Journal of King Saud University – Science (2011) 23, 293–300

King Saud University
Journal of King Saud University – Science
www.ksu.edu.sa
www.sciencedirect.com

ORIGINAL ARTICLE

Application of particle swarm optimization to transportation network design problem

Abbas Babazadeh [a,*], Hossain Poorzahedy [b], Saeid Nikoosokhan [a]

[a] School of Civil Engineering, University of Tehran, P.O. Box 11155-4563, Tehran, Iran
[b] Institute for Transportation Studies and Research, Sharif University of Technology, P.O. Box 11365-9313, Tehran, Iran

Available online 5 March 2011

KEYWORDS
Transportation;
Network design;
Optimization;
Meta-heuristics;
Particle swarm;
Ant colony

Abstract Transportation network design problem (TNDP) aims to choose from among a set of alternatives (e.g., set of new arcs) which minimizes an objective (e.g., total travel time), while keeping consumption of resources (e.g., budget) within their limits. TNDP is formulated as a bilevel programming problem, which is difficult to solve on account of its combinatorial nature. Following a recent, heuristic by ant colony optimization (ACO), a hybridized ACO (HACO) has been devised and tested on the network of Sioux Falls, showing that the hybrid is more effective to solve the problem. In this paper, employing the heuristic of particle swarm optimization (PSO), an algorithm is designed to solve the TNDP. Application of the algorithm on the Sioux Falls test network shows that the performance of PSO algorithm is comparable with HACO.

1. Introduction

In transportation planning and development, transportation network design problem (TNDP) is an important subject in which certain objective(s) is(are) minimized through choosing among a given set of projects under resource constraints. Objectives are (usually) related to user benefits (or costs), and constraints are related to various resources which bring about such benefits at the cost of the operator of the network. However, solving such a problem requires too much time. For an n-project case, considering an accept/reject decision for each project, there are 2^n alternative networks which are to be compared. Assuming that half of the networks may be rejected on accounts of resource constraints, and considering 1 min cpu time for evaluating the value of the objective function for each alternative network, the computation time would be 2^{n-1} minutes. For $n = 20$, for instance, one needs over 1 year computation time to reach the optimum network.

Various approaches have been taken to solve TNDP. Steenbrink (1974a), Wong (1984), and Magnanti and Wong (1984) surveyed some earlier algorithms of solving this problem. A branch and bound algorithm was presented by LeBlanc (1975) to solve TNDP. Since this algorithm does not perform well in large scale problems, the need for trade-off between the solution accuracy and speed of attaining it was felt early in the development of such solution algorithms.

* Corresponding author. Tel.: +98 21 61112176; fax: +98 21 66403808.
E-mail address: ababazadeh@ut.ac.ir (A. Babazadeh).

Peer review under responsibility of King Saud University.
doi:10.1016/j.jksus.2011.03.001

Production and hosting by Elsevier

There have been several methods to trade-off accuracy with speed. These are as follows: (a) using system equilibrium flows instead of user equilibrium (UE) ones (Sheffi, 1985) in the network loading (Steenbrink, 1974b; Dantzig et al., 1979; Chen and Sul Alfa, 1991); (b) assuming constant link cost functions (Boyce et al., 1973; Holmberg and Hellstrand, 1998); (c) relaxation of the integer constraints on decision variables (Steenbrink, 1974b; Abdulaal and LeBlanc, 1979; Dantzig et al., 1979); (d) decomposition of the problem (Steenbrink, 1974b; Dantzig et al., 1979; Hoang, 1982; Solanki et al., 1998); (e) aggregation of the network by link and node abstraction or extraction (Haghani and Daskin, 1983); (f) using an intrinsic approach by defining a surrogate problem which lacks the complexity of the original one (Yang and Bell, 1998); (g) heuristic procedures (Poorzahedy and Turnquist, 1982; Chen and Sul Alfa, 1991); (h) meta-heuristic (evolutionary) procedures, such as genetic algorithm (GA) (Yin, 2000), simulated annealing (SA) (Lee and Yang, 1994), GA, SA, and Tabu search (TS) (Cantarella et al., 2002), ant colony optimization (ACO) (Poorzahedy and Abulghasemi, 2005); (i) hybrid meta-heuristics, such as hybridized ACO (HACO) with GA, SA, and TS (Poorzahedy and Rouhani, 2007).

In this paper, an application of a modern evolutionary method, namely particle swarm optimization (PSO), to solve the TNDP is presented. The results are compared with those of the ACO and HACO existing on the same problem network in the last reference. The reminder of the paper is organized as follows. The next section is devoted to define the TNDP mathematically. In the two subsequent sections, the PSO is described in detail, and then adapted to the TNDP. Computational results are reported in the final section.

2. The TNDP

Let $G = (V, A)$ be a graph representing a transportation network with node set V and arc set A, and define $P \subseteq \{(r, s) \in V \times V : r \neq s\}$ as the set of origin–destination (OD) pairs. Each arc corresponds to a pair (i, j) of the nodes, where i is the tail and j is the head node of the arc. For each OD pair $(r, s) \in P$, there is a nonnegative flow rate (travel demand) from r to s, denoted by d_{rs}. In order to simplify the presentation, suppose that G is strongly connected, that is each node j can be reached from every other node i by following a directed path in G, and let K_{rs} be the non-empty set of paths from the origin r to the destination s.

