An efficient hybrid heuristic method for prioritising large transportation projects with interdependent activities

Saeed A. Bagloee

Parsons Overseas Limited,
Gulf Tower-B, Oud Metha Road,
Dubai, UAE
Fax: +971-4-336-7920
E-mail: saeed.asadi@parsons.com

Madjid Tavana*

Management Information Systems,
Lindback Distinguished Chair of Information Systems,
La Salle University,
Philadelphia, PA19141, USA
Fax: +1.267.295.2854
E-mail: tavana@lasalle.edu
Website: http://tavana.us
*Corresponding author

Abstract: Transportation projects are generally large, with limited resources and highly interdependent activities. The complexities and interdependencies apparent in large transportation projects have prohibited effective application of management science and economics methods to these problems. We propose a heuristic method with several hybrid components. We formulate the problem as a Travelling Salesman Problem (TSP). A Neural Network (NN) is used to cope with the interdependency concerns. An algorithm with an iterative process is confined to search for the longest path (most benefit or most reduction in the user-time) in the NN as a solution to the TSP. The solution from each iteration step is utilised to update and train the NN and enhance its prediction. A search engine inspired by the concept of Ant Colony (AC) and hybridised with Genetic Algorithm (GA) is developed to find a suitable solution to the TSP. The hybrid heuristic method proposed in this study is applied to the real data for the city of Winnipeg in Canada to demonstrate the applicability of the proposed framework and exhibit the efficacy of the procedures and algorithms.

Keywords: TSP; travelling salesman problem; hybrid heuristic; NN; neural network; ant colony; GA; genetic algorithm; project prioritisation.

Reference to this paper should be made as follows: Bagloee, S.A. and Tavana, M. (2012) 'An efficient hybrid heuristic method for prioritising large transportation projects with interdependent activities', *Int. J. Logistics Systems and Management*, Vol. 11, No. 1, pp.114–142.

Biographical notes: Saeed Asadi Bagloee is a Senior Transportation Planner and lead modeller at Parsons Overseas Limited in Dubai, UAE. He received his BS in Civil Engineering from Khaje Nasir Toosi University of Technology and his MS in Transportation Planning and Engineering from Sharif University of Technology in Tehran, Iran. His research interest is transportation planning and modelling including travel demand forecasting, traffic impact studies, road network design and transit planning and economic analysis of transportation projects. He has conducted extensive research at the Institute for Transportation Studies and Research at Sharif University of Technology and has presented and published his research in several international conferences and proceedings.

Madjid Tavana is a Professor of Management Information Systems and Decision Sciences and the Lindback Distinguished Chair of Information Systems at La Salle University where he served as Chairman of the Management Department and Director of the Center for Technology and Management. He has been a Distinguished NASA Research Fellow at Kennedy Space Center, Johnson Space Center, Naval Research Laboratory – Stennis Space Center, and Air Force Research Laboratory. He was recently honoured with the prestigious Space Act Award by NASA. He holds an MBA, a PMIS and a PhD in Management Information Systems and received his post-doctoral diploma in strategic information systems from the Wharton School of the University of Pennsylvania. He is the Editor-in-Chief for the *International Journal of Strategic Decision Sciences*, the *International Journal of Enterprise Information Systems* and the *International Journal of Applied Decision Sciences*. He has published over 70 scholarly research papers in academic journals such as *Decision Sciences*, *Information Systems*, *Interfaces*, *Annals of Operations Research*, *Omega*, *Information and Management*, *Expert Systems with Applications*, *European Journal of Operational Research*, *Journal of the Operational Research Society*, *Computers and Operations Research*, *Knowledge Management Research and Practice*, *Computers and Industrial Engineering*, *Applied Soft Computing*, *Journal of Advanced Manufacturing Technology*, and *Advances in Engineering Software*, among others.

1 Introduction

An efficient transportation system is a prerequisite for economic growth and development (Adler, 1987). A transportation system is a highly complex infrastructure with a wide array of interdependent activities. When confronted by the range of transportation infrastructure projects, organisations struggle to identify those most appropriate to their needs. Tsamboulas (2007) argues that a coherent, well-structured and flexible method is needed to evaluate transportation infrastructure projects. The review of literature provides a wide range of methods used to evaluate transportation infrastructure projects with contradictory criteria such as minimising construction costs and environmental impacts (Giorgi and Tandon, 2002; Hartgen and Neuman, 2002; Iniestra and Gutiérrez, 2009; Leleur, 1995; Little and Mirlees, 1994; Mackie and Preston, 1998; Sinha and Zongzhi, 2004; Tsamboulas, 2007; Tsamboulas et al., 1999). We present an efficient hybrid heuristic method for managing large transportation projects with interdependent activities. We formulate a large transportation project as a TSP and propose a heuristic method with several hybrid components. A NN is used to cope with the interdependency

concerns and a search engine inspired by the concept of Ant Colony (AC) is developed to find a suitable solution to the TSP.

Due to increase in travel demand, providing new civil transportation facilities alongside demand management policies is inevitable. In this study, we address the prioritisation of the road construction projects involving capacity expansion and new road construction activities. This study is motivated by the need to manage the budget and completion time for five simultaneous road construction projects undertaken by the municipal government of a notoriously congested city in the Middle East. While the overall goal in this study was to identify a construction priority plan regardless of construction time variation, the proposed framework is also applicable to emergency situations such as natural disastrous, military or terrorist attack where the damaged transportation network has to be repaired in a shortest period of time possible. In emergency management situations, there are no concerns for variation of travel demand since some temporarily and quick-to-build projects are initiated to alleviate the immediate concerns.

Generally, the customers in a transportation system are the commuters (referred to as users hereafter) and the PPPs impact the traffic flow. Therefore, the prioritisation benefits can be represented with users' cost saving represented by an overall measure representing a variety of factors such as travel time, safety and pollution. Without losing generality of discussion for simplicity, we refer to this measure as travel time. Obviously, the users in a transportation network choose paths of links (roads) with least travel time. To identify travel time on each road and thus on the shortest path, each road is associated with a non-decreasing congestion function of traffic flow. The traffic assignment modules in transportation networks yield an equilibrium traffic flow across the network wherein every user chooses shortest path and nobody can make a shortcut unilaterally. As such, the desired path to a user is highly sensitive even to tiny changes in the road network especially in a congested area. Therefore, the network-wide benefits for solutions involving road expansion (adding more traffic lanes) and new road construction are highly interdependent. Different project combinations generally yield different traffic flows. Therefore, as the number of projects increases, determining the benefits associated with different project combinations becomes more and more difficult making traditional evaluation methods such as cost benefit analysis impractical.

The PPP may be considered as a general Network Design Problem (NDP) known as a bi-level optimisation problem (Yang and Bell, 1998). An NDP has two layers: traffic flow prediction (traffic assignment) and selecting best combination of the candidate projects with respect to some budget constraints. A third layer may define the priority of the best selected projects in the preceding level. Earlier studies have ignored the interdependency among the transportation projects. Nemhauser and Ullmann (1969) have defined project interdependency as a pair-wise interaction using a dynamic programming model. Gear and Cowie (1980) and Fox et al. (1984) proposed an extension of the Nemhauser and Ullmann's (1969) model to higher-order interactions. Weingartner (1966), Cochran et al. (1971) and Johnson et al. (1985) used integer programming to model projects with interdependent activities. In general, these procedures are applicable to problems with a relatively small number of projects and simple or easy to define interdependencies (Sandhu, 2006).

Janson et al. (1991) proposed a heuristic approach and a trip distribution model. Although their methodology could generate good solutions for simple networks, the applicability of their procedure for complex networks remains to be investigated. Jong and Schonfeld (2001) used a GA method and evaluated project interactions by adapting a simple approximation model estimated from simulation results, but with lower precision than a simulation model. Tao and Schonfeld (2006) developed an island model (a variations of GA), while relying on equilibrium traffic assignment model to evaluate the impacts of transportation projects. They studied a network with eight nodes but they neither discussed the efficiency of their algorithm nor mentioned the goodness of their solution. Tao and Schonfeld (2007) emulated the stochastic nature of the transportation systems. While their proposed formulation was simple and intuitive, they did not take into consideration the calibration of their model and the efficiency of their methodology to real world problems remain to be seen. In summary:

- The large transportation projects with interdependent activities are difficult to manage.
- The earlier heuristic methods proposed in the literature were either customised to some specific cases or were not applicable to real-world problems. At best, they indirectly addressed the interdependency concern among the projects.
- The current methods directly take the interdependency concern into account where a traffic assignment model is inevitable. In these methods, heuristic methods proposed do not tackle large real world problems.

Recent natural and human caused disasters present new challenges for transportation systems. After the terrorist attack on the World Trade Center in 2001 and hurricane Katrina in 2005, several studies have proposed appropriate network measures to rank the importance of the network components (Murray-Tuite and Mahmassani, 2004; Qiang and Nagurney, 2008; Sheffi, 2005). The focus of these studies is avoiding the high-ranked roads affected by special circumstances (i.e., emergency activities, natural disaster). Such ranking is established according to some characteristics used to determine the network vulnerabilities. These methods cannot be used in problems involving project combinations and/or interdependencies.

In this study, we propose a heuristic approach to solve real world PPPs as an exact method is not available due to computational hurdles. We consider the tradeoff between a good solution for a real-world problem and an exact solution for an out-of-this world problem. We formulate the problem as a TSP since any priority plan maps a trip through the projects (as the cities must be visited once in TSP). We also build an NN to cope with the interdependency among the projects. An iterative process is used to determine the Priority-Plan Scenario (PPS) of the TSP in each iteration. The PPS is analysed to improve the solution in the next iteration. A total benefit for each PPS is using traffic assignment modules. The assignment results are further used to update and train the NN to enhance its prediction power. The PPSs are encoded with GA. Finally, we hybridise a search engine (inspired by the concept of AC) with GA to determine the PPS solution to the TSP.

The rest of this paper is organised as follows. In Section 2, we briefly review the literature on TSP, GA, AC and NN. In Section 3, we define the mathematical formulation and the methodological foundations of the PPPs. In Section 4, we present the details of our proposed algorithms. In Section 5, we present a numerical example to exhibit the efficacy of the procedures and algorithms. The conclusions and future research directions are discussed in Section 6.

2 Literature review

The TSP is a non-deterministic polynomial-time complete (NP-complete) problem where the goal is to find a complete route with a minimal cost (Lawler et al., 1985). Several optimisation methods have been developed to solve the TSPs (Adachi and Yoshida, 1995; Baraglia et al., 2001; Chien and Chen, 2009; Ellabib et al., 2007; Fiechter, 1994; Freisleben and Merz, 1996; Hopfield and Tank, 1985a; Kirkpatrick et al., 1983; Liu and Zeng, 2009; Martin and Otto, 1996; Naimi and Taherinejad, 2009; Saadatmand-Tarzjan et al., 2007; Xie and Liu, 2009; Yi et al., 2008). Kirkpatrick et al. (1983) proposed a simulated annealing method to solve the TSP by simulating the annealing action in metallurgy. Martin and Otto (1996) used the simulated annealing in conjunction with the local heuristics to solve the TSP. Fiechter (1994) proposed a tabu searching method with a parallelised mechanism to solve the TSP. Hopfield and Tank (1985a, 1985b) proposed an NN method to solve the TSP by mimicking the properties of biological neurons. Saadatmand-Tarzjan et al. (2007) introduced a novel constructive-optimiser NN and Yi et al. (2008) proposed a fast elastic NN method for solving the TSPs.

