Models and Algorithms for Shortest Paths in a Time Dependent Network

Yinzhen Li[1,2,*] Ruichun He[1] Zhongfu Zhang[1]
Yaohuang Guo[2]

[1] Lanzhou Jiaotong University, Lanzhou 730070, P. R. China
[2] Southwest Jiaotong University, Chengdu, 610031, P. R. China

Abstract *The shortest path problem in the time dependent network is an important extension of the classical shortest path problem and has been widely applied in real life. It is known as nonlinear and NP-hard. Therefore, the algorithms of the classical shortest path are incapable to solve this problem. In this paper, the models of the shortest path problem in the time dependent network are formulated and algorithms are designed for solving the proposed models. Finally, a numerical example is given.*

Keywords time dependent networks, shortest path, model, genetic algorithm,traffic and transportation

1 Introduction

The classical shortest path problem(SPP) is an important branch of the optimization problems and has been extended to a wide varieties[1]. The derivative SPP is an important extension of SPP and has been widely applied in ITS, computer science and communications. There are variety of derivative problems. Time dependent shortest path problem(TDSPP)[2] is a kind of them.

Since it was initialized by Cherkassky, Goldberg and Radzik[2] in 1996, the TDSPP has been widely applied in the real life. For example, the vehicle's running time in a road section varies with the time, because of the degree of traffic jam which is usually different in different time in same road section. Therefore, the roadblock function is actually a function of time. However, the classical SP algorithm such as Dijkstra algorithm can not be used to solve the TDSP problem. There are some algorithms for TDSPP in recent years . Chen[3] analyzed the TDSP problem, however, he did not give any suggestion for the basic algorithm of TDSPP. Dreyfus[2] proposed the opinion to solve the TDSPP by using improved Dijkstra algorithm, however, this algorithm is proved to be wrong by Kaufman[4].

*Corresponding author. E-mail address: liyz01@mail.lzjtu.cn

Futhermore, an efficient algorithm for TDSPP, which was added some constraints on the weights of edges, was given by Chen and Tang[4]. Orda, Rom and Cai[5][6][7] proposed the algorithm to TDSPP with the conditions that vehicles are permitted to wait in nodes. However, it is forbidden for vehicles to wait in many cases such as at intersections in urban traffic networks. G.Tan proved that TDSP problem can be divided into FIFO network and NFIFO network (refer to the reference[8]). He proposed that TDSPP of FIFO networks can be solved by Dijkstra algorithm, and TDSPP of NFIFO networks can be solved by inverse ordering Dijkstra algorithm in the case of taking into account discrete time.

In this paper, we discuss and put forward to models of TDSPP under the conditions of not limited by discrete or continuous travel times and a hybrid intelligent algorithm is designed in which the priority-based encoding genetic algorithm is embedded. Finally, a numerical example is given.

2 Models

Definition 2.1 *Let $G = \{V, E, F(t)\}$ be a directed graph, where $V = \{1, 2, \cdots, n\}$ is the set of nodes,*

$$E = \{(i, j) | i \neq j, i, j \in V\}$$

the set of edges and

$$F(t) = \{f_{i,j}(t) | (i, j) \in E\}$$

that of weights of edges, $f_{i,j}(t)$ is the function of time $t \in [a, b]$, $b > a \geq 0$. Then G is called as a time dependent network (TDN).

Definition 2.2 *For a TDN, let $p_{1,n}(t)$ be a path from source node 1 to destination node n. The length of $p_{1,n}(t)$ is defined as a function of time t. Let $P_{1,n}$ denotes the set of all paths from 1 to n. The path $p^*_{1,n}(t^*)$ such that*

$$W(p^*_{1,n}(t^*)) \leq W(p_{1,n}(t)), \forall p_{1,n}(t) \in P_{1,n},$$

is called the minimum time path, where $W(.)$ is the path length (including waiting time in nodes).

Firstly, we define some notations. Let

$$N^+(j) = \{i | (i, j) \in E\},$$

$$N^-(j) = \{i | (j, i) \in E\}.$$

$x_{i,j}$: the decision variable of which the value equal to 1 if edge (i, j) in the corresponding path in $P_{1,n}$ and equal to 0 otherwise.

a_i: the arriving time at node i by path $p_{1,n}(t)$.

l_i: the departing time from node i by path $p_{1,n}(t)$.

