Research Note

A Hybrid Meta-Heurisitic Algorithm for Vehicle Scheduling Problem

— Genetic Algorithm and Tabu Search —

Sangheon Han

Abstract

The delivery problem is formulated and an application package is developed for the vehicle routing, scheduling with single depot or multiple depots. A Hybrid Model is proposed and constructed as a optimization tools in these years. In this paper, we propose a Hybrid Meta-Heuristic Method (HMHM) consisting of Gentic Algorithm and Tabu Search Algorithm. The HMHM is an efficient and flexible tool in NP-Hard Problems. Our objective is to construct optimal route to minimize the number of vehicles and vehicle movements. A detailed numerical study is conducted and its results show the advantages of our proposed algorithm.

Keywords : *Vehicle Scheduling Problem, Hybrid Meta-Heurisitic Algorithm, Genetic Algorithm, Tabu Search*

1 Introduction

Genetic Algorithms (GAs) have been demonstrated to be a promising search and optimization technique [11]. It has been successfully applied to system identification [19], [27] and a wide range of applications including filter design [14], scheduling [37], routing [5], control [32], and others [25], [36].

One of the main obstacles in applying GAs to complex problems has often been the high computational cost due to their slow convergence rate. The convergence rate of a GA is typically slower than that of local search techniques, because it does not use much local information to determine the most promising search direction. Consequently, a GA explores a wider frontier in the search space in a less directional fashion.

The Tabu Search Algorithm has established its position as an meta-algorithm guiding the design and implementation of algorithms for the solution of large scale combinatorial optimization problems in a number of different areas [20]. A key reason for this success is the fact that the algorithm is sufficiently flexible to allow designers to exploit prior domain knowledge in the selection of parameters and sub-algorithms. Such prior domain knowledge is frequently obtained by repeatedly solving a few representative test cases using a spectrum of different techniques, attributes and/or parameters.

Vehicle Scheduling Problems (VSP) are all around us in the sense that many consumer products such as soft drinks, beer, bread, gasoline and pharmaceuticals are delivered to retail outlets by a fleet of trucks whose operation fits the vehicle scheduling model. In practice, the VSP has been recognized as one of the great success stories of operations research and it has been studied widely since the late fifties. Public services can also take advantage of these systems in order to improve their logistic chain, garbage collection, or budget of local authorities.

A typical vehicle scheduling problem can be seen as the problem of designing minimum cost routes from one depot to a set of geographically scattered points (cities, stores, warehouses, schools, customers and so on). The routes must be designed in such a way that each point is visited only once by exactly one vehicle, all routes start and end at the depot, and the total demands of all points on one route must not exceed the capacity of the vehicle.

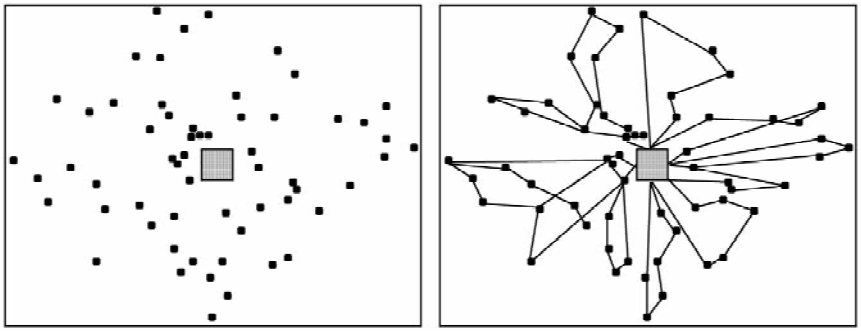
Besides being one of the most important problems of operations research in practical terms, the vehicle scheduling problem is also one of the most difficult problems to solve. It is quite close to one of the most famous combinatorial optimization problems, The Travelling Salesperson Problem (TSP), where only one person has to visit all customers. The TSP is an NP-hard problem. It is believed that many never find a computational technique that will guarantee optimal solutions to larger instances for such problems. The vehicle scheduling problem is even more complicated. Even for small fleet sizes and a moderate number of transportation requests, the planning task is highly complex. Hence, it is not surprising that human planners soon get overwhelmed, and must turn to simple, local rules for vehicle routing. Next we will describe the basic principle of genetic algorithms and some applications for vehicle scheduling problems.

The formation of vehicle routes is a problem that has engaged the attention of researchers in distribution management for over two decades. This paper proposes a developed method with a solution procedure based on a genetic algorithm and tabu search algorithm. Moreover, we developed a hybrid approach that combines GA with TS to solve VSP. Overall, computational results showed that our hybrid approach is an effective and robust optimization technique.

Trials on a sample problem suggest that the proposed algorithm is a powerful tool that can be successfully employed in a real distribution environment.

2 The Vehicle Scheduling Problem and Problem formulation

The Vehicle Scheduling Problem (VSP) was originally proposed by Dantzig and Ramser [7] and defined as follows: vehicles with a fixed capacity Q must deliver order quantities q_i ($i = 1,...,n$) of goods from a single depot ($i = 0$) to n customers. Knowing the distance d_{ij} between customers i and j ($i, j = 0,...,n$), the objective of the problem is to minimize the total distance travelled by the vehicles in such a way that only one vehicle handles the deliveries for a given customer and the total quantity of goods that a single vehicle delivers is not larger than Q.



(a) A Vehicle Scheduling Problem (b) A possible solution

Fig. 1 An example of a VSP

Figure 1 gives a graphical representation of a VSP and one possible solution. The square (in the middle of Fig 1 (a) and (b)) represents the base (where the trucks start and finish their tour) and the diamonds represent the sub-routes. Figure 1 (b) shows the tours of the different trucks. It should be observed that, in this case, all the customers have been allocated a visit.

