network design based on immune clone annealing algorithm. J Transp Syst Eng Inf Technol 9(3):103–108

Wang DZW, Lo HK (2010) Global optimum of the linearized network design problem with equilibrium flows. Transp Res 44B:482–492

Xiong Y, Schneider JB (1995) Transportation network design using a cumulative genetic algorithm and neural network. Transp Res Rec 1364:37–44

Xu T, Wei H, Hu G (2009) Study on continuous network design problem using simulated annealing and genetic algorithm. Expert Syst Appl 36:1322–1328

Yang H, Bell MGH (1998) Models and algorithms for road network design: a review and some new developments. Transport Rev 18:257–278

Zhang H, Gao Z (2009) Bilevel programming model and solution method for mixed transportation network design problem. J Syst Sci Complex 22:446–459

Zhang L, Lawphongpanich S, Yin Y (2009) An active-set algorithm for discrete network design problems. The 18th International Symposium on Transportation and Traffic Theory, Hong Kong, Springer. In: Lam, W. H. K., Wong, S. C., Lo H. K., 2009. Transportation and Traffic Theory 2009: Golden Jubilee, 283–300, Springer