The Sixth International Symposium on Operations Research and Its Applications (ISORA'06)
Xinjiang, China, August 8–12, 2006
Copyright © 2006 ORSC & APORC, pp. 380–389

Faster Genetic Algorithm for Network Paths*

Yinzhen Li[1] Ruichun He[1] Yaohuang Guo[2]

[1] Lanzhou Jiaotong University, Lanzhou, Gansu 730070, China
[2] Southwest Jiaotong University, Chengdu, Sichuan 610031, China

Abstract Through analyzing the algorithm presented by Mitsuo.Gen, and taking into account the schema theorem and the architecture block hypothesis of a genetic algorithm, we showed some flaws in Mitsuo.Gen algorithm in this paper. A better genetic algorithm for solving shortest path problems is presented further, which is based on the technology of dynamic coding of the priority of vertex and gene weight. The microevolution strategy is also considered fully in the paper. After putting forward the measure of the importance of a vertex in a network structure and its formula, the real coding priority of a vertex is generated in a non-uniform distribution function with the parameters of the measure of the vertex importance. Flexible fitness functions, elitist genes selection, the mountain climbing method for local optimum and other methods are adopted. It indicates that the time efficiency of the algorithm is higher than Mitsuo.Gen algorithm through a lot of numeric examples.

Keywords network; optimal path; genetic algorithm

1 Introduction

As an important branch of graph and network, the traditional shortest path problem (SPP) has extensive applications. That is to say, the SPP is a classical research topic. It was proposed by Dijkstra in 1959 and has been widely researched [1,4–7]. The research work in the past relates to two aspects: the research of the time efficiency of algorithms and the research of the derivative problems from SPP. Though the Dijkstra algorithm based on the Bellman optimization theory is considered as the most efficient method, it become inefficient when needing to repeat much computation or when the network being very big, and it also cannot be implemented in the permitted time. For example, there are tens of thousands of freight bills which need to be processed each day in the cargo accounting system in one Railway Bureau of China. 60 percent of the magnitude computation time is consumed in the calculation of the freight paths [2,3]. 95 percent of the time in the urban traffic flow assignment system is spent on calculating the SPP [8] . For instance, we need 55 hours to compute paths for solving the 3-hour dynamic urban traffic flow assignment problem [9]. The efficient research of SPP embodies two aspects. The first aspect is deflating the search

*Doctoral Foundation of Ministry Education of China (20050732002) and Qinglan Project of Lanzhou Jiaotong University support this work.

scope of the nodes and reducing the computation complexity through defining the artful data structure [1,9–15]. The second aspect is adopting the parallel algorithms for accelerating the process [16–18]. With the development of communications, computer science, traffic and transportation systems, and so on, derivative problems from the traditional SPP are becoming more and more important in real life. For instance, in the communication nets, the nodes and the edges have costs. The calculation of the railway freight tariff routes is constrained by the freight category and the appointed routes in the railway transportation network. In urban traffic system the restrictions on the path calculation are more popular, such as the restriction of the transit ability of the path and the no entry restriction and so on. Therefore, the most practical applications are considered to be the derivative problems, such as SPP with multi-weights, SPP with constraints, SPP with stochastic environment and SPP with fuzzy environment.

The SPP with arbitrary constraints is proved to be NP-Hard [19]. Therefore, a polynomial algorithm for this problem is impossible even though it is one of the most practical applications, to which many researchers have paid attention [20–25].

2 Shortest Path Problem and Genetic Algorithms

There are many heuristic algorithms used to solve complex optimal problems. Genetic algorithms (GAs) are one of the most powerful and successful methods in stochastic search and optimization techniques based on the principles of the evolution theory. In fact, GAs have succeeded in the combination and optimization field, but the research results on computing SPP by use of genetic algorithms are scarce. The reason is that encoding for a path in a network is critical for designing a GA. Mitsuo.Gen, the professor of Waseda University in Japan, pointed out that the encoding of the chromosome for SPP is more difficult than TSP. The special difficulties are:

(1) The number of nodes in each path is not fixed. Or, the number of nodes in a path from the original vertex to the destination vertex is uncertain. If $|V| = n$, it is possible that the path with the most nodes includes $n-1$ nodes, while the path with the fewest nodes includes 2 nodes.

(2) A random sequence of edges usually does not correspond to a path.

