THE APPLICATION OF GENETIC ALGORITHM IN GIS NETWORK ANALYSIS

Qishi WU **Jeffrey J. SHAN**
Geomatics Engineering
School of Civil Engineering
Purdue University
West Lafayette, IN 47907-1284
U.S.A
wuq@ecn.purdue.edu jshan@ecn.purdue.edu

Working Group IV/1

KEY WORDS: Algorithm, Geographic Information System, Network Analysis, Genetic Algorithm

ABSTRACT

As one of the advanced analysis capabilities in GIS, network analysis provides strong decision support for users in searching optimal path, finding the nearest facility and determining the service area. To lead to an effective solution for GIS network analysis, a new random searching method -- genetic algorithm is introduced and applied in this article. The classical combinatorial optimization problem (knapsack problem) is used to introduce the concepts of genetic algorithm and describe its various operations in detail, such as encoding, crossover, mutation and inversion. Selection of parameters to reach the optimal performance for the genetic algorithm is also discussed in general. The GIS network analysis is then formalized as a general combinatorial optimization problem consisting of an objective function and a general constraint condition. The implementation details of this algorithm are discussed in terms of encoding and genetic operations. Test results for a network with eighty nodes are presented to demonstrate the efficiency of the genetic algorithm and its application potential.

1 INTRODUCTION

As an interdisciplinary subject across computer science, remote sensing, geodesy and geography, GIS (Geographic Information System) is being widely used as an effective tool to explore geospatial related phenomena in natural resource, environment, planning and management. The power of the GIS relies on the geospatial analysis capabilities imbedded within the system. Various analysis modules have been developed for GIS to broaden its application horizon and enhance its capability.

Network analysis is an important advanced analysis function in GIS. In present GIS network analysis modules, heuristic algorithms have been used to carry out its search strategy. Due to the lack of globally sampling in the feasible solution space, these algorithms have considerable possibility of being trapped into local solution [1,8,10]. Recently developed genetic algorithm (GA), which simulates the process of biologic evolution, shows favorable performance and effectiveness in solving typical combinatorial optimization problem [2]. It necessitates the application of genetic algorithm into GIS network analysis.

A network is a directed graph with arcs and nodes. A typical network in GIS may contain roads and bus stops, railways and stations, power lines and transformers. All these commutation, transportation and transmission activities take place within the network. Network analysis modules are therefore needed for a GIS to plan the optimal route, find the nearest facilities and determine the service areas. This article will apply the genetic algorithm to optimal route selection in network analysis.

This remaining of this article is organized as follows. In section 2, the optimization strategy based on genetic algorithm is presented. After a brief introduction to genetic algorithm, a general method of genetic algorithm is described by using the classical knapsack problem as an example. Moreover, discussions made on the selection of parameters and its affect on the performance of the genetic algorithm. Section 3 formalizes the best route selection problem in network analysis according to the generic model of the genetic algorithm. Detailed discussions are then presented on its algorithmic implementation, such as encoding, parameter selection, fitness function, and genetic operations. Test results are also given in this section to illustrate the designed algorithm and its effectiveness. Concluding remarks and further research efforts are addressed in Section 4.

2 OPTIMIZATION STRATEGY BASED ON GENETIC ALGORITHM

2.1 Introduction to genetic algorithm

Genetic algorithm is a computational model simulating the process of genetic selection and natural elimination in biologic evolution. Pioneering work in this field was conducted by Holland in the 1960 s [3,4]. It was proposed to find global or local optimal solution in the enormous search space. Comparing to traditional search algorithms in artificial intelligence (AI), genetic algorithm is able to automatically acquire and accumulate the necessary knowledge about the search space during its search process, and self-adaptively control the entire search process through random optimization technique. Therefore, it is more likely to obtain the global optimal solution without encountering the trouble of combinatorial explosion caused by disregarding the inherent knowledge within the search space. In addition, the genetic algorithm is characterized by its simplicity, robustness, adaptability to parallel process and flexibility in applications. It has been used to solve combinatorial optimization problems and non-linear problems with complicated constraints or non-differentiable objective functions.

