Enumeration of Dominant Solutions An Application in Transport Network Design

Amirali Zarrinmehr[1] Yousef Shafahi[2]

Received: 23.08.2013 Accepted: 11.11.2013

Abstract

A One-Dimensional Binary Integer Programming Problem (1DB-IPP) is concerned with selecting a subset from a set of k items in budget constraint to optimize an objective function.In this problem a dominant solution is defined as a feasible selection to which no further item could be added in budget constraint. This paper presents a simple algorithm for Enumeration of Dominant Solutions (EDS) and investigates its functionality. The algorithm is then applied on the formulation of the Network Design Problem (NDP) with fixed travel-time links. The problem is a case study of 1DB-IPPs in the transportation planning literature which arises in the networks where the link travel-times are not sensitive to the amount of flow. The results are reported in detail for three illustrative examples and compared with the results of the Branch-and-Bound (B&B) algorithm. These examples suggest that in lower budget levels up to 40.2, 40.3 and 27.1 percentages the EDS algorithm outperforms the B&B algorithm. However, the overall performance of the B&B algorithm is notably faster in higher budget levels.

Keywords: Enumeration, dominant solution, branch-and-bound algorithm, network design problem

Corresponding author Email: amirali.zarrinmehr@modares.ac.ir

1- MSc. Graduate, Department of Civil Engineering, Sharif University of Technology, Tehran, Iran

2- Professor, Department of Civil Engineering, Sharif University of Technology, Tehran, Iran

1. Introduction

Combinatorial problems arise in various applications of planning, scheduling and computer aided design [Grama and Kumar, 1993]. Many of these problems are classified in the NP-Hard complexity class. To address the exact solution of such problems, an exhaustive search would require impractical running time in real instances. Therefore, implicit enumeration methods have been widely devised by the researchers to tackle these problems [Ibaraki, 1976; Rust, 2006].

There are instances in combinatorial problems where the optimal solution could be searched in a smaller subset of feasible solutions, i.e. dominant solutions, instead of the entire search space. As an example consider the subset-sum problem in which a subset of weights w_1, w_2, … , w_k must be chosen in a manner that the total sum of weights is maximized without exceeding a given capacity, C. The formulation is as follows [Pisinger, 1995]:

$$\text{Max.} \quad Z = \sum_{(i=1)}^{k} 1 w_i\, y_i \qquad (1)$$

$$\text{s.t. :} \quad \sum_{(i=1)}^{k} 1 w_i\, y_i \le C, \qquad (2)$$

$$y_i = 0 \text{ or } 1, \quad 1 \le i \le k, \qquad (3)$$

Assuming the weights in the subset-sum problem to be non-negative, it is clear that the selection of more items, if capacity-wise feasible, will not result in a worse solution. Therefore, solutions with more selected items clearly dominate those with fewer items. In other words, one could ignore solutions with fewer items and concentrate on the remaining search space.

For a more general discussion, consider a 1DB-IPP which is the problem of selection among k items in budget constraint in order to achieve the optimum for an objective function.This problem can be formulated like (4)-(6).

$$\text{Opt.} \quad F(Y) \qquad (4)$$

$$\text{s.t. :} \quad \sum_{(i=1)}^{k} 1 c_i\, y_i \le B \qquad (5)$$

$$y_i = 0 \text{ or } 1, \quad 1 \le i \le k, \qquad (6)$$

where

y_i is the decision variable taking the value 1 (0) if project i is (is not) selected for construction, $1 \le i \le k$,

Y is k-length vector of decision variables,

F(Y) is an objective function.over Y,

c_i is the selection cost of item i, $1 \le i \le k$,

B is the available budget.

It is clear that each feasible solution of a 1DB-IPP can be shown as a k-length binary string. Define in this problem a dominant solution as a feasible k-length binary string in which none of the '0' values could be changed to '1' in budget constraint. In other words, a dominant solution is a feasible selection of items where adding any more items makes it infeasible. This definition will be frequently used throughout this paper.