The 0-1 linear program model of SPP is formulated as follows:

$$\begin{aligned}
\min \quad & \sum_{(i,j)\in E} w_{i,j} x_{i,j} \\
\text{s.t.} \quad & \sum_{j:(i,j)\in E} x_{i,j} - \sum_{j:(j,i)\in E} x_{j,i} = \begin{cases} 1, & i = s \\ -1, & i = t \\ 0, & i \neq s,t \end{cases} \\
& x_{i,j} = 0 \text{ or } 1
\end{aligned}$$

In order to solve this problem, some researchers proposed the binary code method. In this method, $x_{i,j} = 1$ indicates that the result path includes the arc (i, j), and $x_{i,j} = 0$

means opposite circumstance. Intuitively, this code method is very simple. However, it is very difficult to practice for the reason above that Mitsuo.Gen pointed out.

To overcome such difficulties, Mitsuo.Gen and R.Cheng *et al.* presented a very effective coding method in 1998. They solved this problem by encoding some guiding information for constructing a path, but not a path itself, in a chromosome. They pointed out that the position of a gene is used to represent a node and its value is used to represent the priority of the node for constructing a path. The chromosome length n is fixed if $|V| = n$. We can get a feasible path by decoding the chromosome based on the gene value. The encoding method is called priority-based encoding [26]. The detail of this method is presented in [26].

3 The Priority-Based and Dynamic-Gene-Weight-Based Encoding GA

The priority-based encoding genetic algorithm is a very effective method due to the position of a gene to be used to represent a node while the value is used to represent the priority of the node for constructing a path among the candidates. The method gets rid of the limitation of the binary coding or the decision-variable coding based on $x_{i,j}$. But at the same time, we find some deficiencies in their method from the view of the model theorem and the architecture block hypothesis of a genetic algorithm. The main deficiencies are as follows:

(1) The convergence would be accelerated if the initial population covers the neighborhood that includes the optimal solution. We present two methods for solving this problem. Firstly, we increase the probability of the neighborhood including the optimal solution or improve the diversity of the chromosomes by augmenting the initial population. Secondly, we artificially generate the initial population, in order to contain the optimal solution in the initial population or the optimal gene slice in the chromosomes, through analyzing the property of the solution space and the objective function. It is obvious that the former is superior based on the inherent parallel of GA. However, the calculation time becomes longer along with augmenting the initial population. To the latter, it has trouble in the analysis of the problem. But we can exclude this trouble that is not considered in Mitsuo.Gen method after investigating this problem thoroughly.

(2) The merit of Mitsuo.Gen method lies in the fact that the decoding is simple and capable of mapping one chromosome to the only one solution (path) by encoding the chromosome with the non-repeated integer $(1 \le v(i) \le |V|, v(i) \ne v(j))$. However, this encoding is unfit for the genetic operators. It will result in the destruction of the optimal chromosome slice, the influence of multiple gene positions, and the infeasible offspring derived form the cross operation and the mutation operation. Consequently, many other computations are needed to mend the unfeasible chromosome.

(3) In general, the cross operation should not augment the gene diversity, for it only inherits the gene slice from the parent chromosome. It should generate the

multiplex chromosomes in order to satisfy the global optimal. However, the cross operation in Mitsuo.Gen method generates not only a diversity of chromosomes but also a diversity of genes. As we know the purpose of the mutation operation is to keep the diversity of a gene and to satisfy the local optimal. Moreover, some researchers insist that the high mutation probability would destruct the valuable gene and compel the method to do the stochastic optimal search work. From this point of view, if the mutation probability is low, we can accelerate the convergence. Mitsuo.Gen method neglects this problem .

(4) In GA, the essence of the selection operator is "the survival of the fittest" and the fitness function is the key to this objective. Generally we can use the Roulette Wheel selection technique that selects the highest fitness value chromosome to get into the next generation. It is important to point out that the selection operator will bring the pre-mature and the model cheat problem at the cost of the convergence of the algorithm.

Based on Mitsuo.Gen method, we proposed the improved GAs which is based on the technology of dynamic coding of the priority of vertex and gene weight for solving the SPP.

The following factors are considered in our method:

(1) Real encoding. Generate a real number $M \cdot h(i)$ for each node, where M is a rather big integer, $h(i) = W(i)Rand\,(0,1)$ is a non-uniform distribution stochastic number in $(0,1)$, and $W(i)$ is the priority of the node. This encoding method does not needing mend operations because cross or mutation operation wouldn't destroy the feasibility of the chromosomes.