2.2 General method of genetic algorithm

The computation of genetic algorithm is an iterative process that simulates the process of genetic selection and natural elimination in biologic evolution. For each iteration, candidate solutions are retained and ranked according to their quality. A fitness value is used to screen out unqualified solutions. Genetic operators, such as crossover, mutation, translocation and inversion, are then performed on those qualified solutions to estimate new candidate solutions of the next generation. The above process is carried out repeatedly until certain convergent condition is met. To illustrate the principal and its algorithm, the classical knapsack problem [1,5] is taken as an example. The classical knapsack problem can be described as follows:

$$\text{Maximize: } \sum_{i=1}^{n} B_i X_i \tag{1}$$

$$\text{Constrain: } \sum_{i=1}^{n} S_i X_i \le C \quad X_i \in \{0,1\}, \quad 1 \le i \le n \tag{2}$$

where S_i represents the resource consumption for the i-th activity, C represents the total available resource and B_i represents the gained profit from the i-th activity. X_i is set to either 1 or 0: $X_i = 1$ means that the i-th activity is carried out, $X_i = 0$ means the i-th activity is not carried out. The essence of knapsack problem is to pursue the maximum profits with the constraint of limited total available resource. Following steps are carried out to obtain the solution.

(1) Genetic encoding

A string T of n binary bits is used to represent one possible solution. If the i-th activity is carried out, $T(i)(1 \le i \le n) = 1$, otherwise $T(i)(1 \le i \le n) = 0$.

(2) Select initial population size M and produce M random solutions as initial solution set $T_k \ (1 \le k \le M)$.

(3) Calculate the fitness value $f(T_k)$ for each solution $T_k (1 \le k \le M)$.

The objective function of the knapsack problem can be represented as:

$$J(T_k) = \sum_{i=1}^{n} T_k(i) B_i \tag{3}$$

$$\text{subject to: } \sum_{i=1}^{n} T_k(i) S_i \le C \qquad 1 \le k \le M \tag{4}$$

where C is a bounding constant. We can construct fitness function for knapsack problem as follows:

$$f(T_k) = J(T_k) + g(T_k), \qquad 1 \le k \le M \tag{5}$$

where $g(T_k)$ is the penalty function when T_k violates the constraints. It can take the form

$$g(T_k)=\begin{cases}0 & C-\sum_{i=1}^{n}T_k(i)S_i\geq 0\\ \beta E_m(C-\sum_{i=1}^{n}T_k(i)S_i) & C-\sum_{i=1}^{n}T_k(i)S_i<0\end{cases} \tag{6}$$

where E_m is the maximum value of $B_i/S_i(1\leq i\leq n)$, and β is a proper penalty coefficient.

For each individual solution, compute its fitness value $f(T_k),1\leq k\leq M.$

(4) Compute the survival probability P_k for each individual (solution) T_k by using the fitness proportional model:

$$P_k=f(T_k)/\sum_{j=1}^{M}f(T_j) \tag{7}$$

Then design a random selector and produce the hybridization individuals according to each P_k.

(5) Generate the solution set of next generation.

Select two hybridization individuals T_u and T_v, then combine them to get two other individuals T_u' and T_v' of new generation by using combinatorial rules of selection, crossover, mutation and inversion. This process continues until all M individual solutions of new generation are obtained. There are several genetic operators involved in this calculation.

Crossover is an operation of segment exchange for two solutions. Given two parents (hybridization individuals) solutions:

T_u=01010/1011100

T_v=10100/1101010

where " /" represents the crossover point. The crossover of above two solutions produces two children:

T_u'=01010/1101010

T_v'=10100/1011100

Inversion is to reverse the order of data in a solution segment. Given one parent solution

T_u=010/01101/1100

where the inversion segment is defined by left " /" and right " /". Performing inversion operation produces the following child:

T_u'=010/10110/1100

The mutation operator chooses one or more gene loci randomly in the individual string and changes (for example, 0-1 reverses) their values by the preset mutation probability. Given one parent solution

T_v=010101011100

two selected gene loci

Performing mutation operation produces the following child:

T_v'=011101010100

gene loci after mutation

(6) Repeat step (2) to step (5) until the predefined convergent condition is met, e.g., the maximum generation number is reached or the solution quality is satisfied.

