「정보시스템연구」 제17권 제3호
한국정보시스템학회
2008년 9월, pp. 39~58

GENIIS, a New Hybrid Algorithm for Solving the Mixed Chinese Postman Problem

최명길* · 응우엔만탕** · 황원주***

〈목 차〉

Ⅰ. Introduction

Chinese Postman Problem(CPP) tries to find the shortest path in which a postman starts from a node, visits each of edge (arc) of the network (graph) at least once, and returns to the starting node in a given network (Guan,1962). The purpose of CPP is to find a path in which postman visits all streets in a town at a minimum cost.

There are three basic kinds of Chinese Postman Problems, which consist of undirected Chinese Postman Problem (UCPP) for undirected graph (graph with edges), directed Chinese Postman Problem (DCPP) for directed graph (graph with arcs) and mixed Chinese Postman Problem (MCPP) for mixed graph (graph with both arcs and edges).

UCPP can be solved in a polynomial time by a algorithm based on Minimum Weighted Perfect Matching Problem. DCPP could be solved in polynomial time and Minimum Cost Maximum Flow Problem. MCPP is a generalized concept of Chinese Postman Problem for mixed graph (Edmonds 1965; Edmonds et al.1973; Guan,1962; Schrijver,2007). Although these two problems

* 중앙대학교, 경상학부 경영학과 조교수(주저자), mgchoi@cau.ac.kr
** 인제대학교, 시스템경영공학과 석사과정, nmtdhbk@yahoo.com
*** 인제대학교, 정보통신공학과 조교수, ichwang@inje.ac.kr

could be solved in polynomial time, MCPP on a mixed network could be considered NP-complete (Papadimitriou,1976; Zaragoza, 2003).

There are various kinds of CPPs such as Windy Postman Problem (NP-complete), Rural Postman Problem (NP-complete), and et al.. MCPP is a special case of other CPP NP-complete problems like Windy Postman Problem. Therefore, a solution utilizing a good method for MCPP can suggest a new approach for other versions of Chinese Postman Problem. In this paper, we propose an small effective searching space for a brute force algorithm and a genetic algorithm in order to solve MCPP in the space. To find near optimal solutions, we propose a hybrid algorithm, GENIIS(GENetic algorIthm with approxImate algorithmS). GENIIS is a kind of a hybrid algorithm composed of approximate algorithm and genetic algorithm.

In an effective searching space, GENIIS uses approximate algorithms to make some good individuals for the first population and runs genetic algorithm in a evolution process which consists of selection, crossover, and mutation until it finds best solutions. GENIIS utilizes the approximate algorithm suggested by Raghavachary & Veerasamy. For a genetic algorithm utilized in GENIIS, we devise an effective method for encoding gene and decoding gene.

To demonstrate the performance of our study, we simulate optimal solutions compared with other values produced by an approximate algorithms and a brute force algorithm. Chinese Postman Problem can be applied in many fields such as transportation, network routing, graph drawing, inspection of electronic power lines, routing of street sweepers, and circuit embedding(박송미 외, 2007). In terms of graph theory, the postman problem seeks a tour of graph with a minimum cost in a condition that a postman traverses all its arcs (one-way streets) and edges (two-way streets) in mixed graph at least once.

Ⅱ. Problem Definition

*Definition*1: Let $G=(V,E,A,w)$ be a connected, weighted graph. V,E,A could be defined as a set of vertexes, edges and arcs, respectively. In G, w is a weight function ($w: A \cup E \rightarrow R^{+}$). As figure 1 shows, the graph has 6 vertexes, 5 arcs, and 8 edges.

Let A be absent, then G is undirected graph. Let E is absent, then G is directed graph. Let both A and E be present, then G is mixed graph. G is called Eulerian if it has a tour that traverses all edges and arcs of G, at exactly one time.

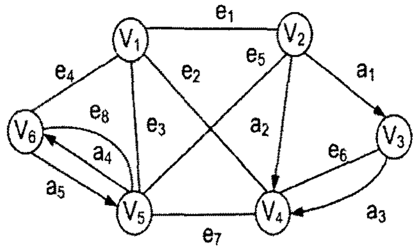


Figure 1. Mixed Graph with: 6 Vertexes, 5 Arcs and 8 Edges

Definition 2: Postman tour in graph $G=(V,E,A,w)$ is a tour that travels all edges and arcs of graph at least once.

Definition 3: The Chinese Postman Problem of weighted graph $G=(V,E,A,w)$ is to find an optimal postman tour that traverses all edges and arcs of G at least once.

Theorem 1: If G is a connected, it is undirected graph. G is Eulerian if and only if every vertex of G has even degree. Degree of one vertex is the number of edges connected to the vertex. However, a condition, that undirected and connected graph G is Eulerian, is too difficult to be satisfied in a real world. UCPP can be converted to Minimum Weighted Perfect Matching Problem. To make G become Eulerian graph or to make all vertexes become even vertex, we have to add more edges to G.

We define a new graph G_M (graph for match) which V_M has odd vertexes in G and has weight of edge $\{u,v\}$ in G_M , which is the shortest distance between u and v in G.

G_M, V_M, E_M, w_M are defined as followings:

- $G_M=(V_M, E_M, V_M)$
- $V_M=\{v: v\in V, v\ has\ odd\ \mathrm{degree}\}$
- $E_M=\{\{u,v\}: u,v\in V_M\}$
- $w_M(\{u,v\})=the\ shortest\ distance$ $between\ u$ and $v\ on\ G$

The shortest distance problem can be solved by Floyd-Warshall algorithm in polynomial time. We can find Minimum Weighted Perfect Matching Problem in graph G_M by Edmonds' Blossom Algorithm.

We operate the following: If $\{u, v\}$ is matched in Minimum Weighted Perfect Matching Problem in graph G_M, edges in the shortest path from u to v in graph G could be added, making graph for a minimum postman tour. After finishing the operation for all matched pair vertexes in Minimum Weighted Perfect Matching Problem of graph G_M, we can obtain a Eulerian graph, in which Euler tour is a minimum postman tour for graph G. Minimum Weighted Perfect Matching Problem can be solved in polynomial time. We, therefore, can set following theorems:

Theorem 2 (Edmonds): There exists a polynomial-time algorithm that computes a minimum cost postman tour of G for a given connected, undirected graph G. Whereas, DCPP is to find a postman tour that traverses all its arcs with a minimum cost in directed graph G. When G is Eulerian, a postman tour or Eulerian tour passes all arcs at exactly one time with a minimum postman tour.