(2) Optimal gene block. We prefer the optimal gene block in the chromosome when initializing the population. Taking into account the property of the SPP, the optimal solution usually composed with the shorter edge and the fewer edges. Now we cannot know how many edges that composed the shortest path before hand, but we can utilize the edge length. So we can impose the function $h(i)$ to ensure that the two adjacent nodes with the shorter edge weight will be included in the chromosome. This is our so-called encoding based gene weight. The microevolution strategy is also considered here. In fact, the effect of each node in a network is different according to its structure and parameters. So we put forward the concept of the importance measure of a vertex in a network structure. Its formula is as follows:

$$W\,(i) = \left[1 - \frac{\sum\limits_j c_{i,j}}{\sum\limits_i \sum\limits_j c_{i,j}} \right], \quad (i,j) \in E,\ \forall i \in V.$$

(3) Dynamic encoding based on the varying fitness function. Here we adopt the two phases method. In the forepart $[1, \beta T]$ ($T$is the iteration number, $0 < \beta < 1$) of evolution, in order to prevent the pre-mature and the model cheating which would compel the algorithm to be limited in the local optimal search, we adopt the rather large population and lower the pressure of the selection operation (e.g. weaken the effect of the fitness function). In the second phase $(\beta T, T]$, we adopt the smaller

node	1	2	3	4	5	6
priority	12.85	42.60	30.00	10.25	9.83	20.41

Figure 1: The code of chromosome

population and increase the pressure of the selection operation in order to promot the local optimal search and accelerate the convergence.

(4) Mutation operator based on gene weight. To guarantee the convergence we must retain the best chromosome. We can retain the best chromosome if we adopt the non-uniform distribution of the gene position for the mutation operator according to the gene weight.

(5) Mountain climbing method. Mountain climbing method is also adopted for the local optimal searching. The mountain climbing method has superiority in solving the local optimal search. There is a local optimal search in the final stages of GA. So we adopt the uniform stochastic sequence mountain climbing method for local optimal search in our method.

The priority-based and dynamic-gene-weight-based encoding genetic algorithm is a very effective method.

The steps of the algorithms are as follows.

Step1. Generate initial population.

Step1-1. Calculate the importance measure of the vertex i in the network structure.

$$\forall i \in V, \quad W(i) = \left[1 - \frac{\sum\limits_j c_{i,j}}{\sum\limits_i \sum\limits_j c_{i,j}} \right], \quad (i,j) \in E$$

Step1-2. Generate random priority based on the structure of importance measure

$$\begin{aligned} v(i) &= M \cdot h(i) \\ &= M \cdot W(i) \cdot Rand(), \quad \forall i \in V \end{aligned}$$

where, $Rand()$ is a uniform distribution stochastic number in $(0,1)$. The code of the chromosome is shown in figure 1:

Step1-3. Generate *size* chromosomes according to Step1-2 as the initial population.

Step2. Crossover operator.

Select $size1$ chromosomes by crossover probability P_c and randomly group them by pairs. In this way we can get $\left\lfloor \frac{size1}{2} \right\rfloor$ pairs as crossover parents. We select the substrings of two parents at random and exchange substrings between parent1 and

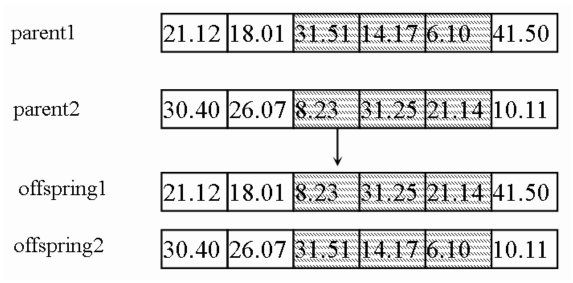

Figure 2: Example of the crossover

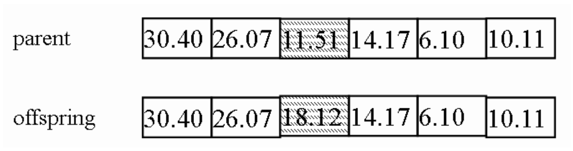

Figure 3: Example of mutation

parent2, then determine the mapping relationship, and finally legalize offspring with the mapping relationship. The process is shown in figure 2.

Step3. Mutation operator.

Select *size*2 chromosomes by mutation probability P_m.

Step3-1. Calculate the weight of each gene:

$$GW(i) = \frac{1}{v(i)} \div \sum_j \frac{1}{v(j)}$$

Step3-2. Calculate the selected cumulative probability of each gene:

$$\begin{cases} q_0 = & 0 \\ q_i = & \sum\limits_{j=1}^{i} GW(i), \quad i = 1, 2, \cdots n \end{cases}$$

Step3-3. Generate a uniform distribution stochastic number r in $(0,1)$, if $q_{i-1} < r \leq q_i$, then the ith gene position is the mutation gene position.

Step3-4. Generate a priority for ith gene. The function is defined as follows:

$$v(i) = M \cdot W(i) \cdot Rand()$$

The process is shown in figure 3.