To tackle the exact solution of many 1DB-IPPs, one approach might be to confine the search to dominating feasible space, namely dominant solutions. This paper presents, in such 1DB-IPPs, an algorithm named EDS for enumerating all dominant solutions and investigates its functionality. As a case-study the algorithm is then applied on one of NDP formulations in transportation planning and the results are reported and compared with the B&B algorithm. The results experimentally suggest that the EDS algorithm could outperform the B&B algorithm in certain cases, whereas the overall performance of the B&B algorithm is notably faster.

The rest of this paper is structured as follows: In section 2 the EDS algorithm is introduced and its functionality is formally discussed via some lemmas. A NDP formulation, as a case for 1DB-IPPs is introduced in section 3 and detailed experimental results are reported. The paper is concluded and remarks for further research are added in section 4.

2. EDS Algorithm

Suppose that k items are to be selected each with a non-negative selection cost and a predefined budget is available for selection. In such circumstances, the EDS algorithm is going to enumerate dominant solutions, as previously defined. Before the formal presentation of the EDS algorithm, some prerequisite points must be highlighted.

• The EDS algorithm holds a k-length binary number as the current string which stands for a dominant solution. For each string, there will be a remaining budget which is defined as the available budget in the algorithm.

• The functionality of the EDS algorithm is based on the increasing order of items according to their costs. Items are therefore sorted initially in an increasing order from right to left.

• The EDS algorithm, at each iteration, would fall into one of the three states defined in the algorithm as current states:

• state 0: The current string is a budget-wise feasible, but non-dominant solution,

• state 1: The current string is a dominant solution,

• state 2: The current string is a budget-wise infeasible solution.

• To perform a proper operation the EDS algorithm would need to specify the current state. This is done simply by checking the current string:

• If the available budget ≥ 0 and the rightmost ‘0’ can be changed to ‘1’ in budget constraint, the state is 0,

• If the available budget ≥ 0 and the rightmost value is ‘1’, or is ‘0’ but cannot be changed to ‘1’ in budget constraint, the state is 1,

• If the available budget < 0, the state is 2.

Now the pseudo-code of the EDS algorithm can be written:

Begin;

// definitions and initialization:

- define the available budget and set it to B;
- define a string of binary values and set its values to ‘0’;
- sort items in an increasing order from right

to left according to their costs;
// iterations:
While iterations are not stopped
- specify the current state;
if the current state = 0
- while holding the available budget ≥ 0, change the rightmost ‘0’ to ‘1’ and update the available budget;
else if the current state = 1
- enumerate the current state as a dominant solution;
- change the rightmost sequence of ‘1’s to ‘0’, and update the available budget;
- if the next rightmost ‘0’ is existent, change it to ‘1’ and update the available budget; else stop the iterations;
else // current state = 2
- if the second rightmost ‘1’ is existent, change the two rightmost ‘1’s to ‘0’ and update the available budget; else stop the iterations;
- if the next rightmost ‘0’ is existent, change it to ‘1’ and update the available budget; else stop the iterations;
endwhile
End;

To observe the functionality of the EDS algorithm, a simple example is presented in Table 1. In this example, a total budget of 60 is assumed to be at hand and there are eight items for selection, with costs: 8, 16, 28, 29, 32, 40, 45 and 58. Table 1 shows the way the iterations are performed by the EDS algorithm. The sequence of states in this table suggests a functionality diagram for the algorithm which is shown in Figure 1. According to this diagram, the algorithm starts with the state 0 from which it necessarily moves to state 1. In either states of 1 or 2, the algorithm may move to any of the three states as well as the end of the iterations. This functionality is discussed further in the next subsections.

2.1 Notes about the EDS Algorithm

To formally discuss the functionality of the EDS algorithm, three preliminary lemmas are provided. These lemmas support the diagram shown in Figure 1 and also indicate that neither of dominant solutions is ignored by the algorithm. Note that in this subsection, wherever binary strings are compared, the corresponding binary numbers are considered.

Lemma 1: If the current state is 1, the EDS algorithm moves from the current string, S_1, to a new string, S_2, that $S_1<S_2$. If the algorithm does not stop, the new state can be 0, 1, or 2 (as shown in Figure 1) and no dominant solution like S that $S_1<S<S_2$ is ignored.