Theorem 3: Let G be a connected and directed graph. If and only if every vertex of G has an equal indegree and outdegree, G is Eulerian. However, a condition that every vertex of G has an equal indegree and outdgree, is difficult to be satisfied in a real world. To solve DCPP, we have to add more arcs to G, resulting in making

Eulerian graph (every vertex of G having equal indegree and outdegree), which has a minimum weight from arcs of G. The problem could be applied to transportation algorithm or minimum cost maximum flow algorithm. An algorithm based on Minimum Cost Maximum Flow algorithm consists the following two steps:

$Step1$: Calculate the degrees of all vertexes as follows: *degree(v)=indegree(v)-outdegree(v).* The result is shown in figure 2.

$Step2$: Add a new source vertex, a new destination vertex and new arcs. To make flow graph, we have to assign capabilities and weights to the arcs of the extended graph and the directed graph as followings:

- Make new arcs from vertexes in G with degree< 0 to destination. Assign 0 as weight $(weight = 0)$ and assign the negative degree of that vertex to the capability for those arcs.
- Make new arcs from source to vertexes in G with degree> 0. Assign 0 as weight $(weight = 0)$ and assign the positive degree of that vertex to the capability for those arcs.
- For old arcs in G, the capability is infinite and the weight equals that of arc in G.

Finally, we obtain a flow graph for Minimum Cost Maximum Flow Algorithm. Figure 3 shows the alloted capabilities and degrees of the graph in figure 2.

We can define the processes mentioned above as followings:

- $G_F = (V_F, A_F, w_F, c_F)$
- $V_F = V \cup \{source, destination\}$
- $A_F = A \cup \{(source,v) : v \in V, \text{degree}(v) > 0\} \cup \{(v,destination) : v \in V, \text{degree}(v) < 0\}$
- $w_F(u,v) = w(u,v) \text{ if } (u,v) \in A,\ 0 \text{ if } u = source,\ 0 \text{ if } v = destination$
- $c_F(u,v) = \infty \text{ if } (u,v) \in A,\ \text{degree}(v) \text{ if } u = source,\ -\text{degree}(u) \text{ if } v = destination$

Flows in each arc of graph G present the number of the arc, which could be added to make the cost of postman tour minimum. The added arcs and graph G make graph G Eulerian graph.

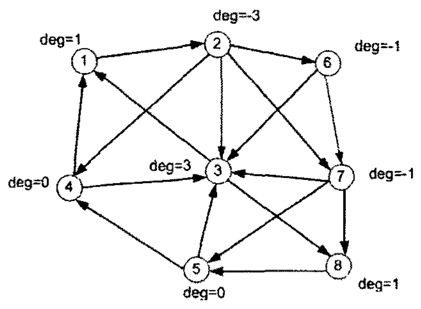


Figure 2. Directed Graph with Degrees of Vertexes

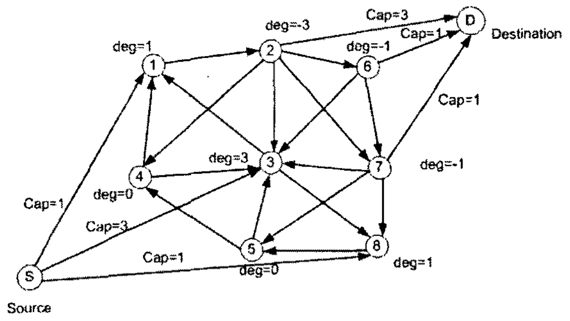


Figure 3. Flow Graph with Capabilities from Source to Destination

Minimum Cost Maximum Flow Problem can be solved in polynomial time. DCPP based on Minimum Cost Maximum Flow Algorithm is very useful for reducing searching space for MCPP and for encoding and decoding in Section 3.

UCPP and DCPP can be solved in polynomial time but MCPP is a NP-complete problem. Papadimitriou showed that MCPP is equal to 3 SAT problem. Because 3 SAT problem is NP-complete, MCPP is also NP-complete(Zaragoza, 2003).

To find an exact optimal postman tour solution with minimum weight, we can use a brute force algorithm which tries to find all feasible postman tours and choose the best one. A number of postman tours in a graph is infinite. To find the best solution efficiently, we show an effective searching space of MCPP and utilize GENIIS for solutions in the space. To find an optimal solution for MCPP, a brute force algorithm also can be applied in the effective searching space. However a brute force algorithm could not be solved in polynomial time. Because MCPP is a NP-complete problem, many researchers propose approximate algorithms. The approximate algorithm is defined as followings:

Definition 4: For minimum problem P, an assumption that f^* is an optimal solution of problem P and $r > 1$ is real number, can be given. An algorithm is called approximate algorithm with ratio r (or r-approximate algorithm) if and only if the solution f of algorithm satisfies the following constraint: $1 \le \frac{f}{f^*} \le r$.

The first approximate algorithm with ratio 2 came from Edmond. Based on the algorithm of Edmonds' algorithm, Fredrickson proposed $5/3$ approximate algorithm. Figure 4 shows

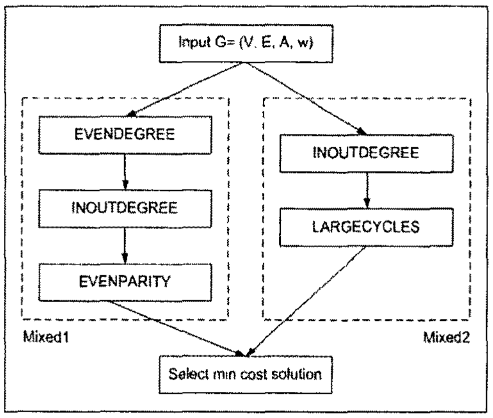


Figure 4. Fredrickson's $5/3$ Approximate Algorithm

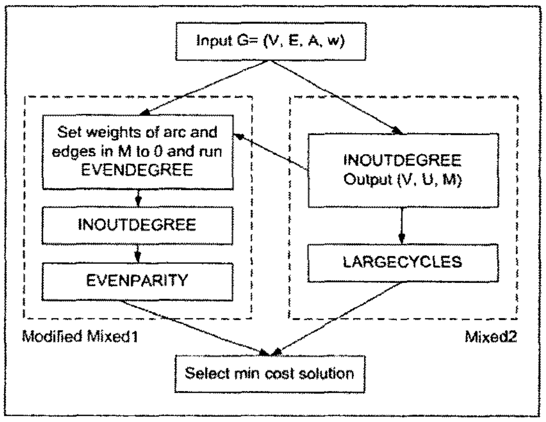


Figure 5. Raghavachary & Veerasamy's $3/2$ Approximate Algorithm

Fredrickson's algorithm(Federickson, 1979).

