Available online at www.sciencedirect.com

ScienceDirect

European Journal of Operational Research 182 (2007) 578–596

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

Discrete Optimization

Hybrid meta-heuristic algorithms for solving network design problem

Hossain Poorzahedy *, Omid M. Rouhani

Institute for Transportation Studies and Research, Sharif University of Technology, P.O. Box 11365-9313, Tehran, Iran

Received 26 November 2005; accepted 21 July 2006
Available online 15 November 2006

Abstract

Network design problem has been, and is, an important problem in transportation. Following an earlier effort in designing a meta-heuristic search technique by an ant system, this paper attempts to hybridize this concept with other meta-heuristic concepts such as genetic algorithm, simulated annealing, and tabu search. Seven hybrids have been devised and tested on the network of Sioux Falls. It has been observed that the hybrids are more effective to solve the network design problem than the base ant system. Application of the hybrid containing all four concepts on a real network of a city with over 2 million population has also proved to be more effective than the base network, in the sense of finding better solutions sooner.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Transportation; Network design; Ant system; Hybrid algorithm

1. Introduction

Transportation network design problem (NDP) is the problem of improving or expanding transportation networks under resource constraint(s). More specifically, NDP is the problem of choosing among a set of projects so as to optimize certain objective(s) under resource constraint(s). 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.

Traditionally, when origin–destination (O/D) demand is fixed (inelastic), the benefits of the users of the network is measured by the reduction of the total travel time accrued by the choice of certain subset of the given project set, as it is proportional to most user transportation costs (such as monetary travel cost) as well. Moreover, most resources needed to materialize the decisions may be provided by monetary budget, and hence a single budget resource replaces all resource constraints to introduce NDP as follows.

Let, $N(V, A)$ be a network of concern with V as the set of nodes and A the set of links. Let y be the (row) vector of project decisions y_{ij}, where $y_{ij} = 1/0$ represents the acceptance/rejection of project link (i,j), respectively. Let, A_y be the set of project links, and A_{y1} the set of project links which are accepted in y: $A_{y1} = \{(i,j) \in A_y : y_{ij} = 1\}$. Each project link (i,j) has a (construction) cost C_{ij}, and the

* Corresponding author.
E-mail address: Porzahed@sharif.edu (H. Poorzahedy).

0377-2217/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2006.07.038

total construction cost is limited to the budget level B. Let, also, x_{ij} be the flow in link (i,j), and x the (row) vector of link flows. Moreover, let x_p^{ks} be the flow in path p, $p \in P_{ks}$, from origin k to destination s, $(k,s) \in P$, where P is the set of origin–destination pairs, and P_{ks} the set of paths from origin k to destination s. Such flows are generated by O/D demand for travel, d^{ks}, from k to s, $(k,s) \in P$. Finally, let $t_{ij}(x_{ij})$ be the travel time function of link (i,j), $(i,j) \in A \cup A_y$, representing average travel time, which is assumed to be only a function of flow in link (i,j), continuous, differentiable, Riemann integrable, and strictly convex, defined for $x_{ij} \geqslant 0$. Then the traditional NDP may be stated as follows:

$$\text{(NDP)} \quad \min_y \; f(y) = \sum_{(i,j)\in A\cup A_{y1}} x_{ij}^* t_{ij}(x_{ij}^*)$$

$$\text{s.t.:} \quad \sum_{(i,j)\in A_y} C_{ij} y_{ij} \leqslant B, \tag{1}$$

$$y_{ij} = 0/1, (i,j) \in A_y, \tag{2}$$

$$x^* \text{ is the user equilibrium (UE) flow in the network } N(V, A \cup A_{y1}), \tag{3}$$

where x^* is the solution of the following problem. For any given vector of y:

$$\text{(UE)} \quad \min_x \; \sum_{(i,j)\in A\cup A_{y1}} \int_0^{x_{ij}} t_{ij}(u)\,\mathrm{d}u$$

$$\text{s.t.:} \quad \sum_{p\in P_{ks}} x_p^{ks} = d^{ks}, \quad \forall (k,s) \in P, \tag{4}$$

$$x_p^{ks} \geqslant 0, \quad \forall p \in P_{ks}, \; \forall (k,s) \in P, \tag{5}$$

$$x_{ij} = \sum_{(k,s)\in P} \sum_{p\in P_{ks}} x_p^{ks} \cdot \delta_{ij,p}^{ks}, \quad \forall (i,j) \in A \cup A_{y1}, \tag{6}$$

where $\delta_{ij,p}^{ks}$ is 1 if link $(i,j) \in p$, $p \in P_{ks}$, otherwise 0. This is a well-known bi-level programming problem, which is described well in the literature of transportation network design. Steenbrink (1974a), Wong (1984), Magnanti and Wong (1984), and Yang and Bell (1998) review a range of the network design problem and solution algorithms.

NDP is a difficult problem to solve. This was found soon after the early work of LeBlanc (1975) who proposed a branch-and-bound procedure to solve this non-convex programming problem. Thus, a wide spectrum of approaches have been investigated to assess the trade-off between accuracy and speed of the solution. This is done by: (a) attaining a convex programming problem through replacing user equilibrium flow by system equilibrium flow (Dantzig et al., 1979); (b) bringing the complexity of the problem down to a linear function of user travel time by assuming constant link cost function (Boyce et al., 1973; Holmberg and Hellstrand, 1998); (c) constraint relaxation, particularly those regarding the integrality of the projects or integer decision variables (Steenbrink, 1974b; Abdulaal and LeBlanc, 1979; Dantzig et al., 1979); (d) decomposition of the problem based on link type, geographical area, or links of the network to overcome the size of the problem (Steenbrink, 1974b; Dantzig et al., 1979; Hoang, 1982; Solanki et al., 1998), (e) expediting the solution procedure through aggregation of the network by link and node abstraction or extraction (Haghani and Daskin, 1983); (f) facilitating the design procedure using an intrinsic approach which defines a new design problem that lacks the complexity of NDP (Yang and Bell, 1998); (g) good approximate and heuristic procedures to solve the problem by, e.g., replacing the objective function by another well-behaved function (Poorzahedy and Turnquist, 1982), or other heuristics to solve the problem (Chen and Sul Alfa, 1991); and (h) heuristic procedures of modern types, such as Genetic algorithm (GA) (Yin, 2000), Simulated Annealing (SA) (Friesz et al., 1992; Lee and Yang, 1994), GA, SA, and Tabu Search (TS) (Cantarella et al., 2002), and Ant system (AS) (Poorzahedy and Abulghasemi, 2005). To observe brevity of the discussion, the interested readers are referred to the first three reviewing references and the last reference for more discussion and references in the area of network design problem.

This paper, in the continuation of a previous work, aims to present some hybrid algorithms which enhance the performance of Ant System presented by Poorzahedy and Abulghasemi (2005) to solve NDP. This paper is organized as follows: First, relevant modern heuristic algorithms are reviewed in Section 2. Next, the proposed hybrid algorithms are presented and discussed in Section 3. Numerical examples to investigate the properties of the proposed algorithms are given in Section 4. Section 5 is devoted to present the results of the application of the algorithms to a real case. Finally, Section 6 concludes the paper.

2. Meta-heuristic algorithms

Modern heuristic techniques, also called meta-heuristics are a family of procedures which benefit some sort of intelligence in their search for finding the solution of a problem. Although one may not prove that the solutions found by most of them are optimal solutions, experiences during the past two decades have shown that they find the solution, or a very good solution, rapidly and effectively. This is particularly so, for the combinatorial problems, such as NDP. Neural Network (NN), Simulated Annealing (SA), Genetic algorithm (GA), Scatter Search (SS), Tabu Search (TS), and Ant system (AS) are among these modern heuristics (Onwubolu, 2002; Goldberg, 1989). Each of such techniques possesses certain specific intelligence to search for the solution of a problem. Clearly, a collection of such abilities enhance the power of the search engine. This is why that attention is being paid recently to hybrid algorithms which collect some of these intelligent senses and procedures. Cantarella et al. (2002) present one such endeavor and Greistorfer (1998) introduces another.

It is not the intention of this paper to review meta-heuristic algorithms. However, to make this paper self-contained, a brief description of each such algorithm which will be used in this paper will be presented in what follows. For more detailed discussion, the interested readers are referred to the appropriate references in the literature and those cited in this paper.

2.1. Genetic algorithm

Genetic algorithm (GA) is a population-based mechanism in the traditional procedure of which every two parent solutions give birth to two child (successor) solutions, transferring a new combination of genes in the form of new chromosomes to them. Each chromosome is identified by genes (decisions) accepting some values such as 1 (acceptance of a character), and 0 (rejection of that character), carrying a value which shows its fitness or effectiveness (in solving the problem). The population size is kept fixed, always made up of the best valued individuals (chromosomes). Like the natural phenomena in genetics, genes are mutated; and to make sure that new generations never fall behind their predecessors the chromosomes of the elites of the population are always kept in the new generation. To have a workable genetic algorithm, one needs to have (a) a mating procedure, (b) a chromosome value estimation procedure, and (c) supplemental procedures, such as mutation, and elite preservence. A general version of this algorithm is as follows:

Step 0. Initialize.
Step 1. Create a population of individuals.
Step 2. Compute the (fitness) values of the individuals.
Step 3. If convergence criteria are satisfied, go to step 7.
Step 4. Identify parent individuals randomly based on a probability proportional to their fitness values.
Step 5. Create the children of a pair of parent individuals by their crossovers.
Step 6. Choose chromosomes and genes for mutation, and go to step 2.
Step 7. Stop.