By such a way, the TDSPP can be formulated as follows:

$$
\begin{aligned}
\min \quad & Z(x,t) = \sum_{i,j \in V} x_{i,j} \cdot f_{i,j}(l_i) + \sum_{i,j \in V} (l_i - a_i) \cdot x_{i,j}, \\
\text{s.t.} \quad & \sum_{i \in N^+(j)} x_{i,j} - \sum_{i \in N^-(j)} x_{j,i} = \begin{cases} 1, & \text{if } i = 1, \\ 0, & \forall i, j \in V \backslash \{1, n\}, i \neq j, \\ -1, & \text{if } i = n. \end{cases} \\
& a_i \leq l_i \leq a_i + b_i, i \in V, \\
& a_1 = a, \\
& a_i = l_j + f_{j,i}(l_j), i \geq 2, \\
& x_{i,j} = 0 \text{ or } 1.
\end{aligned} \tag{1}
$$

In the constraints of equation (1), the first is the constraints of the path from node 1 to node n, the second is the constraint of wait of node i, and the third and forth are the travelling time relation from j to i.

Clearly, the equation (1) is nonlinear and is also a NP-Hard problem. And it characterizes the general model of TDSPP. If $l_i = \{a_i | \forall i \in V\}$, the equation (1) characterizes the TDSPP with not permitted to wait in any node. If $l_i = \{a_i | \forall i \in V, i \geq 2\}$, model (1) characterizes the TDSPP which the source node is permitted to wait. The problem degenerated to classical SP problem when $f_{i,j}(t)$ are constants and $l_i = a_i$, $(i,j) \in E$.

3 Hybrid Genetic Algorithm

We consider both finding optimal paths and optimal waiting times in each node in this TDSP problem. Finding optimal paths can be solved by the priority-based encoding genetic algorithm in Section 3.1 and optimal time scheme for each path can be achieved by the genetic algorithm designed in Section 3.2. The optimal waiting times are exactly the values of chromosome fitness functions in the Section 3.1. Combining the algorithms in Section 3.2 and that in Section 3.2, we can get the hybrid genetic algorithms in Section 3.3.

3.1 Priority-Based Encoding Genetic Algorithm

R.Cheng and M.Gen proposed the priority code genetic algorithm to solve project sequencing problems with restricted resources in 1997, afterwards they solved SP problem in this method[2]. M.Gen also solved the bi-criterion SP problem by the genetic algorithm in 2004[9, 10, 11].

Generate initial population. Generate a random integer $v_i \in [1..n]$ for any node $i, i = 1, 2, \cdots, n$, and $v_i \neq v_j, i \neq j, v_i$ denotes an unique priority of node i. Then, to begin with node 1, find out the conterminous node i^* with node 1 with the maximal priority and put i^* into the set of paths. And then begin with i^*, find out the conterminous node j^* with node i^* with the maximal priority and put j^* into the set of paths, $\cdots$. Repeat this process until the node n is in the set of paths. In this way, we get a path from node 1 to node n. We can get $size1$ paths by

repeating the above process. This is initial population. The code of a chromosome is shown in figure 1:

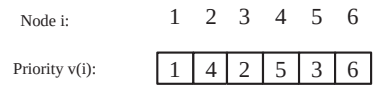

Figure 1. The code of a chromosome

Evaluate the fitness of a chromosome. Apparently, the fitness function is the path length of classical SP problem, and the path length is a constant if having gotten the path. But the path length in TDSPP is a function of time if we have gotten the path $p_{1,n}(t)$. In other words, the value of the fitness function is an optimal value of constrained optimization problem as follows:

$$\begin{aligned} \min \quad & Z^* = \sum_{(i,j)\in p_{1,n}(t)} f_{i,j}(l_i) + \sum_{(i,j)\in p_{1,n}(t)} l_i - a_i, \\ \text{s.t.} \quad & a_i \le l_i \le a_i + b_i, \\ & a_1 = a, \\ & a_i = l_k + f_{k,i}(l_k), \end{aligned} \tag{2}$$

where l_i is the decision variable.

The evaluating function of chromosome i (path i) is defined as follows:

$$eval(Z_i^*) = \frac{1}{Z_i^*} \div \sum_{k=1}^{size1} \frac{1}{Z_k^*} \tag{3}$$

Selection operation. The roulette wheel approach is adapted here. We select $size1$ chromosomes from parent population as an offspring according to $eval(Z^*)$ of each path.