Proof: The current string could be shown like S_1=CBA (e.g. S_1=001011011100) in which:
- A is the sub-string of rightmost successive elements of ‘0’ (e.g. A=00),
- B is the sub-string of the rightmost successive elements of ‘1’ (e.g. B=111),
- C is the sub-string which immediately follows B (e.g. C=0010110).

Assume that C≠Ø. Then, according to the pseudo-code, all elements of ‘1’ in B change

Table 1. An example for the EDS algorithm functionality

Iteration number	Costs of projects: 58	45	40	32	29	28	16	8	Budget: 60	
	Current string								Available budget	Current state
1	0	0	0	0	0	0	0	0	60	0
2	0	0	0	0	0	1	1	1	8	**1**
3	0	0	0	0	1	0	0	0	31	0
4	0	0	0	0	1	0	1	1	7	**1**
5	0	0	0	0	1	1	0	0	3	**1**
6	0	0	0	1	0	0	0	0	28	0
7	0	0	0	1	0	0	1	1	4	**1**
8	0	0	0	1	0	1	0	0	0	**1**
9	0	0	0	1	1	0	0	0	-1	2
10	0	0	1	0	0	0	0	0	20	0
11	0	0	1	0	0	0	0	1	12	**1**
12	0	0	1	0	0	0	1	0	4	**1**
13	0	0	1	0	0	1	0	0	-8	2
14	0	1	0	0	0	0	0	0	15	0
15	0	1	0	0	0	0	0	1	7	**1**
16	0	1	0	0	0	0	1	0	-1	2
17	1	0	0	0	0	0	0	0	2	**1**

Stop

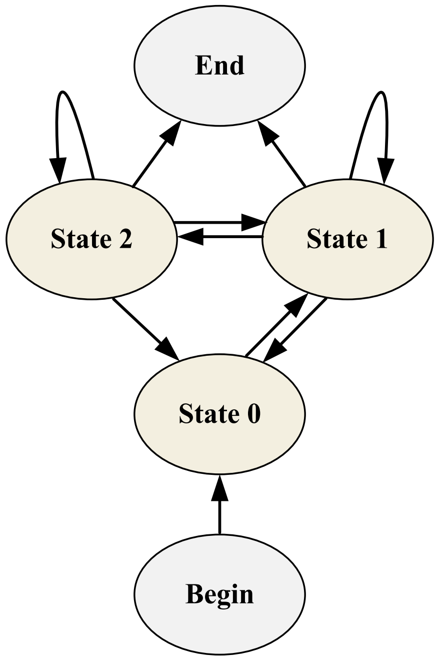


Figure 1. The Functionality Diagram of the EDS Algorithm

into '0' and the rightmost element of C, which is necessarily '0', changes into '1' (e.g. S_2=001011100000). The next state would be either '0', '1' or '2' with respect to the costs of the changed elements. The emerging string, S_2, clearly as a binary value has a greater value than S_1.

For any string like S that $S_1<S<S_2$ (e.g. S=001011011101), S differs with S_1 but only in some elements of A that are changed into '1'. The string S, as a result, cannot be a dominant solution, because this contradicts the initial assumption that S_1 is a dominant solution. Therefore there is no dominant solution like S that the EDS algorithm ignores in the transition from S_1 to S_2.

Also if C=Ø, the algorithm stops and a similar argument would prove that no dominant solution like S such that $S>S_1$ is ignored by the algorithm.

Lemma 2: If the current state is 2, the EDS algorithm moves from the current string, S_1, to a new string, S_2, that $S_1<S_2$. If the algorithm does not stop, the new state can be 0, 1, or 2 (as shown in Figure 1) and no dominant solution like S that $S_1<S<S_2$ is ignored.

Proof: According to the pseudo-code, if the current string, S_1, is comprised of only one element of '1', the algorithm stops and because the current string has been an infeasible solution, neither of greather strings would be feasible.

Assume that there exist at least two elements of '1' in the current string, (e.g. S_1=010011100010) and that B is the sub-string of successive elements of '1' starting from the second rightmost '1' (e.g. B=111). The current string, S_1, could now be written like CBA in which:

- A is the sub-string of rightmost successive elements before the second rightmost '1' (e.g. A=00010),
- C is the sub-string immediately following B at the left side (e.g. C=0100).