Problem formulation

Let $G = (V, A)$ be a graph with a set V of vertices and a set A of arcs. We have $V = 0 \cup N$, where 0 corresponds to the depot

and $N = 1,...,n$ is the set of customers. For the set of arcs, we have $A = (\{0\} \times N) \cup I \cup (N \times \{0\})$, where $I \subseteq N \times N$ is the set of arcs connecting the customers, $\{0\} \times N$ contains the arcs from the depot to the customers, and $N \times \{0\}$ contains the arcs from the customers to the depot. Every customer $i \in N$ has a positive demand q_i. For each arc $(i, j) \in A$ we have a cost c_{ij}. Furthermore, we assume that the vehicles are identical and have the capacity Q. All the above mentioned factors are assumed to be known in advance. Thus the model examined is deterministic.

We have the following variables: For each customer $i \in N$, y_i is the load of the vehicle when it arrives at the customer. Now the problem is to determine which of the arcs $(i, j) \in A$ are used by routes. For each arc $(i, j) \in A$, the decision variable x_{ij} is equal to 1 if arc (i, j) is used by a vehicle and 0 otherwise. Formally

$$\text{Minimize} \quad \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{j\in V} x_{ij} = 1 \qquad \forall i \in N \tag{2}$$

$$\sum_{j\in V} x_{ji} = 1 \qquad \forall i \in N \tag{3}$$

$$x_{ij} = 1 \Rightarrow y_i - q_i = y_j \qquad \forall (i,j) \in I \tag{4}$$

$$q_i \le y_i \le Q \qquad \forall i \in V \tag{5}$$

$$x_{ij} \in \{0, 1\} \qquad \forall (i,j) \in A \tag{6}$$

We minimize the total costs that consist of travel costs and a fixed cost c of vehicles (included in the travel cost c_0 between depot and first customer). The object is, firstly minimize the number of routes or vehicles, and then the total distance of all routes. By equation (2), (3) and (6), we require that every customer be visited exactly once. Equation (4), (5) enforce the condition that the loads of the vehicles when arriving at the customers are feasible.

3 Application for the vehicle scheduling problem

Potvin and Bengio [33] propose a genetic algorithm GENEROUS that directly applies genetic operators to the solution, thus avoiding the coding issue. The initial population is created with the cheapest insertion heuristic of Solomon [37] and the fitness values of the proposed approach are based on the number of vehicles and total route time. The selection process is stochastic and biased toward the best solutions. For this purpose a linear ranking scheme is used. During the recombination phase, two parent solutions are merged into a single one, so as to guarantee the feasibility of the new solution. Two types of crossover operators are used to modify a randomly selected route or to insert a route into the other parent solution. A special repair operator is then applied to the offspring to generate a new feasible solution. Mutation operators are aimed at reducing the solution, a mutation operator based on Or-opt exchanges (Or [24]) is included.

GAs have been used with great success to solve several problems with high degrees of complexity in Combinatorial Optimization, including models of daily VSP [3], [29]. More recently, Rocha, Ochi and Glover proposed a Hybrid Genetic Algorithm (HGA) for the VSP. Their algorithm is based on concepts of GA and local Search Heuristics. Computational experiments using tests available in the literature showed the superiority of this method when it is compared to other existing meta-heuristics.

4 The Problem solving Methodology

4.1 General principles of Genetic Algorithms

The Genetic Algorithm (GA) is an adaptive heuristic search method based on population genetics. The basic concepts were developed by Holland (1975), while the practicality of using the GA to solve complex problems is demonstrated in Dejong (1975) and Goldberg (1989). References and details about genetic algorithms can also be found for example in Alander (2000) and Mühlenbein (1997) respectively.

The creation of a new generation of individuals involves primarily four major steps or phases: representation, selection, recombination (crossover), and mutation. The representation of the solution space consists of encoding significant features of a solution as a chromosome defining an individual member of a population. Typically pictured by a bit of string, a chromosome is made up of a sequence of genes, which capture the basic characteristics of a solution.

The recombination or reproduction process makes use of genes of selected parents to produce offspring that will from the next generation. It combines characteristics of chromosomes to potentially create offspring with better fitness. As for mutations, it consists of randomly modifying gene(s) of a single individual at a times to further explore the solution space and ensure, or preserve, genetic diversity. The occurrence of mutation is generally associated with low probability. A new population replaces those from the old one. A proper balance between genetic quality and diversity is therefore required within the population in order to support an efficient search.

Although theoretical results that characterize the behaviour of the GA have been obtained for bit-string chromosomes, not all problems lend themselves easily to this representation. This is the case, in particular, for sequencing problems, like vehicle the scheduling problem, where an integer representation is more often appropriate. We are aware of only one approach by Thangiah (1995), that uses bit-string representation in a vehicle scheduling context.

A basic scheme of a typical algorithm is as follows.

Randomly create an initial population
While not (termination condition) **do**
    Evaluate each member's fitness
    Kill the bottom x% elements of the population
    Let the fitness reproduce themselves
    Randomly select two members/parents (many other selection methods are also used)
    Perform crossover on the selected elements to generate two children
    (many variations of crossover exist)
Endwhile

Next we describe the basic principles of the Hybrid genetic algorithms used to solve the vehicle scheduling problem.

4.2 Types of Hybrid Architecture

There are many different definitions for a hybrid system. Our definition is one that uses more than one problem-solving technique in order to solve a problem. This immediately leads to defining a technique. It's hard to precisely formulate what is meant by a technique, but as far as we are concerned each of the following are individual techniques:

・Supervised neural networks

- Unsupervised neural networks
- Regression techniques
- Data reduction techniques
- Fuzzy logic
- Genetic algorithms
- Case-based reasoning

This list is certainly not intended to be complete. It is just a list of the techniques that we are interested in. Other techniques that can be used in hybrid systems include:

- Expert systems
- Regression and decision trees
- Clustering techniques
- Artificial life
- Simulation techniques

There are, in general, three ways for two paradigms to be used:

Sequential Hybrid

The first paradigm passes its output to the second

$$INPUT \rightarrow PARADIGM1 \rightarrow PARADIGM2 \rightarrow OUTPUT$$

An example of this paradigm combination would be a statistical pre-processor that passes its output based on factor analysis to a neural network. This is the weakest form of hybridization, and some would argue that it is not, in fact, a hybrid system at all.