Adachi and Yoshida (1995) proposed the concept of protected chromosomes to reduce the searching space and speed up the processing time of GA for solving the TSP. GA, capable of searching the domain space globally, has been widely used to solve large-scale combinational optimisation problems. The GA can produce good results as long as the fitness function is determined. Freisleben and Merz (1996) proposed a hybrid method combining the GA and local search heuristics to find a near-optimum solution for TSPs. Baraglia et al. (2001) proposed a hybrid heuristic GA and Liu and Zeng (2009) proposed a GA method with reinforcement learning to solve the TSPs. Ellabib et al. (2007) proposed a method with exchange strategies for multiple AC systems to solve the TSP. Naimi and Taherinejad (2009) proposed an AC algorithm using a local updating process. Chien and Chen (2009) introduced a parallelised genetic AC method and Xie and Liu (2009) proposed a multi-agent optimisation method for solving the TSPs.

AC optimisation is the most popular method for solving the TSPs. AC optimisation is a cooperative population-based searching algorithm inspired by the foraging behaviour of real ants. When real ants leave their nests to search for food, they form a route based on the quantities of the pheromone on each edge they pass through. Each ant deposits the pheromone on each travelled edge such that the pheromone level of the travelled edges is increased and the pheromone level on the unvisited edges is decayed. The other ants search their own routes according to their experiences based on the pheromone levels deposited on the edges that they choose to travel. Edges with higher pheromone levels are more attractive to the ants. As more ants pass through an edge, more pheromones are deposited on an edge. This searching procedure is used in AC optimisation to solve TSPs.

The original algorithm of AC, known as the ant system, was proposed by Dorigo (1992) to solve the TSP. Several algorithms have been developed based on the idea of updating the pheromone information to search the shortest route including the AC system (Dorigo and Gambardella, 1997a, 1997b), the attribute-based ant system (Yang and Wu, 2009), the multi-objective AC system (Yagmahan and Yenisey, 2010), the MAX–MIN ant system (Ning et al., 2010; Stützle and Hoos, 1996, 2000), the rank-based ant systems (Bullnheimer et al., 1997) and KCC- and ELU-Ants (Naimi and Taherinejad, 2009).

GAs are metaheuristics inspired by the efficiency of natural selection in biological evolution. GA has been widely used to solve a wide range of combinatorial optimisation problems (Lin and Gen, 2008; Man et al., 1999; Winter et al., 1995; Wright et al., 2005; Zalzala and Fleming, 1997). Unlike traditional heuristics (and some metaheuristics like tabu search) where a single solution is generated and further improved, GAs perform comparatively little work on a large number of solutions called the population. Each member of the population (referred to as a chromosome) is an encoded version of a solution. The goal of encoding is to translate a solution into a string of genes that make up the chromosome, similar to biological genetics. Each iteration step in a GA consists of several operators that form a new generation of solutions from the old one. The new solution is designed to preserve the genetic material of the better solutions.

The most common GA operators are reproduction, crossover and mutation (Balamurugan et al., 2006; Noorul Haq and Kannan, 2006; Raj and Shahabudeen, 2006). In reproduction, the best solutions are copied from the previous generation into the next generation while preserving the very high-quality solutions. In crossover, two randomly chosen parents produce one or more offspring with some combination of the parents' genes. Crossover is performed in a deterministic manner (where genes appearing before a predetermined cutoff point comes from one parent and genes appearing after the cutoff point come from the other parent) or in a random manner (where genes come from one parent with a certain probability). In mutation, a few genes are randomly changed based on the evolutionary concept that random genetic mutations may produce superior offspring.

Several hybrid GA methods have been proposed for solving scheduling problems. In general, GA, hybridised with a local search scheme, acts as a global search scheme to enhance both diversification and intensification (Reeves, 1994). Gonazelez et al. (1995) hybridised the GA with three problem-oriented operators based on the heuristics developed by Gupta (1971), Palmer (1965) and Rajendran (1994). Park (2001) and Reeves (1995) embedded a greedy local optimisation algorithm in a hybrid GA to solve vehicle scheduling problems. Liaw (2000) hybridised the GA with the tabu search to solve the open shop scheduling problems. Gonçalves et al. (2005) hybridised GA the critical path method to solve job shop scheduling problems. Valls et al. (2008) proposed a hybrid GA to solve the resource-constrained project scheduling problems.

Hopfield and Tank (1985b) introduced the idea of using NN to solve non-deterministic polynomial-time hard (NP-hard) optimisation problems. Aiyer et al. (1990) introduced a subspace approach by using a single term to represent all constraints into an energy function. Peng et al. (1996) and Andresol et al. (1999) have improved the local optima of the Hopfield and Tank (1985b) in small-sized TSPs. However, it is computationally more expensive and difficult to find feasible solutions in large-sized and make the Hopfield and Tank (1985b) network comparable with meta-heuristics (Osman and Laporte, 1996).

Another promising NN approach for routing problems is based on the Kohonen's Self-Organising (Feature) Maps (SOFM) (Kohonen, 1982, 1995) where an unsupervised NN simply inspects the input patterns of arbitrary dimensions for regularities and patterns and converts them into the responses of a one- or two-dimensional array of neurons. Learning is achieved through adaptation of the weights connecting the input pattern to the array on neurons.

Durbin and Willshaw (1987) introduced deterministic elastic networks with no energy function and a series of nodes lying on an artificial 'elastic band' for solving the TSPs. The nodes are moved in the Euclidean space to minimise an energy function by stretching the elastic band. Fort (1988) introduced the applications of the SOFM to the TSPs by incorporating stochasticities and mapping one-dimensional circular array of neurons onto the TSP cities in such a way that two neighbouring points on the array are also neighbours in distance. Other researchers have combined features of both elastic network and SOFM to derive better algorithms for solving TSPs (Angeniol et al., 1988; Aras et al., 1999; Burke, 1994; Ghaziri and Osman, 2003; Vakhutinsky and Golden, 1995).

3 Project prioritisation problem

3.1 Mathematical formulation

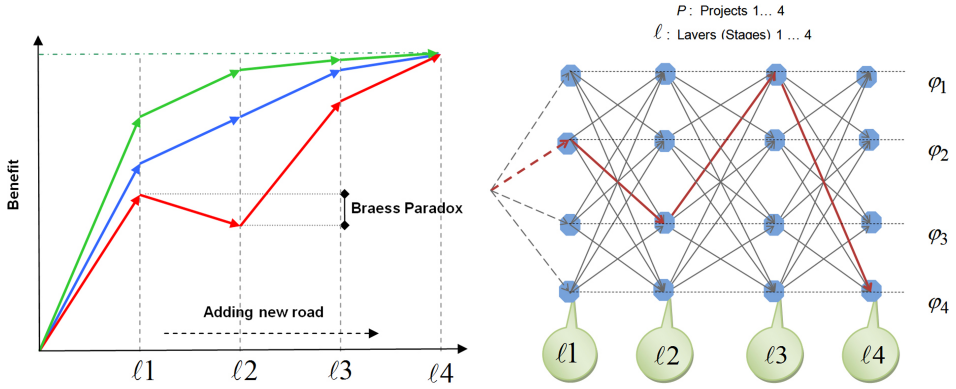
Consider a road network and a set of road constructions projects. The overall goal is to develop a PPS and maximise the overall benefits. A PPS is a string of integer and sequential numbers assigned to projects to indicate their priority on the construction list. In this model, we do not consider the construction time or the travel demand variations associated with the projects. The travel demand is assumed as a fixed and exogenous parameter. The total benefits is defined in terms of the reduction in the users' travel time compared to the 'do nothing' (no new project) scenario. The total benefits for each PPS is defined as the total sequential benefits of constructing the projects according to their priorities. Figure 1(a) demonstrates the process of implementing three different PPSs in which the cumulative benefits of sequential construction of four projects (coloured by green, blue and red) are shown. The horizontal axis shows the process of adding a new project to each stage (or layer) denoted by $\ell 1 \ldots \ell 4$ on the construction list on the basis of their priority.

As Figure 1(a) depicts, the area under the curve for each scenario is the total benefits of the respective scenario. As shown, in this figure, at stage 2 of the red scenario, despite adding a new street, the overall benefits topple. This (seemingly) counter-intuitive result is known as Braess paradox (Sheffi, 1985) in the transportation literature. Let us denote:

$G(V,E)$	The road network, a directed graph with V set of vertices or nodes or junctions and E set of edges or arcs or links or roads. Each edges is denoted by $(i,j) \in E$ where $i,j \in V$
Y	The set of construction projects to be prioritised $(Y \subset E)$. Y can represent new road construction projects as well as lane increasing (road widening) projects. For the road widening case, assuming that a road represented by $(i,j) \in Y \subset E$ already exists, in order not to violate directness requirement of the network in graph theory, the concerned arc (i,j) can be split to $(i-k-j)$, thus (i,j) would represent the road widening project

n	The number of projects to be prioritised $(n = \|Y\|)$
y_{ij}	An integer value (1, 2, 3, …, n) corresponding to the position of the project $(i,j) \in Y$ in a priority plan
O	The set of travel origins $(O \subseteq V)$
D	The set of travel destinations $(D \subseteq V)$
D_{od}	The hourly rate of vehicle travel demand from $o \in V$ to $d \in V$
Q	The set of origin-destination pairs $(o,d) \in O \times D$ with non-zero travel demand $(D_{od} > 0)$
t_{ij}	The travel time (minutes) on the road or the link $(i,j) \in E$
P_{od}	The set of paths from $o \in V$ to $d \in V$
t_{od}^{ℓ}	The least travel time (minutes) over a network denoted by ℓ from $o \in V$ to $d \in V$
$h_p^{\ell,od}, t_p^{\ell,od}$	The hourly traffic flow and travel time (minutes) respectively on the path $p \in P_{od}$ from $o \in V$ to $d \in V$ over a network denoted by ℓ
$t_{ij}(x_{ij})$	A positive and none-decreasing congestion function of traffic volume (x_{ij}) for each road of any sub-network $(i,j) \in X \subset E$

Figure 1 (a) The process of implementing three different PPSs and (b) the NN structure simulating the PPP problem as a TSP (see online version for colours)



(a) (b)

The total travel time incurred by users (T_X) is calculated as:

$$T_X = \sum_{(i,j) \in X \subset E} x_{ij} . t_{ij}(x_{ij}). \tag{1}$$

We use User Equilibrium (UE) traffic flow, the most widely recognised traffic flow model in transportation, to ensure that users choose their shortest path and no one unilaterally can make a shortcut. This is also called Nash equilibrium flow (Nash, 1951).