A general C language description of the iterative process is given as follows:

```
Main()
{
        int gen_no;
        generate(oldpop);
        for(gen_no=0;gen_no<maxgen;gen_no++)
        {
                evaluate(oldpop);
                newpop=select(oldpop);
                crossover(newpop);
                mutation(newpop);
                inversion(newpop);
                oldpop=newpop;
        }
}
```

Above C language codes only describe major steps in genetic algorithm. To build a complete and practical application, other additional functions are needed. During the search process, genetic algorithm does not require any outside knowledge except fitness function to select qualified solutions. Therefore, the design of fitness function has direct impact on the algorithmic performance. Moreover, a proper data model will be very helpful for the implementation of the algorithm.

2.3 Discussion of key parameters and techniques in GA

2.3.1 Population size

Population size has significant impact on the final result and performance of genetic algorithm. According to the Schema Theorem [5], given the population size M, the genetic operators are able to produce M^3 schemas. Based on this fact, more and more building blocks can be generated and optimized till the optimal solution is found. Obviously, the larger the population size, the more likely to obtain the global optimal solution. The larger population size also means larger variety of individuals and less danger of being trapped into local optimization. However, the large population size may also bring some drawbacks. For instance, with the larger population size, the computation complexity will increase, and some good individuals with high fitness values may be eliminated during the selection operation [6]. On the other hand, small population size will limit the searching space and the premature convergence may occur under this circumstance, which may impair the performance greatly.

2.3.2 Crossover, mutation and inversion

Similar to the gene recombination, which plays an essential role during the process of natural biologic evolution, the crossover is the most significant operation in the genetic search strategy. It determines the major behavior of optimization process. Several crossover schemes are used, such as one-point crossover, two-point crossover and multi-point crossover. However, as a common criterion, any crossover operator should ensure that the proper genes of good individuals be inherited by the new individuals of next generation. A big crossover probability may improve genetic algorithm s capability to search new solution space, while increase the likelihood of disordering the combination of good genes. However, if the crossover probability is set too small, search process may be trapped in a dull status and is prone to ceasing.

The main function of mutation operation is to prevent losing single important gene segment to maintain the variety of solution population. Because frequent mutation operation may make the genetic algorithm tend to be random search, relatively small probability for mutation operation, e.g. 0.001, is favorable.

The inversion is a special form of mutation. It is designed to carry out reordering operation and improve the local search ability for genetic algorithm. The crossover operator has wide span of activity in the feasible solution space. Due to the pressure of genetic selection and natural elimination, it is also not easy for the mutation operation to carry out local search activity (especially at the late period when genetic algorithm tends to convergence) [5].

2.3.3 Encoding and fitness function

As genetic algorithm is unable to directly handle the parameters in the problem, it is necessary to convert them to the individuals made up of genes in genetic algorithm. This conversion is called encoding. Due to the robustness of genetic

algorithm, it doesn t have critical requirement for encoding technique, as long as minimum three encoding criteria, such as completeness, soundness and non-redundancy, are satisfied [7].

In general, genetic algorithm doesn t need outside information except the fitness function (or objective function) to control the convergence of computation. The objective function of genetic algorithm may have discontinuous or non-differentiable constraints. To handle the general optimization problem with constraints, the penalty method is often used in the design of fitness function. For example, an original minimization problem with constraint can be described as follows [5]:

$$\text{Minimize: } F(x) \tag{8}$$

$$\text{with constraints: } b_l(x) \geq 0 \qquad l = 1,2,\cdots,p \tag{9}$$

where $F(x)$ is the objective function and $b_l(x)$ are a group of constraint functions. By using the penalty method, the above problem can be converted to non-constraint problem:

$$\text{Minimize: } F(x) + \lambda \cdot \sum_{l=1}^{p} \Phi[b_l(x)] \tag{10}$$

where, λ is the penalty coefficient, and Φ is the penalty function, which may take the form of Eq.(6).