Auxiliary Hybrid

The second paradigm is called by the first paradigm and returns some information

$$\begin{array}{ccccc} INPUT & \rightarrow & PARADIGM1 & \rightarrow & OUTPUT \\ & & PARADIGM2 & & \end{array}$$

An example of this paradigm combination would be a neural network that calls a genetic algorithm module to optimize its structure. As with the Sequential hybrid, two distinct systems can be identified. The level of hybridization is higher, as the second (auxiliary) paradigm is intimately involved with the first. Paradigm 1 can exist without the auxiliary system.

Embedded Hybrid

The first paradigm contains the second paradigm.

$$INPUT \rightarrow PARADIGM1 + PARADIGM2 \rightarrow OUTPUT$$

An example of this paradigm combination would be a neural network-fuzzy logic hybrid where the neural network simulates some characteristics of the fuzzy system. Here the hybridization is absolute. In the extreme case, neither paradigm can be defined without the other. For more than two paradigms, these can be considered building blocks out of which larger system can be built.

4.3 Tabu Search Algorithm

Tabu search [17], [18] is a meta-heuristic which uses memory to guide an iterative search. At each iteration of the search, a neighborhood is examined to construct new solutions. These solutions are compared against the memory structure, i.e., tabu list, to prevent cycling. The best new solution which is not tabu is selected and the system moves to that new solution. This process continues until a predetermined termination criteria is reached, e.g., every move is tabu or a maximum number of iterations has been reached.

Tabu search has been found to be very effective for a variety of combinatorial problems [12], [29], [38]. Tabu search has been compared against simulated annealing and genetic search and found to find better solutions more efficiently for many combinatorial optimization problems [1], [2].

4.3.1 Elements of a tabu search

The tabu search algorithm combines a few simple ideas into a remarkably efficient framework for heuristic optimization. Among the main elements of this framework are:

- A (usually) greedy local search; the next solution is usually the best not-yet visited solution in the current neighborhood.
- A mechanism (the tabu list) discouraging returns to recently visited solutions.
- A mechanism that changes the solution path (perhaps by a random move) when no progress has been made for a long time.

Although a tabu search is conceptually simple, any implementation of an efficient tabu search algorithm is problem specific, and no generic tabu search software is available at this time. Among the issues faced by designers of tabu search algorithms are:

- The nature of the information included in the tabu list.
- The way the tabu list is organized.
- The lengths of the tabu lists.
- The types of moves used to create new solutions.
- The implementation of the local greedy search algorithm.
- Strategies for diversifying the search when no progress has been made for a while.

The overview of a conceptual tabu search algorithm is as follows.

```
Initialize
   Identify initial Solution
   Create empty TabuList
   Set BestSolution = Solution
   Define Termination Conditions
   done = FALSE
Repeat
   if value of Solution〉value of BestSolution then
      BestSolution = Solution
   if no TerminationConditions have been met then begin
      add Solution to TabuList
         if TabuList is full then
            delete oldest entry from TabuList
         find NewSolution by some transformation on Solution
```

if no NewSolution was found **or**
if no improved NewSolution was found for a long time **then**
  generate NewSolution at random
if NewSolution not on TabuList **then**
  Solution = NewSolution
end
else
  done = TRUE
until done = TRUE

5 A hybrid methodology for Vehicle Scheduling Problems

The basic concept behind the hybrid methodology is not to use the TS to directly optimize the parameters of the solution, but rather to use the TS to generate the seeds of a GA heuristic. The approach is shown in the following diagram:

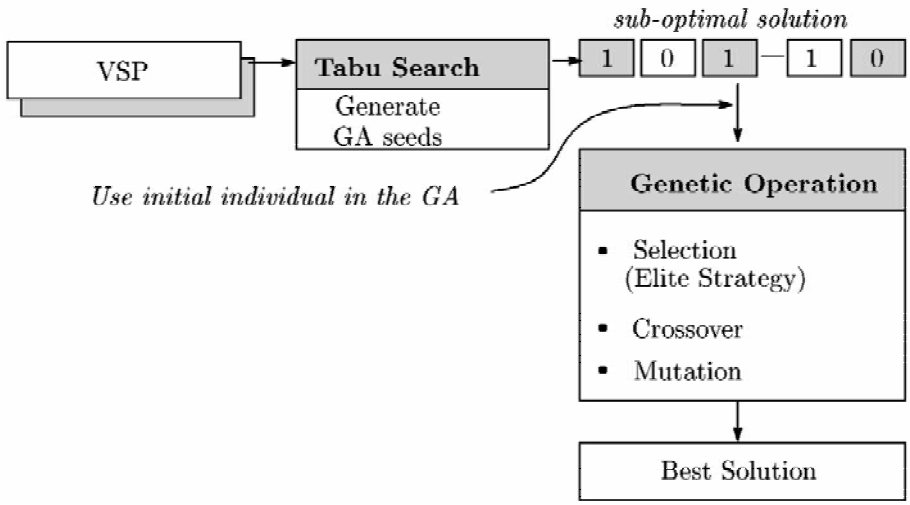
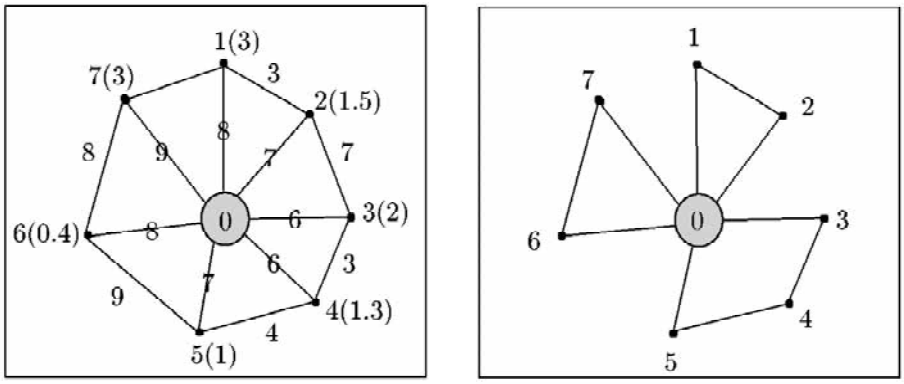