Based on the adopted definition of the benefit for each priority scenario under the condition of a UE traffic flow, the PPP may be written as follows:

$$\text{(PPP)} \quad \text{Max} \quad B(x_{ij}^{\ell}, \delta_{ij}^{\ell}) = \sum_{\ell=1}^{n} \left\{ T_{E-Y} - \sum_{(i,j) \in E} (x_{ij}^{\ell} . t_{ij}(x_{ij}^{\ell})) \right\} \tag{2.1}$$

s.t.:

$$\delta_{ij}^{\ell} \cdot (1-\delta_{ij}^{\ell}) = 0 \quad \forall (i,j) \in E, \ \ \ell = 1,2,...,n \tag{2.2}$$

$$\delta_{ij}^{\ell} = 1 \quad \forall (i,j) \in E - Y, \ \ \ell = 1,2,...,n \tag{2.3}$$

$$\sin(\pi \cdot y_{ij}) = 0 \quad \forall y_{ij} \in Y \tag{2.4}$$

$$M_{ij,rs} \cdot (y_{ij} - y_{rs}) > 0 \quad \forall (i,j),(r,s) \in Y, \ (i,j) \neq (r,s), \quad M_{ij,rs} \in \Re \tag{2.5}$$

$$\sum_{(i,j) \in Y} y_{ij} = n(n+1)/2 \tag{2.6}$$

$$\sum_{\ell=1}^{n} \delta_{ij}^{\ell} + y_{ij} = n+1 \quad \forall (i,j) \in Y \tag{2.7}$$

$$x_{ij}^{\ell} \cdot (1-\delta_{ij}^{\ell}) = 0 \quad \forall (i,j) \in E, \ \ \ell = 1,2,...,n \tag{2.8}$$

$$t_p^{\ell,od} - t_{od}^{\ell} \geq 0 \quad \forall p \in P_{od}, \quad \forall (o,d) \in Q, \quad \ell = 1,2,...,n \tag{2.9}$$

$$\sum_{p \in P_{od}} h_p^{\ell,od} = D_{od}, \quad \forall p \in P_{od}, \quad \forall (o,d) \in Q, \quad \ell = 1,2,...,n \tag{2.10}$$

$$h_p^{\ell,od} \geq 0, \quad \forall p \in P_{od}, \quad \forall (o,d) \in Q, \quad \ell = 1,2,...,n \tag{2.11}$$

$$x_{ij}^{\ell} = \sum_{\substack{p \in P_{od} \\ (o,d) \in Q}} h_p^{\ell,od} \cdot \mu_{ij,p}^{\ell,od} \quad \forall p \in P_{od}, \forall (o,d) \in Q, \forall (i,j) \in E, \quad \ell = 1,2,...,n, \tag{2.12}$$

$$h_p^{\ell,od} (t_p^{\ell,od} - t_{od}^{\ell}) = 0 \quad \forall p \in P_{od}, \quad \forall (o,d) \in Q, \quad \ell = 1,2,...,n. \tag{2.13}$$

Similar to the notation used in Figure 1(a), each stage or layer of adding a new road is denoted by ℓ and the number assigned to which $(\ell = 1,2,...,n)$ indicates how many new roads contributed to the initial network (the original network with no new project). Therefore, ℓ denotes the corresponding networks. In equation (2.1), the objective function is computed as the area under curve in Figure 1(a), in which $\sum_{\ell=1}^{n} \{\cdot\}$ calculates the total saved travel time experienced by the users on all networks corresponding to $\ell = 1 \ldots n$. In equation (2.1), T_{E-Y} is the total user travel time for the initial network and the last term $\left(\sum_{(i,j) \in E} (x_{ij}^{\ell} . t_{ij}(x_{ij}^{\ell}))\right)$, which computes the total user travel time of the network corresponding to ℓ. As set E encompasses all the roads including the new projects (Y), we delineate a sub-set of the links contributing to the network of ℓ by δ_{ij}^{ℓ}. In this regards, equation (2.2) assigns a binary value $(\delta_{ij}^{\ell} = 0 \ or \ 1)$ to all the links $((i,j) \in E)$, which means $\delta_{ij}^{\ell} = 1$ if project $(i,j) \in Y$ has contributed to the corresponding network of ℓ and $\delta_{ij}^{\ell} = 0$ otherwise. Equation (2.3) ensures the contribution of the initial networks' links to all the ℓ networks. In conjunction with δ_{ij}^{ℓ}, an integer variable (y_{ij}) is embedded into the formulation to indicate the priority of the projects $(y_{ij} = 1,2,...,n)$. This concept is enforced by equations (2.4)–(2.7): Equation (2.4) ensures that all the y_{ij} would be integer numbers. $M_{ij,rs}$, a none-zero real number in equation (2.5) does not

allow identical priority for (some) projects $(y_{ij} \neq y_{rs})$. Equation (2.6) along equations (2.4) and (2.5) guarantee a sequence of n integer numbers from 1 to n wherein $n(n+1)/2 = 1+2+...+n.$ The priority indices (y_{ij}) are present in the objective represented by δ_{ij}^{ℓ} and enforced by equation (2.7). According to the priority indices of $y_{ij}: 1, 2, ..., n,$ the total contribution number of the corresponding projects into the successive networks of ℓs are $\sum_{\ell=1}^{n} \delta_{ij}^{\ell}: n, n-1, ..., 1,$ respectively.

Therefore, in any circumstances, we have $\sum_{\ell=1}^{n} \delta_{ij}^{\ell} + y_{ij} = n+1.$ Equations (2.8)–(2.13) guarantee that the UE flow through all networks of ℓs wherein $\mu_{ij,p}^{\ell,od}$ is 1 if link (i, j) belongs to the path $P \in P_{od}$ on the corresponding network of ℓ, and 0 otherwise. Equations (2.2), (2.3), and (2.8) translate the concept of 'contribution' for the new project $((i, j) \in Y)$ to the carrying traffic flow $(x_{ij} > 0)$ and convey it to the objective function. If project $(i, j) \in Y$ does not exist in the network ℓ, then $\delta_{ij}^{\ell} = 0$ or $(1 - \delta_{ij}^{\ell}) \neq 0$ and there should be no traffic volume on the link $(x_{ij}^{\ell} = 0)$. The remaining constraints, equations (2.9)–(2.13), enforce a regular UE flow on the network of ℓ (see Sheffi (1985) for further discussion).

The above formulation is a mixed integer nonlinear mathematical programming, which in essences is highly intractable. The main advantage of the above formulation is to have all the components of the problem including the sequential nature of adding new project one by one, integrated into a 'uni-level' mathematical programming model.

3.2 Methodological foundations

The NN have demonstrated to be a good tool to characterise and model complex systems in which the behaviour of the system is not known and there is no adequate alternative. They can be categorised as nonlinear regression model with high degree of freedom (Haykin, 1999). Facing the complexity of projects' interdependency in PPP along with no prior knowledge of the impacts of different priority plans in the absence of any other alternative were the main motivations to utilise the NN in this study.

Figure 1 graphically outlines a priority plan consist of four projects in which at each move denoted by ℓ on the basis of the priority plan, one project is constructed and added to the previous network. Figure 1(b) provides an imagination of this move on a devised NN on which the movement or path of one arbitrary PPS is depicted. The NN shown in this figure consists of four layers $(\ell_1, ..., \ell_4)$ with four projects $(\phi_1, ..., \phi_4)$. The layers represent sequential stages in which a new project is added to the corresponding network of preceding layers on priority basis. For instance, the added projects on layers ℓ_1 and ℓ_4 have the first and the fourth priorities. The layers are connected by arrows representing the movement to the next prioritised project from the previous project.

This figure presents a feedforward NN with four projects $(\phi_1, ..., \phi_4)$ considering dots as neurons and arrows as synapses. A sample priority plan corresponds to a PPS has been bolded $(\phi_2 \rightarrow \phi_3 \rightarrow \phi_1 \rightarrow \phi_4)$ in this figure. This phenomenon can be modelled as a TSP because each project in the priority plan must be visited only once on the priority plan path. However, contrary to the TSP in which salesman seeks the shortest path, the longest path (with the most benefit) is preferred. Therefore, we look for the longest path on the NN. With reference to Figure 1(b), the salesman starts from the leftmost, at each layer,

one unselected project is chosen. This process is continued until all projects are selected while the total benefits are gained. The arrows are associated with attributes as proxy for 'benefits'. Arrows' attribute predicts the benefit of choosing the pointed out project by the corresponding arrow. In order to stipulate the search a similar benefit-proxy attribute for each dot (discussed in Section 3) is introduced. However, exact benefit (longest path) of each PPS is computed by carrying out the traffic assignment modules.

This concept is conducted through an iterative process. At each iteration, a solution to the TSP-like problem over the NN is sought and the NN's attributes are updated (NN training) based on the assignment results executed at the end of each iteration. The PPSs are encoded by GA's terminology as a string of genes containing the projects. The first gene's content means that the corresponding project received the first priority. A search engine inspired by the concept of AC and hybridised to GA is developed to find the appropriate PPS (solution) at each iteration step in the network.

AC is inspired by the foraging behaviour of real ants. Ants explore the area surrounding their nest in a random manner. As soon as an ant finds a source of food, it evaluates the quantity and quality of the food and carries some of this food to the nest. During the return trip, the ant deposits a pheromone trail on the ground, which gradually evaporates. Ants determine their movements by judging the pheromone density on a path and choose the one with the most pheromone. Hereby ants can find the optimal path to final destinations at the end (Dorigo et al., 1996). To solve the TSP-like problem, each ant starts from its nest moving towards the best unvisited project where more food or benefit is expected, stochastically. The ant memorises its path and once it completes its tour (PPS), the pheromone on the walked path is updated (training). This procedure is repeated until a termination criterion is met.

The NDP problems are bi-level programming problems belonging to the category of cumbersome problems known as NP-hard problems (Reinelt, 1994). NP-hard, in computational complexity theory, is a class of problems that are, informally, at least as hard as the hardest problems in NP. Supposed that, in NDP, n is number of candidates. The total number of possible permutations for enumeration is $2n$. Meanwhile, the number of permutations in PPP (number of PPS) is $n!$ in which the number of traffic assignment is 2^n. Moreover, the size of the solution space in TSP is $(n-1)!/2$ for $n > 2$, where n is the number of cities. The TSP is a prominent illustration of a NP-hard class of problems in computational complexity theory (Reinelt, 1994). These comparisons summarised in Table 1 reveals the extreme difficulty of tackling the PPP problems.

Table 1 Comparing the difficulty of PPP with NDP and TSP

Problem	*n*	*No. permutations* (*NoP*)	*No. assignments* (*NoA*)	*n = 4*		*n = 8*		*n = 10*	
				NoP	*NoA*	*NoP*	*NoA*	*NoP*	*NoA*
NDP	No. candidates	2^n	2^n	16	16	256	256	1024	1024
TSP	No. cities	$(n-1)!/2$	–	3	–	2520	–	181,440	–
PPP	No. projects	$n!$	2^n	24	16	40,320	256	3,628,800	1024

4 The algorithms

Large transportation projects with interdependent activities could be formulated as a TSP. However, TSPs categorised as NP-hard problems in combinatorial optimisation often are solved with simplistic and unrealistic assumptions. We suggest that having a good solution for a realistic problem is more crucial to the successful management of large transportation projects than having an exact solution to an unrealistic problem with simplistic assumptions. Let us introduce the following symbols in addition to the previously defined symbols (the symbol ': =' means 'set'):

T_{E-Y}	The total travel time of the initial network (see equation (1))
T_E	The total travel time of the full network (E) with all the projects constructed
$\phi_1, ..., \phi_k, ..., \phi_n$	The project names $(\phi_k = (i, j) \in Y)$
B_{ϕ_k}	The intrinsic benefits of the project
λ	A uniform real random number between 0 and 1 representing the value of 0.5 so $2.\lambda \approx 1$ and $\lambda \neq \lambda$
$[.]_1^n$	A customised function passes the nearest integer number between 1 and n to its argument
$PPS(\ell)$	The set of prioritised projects up to corresponding network of layer ℓ in the NN (or the first ℓ prioritised projects in the PPS)
$p^{(m)}$	The mth project in set PPS(ℓ), i.e., $\text{PPS}(\ell) = \{p^{(1)}, p^{(2)}, ..., p^{(m)}, ..., p^{(\ell)}\}$
c	A counter for the number of prioritised projects as the priority plan is being shaped successively through the iterations $(c = 1, ..., n)$

The PPSs as solution to the TSP-like problem are latent in the NN. Inspired by AC algorithm, in an iterative process at each iteration, one PPS is specified by finding the longest path (maximum benefit) on the recently trained and updated NN. These processes are elaborated within an algorithm framework depicted in Figure 2.