There are several proposed procedures for implementing each of the above steps. The readers are referred to Goldberg (1989) for further readings in this area.

To adapt this concept to network design problem, one may define a gene to represent a project, taking a value of 1 to show its presence in a network, and 0 otherwise. A chromosome is a vector of such genes, whose fitness value is the amount of reduction of the total travel time of the users in the network as a result of the presence of the collection of the projects in the chromosome (network). Pairs of chromosomes break in one or more points, and appropriate segments of mating chromosomes interchanged to create the respective off-springs. Population survival obey the natural law of ‘‘Stronger creatures live longer’’.

2.2. Ant systems

Ants have remarkable capability in finding sources of food through a cooperative process. Ants, belonging to a colony, leave the nest and move randomly around in the search of a nearby feeding source. When such a source is discovered, they come back to inform others about it, while depositing a chemical substance, called pheromone, sensible by ants, to mark the path leading to the food source from the nest. Pheromone evaporates over time. Hence, only trails with higher pheromone concentration persist in leading ants. This can happen when other searching ants find the food and emphasize this information by laying their pheromone on the ground on the same trails. Thus, less frequently visited (e.g., distant) food sources lose address by evaporation of pheromone marker of the paths leading to them from the nest. This is the cooperative mechanism that an ant colony converges towards an optimal solution. Dorigo et al. (1991) and Dorigo et al. (1996) describe this approach:

Let t be a time counter, s a tabu list index, i a point (vertex) in a space ($i = 1, \ldots, n$), $\tau_{ij}(t)$ pheromone intensity on the edge joining i and j at time t, and $\Delta\tau_{ij}$ an increment of such intensity. Let, also, m be the number of ants, tabu$_k(s)$ represent the "visited" vertex by ant k in its sth move, and ρ be the pheromone evaporation rate, $\circ \leqslant \rho \leqslant 1$. A general version of an ant system (AS) algorithm may be stated as follows (Dorigo et al., 1996):

Step 0. Initialization. $t := 0$, $\tau_{ij}(t) := c$ and $\Delta\tau_{ij} := 0$ for all (i,j). Place m ants on the n nodes.

Step 1. Move. $s := 1$, and for $k = 1$ to m do: place the starting vertex in tabu$_k$, and repeat until ant energy is exhausted (a resource constraint violation). Choose vertex j to move to with probability p^k_{ij} (as defined below). $s := s + 1$. Insert j in tabu$_k$ (i.e., tabu$_k(s) = j$).

Step 2. Pheromone update. For $k = 1$ to m do: move ant k to its nest. Compute the value of the objective function for ant k path, and the respective pheromone increment $\Delta\tau^k_{ij}$ (as defined below). Save the best solutions found, and the respective characteristics. For edge (i,j) and for $k = 1$ to m do:

$$\Delta\tau_{ij} := \Delta\tau_{ij} + \Delta\tau^k_{ij}, \tag{7}$$

$$\tau_{ij}(t+1) := \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}, \tag{8}$$

$$t := t+1, \quad \Delta\tau_{ij} := 0, \quad \text{for all } (i,j).$$

Step 3. Stopping criteria. If stopping criteria are met (and stagnation behavior not observed), stop. Else, empty tabu lists for all ants, and go to step 1.

Let v_{ij} be a measure of the "visibility" of vertex j from the vertex i, associated with the goodness of visiting vertex j as seen from the vertex i (and possibly, given the past history of the vertices visited in tabu$_k(\cdot)$). One such measure may be defined as the net benefit of visiting a vertex alone (irrespective of visiting other vertices). Then, p^k_{ij} is a choice function (for ant k currently at vertex i, to move to a vertex j) which utilizes visibility of j from i (v_{ij}) and amount of pheromone on edge (i,j) (τ_{ij}) such as

$$p^k_{ij} = \begin{cases} \tau_{ij}(t)^\alpha \cdot v^\beta_{ij} / \sum_{m \in F_k} \tau_{im}(t)^\alpha \cdot v^\beta_{im}, & \text{if } j \in F_k, \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

where F_k is the set of vertices not in the tabu$_k$ (i.e., the set of allowable vertices), and α and β are two parameters representing the relative importance of trail intensity (at time t) and the visibility information.

To determine the amount of pheromone to be laid on the edges, let $\Delta f(y^k)$ be the net benefit of the path chosen by ant k in the space of concern, y^k. Then, $\Delta\tau^k_{ij} := \Delta f(y^k)$ if $(i,j) \in y^k$, and 0 otherwise.

2.3. Simulated annealing

Simulated Annealing (SA) is an algorithm simulating the annealing process of metals. It consists of heating the metal to a suitable temperature, "soaking" it at that temperature for certain time period, and then reducing its temperature at a specified rate, so as to create a specific molecular structure order for possessing certain mechanical properties (Salmon et al., 2002).

Analogous to an annealing process, SA procedure is based on a Monte-Carlo model to solve a combinatorial optimization problem (Onwubolu, 2002). Any solution, y^k, to the combinatorial optimization problem is a configuration of the problem (matter) corresponding to a state of a solid material. The objective function $f(y^k)$ corresponds to energy of a solid, and the control parameter c corresponds to the temperature of the solid. SA is a procedure which generates a sequence of configurations, each being found by perturbing a current configuration and accepted according to Metropolis criterion, to reach an equilibrium configuration for a given temperature (c); and then reduces the temperature until a frozen configuration (with the lowest energy) is achieved. This equilibrium configuration corresponds to the solution of the problem. A general statement of this algorithm follows (see, e.g., Onwubolu, 2002):

Step 0. Initialization. c is given a high value. Consider an initial configuration i.

Step 1. Solution generation. Perturb the current configuration, i, obtain configuration j, with costs $f(y^i)$ and $f(y^j)$, and $\Delta f_{ij} = f(y^j) - f(y^i)$.

Step 2. Acceptance criterion. Accept y^j with the following probability (Metropolis criterion): Pr(accepting y^j) = 1, if $\Delta f_{ij} \leqslant 0$, and $e^{-\frac{\Delta f_{ij}}{c}}$ if $\Delta f_{ij} > 0$. (That is, $y^i := y^j$ if $\Delta f_{ij} \leqslant 0$, or if a generated random number in $[0, 1)$ is less than $e^{-\frac{\Delta f_{ij}}{c}}$.) Continue this process until the new configuration reaches an equilibrium, when the probability distribution of configuration y^j approaches Boltzman distribution given below:

$$\Pr(\text{configuration} = y^j) = q_j(c) = \frac{1}{Q(c)} e^{-\frac{\Delta f_{ij}}{c}}, \tag{10}$$

where $Q(c)$ is the normalizing constant.

Step 3. Change of temperature. Lower the value of the parameter, c, in steps, while allowing the system to reach equilibrium through steps 1 to 2 for each temperature level.

Step 4. Stopping criterion. Terminate the process when no improvement in the cost of the alternative solutions exists (approximately). Solution of the combinatorial optimization is the final "frozen" configuration.

2.4. Hybrid algorithms

Various heuristic algorithms have different mechanisms as constituents of their intelligence. There are two points worth emphasizing in this respect: (a) Some of these algorithms may have better performance to solve certain problems, because of the structure of the problem which is more exposed to the investigating algorithm. Moreover, one expects (b) an algorithm with a union of such intelligent mechanisms to perform better than the constituting algorithms alone, and such hybrid algorithms should be able to solve general problems more efficiently.

This very belief has led many researchers to build hybrid algorithms. Such endeavors include three approaches: (a) One approach is to build parallel heuristic techniques for solving combinatorial optimization problems (Resende et al., 1999). (b) Crossing two or more breeds of algorithms to create hybrids for optimization is another approach. One popular avenue of research in this respect is combining population-based procedures with local search ones so as to gather two aspects of a good searching technique, viz diversification and intensification, together. This approach includes genetic algorithm-tabu search hybrids for solving optimization problems (e.g., Glover et al., 1995; Costa, 1995; Greistorfer, 1998; Fleurent and Ferland, 1999); genetic algorithm-simulated annealing hybrid solution procedures (e.g., Kim et al., 1994); simulated annealing–case based reasoning (Sadek, 2001); and tabu search and neural networks (De Werra and Hertz, 1989). And, the third approach is (c) using meta-heuristic solutions to start or guide other algorithms (Chelouah and Siarry, 2003; Taillard and Gambardella, 1997).

Hertz and Widmer (2003) present some guidelines for the use of meta-heuristics in solving combinatorial optimization problems, which may also be used in creating such algorithms in hybrid forms.

3. The proposed hybrid algorithms

Following a previous endeavor by Poorzahedy and Abulghasemi (2005), in presenting an AS for solving transportation network design problems, this paper tries to enhance the ability of such an algorithm by hybrids which imbed additional intelligence into the solution procedure. To do this, first a base AS algorithm will be introduced, and next some hybrid algorithms are presented, whose solution capabilities are measured based on their applications in solving one well-known test network of Sioux Falls, and one large real case network.