Suppose that C≠Ø. Then, according to the pseudo-code, all elements of B and A are set into '0' and the rightmost element of C, which is necessarily '0', is changed into '1' (e.g. S_2=010100000000). Similar to the argument for lemma 1, the new string, S_2, is greater than S_1 and the new state would be either 0, 1 or 2.

The sub-string A in S_1 can further be decomposed like A_21A_1 in which:

- A_1 is the sub-string of rightmost successive elements of '0' before the first '1' (e.g. A_1=0),
- 1 is an intermediate element of A, separating the two sub-strings of A_1 and A_2,
- A_2 is the sub-string of successive elements of '0' between the first and second rightmost '1's of S_1 (e.g. A_2=000).

For binary strings like S that $S_1<S<S_2$ (e.g. S=010011100011 or 010011101000), either one of two cases might happen:

(a) Some elements of A_1 are changed into '1' and all elements at the left side of A_1 are the same as S_1.

(b) Some elements of A_2 are changed into '1' and all elements at the left side of A_2 are the same as S_1.

In both cases of (a) and (b), the corresponding cost for S would be equal to or greater than S_1 which results in infeasibility of S (In case (b) this is true due to the sorting of items according to their costs, which was an initial assumption). Therefore, all strings between S_1 and S_2 become infeasible and the EDS algorithm does not ignore any dominant solution in this transition.

Also, if C=Ø, the algorithm stops according to the pseudo-code and a similar argument shows that no dominant solution like S that $S>S_1$ is ignored.

Lemma 3: If the current state is 0, the EDS algorithm moves from the current string, S_1, to a new string, S_2, that $S_1<S_2$. The new state will always be 1 (as shown in Figure 1) and no dominant solution like S that $S_1<S<S_2$ is ignored.

Proof: In order to prove this lemma, a sufficient condition would be the following auxiliary lemma and thereafter, the lemma can be easily proved using induction over states of 0.

Auxiliary lemma: If the current state is 0 and the previous state has been either 1 or 2, the EDS algorithm moves from the current string, S_1, to a new string, S_2, that $S_1<S_2$. The new state will always be 1 and no dominant solution like S that $S_1<S<S_2$ is ignored.

Proof of auxiliary lemma: Suppose that the previous string has been S_0 (e.g. S_0=001100111000) in either states of 1 or 2. This string can be shown like BA in which:

- A is the sub-string of rightmost successive elements including the rightmost elements of '1' and ending at the subsequent element of '0' (e.g. A=0111000),
- B is the sub-string immediately following A at the left side (e.g. B = 00110).

According to the instructions related to states 1 and 2 in the pseudo-code, after the transition from S_0 to S_1, all elements of A would become '0' except the leftmost element which changes into '1' (e.g. . S_1=001101000000). Then, while the budget constraint is not violated, the algorithm changes the rightmost elements of '0' into '1' (e.g. S_1=001101001111). This clearly can not result in changing all '0' elements of A into '1', because it contradicts the assumption that S_0 (with some '0' elements in A) is a dominant solution. Therefore, in the new string, S_2, the sub-string A will have a structure like $1A_2\,A_1$ in which:

- A_1 is the sub-string of rightmost successive elements of '1' (e.g. A_1=1111),
- A_2 is the sub-string of rightmost successive elements of '0' (e.g. A_2=00),

1 is the leftmost element of A.

The string S_2 is clearly greater than S_1. It is also a dominant solution, because the rightmost element of '0' in A_2 (and consequently neither of the following elements of '0' at the left side) can be changed into '1' while holding the feasibility constraint.

Moreover, for any string like S that $S_1<S<S_2$, some elements of A_2 should be '0' and therefore, S can not be a dominant solution, be-

cause this would contradict the previous argument that S_2 is a dominant solution. Therefore, the algorithm does not ignore any dominant solutions in its transition form S_1 to S_2.

Lemmas 1 to 3 indicate that the EDS algorithm traverses k-length binary numbers in an increasing order. These numbers are clearly finite. Hence, in order to prove that the EDS algorithm enumerates all dominant solutions, it is sufficient to prove the following corollary. The proof is trivial by induction.