Raghavachary & Veerasamy improved Federickson's $5/3$ approximate algorithm to get $3/2$ approximate algorithm (figure5)(B. Raghavacharwe and J. Veerasamy, 2001). $3/2$ is the best approximate ratio for MCPP until now. The approximate solutions from approximate algorithm of Raghavachary & Veerasamy are included in the first population of genetic algorithm in our genetic algorithm.

Ⅲ. Literature Review

MCPP was proposed by Edmonds and et al.(Edmonds et al., 1973) and became a generalized concept of Chinese Postman Problem(Edmonds 1965; Guan, 1962; Schrijver, 2007). The polynomial algorithms for Chinese Postman Problem in undirected graph and directed graph have been issued by Edmonds & Johnson and Edmonds. However MCPP is a NP-complete problem which was already proved by Papadimitriou(Papadimitriou, 1976).

Nowadays, some researchers try to solve other Chinese Postman Problems such as Rural Postman Problem, Windy Postman Problem(Corberan et al., 1984; Corberan et al., 2000; Corberan et al., 2007), and k-Postman Problem. J.Zaragoza summarized all different Chinese Postman Problems(Zaragoza, 2003). G. Ghiani and G. Laporte propose a Branch and Cut Algorithm for the Undirected Rural Postman Problem(Ghiani et al., 2000; Laporte, 1997). E. Benavent and et al. introduce Lower Bounds and Heuristics for the Windy Rural Postman Problem(Benavent et al., 2007). M. Grotschel and et al. introduce Cutting Plane Algorithm for the Windy Postman Problem(Grotschel et al., 1992).

Some heuristic methods for searching space are also applied in Min Max k-Chinese Postman Problem such as Tabu Search Algorithm, and Heuristic and Lower Bound Algorithm(Ahr et al., 2002; Ahr et al., 2006; 조희연, 2003). New Chinese Postman Problems are considered difficult to solve because those were based on half integer.

Jun, ByungHyun and et al. show a traditional genetic algorithm that does not utilize approximate algorithm to reach approximate solutions or approximate individuals before applying genetic algorithm(Jun et al., 1998).

In this paper, we show an effective searching space for MCPP with 2^E elements. To solve MCPP, we try to find solutions in an effective searching space. Our approach utilizes approximate solutions which come from other algorithms for the first population in genetic step. After having good individuals from approximate solutions, genetic algorithm will be used to find better solution. GENIIS is an approach using genetic algorithm with $3/2$ approximate algorithm of Raghavachary & Veerasamy.

Ⅳ. A Suggested Solution for Mixed Chinese Postman Problem

4.1. An Effective Searching Space for MCPP

The number of postman tours in MCPP is infinite. To find an optimal solution for MCPP efficiently, we show that we can find an optimal solution for MCPP in an effective searching space with 2^E elements. Before we show searching space, we establish some definitions.

$Definition 5$: Assume that $E=\{e_1=\{u_1,v_1\},\ldots,e_m=\{u_m,v_m\}\}$ is a set of edges of graph G. A set of arcs, $AE=a_1a_2\cdots a_m$ can be called as one way assigning direction for E if $a_i=(u_i,v_i)$ or $a_i=(v_i,u_i)\ \forall\, i=1..n.\ AE$ can also be expressed to the following form. $s(AE)=s_1..s_m$ where if $a_i=(u_i,v_i)$ then $s_i=1$ and if $a_i=(v_i,u_i)$ then $s_i=0$.

Based on the definition, each edge in E has two ways to assign direction. There are 2^m or 2^E ways to assign direction in m edges. The set of all strings m bits has 2^m elements. The space of set $S=set\ of\ AE$ has 2^m elements. We prove that the space S is an effective searching space for MCPP to find an optimal solution.

Assuming that P is a postman tour of mixed graph G. It is easy to see that P includes a way to assign direction for E. Each postman tour has its direction based on the direction of arcs or the order of vertexes tour. Figure 6 shows examples of postman tour with AE and $s(AE)$.

In figure 6(a), the direction of a postman tour is $AE=\{a_1=(1,2),a_2=(4,1)\}$, which has a respective string $s(AE)=10$ for $E=\{e_1=(1,2),e_2=(1,4)\}$. As e_2 is repeated twice with two different directions in figure 6(b), a postman tour in b includes two ways to assign direction for $E=e_1=\{(1,2),e_2=(1,3),e_3=(1,4)\}$.

The first way is $AE=\{a_1=(1,2),a_2=(1,3),a_3=(4,1)\}$ with $s(AE)=110$ and the second way is $AE=\{a_1=(1,2),a_2=(3,1),a_3=(4,1)\}$ with $s(AE)=100$. Denote $G^*=(V,A^*,w^*)$ as directed graph made from mixed graph $G=(V,E,A,w)$, which has all edges in G becoming two arcs in G^* with two different directions.