3.1. The base AS algorithm

The base AS algorithm for solving NDP is introduced in Poorzahedy and Abulghasemi (2005). A graph is constructed as in Fig. 1, where the vertices represent the projects (i), which are connected to other projects (j) by undirected edges (i,j) with costs equal to the construction cost of vertex j. Ants are placed at starting vertices, called nests, to start moving from vertex to vertex in the hope of finding a good basket of food, while their energy for search last. They leave a pheromone trail on the edges, and return to their nests, when energy-wise it is time to return. This energy comes from the budget avail-

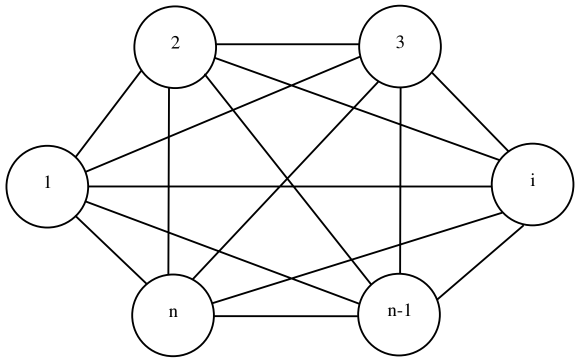

Fig. 1. A graph representation of project selection for ants.

able to the ant, B, and will be spent by an amount c_l, when project l (including that of the nest) is picked. Previous experiments show that, a good strategy is having a number of ants equal to the number of candidate projects, and let ant m possess nest m. This would diversify the solutions (Poorzahedy and Abulghasemi, 2005).

Ants find their paths from vertex (i) to vertex (j) according to the visibility of vertex j from vertex i, v_{ij}, and the concentration of pheromone on the edge (i,j), τ_{ij}. As in Poorzahedy and Abulghasemi (2005), the choice of the next vertex, j, by an ant k currently (at time t) at vertex i, follows a choice process governed by a multinomial logit:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{e^{u_{ij}(t)}}{\sum_{m\in F_k} e^{u_{im}(t)}}, & \text{if } j \in F_k, \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

where F_k is as defined in Eq. (9), and $u_{ij}(t)$ is the utility of choosing j from i at time t. At time t, the utility of vertex j, among such vertices allowable to choose from by ant k in F_k, is a function of the pheromone concentration on the edge leading i to j, $\tau_{ij}(t)$, and other variables such as the goodness of project j alone,

$$v_{ij} = v_j = f(y^0) - f_j, \tag{12}$$

where f_j is the value of the objective function when only project j is constructed in the network, and $f(y^0)$ is this value for the do nothing alternative. Then, one may define the utility of choosing vertex j, for example, as follows:

$$u_{ij}(t) = \alpha \cdot \tau_{ij}(t) + \beta \cdot v_j, \tag{13}$$

where α, and β are the relative importance of the variables in the utility function, and are given parameters.

After completing the tour and returning to the nest, ant-gathered information may be transferred to the edges of the design graph by computing the objective function of the selected set of projects (set of visited vertices), y^k, and computing the net benefit of this set found by ant k, as follows:

$$\tau_{ij}^{k}(t) = \begin{cases} [f(y^0) - f(y^k)] - \gamma c^k, & \text{if } (i,j) \in A^k, \\ 0, & \text{otherwise}, \end{cases} \tag{14}$$

where A^k is the set of edges in the path traversed by ant k, and c^k is the total cost of the projects selected by ant k. γ is a constant which converts monetary units into travel time saved (in vehicle-hours). For the m ants sent each time before updating the edge pheromone, one may write:

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{m}. \tag{15}$$

Then, for the next iteration $t+1$, or $t+m$ depending on the definition, the pheromone concentration of the edges may be updated as follows:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij}, \tag{16}$$

where $1-\rho$ is the evaporation rate of pheromone trail, $0<\rho<1$.

The following is an adaptation of the ant-cycle algorithm (Dorigo et al., 1996) to the NDP, referred to as the base algorithm. The parameters of the algorithm are set equal to the best values obtained by Poorzahedy and Abulghasemi (2005), as $m =$ no. of projects, $\rho = 0.5$, $\beta = 1.0$, $\gamma = 1.0$, and vertex $k =$ nest of ant k. One other parameter, α, is chosen appropriately, and kept fixed for each run.

Algorithm 1. The base Ant System network design algorithm

Step 0. *Initialization.*
• Set $t := 0$; solve UE flow problem for $y = y^0$ and compute the objective function of the design problem $f(y^0)$. $f^* := f(y^0)$, $y^* := y^0$. • Set $\tau_{ij} := 0$, and solve UE flow problem for each project j and compute the respective design objective function f_j, $j = 1,\ldots,m$. Set iteration counter $c := 0$.

Step 1. *Move.* • Empty tabu lists for all ants. • For $k = 1$ to m, do: Set ant k on project vertex k, and place vertex k in the respective tabu list.
• Repeat the followings until tabu list becomes budget infeasible: specify the allowable projects F_k; compute the utility of each project $j \in F_k$ according to Eq. (13), and then the respective probability of choice according to Eq. (11); choose the next project vertex using a Monte-Carlo simulation based on the latter values.
• Next, identify the set of chosen projects for ant k, and solve the UE flow problem for the selected projects of vector $y = y^k$, and compute the respective design objective function $f(y^k)$. (Save y^k and $f(y^k)$ to avoid re-computation of same information in the future.) If $f(y^k) < f^*$, set $f^* := f(y^k)$ and $y^* := y^k$.

Step 2. *Pheromone Update*. • For each ant $k = 1$ to m do: Compute pheromone increment according to Eq. (14). • Add pheromone increments for all ants to compute the total such increment $\Delta\tau_{ij}$ as in Eq. (15), and update the pheromone of each edge (i,j) according to Eq. (16). • Set $t := t + 1$, and $\Delta\tau_{ij} := 0$ for all (i,j).

Step 3. *Stopping Criteria*. Set $c := c + 1$, and if $c \leqslant c^0$ (say 10) go to step 1, otherwise y^* is the best solution found so far, with design objective function value $f(y^*)$. Collect the necessary information, and stop.

A description of the algorithm may be found in Poorzahedy and Abulghasemi (2005). However, before passing it is worth noting that since the presence of all projects in the path found by an ant k in the search graph contribute to the total benefit, it is advisable to lay pheromone on all edges of the graph linking these projects (and not only the edges in the path of ant k). As an example, if projects i, j, and k are found by an ant in a path i–j–k, then pheromone is laid on edges (i,j), (i,k), and (j,k), and not only the edges (i,j) and (j,k) in the path. This is the practice in this study.

3.2. Improvement mechanisms of the base algorithm

3.2.1. (I) Improvement 1

Dorigo and Gambardella (1997) present an Ant Colony System (ACS) which follows a rule similar to the following in choosing the vertex among the allowable ones for ant k:

$$j = \begin{cases} \arg\max_{m\in F_k}\{u_{im}\}, & \text{if } r \leqslant r_0, \\ J, & \text{otherwise}, \end{cases} \tag{17}$$

where r is a random variable with uniform distribution in $[0,1]$, and r_0 is a pre-specified parameter ; and J is a random variable selected according to Eq. (11) (Also, see Maniezzo et al., 2003). Incorporating (17) and (11) into ant's decisions, instead of (11) alone, favors projects with higher utilities (higher pheromone trails, higher visibility, lower cost, or higher goodness or values).

The following improvement mechanism has been devised to favor solutions which are stronger, without reducing the width of the spectrum of the solutions that could be found by the ants, but only emphasizing the stronger solutions.

Imp.1: In step 2 of the base algorithm: • Find the top 3 solutions found by the m ants (which have the least values of the design objective function). Consider pheromone updating only for ants k corresponding to these 3 solutions (instead of the m solutions of all ants).

3.2.2. (II) Improvement 2

Mutation may be viewed as a local search in the solution space. In another experiment, a mutation mechanism has been devised to perform such a search in the vicinity of the preferred solutions, with the hope that they have higher potential to reveal better solutions than others. With such intention, then, naturally these mutations must be motivated consciously based on the available information, rather than randomly. In order to make sure that the available information is mature enough and valid to guide such search, mutation is performed in mid-iteration numbers (here iteration 5) of the algorithm. Since the mutated solutions should be feasible resource-wise, accepting a project which is rejected in the solution under mutation operation, should coincide with rejection of one (or more) accepted project(s) depending on the resource use, or costs, of the projects.

The second proposed improvement mechanism borrows ideas from local search techniques such as tabu search, and takes advantage of a measure of relative merit for projects such as the frequency of presence in the solutions found. The improvement, which is a kind of embedding a local search mechanism within a population search procedure, is as follows:

Imp.2: In the 5th iteration of the base algorithm, replace the usual operations in step 1 by the following mutation operation. Find the set of top 3 solutions of each of the previous iterations skip repeated solution and choose the next best solution, so that a maximum of 12 different solutions are obtained. Compute the frequency of the presence of the genes (projects) in the solution chromosomes (vectors) in this set. Next, find the set of top 2 solutions of each, and all, of previous iterations (maximum of $m = 10$ solutions, and repeated solutions are allowed). For each (repeated) solution in this set, replace the project with the (next) least frequency of presence, with the (next) most frequently present project not chosen in that solution, while keeping the solution feasible, and making sure that the new solution has not been found and examined before. Call the new set of solutions the mutated set.

Perform the UE traffic assignment procedure for the mutated set of solutions, Compute the respective objective functions, and treat them as the solutions found by (the $m=$) 10 ants in an iteration of the algorithm. Save the solution information $(y^k, f(y^k))$, and if $f(y^k) < f^*$, set $f^* := f(y^k)$ and $y^* := y^k$.

Next, perform step 2 (pheromone updating) of the algorithm, as usual.

