Map algorithm in routing problems using genetic algorithm.

James Cunha Werner (wernerjc@sbu.ac.uk)
Terence C. Fogarty (fogarttc@sbu.ac.uk)
SCISM
South Bank University
103 Borough Road
London SE1 0AA

Abstract. This paper address a new routing problem generation algorithm for travelling salesman problem, delivery or vehicle scheduling attending to their constraints, where usually incompatible solutions are discarded or a punishment factor is applied, with waste computer processing.

Introduction

Delivery problem is an optimisation problem of assigning a route for a vehicle between some nodes.
The scheduling problem searches for solutions with minimal distance or time between fixed nodes for delivery attending some constraints: each node must be visited only and at least once. This constraint limits the acceptable solution space.
The generation of solutions must deal with impossible solutions, like visit a demand point more than one time and don't visit others points. Usually this solution after consistency is discarded, or a punishment factor is applied to the solution.
This paper proposes a new method where the sequence vector contains real numbers between 0 and 1, coding different nodes. Genetic Algorithm with a fixed size chromosome is used for search an optimal solution for a delivery problem with 10 nodes with is compared with Boltzmann Machine solution [1].

Genetic algorithms.

Genetic algorithms (GA) mimics the evolution and improvement of life through reproduction, when each individual contributes with its own genetic information building a new one with fitness to the environment and more surviving chances. These are the bases of genetic algorithms and programming ([2], [3] and [4]). Specialized Markov Chains underline the theoretical bases of this algorithm change of states and searching procedures.

Each 'individual' of the generation represents a feasible solution to the problem, coding distinct algorithms/parameters that should be evaluated by a fitness function.

GA operators are mutation (the change of a random position of the chromosome) and crossover (the change of slices of chromosome between parents).

The best individuals are continuously being selected, and crossover and mutation take place. Following few generations, the population converges to the solution that better attend the performance function.

Genetic algorithm concepts and implementation are structured in Fig.1.

Population initialization

Step 1 : Parent selection.

	First population	Objective value z=f(x,y)
	1100110110101000	3.481746
	0101010110110101	3.668023
==>	1000010100110110	6.261380
==>	1101011111001100	12.864222

Step 2: Crossover

11010	11111001100	------- \		11010	10100110110
10000	10100110110	------- /		10000	11111001100

Step 3: Mutation

1101010100110110	1000011111001100
1111010100100110	1000011111001100
z = 8.044	z = 6.092

Step 4: Reinsertion

	second population	Objective value z=f(x,y)
==>	1111010100100110	8.044
==>	1000011111001100	6.092
	1000010100110110	6.261380
	1101011111001100	12.864222

Step 5: If the result doesn't converge, go to step 1.

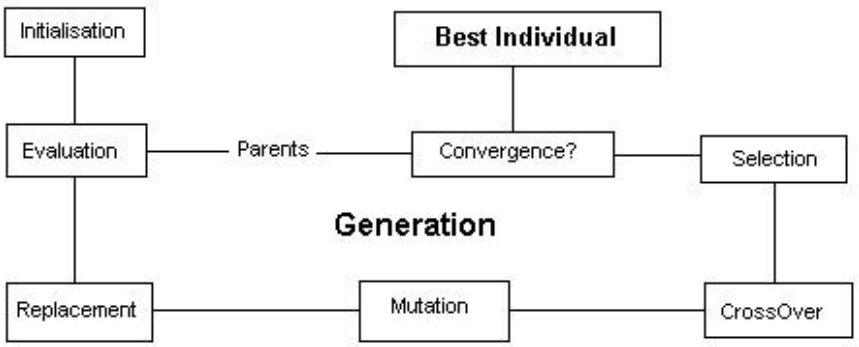

Fig. 1 General overview of genetic algorithms.

GA centre point consists in code problem variables in the right way and use its values to evaluate the fitness of each individual.

Coding the route using real numbers.

The chromosome could code the route sequence of nodes as a string varying between [0.0,1.0]. For example, for a 5 nodes route, let the chromosome value:

0.3	0.7	0.2	0.5	0.8

where each value represent the node sequence to be delivered.
Let "opcrot" a vector of possible nodes:

0	1	2	3	4

To obtain the relation between the chromosome vector (termed "rota") and the possible nodes, lets obtain the node pointer to "opcrot" vector:

$$\frac{(\max\ value - \min\ value)}{value} = \frac{number\ of\ nodes}{pointer} \quad \Rightarrow \quad pointer = \frac{value * number\ of\ nodes}{(\max\ value - \min\ value)}$$

Lets define the range for each value between [0.0,1.0], with means max value-min value=1.0. Then the pointer that relate the "opcrot" vector to each value of chromosome is:

$$pointer = value * number\ of\ nodes$$

Initially, number of nodes value 5, the first chromosome element value is 0.3, and pointer=5*0.3=1.5.
To use it as a pointer, take the integer part (1) and catch the value in "opcrot" as the first destination. Displace all posterior values left to remove the already selected element, and subtract 1 from number of nodes.
After that, "opcrot" contain:

0	2	3	4

Number of nodes=4;

The second value of the chromosome (0.7) results in pointer= 2.8 with select element 3 as second destination. For chromosome value 0.2 pointer=0.6 and selects 0, for 0.5 pointer=1 and selects 4, and finally select 2.
The route generated is: 1; 3; 0; 4; 2.
The C code for this routine and total distance evaluation is:

```
int rota[NumNos],opcrot[NumNos],ptr,i,j,contnos;
float distot;

contnos=NumNos;
for(i=0;i<NumNos;i++){
  rota[i]=0;
  opcrot[i]=i;
}
for(i=0;i<NumNos;i++) {
  ptr=varchr[i]*contnos;
  rota[i]=opcrot[ptr];
  for(j=ptr+1;j<NumNos;j++)
    opcrot[j-1]=opcrot[j];
  contnos--;
}
distot=0.0;
for(i=1;i<NumNos;i++)
  distot+=Distance[rota[i-1]][rota[i]];
```

where Distance is a matrix with distance between two nodes.