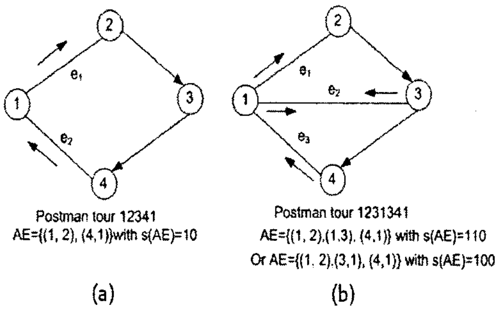

2
3
1
4
e1
e2
Postman tour 12341
AE={(1, 2), (4,1)}with s(AE)=10
(a)
2
3
1
4
e1
e2
e3
Postman tour 1231341
AE={(1, 2),(1,3), (4,1)} with s(AE)=110
Or AE={(1, 2),(3,1), (4,1)} with s(AE)=100
(b)

Figure 6. Postman Tours to Assign Direction and Strings to Assign Direction

We can define A^*, w^* as followings;

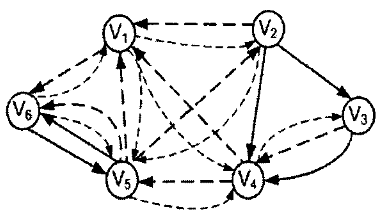


Figure 7. Graph G^*

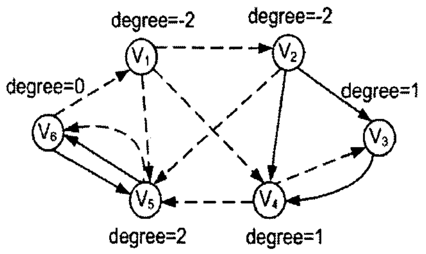


Figure 8. One Way to Assign Direction for Edges and Degrees of Vertexes

- $A^* = A \cup \{(u,v),(v,u) | \{u,v\} \in E\}$
- $w^*(a) = w(a), with\ a \in A$
- $w^*((u,v)) = w^*((v,u)) = .$ (Every edge $w(\{u,v\}), e = \{u,v\} \in E$

of G assigns two different directions for $G*$ with the same weight). The $G*$ can be shown as figure 7.

A postman tour P can be expressed by $P = A \cup AE \cup T$ with A(set of arcs in G), AE (one way to assign direction for E) and T (set of arcs in G^*, every arc can be repeated many times). To make a postman tour with $A \cup AE$, T has to be added. Because P includes $A \cup AE$, P traverses all arcs and edges of G at least once. Obviously there are many Ts to make $P = A \cup AE \cup T$ become a postman tour. If each AE and each T^* is a set of arcs in G^*. $P^* = A \cup AE \cup T^*$ is a postman tour and T^* has minimum weight. T^* exists for each AE. The weight of T is more than or equals to that of T^* in that T^* has a minimum weight. We have $w(P) \geq w(P^*)$.

Before showing an effective searching space, we define notations as followings;

- SP is a set of postman tours in G.
- SP^* is a set of P^*. SP^* includes 2^E elements because we have 2^E ways to assign direction for E.

Obviously, $\min SP \geq \min SP^*$. Because SP^* is a subset of SP, we have equation $\min SP = \min SP^*$. Finding an optimal postman tour in G is like finding an optimal postman tour in SP^*. Each element in SP^* can be expressed by an AE. SP has infinite elements and SP^* has 2^E elements. We can call SP^* as an effective searching space for MCPP.

$G' = (V, A \cup AE)$ is a directed graph and we have to find T^* in arcs of G^*. This is similar to DCPP. G^* is same to G and AE is absent in DCPP.

To find T^* for each AE, we can apply the same Minimum Cost Maximum Flow to G^* by the following steps.

$Step\ 1$: Assign direction for all edges as AE and calculate degrees of vertexes in $G' = (V, A \cup AE)$. The degrees of each vertex equal differences between indegree and outdegree of the vertex in $G' = (V, A \cup AE)$. Figure 8 shows one way to assign direction for edges and degrees of vertexes after assigning direction for graph in figure 1.

Step 2: Add a new source vertex, a new destination vertex, and new arcs to G^* and assign capabilities and weights to the arcs.

- Make new arcs from vertexes which have degree < 0 in $G' = (V, A \cup AE)$ to destination. Assign 0 as weights ($weight = 0$) and assign the negative degree (in $G' = (V, A \cup AE)$) as capability to the vertex for those arcs.
- Make new arcs from source to vertexes which have degree> 0 in $G' = (V, A \cup AE)$. Assign 0 as weight ($weight = 0$) and assign positive degree (in $G' = (V, A \cup AE)$) as capability to the vertex of those arcs.
- For old arcs in G^*, the capability is infinite and the weight equals that of arc in G^*.

Finally, the flow graph for Minimum Cost Maximum Flow Algorithm is shown in figure 9. The values of flows in G^* represent T^* and $P^* = A \cup AE \cup T^*$ is the best postman tour with minimum cost corresponding to AE. Based on the effective searching space SP^*, we can find an optimal postman tour for MCPP. T^* can be found by Minimum Cost Maximum Flow Algorithm in a polynomial time.

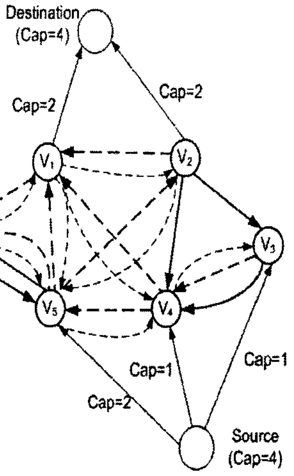


Figure 9. Flow Graph for One Way to Assign Direction

4.2. GENIIS Algorithm

GENIIS is a hybrid algorithm, which combines approximate algorithms and genetic algorithm., is able to improve good solutions which came from other algorithms. Figure 10 shows GENIIS. Genetic algorithm is a kind of searching technique to find approximate solutions for optimization and searching problems. To find approximate solutions in first population of genetic algorithm, GENIIS includes two good individuals which are approximate solutions came from Mixed1, Mixed2 and approximate algorithm with ratio $3/2$ of Raghavachary & Veerasamy. GENIIS can use other solutions from other approximate algorithms.