Non-random mutation operations have been exploited by the other researchers, as well. Cree et al. (1998) exploits gradient of the objective function to guide local search. Fleurent and Ferland (1999) have, also, combined genetic mutation concept with ideas in tabu search. Such ideas of project replacements are also present in earlier works in project selection problems. Ahmad (1983) has exploited such an idea in his gradient projection technique in choosing road maintenance projects.

3.2.3. (III) Improvement 3

The two improvement mechanisms (1 and 2) discussed above favor strong genes, and species and genes, respectively. The next improvement mechanism borrows ideas from Simulated Annealing (SA) to reduce the computational burden of the search. Kim et al. (1994) have brought in the idea of controlling the population of the weaker species by exploiting Metropolis Criterion.

In solving NDP, the major computational burden of the solution procedure is the time it takes to solve a UE flow problem to compute the objective function value for a given solution y^k. To prevent solving the latter problem for a seemingly (or potentially) irrelevant solution, one may take advantage of this idea that information regarding the fitness of a solution increases as the procedure proceeds. SA procedure provides such an increasingly tighter sieves to sift coarser entities, while providing higher diversity at the beginning.

To implement this idea, define $g_1(n,k)$ as the goodness of the combination of the projects in a solution found by ant k, y^k, in iteration n, to be equal to the sum of the pheromone laid upon the edge joining each pair of the projects accepted in y^k. The more each pair of projects reveal themselves in solutions of the m ants in a given iteration, the higher the pheromone laid on the edge joining them. (This concept may be conceivably generalized for each p-project combination, $p = 2,3,\ldots$) And, hence pairs of projects which accentuate the net benefit of each other more than other combinations of projects happen to occur more frequently in the solutions found by the ants, which in turn reflect themselves into measure $g_1(.)$ through τ_{ij}. Thus, let

$$g_1(n,k) = \sum_{s=1}^{m} \sum_{t=s+1}^{m} a \cdot \tau(\text{tabu}_k(s), \text{tabu}_k(t)), \tag{18}$$

where $\text{tabu}_k(s)$ contains the index of the project visited in the sth move of ant k, $s = 1,2,\ldots$ If $\text{tabu}_k\,(s) = i$ and $\text{tabu}_k(t) = j$, then τ_{ij} is the pheromone on the edge joining them. a is a scaling constant in Eq. (18). Let $\bar{g}_1(n)$ be the average value of such goodness measures for all ants in iteration n:

$$\bar{g}_1(n) = \frac{1}{m}\sum_{k=1}^{m} g_1(n,k). \tag{19}$$

Define $\bar{\bar{g}}_1(n)$ as the moving average of the two previous values of $\bar{g}_1(.)$:

$$\bar{\bar{g}}_1(n) = [\bar{g}_1(n-1) + \bar{g}_1(n-2)]/2, \tag{20}$$

$g_1(n,k)$ in Eq. (18) may be envisaged as a measure of the energy level of the configuration (of the network or projects) found by ant k in iteration n, and $\bar{g}_1(n)$ and $\bar{\bar{g}}_1(n)$ as the respective average values defined according to Eqs. (19) and (20). Define the change in energy levels of the configuration found by ant k in iteration n as compared to an average value (here, of the two previous average energy levels), $\bar{\bar{g}}_1(n)$, be as follows:

$$\Delta E(n,k) = g_1(n,k) - \bar{\bar{g}}_1(n). \tag{21}$$

Thus, analogous to the Metropolis Criterion in SA algorithm, a solution y^k with lower energy level (goodness) than the average $\bar{\bar{g}}_1(.)$ will only be accepted with probability of $e^{\frac{\Delta E(k)}{C}}$. (Note that, unlike the reduction of energy level in simulated annealing, we would like to see the energy of a configuration found by an ant to increase in our case. Of course, there is a limit to this energy level, which is defined by the optimal configuration. If a solution is rejected we may save the computation time of solving the respective UE flow problem. Thus, the proposed improvement in the base algorithm may be stated as follows.

Imp.3: In step 1 of the base algorithm, in iteration n, $n > 2$: Compute $\Delta E(k) = \Delta E(n,k)$ in Eq. (21). When the solution of ant k, y^k, is found, the UE flow problem is solved for this solution only if $\Delta E(k) > 0$, and if $\Delta E(k) \leqslant 0$ this problem is solved with a probability of $\Pr\{\text{accepting } y^k\} = e^{\Delta E(k)/c(n)}$, where $c(n)$ is a decreasing function of n (e.g., $c(n) = 0.9c(n-1)$). (To check the acceptability of y^k with non-positive ΔE, a uniformly distributed random variable r in $[0,1]$ is generated, and y^k is accepted if $r < e^{\Delta E(k)/c(n)}$.)

An alternative measure of the goodness of the configuration (solution) y^k is defined as follows. Let, $N(n, y_l^k)$ be the frequency of accepting project l (=1 to the number of projects = m, the number of ants, here) in the solutions found by the m ants in iteration n. Let,

$$g_2(n,k) = \sum_{l=1}^{m} N(n, y_l^k) \tag{22}$$

be the sum of such frequencies of the projects accepted in y^k. Defining, as before:

$$\bar{g}_2(n) = \frac{1}{m}\sum_{k=1}^{m} g_2(n,k), \tag{23}$$

$$\bar{\bar{g}}_2(n) = [\bar{g}_1(n-1) + \bar{g}_1(n-2)]/2, \tag{24}$$

$$\Delta E(n,k) = g_2(n,k) - \bar{\bar{g}}_2(n), \tag{25}$$

a similar procedure of Imp.3 may be run using this new definition of ΔE(·).

3.2.4. (IV) Improvement 4–7

Each improvement in (I) through (III) above brings a new variation of the base algorithm, increasing its efficiency in solving NDP. It would be interesting to see the resulting effect of hybrids of such improvements, as follows:

Imp.4: Imp.2 + Imp.3, Imp.5: Imp.1 + Imp.2, Imp.6: Imp.1 + Imp.3, Imp.7: Imp.1 + Imp.2 + Imp.3.

4. Numerical example

To show the performance of the proposed algorithms, they will be applied to the case of the network of Sioux Falls. This network, shown in Fig. 2, has 24 nodes and 76 links. There are 10 pairs of project links, of which 5 projects are improvements on the existing links, and 5 are new links. The parameters of the average travel time functions $t_{ij} = a_{ij} + b_{ij}x_{ij}^4$, for link (i,j), and the origin/destination (O/D) demand, are basically those given in Poorzahedy and Turnquist (1982) and LeBlanc (1975), and are omitted here for brevity. The construction costs of projects 1 to 10 are, respectively, 625, 650, 850, 1000, 1200, 1500, 1650, 1800, 1950, and 2100 units of money. To measure the accuracy of the solutions given by the algorithms, all possible combinations of 2^{10} (=1024) different networks have been created for the network under study, in order to identify the exact solution of the problem under various budget levels. Since the meta-heuristic algorithms of concern are of stochastic nature, each case has been solved 50 times, and the average measures of such runs have been reported as the measure of performance of the algorithms. The average CPU time for solving one UE flow problem is under 0.1 seconds of a personal computer equivalent to Pentium IV, 1400 MHz.

4.1. Application of base AS algorithm

First the performance of the base AS algorithm will be discussed. In all experiments the following parameters are set as shown, unless otherwise specified: $m = |A_y| = 10$, $\rho = 0.5$, $\beta = 1$, $\gamma = 1$, nest of ant k = project k. Table 1 shows the results of solving the NDP for the test network under various budget levels, as measured by budget to total project cost ratio (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, as shown in Table 1. In this table the cost of solving the problem is measured by the CPU time of the computer, which is (almost) directly proportional to the number of UE flow problem (assignments) solved. 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 best values of the objective function of the design problem also show the level of "cost" of non-optimal solution found by the algorithm.

Table 1 shows that no. of assignments increases as the B/C increases from a low value of 0.2 to a mid value of 0.5, and then decreases when B/C reaches a high value of 0.8, a phenomenon expected to occur because of the level of feasible and dominating alternative networks which has similar variation as the no. of assignments.

The following experiments have some interesting results. Suppose projects are of equal cost (this is no restriction), and that budget allows choosing 5 projects. Fig. 3 shows that increasing α from 0 to 0.5 reduces the average no. of assignments (in 50 runs) required to solve the problem. This is because increasing α increases the effect of pheromone concentration on the choice process in Eq. (11), which means better and more frequently chosen projects will be chosen more, and hence there will be less