Crossover operation. The OX(Order Crossover) proposed by Davis is adapted here. We select chromosomes according to the crossover probability P_c and randomly group them by pairs. We can get an offspring by selecting preserved genes at random of $parent1$ and replace unpreserved genes of $parent1$ with corresponding genes of $parent2$. The process is shown in figure 2:

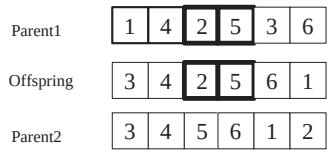

Figure 2. Example of OX operator

Mutation operation. We select chromosomes from parent population according to the mutation probability P_m. Select two genes at random for mutation and exchange them. The mutation process is shown in figure 3:

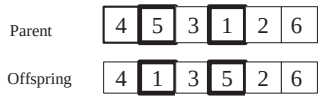

Figure 3. Example of mutation process

3.2 Genetic Algorithm for Optimizing the Value of Fitness Function

The model of fitness function of path m in Section 3.1 can be simplified as follows:

$$\begin{array}{ll} \min & Z_m^* = a_n - l_i, \\ \text{s.t.} & a_i \le l_i \le a_i + b_i, \\ & a_i = l_k + f_{k,i}(l_k), \\ & a_1 = a. \end{array} \tag{4}$$

Apparently, in order to get optimal l_i^*, assume that we have gotten a path from node 1 to node n, $p_{1,n}(t) = \{1, 2, 3, \ldots, n\}$. It is clear that $l_1 \in [a, a + b_1]$, and if $l_1 = x_1$ then $l_2 \in [a_2, a_2 + b_2]$, where $a_2 = x_1 + f_{1,2}(x_1)$, $\cdots$. So we can design the following genetic algorithm to get l_i^*.

Coding and initial population. We adopt the real number encoding in this algorithm. Given a chromosome $l = (l_1, l_2, \cdots, l_n)$. Generate a real number $x_1 \in [a, a + b_1]$at random, and let $l_1 = x_1$, then $a_2 = x_1 + f_{1,2}(x_1)$. Generate a real number $x_2 \in [a_2, a_2 + b_2]$ at random again, and let $l_2 = x_2$, then $a_3 = x_2 + f_{2,3}(x_2)$,$\cdots$. At last we can get a feasible chromosome l. Through repeating the above process we can get $size2$ initial population. The fitness function of chromosome i is formulated as follows:

$$eval(i) = \frac{1}{a_n^i - l_1} \div \sum_{k=1}^{size2} \frac{1}{a_n^k - l_1}, \tag{5}$$

where, a_n^i denotes the time that the ith chromosome arrives at node n. And the optimal chromosome is preserved in order to guarantee convergence.

Selection operator. The roulette wheel approach is adopted.

Step 1. *Calculate.* $q_i = \sum_{j=1}^{i} eval(j), i = 1, 2, \cdots size2.$

Step 2. *Generate a uniform random number* $r \in [0, q_{size2}]$.

Step 3. *If* $q_{i-1} < r \le q_i$ *satisfied then the chromosome* i *is selected.*

Step 4. *Repeat Step 2 to Step 3 until* $size2$ *chromosomes are selected.*

Crossover operator. We select chromosomes by crossover probability P_c and randomly group them by pairs. Assume a crossover operation is between the chromosome j and k, generate a uniform random integer $i \in (1, m)$, where m is the

number of nodes in path $p_{1,n}$ and i is the position for crossover operation. Then exchange the ith gene in the chromosome j and k. After this operation, if the chromosome j and k are feasible or not,we should check up them. If not feasible, then need to mend them. The process is as follows.

Step 1. *If l_i satisfies the first constraint of equation (4), go to Step 2, otherwise go to Step 4.*

Step 2. *If $i = m$, go to Step 5, otherwise then, let $i = i + 1$, calculate $a_i = l_{i-1} + f_{i-1,i}(l_{i-1})$.*

Step 3. *If l_i satisfies the first constraint of equation (4), go to Step 5.*

Step 4. *Generate random number $x \in [a_i, a_i + b_i]$, let $l_i = x$, then go to Step 2.*

Step 5. *The chromosome is feasible, stop.*

Mutation operator. We select chromosomes from parent population by mutation probability P_m, generate a random position of the gene $i \in [1, m]$ for mutation.

Step 1. *Generate a random number $x \in [a_i, a_i + b_i]$, let $l_i = x$.*

Step 2. *If $i = m$, then go to Step 4. Otherwise, we must check up this chromosome if feasible, if not feasible, need to mend it, according to Step 1 to Step 5 in crossover operator.*

Step 3. *Let $i = i+1, a_i = l_{i-1} + f_{i-1,i}(l_i)$. If l_i satisfies the first constraint of equation (4), then go to Step 4, otherwise go to Step 1.*

Step 4. *The chromosome is feasible, stop.*

3.3 Hybrid Genetic Algorithm

We design the following hybrid genetic algorithm in Section 3.2 and Section 3.1.