4.3. Genetic Algorithm in GENIIS

Individuals are encoded to describe a postman tour in MCPP. After initiating population at the

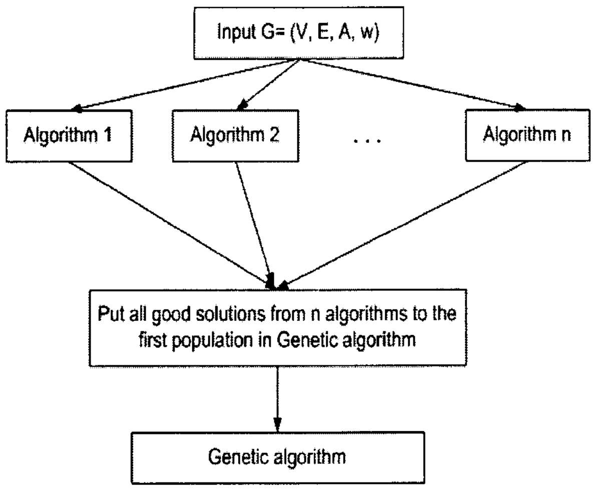

Input G= (V, E, A, w)
Algorithm 1
Algorithm 2
. . .
Algorithm n
Put all good solutions from n algorithms to the first population in Genetic algorithm
Genetic algorithm

Figure 10. GENIIS Combining Approximate Algorithms and Genetic Algorithm

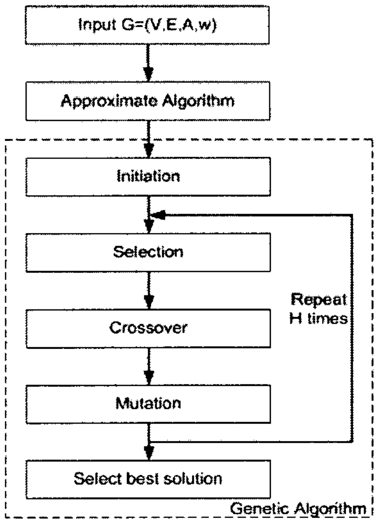


Figure 11. Processes of Genetic Algorithm for Finding Better Solution

first generation, the genetic algorithm in GENIIS operates selection, crossover and mutation steps for the purpose of finding better solutions shown in figure 11.

To find a near optimal solutions, GENIIS operates followings; First, the approximate algorithm of Raghavachary & Veerasamy seeks to find two approximate solutions. In the initiation step of the genetic algorithm, we create N individuals for the first population. The two approximate solutions are included as two individuals in the first population and other individuals except the two approximate solutions are generated, simultaneously. Second, the initiations of the first population evolution process are repeated H times after operations of selection, crossover, and mutation. The best solutions of populations after evolution processes are recorded as the output of the genetic algorithm. To make the final solution better than the first population, two approximate solutions are included in the first population.

4.3.1. Encoding and Decoding in the Genetic Algorithm

Encoding influences the efficiency of the genetic algorithm. SP^* has 2^E elements and each element P^* has an AE, respectively. To encode gene, we express P^* or AE by m bits $s(AE)$. To decode gene $s(AE)$, we find T^* based on Minimum Cost Maximum Flow algorithm and $P^* = A \cup AE \cup T^*$.

Gene encoding: Assume that $E = \{e_1 = \{u_1, v_1\}, .., e_m = \{u_m, v_m\}\}$ is a set of edges in mixed graph G. An T^* or P^* in SP^*, or AE (as one way to assign direction for E) can be expressed as $s(AE)$, respectively.

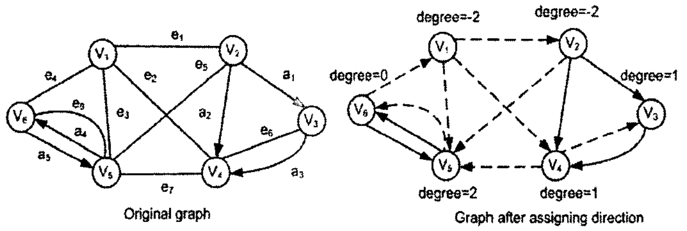


Figure 12. Encoding Gene

Assume that there are 8 edges with $e_1=\{1,2\}, e_2=\{1,4\}, e_3=\{1,5\}, e_4=\{1,6\}, e_5=\{2,5\}, e_6=\{3,4\}, e_7=\{4,5\}, e_8=\{5,6\}$ are shown in figure 13. If we assign direction for edges as figure 7, we can get the gene encoding as followings: $s(AE)=s_1s_2s_3s_4s_5s_6s_7s_8=11101011$.

Gene Decoding: $s(AE)=s_1s_2..s_m$ is a string to express AE (as one way to assign direction for E). To decode gene $s(AE)=s_1s_2..s_m$, we find T^* based on Minimum Cost Maximum Flow algorithm and $P^*=A\cup AE\cup T^*$.

Assume that we have encoded gene $s(AE)=s_1s_2s_3s_4s_5s_6s_7s_8=00010111$. Direction of edges can be assigned in figure 13.

4.3.2 Fitness of Individuals

For the fitness of individuals, we assign weights to the individuals. The weight of individual, $w(s(AE))=w(P^*)=w(A)+w(AE)+w(T^*)=w(G)+w(T^*)$, is assigned to an individual corresponding to a gene $s(AE)$. $w(P^*(AE))$ is a minimum postman tour with corresponding to AE. If an AE has minimum postman tour P^*, the fitness of individual is the inversion of the weight of individual. We can operation as following:

$$f_{fitness}(s(AE))=\frac{1}{w(P^*)}$$

Adjusting individual function: When we can find an arc (u,v) in AE. (v,u) appears more than one time in P^*. We replace (u,v) with (v,u) in AE to make AE' (a better way to assign direction). Adjusting individual function which can be applied as a mutation technique makes new individuals to be better value.

4.3.3. Selection Operation

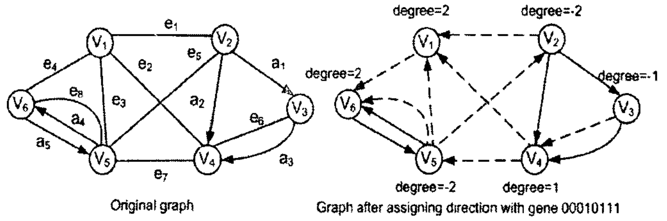


Figure 13. Decoding Gene 00010111

An assumption is that we have N individuals in population with finesses $f_1, f_2, .., f_N$. We applied four selection operations in the genetic algorithms. They are Roulette Selection, Competitive selection, Random selection and Selection with ratio g.

Roulette Selection: we set $r_0 = 0, r_i = r_{i-1} + f_i, i = 1...N$ and generate a random real number r in interval $[0, r_N]$. If $r \in [r_{i-1}, r_i]$, we select the i^{th} individual for crossover step.

Competitive selection: we choose random k individuals from old population and choose the individual with max fitness for the next step. The individual can be added to new population before mutation and crossover steps.

Random Selection: we choose individuals randomly from the population based on random function.

Selection with ratio g: the population after selection has only $g \times N$ individuals from the old population and remaining individuals are created randomly. The g parameter is used to increase random property of the population for the purpose of overcoming local extremity.