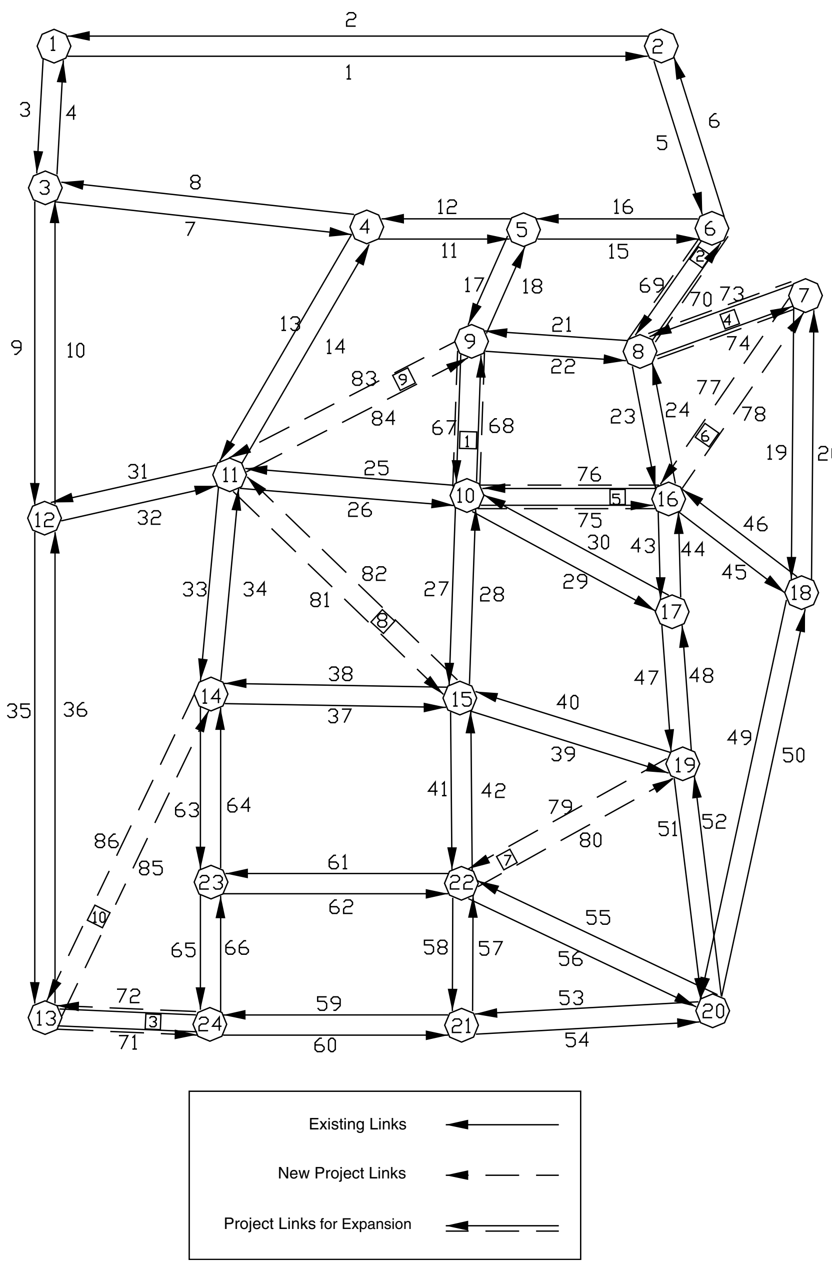

Fig. 2. The test network.

diversity in the solution set found by various ants. This results lower no. of assignments for higher value of α. This observation suggests the use of lower α values for lower B/C levels for which alternative feasible solutions become more diverse, and examining them requires lower α values.

Two experiments have been designed to investigate the effect of the goodness of project j, v_j, upon the efficiency of the algorithm. For the above 5 project budget example with $B/C = 0.5$, first the algorithm has been run for the standard case ($\alpha = 0.05$, and $\beta = 1.0$), and next with $\alpha = 0.05$, and $\beta = 0.0$ so as to omit any influence of the project goodness measure, v_j's. The average no. of assignments (of the 50 runs for each case) increased from 34 for the standard case ($\beta = 1$) to 66.5 for $\beta = 0.0$, while the frequency of finding the optimal solution decreased from 44 (out of 50) for $\beta = 1$ to 19 (out of 50) for $\beta = 0.0$. That is, for about 100% extra effort the accuracy of the solution dropped

Table 1
The performance of the base AS algorithm for different levels of B/C[a]

Row	B/C	Budget level	Average number of UE assignments[b]	Frequency of finding optimal solution[b]	Value of objective function	
					Best[b]	Worst[b]
1	0.20	2700	19.2	50	76,297	76,297
2	0.32	4330	28.0	50	70,353	70,353
3	0.45	6000	25.9	50	66,650	66,650
4	0.49	6500	23.9	50	65,465	65,465
5	0.53	7075	26.6	42	64,580	65,064
6	0.63	8330	25.1	41	61,456	62,560
7	0.75	9980	20.7	50	58,839	58,839
8	0.81	10,820	18.1	39	58,829	58,839

[a] $\alpha = 0.05$, no. of iterations $= c = 8$ for all runs.

below 50%. This shows that v_j's have a profound effect in guiding the search toward the optimal solution with lower effort.

Fig. 4 shows several performance measures of the base algorithm for solving the NDP for a medium budget level ($B/C = 8330/13320 = 0.625$) for three values of α ($=0.00$, 0.05, and 0.20). Other parameters are set at their standard levels mentioned before. As may be seen in this figure, lower α levels results in higher number of assignment problems to be solved. A favorable value of α seems to be $\alpha = 0.05$, for which a good accuracy (frequency of finding the optimal solution) exists at an acceptable cost (average number of assignment problems solved). Fig. 4a shows that on the average fewer traffic assignment problems are solved as iteration number increases, and Fig. 4b indicates that the frequency of finding the optimal solution in the first few iterations of the problem is rather high, because of exploiting the value of the information v_j. Note that in this case it happens that only with the information of v_j the frequency of finding the optimal solution is higher than the other two cases with positive value of α. Fig. 4c shows that convergence rate of the algorithm increases with α, which in turn increases the chance of getting trapped at the local minima. For $\alpha = 0$, there is no information exchange among ants so that the average value of the objective function (of all solutions of each iteration) alternates over iterations. And, finally, Fig. 4d shows that the objective function of the new solutions found by the algorithm in an iteration increases as the iteration number increases, particularly for the last iterations. This observation may be used to enhance the efficiency of the algorithm. Similar trends as discussed above for Fig. 3 are observed for other values of budget as well.

In what follows, the effects of various improvements on the base algorithms are investigated, both separately and in combination with each other.

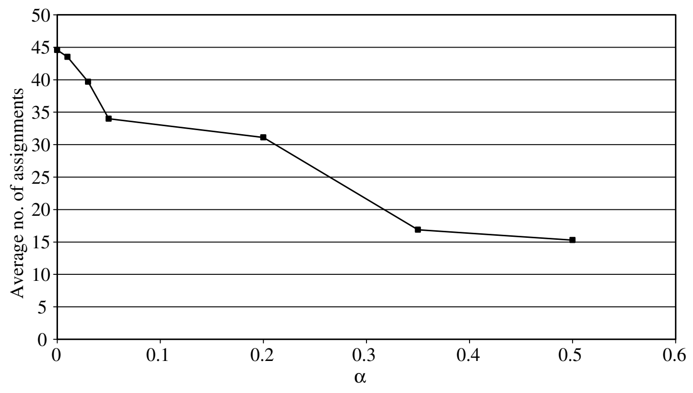

Fig. 3. Effect of α on the no. of assignment problems to be solved ($B/C = 0.5$, all project costs are equal).

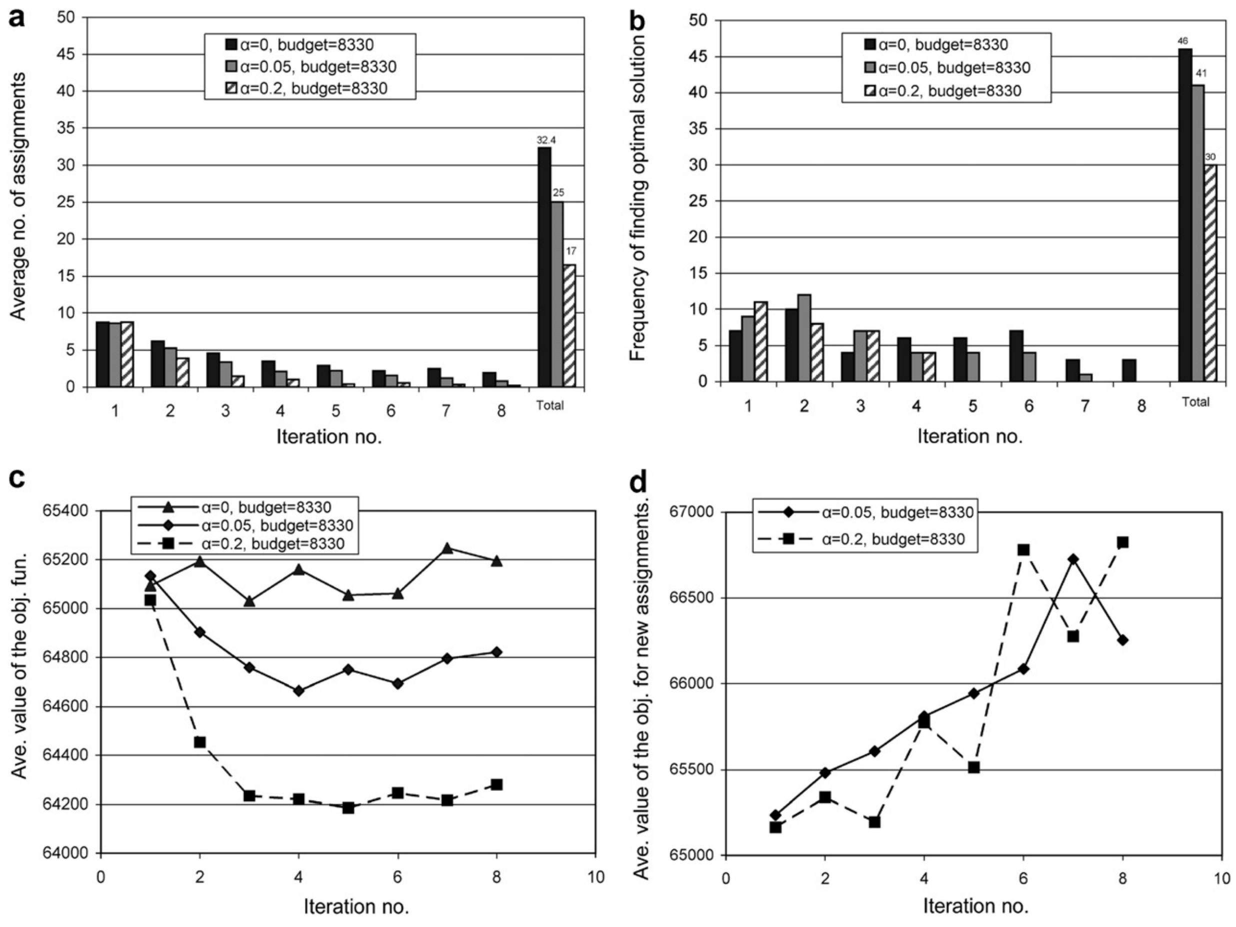

Fig. 4. Effect of α on some performance measures of the base algorithm (in 50 runs, $B/C = 0.625$).