Define $\overline{A}$ $(\overline{A} \neq A)$ as the set of project arcs, and let the decision vector be $y = (y_a)_{a \in \overline{A}}$ with y_a being the binary project decision variable, taking values 0 or 1 depending on rejection or acceptance of any project $a \in \overline{A}$. For a given vector y, define the decision network $G(y) = (V, A(y))$ with $A(y) = A \cup \{a \in \overline{A} : y_a = 1\}$ as the set of arcs followed by decision y, and for each $(r, s) \in P$ denote by $K_{rs}(y)$ the set of paths joining r to s in $G(y)$. For each path $k \in K_{rs}(y)$, let f_k be the flow of path k from origin r to destination s. Moreover, let δ_{ak} equals 1 if arc $a \in A(y)$ lies on path k, and 0 otherwise.

Assume further that each arc $a \in A \cup \overline{A}$ has a nondecreasing and continuously differentiable travel time function $t_a(x_a)$: $[0, \infty) \to [0, \infty)$ with x_a being the flow rate assigned to arc a. Then, letting c_a be the construction cost of project arc $a \in \overline{A}$, and considering the total construction cost being limited to the level of budget B, the TNDP can be illustrated with the upper level problem, ULP:

$$
\begin{aligned}
[\mathrm{ULP}] \quad \min_{y} \quad & T(y) = \sum_{a \in A(y)} x_a t_a(x_a) \\
\text{s.t.} \quad & \sum_{a \in \overline{A}} c_a y_a \leqslant B \\
& y_a = 0 \text{ or } 1 \quad \forall a \in \overline{A} \\
& x(y) \text{ is a solution of } [\mathrm{LLP}(y)]
\end{aligned}
$$

where $x(y) = (x_a)_{a \in A(y)}$ is the user equilibrium flow in the decision network $G(y)$, given as the solution of the lower level (traffic assignment) problem, LLP(y), for given y:

$$
\begin{aligned}
[\mathrm{LLP}(y)] \quad \min \quad & \sum_{a \in A(y)} \int_{\circ}^{x_a} t_a(w) dw \\
\text{s.t.} \quad & \sum_{k \in K_{rs}(y)} f_k = d_{rs} \quad \forall (r, s) \in P \\
& f_k \geqslant 0 \quad \forall k \in K_{rs}(y), \ \forall (r, s) \in P \\
& x_a = \sum_{(r,s) \in P} \sum_{k \in K_{rs}(y)} f_k \delta_{ak} \quad \forall a \in A(y)
\end{aligned}
$$

This is a well-known bilevel programming problem, where the [ULP] seeks a decision vector y for minimizing the total travel time $T(y)$ of the (assigned) traveler, and the [LLP(y)] is the traffic assignment model which estimates the traveler flows, given the decision y.

3. The PSO

Particle swarm optimization, also called PSO, is a population based stochastic optimization technique developed by Kennedy and Eberhart (1995) and Eberhart and Kennedy (1995). PSO mimics the behaviour of flocks of birds, swarms of insects or schools of fish, in which individuals are called particles and the population is called a swarm. In a problem space, each particle is given a position and a velocity. Once a particle finds a good direction to food, other particles are notified and will be able to speed toward that, immediately. The particles roam in the space, convey good positions to each other, and adjust their own positions and velocities based on these good positions (Abraham et al., 2006).

PSO is analogous to evolutionary algorithms, like GA, in a sense that it starts with randomly generated solutions, and evolves the solution until a desirable one is found. However, unlike GA, the evolutionary process in PSO only evolves the positions of the particles, rather than creating new particles (Shi and Eberhart, 1998a). The main strength of PSO is its fast convergence, which compares favourably with many meta-heuristics like GA and SA (Abraham et al., 2006). Moreover, it may be easily implemented, and requires few parameter settings and computational memory (You, 2008).

PSO has been successfully applied to many areas. Voss and Feng (2002), Jiang et al. (2007), Yisu et al. (2008) and You (2008) report some of this applications. This paper describes an application of such method in solving the TNDP.

3.1. The canonical PSO

The canonical PSO is initialized with a group of random candidate solutions as a swarm of particles. Each particle searches iteratively the new solutions by moving through the problem space with a velocity adjusted according to both the previous best solutions of itself and of the swarm. The best solution that has been monitored by the current particle is typically denoted by *local best*, while the best solution that has been discovered

by the group is denoted by *global best*. The global best conceptually connects all particles together, that is, each particle is influenced by the best solution in the entire population; the local best is used to take into account the ability of each particle to remember its past personal successes (Voss and Feng, 2002).

Consider a positive integer D as the dimension of the problem space. The position of the ith particle is represented by $p_i = (p_{ij})_{j=1,\ldots,D}$, where p_{ij} is the jth dimensional value for the ith particle. Also, the rate of the position change (velocity) is represented as $v_i = (v_{ij})_{j=1,\ldots,D}$ with v_{ij} being the velocity of ith particle along the dimension j. Consider $f(x_i)$: $R^D \to R$ as the fitness function which measures the quality for the position of the particle i. Each particle remembers its own best position so far achieved (the position that gives the best fitness function) as $p_i^* = (p_{ij}^*)_{j=1,\ldots,D}$, and the best position so far recorded by the population represented as $p_g^* = (p_{gj}^*)_{j=1,\ldots,D}$.