Step 0: Initialisation and preparation

Let us denote the ultimate number of iteration as $\overline{s}$ and the current iteration number as s ($\overline{s}$ is a parameter). Thus:

- $s := 1$

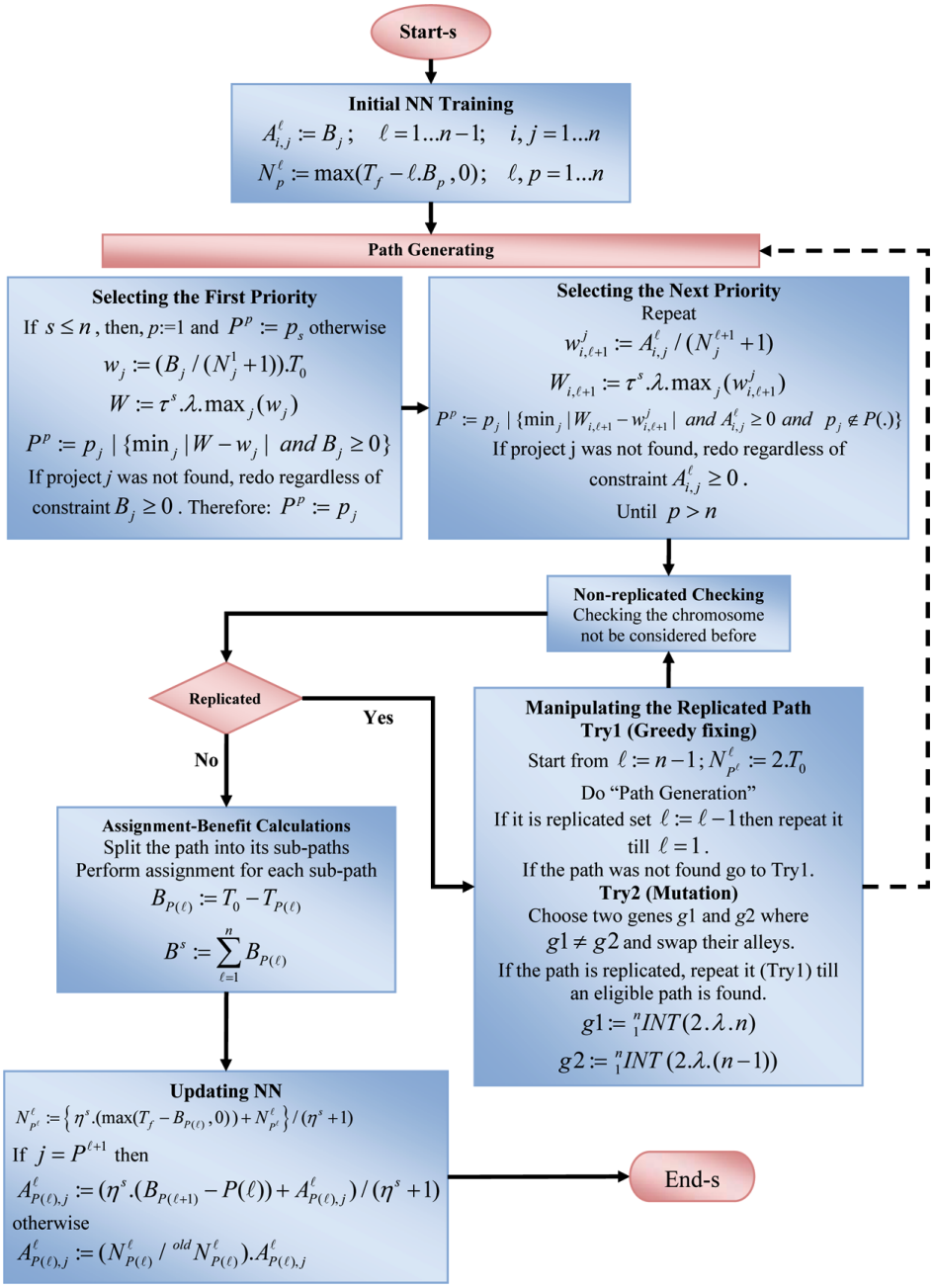
  The algorithm starts by establishing a feedforward NN consisting of n layers, n dotes in each layer, and $n + (n-1)^2$ arrows connecting the layers. The arrows and dotes are attributed with some variables called *utility* and *inertia*, respectively (sometimes we collectively call them *attributes*). Along the shape of the NN, these attributes predict or determine the outputs. Training means calibrating or updating these attributes in the successive iterations. Thus, the attributes are initialised in the beginning before the training starts:

- $U^{\ell}_{\phi_i, \phi_j} := B_{\phi_j} \quad \ell = 1..n-1, \quad \forall \phi_i, \phi_j \in Y$ (3.1)

- $I^{\ell}_{\phi_i} := \max((T_E - T_{E-Y}) - \ell.B_{\phi_i}, 0) \quad \ell = 1..n, \quad \forall \phi_i \in Y$ (3.2)

$U^{\ell}_{\phi_i,\phi_j}$ is the utility of arrow (ϕ_i,ϕ_j) while dotes ϕ_i and ϕ_j belong to the layers ℓ and $\ell+1$. The more utility means the more selection likelihood of project ϕ_j after having ϕ_i already selected in the previous layer (ℓ) of the PPS. With no prior knowledge, B_{ϕ_j}, the intrinsic benefit of the project, is assigned as the utility of all layers. B_{ϕ_j}, is the benefit gained from constructing only the corresponding project.

Figure 2 The proposed algorithm (see online version for colours)



$I_{\phi_i}^{\ell}$ is the inertia of dote or project ϕ_i in layer ℓ. It implies how much benefit is remained to be charged. As the algorithm looks for the maximum benefit in the very beginning of the priority plans (see Figure 1(a)), the project's inertia refers to the probability of the corresponding project NOT being selected as the next one in the PPS. The maximum benefit is the ultimate benefit $(T_E - T_{E-Y})$. In other words, less inertia for projects is desirable. Therefore, the initial values of inertia are set high in the early layers (for the first layer is $(T_E - T_{E-Y}) - 1.B_{\phi_i}$) and get less gradually to the end layers not less than zero.

At the end of each iteration, the NN (attributes) is trained and updated according to the last findings (traffic assignment results). Naturally, the process or the speed of learning or training in the beginning must be high for NN's learning to stabilise. To accommodate this goal, two factors are introduced: perturbation and cognitive factors, denoted by τ^s and η^s at iteration s, respectively. τ^s and η^s are linear gradual reducing factors from upper bounds of $\bar{t}, \bar{n}$ in the early iteration to lower bounds of $\underline{t}, \underline{n}$, respectively:

$$\tau^s := \underline{t} + (s-1).(\bar{t} - \underline{t})/(\hat{s} - 1) \tag{4}$$

$$\eta^s := \bar{n} + (s-1).(\bar{n} - \underline{n})/(\hat{s} - 1). \tag{5}$$

Step 1: Scenario (path) finding

As discussed earlier, the solutions to the PPP have been translated into finding the longest path over the NN. Layer by layer (from left to right in Figure 1(b)) one unselected project with more utility and less inertia stochastically is added to the PPS:

Step 1.1: Selecting the first priority

Each scenario represents a part of the solution space. These scenarios should cover the entire solution space so that all the possible solutions (or the global optimal solution) could be evaluated. This premise is fundamental to all heuristic search algorithms. To address this concern, the selection of the first priority for up to $s \le n$ regardless of any consideration for projects' competence is restricted to ϕ_s (s is the iteration number). Otherwise $(s > n)$, based on the intrinsic benefit of the projects and the project's inertia a project is labelled as the first priority in PPS stochastically:

- If $s \le n$ then
  - $c := 1$ and $P^{(c)} := \phi_s$, (6.0)
- Otherwise $(s > n)$
  - $A_{\phi_j}^{\ell} := \dfrac{B_{\phi_j}}{I_{\phi_j}^{1} + 1} \cdot (T_E - T_{E-Y}) \quad \forall \phi_j \in Y$ (7.1)
  - $A^{*\ell} := \tau^s \cdot \lambda \cdot \max_{\phi_j}(A_{\phi_j}^{\ell})$ (7.2)
  - $P^{(c)} := \phi_j \mid \{\min_{\phi_j} \mid A^{*\ell} - A_{\phi_j}^{\ell} \mid \text{ and } B_{\phi_j} \ge 0\}$ (7.3)
  - If project ϕ_j is not found, then $P^{(c)} := \phi_j \mid \{\min_{\phi_j} \mid A^{*\ell} - A_{\phi_j}^{\ell} \mid\}$ (7.4)
- End If

$A^{\ell}_{\phi_j}$ is called the attractiveness of project ϕ_j wherein the more intrinsic-benefit (B_{ϕ_j}) and less inertia $I^{1}_{\phi_j}$ make the project more attractive to be selected as the first priority. In equation (7.1), the ultimate benefit is embedded just to set the dimension of attractiveness as benefit. To avoid mathematic degeneration dominator in equation (7.1), the inertia is added by 1. In equations (7.2)–(7.4), $A^{*\ell}$ is a stochastic attraction benchmark. The project closest to the benchmark is selected as the first priority. Synergy of τ^s (perturbation factor) and λ (random number) in equation (7.2) make the attraction-benchmark extremely volatile in early iterations in order to explore a wide range solution space. In equation (7.3) constraint $B_{\phi_j} \geq 0$ aims to avoid possible Braess paradox. In cases where the paradox persists, the algorithm proceeds by ignoring the constraint in equation (7.4).

Step 1.2: Selecting the next priorities

Similar to the procedure of specifying the first priority, the remaining priorities are selected by engaging the concept of stochastic benchmark based on the utility of the projects as follows:

- Repeat
  - $\ell := c$
  - $i := P^{(\ell)}$
  - $A^{*\ell}_i := \tau^s \cdot \lambda \cdot \max_j (U^{\ell}_{i,\phi_j})$ (8.1)
  - $P^{(\ell+1)} := \phi_j \mid \{\min_{\phi_j} \mid A^{*\ell}_i - U^{\ell}_{i,\phi_j} \mid \text{ and } U^{\ell}_{i,\phi_j} \geq 0 \text{ and } \phi_j \notin \text{PPS}(\ell)\}$ (8.2)
  - If project ϕ_j is not found, then
  - $P^{(\ell+1)} := \phi_j \mid \{\min_{\phi_j} \mid A^{*\ell}_i - U^{\ell}_{i,\phi_j} \mid \text{ and } \phi_j \notin \text{PPS}(\ell)\}$ (8.3)
  - $c := c + 1$
- While $c \leq n$.