Step4. Fitness calculation and selection operator

Step4-1. Decoding. We perform the decoding work for each chromosome in the population.

For the kth chromosome, the decoding work is as follows:

(a) Let $i = 1, P(k) = \{1\}$.

(b) $j^* = \arg\max_j \{v(j)\,|(i,j) \in E\ AND\ \{j\} \notin P(k)\}$. If several identical $v(j)$ exist, the smallest j is selected. In this way, the exclusive solution is guaranteed. The node i is a dead node if we cannot find such j. In the circumstance of a dead node, we can break the process by adding the virtual edge (i,n), where $j^* = n, w_{i,n} = Q$ and Q is the maximal integer.

(c) $i = j^*$, $P(k) + \{i\} \to P(k)$.

(d) If $i = n$, the path $P(k)$ is the derived solution path according to the chromosome k, otherwise, jump to (b).

Step4-2. Calculate the length of the result path.

$$L(k) = \sum w_{i,j}\,|(i,j \text{ are the two adjacent vertexs in } P(k))$$

Step4-3. Calculate the fitness.

$$f(i) = \frac{1}{L(i)} \div \frac{1}{\sum\limits_k L(k)}, \quad i,k = 1,2,\cdots,size$$

Step4-4. Calculate the dynamic variable fitness function. The exponential function is adopted to be as the dynamic variable fitness function.

$$f'(i) = e^{\alpha \cdot f(i)},$$

Let $\alpha = \alpha_1$ during the forepart of the iteration $[1, \beta T]$ and $\alpha = \alpha_2$ in the $(\beta T, T]$ phase, $\alpha_2 > \alpha_1$. The iteration number is T, $0 < \beta < 1$, generally $\beta = \frac{2}{3}$.

Step4-5. Selection. Let $\delta \cdot size \to size$, $\delta \in [0.6\ 0.9]$, in the second iteration phase for reducing the population.

(a) $eval(i) = f'(i)/\sum\limits_k f'(k), \quad i,k = 1,2,\cdots,size$

(b) $\begin{cases} q_0 = & 0 \\ q_i = & \sum\limits_{j=1}^{i} eval(j) \end{cases}$

(c) Generate a uniform stochastic number $r \in (0,1)$. Select the ith chromosome as the offspring individual.

(d) Repeat (c) until we get *size* offspring.

Step5. Adopt the mountain climbing method for the local optimal search in the iteration phase $(\gamma T, T]$. Let $\gamma = 0.95$ according to the problem scale.

We propose the uniform stochastic sequence mountain climbing method for the local optimal search in our method.

Table 1: The computation results

No.	The number of nodes	The number of edges	Mitsuo.Gen algorithm		Our algorithm	
			Convergent solution	Convergent probability	Convergent solution	Convergent probability
1	100	258	1791	0.4	1791	0.9
2	100	253	2159	0.6	2159	1.0
3	100	252	2216	0.5	2216	0.9
4	92	234	1712	0.4	1712	1.0
5	90	246	1736	0.3	1736	0.8
6	90	232	1542	0.4	1542	0.8
7	90	220	1935	0.4	1935	0.9
8	80	206	1519	0.4	1519	1.0
9	80	187	2126	0.5	2126	0.8
10	80	195	2533	0.4	2533	0.9

For the optimal chromosome i in the current population, let $l^* = L(i)$:

(a) Generate the stochastic sequence *List* of the number $1 - n$.

(b) For each item k in the *List*, mutate the kth gene value:

$$v(k) = M \cdot W(k) \cdot Rand().$$

(c) Evaluate the mutation chromosome $l = L(i)$. If $l < l^*$, then $l^* = l$, the mutation is successful; otherwise, resume the original gene value.

(d) Go to (b); Repeat until all the items in the *List*are processed.

(e) Go to (a); Repeat until the mountain climbing iteration number is C.

Step6. Repeat the Step2–Step6 until the T round iterations are completed.

4 Numeric examples

To evaluate the performance of Mitsuo.Gen algorithm and our method, 10 different synthetic networks are generated by the random network generator designed by us. For the 10 different networks, the iteration number is 500. The computation results of the average of the 10 runs with different initial populations are listed in table 1 and figure 4.

The results show that our method is better on the convergence. The convergence process on the network with 100 nodes and 258 edges is shown in figure 4. Mitsuo.Gen algorithm is convergent at the 277 generation whereas our method is convergent at the 182 generation.