During the iteration time t, the velocity for the jth dimension of each particle i is updated by (Abraham et al., 2006):

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_1 (p_{ij}^*(t) - p_{ij}(t)) + c_2 r_2 (p_{gj}^*(t) - p_{ij}(t)), \quad (1)$$

where w is called as the *inertia weight* that was first employed in the range of 0.9–1.2 (Shi and Eberhart, 1998a), and c_1 and c_2 are constant values that were originally set to 2 (Eberhart and Kennedy, 1995). These two constants are multiplied by the random numbers r_1 and r_2, respectively, which are used to maintain the diversity of the population, and are uniformly distributed in the interval [0, 1] (Abraham et al., 2006). From a social point of view, as it is described by Voss and Feng (2002), each particle moves based on its current direction (v_i), its memory of where it found its personal best (p_i^*), and a desire to be like the best particle in the population (p_g^*). The new position of the ith particle is, then, updated by the sum of the previous position and the new velocity as (Abraham et al., 2006)

$$p_{ij}(t+1) = p_{ij}(t) + v_{ij}(t+1). \quad (2)$$

In the PSO, each particle i searches the solution p_i in the problem space with a range $[0, p_{\max}]$ (any other range can be translated to this range). In order to guide the particles effectively in the search space, the maximum moving distance during any iteration must be clamped in between the maximum range $[-v_{\max}, v_{\max}]$ with $0 < v_{\max} \leqslant p_{\max}$ (Abraham et al., 2006).

The inertia weight w in Eq. (1) affects the convergence speed of the PSO through controlling the impact of the history of velocities on the current velocity (Abraham et al., 2006). The role of this parameter is providing a balance between the global and local search abilities of PSO; in the sense that a larger value facilitates global exploration, while a smaller one tends toward local exploration (Shi and Eberhart, 1998b). Eberhart and Shi (2000) indicated that, initially setting the inertia weight to a large value and linearly decreasing it with time has a better performance than using a fixed value. Abraham et al. (2006) suggested an initial value around 1.2 with gradually reducing towards 0 as a good choice for w.

The parameters c_1 and c_2 are less critical for convergence of the PSO (Abraham et al., 2006); instead, they affect how much the movement of each particle would be influenced by its personal best and by the global best, respectively. Usually, $c_1 = c_2 = 2$ are used as default values (Abraham et al., 2006), while the work by Clerc and Kennedy (2002) shows that using a larger parameter c_1 than a parameter c_2, but with $c_1 + c_2 \leqslant 4$, might have better performance.

Denoting the size of the particle swarm by n, the pseudo code of the PSO algorithm is illustrated as follows:

- Select the size n, and the other parameters. Set $t = 0$.
- Initialize the positions $p_i(0)$ in $[0, p_{\max}]$ and the velocities $v_i(0)$ in $[-v_{\max}, v_{\max}]$ for all the particles, randomly.
- Set $p_i^*(0) = p_i(0)$ for $i = 1$ to n, and $p_g^*(0) = \arg\min(f(p_1(0)), \ldots, f(p_n(0)))$.
- While (the end criterion is not met) do
- For $i = 1$ to n
- For $j = 1$ to D
- Update $v_{ij}(t+1)$ and $p_{ij}(t+1)$ according to Eqs. (1) and (2);
- Next j
- Set $p_i^*(t+1) = \arg\min f(p_i^*(t)), f(p_i(t+1)))$;
- Next i
- Set $p_g^*(t+1) = \arg\min(f(p_g^*(t)), f(p_1(t+1)), \ldots, f(p_n(t+1)))$;
- $t \leftarrow t + 1$;
- End While.

The end criterion is usually one of the following (Abraham et al., 2006):

- *Maximum number of iterations:* the algorithm is terminated after a fixed number of iterations, for example, 100 iterations.
- *Number of iterations without improvement:* the optimization process is terminated after some fixed number of iterations, say 30, without any change of p_g^*.
- *Minimum objective function difference:* the difference between the last obtained objective function and the best fitness value is less than a prefixed threshold. For example, selecting a threshold of 1e−25, the algorithm will be terminated at iteration t when $f(p_g^*(t-1)) - f(p_g^*(t)) < 1e-25$.

4. Adapting the PSO to the TNDP

Employing the PSO for solving TNDP needs some modifications to the algorithm given in the previous section. First, the PSO is basically developed for continuous optimization problems. This is while the TNDP is formulated as a combinatorial optimization problem in terms of variables y denoted as $|\overline{A}|$-bit binary strings. To adapt the algorithm for this combinatorial nature, one may provide some mapping from the one-dimensional real-valued space to the $|\overline{A}|$-dimensional binary space. This is done here by transforming each real number p_i to its nearest integer in $[0, 2^{|\overline{A}|} - 1]$, and then transforming the resulting integer into the base-2 number system as an $|\overline{A}|$-bit binary code. To facilitate the presentation, the latter transformation is illustrated by the function $y(p_i) : [0, 2^{|\overline{A}|} - 1] \subset Z \to \{0,1\}^{|\overline{A}|}$.