In equation (8.1), with respect to the last specified priority, the maximum utility of moving on the NN's arrows to the next layer is mixed with τ^s and λ to yield a volatile and stochastic attraction benchmark denoted by $A^{*\ell}_i$. Then, in equation (8.2), the algorithm seeks the closest project to the benchmark with positive utility. The algorithm will proceed with equation (8.3) by ignoring the positive utility constraint if no one found.

In general in step 1 of the algorithm, in the earlier iterations; τ^s in its lower bound ($\underline{t}$) along λ allows the benchmarks $(A^{*\ell}, A^{*\ell}_i)$ to become too loose to generate a wide variety of diversified generations by covering the entire solution space. As τ^s ascends to its upper bound ($\overline{t}$), linearly the process becomes free of randomness to select the next-best possible project with respect to the maximum attraction or utility. At the end of this step, a PPS is specified but still needs to be qualified in step 2.

Step 2: Non-replicated checking

The algorithm proceeds to inspect whether the created path over the NN (or PPS) has been generated in the past iterations or not. If the path is replicated, the algorithm initially tries to fix the path by detouring it at the weak joints. A weak joint refers to dotes located at the end of the longest path or projects in the end of the priority list (PPS). It starts from the end of path (layer $\ell = n-1$) and virtually uplifts the project inertia and scraps the tail of the path at that node. Afterward, it turns back to step 1.2 to complete the aborted path in which selecting the same project with uplifted inertial is highly unlikely. Nevertheless, if the path is still replicated, it proceeds backward (preceding layer) and repeats this procedure, until $\ell = 1$ (first layer). If the path is still replicated despite these endeavours, the mutation operator inspired by GA is applied. The process is as follows:

- If the path is replicated then:
  - Try 2.1 (Path-fixing)
  - Reserve nodes' Inertia $(I^{\ell}_{p^{(\ell)}} \mid \ell, p = 1,\ldots,n)$
  - $counter := counter + 1$
  - $\ell := n - counter$
  - If $counter < n$ then:
    - $I^{\ell}_{p^{(\ell)}} := 2 \cdot T_{E-Y}$ (9.1)
    - $\text{PPS}(.) := \text{PPS}(\ell)$, (9.2)
    - $c := \ell - 1$, go to Step 1.2
- End If
  - Restore nodes' inertia $(I^{\ell}_{p^{(\ell)}} \mid \ell, p = 1,\ldots.n)$
  - ***Try 2.2 (Mutation)***
  - Loop
    - Choose two genes (two positions on the path string) as $g1$ and $g2$, where $g1 \neq g2$:
      - $g1 := [2 \cdot \lambda \cdot n]^{n}_{1}$ (10.2) (10.1)
      - $g2 := [2 \cdot \lambda \cdot (n-1)]^{n}_{1}$ (10.2)
    - Now, swap their contents (allelics): $p^{(g1)} \leftrightarrow p^{(g2)}$
    - Go to Step 2
  - End Loop
- End If

In equation (9.1), the inertia of project $p^{(\ell)}$ is temporarily boosted to a very large value $(2 \cdot T_{E-Y})$ at which equation (9.2) discards the end tail of the current PPS string and sends it back to step 1.2 to complete the aborted string.

Mutation in GA avoids straying around the local optimums and targets wider solution space. In other words, the mutation is conducted by altering value(s) of (some) random selected gene(s). In GA, gene' value is called allelic.

The adopted mutation mechanism takes only two priority positions, most likely from the bottom of the priority list and swaps their contents. Knowing that $2.\lambda \approx 1$ in equation (10.1) and equation (10.2), those priority positions ($g1$ and $g2$) would be biased to n and $n-1$, as $\left[\cdot\right]_1^n$ yields nearest integer value to its random argument of $1 \cdot n$ and $1 \cdot (n-1)$. Thus, strong projects on top of the priority list are not likely to be affected. Meanwhile, it provides the opportunity to evaluate the entire solution space ($g1$ might be greater than $g2$). The mutation process is repeated up to a finite number (e.g., n), and if it does not succeed, the algorithm is terminated at last iteration.

At the end of this step, the algorithm moves forward to evaluate the benefits of the created PPS by carrying out the traffic assignment module as follows.

Step 3: Scenario evaluation-benefit computation

According to the developed concept (shown in Figure 1(a)), the cumulative benefit of constructing projects successively on the basis of the priority plan by running the traffic assignment module is computed. The created path is fibred (split) into its sub-paths (i.e., $\text{PPS}(\ell) \mid \ell = 1 \ldots n$) and the benefit of each path (in terms of the users' saved travel time) is computed by running the corresponding traffic assignment module. At the end, the summation of all the sub-paths represents the overall benefit (see the area under the curve in Figure 1(a)):

- $B_{\text{PPS}(\ell)} \coloneqq T_{E-Y} - T_{\text{PPS}(\ell)} \quad \ell = 1,...,n$
- $B^s \coloneqq \sum_{\ell=1}^{n} B_{\text{PPS}(\ell)}.$ (11)

$T_{\text{PPS}(\ell)}, B_{\text{PPS}(\ell)}$ denote the total user travel time and user saved travel time of the sub-path $\text{PPS}(\ell)$, respectively. At the end of the scenario evaluation process, the assignment results are utilised to train or update the NN as follows.

Step 4: Updating the NN

Two components of the NN defined on the arrows and dotes (utility and inertia, respectively), are updated as a part of the training or updating process:

- Reserve: ${}^{\text{old}}I^{\ell}_{p^{(\ell)}} \coloneqq I^{\ell}_{p^{(\ell)}} \quad \ell = 1,...,n$
- $I^{\ell}_{p^{(\ell)}} \coloneqq \{\eta^s \cdot (\max((T_E - T_{E-Y}) - B_{\text{PPS}(\ell)}, 0)) + I^{\ell}_{p^{(\ell)}}\}/(\eta^s + 1) \quad \ell = 1,...,n$ (12)
- For every layer $\ell = 1,...,n$ and every project ϕ_j in layer ℓ do:
  - If $\phi_j = p^{(\ell+1)}$ then
    - $U^{\ell}_{p^{(\ell)},j} \coloneqq (\eta^s \cdot (B_{\text{PPS}(\ell+1)} - B_{\text{PPS}(\ell)}) + U^{\ell}_{p^{(\ell)},j})/(\eta^s + 1)$ (13.1)

- Otherwise
  - $U^{\ell}_{p^{(\ell)},j} := (I^{\ell}_{p^{(\ell)}} / {}^{old}I^{\ell}_{p^{(\ell)}}) \cdot U^{\ell}_{p^{(\ell)},j}$ (13.2)
- Enld If

- End For

The new result from the current iteration in the presence of the cognitive factor η^s is combined with the old established rates (summed up into past iterations) to update or train the NN. η^s regulates the NN's learning speed, which is high in the beginning and decelerate gradually. This helps the algorithm to achieve a convergence status. In equation (12), by moving backward on the layers from $\ell = n$ to $\ell = 1$ and deducting the respective gained benefit $B_{\mathrm{PPS}(\ell)}$ from the ultimate benefit $(T_E - T_{E-Y})$, the updated inertia $(I^{\ell}_{p^{(\ell)}})$ for the project on the corresponding PPS path of iteration s is computed (we omitted superscript s in inertia notation for simplicity). Equation (13) updates the arrows utility explicitly for the arrows on the respected path (equation (13.1)) and implicitly for the remaining arrows in the vicinity of the path (equation (13.2)). Based on the utility's definition, the expected benefit of moving on the arrow originated at layer ℓ is $B_{\mathrm{PPS}(\ell+1)} - B_{\mathrm{PPS}(\ell)}$, the difference of the benefits gained at layers $\ell, \ell+1$. Equation (13.2) combines the latter calculation with the old rates of utility regulated by η^s to provide the last updated rate. As in equations (12) and (13), both $I^{\ell}_{p^{(\ell)}}$ and $U^{\ell}_{p^{(\ell)},j}$ are in linear relation with the $B_{\mathrm{PPS}(\ell)}$, the arrows left out of the PPS path (but still connected to the path's dotes) can be updated by considering changes on the path's dotes in equation (13.2).

In the earlier iterations, η^s is in its upper bound $(\overline{n} = 10)$ and dwindling to its lower bound $(\underline{n} = 0.1)$. Weights of the nodes and utilities of the links lied on the generated path, are adjusted directly and explicitly (equations (12) and (13.1)). Afterwards, the summation of the link's utilities of the path would be close to the calculated total benefit. The weights of the connected links to the generated path are modified according to their corresponding node weights (equation (13.2).

Step 5: Ending the algorithm

- $s := s + 1$
- If $s \le \overline{s}$ then Step 1.
- Stop.

5 Numerical example

In this section, we use the Winnipeg demonstration database presented in EMME/2 User's Manual Software Release 9.0 (1999) to demonstrate the applicability of the proposed framework and exhibit the efficacy of the procedures and algorithms. EMME/2, which stands for equilibre multimodal/multimodal equilibrium, was first developed in the late 1970s at the Center for Research on Transportation (CRT) of the University of Montreal. The network in this database consists of 154 zones, 903 nodes and 2975 links.

The total travel demand with 4420 non-zero origin-destination pairs is 56219. The total travel time of the 'no new project' scenario is $T_{E-Y} = 814678.12$ (relative gap: 0.1%). Eight new bi-directional streets with two lanes in each directions shown in Table 2 are scheduled for construction. The type and congestion function are set to 66 and 8 in the Winnipeg network model.

Table 2 The Winnipeg study with 8 new two-way projects

φ_k	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
i-node	551	424	330	428	494	437	423	297
j-node	610	327	325	330	441	424	304	1057
$B_{\varphi k}$ (user-min)	9629.4	1111.6	1909.3	1044.0	1267.6	682.3	5466.8	1927.1

A quick, simple and intuitive PPP solution with total gained benefits of $B_{\mathrm{PPS}(\ell=n)} = 161,960.6$ user-minutes can be found by sorting the project according to B_{ϕ_j} in a descending order. We can also find an exhaustive enumeration PPP solution with total gained benefits of 169,599.0 user-minutes using the following algorithm developed in Visual Basic, linked to MS-Excel (for inputting data), MS-ACCESS (for managing a database) and EMME/2 (for assigning traffic routes). Therefore, the optimum solution is 4.5% better than the intuitive solution. Moreover, there are 404 scenarios better than the intuitive solution.

5.1 Parameter calibration

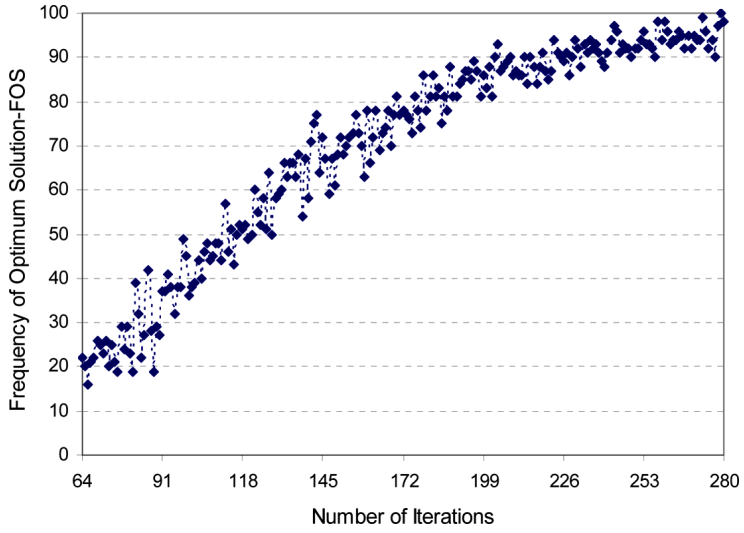
The first step of the algorithm is the calibration of the model.