Corollary: At each iteration, all dominant solutions with binary values smaller than the current string have been enumerated by the EDS algorithm.

2.2 A Rough Computational Analysis for the EDS Algorithm

According to the pseudo-code of the EDS algorithm, it is simple to see that:

- Operations related to current state 0, are performed at most in O(k).
- Operations related to current state 1, are performed at most in O(k).
- A sequence of operations related to current state 2, can be performed computationally in no more than O(k).

Therefore, regarding Figure 1, the generation of each dominant solution takes place computationally in at most C = O(3k). A more accurate analysis, however, may be possible for the EDS algorithm via amortized analysis [Cormen et al, 2001]; nonetheless this rough analysis leads to an interesting property for the algorithm: Given that N is the total number of dominant solutions in a 1DB-IPP feasible space, the EDS algorithm enumerates all dominant solutions in at most C*N computational cost, no matter what the size of the search space is.

3. Case-study on NDP

This section briefly reviews a definition of NDP as a 1DB-IPP in transportation planning: NDP with fixed travel-time links. The B&B algorithm as a traditional exact method to tackle the problem is first overviewed. Next, it is discussed how the EDS algorithm could be applied to this problem. The section finally compares the performances of the B&B algorithm and the method developed in this paper, namely the EDS algorithm.

3.1 ND with fixed travel-time links

Network design includes a variety of applicable problems in transportation planning, logistics, telecommunication, and production. In this problem, usually the aim is to choose arcs in a network to enable demand to flow between Origin-Destinations (ODs) at the lowest system cost [Alba, 2005]. The complexity of this problem stems from the combinatorial number of alternative decisions to be selected among. In transportation planning, network design is known as a classical problem with a bi-level structure which aims at choosing the best subset of proposed projects (i.e. new links) for construction in a given network, while considering users' behaviour in routing. an objective function. is usually minimizing the total travel time of the network users [Poorzahedy and Abulghasemi, 2005].

NDP falls into the category of NP-Hard problems [Garey and Johnson, 1979] for which there are no effective algorithms, yet, to tackle the exact solutions for large problem instances. As a result, researchers have devised various met-heuristics such as genetic algorithms [Yin, 2000], ant colony optimization [Poorzahedy and Abulghasemi, 2005; Vitins and Axhausen, 2009] and particle swarm [Babazadeh, Poorzahedy and Nikoosokhan, 2011; Angulo et al, 2013] to address the problem. Because the focus of this paper is not on meta-heuristic approaches, the interested reader is referred to references [Poorzahedy and Abulghasemi, 2005; Vitins and Axhausen, 2010] to find more about these approaches.

As a special simple formulation of NDPs [Garey and Johnson, 1979], NDP with fixed travel-time links, arises in networks in which the travel-times are not sensitive to the amount of flow between nodes. This is the case, for example, in networks with low levels of congestion or in freight transportation networks. The problem of NDP with fixed travel-time links, as a 1DB-IPP, can be formulated as (7)-(9). Henceforth, this problem is addressed simply as NDP.

$$\text{Min.} \quad f_T(Y) \tag{7}$$

$$\text{s.t.}: \quad \sum\nolimits^{n}_{(i=1)} c_i\, y_i \le B \tag{8}$$

$$y_i = 0 \text{ or } 1, \qquad 1 \le i \le k, \tag{9}$$

In the above formulation, $f_T(Y)$ is the total travel time in the network when the network is augmented by adopting decision vector of projects, Y, for construction and the demand is rooted from shortest-paths for all origin-destinations. Routing demand from the shortest-paths between all origin-destinations is also known as an All-or-Nothing Traffic Assignment (ANTA) procedure in the transportation literature [Sheffi, 1985].

To address the exact solution of a NDP, a general method is the well-known B&B algorithm. Ochoa-Rosso and Silva [15], in an early study, suggested a B&B tree in the form of a rooted binary tree in which at level i ($1 \le i \le k$) the decision variable y_i was decided over. To evaluate lower bounds on partial solutions, they assumed all undecided projects to be constructed and applied a traffic assignment for the corresponding network [Ochoa-Rosso and Silva, 1967].