2.3.4 Selection mechanism

The selection operation is to select good individuals and at the same time eliminate bad individuals from population based on the evaluation of individual fitness. It is also called reproduction operation. Its aim is to inherit good individuals directly from last generation or indirectly from the new individuals produced by mating the old individuals. The frequently used selection mechanisms include fitness proportional model, rank-based model, expected value model and elitist model etc [2,5].

3 ALGORITHM IMPLEMENTATION

3.1 Data preprocessing

Data preprocessing is to reorganize the spatial data in GIS to facilitate the implementation of genetic algorithm. Spatial data is often organized in a way that will benefit their display and analysis in a GIS environment. Different GIS software products may have different types of data organization and management. These data models are useful for convenient layer display, but may not be suitable for the implementation of search strategies of genetic algorithm [8,9]. Therefore, the spatial data provided by GIS database need to be preprocessed and re-modeled before they can be used for network analysis with genetic algorithm.

3.2 Description of algorithm optimization process

3.2.1 Encoding of network problem

A successful encoding method will benefit the performance of the genetic algorithm. Assume that there are n nodes in the problem space. For each node, a unique number chosen from 1 to n is assigned to it. We use a string T, which contains the nodes numbers, to represent a candidate solution to the optimal route problem. For example, if the string T is [3,5,8,1,…,9], that means its corresponding route begins from node 3, passing through node 5, then node 8, 1, .., and ends at node 9.

3.2.2 Generating initial population and computing fitness values

Given the initial population size M and n the number of nodes in the network, a set of M strings of n dimension is generated randomly based on the above encoding method to form the initial solution. Each string $T_k \quad (k = 1,2,M)$ represents one candidate solution.

Construct objective function as follows:

$$\text{Minimize: } D(T) = \sum_{i=1}^{n-1} r_{i,i+1} d_{i,i+1} \tag{11}$$

$$\text{With constraint: } \sum_{i=1}^{n-1} d_{i,i+1} < C \tag{12}$$

Where, $d_{i,i+1}$ and $r_{i,i+1}$ are the distance and the resistance value between two neighbor nodes respectively, i is the index of node in a solution string T of n numbers, and C represents a constant length constraint of the whole path. The resistance value $r_{i,i+1}$ is retrieved dynamically from a pre-defined geographical information table during the optimization process. Through the penalty method, the above problem can be converted to non-constraint problem**:**

$$\text{Minimize: } D(T)+\lambda(\sum_{i=1}^{n-1} d_{i,i+1}-C) \tag{13}$$

Here, λ is a proper penalty coefficient. If one solution satisfies the constraint, i.e. $\sum_{i=1}^{n-1} d_{i,i+1} < C$, λ is set to be 0, otherwise λ carries a positive value. That means those good candidate solutions are not punished. Their fitness is determined only by their own objective function values.

Finally, the fitness function can be constructed as follows:

$$f(T)=[D(T)+\lambda(\sum_{i=1}^{n-1} d_{i,i+1}-C)]^{-1} \tag{14}$$

3.2.3 Selection, crossover, mutation and inversion

The individuals are selected randomly to perform the crossover, mutation and inversion operations. To maintain the same population size for each generation, the fitness of every newly generated child is compared with the minimum fitness of the whole population. If it is bigger than the minimum fitness, add this child to the population and remove the individual with the minimum fitness; otherwise, the new child will be eliminated.

Two-point crossover is applied in our case. These two points are selected randomly. Given two parents as follows:

8-2-7-3-/-5-1-6-/-4-9-10

1-3-5-2-/-9-6-4-/-10-7-8

Again, where " /" represents the crossover points, the crossover operator produces two children:

5-1-6-3-2-9-4-10-7-8

9-6-4-8-2-7-3-5-1-10

The crossover operation generates new individuals by placing the crossover portion at the beginning of the other individual string and removing the duplicate genes.