The canonical PSO must also be adapted for the budget constraint embedded in the [ULP]. In this regard, one may apply a very simple modification that is assigning an adequately large fitness value (say M) to any infeasible solution p_i, i.e. $f(p_i) = M$.

The following is a formal statement of the proposed PSO algorithm:

Step 1. Initialization.
Select the particle swarm size n, the parameters c_1 and c_2, the initial and final values of the inertia weight w, and the maximum velocity $v_{\max}$.
For $i = 1$ to n do: initialize (randomly or partially randomly) the decision variable p_i in $[0, 2^{|\overline{A}|} - 1]$ so that $\sum_{a\in\overline{A}} c_a y_a(p_i) \leqslant B$; set $p_i^* = p_i$; initialize v_i randomly in $[-v_{\max}, v_{\max}]$.
Set $p_g^* = \arg\min(f(p_1), \ldots, f(p_n))$. Set the iteration counter $t = 0$.

Step 2. Updating each particle's position and velocity.
For $i = 1$ to n do: generate random numbers r_1 and r_2 in [0, 1]; update $v_i \leftarrow wv_i + c_1 r_1(p_i^* - p_i) + c_2 r_2(p_g^* - p_i)$; clamp in v_i between the range $[-v_{\max}, v_{\max}]$ as $v_i = \text{sign}(v_i)\min(|v_i|, v_{\max})$; update $p_i \leftarrow p_i + v_i$; clamp p_i to the range $[0, 2^{|\overline{A}|} - 1]$.

Step 3. Calculating each particle's fitness value.
For $i = 1$ to n do: set $y = y(p_i)$; if $\sum_{a\in\overline{A}} c_a y_a > B$ then set $f(p_i) = M$; else, solve the user equilibrium problem [LLP(y)] to compute $T(y)$, and set $f(p_i) = T(y)$.

Step 4. Updating local bests and global best.
For $i = 1$ to n do: update $p_i^* \leftarrow \arg\min(f(p_i^*), f(p_i))$.
Update $p_g^* \leftarrow \arg\min(f(p_g^*), f(p_1), \ldots, f(p_n))$.

Step 5. End criterion.
Set $t = t + 1$. If end criterion is not met, go to Step 2. Otherwise, $y = y(p_g^*)$ is the best solution found so far with the objective function value $T(y) = f(p_g^*)$. Collect the necessary information, and stop. □

5. Numerical example

In order to demonstrate the capability of the PSO algorithm in solving the TNDP, it will be applied on the network of Sioux Falls. This network has 24 nodes and 76 arcs, as shown in Fig. 1. The parameters of the travel time function $t_a(x_a) = \alpha_a + \beta_a x_a^4$ for each arc $a = (i, j)$, and the OD (origin/destination) demands are basically those given in Poorzahedy and Turnquist (1982), and LeBlanc (1975), and are given in Tables 1 and 2, respectively.

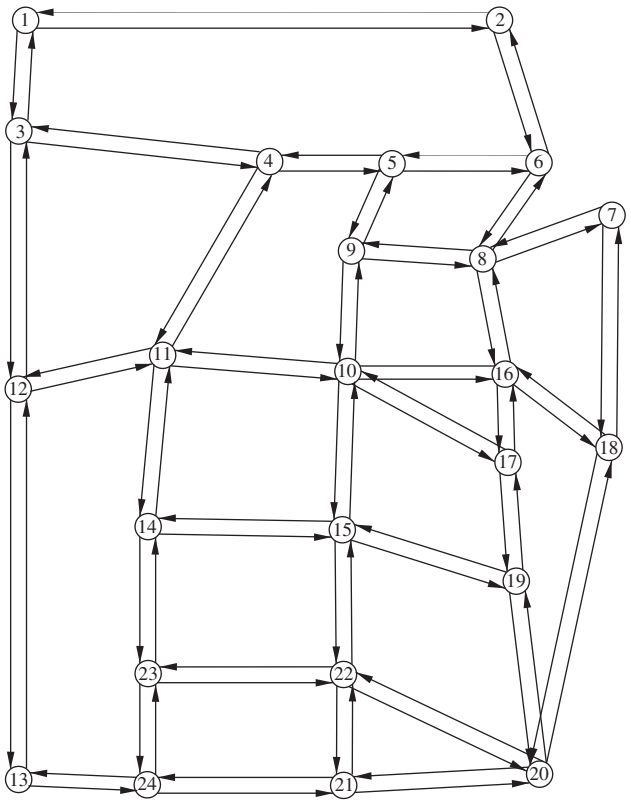

Figure 1 The Sioux Falls network.

There are 10 pairs of project arcs ($|\overline{A}| = 10$), of which 5 projects are improvement on existing arcs, and 5 are new arcs. The parameters of the travel time functions and the construction costs (in units of money) of the projects 1–10 are given in Table 3 (Poorzahedy and Abulghasemi, 2005). Considering 10

Table 1 Parameters of the travel time functions for the network in Fig. 1 (α is given in hours, and β in hours per thousand vehicles).