4.2. Application of hybrid ant systems

In this section the performance measures of the proposed hybrids will be compared with the respective ones for the base algorithm. Recall that the following improvements have been introduced before:

Imp.1: Laying pheromone on the top 3 solutions (AS-ACS);
Imp.2: Purposeful mutation of inferior genes (AS-GA-TS);
Imp.3: Incorporation of Metropolis Criterion in rejecting lower energy configurations (AS-SA);
Imp.4: Imp.2 + Imp.3;
Imp.5: Imp.1 + Imp.2;
Imp.6: Imp.1 + Imp.3;
Imp.7: Imp.1 + Imp.2 + Imp.3.

Table 2 shows the result of applying the 7 hybrid algorithms, as well as the base algorithms, on the network of Sioux Falls for solving a network design problem with $B/C = 0.625$ (a mid-value with higher difficulty than more extreme values), $\alpha = 0.05$ (the better value in Fig. 4), $\beta = 1$, and other parameters as specified before. Changes in parameter values are noted in this table. The change in the value of α from 0.05 to 0.2 for algorithms with mutation operation, is to keep the overall effort in solving the problem (in terms of the number of traffic assignment problems solved) at a comparable level to those without this operation so that they could be compared better. Four measures of performance are considered in Table 2, as in Fig. 4. These are (a) average no. of assignments, (b) frequency of finding the optimal solution, (c) average value of the objective function of the solutions, and (d) average value of the objective function for the new assignments. These information are for 50 runs of each algorithm, and they are given for each 8 iterations of the algorithms, and in total. The hybrid

Table 2
Effects of various improvements upon the performance of the algorithms (50 runs)

Algorithm	Base	Hybrids with improvement (Imp)								
Algorithm ID	0	1	2[a]	3		4[a]	5[a]	6	7[a]	Ave.
				3-1[b]	3-2[b]					
Measure 1: Ave. no. of traffic assignments										
Iteration 1	8.6	8.7	8.6	8.7	8.9	8.8	8.7	8.4	8.7	8.7
Iteration 2	5.3	4.8	4.2	6.0	5.4	3.6	3.8	5.3	3.9	4.7
Iteration 3	3.4	3.4	1.8	3.9	3.9	1.8	1.6	3.5	2.0	2.8
Iteration 4	2.1	2.4	0.9	2.7	1.7	0.6	0.9	1.1	0.7	1.4
Iteration 5	2.2	1.9	10.0	1.3	1.3	8.3	10.0	1.0	8.4	4.9
Iteration 6	1.6	1.3	0.8	0.7	1.0	0.2	0.8	0.8	0.2	0.8
Iteration 7	1.2	1.3	0.4	0.6	0.7	0.1	0.5	0.7	0.2	0.6
Iteration 8	0.8	0.8	0.3	0.5	0.7	0.2	0.4	0.4	0.1	0.5
All iterations	25.1	24.8	27.0	24.5	23.6	23.6	26.7	21.2	24.2	24.5
Measure 2: Frequency of finding the optimal solution										
Iteration 1	9	11	15	8	11	12	13	10	12	11.2
Iteration 2	12	7	9	5	6	8	10	10	12	8.8
Iteration 3	7	3	3	6	5	4	4	10	2	4.9
Iteration 4	4	3	1	9	6	2	2	4	1	3.6
Iteration 5	4	7	18	5	6	20	15	4	20	11.0
Iteration 6	4	4	0	3	6	0	1	2	0	2.2
Iteration 7	1	5	0	3	3	0	0	4	1	1.9
Iteration 8	0	2	0	2	1	0	0	0	0	0.6
All iterations	41	42	46	41	44	46	45	44	48	44.1
Measure 3: Ave. value of the objective function										
Iteration 1	65,133	65,181	65,035	65,021	65,138	65,167	65,055	65,139	65,095	65,107
Iteration 2	64,903	64,965	64,447	65,204	64,925	64,700	64,387	65,070	64,595	64,800
Iteration 3	64,758	64,757	64,296	64,973	64,903	64,366	64,239	64,920	64,330	64,616
Iteration 4	64,663	64,789	64,255	64,724	64,847	64,130	64,046	64,506	64,197	64,462
Iteration 5	64,750	64,749	66,007	64,739	64,765	65,837	65,949	64,392	65,631	65,202
Iteration 6	64,693	64,677	63,970	64,730	64,696	63,913	63,873	64,441	63,851	64,316
Iteration 7	64,794	64,763	63,912	64,559	64,746	63,972	63,917	64,446	63,937	64,338
Iteration 8	64821	64,750	63,986	64,799	64,583	63,960	63,929	64,431	63,924	64,354
All iterations	64,814	64,829	64,489	64,844	64,825	64,506	64,424	64,668	64,445	64,649
Measure 4: Ave. value of the objective function for new assignments										
Iteration 1	65,235	65,310	65,205	65,152	65,249	65,289	65,211	65,253	65,267	65,241
Iteration 2	65,480	65,573	65,061	65,911	65,540	65,399	65,040	65,355	65,154	65,390
Iteration 3	65,606	65,731	65,864	65,792	65,995	65,288	65,436	65,872	65,823	65,712
Iteration 4	65,810	65,727	66,523	65,213	64,788	65,026	65,489	64,769	65,285	65,403
Iteration 5	65,944	66,294	66,007	65,308	65,049	65,837	65,949	65,086	65,631	65,678
Iteration 6	66,085	65,957	66,411	65,017	64,998	66,475	66,141	65,061	64,920	65,674
Iteration 7	66,726	65,881	66,379	64,445	65,078	65,454	67,883	65,024	64,297	65,685
Iteration 8	66,254	66,160	66,954	64,821	65,324	66,465	66,944	64,929	67,211	66,118
All iterations	65,893	65,829	66,051	65,207	65,253	65,654	66,012	65,169	65,449	65,613

$B/C = 8330/13{,}320 = 0.625$, $\alpha = 0.05$, $\beta = 1$.
[a] $\alpha = 0.2$.
[b] Initial temperature, $c(0)$, for Imp.3 has been set as (3-1) $c(0) = 20{,}000$, and (3-2) $c(0) = 2$.

algorithm under the heading of the Imp.3 has been run for the initial temperature of (3-1) $c(0) = 20{,}000$, and (3-2) $c(0) = 2$. The latter version of algorithm 3 has been hybridized with the other concepts in algorithms 4, 6, and 7.

As may be seen in Table 2, except for the iteration 5 of the hybrids with Imp.2 that involve mutation and the respective UE flow computations for the top solutions of the previous iterations, other hybrids experience decreasing no. of traffic assignments as iteration no. increases. Fig. 5 shows the performance of the hybrid algorithms in the space of effort-accuracy, where effort is measured by the no. of traffic assignments, and accuracy is measured

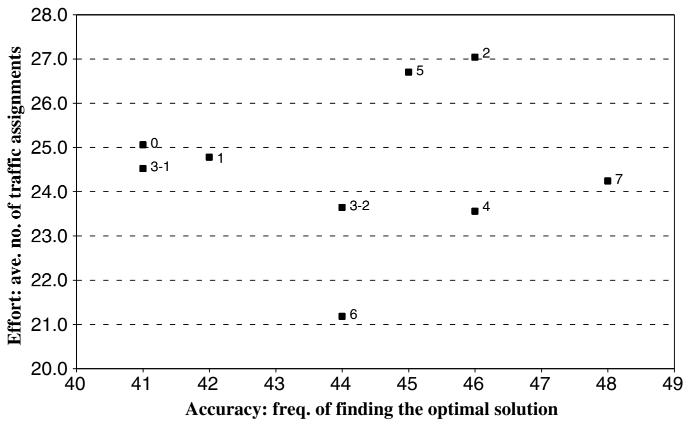

Fig. 5. Performance of the proposed hybrid algorithms in the space of effort-accuracy in 50 runs.

by the frequency of finding the optimal solution (in 50 runs). According to Fig. 5, hybrid 6 (Imp.6) has had the least average effort (21.2), and hybrid 7 (Imp.7) the highest accuracy (48). This accuracy (48 out of 50) at a comparable effort (24.2 as compared with 24.5, the average effort of all alternative algorithms) seems a remarkable performance. This is particularly so, if one notes that algorithm 7 reaches almost this accuracy level after iteration 5. This phenomenon happens in algorithm 7, 5, 4, and 2, all having Imp.2 in common. That is, the mutation operation seems to be very effective in identifying the optimal solution of the problem. Other concepts seem to have had marginal effects, when compared to the mutation of inferior genes. It is worth noting that Ahmad (1983) has also exploited similar mechanism and found it effective in an integer programming context. The average value of the objective function for the new assignments in Table 2 generally rises for algorithms 2, 4, 5, and 7 (having Imp.2), but fluctuates for the other algorithms. The average values of the objective function for the four hybrid algorithms with Imp.2 are considerably lower than these values for the other four algorithms without this improvement, as Table 2 shows. Never-the-less, algorithm 5 (without Imp.3) shows a little lower average value for the objective function in 50 runs than algorithm 7.