5.1.1 Number of iterations

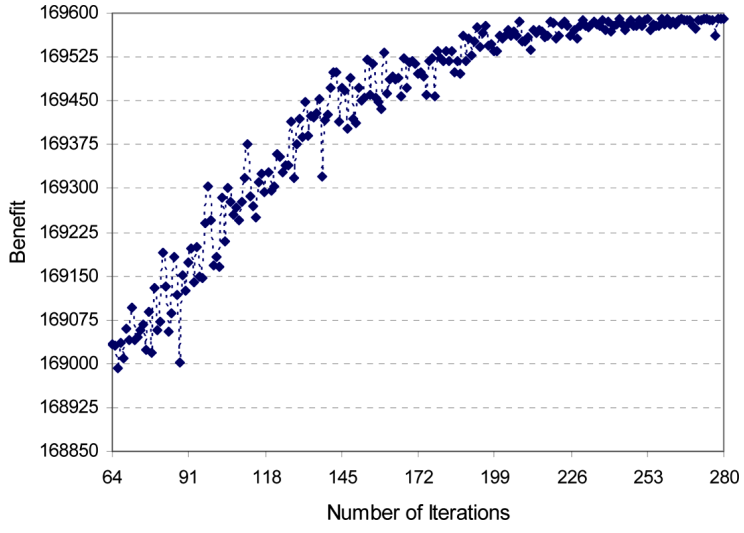
In each iteration, the number of explicitly updated NN's parameters is $n+(n-1)$, with order n. Meanwhile, the total number of NN's attributes is $n \cdot n + (n-1) \cdot n \cdot (n-1)$; in order n^3. The proportion of the total number of parameters to the number of explicitly updatable parameters (in each iteration), is a tip to set a minimum required iteration $\overline{s} = n^3/n = n^2 = 64$. To get a reasonable and appropriate number of iterations, the algorithm ran up to $\overline{s} = 280$, incrementally, at which the frequency of the observed optimum solution (FOS) as shown in Figure 3(a) was observed 100%.

In this research, each run (or trial) was repeated 100 times. Figure 3(b) shows the overall benefit (decreased user travel time) over incremental number of iterations. The large number of iterations in this study is a drawback. For example, 100 trials for $\overline{s} = 280$ lasted lasts 149 min regardless of the assignment time. At 2 times of the minimum required iterations ($\overline{s} = 2*64 = 128$), the FOS remained at 50% and the running time lasted 48 min. In other words, at 2 times of the minimum required iterations, reaching to the exact solution is highly guaranteed ($2*50\% = 100\%$). Therefore, the number of required iterations is defined as $\overline{s} = 2.n^2$.

Figure 3 (a) The frequency of the optimal solution over the number of iterations and (b) the overall benefit (decreased user travel time) over the number of iterations (see online version for colours)



(a)



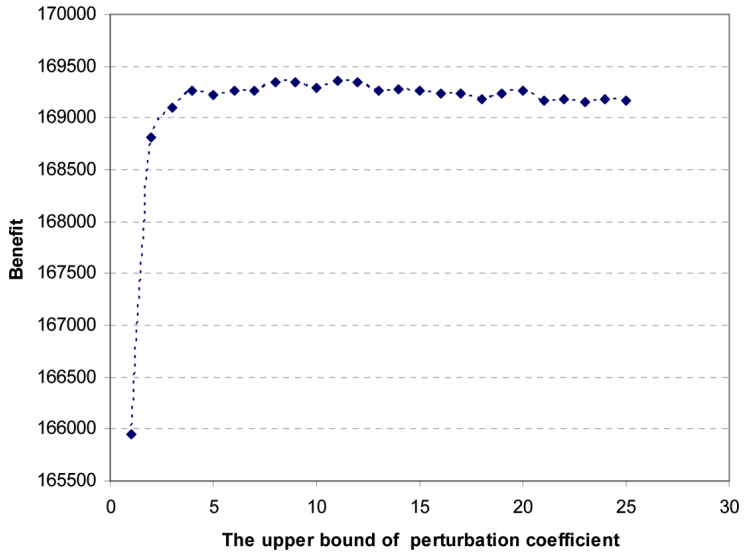
(b)

5.1.2 Perturbation and cognitive factors

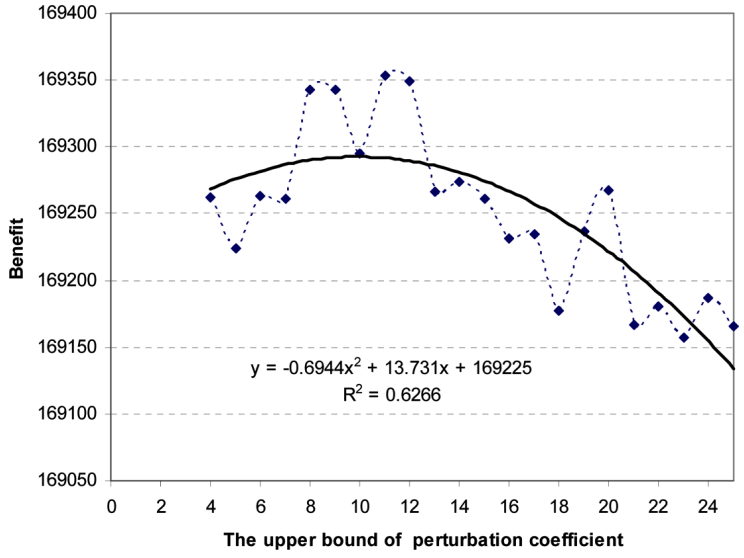
In the Winnipeg study, we assumed the initial values of $\overline{t} = 10$, $\underline{t} = 1$, and $\overline{n} = 10$, $\underline{n} = 0.1$. For the perturbation factor in the beginning, we assumed $\underline{t} = 1$ to have a completely volatile situation by setting $\underline{t} = 1$. To define the upper bound, sensitivity of the result, over various values of $\overline{t}$ (from 1 to 25) is analysed. The benefit values from the algorithm are presented in Figure 4(a). A quadratic function

is fitted to the sieved data in Figure 4(b) to find the optimal value of $\overline{t}$. The zero derivative of the fitted equation gives the optimal upper bound as: $\overline{t} = 9.88$.

Figure 4 (a) A sensitivity analysis of the upper bound of the perturbation coefficient $(\overline{t})$; (b) a quadratic function is fitted to the sieved data to find the optimal value of $\overline{t}$; (c) a sensitivity analysis of the upper bound of the NN's cognitive coefficient $(\overline{n})$ and (d) a quadratic function is fitted to the Sieved data to get the optimal value of $\overline{n}$ (see online version for colours)

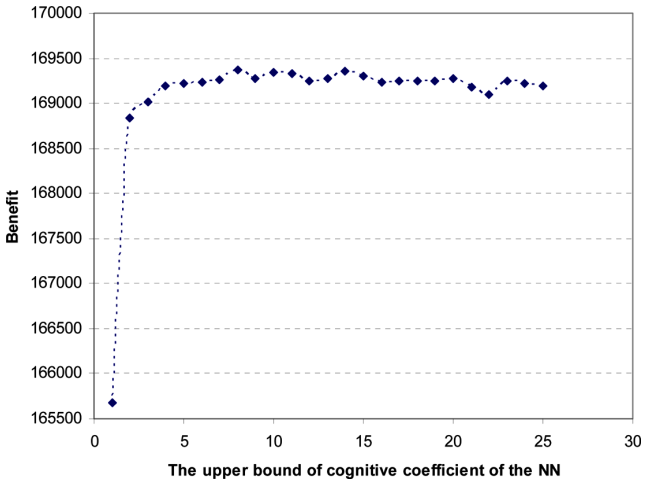


(a)

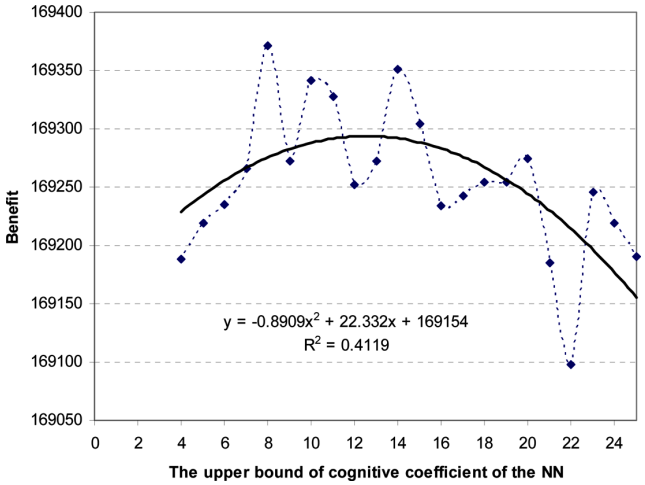


(b)

Figure 4 (a) A sensitivity analysis of the upper bound of the perturbation coefficient $(\overline{t})$; (b) a quadratic function is fitted to the sieved data to find the optimal value of $\overline{t}$; (c) a sensitivity analysis of the upper bound of the NN's cognitive coefficient $(\overline{n})$ and (d) a quadratic function is fitted to the Sieved data to get the optimal value of $\overline{n}$ (see online version for colours) (continued)



(c)

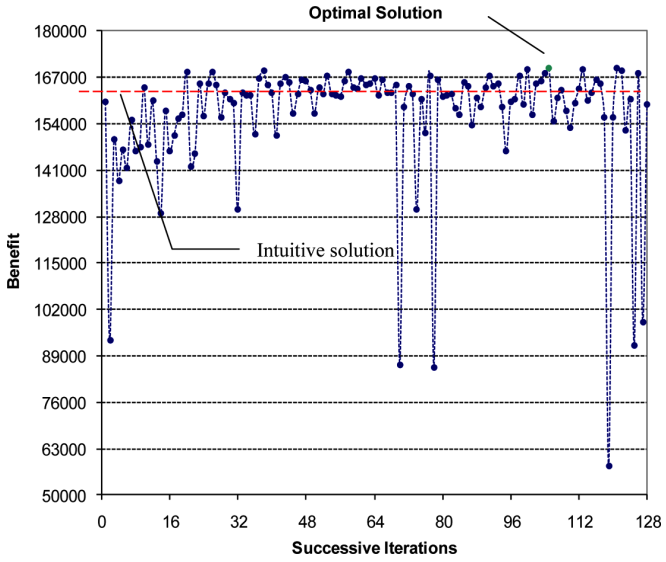


(d)

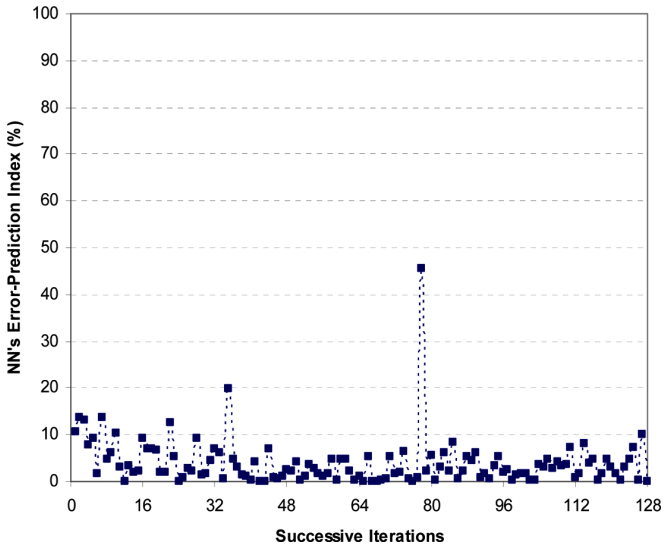
Based on the initial rates shown in equation (3), we assumed a relatively small value of $\underline{n} = 0.1$ for the cognitive factors used for the NN training during the updating (equations (12) and (13)). The $\underline{n} = 0.1$ represents a 10% impact on the new results in the first iteration, which makes scene due to the stochastic nature of the early scenarios (see equation (6)). To define the upper bound, sensitivity of the result over various values of $\overline{n}$ (from 1 to 25) is analysed (Figure 4(c)). A quadratic function is fitted to the sieved data in Figure 4(b) to find the optimal value of $\overline{t}$. The zero derivative of the fitted equation gives the optimal upper bound as: $\overline{n} = 12.53$.