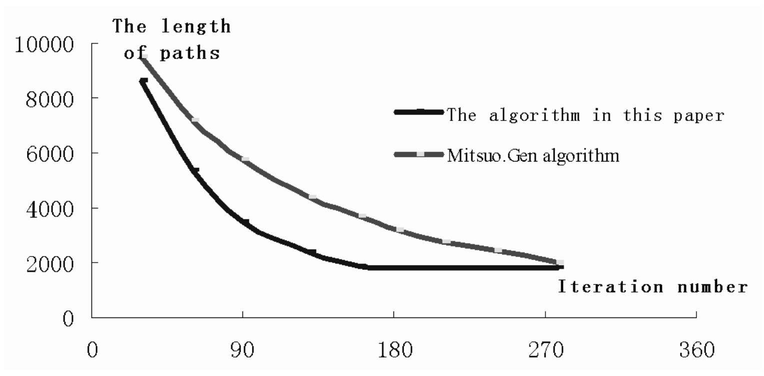

Figure 4: The convergence performance of the two algorithms

References

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

[2] Yinzhen Li. Calculation of freight tariff routes on railway network. *J. of the China Railway Sosciety*, 19(3), 14–18, 1997.

[3] Yinzhen Li, Shouhuai Gu. Directional searching algorithm for finding shortest path between two vertexes in railway network. *J. of the China Railway Sosciety*, 19(2), 25–27, 1997.

[4] Mikkel Thorup. Floats, integers, and single source shortest path. *J. of Algorithms*, 35(2), 189–201, 2000.

[5] H. Shi. Time work tradeoffs of the single source shortest paths problem. *J. of Algorithms*, 30(1), 19–32, 1999.

[6] F. M. Heide and B. Vocking. Shortest path routing in arbitrary networks. *J. of Algorithms*, 31(1), 105–131, 1999.

[7] D. Goldfarb. An O(mn)-time network simple algorithm for the shortest path problem. *Oper. Res.* 47(3), 445–448, 1999.

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

[9] P. van Emde Boas, R. Kaas and E. W. Dijkstra. Design and implementation of an efficient priority queue. *Math. Syst. Theory*, 10, 99–127, 1997.

[10] D. Shier and C. Witzgall. Properties of labeling methods for determining shortest paths trees. *J. Res. Natl. Bur. Stand.*, 86, 317–330, 1981.

[11] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. Assoc. Comput. Math.*, 34, 596–615, 1987.

[12] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for network problems. *SIAM J. Comput.*, 1013–1036, 1989.

[13] A. V. Goldberg. Scaling algorithms for the shortest path problem. In *Proceedings 4th ACM-SIAM Symposium on Discrete Algorithms*, 222–231, 1993.

[14] A. V. Goldberg and T. Radzik. Heuristic improvement of the Bellman-Ford algorithm. *Applied Math. Let.*, 6, 3–6, 1993.

[15] R. K. Ahuja, K. Mehlhorm, J. B. Orlin and R. E. Tarjan. Faster algorithms for the shortest path problem. *J. Assoc. Comput. Math.*, 37(2), 213–223, 1990.

[16] J. L. Traff. An experimental comparison of two distributed single-source shortest path algorithms. *Parallel Computing*, 21(12), 1505–1532, 1995.

[17] H. E. Romeijn and R. L. Smith. Parallel algorithms for solving aggregated shortest pathproblem. *Computers & oper. Res.*, 26, 941–953, 1999.

[18] M. Hribar, V. E. Taylor and D. E. Boyce. Implementing parallel shortest path for parallel transportation applications. *Parallel Computing*, 27(12), 1537–1568, 2001.

[19] M. R. Garey and D. S. Johnson. *Computers and Intractability*, Science Publishing Company, 1987.

[20] M. Mohsen and D. Hassan. Network Reduction for the acyclic constrained shortest path problem. *European Journal of Operational Research*, 63, 124–132, 1992.

[21] C. D. Luis and J. N. Maco. Shortest path problems with partial information: Models and algorithms for detecting dominance. *European Journal of Operational Research*, 121, 16–31, 2000.

[22] R. Hassin. Approximation schemes for the restricted shortest path problem. *Math. of Oper. Res.*, 17(1), 36–42, 1992.

[23] X. Cai. Time varying shortest path problem with constraints. *Networks*, 29(2), 141–149, 1997.

[24] M. I. Hening. The shortest path problem with two objective functions. *European Journal of Operational Research*, 25, 281–291, 1985.

[25] Y. L. Chen and K. Tang. Minimum time paths in a network with mixed time constraints. *Computer Operational Research*, 25(10), 793–805, 1998.

[26] R. Cheng and M. Gen. A priority based encoding and shortest path problem. *Technical Report*, Ashikaga, Institute of Technology, 1998.