Two inversion points are also selected randomly to determine the inversion portion of the individual. The inversion portion is reorganized by reversing the order of the original individual. Given one parent as follows:

3-5-7-/-1-2-8-9-/-6-10-4

Where the inversion portion are defined by two " /" signs, the inversion operator produces the following child:

3-5-7-/-9-8-2-1-/-6-10-4

To perform the mutation operation, two mutation points are selected randomly and the values of these two points are exchanged. Given one parent as follows:

2-9-3-7-1-4-5-1-8-6-10

The mutation operator produces the following child:

2-9-5-7-1-4-3-1-8-6-10

3.3 Analysis of example

To illustrate the performance of search strategy for genetic algorithm, a network of 80 nodes is used for optimization analysis. The population size is set to be 80. The probability of crossover and mutation is set to be 0.95 and 0.005 respectively. The convergence condition is that the generation number reaches the maximum generation number 350. The following figures show the network graphs at generation 100, 199, 299 and 350 as the computation proceeds.

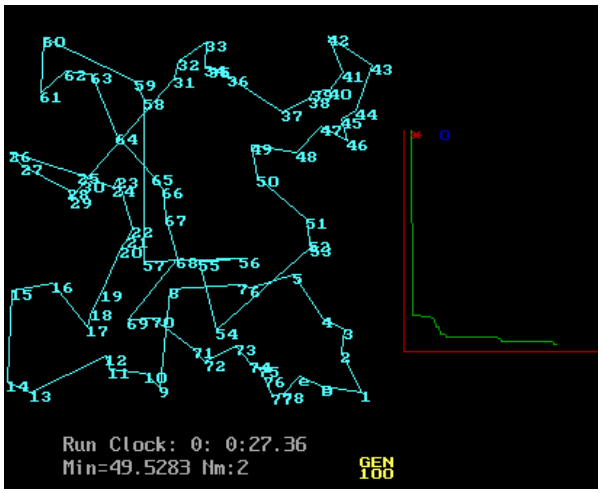

Figure 1. Network optimization graph with Gen=100

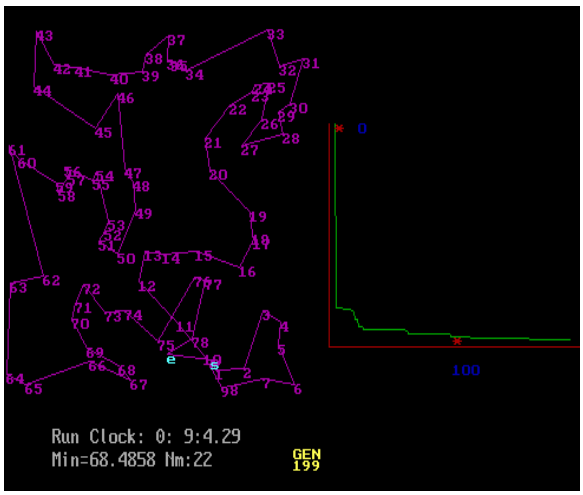

Figure 2. Network optimization graph with Gen=199

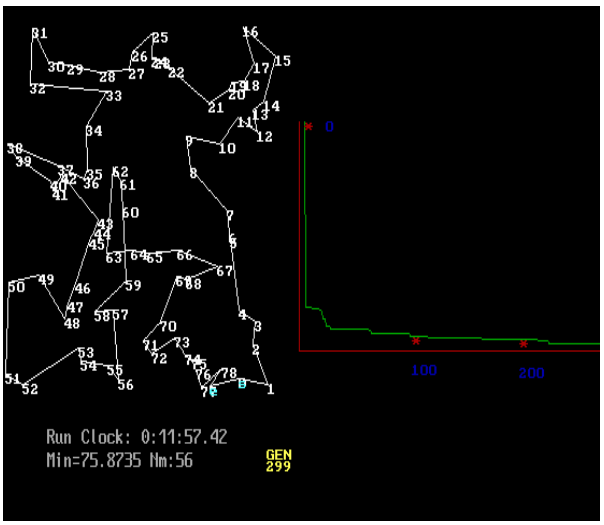

Figure 3. Network optimization graph with Gen=299

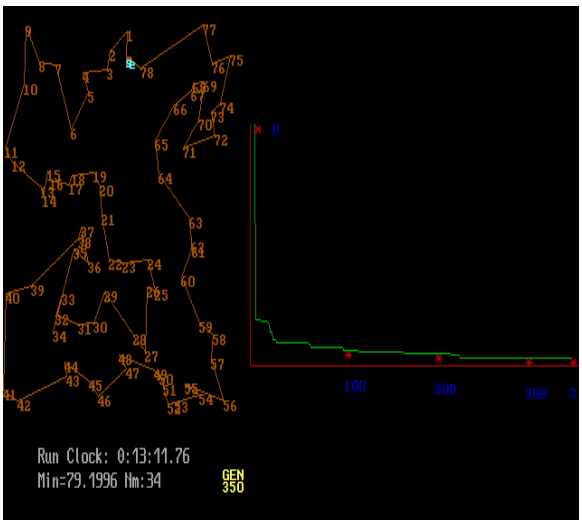

Figure 4. Network optimization graph with Gen=350

The curves on the right part of the above figures are the indicators of optimization process, which are related to the minimum distance of each generation. It shows that the optimization is proceeding rapidly in the beginning, and getting slow and stable at its late period. The difference of the indicator values between two consecutive generations can be used as an alternative convergence condition. The total CPU time is about 30 seconds on Pentium (II) 300 MHz. The last figure shows that the final result has approached the optimal route.

4 CONCLUDING REMARKS