Step 1. *Generate $size1$ initial population at random according to priority-based encoding algorithm.*

Step 2. *Crossover operation.*

Step 3. *Mutation operation.*

Step 4. *Optimize the value of fitness function $Z_i^*, i = 1, 2, \cdots, size1$.*

- ***Step 4.1.*** *Generate $size2$ chromosomes at random as initial population.*
- ***Step 4.2.*** *Crossover operation.*
- ***Step 4.3.*** *Mutation operation.*
- ***Step 4.4.*** *Evaluate the fitness function which is denoted by equation (5) in Section 3.2.*
- ***Step 4.5.*** *Selection operation.*

Table 1: Network parameters

i	b_i	arc	$f_{i,j}(t)$	i	b_i	arc	$f_{i,j}(t)$	i	b_i	arc	$f_{i,j}(t)$
1	5	1,2	$3+te^{-t}$	9	2	9,10	5	17	1	17,18	$3+3e^{-(6-t)^2}$
		1,3	$2t+3e^{-4t}$			9,13	$8-2e^{-t^2}$			17,22	$4-2e^{-(t)^2}$
		1,4	$5+2e^{-t^2}$			9,14	$9-3e^{-(t-2)^2}$	18	2	18,22	$6+3e^{-2t^2}$
		1,5	$3+2te^{-3t}$			9,15	$6+(10-t)^2/8$			18,23	8
		1,6	$4+(3-t)^2$	10	3	10,11	$3+2e^{-t^2}$			18,24	$5+(18-t)^2/10$
2	3	2,7	$6+(2-3t)^2$			10,15	$5+(12-t)^2/10$	19	2	19,18	$9-3e^{-(20-t)^2}$
		2,8	$3t+6e^{-t^2}$			10,16	6			19,24	$12-3e^{-(16-t)^2}$
3	2	3,2	$2+3e^{-t^2}$	11	0	11,16	$5+3e^{-2t^2}$			19,25	$10+2e^{-t^2}$
		3,4	5	12	1	12,13	4	20	1	20,18	$3+5e^{-2t^2}$
		3,8	$4+(1-t/3)^2$			12,117	$5+2e^{-t^2}$			20,21	$5+2e^{-t^2}$
4	3	4,5	$6-3e^{-2t^2}$	13	2	13,14	2			20,25	$4+4e^{-3t^2}$
		4,9	4			13,17	$6+3e^{-2t^2}$	21	2	21,25	$3+(20-t)^2/5$
5	0	5,9	$3+(3-2t)^2$			13,18	$7+2e-t^2$			21,26	$4+(18-t)^2/8$
		5,10	$4+2e^{-(1/2)t^2}$	14	1	14,18	$3+(14-t)^2/10$	22	2	22,23	$6-3e^{-(2+t)^2}$
6	1	6,5	6			14,19	6			22,27	$7+2e^{-t^2}$
		6,11	$6+(5-t)^2$			14,20	$5+3e^{-t^2}$	23	1	23,24	$8+(25-t)^2/6$
7	3	7,12	$2t+2e^{-t2}$	15	2	15,14	$3+2e^{-(10-t)^2}$			23,27	$6+(23-t)^2/8$
		7,13	10			15,20	4	24	2	24,25	$5+(24-t)^2/6$
8	0	8,7	4			15,21	$5-2e^{-3t^2}$			24,27	$4+(26-t)^2/7$
		8,9	$6+(7-t)^2$	16	3	16,18	$8-3e^{-(7-t)^2}$	25	3	25,26	$6+2e^{-3t^2}$
		8,13	11			16,21	5			25,27	$5+4e^{-(3+t)^2}$
								26	2	26,27	$10+3e^{-(4-t)^2}$

Step 4.6. *Terminate this process after running predetermined rounds or the satisfied solution is obtained, report the evaluating Z_i^* and go to Step 5; otherwise go to Step 4.2.*

Step 5. *Evaluate fitness function(equation (3) in Section 3.1).*

Step 6. *Selection operation.*

Step 7. *If a satisfied solution is obtained then stop; otherwise go to Step 2.*

4 Numerical Example

A time dependent network with 64 edges and 27 nodes is given for numerical experiment (Figure 4). The related parameters are given in Table 1.

We design computer programs of the hybrid genetic algorithm. By running the program with the given example, we get a shortest path and the times scheme, where $size1 = 500, P_c = 0.2, P_m = 0.02, size2 = 500$.

shortest path: $1, 5, 10, 15, 20, 25, 27$.