The Boltzmann Machine.

The Boltzmann machine is a stochastic version of the Hopfield model, whose network dynamics incorporate a random component in correspondence with a given finite temperature. Starting with a high temperature and gradually cooling down, allowing the network to reach equilibrium at any step, chances are good, that the network will settle in a global minimum of the corresponding energy function. This process is called simulated annealing. The network is then used to solve a well-known optimisation problem: The weight matrix is chosen such that the global minimum of the energy function corresponds to a solution of a particular instance of the travelling salesman problem.

Experimental comparison.

Lets suppose a net with 10 nodes, and the distances as show in table 1. The results for both approaches are in table 2.

Tab 1. Distances between two nodes (bold for minimal values).

	0	1	2	3	4	5	6	7	8	9
0	0.000000	**0.618034**	1.175571	1.618034	1.902113	2.000000	1.902113	1.618034	1.175571	0.618034
1	**0.618034**	0.000000	**0.618034**	1.175571	1.618034	1.902113	2.000000	1.902113	1.618034	1.175571
2	1.175571	**0.618034**	0.000000	**0.618034**	1.175571	1.618034	1.902113	2.000000	1.902113	1.618034
3	1.618034	1.175571	**0.618034**	0.000000	**0.618034**	1.175571	1.618034	1.902113	2.000000	1.902113
4	1.902113	1.618034	1.175571	**0.618034**	0.000000	**0.618034**	1.175571	1.618034	1.902113	2.000000
5	2.000000	1.902113	1.618034	1.175571	**0.618034**	0.000000	**0.618034**	1.175571	1.618034	1.902113
6	1.902113	2.000000	1.902113	1.618034	1.175571	**0.618034**	0.000000	**0.618034**	1.175571	1.618034
7	1.618034	1.902113	2.000000	1.902113	1.618034	1.175571	**0.618034**	0.000000	**0.618034**	1.175571
8	1.175571	1.618034	1.902113	2.000000	1.902113	1.618034	1.175571	**0.618034**	0.000000	**0.618034**
9	**0.618034**	1.175571	1.618034	1.902113	2.000000	1.902113	1.618034	1.175571	**0.618034**	0.000000

Table 2. Comparative results.

	Distance	Route									
Boltzmann	7.29	8	9	0	1	2	3	5	4	6	7
GA	5.56	2	3	4	5	6	7	8	9	0	1

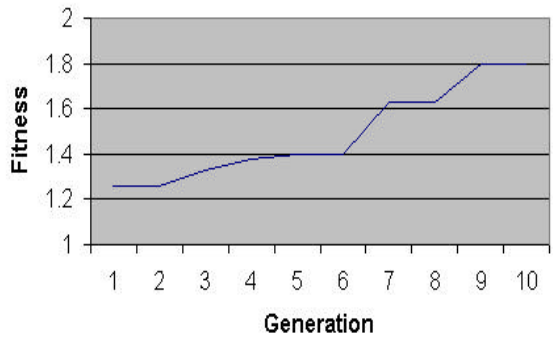

Fig. 2 Fitness evolution for genetic algorithm.

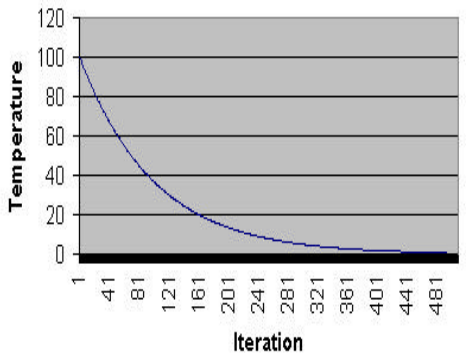

Fig. 3 Boltzmann Machine temperature.

Boltzmann Machine didn't find the optimal value like GA with route coding proposed.

Conclusion.

The introduction of a pointer function mapping the remainder nodes produce a performance improvement in genetic algorithm, with deal with constraints of scheduling problem.
The algorithm should be applied in all scheduling problem class, like vehicle scheduling, delivery, travelling salesman problem, etc with adequate customisation.

References.

[1] Karsten Kutza; *Neural Networks at your Fingertips*; http://www.geocities.com/CapeCanaveral/1624/

[2] HOLLAND,J.H. "Adaptation in natural and artificial systems: na introductory analysis with applications to biology, control and artificial intelligence." Cambridge: Cambridge press 1992 reedição 1975.

[3] GOLDBERG,D.E. "Genetic Algorithms in Search, Optimization, and Machine Learning." Reading,Mass.: Addison-Whesley, 1989.

[4] KOZA,J.R. "Genetic programming: On the programming of computers by means of natural selection." Cambridge,Mass.: MIT Press, 1992.