Table 3 summarizes the results of Table 2 and compares these figures with the respective ones for B/C ratio of 0.45 for the algorithms 0 (base), 1 (Imp.1), 2 (Imp.2), 3-2 (the better version of Imp.3 in Table 2), and 7 (the superior hybrid algorithm in Table 2). As may be seen in Table 3, all five algorithms for $B/C = 0.45$ happen to find the optimal solutions in all 50 runs. In this case, algorithm 3-2 has done this with much lower effort than others. It is worth noting that the number of feasible alternative networks for B/C of 0.625 and 0.45 are 781 and 398, respectively. The average number of assignments for the five algorithms in Table 3 for

Table 3
The performance of selected algorithms for two medium budget levels

Algorithm	B/C	α	Average number of assignments, all iterations	Frequency of finding the optimal solution	Average value of the objective function for the solution
0	0.45	0.05	25.9	50	66,650
1	0.45	0.05	26.6	50	66,650
2	0.45	0.2	31.6	50	66,650
3-2	0.45	0.05	20.8	50	66,650
7	0.45	0.2	29.4	50	66,650
0	0.625	0.05	25.1	41	61,532
1	0.625	0.05	24.8	42	61,525
2	0.625	0.2	27.0	46	61,477
3-2	0.625	0.05	23.7	47	61,476
7	0.625	0.2	24.2	48	61,469

the 50 runs are 24.96 and 26.86, respectively, which are 3.2% and 6.7% of the total feasible networks at the respective budget levels.

5. Application of algorithms to a real case

The next case under study is the network of the City of Mashhad, Iran, with a population of about 2.3 million in 2005. The forecasted population for the year 2021 is about 3.0 million. The network under study has 1298 nodes and 1726 links, and the overall vehicle-km in morning peak hour for the target year is about 1,450,000 (vehicles are in pce). Total vehicle-hours spent to cross the intersections in this city is about 12000, which is around 20% of this measure in the overall network. The problem here is to choose a subset of a given set of intersections to be upgraded into interchanges, so as to minimize the total user travel time for a given budget.

Projects in this case relate to the investments in a set of 26 intersections, which are identified based on several criteria extracted from expert opinions, MUTCD warrants, HCM analyses, analyses of changes in signal settings (phases, timing, turn prohibitions, etc.). The network, and the candidate projects (intersections) are shown in Fig. 6. Each project here is a set of links and nodes representing an interchange, which replaces another set of links and nodes representing the current state of the intersection, and effectively reduce the delay at the intersection to zero. The construction cost of the projects are assumed to be equal, so that budget would be equivalent to the total number of projects intended to be constructed. The algorithms, written in Visual Basic, are linked to EMME/2 for the traffic assignment routine. The parameters of the algorithms are as specified before, except for α which needs recalibration because of the changes in the scale of the network and demand. The followings are some of the results obtained from the experiments done for this case.

Table 4 compares the performance of the base ant system algorithm with the best hybrid algorithm (Hybrid 7). As is shown in this table, these two algorithms are run for some medium budget levels of 8, 12, and 17 out of 26. The parameter α in this case is set about 1/10 of those for the network of Sioux Falls to suit the new order of the value of the objective function in this case. Table 4 shows that for lower (than 0.005) values of α, Hybrid 7 performs better than ant system and with higher accuracy in terms of the frequency of finding the best solution found so far, and the frequency of finding the top 10 solutions found so far.

Table 5 shows the effect of the number of candidate projects and the budget level on the efficiency of the base and Hybrid 7 algorithms. In preparing this table parameter α, and in one case the number of iterations of the algorithm, have been adjusted so as to obtain a comparable solution by the two algorithms for the specified number of candidate

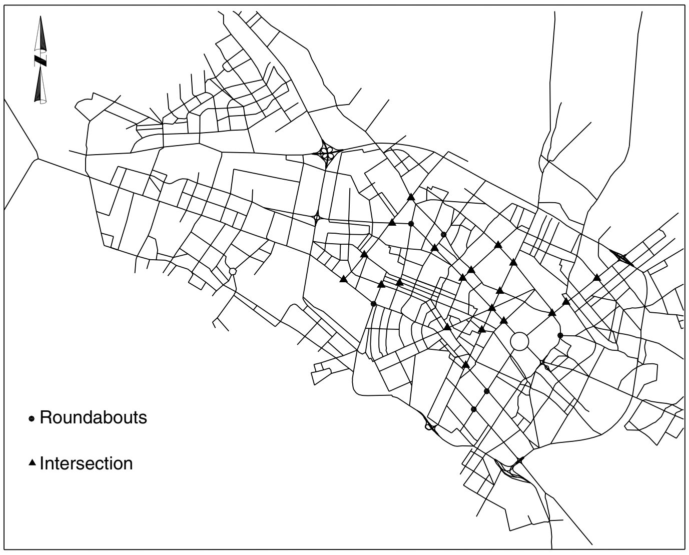

Fig. 6. The network of the City of Mashhad and the candidate intersection projects.

Table 4
Comparison of ant system and the better hybrid algorithm at various budget levels

Row	Algorithm	Budget	Number of runs	α	β	$C(0)$	Number of iterations	CPU time (minutes)	Average number of assignments	Frequency of finding the best solution found	Frequency of finding top 10 solutions found
1	AS	8	2	0.005	1	–	20	23	77	2	8.5
2	AS	12	10	0.005	1	–	200	37	91.1	2	1.5
3	AS	12	10	0.001	1	–	20	71	245.3	4	3.3
4	AS	17	2	0.003	1	–	200	75	245.3	0	1
5	AS	17	2	0.001	1	–	20	83	290	1	8.5
6	H7	8	2	0.005	1	20	15	53	184.5	2	9
7	H7	12	10	0.005	1	20	20	48	165.3	10	8.1
8	H7	12	10	0.02	1	20	20	37	114	10	9
9	H7	17	2	0.003	1	20	20	61	209	1	9.5

Table 5
Effect of the number of projects and the budget level on the efficiency of the base and hybrid 7 algorithms

No. of candidate projects (N)	Algorithms	α	Budget	Possible permutations of budget level	Average number of network assignments
10	AS	0.005	3	120	54.5
	AS	0.005	5	252	60.5
	AS	0.005	7	120	40.0
10	H7	0.02	3	120	52.0
	H7	0.02	5	252	49.0
	H7	0.02	7	120	37.0
17	AS	0.003	5	6188	62.0
	AS	0.003	9	24,310	137.0
	AS	0.003	12	6188	57.0
17	H7	0.02	5	6188	78.0
	H7	0.02	9	24,310	97.0
	H7	0.02	12	6188	52.0
26	AS	0.005	8	1,562,275	77.0
	AS	0.001	12	9,657,700	873[a]
	AS	0.001	17	3,124,550	290.0
26	H7	0.02	8	1,562,275	83.0
	H7	0.02	12	9,657,700	114.5
	H7	0.02	17	3,124,550	97.0

[a] For 200 iterations of AS algorithm instead of 20 in other runs in the stable in order to obtain a solution comparable to those of H7.

projects and budget level. The first column in Table 5 from right is the average number of network assignments in 20 runs of the algorithm (except in one case as mentioned in the table). These figures may be compared with the corresponding ones in column 2 from right in Table 5, which contains the number of budget level permutations of the projects, in order to find the level of efforts of each algorithm to find the solution. Fig. 7 shows the results of Table 5 pictorially. As may be seen in this figure Hybrid 7 algorithm generally outperforms ant system algorithm in all budget levels and number of candidate projects, yet finding better solutions (according to Table 5). It seems that the reduction in the number of assignments lends itself mostly to the SA part of the Hybrid 7 algorithm than other aspects of this algorithm. Moreover, it is worth noting that as the size of the problem increases the performance of Hybrid 7 algorithm increases too, which seems to be an encouraging result for the real-size problems.

Before passing, it might be useful to present the results of another comparative study. In another experiment the ant system algorithm was compared to a genetic algorithm to solve the same problem. The budget level has been changed from 6 to 16

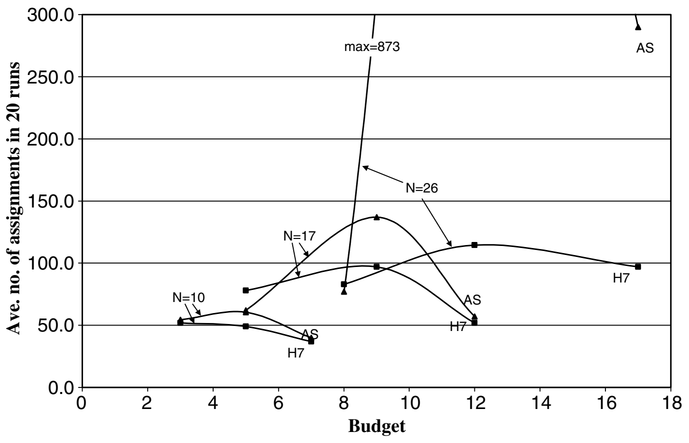

Fig. 7. Effect of the number of projects (N) and the budget level on the efficiency of ant system (AS) and hybrid 7 (H7) algorithms.