Next, the algorithm was executed with the calibrated parameters ($\hat{s}=128$, $\tau;\{\underline{t}=1,\overline{t}=9.88\}$, $\eta;\{\underline{n}=0.1,\overline{n}=12.53\}$). The algorithm produced significant improvement. The FOS significantly increased from 50% to 69% and the expected benefit remained at 169430.07 (or 0.01% behind the optimum solution). Figure 5(a) demonstrates the variations of the successive scenarios and the trend of the algorithm to reach the optimum solution for the last trial of this run. As shown in this figure, half of the iterations are above the intuitive solution.

Figure 5 (a) The overall benefits of the successive scenarios for the last 100 runs and (b) the NN's Error-Index in the last 100 runs (see online version for colours)



(a)



(b)

5.2 Mutation

The frequency of the mutation remained at 8% in our trials, which is fairly low but not unusual (Holland, 1975). The sharp reductions in Figure 5(a), mostly in the final scenarios, were due to the application of the mutation operator.

5.3 NN's weights

The arrows' attributes (utilities) of the NN were assumed as the expected benefits of adding the heading project to the priority plan. Comparing the collected benefits on the arrows of longest path (B^s_{NN}) and the actual value $(B^s_{\mathrm{PPS}(\ell=n)})$ can indicates the prediction capability of the NN. So, the error index of NN (NN_Error %) is defined as:

$$NN\_\mathrm{Error}\% = | B^s_{NN} - B^s_{\mathrm{PPS}(\ell=n)} | / B^s_{\mathrm{PPS}(\ell=n)} * 100). \tag{14}$$

Figure 5(b) depicts the NN's error index. The average value is fairly low (3.89%). At the beginning of training, the index is tangible and as it proceeds, regardless of some sharp increase (due to the mutation), the index depreciates. At the end of run, the mutation has little effect as the cognitive coefficient (η) decline (equation (13)).

5.4 Computational time

Each assignment takes around 5 s. For the proposed number of iterations $(\bar{s} = 2 * 64 = 128)$, the 100 trials took around 48.32 min excluding the assignment running time. By analysing the running time at span of $1 \le \ell \le n = 8$, it turns out that the running time is proportional to the number of projects with a light slope (around 0.52). In other words, the computational time is a linear and not exponential function of the problem size. In conclusion, the proposed algorithm can handle large-scale problems without difficulty.

6 Conclusions and future research directions

The earlier studies in transportation project evaluation simply used cost-benefit analysis to examine whether a particular infrastructure project (a road, railroad, bridge, port or airport) was likely to repay its investment related costs with some combination of savings in travel time, vehicle operating expenses and accident reductions. However, the complexities inherent in large infrastructure projects have prohibited effective application of cost-benefit analysis methods for project prioritisation. In addition, the traditional cost-benefit analysis underestimates the environmental effects of transportation projects and results in an unreliable and inaccurate prioritisation decisions (Talvite, 2000).

We formulate large transportation projects with interdependent activities as a TSP with the ultimate goal of improving traffic flow, characterised by a reduction in the total travel time experienced by the users. However, despite tremendous progress in combinatorial optimisation, there is no efficient algorithm for solving the TSPs. In other words, the running time for any TSP algorithm increases exponentially with the number of cities and a TSP problem with only hundreds of cities will take many CPU years to

solve exactly. The TSP is regarded as difficult to solve. Even if there is a way to break this problem into smaller sub-problems, the sub-problems will be at least as complex as the original problem. This is what computer scientists call NP-hard problems.

We consider the tradeoff between a good solution for a real-world problem and an exact solution for an out-of-this world problem. We proposed a heuristic method with several hybrid components. An NN was used to cope with the interdependency concerns. An algorithm with an iterative process was confined to search for the path with the most benefit or most reduction in the user-time in the NN as a solution to the TSP. The solution from each iteration step was utilised to update and train the NN and enhance its prediction. A search engine inspired by the concept of AC hybridised with GA was developed to find a suitable solution to the TSP. The hybrid heuristic method proposed in this study was applied to the real data for the city of Winnipeg in Canada to demonstrate the applicability of the proposed framework and exhibit the efficacy of the procedures and algorithms.

The proposed heuristic method in this paper can be utilised for further studies in economic planning and mathematics (especially solving an iconic problem like TSP). Orabi (2010) has shown the pressing need for optimising post-disaster reconstruction efforts of damaged transportation networks in order to simultaneously minimise reconstruction costs and network service disruption. The proposed method could be integrated into planning models for post-disaster reconstruction of damaged transportation networks. Future research directions could also include the consideration of other factors such as construction time, travel demand variation and fiscal changes as well as studying the computational hardness of this problem in computational complexity theory.

Acknowledgement

The authors would like to thank the anonymous reviewers and the editor for their insightful comments and suggestions.

References

Adachi, N. and Yoshida, Y. (1995) 'Accelerating genetic algorithms: protected chromosomes and parallel processing', *Proceedings of the First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, Sheffield, UK, p.414.

Adler, H.A. (1987) '*Economic Appraisal of Transportation Projects: A Manual with Case Studies*, Economic Development Institute of the World Bank.

Aiyer, S.V.B., Niranjan, M. and Fallside, F. (1990) 'A theoretical investigation into the performance of the Hopfield model', *IEEE Transactions on Neural Networks*, Vol. 1, pp.204–215.

Andresol, R., Gendreau, M. and Potvin, J-Y. (1999) 'A Hopfield–Tank neural network model for the generalized traveling salesman problem', in Voss, S., Martello, S., Osman, I.H. and Roucairol, C. (Eds.): *Meta-heuristics Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academic Publishers, Boston, pp.393–402.

Angeniol, B., De-La-Croix, G. and Le-Texier, J-Y. (1988) 'Self-organizing feature maps and the traveling salesman problem', *Neural Networks*, Vol. 1, pp.289–293.

Aras, N., Oommen, B.J. and Altinel, I.K. (1999) 'The Kohonen network incorporating explicit statistics and its application to the traveling salesman problem', *Neural Networks*, Vol. 12, pp.1273–1284.

Balamurugan, K., Selladurai, V. and Ilamathi, B. (2006) 'Solving unequal area facility layout problems using genetic algorithm', *International Journal of Logistics Systems and Management*, Vol. 2, No. 3, pp.281–301.

Baraglia, R., Hidalgo, J.I. and Perego, R. (2001) 'A hybrid heuristic for the traveling salesman problem', *IEEE Transactions on Evolutionary Computation*, Vol. 5, No. 6, pp.613–622.

Bullnheimer, B., Hartl, R.F. and Strauss, C. (1997) 'A new rank based version of the ant systems – a computational study', *Central European Journal for Operations Research and Economics*, Vol. 7, No. 1, pp.25–38.

Burke, L.I. (1994) 'Adaptive neural networks for the traveling salesman problem: insights from operations research', *Neural Networks*, Vol. 7, pp.681–690.

Chien, C.Y. and Chen, S.M. (2009) 'A new method for solving the traveling salesman problem based on parallelized genetic ant colony systems', *Proceedings of the 2009 International Conference on Machine Learning and Cybernetics*, Baoding, Hebei, China, pp.2828–2833.

Cochran, M.A., Pyle, E.B., Greene, L.C., Clymer, H.A. and Bender, D. (1971) 'Investment model for R&D project evaluation and selection', *IEEE Transactions on Engineering Management*, Vol. EM-18, pp.89–100.

Dorigo, M. (1992) *Learning and Nature Algorithm* (in Italian), PhD Dissertation, Dipartimento di Electtonica, Politecnico di Milano, Italy.

Dorigo, M. and Gambardella, L.M. (1997a) 'Ant colony system: a cooperative learning approach to the traveling salesman problem', *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp.53–66.

Dorigo, M. and Gambardella, L.M. (1997b) 'Ant colonies for the traveling salesman problem', *Biosystems*, Vol. 43, No. 2, pp.73–81.

Dorigo, M., Maniezzo, V. and Colorni, A. (1996) 'Ant system: optimization by a colony of cooperating agents', *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, Vol. 26, No. 1, pp.29–41.

Durbin, R. and Willshaw, D. (1987) 'An analogue approach to the traveling salesman problem using elastic net method', *Nature*, Vol. 326, pp.689–691.

Ellabib, I., Calamai, P. and Basir, O. (2007) 'Exchange strategies for multiple ant colony system', *Information Sciences*, Vol. 177, No. 5, pp.1248–1264.

EMME/2 User's Manual Software Release 9.0 (1999) INRO Consultants Inc., Montreal, Canada.

Fiechter, L. (1994) 'A parallel tabu search algorithm for large traveling salesman problems', *Discrete Applied Mathematics*, Vol. 51, No. 3, pp.243–267.

Fort, J.C. (1988) 'Solving a combinatorial problem via self-organizing process: an application to the traveling salesman problem', *Biological Cybernetics*, Vol. 59, pp.33–40.

Fox, G.E., Baker, N.R. and Bryant, J.L. (1984) 'Economic models for R&D project selection in the presence of project interactions', *Management Science*, Vol. 30, pp.890–902.

Freisleben, B. and Merz, P. (1996) 'A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems', *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, pp.616–621.

Gear, T.E. and Cowie, G.C. (1980) 'A note on modeling project interdependence in research and development', *Decision Sciences*, Vol. 11, pp.738–748.

Ghaziri, H. and Osman, I.H. (2003) 'A neural network algorithm for the traveling salesman problem with backhauls', *Computers and Industrial Engineering*, Vol. 44, No. 2, pp.267–281.

Giorgi, L. and Tandon, A. (2002) *Introduction: The Theory and Practice of Evaluation, Project and Policy Evaluation in Transportation*, Ashgate, England, pp.1–13.

Gonazelez, B., Torres, M. and Moreno, J.A. (1995) 'A hybrid genetic algorithm approach for the 'no-wait' flowshop scheduling problem', *Proceedings of the First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, London, UK, pp.59–64.