Fig. 2 A Hybrid Meta-Heuristic Approach for VSP

The representation of a feasible solution in a chromosome structure may be much more complex for the VSP than other problems. In addition to the problem of finding an optimal route for each vehicle, there is also the problem of distributing the number of visits required by each customer in the planning horizon, while satisfying all the constraints.

5.1 Chromosome Representation

Like in other GA applications, the members of a population in our GA for VRSP are string entities of an artificial chromosome. The representation of the solution we present here is an integer string of length N, where N is the number of customers including depots in question. Each gene in the string or chromosome, is the integer node number assigned to that customer originally. And the sequence of the genes in the chromosome is the order of visiting the customers.



(a) delivery pattern (b) sub-routes in the solution

Fig. 3 An example of a VSP

If we have the following solution:

Route No. 1 is $0 \rightarrow 1 \rightarrow 2 \rightarrow 0$

Route No. 2 is $0 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 0$

Route No. 3 is $0 \rightarrow 6 \rightarrow 7 \rightarrow 0$,

the chromosome string of Fig. 3 (b) represent the solution as below:

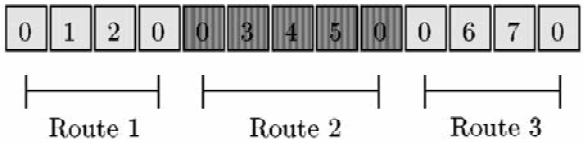


Fig. 4 A chromosome representation

In the Fig 3., a number 0 indicates the delivery center (Depot), and the number written on each like corresponds to the distance between depot and customers and between customers. The numbers 1, 2, 3, 4, 5, 6, 7 correspond to customers. Moreover, the portion of the visited route is called a sub-route, for example route (0, 1, 2, 0) in the Fig. 3 (b) is a sub-route of all the feasible routes. Besides, the numbers in brackets correspond to the quantities required by each customer.

Note that we link the last customer visited in route i with the first customer visited in route $i+1$ to form one string of all the routes involved. Furthermore, we put any bit like 0 in the string to indicate the end of a route. To decode the chromosome into route configurations, we simply insert the gene values into routes sequentially (Fig. 4). In the above-mentioned expression, the length (the length of a sequence) of the chromosome becomes a variable length instead of a fixed length.

5.2 Crossover Operation for VSPs

Conventional single/double point crossover operations are relevant to string entities that are orderless, or of different length. They put two integer/binary strings side by side and make a cut point (or two cut points) on both of them. A crossover is then completed by swapping the portions after the cut point (or between two cut points) in both strings. In the context of a VSP, where each integer gene appears only once in any chromosome, such a simple procedure unavoidably produces invalid offsprings that have duplicated genes in one string. To prevent such invalid offsprings from being reproduced, we propose order based crossover operators as below.

Let P_1 and P_2 be the parent strings, $P_1[1],...,P_1[n]$ and $P_2[1],...,P_2[n]$, respectively. And, let C be the child string. Assuming

a minimization problem, then for all $i = 1,...,n$:

S0 Let $U[\cdot]=[1,...,i,i+1,...,n]$ as set of all customers.

S1 Select any customer i in U.

S2 Pick out subroutes including i from P_1 and P_2, respectively.

$S_1[\cdot]$: subroute of P_1

$S_2[\cdot]$: subroute of P_2

S2-1 If $S_1[\cdot]$ and $S_2[\cdot]\subset U$,

then the one is selected of them with below probabilities.

Probability to select $S_1[\cdot]= f_2/(f_1+f_2)$

Probability to select $S_1[\cdot]= f_1/(f_1+f_2)$

f_1 and f_2 are the number of times that $S_1[\cdot]$, $S_2[\cdot]$ are selected as subroute of C, respectively.

S2-2 If $S_1[\cdot]\subset U$ or $S_2[\cdot]\subset U$,

then the subset inclued in U is used for C as subset.

S2-3 If $S_1[\cdot]\not\subset U$ and $S_2[\cdot]\not\subset U$,

then both $S_1[\cdot]$ and $S_2[\cdot]$ are not selected.

S2-3-1 Select any customer j in U at random.

Let $S[\cdot]$ is subroute of C.

S2-3-2 $S[\cdot]+[j]\rightarrow$ Calculate amount of deliveries

If amount of deliveries of $S > Q$,

then end.

else $S[\cdot]+[j]$ then return to **S2-3-1**.

S2-3-3 If U becomes a null set, then end.

S4 Omit $S[\cdot]$ (decided upper procedure) in U and then, go to **S1**

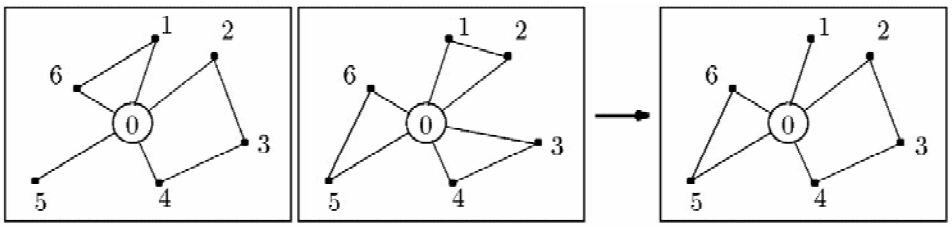
If U becomes a null set, then end.