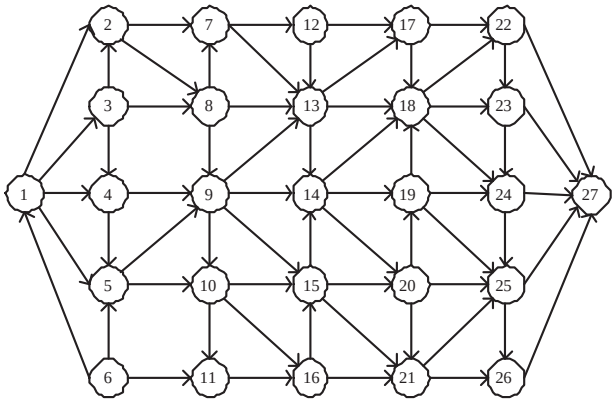

Figure 4 An example of a network

times scheme:

$a_1 = 0$, $l_1 = 4.8$, $a_5 = 7.80$, $l_5 = 7.80$, $a_{10} = 11.80$, $l_{10} = 11.83$, $a_{15} = 16.84$, $l_{15} = 16.87$, $a_{20} = 20.87$, $l_{20} = 20.88$, $a_{25} = 24.88$, $l_{25} = 24.92$, $a_{27} = 29.92$.

The figure 5 shows us the evolutional process of the time scheme of an optimal path in 193th path population.

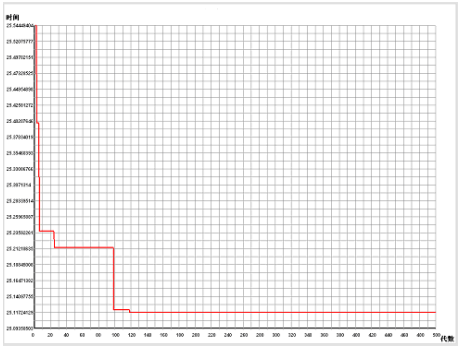

Figure5 . Evolutional process of the time scheme of an optimal path

The figure 6 shows us the evolutional process of optimal paths.

Acknowledgement

This work is supported by National Natural Science Foundation of China (No. 70071028), and Qinglan Project of Lanzhou Jiaotong University.

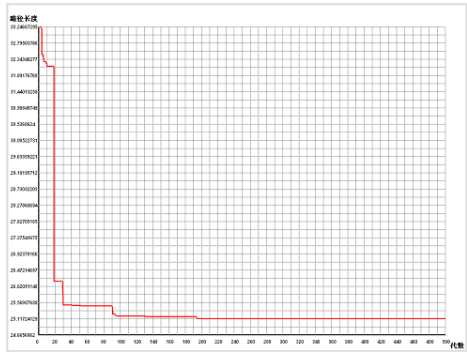

Figure6 Evolutional process of optimal paths

References

[1] E. Erkut. The discrete p-dispersion problem[J]. European Journal of operational Research, 1990:46, 48–60.

[2] B.V. Cherkassky, Andrew V. Goldberg, Tomasz Radzik. Shortest paths algorithms:Theory and experimental evaluation[J]. Mathematical programming, 1996:73, 129–174.

[3] E. S. Dreyfus. An appraisal of some shortest path algorithms[J]. Oper. Res., 1969:17, 395–412.

[4] Y. L. Chen, K. Tang. Minimum time paths in a network with mixed time constraints[J]. Computer Operational Research, 1998: 25, 793–805.

[5] D. E. Kaufman, R. L. Smith. Fatest path in time-dependent networks for intelligent vehicle-highway systems application. IVHS Journal, 1993:11, 1–11.

[6] A. Orda, R. Rom. Shortest path and minimum-delay algorithms in networks with time-dependent edge-length[J]. Joutnal of ACM, 1990:37, 607–625.

[7] A. Orda, R. Rom. Distributed shortest path protocols for time dependent networks[J]. Distributed computing, 1996:10, 49–62.

[8] X. Cai, T. kloks, C. K. Wong. Shortest path problems with time constraints[C]. In: Proc 21st international symposium on mathematical foundations of computer science, Cracow, Poland, 1996, 255–266.

[9] G. Tan, W. Gao. Shortest path algorithm in time-dependent networks[J]. Chinese J. Computers, 2002:2, 1–6.

[10] R. Cheng, M. Gen. Resource constrained project scheduling problem using genetic algorithms[J]. Inter. J. of Intelligent Automation and Soft Computing, 1997:3, 273–286.

[11] M. Gen. Bicriteria network design problem[R]. In: 3rd International Conference of Information and Management Science, Dunhuang, China, 2004:411–417