4.3.4. Crossover Operation

To make two new children individuals as new population from two parents individuals in the old population, we can use two crossover methods. They are 1 cut point crossover and 2 cut point crossover

1 cut point crossover: Choose random position in the gene strings of parents and exchange the parts of parents as shown in figure 14. Figure 14 shows an example, which has parents genes 10001000 and 00011111. We make a cut point in the fifth gene so that we can create two children which are 10010111 and 00011000, respectively.

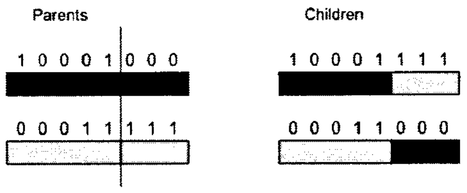


Figure 14. 1-Cut Point Crossover Operation

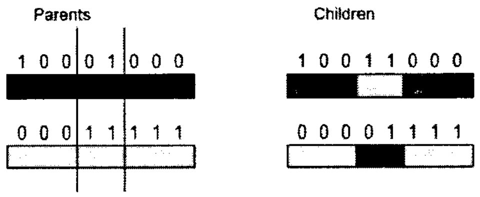


Figure 15. 2-Cut Points Crossover Operation

2 cut point crossover: Choose random 2 positions in the gene strings of parents and exchange the parts of parents as shown in figure 15. Figure 15 shows an example, which has parents genes 10001000 and 00011111. We make two cut point in the third gene and the fifth gene so that we can create two children which are 10011000 and 00001111.

Each gene is a string m bits that are easy to operate crossover function and mutation function.

4.3.5. Mutation Operation

We applied three kinds of mutation operations to the genetic algorithm. They are adjusted mutation, random mutation, and the combined mutation with adjusted mutation and random mutation.

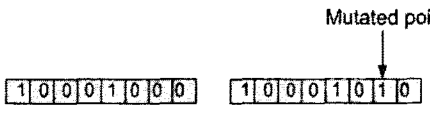


Figure 16. Reassign Direction for One Edge or Mutating One Chromosome

- Adjusted mutation: we travel all edges and reassign direction of edge in order to get the smaller value of a postman tour.
- Random mutation: If the probability of chromosome is $0 < p_c < 1$, we can make a random probability real number r in interval $[0, 1]$ with one edge or one chromosome of gene. If $r \le p_c$, we can reassign inverse direction.
- Combined mutation with adjusted mutation and random mutation: we can combine the two previous mutations. We inverse a chromosome in a condition that $r \le p_c$ makes a better postman tour. Each gene is a string m bits that are easy to operate the operations of crossover, mutation function.

The following code shows the pseudo code of the genetic algorithm in GENIIS.

```
Main program
Input : Graph G=(V,A,E,w)
Step 1:
//Execute approximate algorithms
MIXED2_(G)
Modify_MIXED1(G) // Because Modify_MIXED1 needs result from Mixed2
Step 2: Genetic Algorithm
Initiation(); // choose some approximate solution from approximate algorithms, other individuals is generated randomly.
Caculate_Weight_of_Individuals(); //calculate weights and record best temporary solution
Index_Number_of_Generation = 0;
For (Index_Number_of_Generation<H)
{
Selection(); // select individuals for crossover step
Crossover(); // crossover to get children
Mutation(); // execute mutation to get new individuals
Caculate_Weight_of_Individuals();
Index_Number_of_Generation=Index_Number_of_Generation +1;
}
SelectBestSolution(); //choose best solution and describe best solution
End
```

V. Simulation

To demonstrate the efficiencies of the proposed approach, we simulate the approach for MCPP and compare with the efficiencies of $3/2$-approximate algorithm suggested by Raghavachary &

Veerasamy. Table 1 shows the conditions of the simulation. To simulate test data, we make a simulation program using Visual C^{++} and run the simulation in desktop PC Pentium3 with 1.2 G/384 MB RAM. The simulation is conducted in a condition that the number of individuals in population is $N=2\times E$ and the number of repeated iteration is 50 times.

Table 2 shows the results of the simulation. We use $3/2$-approximate algorithm of Raghavachary & Veerasamy as the approximate algorithm. The brute force algorithm is a non-polynomial time algorithm that searches all solutions in the small searching space. It assigns direction for E and uses Minimum Cost Maximum Flow to find $P*$ for each AE. The brute force algorithm takes power time. In the effective searching space, the proposed approach uses approximate solutions from the approximate lgorithms and we input approximate solutions as the first population in the genetic algorithm.

We compare the efficiencies among the three algorithm. First, table 2 shows that the performance of the proposed approach for MCPP is better than those of the approximate algorithm. The costs of the proposed approach are always lower than those of the approximate algorithm. The results show that the performance of proposed approach is more efficient than that of approximate algorithm. Second, the performance of the brute force algorithm is more efficient than that of the proposed approach. But the problem is that the brute force algorithm is a non-polynomial time algorithm that searches all solutions in a small searching space. In other words, we are not able to use the brute force algorithm in that it takes much time to compute optimal solutions.

We can try to find a better postman tour in polynomial time, which can be acceptable in every cases. But we cannot use the brute force algorithm in many cases, especially in the large graph G. As the proposed approach shows that the efficiencies of solutions are better than those of the approximate algorithm and they are near an optimal solutions.

The complexity of GENIIS algorithm is $N\times H\times V\times(A+E)^2$. Whereas, the complexity of the brute force algorithm Force Algorithm is $2^E\times V\times(A+E)^2$ and that of the approximate algorithm is $V\times(A+E)^2$.

Figure 17 compares the best solutions among those of approximate algorithm, those of the brute force algorithm and those of the proposed approach in terms of costs of postman tour. As figure 17 and table 2 show, the proposed approach is able to find the approximate solutions which are near optimal solutions in a polynomial time. If we use the brute force algorithm, we spend power time to get an optimal solution. It is not assured to predict how much time is taken to get the results.