Arcs	Parameters		Arcs	Parameters	
	α	β		α	β
(1, 2), (2, 1)	0.06	0.00000002	(11, 12), (12, 11)	0.06	0.00001550
(1, 3), (3, 1)	0.04	0.00000002	(11, 14), (14, 11)	0.04	0.00001061
(2, 6), (6, 2)	0.05	0.00001241	(12, 13), (13, 12)	0.03	0.00000001
(3, 4), (4, 3)	0.04	0.00000007	(13, 24), (24, 13)	0.04	0.00000893
(3, 12), (12, 3)	0.04	0.00000002	(14, 15), (15, 14)	0.05	0.00001085
(4, 5), (5, 4)	0.02	0.00000003	(14, 23), (23, 14)	0.04	0.00001020
(4, 11), (11, 4)	0.06	0.00001550	(15, 19), (19, 15)	0.03	0.00000010
(5, 6), (6, 5)	0.04	0.00001001	(15, 22), (22, 15)	0.03	0.00000053
(5, 9), (9, 5)	0.05	0.00000075	(16, 17), (17, 16)	0.02	0.00000401
(6, 8), (8, 6)	0.02	0.00000521	(16, 18), (18, 16)	0.03	0.00000003
(7, 8), (8, 7)	0.03	0.00000119	(17, 19), (19, 17)	0.02	0.00000554
(7, 18), (18, 7)	0.02	0.00000001	(18, 20), (20, 18)	0.04	0.00000002
(8, 9), (9, 8)	0.10	0.00002306	(19, 20), (20, 19)	0.04	0.00000958
(8, 16), (16, 8)	0.05	0.00001157	(20, 21), (21, 20)	0.06	0.00001373
(9, 10), (10, 9)	0.03	0.00000012	(20, 22), (22, 20)	0.05	0.00001130
(10, 11), (11, 10)	0.05	0.00000075	(21, 22), (22, 21)	0.02	0.00000401
(10, 15), (15, 10)	0.06	0.00000027	(21, 24), (24, 21)	0.03	0.00000790
(10, 16), (16, 10)	0.04	0.00001080	(22, 23), (23, 22)	0.04	0.00000960
(10, 17), (17, 10)	0.08	0.00001930	(23, 24), (24, 23)	0.02	0.00000451

Table 2 Matrix of demands between OD pairs (in thousand vehicles per hour).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0	1	1	5	2	3	5	8	5	13	5	2	5	3	5	5	4	1	3	3	1	4	3	1
2	1	0	1	2	1	4	2	4	2	6	2	1	3	1	1	4	2	0	1	1	0	1	0	0
3	1	1	0	2	1	3	1	2	1	3	3	2	1	1	1	2	1	0	0	0	0	1	1	0
4	5	2	2	0	5	4	4	7	7	12	14	6	6	5	5	8	5	1	2	3	2	4	5	2
5	2	1	1	5	0	2	2	5	8	10	5	2	2	1	2	5	2	0	1	1	1	2	1	0
6	3	4	3	4	2	0	4	8	4	8	4	2	2	1	2	9	5	1	2	3	1	2	1	1
7	5	2	1	4	2	4	0	10	6	19	5	7	4	2	5	14	10	2	4	5	2	5	2	1
8	8	4	2	7	5	8	10	0	8	16	8	6	6	4	6	22	14	3	7	9	4	5	3	2
9	5	2	1	7	8	4	6	8	0	28	14	6	6	6	9	14	9	2	4	6	3	7	5	2
10	13	6	3	12	10	8	19	16	28	0	40	20	19	21	40	44	39	7	18	25	12	26	18	8
11	5	2	3	15	5	4	5	8	14	39	0	14	10	16	14	14	10	1	4	6	4	11	13	6
12	2	1	2	6	2	2	7	6	6	20	14	0	13	7	7	7	6	2	3	4	3	7	7	5
13	5	3	1	6	2	2	4	6	6	19	10	13	0	6	7	6	5	1	3	6	6	13	8	8
14	3	1	1	5	1	1	2	4	6	21	16	7	6	0	13	7	7	1	3	5	4	12	11	4
15	5	1	1	5	2	2	5	6	10	40	14	7	7	13	0	12	15	2	8	11	8	26	10	4
16	5	4	2	8	5	9	14	22	14	44	14	7	6	7	12	0	28	5	13	16	6	12	5	3
17	4	2	1	5	2	5	10	14	9	39	10	6	5	7	15	28	0	6	17	17	6	17	6	3
18	1	0	0	1	0	1	2	3	2	7	2	2	1	1	2	5	6	0	3	4	1	3	1	0
19	3	1	0	2	1	2	4	7	4	18	4	3	3	3	8	13	17	3	0	12	4	12	3	1
20	3	1	0	3	1	3	5	9	6	25	6	5	6	5	11	16	17	4	12	0	12	24	7	4
21	1	0	0	2	1	1	2	4	3	12	4	3	6	4	8	6	6	1	4	12	0	18	7	5
22	4	1	1	4	2	2	5	5	7	26	11	7	13	12	26	12	17	3	12	24	18	0	21	11
23	3	0	1	5	1	1	2	3	5	18	13	7	8	11	10	5	6	1	3	7	7	21	0	7
24	1	0	0	2	0	1	1	2	2	8	6	5	7	4	4	3	3	0	1	4	5	11	7	0

Table 3 Parameters of the travel time functions and the construction costs of the project arcs.