3.2 The EDS algorithm as a Solution Method

In a NDP the optimal solution can be found among the dominant solutions. This is clearly true because addition of more projects to the network will augment the feasible space of the problem. As a result, the EDS algorithm can be simply applied to achieve the exact solution of the NDP. For this application, dominant solutions are iteratively enumerated and evaluated. The solution with the best evaluation, at last, would be the optimal solution of the problem.

3.3 Numerical Experiment

To experimentally study the performance of the presented algorithm in a 1DB-IPP, the EDS algorithm and the B&B algorithm (previously described in 3.1) were implemented in Java programming language. The Sioux-Falls transportation network [Bar-Gera, 2011] was considered with 20 proposed projects. Figure 2 shows the network configuration and proposed projects therein. Details about these projects are given in Table 2.

Both algorithms of EDS and B&B were run, on a 2.53 GHz Intel(R) Core(TM) i5 CPU, for three illustrative examples of NDP with 12, 16 and 20 projects (which are projects 1-12, 1-16 and 1-20 respectively as defined in Table 2). Given that the total selection cost for projects is C, different levels for B/C, namely different budget levels, were also taken into consideration. Running results are given in detail in Table 3 (a)-(c) and corresponding curves of the run-time versus the budget level are plotted in Figure 3 (a)-(c).

As the results of Table 3 and Figure 3 suggest, the B&B algorithm has an overall faster performance than the EDS algorithm. The difference between performances increases combinatorially, as the problem enlarges. However, in the related results of three examples, it is shown that the EDS algorithm could outperform the B&B algorithm in lower levels of budget and emerge as a competitive algorithm. Using a linear interpolation on the data presented in Table 3, this holds for B/C values of 0.402, 0.403, and 0.271 respectively in case of three examples in this paper.

4. Concluding Remarks

In a 1DB-IPP, k items are given each with a specific cost and a limited budget level is available to select items. It is intended to find, in budget constraint, a selection of items to achieve the optimum of an objective function.In this paper, dominant solution was defined as a feasible selection of items in 1DB-IPP to which no further items could be added in budget constraint. For enumeration of these dominant solutions, the paper introduced a simple enumeration algorithm namely the EDS algorithm. It was proved that the algorithm does not ignore enumeration of any dominant solution while each enumeration is done in at most O(3k).

As an application of the EDS algorithm, the paper considered the NDP with fixed travel-time links, where the optimal solution can be found among dominant solutions. Two programs corresponding to the B&B algorithm and the EDS algorithm were implemented in Java and detailed results were reported on the Sioux-Falls transportation network for three examples of NDP. These results suggest that