The results for mutation operations in the simulation are described in Figure 18. Random mutation operation makes better value than other two mutations in that two remaining mutation operations can guide the genetic algorithm to local extreme. Random mutation operation can increase

Table 1. Conditions of Simulation

Test	V	E	A
Test1	8	19	34
Test2	8	16	32
Test3	10	23	50
Test4	11	12	7
Test5	14	11	10
Test6	18	14	15
Test7	20	28	7
Test8	23	32	7
Test9	26	13	26
Test10	30	15	30

Table 2. Best Solutions for Algorithms

Test	Approximate Algorithm		Brute Force Algorithm		The Suggested Approach	
	Cost	Time (ms)	Cost	Time (ms)	Cost	Time (ms)
Test1	1,926	60	1,642	82,390	1,682	260
Test2	1,830	0	1,334	10,780	1,404	335
Test3	2,950	50	2,318	3,363,030	2,328	350
Test4	1,242	32	952	1,875	989	532
Test5	528	0	430	455	455	70
Test6	1,751	54	1,297	5,587	1,327	254
Test7	608	140		-	461	2,140
Test8	3,056	235		-	2,456	2,351
Test9	287	313	204	12,323	228	3,131
Test10	364	547	276	67,110	276	3,547

various properties of population after mutation. Random mutation is easy to overcome local extremity.

In the experiment of the genetic algorithm, which does not utilize the approximate solutions from the approximate algorithm, we can not get better solutions than the approximate solutions in many cases. In the approximate algorithms, the solutions of postman tours in many cases could be saved 70% comparing with the optimal solutions of the brute force algorithm in terms of assigning direction for edges. The approximate solutions from the approximate algorithms are useful for genetic algorithm in searching an optimal solution in the solution space.

To increase the effectiveness of our study, we try to compare the proposed approach with the other known algorithms which utilize genetic algorithm. Due to absence of information concerning test data, program and simulation environments, we just compare the proposed approach with known algorithms, indirectly. In the

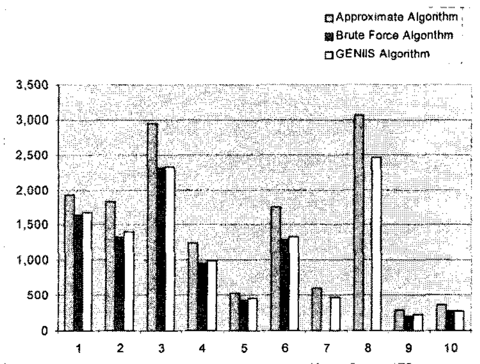


Figure 17. Comparison of Algorithms

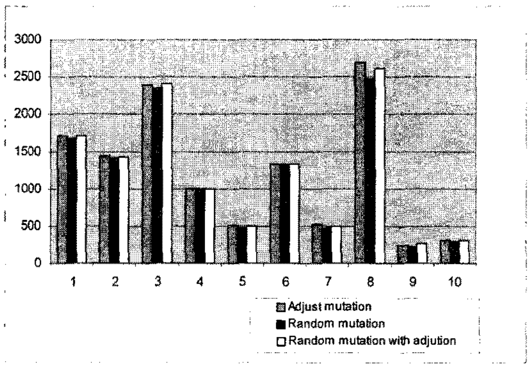


Figure 18. Comparison of Mutation Operations

study of Jun et al, they conducted experiment in small tests (Table 3) and compared with Mixed2 algorithm(Jun et al.,1998). In Jun and et al.'s study, they simulate their algorithm with the 100 number of individuals ($N = 100$), and the 1,000 number of generation ($H = 1000$).

When we consider the attributes of the graph in their study, the parameters are much higher than those in the our simulation. It is possible for them to produce more efficient time in their proposed algorithm. The objective function in MCPP is to minimizing cost of postman tour, in which time could be acceptable as polynomial time. In the result of Jun and et al.'s simulation (Table 4), their algorithm just improved computing time but the cost is almost same with Mixed 2. The minimum solution of Mixed 1 and Mixed 2 (as shown in Federickson's approximate algorithm and Raghavachary & Veerasamy's approximate algorithm) is better than that of Mixed 2. Our algorithm uses both Mixed 1 and Mixed 2 and Raghavachary & Veerasamy's approximate algorithm and utilizes genetic algorithm to improve solutions of Mixed 1 and Mixed. It is also difficult compare their proposed algorithm with other algorithm due to the absence of information.

Table 3. Test Data in Jun et al.

Problem	Number of Nodes	Number of Arcs	Number of Edges
1	6	3	6
2	9	6	6
3	6	2	6
4	9	8	8
5	9	11	6
6	11	10	16
7	31	21	18
8	16	4	19
9	17	4	21
10	18	3	25

Table 4. Result of Genetic Algorithm with Mixed2 Algorithm in Jun et al.

Problem	Number of Nodes (Mixed 2)		Number of Arcs (Jun et al.)	
	Cost	Time(sec)	Cost	Time(sec)
1	45	0.571	45	0.1
2	96	0.139	96	0.23
3	81	0.35	64	0.1
4	82	1.78	80	0.24
5	117	1.312	118	0.41
6	228	4.176	222	2.194
7	178	12.488	175	10.5
8	64	9.117	62	3.256
9	58	20.611	59	3.107
10	55	24.606	55	9.23

Ⅵ. Conclusion

To find an optimal solution in MCPP, we propose an effective searching space for MCPP and GENIIS. The effective searching space has 2^E elements and is easy to describe elements with string $s(AE)$. The proposed approach is a hybrid approach which combines the approximate algorithm of Raghavachary & Veerasamy and the Genetic Algorithm. The genetic algorithm continues to find an optimal solution among populations. The genetic algorithm can create a unique method of encoding gene and decoding gene. The usefulness of the encoding method is that it helps to find optimal solutions in an effective set of postman tours SP^* $(set\ of\ P^*)$ and use Minimum Cost Maximum Flow Algorithm to find P^* in polynomial time. The method of encoding gene and decoding gene can be used for some special cases as Rural Postman Problem, Windy Postman Problem, k-Postman Problem and the other Chinese Postman Problems.

To demonstrate the efficiencies of the proposed approach, we compare the proposed approach with the approximate algorithms and the brute force algorithm. The results of simulation can validate the usefulness of the proposed approach in terms of cost and tims. The costs of the proposed approach is lower than those of the approximate algorithm. But the costs of the proposed approach is near to those of the brute force algorithm. But It takes much for brute force algorithm to simulate the same environments than the proposed approach. Time taken in the brute force algorithm is power time so that it could be applied to only small graph. When the brute force algorithm is applied to a real application, it is difficult to predict consuming time.