Project	Arcs	Parameters		Construction cost
		α	β	
1	(9, 10), (10, 9)	0.02	0.00000037	625
2	(6, 8), (8, 6)	0.01	0.00000156	650
3	(13, 24), (24, 13)	0.02	0.00000268	850
4	(7, 8), (8, 7)	0.01	0.00000035	1000
5	(10, 16), (16, 10)	0.03	0.00000324	1200
6	(7, 16), (16, 7)	0.03	0.00000032	1500
7	(19, 22), (22, 19)	0.01	0.00000004	1650
8	(11, 15), (15, 11)	0.01	0.00000041	1800
9	(9, 11), (11, 9)	0.02	0.00000003	1950
10	(13, 14), (14, 13)	0.01	0.00000016	2100

Table 4 Particle swarm parameter settings.

Parameter	Setting
Population size	10
Number of iterations	8
Initial w	1.2
Final w	0.4
c_1	2
c_2	2
γ_{max}	512

projects, there are 2^{10} (= 1024) alternative networks. A complete enumeration was used to compute the optimal solution of the TNDP for any given budget level for checking purposes (Poorzahedy and Abulghasemi, 2005; Poorzahedy and Rouhani, 2007). The PSO network design algorithm was implemented in a program in MATLAB on a Laptop with Intel Core 2 due 2 GHz processor. In all experiments, the PSO parameters are set as shown in Table 4.

5.1. Application of PSO algorithm

First the performance of the PSO algorithm will be discussed. Table 5 shows the results of solving the TNDP for the test network under various budget levels, as measured by budget to total construction cost (of the 10 projects), denoted as B/C. This ratio shows the level of limitation of the budget in the design problem, a determinant of the level of efforts needed to solve the problem. Since the PSO is of stochastic nature, each case has been solved 50 times, as in the case of Poorzahedy and Rouhani (2007), and the average number of traffic assignment problems solved (NTAPS) has been reported as the cost of problem solving, which is (almost) directly proportional to the CPU time of the computer. The performance of the algorithm is measured by the frequency of finding the optimal solution in 50 runs of the algorithms to solve the same problem. The worst and the best objective function values (OFVs) of the design problem also show the range of non-optimality of the best solutions found by the algorithm.

Table 5 shows that NTAPS increases as the B/C increases from a low value of 0.2 to a mid value of 0.5, and then decreases until B/C reaches a high value of 0.8, a phenomenon expected to occur because the level of feasible and dominate alternative networks has similar variation as the NTAPS. This result is in accordance with that reported by Poorzahedy and Rouhani (2007) for application of ACO.

5.2. Comparison of PSO with ACO and HACO

In this section, the performance of the PSO in solving the TNDP is compared with those of the ACO and the best

Table 5 Performance of PSO algorithm for different levels of B/C.

Row	B/C	Budget level	Average NTAPS[a]	Frequency of finding the optimal solution[a]	Solution OFVs	
					Best[a]	Worst[a]
1	0.20	2700	19.1	50	76,297	76,297
2	0.32	4330	19.6	49	70,353	71,180
3	0.45	6000	20.8	48	66,650	67,576
4	0.49	6500	21.1	48	65,465	66,187
5	0.53	7075	25.2	43	64,580	65,064
6	0.63	8330	24.3	48	61,456	62,560
7	0.75	9980	19.5	48	58,839	60,326
8	0.81	10,820	18.5	43	58,829	58,839

[a] In 50 runs.

Table 6 Performance of PSO, ACO and HACO algorithms (in 50 runs, B/C = 0.625).

Iteration No.	Average NTAPS			Average OFV[a]			Frequency of finding the optimal solution		
	PSO	ACO	HACO	PSO	ACO	HACO	PSO	ACO	HACO
1	10.0	8.6	8.7	65,134	65,134	65,095	12	9	12
2	5.8	5.3	3.9	64,878	64,903	64,595	6	12	12
3	1.5	3.4	2.0	64,637	64,758	64,330	6	7	2
4	1.7	2.1	0.7	64,528	64,663	64,197	2	4	1
5	1.1	2.2	8.4	64,853	64,750	65,631	18	4	20
6	1.6	1.6	0.2	63,654	64,693	63,851	2	4	0
7	1.3	1.2	0.2	63,934	64,794	63,937	1	1	1
8	2.0	0.8	0.1	63,914	64,821	63,924	1	0	0
All	24.3	25.1	24.2	64,452	64,814	64,445	48	41	48

[a] For all particles.

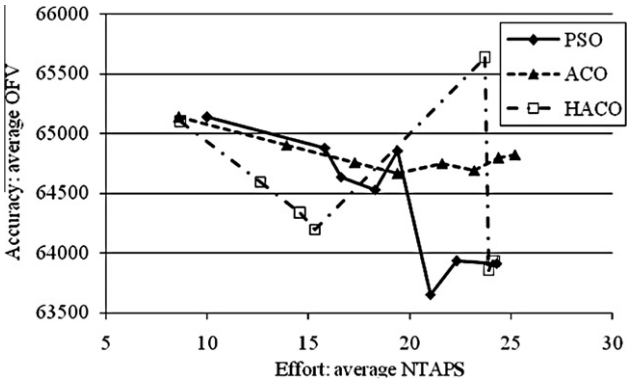

Figure 2 Comparison of PSO, ACO and HACO regarding objective function value and computation effort (in 50 runs, B/C = 0.625).