Gonçalves, J.F., Mendes, J.J.D.M. and Resende, M.G.C. (2005) 'A hybrid genetic algorithm for the job shop scheduling problem', *European Journal of Operational Research*, Vol. 167, No. 1, pp.77–95.

Gupta, J. (1971) 'A functional heuristic algorithm for the flowshop scheduling problem', *Operational Research Quarterly*, Vol. 22, 39–47.

Hartgen, D.T. and Neuman, L.A. (2002) 'Performance (A TQ Point/Counterpoint Exchange with David T. Hartgen and Lance A. Neumann), *Transportation Quarterly*, Vol. 56, No. 1, pp.5–19.

Haykin, S. (1999) *Neural Networks: A Comprehensive Foundation*, Prentice-Hall, Inc., Upper Saddle, NJ.

Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Michigan.

Hopfield, J.J. and Tank, D.W. (1985a) 'Computation of decisions in optimization problems', *Biological Cybernetics*, Vol. 52, No. 3, pp.141–152.

Hopfield, J.J. and Tank, D.W. (1985b) 'Neural computation of decisions in optimization problem', *Biological Cybernetics*, Vol. 52, pp.141–152.

Iniestra, J.G. and Gutiérrez, J.G. (2009) 'Multicriteria decisions on interdependent infrastructure transportation projects using an evolutionary-based framework', *Applied Soft Computing*, Vol. 9, No. 2, pp.512–526.

Janson, B.N., Buckels, L.S. and Peterson, B.E. (1991) 'Network design programming of US highway improvements', *ASCE Journal of Transportation Engineering*, Vol. 117, No. 4, pp.457–478.

Johnson, E.L., Kostreva, M.M. and Suhl, U.H. (1985) 'Solving 0-1 integer programming problems arising from large-scale planning models', *Operations Research*, Vol. 34, No. 4, pp.803–819.

Jong, J.C. and Schonfeld, P. (2001) 'A genetic algorithm for selecting and scheduling interdependent projects', *Journal of Waterway, Port, Coastal, and Ocean Engineering*, Vol. 127, No. 1, pp.45–52.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983) 'Optimization by simulated annealing', *Science*, Vol. 220, No. 4598, pp.671–680.

Kohonen, T. (1982) 'Self-organized formation of topologically correct feature maps', *Biological Cybernetics*, Vol. 43, pp.59–69.

Kohonen, T. (1995) *Self-organizing Maps*, Springer, Berlin.

Lawler, E.L., Lenstra, J.K. and Shmoys, D.B. (1985) *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley Series in Discrete Mathematics and Optimization, Chichester.

Leleur, S. (1995) *Road Infrastructure Planning: A Decision-Oriented Approach*, Polyteknisk Forlag, Denmark.

Liaw, C.F. (2000) 'A hybrid genetic algorithm for the open shop scheduling problem', *European Journal of Operational Research*, Vol. 124, No. 1, pp.28–42.

Lin, C.M. and Gen, M. (2008) 'Multi-criteria human resource allocation for solving multistage combinatorial optimization problems using multiobjective hybrid genetic algorithm', *Expert Systems with Applications*, Vol. 34, No. 4, pp.2480–2490.

Little, I.M.D. and Mirlees, J.A. (1994) 'The costs and benefits of analysis. Project appraisal and planning twenty years on', *Cost-Benefit Analysis*, Cambridge University Press, Cambridge.

Liu, F. and Zeng, G. (2009) 'Study of genetic algorithm with reinforcement learning to solve the TSP', *Expert Systems with Applications*, Vol. 36, No. 3, pp.6995–7001.

Mackie, P. and Preston, J. (1998) 'Twenty-one sources of error and bias in transportation project appraisal, *Transportation Policy*, Vol. 5, pp.1–7.

Man, K.F., Tang, K.S. and Kwong, S. (1999) *Genetic Algorithms: Concepts and Designs*, Springer, New York.

Martin, O.C. and Otto, S.W. (1996) 'Combining simulated annealing with local search heuristics', *Annals of Operations Research*, Vol. 63, No. 1, pp.57–75.

Murray-Tuite, P.M. and Mahmassani, H.S. (2004) 'Methodology for determining vulnerable links in a transportation network', *Transportation Research Record*, Vol. 1882, pp.88–96.

Naimi, H.M. and Taherinejad, N. (2009) 'New robust and efficient ant colony algorithms: using new interpretation of local updating process', *Expert Systems with Applications*, Vol. 36, No. 1, pp.481–488.

Nash, J. (1951) 'Noncooperative games', *Annals of Mathematics*, Vol. 54, pp.286–295.

Nemhauser, G.L. and Ullmann, Z. (1969) 'Discrete dynamic programming and capital allocation', *Management Science*, Vol. 15, No. 9, pp.494–505.

Ning, X., Lam, K.C. and Lam, M.C.K. (2010) 'Dynamic construction site layout planning using max-min ant system', *Automation in Construction*, Vol. 19, No. 1, pp.55–65.

Noorul Haq, A. and Kannan, G. (2006) 'Two-echelon distribution-inventory supply chain model for the bread industry using genetic algorithm', *International Journal of Logistics Systems and Management*, Vol. 2, No. 2, pp.177–193.

Orabi, W. (2010) *Optimizing Highway Reconstruction and Rehabilitation Projects*, Doctoral Dissertation in Civil Engineering, University of Illinois at Urbana-Champaign, Urbana, Illinois.

Osman, I.H. and Laporte, G. (1996) 'Meta-heuristics: a bibliography', *Annals of Operations Research*, Vol. 63, pp.513–623.

Palmer, D. (1965) 'Sequencing jobs through a multi-stage process in the minimum total time – a quick method of obtaining a near optimum', *Operations Research Quarterly*, Vol. 16, pp.101–107.

Park, Y.B. (2001) 'A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines', *International Journal of Production Economics*, Vol. 73, No. 2, pp.175–188.

Peng, E., Gupta, K.N.K. and Armitage, A.F. (1996) 'An investigation into the improvement of local minima of the Hopfield network', *Neural Networks*, Vol. 9, pp.1241–1253.

Qiang, Q. and Nagurney, A. (2008) 'A unified network performance measure with importance identification and the ranking of network components', *Optimization Letters*, Vol. 2, pp.127–142.

Raj, A.M. and Shahabudeen, P. (2006) 'Optimisation of distribution process in a continuous credit based consumer goods supply chain', *International Journal of Logistics Systems and Management*, Vol. 2, No. 2, pp.142–159.

Rajendran, C. (1994) 'A no-wait flowshop scheduling heuristic to minimize makespan', *Journal of the Operational Research Society*, Vol. 45, pp.472–478.

Reeves, C.R. (1994) *Genetic Algorithms and Neighborhood Search, Evolutionary Computing*, Springer, Berlin, pp.115–130.

Reeves, C.R. (1995) 'A genetic algorithm for flowshop sequencing', *Computers and Operations Research*, Vol. 22, No. 1, pp.5–13.

Reinelt, G. (1994) *The Traveling Salesman: Computational Solutions for TSP Applications*, Springer-Verlag, Berlin.

Saadatmand-Tarzjan, M., Khademi, M., Akbarzadeh, T.M.R. and Moghaddan, H.A. (2007) 'A novel constructive-optimizer neural network for the traveling salesman problem', *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, Vol. 37, No. 4, pp.754–770.

Sandhu, M. (2006) 'Project logistics with the dependency structure matrix approach – an analysis of a power plant delivery', *International Journal of Logistics Systems and Management*, Vol. 2, No. 4, pp.387–403.

Sheffi, Y. (1985) *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*, Prentice-Hall, Inc., Upper Saddle, NJ.

Sheffi, Y. (2005) *The Resilient Enterprise: Overcoming Vulnerability for Competitive Advantage*, MIT Press, Cambridge, Massachusetts.

Sinha, K.C. and Zongzhi, L. (2004) 'Methodology for multicriteria decision-making in highway asset management', *Proceedings of the 83rd Annual Meeting of the Transportation Research Board*, Washington DC.

Stützle, T. and Hoos, H.H. (1996) *Improving the Ant System: A Detailed Report on the MAX-MIN Ant System*, FG Intellektik, FB Informatik, TU Darmstadt, Germany, Tech. Rep. AIDA–96–12.

Stützle, T. and Hoos, H.H. (2000) 'MAX–MIN ant system', *Future Generation Computer Systems*, Vol. 16, No. 8, pp.889–914.

Talvite, A. (2000) 'Evaluation of road projects and programs in developing countries', *Transport Policy*, Vol. 7, 61–72.

Tao, X. and Schonfeld, P. (2006) 'Selection and scheduling of interdependent transportation projects with island models', *Transportation Research Record*, Vol. 1981, pp.133–141.

Tao, X. and Schonfeld, P. (2007) '*Island Models for Stochastic Problem of Transportation Project Selection and Scheduling*, CD-ROM, TRB 2007 (07-1895).

Tsamboulas, D., Yotis, G. and Panou, K. (1999) 'Use of Multicriteria methods for assessment of transport projects', *Journal of Transportation Engineering*, Vol. 125, No. 5, pp.407–414.

Tsamboulas, D.A. (2007) 'A tool for prioritizing multinational transport infrastructure investments', *Transport Policy*, Vol. 14, No. 1, pp.11–26.

Vakhutinsky, A.I. and Golden, B.L. (1995) 'A hierarchical strategy for solving traveling salesman problems using elastic nets', *Journal of Heuristics*, Vol. 1, pp.67–76.

Valls, V., Ballestin, F. and Quintanilla, S. (2008) 'A hybrid genetic algorithm for the resource-constrained project scheduling problem', *European Journal of Operational Research*, Vol. 185, 495–508.

Weingartner, H.M. (1966) 'Capital budgeting of interrelated projects: survey and synthesis', *Management Science*, Vol. 12, pp.485–516.

Winter, G., Périaux, J., Galán, M. and Cuesta, P. (Eds.) (1995) *Genetic Algorithms in Engineering and Computer Science*, Wiley, New York.

Wright, A.H., Vose, M.D., De Jong, K.A. and Schmitt, L.M. (Eds.) (2005) *Foundations of Genetic Algorithms*, Springer, Heidelberg.

Xie, X.F. and Liu, J. (2009) 'Multiagent optimization system for solving the traveling salesman problem (TSP)', *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, Vol. 39, No. 2, pp.489–502.

Yagmahan, B. and Yenisey, M.M. (2010) 'A multi-objective ant colony system algorithm for flow shop scheduling problem', *Expert Systems with Applications*, Vol. 37, No. 2, pp.1361–1368.

Yang, H. and Bell, M.G.H. (1998) 'Models and algorithms for road network design: a review and some new developments', *Transport Review*, Vol. 18, No. 3, pp.257–278.

Yang, Y.J. and Wu, C. (2009) 'An attribute-based ant colony system for adaptive learning object recommendation', *Expert Systems with Applications*, Vol. 36, No. 2, pp.3034–3047.

Yi, J., Bi, W., Yang, G. and Tang, Z. (2008) 'A fast elastic net method for traveling salesman problem', *Proceedings of the 8th International Conference on Intelligent Systems Design and Applications*, Kaohsiung, Taiwan, pp.462–467.

Zalzala, A.M.S. and Fleming, P.J. (Eds.) (1997) *Genetic Algorithms in Engineering Systems*, The Institution of Electrical Engineers, London.