As a high efficient search strategy for global optimization, genetic algorithm demonstrates favorable performance on solving the combinatorial optimization problems. The best route selection problem in network analysis can be solved with genetic algorithm through efficient encoding, selection of fitness function and various genetic operations. Crossover is identified as the most significant operation to the final solution. The solution process is robust and can rapidly lead to the optimal solution and remain stable there. Experience also shows that the designed implementation method is effective in terms of computation time and complexity. Tests of route selection for a moderate complicated network are conducted and their results show the efficiency of the algorithm and support our analyses. Further efforts will be made on the integration of genetic algorithm with commercial GIS packages, enhance the adaptability of the algorithm and expand its applications into broader GIS topics.

REFERENCES

[1] Winston, P.H., 1993, Artificial Intelligence, Third Edition, Addison-Wesley Publishing Company

[2] Han, Z.X., Wen, F.S., 1995, Optimization method simulating evolution and its application, Journal of Computer Science, Vol. 22 No. 2.

[3] Holland J.H., 1975, Adaptation in Nature and Artificial Systems. The University of Michigan Press, reprinted by MIT Press, 1992.

[4] Coley, D.A.., 1999, An Introduction to Genetic Algorithms for Scientists and Engineers, World Scientific.

[5] Chen, G.L., 1996, Genetic Algorithm and its application, People s Post Publishing House, China.

[6] Cartwright, H.M. and Mott, G.F., 1991, Looking Around: using Clues from the Data Space to guide Genetic Algorithm Searches. Proceedings of ICGA, pp.108-114.

[7] Goldberg, D.E., 1989, Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley Publishing Company.

[8] Sumic, Z., 1994, APR: A Geographic Information System Based Primary Router For Underground Residential Distribution Design, IEEE PES Transmission and Distribution Conference and Exposition, April 10-15.

[9] Sheble G.B., 1991, Distribution Automation Experiments with Generic Software, IEEE Computer applications in Power, July 1991.

[10] Cai, Z.X., 1996, Artificial Intelligence and its Application, Tsinghua University Press, China