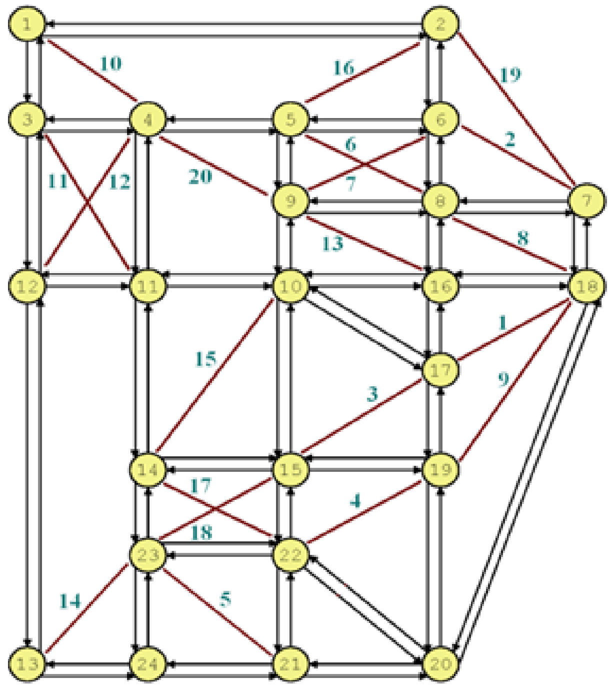


Figure 2. Sioux-Falls network and projects configuration

Table 2. Definitions of projects in the Sioux-Falls network

Project Number	From Node	To Node	Travel Time (minutes)	Related Cost
1	17	18	3.4	5.7
2	6	7	4.6	5.9
3	15	17	3.6	6
4	22	19	4	6.4
5	21	23	3.5	6.4
6	5	8	4.9	7.1
7	6	9	4.9	7.1
8	8	18	5	7.1
9	19	18	4.9	7.1
10	1	4	5.3	8
11	3	11	5.5	8.5
12	12	4	5.5	8.5
13	9	16	5.1	8.5
14	13	23	4.7	9
15	10	14	6.8	9.2
16	2	5	5.8	9.6
17	14	22	5.6	9.6
18	23	15	5.6	9.6
19	2	7	7.1	10.4
20	4	9	6.6	10.7

Table 3. Detailed results of running the EDS and B&B algorithms

(a) NDP with proposed projects 1-12

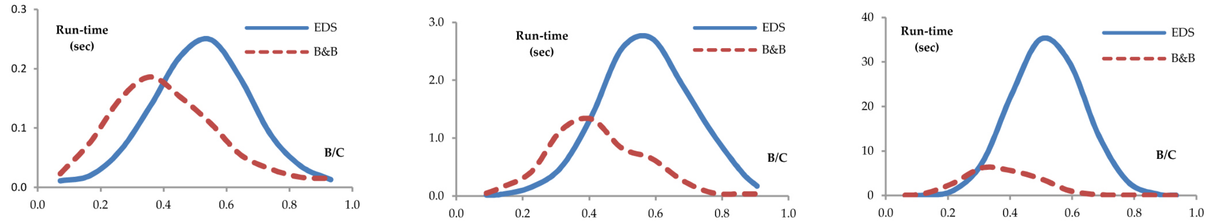
Budget Level		EDS		B&B		Optimum	
B	B/C	ANTAs	Run-time (sec)	ANTAs	Run-time (sec)	Solution	Evaluation
6	0.072	4	0.011	39	0.023	[001000000000]	3157500
14	0.17	36	0.02	223	0.076	[000100100000]	3145830
22	0.26	178	0.062	454	0.149	[000100100010]	3130780
30	0.36	458	0.137	597	0.186	[001100000011]	3119340
38	0.46	754	0.221	479	0.152	[001100100011]	3108770
46	0.55	852	0.249	319	0.108	[001101100011]	3099970
54	0.64	618	0.183	146	0.055	[001101100111]	3094560
62	0.74	283	0.089	67	0.031	[101101100111]	3089280
70	0.84	86	0.036	21	0.017	[111111100111]	3086350
78	0.93	12	0.013	16	0.015	[111111101111]	3085890

(b) NDP with proposed projects 1-16

Budget Level		EDS		B&B		Optimum	
B	B/C	ANTAs	Run-time (sec)	ANTAs	Run-time (sec)	Solution	Evaluation
10	0.09	17	0.016	119	0.046	[0001000000000000]	3156400
15	0.14	62	0.029	421	0.149	[0001000000100000]	3141350
25	0.23	490	0.154	1361	0.419	[0001000000100010]	3127470
35	0.32	1744	0.504	3934	1.142	[0001000000110010]	3114420
45	0.41	4927	1.381	4568	1.327	[0011001000111000]	3098910
55	0.50	9193	2.541	2817	0.828	[0011001000111010]	3084770
65	0.59	9758	2.724	2172	0.647	[0011011000111010]	3078040
75	0.68	6922	1.930	885	0.274	[1011111000111010]	3070510
85	0.77	3907	1.099	146	0.055	[1011011001111110]	3062420
95	0.86	1377	0.407	82	0.035	[1011111001111110]	3060170
100	0.90	543	0.171	69	0.034	[1011111001111111]	3058700