References

조희연, “유전자 알고리즘을 이용한 주식투자 수익률 향상에 관한 연구”, 정보시스템연구, 제12권, 제2호, 2003, pp. 10-28.

박송미, 채영신, “프로젝트 위험요인 인식에 관한 비교 연구”, 정보시스템연구, 제16권, 제4호, 2007, pp. 243-268.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B., *Network Flows, Theory, Algorithms, and Applications*, Prentice Hall, 1993.

Ahr, D. and Reinelt, G., "A Tabu Search Algorithm for the Min Max k-Chinese Postman Problem," *Computers & Operations Research*, Vol. 3, No. 12, 2006, pp. 3403-3422.

Ahr, D. and Reinelt, G., "New Heuristics and Lower Bounds for the Min Max k-Chinese Postman Problem," *Lecture Notes in Computer Science*, Vol. 2461, 2002, pp. 64-74.

Benavent, E., A., Corberan, A., Sanchis, J. M., and Vigo, D., "Lower Bounds and Heuristics for the Windy Rural Postman Problem," *European Journal of Operational Research*, Vol. 176, No. 2, 2007, pp. 855-869.

Christofides, N., Benavent, E., Campos, V., Corberan, A., and Mota, E., "An Optimal Method for the Mixed Postman Problem, System Modelling and Optimization," *Lecture Notes in Control and Information Sciences*, Vol. 59, 1984, pp. 641-649.

Corberan, A., Plana, I., Reinelt, G., and Sanchis, J. M., "New Results on the Windy Postman Problem," *21st European Conference on Operational Research Technical Reports*, 2007.

Corberan, A., and Sanchis, J. M., *"A GRASP for the Mixed Chinese Postman Problem,"* Technical Report, University of Valencia, 2000.

Edmonds, J., "The Chinese's Postman Problem," *Operation Research.* Vol. 13:B-73, S.1, 1965.

Edmonds, J., and Johnson, E. L., "Matching, Euler Tours and the Chinese Postman," *Mathematical Programming*, Vol. 5, 1973, pp. 88-124.

Frederickson, G. N., "Approximation Algorithms for Some Postman Problems," *Journal of the ACM*, Vol. 26, No. 3, 1979, pp. 538-554.

Ghiani, G., and Laporte, G., "A Branch and Cut Algorithm for the Undirected Rural Postman Problem," *Mathematical Programming*, Vol. 87, No. 3, 2000, pp. 467-481.

Guan, M., "Graphic Programming Using Odd and Even Points," *Chinese Mathematics*, Vol. 1, 1962, pp. 273-277.

Grotschel, M., and Win, Z., "A Cutting Plane Algorithm for the Windy Postman Problem," *Mathematical Programming*, Vol. 55, No. 3, 1992, pp. 339-358.

Jun, ByungHyun and Han, ChiGeun, "Solving the Chinese Postman Problem on Mixed Networks Using an Efficient Genetic Algorithm," *Proceeding of the 1st Korea-Japan Joint Conference on Industrial Engineering and Management*, 1998, pp. 121-124.

Laporte, G., "Modelling and Solving Several Classes of Arc Routing Problems As Travelling Salesman Problems," *Computers & Operations Research*, Vol. 24, No. 11, 1997, pp. 1057-1061.

Papadimitriou, C. H., "On the Complexity of Edge Traversing," *Journal of the ACM*, Vol. 23, No. 3, 1976, pp. 544-554.

Raghavachary, B., and Veerasamy, J., "Approximation Algorithms for the Mixed Chinese Postman Problem," *Lecture Notes in Computer Science*, Vol. 1414, 1998, pp. 169-179.

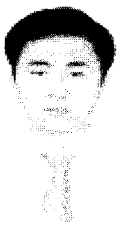
Schrijver, A., *A Course in Combinatorial Optimization Lecture*, 2007.

Zaragoza, F. J., *Postman Problems on Mixed Graphs, Thesis*, 2003.

최명길(Choi, Myeonggil)

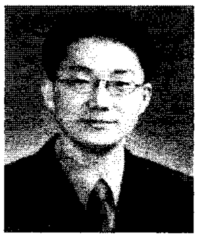
부산대학교에서 경영학사·석사, 한국과학기술원에서 박사를 취득하였다. 국방과학연구소, 한국전자통신연구원, 인제대학교 등에서 재직하였고, 현재 중앙대학교 경영학과 조교수로 재직하고 있다. 주요 관심분야는 정보보호관리, 정보시스템 보안 평가, 산업 보안 등이다.

응우엔 만 탕 (Nguyen Manh Thangi)



Hanoi University of Technology에서 공학사를 취득하였다. 현재 인제대학교 시스템경영공학과 석사과정에 재학하고 있다. 주요 관심분야는 네트워크 정보보호, 알고리즘, 홈 네트워크 정보보호 등이다.

황원주 (Won-Joo Hwang)



부산대학교에서 컴퓨터공학사·석사, 오사카대학 정보시스템 공학박사를 취득하였다. 현재 인제대학교 정보통신공학과 조교수로 재직하고 있다. 주요 관심분야는 홈 네트워크 정보보호, USN/RFID 보안 등이다.

<Abstract>

GENIIS, a New Hybrid Algorithm for Solving the Mixed Chinese Postman Problem

Myeonggil Choi · Nguyen Manh Thang · Won-Joo Hwang

Mixed Chinese Postman Problem (MCPP) is a practical generalization of the classical Chinese Postman Problem (CPP) and it could be applied in many real world. Although MCPP is useful in terms of reality, MCPP has been proved to be a NP-complete problem. To find optimal solutions efficiently in MCPP, we can reduce searching space to be small effective searching space containing optimal solutions. We propose GENIIS methodology, which is a kind of hybrid algorithm combines the approximate algorithms and genetic algorithm. To get good solutions in the effective searching space, GENIIS uses approximate algorithm and genetic algorithm. This paper validates the usefulness of the proposed approach in a simulation. The results of our paper could be utilized to increase the efficiencies of network and transportation in business.

Keywords: Hybrid Algorithm, Search Space, GENIIS, Mixed Chinese Postman Problem, Genetic Algorithm, Approximate Algorithm

* 이 논문은 2008년 3월 27일 접수하여 2차 수정을 거쳐 2008년 7월 10일 게재 확정되었습니다.