To generate another offspring, the upper procedure is performed one more time.

Example

Suppose that there are two parent routes P_1 and P_2 as below:

P_1 : 0 1 6 0 0 5 0 0 4 3 2 0 , P_2 : 0 1 2 0 0 3 4 0 0 5 6 0



(a) parent P_1 (b) parent P_2 (c) offspring C

Fig. 5 Example of crossover operator

S0 U = 1, 2, 3, 4, 5, 6

S1 Suppose, 5 is selected in U at random.

S2 S_1 : 0, 5, 0 S_2 : 0, 5, 6, 0

S_1 and $S_2 \subset U$ and probability to select S_1 or S_2 is 0.5, respectively.

↓

S_2 : 0, 5, 6, 0 is selected at random.

S3 Omit 5, 6 from U. → $U' = 1, 2, 3, 4$

S4 Suppose, 3 is selected in U' at random.

S5 S_1' : 0, 2, 3, 4, 0 S_2' : 0, 3, 4, 0

S_1' and $S_2' \subset U$ and probability to select $S_1'[\cdot] = f_2/(f_1 + f_2) = 1/1 = 1$,

$S_2'[\cdot] = f_1/(f_1 + f_2) = 0/1 = 0$, respectively.

↓

Select S_1' : 0, 2, 3, 4, 0

In present, C : 0 5 6 0 0 2 3 4 0

S6 Omit 2, 3, 4 from U'. → $U'' = 1$

⋮

↓

C : **0 5 6 0 0 2 3 4 0 0 1 0** Fig. 5 (c).

5.3 Mutation operations for VSPs

Mutation is applied to each child after crossover. It works by *inverting* each bit in the solution with some probability. In VSP, the mutation-operator indicates a role of *invert path* or *interchange path*. Mutation is generally seen as a background operator which provides a small amount of random search. It also helps to ground against loss of valuable genetic information by reintroducing information lost due to premature convergence and thereby expanding the search space.

The mutation operator adopted in this paper is as follows:

S0 Select any two customers i, j in the $P[\cdot]$ at random.

S1 Try swap i for j

↓

If subroutes including i or $j < Q$

↓

then swap i from j

else not swapping

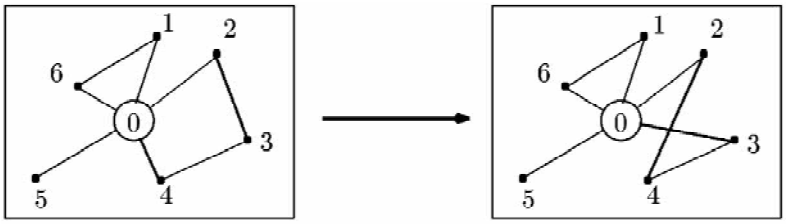
end

Example

P_1 0 1 6 0 0 5 0 0 **4 3** 2 0 → P_1* 0 1 6 0 0 5 0 0 **3 4** 2 0

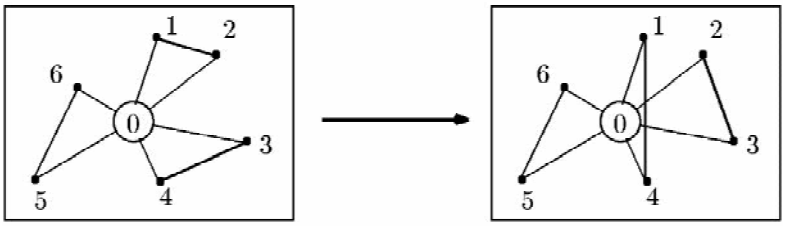
P_2 0 1 **2** 0 0 3 **4** 0 0 5 6 0 → P_2* 0 1 **4** 0 0 3 **2** 0 0 5 6 0

In the string, the characters written in bold indicate selected mutation bits at random.



(a) before mutation (R_1) (b) after mutation (R_1*)

Fig. 6 The sample of mutation (Invert path)



(a) before mutation (R_2) (b) after mutation (R_2*)

Fig. 7 The sample of mutation (Interchange path)

5.4 Fitness, selection methods and generation of initial generation

$$\text{Fitness} = \sum_{i=1}^{n} d_{0i} - \sum_{(i,j)\in A} c_{ij}x_{ij}$$

$$= \text{total distance between depot and customers}$$

$$- \text{result of GA procedure.}$$

Moreover, we applied the *elite strategy* which is often used as a selection strategy.

Creation of initial *C* at random

S0 Let U as a set of all customers.

S1 Select i in U at random.

↓

make subroute $S\{0, i, 0\}$

↓

omit i in U.

S2 select j in U at random.

↓

if $S + j > Q$

then stop, go to **S1**

if $S + j < Q$

then omit i in U, go to **S2**

↓

S3 Repeat **S1** and **S2** until the set U becomes null.

6 Numerical Study

Our numerical experiments were run on a Pentium IV-1.8GHz Processor, Windows XP Operating System using the Program Language C and Borland C++ Builder. We tested three algorithms ; Genetic Algorithm, Tabu Search Algorithm and Hybrid Algorithm, so as to evaluate the performance treating 3-problems. We found that the speed of convergency is very sensitive to the setting of GA-parameters. However, the computational study on a set of benchmark problems indicated that our GA-based heuristic is capable of generating optimal solutions for small-size problems as well as high-quality solutions for large-size problems. The algorithm outperforms any of the previous heuristics in terms of solution quality. The computational times of the algorithm are very reasonable for all problem instances from the heuristic viewpoint. In addition, the numerical experiment used a delivery plan problem which is shown as an example (Table 1).