(c) NDP with proposed projects 1-20

Budget Level		EDS		B&B		Optimum	
B	B/C	ANTAs	Run-time (sec)	ANTAs	Run-time (sec)	Solution	Evaluation
10	0.06	19	0.018	169	0.061	[00010000000000000000]	3156400
20	0.12	207	0.074	1345	0.419	[00110010000000000000]	3136870
35	0.22	4022	1.133	9625	2.784	[00010000001100100000]	3114420
50	0.31	27110	7.503	21790	6.233	[00010010001110100000]	3093730
65	0.41	82745	22.705	19436	5.538	[00110010001110101000]	3075100
80	0.50	128685	35.122	12527	3.745	[10110010001110101100]	3062760
95	0.59	107838	29.725	3616	1.058	[10110110011110101100]	3050620
110	0.69	48446	13.423	548	0.177	[10111110011111101100]	3041250
125	0.78	11030	3.073	277	0.094	[10111110011111101110]	3038200
140	0.87	1111	0.335	117	0.048	[11111110011111101111]	3035700
150	0.94	156	0.055	54	0.028	[11111110011111111111]	3034230



(a) NDP with proposed projects 1-12 (b) NDP with proposed projects 1-16 (c) NDP with proposed projects 1-20

Firure 3. Runtime curves for the EDS and B & B algorithms

while the overall performance of the B&B algorithm is notably faster, the EDS algorithm outperforms the B&B algorithm for lower levels of budget.

This paper investigated the performance of the EDS algorithm on one 1DB-IPP in transportation planning via an experimental approach. More experiments on other 1DB-IPPs as well as theoretical study of the performance of the algorithm would further extend the results of this paper. The comparison of the proposed algorithm with other methods in different conditions can also be interesting for further studies.

5. References

- Alba, E. (2005) “Parallel metaheuristics: A new class of algorithms”, Wiley Series on Parallel and Distributed Computing.

- Angulo, E., Castillo, E., García-Ródenas, R. and Sánchez-Vizcaíno, J. (2013) “A continuous bi-level model for the expansion of highway networks”, Computers and Operations Research, in press.

- Babazadeh, A., Poorzahedy, H. and Nikoosokhan, S. (2011) “Application of particle swarm optimization to transportation network design problem”, Journal of King Saud University-Science, Vol. 23, No. 3, pp. 293-300.

- Bar-Gera, H. (2011) “Transportation network test problems”, Accessed September 19, http://www.bgu.ac.il/~bargera/tntp/.

- Cormen, T. H., Leiserson, C.E., Rivest, R. L. and Stein, C. (2011) “Introduction to algorithms”, Second edition. Cambridge, Massachusetts: MIT press.

- Garey, M. R. and Johnson, D. S. (1979) “Computers and intractability: a guide to the theory of NP-completeness”, W. H. Freeman and Company, San Francisco.

- Grama, A.Y. and Kumar, V. (1993) “A survey of parallel search algorithms for discrete optimization problems”, ORSA Journal on Computing, Vol. 7, pp. 1-40.

- Ibaraki, T. (1976) “Theoretical comparisons of search strategies in branch-and-bound algorithms”, International Journal of Computer & Information Sciences, Vol. 5, No. 4, pp. 315-44.

- Ochoa-Rosso, F. and Silva, A. (1969) "Optimum project addition in urban transportation networks via descriptive traffic assignment models", Transportation Systems Division, MIT.

- Pisinger, D. (1995) "Algorithms for knapsack problems", Ph.D diss., Department of Computer Science, University of Copenhagen.

- Poorzahedy, H. and Abulghasemi, F. (2005) "Application of ant system to network design problem", Transport, Vol. 32, No. 3, pp. 251-273.

- Rust, J. (2006) "Dynamic programming", New Palgrave Dictionary of Economics.

- Sheffi, Y. (1985) "Urban transportation networks: Equilibrium analysis with mathematical programming methods", Prentice-Hall Englewood Cliffs, NJ.

- Vitins, B. J., and Axhausen, K. W. (2009) "Optimization of large transport networks using the ant colony heuristic", Computer-Aided Civil and Infrastructural Engineering Journal of ASCE, Vol. 24, No. 1, pp. 1-14.

- Vitins, B. J. and Axhausen, K. W. (2010) "Patterns and grammars for transport network generation", Proceedings of 14 the Swiss Transport Research Conference.

- Yin, Y. (2000) "Genetic-algorithms-based approach for bi-level programming models." , Transportation Engineering Journal of ASCE, Vol. 126, No. 2, pp. 115-120.