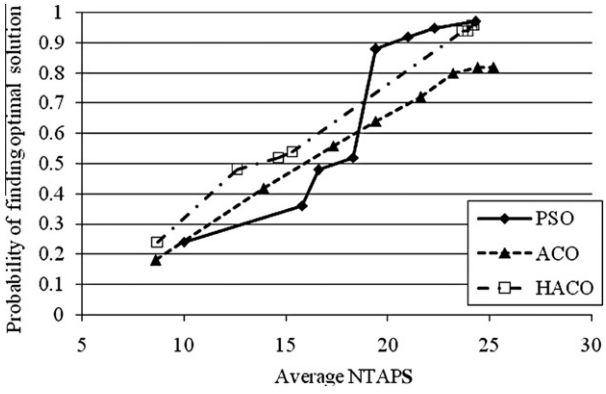

Figure 3 Comparison of PSO, ACO and HACO algorithms in probability of finding the optimal solution (in 50 runs, B/C = 0.625).

HACO algorithms presented by Poorzahedy and Rouhani (2007) for the test network of Sioux Falls. The PSO parameter values used here are as before, and the ACO and HACO results are those given in Poorzahedy and Rouhani (2007). To make the results comparable, the initial solutions of the PSO are made by the same way as described in this reference.

Three measures of performance are considered in Table 6. These are average NTAPS, average of OFVs for all particles, as in Poorzahedy and Rouhani (2007), and frequency of finding the optimal solution. These measures are for 50 runs of each algorithm, and they are given for each of 8 iterations of the algorithms, and in total, for B/C = 0.625. This B/C value is a middle value which requires higher efforts and is exposed to higher errors. As may be seen in this table, except for iteration 5 of the HACO, all algorithms experience decreasing NTAPS as iteration number increases. Moreover, it may be seen that the PSO is comparable with HACO regarding total NTAPS. Fig. 2 demonstrates the performance of these three algorithms in the space of effort-accuracy, where the effort is measured by the average NTAPS, and accuracy is measured by the average OFV for all particles. Fig. 2 is based on 50 runs for each algorithm, and it is given for each of 8 iterations of

Table 7 Performance of PSO, ACO and HACO algorithms for two medium budget levels.

Algorithm	B/C	Average NTAPS[a]	Frequency of finding the optimal solution[a]	Average of solution OFVs[a]
PSO	0.45	20.8	48	66,672
ACO	0.45	25.9	50	66,650
HACO	0.45	29.4	50	66,650
PSO	0.625	24.3	48	61,472
ACO	0.625	25.1	41	61,532
HACO	0.625	24.2	48	61,469

[a] In 50 runs.

them for B/C = 0.625, as in Table 6. According to Fig. 2, the HACO has the least average effort (24.2) and the highest accuracy (64,445), while those of ACO has been the most and lowest ones (25.1 and 64,814, respectively), showing that the HACO is more effective than ACO in solving the problem. In comparison with HACO, the accuracy of PSO (64,452) at a comparable effort (24.3) seems a remarkable performance. This is particularly so, if one notes that the PSO algorithm used here is based on the canonical PSO, which points to the fact that it has the chance of performing better if it is modified and calibrated to suit the problem better.

Fig. 3 depicts another type of the effort-accuracy diagram with the accuracy being measured by the odds (probability) of finding the optimal solution (in 50 runs for B/C = 0.625, as in Table 6), and the effort being measured as before. The figure shows that HACO and PSO have the best accuracy levels (0.96) at comparable effort levels (24.2 and 24.3, respectively). The accuracy of 0.96 seems a very high performance for the PSO, when compares it to that of ACO (0.82) which has been gained at a somewhat higher effort level (25.1).

Table 7 summarizes the results of Table 5 for B/C = 0.625, and compares them with the respective ones for B/C = 0.45 (another mid-value B/C with less difficulty in finding the optimal solution than B/C = 0.625). As may be seen in this table, both ACO and HACO algorithms happen to find the optimal solutions in all 50 runs for B/C = 0.45, while PSO does this in 48 out of 50. In this case, however, PSO has done this with much lower effort than others (20.8 as compared with 25.9 and 29.4, the average efforts of ACO and HACO, respectively). It is worth pointing out that the number of feasible alternative networks for B/Cs of 0.45 and 0.625 are 398 and 761, in that order. Table 7 illustrates that the average NTAPS for the algorithm PSO for B/Cs of 0.45 and 0.625 are 20.8 and 24.3, respectively, which are 5.2% and 3.2% of the total feasible networks at the respective budget levels.

6. Summary and conclusions

In this paper, the meta-heuristic of particle swarm optimization was employed to solve a well-known combinatorial bilevel programming model, namely the transportation network design problem. Various kinds of approaches have been proposed to solve this problem. Following two recent attempts in the solution of this problem by ant colony optimization, ACO (Poorzahedy and Abulghasemi, 2005) and hybrid ant colony optimization, HACO (Poorzahedy and Rouhani, 2007), this paper attempted to show the power of the recent meta-heuristic search, the particle swarm optimization, and compared it to ACO and HACO. The experiments presented in this paper on the network of Sioux Falls showed that the particle swarm algorithm outperforms ACO in that it needs noticeably less effort to find comparable solutions (in the values of the objective function), and have a comparable performance to HACO in that it gives similar solutions at comparable effort. These are promising results which encourage more experiments in this area to explore the capability of this new algorithm further in solving the TNDP, and similar combinatorial problems.