Table 1 The example of VSP (Input data form)

Node	abscissa (X)	ordinate (Y)	amount of delivery
1	295	272	125
2	301	258	84
3	309	260	60
4	217	274	500
5	218	278	300
6	282	267	175
7	242	249	350
8	230	262	150
9	249	268	1100
10	256	267	4100
11	265	257	225
12	267	242	300
13	259	265	250
14	315	233	500
15	329	252	150

the coordinates of the Depot : (266, 235)
the capacity of truck : 4500

The geometrical coordinates on the plane of each delivery point and a delivery center considered as known, and the distance, used the geometrical distance calculated from coordinates. In the present paper, we use an Experimental Design Method to analyze GA parameters proposed by Han [21], set up as follows:

Table 2 Setting of GA-parameters

GA-parameters	
Crossover Probability	0.6
Mutation Probability	0.2
Selection Method	Elite Strategy
Population Size	101
Generations	2000

Table 3 Results of TS Algorithm

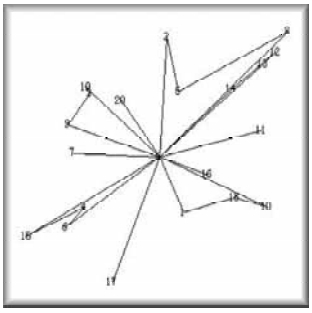
Problems	Number of customers	Distance	time (sec.)
1	20	3430.21	1.2
2	40	8544.30	1.1
3	80	16149.66	1.7

Table 4 Results of Genetic Algorithm

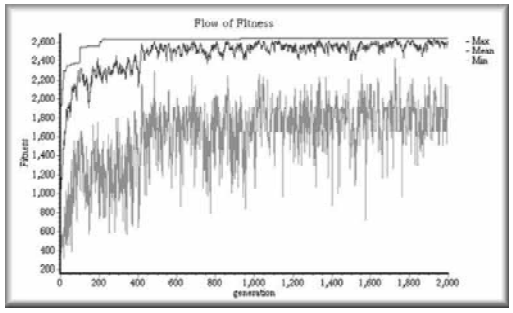
Problems	Number of customers	Distance	time (sec.)	Improvement (%)
1	20	3429.74	27	6.09
2	40	7773.55	31	9.02
3	80	14576.02	44	9.74

Table 5 Results of Hybrid Algorithm

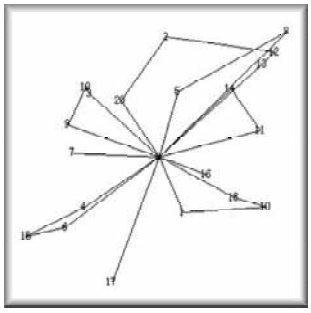
Problems	Number of customers	Distance	time (sec.)	Improvement (%)
1	20	3391.84	28	7.13
2	40	7744.38	68	9.36
3	80	14425.01	66	10.68



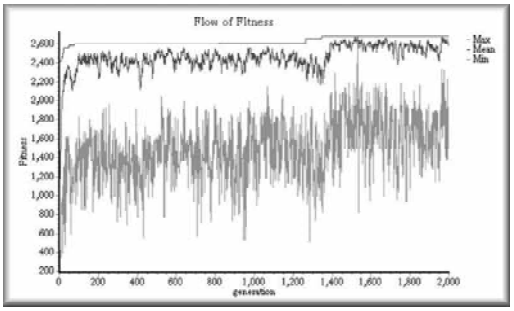
(a) TS (distance : 3430.21)



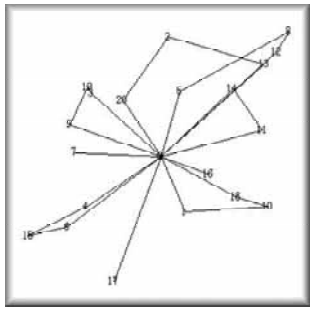
(b) GA (The value of fitness)



(c) GA (distance : 3429.74)

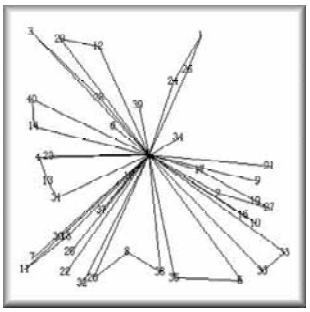


(d) TS + GA (The value of fitness)

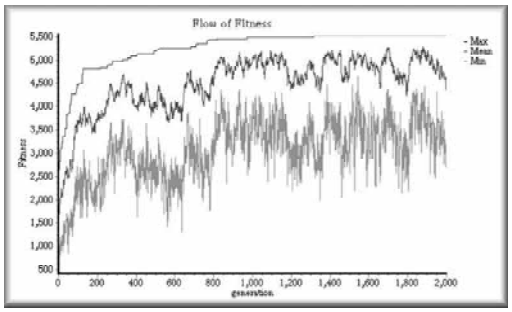


(e) TS + GA (distance : 3391.84)

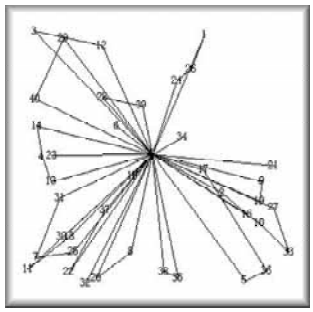
Fig. 8 Result of Problem 1



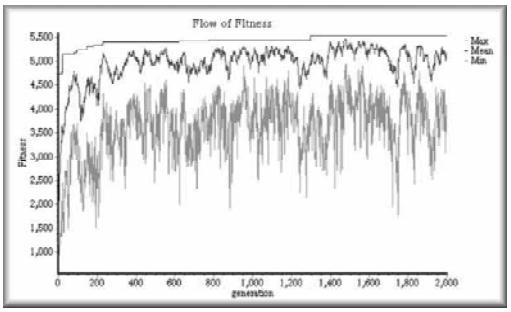
(a) TS (distance : 8544.30)