(projects), a range suitable for the network of Mashhad, the total number of projects was 26. Table 6 shows the results of this experiment. The projects shown on the top of the table are those which have been chosen by either the AS or the GA, at least once, in the course of the experiment for various levels of budget. The table also shows the solution found by either algorithm for different budget levels, and the respective value of the design objective function in vehicle-hour. The "1" in the table shows the acceptance of the project on top of the column by the algorithm on the left of the row, of the corresponding entry of the table. As may be seen in the table both algorithms have many projects in common for the solutions found for a specific budget level. However, the following points are notable in Table 6: (a) AS has consistently found better solutions (with better value of the objective function) than GA for all budget levels. (b) The projects found by AS for various budget levels are more stable than GA which diversifies the solutions. (c) AS solution for a budget level contained the solution found for the previous (lower) budget level, while GA basically diversified its solution. Moreover, the cpu times for the GA was much more than those of AS to reach to the solutions presented. This is the reason for not running GA for all budget levels in Table 6. These observations do not necessarily imply that AS is better than GA in all circumstances, or that GA may not be equipped with some tools to perform better. However, it seems that for the nature of the problem at hand (the network design problem) AS is a more suitable tool than GA (which might perform better for some other problems). It seems that flows of ants over the graph as directed by the flow of traffic over the network are two conformable processes, and that the latter is not as effective when directing the GA process. Thus, it might be more evident now that the hybrid 7 algorithm outperforms the GA-type algorithm for the NDP.

6. Summary and conclusions

Network design problem has been, and is, an important problem in transportation. This paper classifies some of the existing algorithms for the solution of this problem. Following an earlier effort in the solution of this problem by ant system (Poorzahedy and Abulghasemi, 2005), this paper attempts to improve the power of the earlier algorithm by hybridizing it with some of the leading concepts in search processes, such as genetic algorithm, simulated annealing, and tabu search. These algorithms are briefly described, and previous efforts in hybridizing them are mentioned.

The paper, then, describes three improvements based on the above-mentioned heuristics. Seven hybrid algorithms have been devised based on the base ant algorithm and these three improvement concepts.

These seven algorithms have been applied on the test network of Sioux Falls, South Dakota, USA. It has been shown by various experiments on this network that all three improvement concepts are effective ideas in expediting the base ant system solution procedure.

The experiments on the network of Sioux Falls showed the variation of some of the performance

Table 6
The results of a comparison of the base AS with a GA algorithm to the case of Mashhad

Budget level[a]	Algorithm	Project identification number																						Value of objective function
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	
6	AS	1	1	1	1	1	1																	69,070
7	GA	1	1	1	1	1	1					1												68,932
	AS	1	1	1	1	1	1	1																68,841
8	AS	1	1	1	1	1	1	1	1															68,679
9	GA	1	1		1	1	1					1	1	1				1						68,943
	AS	1	1	1	1	1	1	1	1	1														68,532
10	AS	1	1	1	1	1	1	1	1	1	1													68,344
11	GA				1	1							1	1		1		1	1	1	1	1	1	68,324
	AS	1	1	1	1	1	1	1	1	1	1	1												68,202
12	GA	1	1	1	1	1	1	1	1	1	1		1					1						68,055
	AS	1	1	1	1	1	1	1	1	1	1	1	1											68,039
13	GA	1	1	1		1	1	1	1	1	1	1	1	1										68,065
	AS	1	1	1	1	1	1	1	1	1	1	1	1	1										67,907
14	AS	1	1	1	1	1	1	1	1	1	1	1	1	1	1									67,835
15	GA	1	1	1	1	1	1	1	1	1		1	1	1				1	1	1				67,924
	AS	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1								67,720
16	AS	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1							67,641

measures of the base algorithm. More specifically, this paper has reconfirmed that:

(a) The number of traffic assignments is higher when the "budget/total project cost" ratio is around the mid-value of the total construction cost of all projects.
(b) Higher value of α (pheromone parameter) results lower number of assignments, which suggests use of lower α values for low or high B/C levels.
(c) The goodness of project j (v_j) has a profound effect upon the frequency of finding the solution of the problem, and the reduction of the number of assignments in the design problem.
(d) Fewer traffic assignment problems are solved as the iteration no. of the algorithm increases.
(e) The frequency of finding the optimal solution in the first few iterations of the algorithm is rather high.
(f) Increase of α increases the rate of convergence of the algorithm while increasing the chance of getting trapped at the local minima.
(g) The mutation operation (Imp.3) seems to be very effective in identifying the optimal solution.
(h) All hybrid algorithms have shown better performance than the base Ant System.

The proposed algorithms have been applied to solve a design problem in a real-size network of the City of Mashhad, Iran. It has been shown that the hybrid algorithm comprising all concepts of AS, SA, GA, and TS perform better than the AS alone. Never-the-less, it seems that for simpler problems (in concept, formulation, complexity, and size) there is no need to implement a more complex algorithm like Hybrid 7. The base AS would do the job. The cost of using Hybrid 7 algorithm is justified when more complex problems are at hand.

Acknowledgement

The authors would like to thank the Institute for Transportation Studies and Research, at Sharif University of Technology, Iran, for its continued support of this study.

References

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

Ahmad, N.U., 1983. An analytical decision model for resource allocation in highway maintenance management. Transportation Research A 17A (2), 133–138.

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., Heuristics for the network design problem. In: Ewg 2002 (the 13th Mini Euro Conference), Bari, Italy.

Chelouah, R., Siarry, P., 2003. Genetic and Nelder-Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. European Journal of Operational Research 148, 335–348.

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

Costa, D., 1995. An evolutionary tabu search algorithm and the NHL scheduling problem. INFORM 33, 161–178.

Cree, N.D., Maher, M.J., Paechter, B., 1998. The continuous equilibrium optimal network design problem: A genetic approach. In: Bell, M.G.H. (Eds.), Transportation Networks: Recent Methodological Advances. Selected Proceedings of the 4th Euro Transportation Meeting, pp. 163–174.

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 13B, 5–17.

De Werra, D., Hertz, A., 1989. Tabu: A Application to Neural Networks. OR Spectrum 11, 131–141.

Dorigo, M., Gambardella, L.M., 1997. Ant colonies for the traveling salesman problem. BioSystems 43, 73–81.

Dorigo, M., Maniezzo, V., Colorni, A., 1991. The ant system: An autocatalytic optimization process. Technical Report TR91-016, Politecnico Di Milano.

Dorigo, M., Maniezzo, V., Colorni, A., 1996. The ant system: Optimization by a colony of cooperating agent. IEEE Transactions on Systems, Man and Cybernetics, Part B 26 (1), 1–13.

Fleurent, C., Ferland, J., 1999. Genetic Hybrids for the quadratic assignment problem. DIMACS Series in Mathematics Theoretical Computer Science 16, 190–206.

Friesz, T.L., Cho, H.J., Mehta, N.J., Tobin, R.L., Anadalingam, G., 1992. A simulated annealing approach to the network design problem with variational inequality constraints. Transportation Science 26 (1), 18–26.

Glover, F., Kelly, J.P., Laguna, M., 1995. Genetic algorithms and tabu search: Hybrids for optimization. Computer & Operations Research 22 (1), 111–134.

Goldberg, D.E., 1989. Genetic Algorithm Optimization. Addison-Wesley Publishing Company, Reading, Mass.

Greistorfer, P., 1998. Hybrid genetic tabu search for cyclic scheduling problem. In: Meta-Heuristics Advances and Trends in Local Search Paradigms for Optimization. Klower, Boston.

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

Hertz, A., Widmer, M., 2003. Guidelines for the use of metaheuristics in combinatorial optimization. European Journal of Operational Research 151, 247–252.

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

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

Kim, H., Nara, K., Gen, M., 1994. A method for maintenance scheduling using GA combined with SA. Computers Industrial Engineering 27, 477–486.

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), 1–55.

Maniezzo, V., Gambardella, L.M., Deluigi, F., 2003. Ant colony optimization. In: Onwubolu, G.C., Babu, B.V. (Eds.), New Optimization Techniques in Engineering. Springer-Verlag (www.idsia.ch\~luca\ACO.2004.pdf).

Onwubolu, G.C., 2002. Emerging Optimization Techniques in Production Planning and Control. Imperial College Press, London.

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

Poorzahedy, H., Turnquist, M.A., 1982. Approximate Algorithms for the Discrete Network Design Problem. Transportation Research, Part B 16, 45–56.

Resende, M.G.C., Pardalos, P.M., Eksioglu, S.D., 1999. Parallel Metaheuristics for Combinatorial Optimization, www.research.att.com/~mgcr/doc/parheur.pdf.

Sadek, A.W., 2001. A hybrid simulated annealing and case-based reasoning approach for computationally intensive transportation problem: Rationale and design issues. In: TRB, 80th Annual Meeting, Washington, DC.

Salmon, P., Sibani, P., Frost, R., 2002. Facts, Conjectures and Improvements for Simulated Annealing. SIAM Society of Industrial & Applied Mathematics, Philadelphia, PA.

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, NewYork.

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

Taillard, E.D., Gambardella, L.M., 1997. Adaptive memories for the quadratic assignment problem. Technical Report IDSIA-79-97, Lugano, Switzerland.

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. Journal of Transportation Engineering: ASCE 26 (2), 115–120.