References

Abdulaal, M., Leblanc, L.J., 1979. Continuous equilibrium network design models. Transportation Research Part B 13, 19–32.

Abraham, A., Guo, H., Lio, H., 2006. Swarm intelligence: foundations, perspectives and applications. In: Nedjah, N., Mourelle, L.M. (Eds.), Swarm Intelligent Systems. Springer, Netherlands, pp. 18–25.

Boyce, D.E., Farhi, A., Weischedel, R., 1973. Optimal network design problem: a branch-and-bound algorithm. Environment and Planning 5, 519–533.

Cantarella, G.E., Pavone, G., Vitetta, A., 2002. Heuristics for the network design problem. In: Presented at the EWG 2002 (the 13th Mini Euro Conference), Bari, Italy.

Chen, M., Sul Alfa, A., 1991. A network design algorithm using a stochastic incremental traffic assignment approach. Transportation Science 25, 215–224.

Clerc, M., Kennedy, J., 2002. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6, 58–73.

Dantzig, G.D., Harvey, R.P., Lansdowne, Z.F., Robinson, D.W., Maier, S.F., 1979. Formulating and solving the network design problem by decomposition. Transportation Research Part B 13, 5–17.

Eberhart, R.C., Kennedy, J., 1995. A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science. IEEE Press, Piscataway, NJ, pp. 39–43.

Eberhart, R.C., Shi, Y., 2000. Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the IEEE International Congress on Evolutionary Computation, vol. 1. IEEE Press, Piscataway, NJ, pp. 84–88.

Haghani, A.E., Daskin, M.S., 1983. Network design application of an extraction algorithm for network aggregation. Transportation Research Record 944, National Research Council, Washington, DC, pp. 37–46.

Hoang, H.H., 1982. Topological optimization of networks: a nonlinear mixed integer model employing generalized benders decomposition. IEEE Transactions on Automatic Control 27, 164–169.

Holmberg, K., Hellstrand, J., 1998. Solving the uncapacitated network design problem by a Lagrangean heuristic and branch-and-bound. Operations Research 46 (2), 247–259.

Jiang, M., Luo, Y.P., Yang, S.Y., 2007. Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. Information Processing Letters 102 (1), 8–16.

Kennedy, J., Eberhart, R.C., 1995. Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, vol. 4. IEEE Press, Piscataway, NJ, pp. 1942–1948.

LeBlanc, L.J., 1975. An algorithm for discrete network design problem. Transportation Science 9, 183–199.

Lee, C.K., Yang, K.I., 1994. Network design of one-way streets with simulated annealing. Papers in Regional Science 32 (2), 119–134.

Magnanti, T.L., Wong, R.T., 1984. Network design and transportation planning: models and algorithms. Transportation Science 18, 1–55.

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

Poorzahedy, H., Rouhani, O.M., 2007. Hybrid meta-heuristic algorithms for solving network design problem. European Journal of Operational Research 182, 578–596.

Poorzahedy, H., Turnquist, M.A., 1982. Approximate algorithms for the discrete network design problem. Transportation Research Part B 16, 45–56.

Sheffi, Y., 1985. Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods. Prentice-Hall, USA.

Shi, Y., Eberhart, R.C., 1998a. A modified particle swarm optimizer. In: Proceedings of the IEEE International Conference on Evolutionary Computation. IEEE Press, Piscataway, NJ, pp. 69–73.

Shi, Y., Eberhart, R.C., 1998b. Parameter selection in particle swarm optimization. In: Proceedings of the Seventh International Conference on Evolutionary, vol. VII. Springer-Verlag, New York, USA, pp. 591–600.

Solanki, R.S., Gorti, J.K., Southworth, F., 1998. Using decomposition in large-scale highway network design with quasi-optimization heuristic. Transportation Research Part B 32, 127–140.

Steenbrink, P.A., 1974a. Optimization of Transport Network. John Wiley, New York, USA.

Steenbrink, P.A., 1974b. Transportation network optimization in the Dutch integral transportation study. Transportation Research 8, 11–27.

Voss, M.S., Feng, X., 2002. ARMA model selection using particle swarm optimization and AIC criteria. In: Presented at the 15th IFAC World Congress on Automatic Control, Barcelona, Spain.

Wong, R.T., 1984. Introduction and recent advances in network design models and algorithms. In: Florian, M. (Ed.), Transportation Planning Models. North-Holland, Amsterdam.

Yang, H., Bell, M.G.H., 1998. Models and algorithms for road network design: a review and some new developments. Transport Review 18 (3), 257–278.

Yin, Y., 2000. Genetic algorithm-based approach for bilevel programming models. ASCE Journal of Transportation Engineering 26 (2), 115–120.

Yisu, J., Knowles, J., Hongmei, L., Yizeng, L., Kell, D.B., 2008. The landscape adaptive particle swarm optimizer. Applied Soft Computing 8 (1), 295–304.

You, P.-S., 2008. An efficient computational approach for railway booking problems. European Journal of Operational Research 185, 811–824.