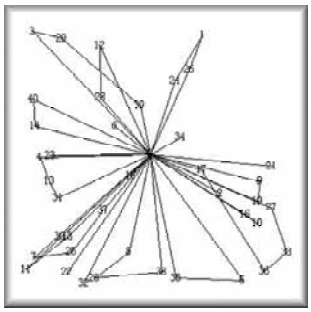
(b) GA (The value of fitness)



(c) GA (distance : 7773.55)

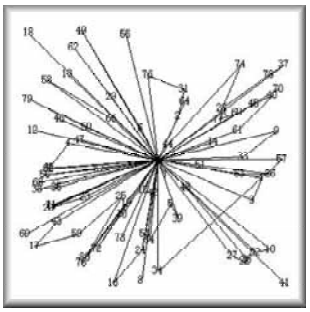


(d) TS ＋ GA (The value of fitness)

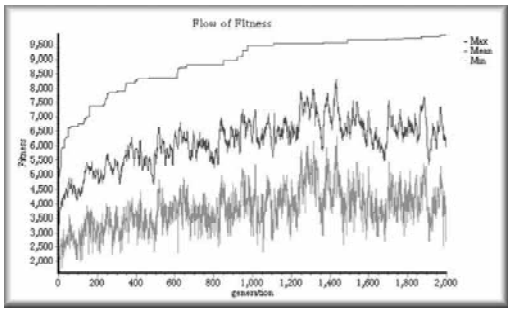


(e) TS ＋ GA (distance : 7744.38)

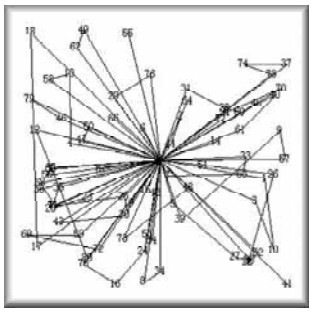
Fig. 9　Result of Problem 2



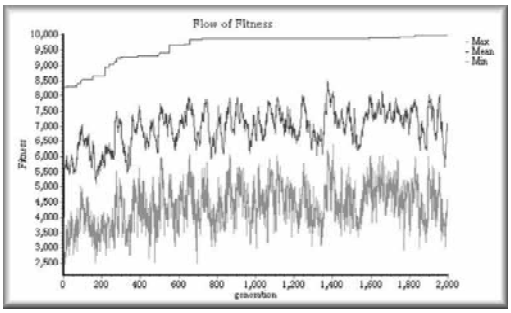
(a) TS (distance : 16149.66)



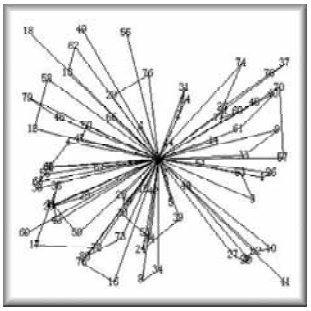
(b) GA (The value of fitness)



(c) GA (distance : 14576.02)



(d) TS ＋ GA (The value of fitness)



(e) TS ＋ GA (distance : 14425.01)

Fig. 10　Result of Problem 3

Since GAs contain a random element theoretically, it cannot guarantee the result of each use of the solution method using GAs to be certainly superior to the result of the TS.

Moreover, in the case of problems 1, 2 and 3, in the above mentioned experiment, a result better than the TS was searched once or twice in the GAs execution. The result of GA is not always better than other methods, because of the randomized property of GA. As a result of comparing TS with GA, and TS with a Hybrid Algorithm (HA), it has been improved 8.283% in GA and 9.056% in HA on average respectively.

7 Conclusions and Remarks

The VSP is one of the classic problems associated with TSP, and it has been applied in various fields. Since, a VSP is difficult to solve in real time, heuristic methods have been adapted to solve it. In this paper, in order to find robust solutions to NP-hard class optimization problems such as a VSP, modified GA and Hybrid-GA are proposed. The TSP is a simplified problem, within the area of VSPs. Moreover, we modified GA-operators (selection, crossover, mutation) to adapt them appropriately to VSP. The influence over solutions, such as crossover and mutation, was investigated and applied by the design of the experiments. Having decided the parameter of GA pertinently, we could discover the rapid convergency of the solution.

Furthermore,we compared HA, GA and TS to identify the validity of discern the Hybrid meta-heuristic algorithm. GA shows the situation in which a better solution is obtained and about 9 percent of the improvement was obtained in the greatest case compared with TS. GA and HA are valid not only to realistic problems but also to the homogeneous VSP. We believe that our method can easily be adapted to solve real-life vehicle scheduling problems.

In further studies, we are trying to investigate the performance for practical examples as well as to refine and improve the algorithm, and consider another type of VSP such as a VSP with a time window constraint, a VSP with multi-depots problems, and others.

References

[1] Aarts. E., van Laarhoven. P., Lenstra. J. and Ulder. N. 1994. A computational study of local search algorithms for job shop scheduling. ORSA Journal on Computing, 6 (2): 118–125.

[2] Battiti. R. and Tecchiolli. G. 1994. Simulated annealing and tabu search in the long run : A comparision on QAP tasks. Computers and Mathematical Apllications.

[3] Back, T., Fogel, D. B., Michalewicz, Z. 1996. *Handbook of Evolutionary Computation*. University of Oxford Press, New York

[4] Christofides, N. 1985. *"Vehicle Routing," The Travelling Salesman Problem*. John Wiley & Sons, Place: pp. 431–448.

[5] Conru A. B. 1994. A genetic approach to the cable harness routing problem, In *Proceedings of the First IEEE Conference on Evolutionary Computing*, Orlando, FL.

[6] Cordeau, J. F., Gendreau, M., Laporte, G. 1995. A Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problems. *Technical Report 95–75*, Center for Research on Transportation, Montreal.

[7] Dantzig G. B. and Ramser J. H. 1959. The Truck Dispatching Problem, *Management Science* 6 (1), pp. 80–91.

[8] David A Coley. 1999. *An Introduction to Genetic Algorithms for Scientists and Engineers*. World Scientific.

[9] Davis L. 1991. *Hanabook of Genetic Algorithm*, Van Nostrand, New York.

[10] Dawsland K. A. 1996. Genetic Algorithm — a tool for OR?. *Journal of the Cperational Research Society* 47: 550–561.

[11] Dejong K. A. 1975. Analysis of the behaviour of a classic of genetic adaptive system, ph. D. thesis, Department of Computer and Communication Sciences, University of Michigan.

[12] Dell'Amico M. and Trubian. M. 1993. Applying tabu search to the job-shop scheduling problem. Annals of Operation Research, 41: 231–252.

[13] Dueck, G. 1990. *The Great Deluge Algorithm and the Record-to-Record Travel*, Scientific Center Technical Report, IBM Germany, Heidelberg Scientific Center.

[14] Etter D. M., Hicks M. J. and Cho K. H. 1982. Recursive adaptive filter design using an adaptive genetic algorithm, In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP82), vol. 2, Paris, France, pp. 635–638.

[15] Gen M., Cheng R. 1997. *Genetic Algorithms and Engineering Design*. Wiley, New York.

[16] Gillet B. E. and Miller L. R. 1974. A heuristic algorithm for the vehicle dispatch problem, *Operations Research* 22: 240–349.

[17] Glover. F. 1989. Tabu search — part I. ORSA Journal of Computing, 3: 190–206.

[18] Glover. F. 1990. Tabu search — part II. ORSA Journal of Computing, 1: 4–32.

[19] Goldberg D. E. 1989. *Genetic Algorithm in Search, Optimization, and Machine Learning*. Addison-Wesley.

[20] Golden, B. L., Chao, I. M., Wasil, E., 1995 An improved Heuristic for the Period Vehicle Routing Problem. *Networks*, pp. 25–44.

[21] Han S. H., 2001. Design of experiments to identify optimal parameters of genetic algorithm *SCJM*

[22] Heinz Mühlenbein., 1997. *Genetic algorithms*. In *Local Search in Combinatorial Optimization*, John Wiley & Sons, Chichester, pp. 137–172.

[23] Holland J. H., 1975. *Adaption in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

[24] I. Or., 1976. Travelling salesman-type combinatorial problems and their relation to the logistics of regional blood banking. Ph. D. Thesis, Northwestern University, Evanston, Illinois

[25] Janson D. J., and Frenzel J. F., 1993. Training product unit neural networks with genetic algorithms, *IEEE Expert*, vol. 8, no. 5, pp. 26–33.

[26] Jarmo T. Alander., 2000. An indexed bibliography of genetic algorithms in operations research. Technical Report series No. 94–1–OR, University of Vaasa, Vassa, Finland.

[27] Kargupta H. and Smith R. E. 1991. System identification with evolving polynomial network, In *Proceedings of the Fourth International Conference on Genetic Algorithms*, SanDiego, CA.

[28] Kitano H. 1995. *Genetic Algorithm 1,2,3*. In Japanese. Tokyo: Sangyou Tosho.

[29] Malek. M., Guruswamy. M., and O. H., 1989. Serial and parallel simulated annealing and tabu search algorithms for the travelling salesman problem. Annals of Operations Research, 21: 59–84.

[30] Michalewicz Z. 1994. *Genetic Algorithm + Data Structure = Evaluation program*. 2nd ed. Springer. Berlin.

[31] Ochi, L. S., Figueiredo, R. M. V., Drummond, L. M. A. 1997. Design and Implementation of a parallel Genetic Algorithm for the Travelling Purchaser Problem, *ACM symposium on Applied Computing*, pp. 257–263.

[32] Park D. and Kandel A. 1994. Genetic-based new fuzzy reasoning models with application to fuzzy control, *IEEE Transaction on Systems, Man, and Cybernetics*, vol. 24, no. 1, pp. 39–47.

[33] Potvin J. Y. and Bengio S. 1996. The vehicle routing problem with time windows part II: genetic search. *Journal on Computing* 8 (2): pp. 165–172

[34] Schaffer, J. D., (ed.)., 1989. Proceedings of the 11th International Joint Conference on Artificial Intelligence, Organ Kaufmann, San Mateo. pp. 750–755.

[35] Sheldon M. Ross. 1997. *Simulation*, 2nd ed. San Diego: Academic Press.

[36] Smith T. and Dejong K. A. 1981. Genetic algorithms applied to the calibration of information driven models of us migration patterns, In *Proceedings of 12th annual Pittsburgh Conference on Modeling and Simulation*, Pittsburgh, PA. pp. 955–959.

[37] Solomon M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints, *Operations Research*, 35 (2), pp. 254–265

[38] Taillard. E., 1991. Taboo search for the quadratic assignment problem. Parallel Computing, 17: 443–455.

[39] Thangiah Sam. 1995. Vehicle routing with timewindows using genetic algorithms. In Application *Handbook of Genetic Algorithms*, New Frontiers, vol II, pp. 253–277, CRC Press, Boca Raton, 1995.

[40] Uckun S., Bagchi and Kawamura K., 1993. Managing genetic search in job shop scheduling, *IEEE Expert*, vol. 8, no. 5, pp. 15–24.

[41] Wren A., 1971. *Computers in Transport Planning and Operation*, Ian Allan, London.

[42] Wren A. and Holliday A., 1972. Computer scheduling of vehicles from one or more depots to a number of delivery points. *Operational Research Quarterly*. 23: 333–344.