International Journal of Hybrid Information Technology
Vol.9, No.1 (2016), pp. 397-404
http://dx.doi.org/10.14257/ijhit.2016.9.1.34

A hybrid Genetic-Monkey Algorithm for the Vehicle Routing Problem

Jiashan ZHANG[1,2] and Jun YI[1]

[1]*Chongqing Vocational Institute of Engineering, JiangJin, Chongqing, 402260, China*

[2]*College of Business Administration，Liaoning Technical University, Huludao, Liaoning, 125000, China*

zh_jiashan@163.com

Abstract

The vehicle routing problem (VRP) is one of the most challenging problems in the optimization of distribution. Genetic algorithm (GA) has been proved capable of solving VRPs. However, the resolution effectiveness of GA decreases with the increase of nodes within VRPs. It often appears premature convergence or poor ability of local search. In this paper, a hybrid genetic-monkey algorithm (HGMA), integrating monkey algorithm (MA) into genetic algorithm framework, is proposed to overcome the shortcomings above. Monkey climbing (A novel climb process is designed for VRP with discrete variables.) and somersault processes improve ability of local and global search. Furthermore, flexible fitness function, through which infeasible solutions are allowed, are developed to expand the search space. The experiment results indicate efficiency of the proposed algorithm.

Keywords**:** *VRP genetic algorithm, premature convergence, monkey algorithm, fitness function*

1. Introduction

The Vehicle Routing Problem (VRP) was first introduced by Dantzig and Ramser in 1959[1] and it has been widely studied since. In the VRP, customers with known demands and service time are visited by a fleet of vehicles with limited capacity and initially located at a distribution center. The objective is to minimize total cost (*e.g.*, traveled distance), such that each customer is serviced exactly once (by a single vehicle), total load on any vehicle associated with a given route does not exceed vehicle capacity, and route duration combining travel and service time, is bounded to a preset limit. Here, the term customer will be used to denote the stops to pick up and /or deliver. As NP-hard problems [3], the VRPs are difficult to solve and actually no optimal algorithm has been found so far, which is able to solve the problem in polynomial time [2]. Finding optimal solution to NP-hard problem is usually time-consuming or even impossible.

Exact algorithms [3], such as branch and bound algorithm, cutting planes approach and dynamic programming, are used to solve the small-scale VRPs. Approximate algorithms are used for large-scale instances. For example, tabu search [4] and simulated annealing [5] are applied to VRP. Ai and Kachitvichyanukul [6] proposed a particle swarm optimization for VRP with random key based representation. Gajpal and Abad [7] applied ant colony optimization to VRPSPD. Zachariadis, and Tarantilis [8] proposed an adaptive memory methodology for VRP. Genetic algorithm (GA) [9] has drawn attention from researchers due to its robustness and flexibility. GA has been used to tackle many combinatorial problems, including certain types of vehicle routing problem. For example, Shieh and May [10] apply GA to optimizing VRP. However, the general Genetic Algorithm has

ISSN: 1738-9968 IJHIT

the weakness of premature and poor ability of local search. Numerous successful applications strongly favor hybrid approach [11]. In the hybrid approach, the GA globally explores the population, while other heuristics mainly focus on local exploitation of chromosomes. As a novel heuristic, monkey algorithm [12] is proposed to solve a variety of difficult optimization problems with non-linearity, non-differentiability, and high dimensionality. It is famous for convergence rate. In this paper, a competitive hybrid genetic-monkey algorithm (HGMA), in which monkey climbing and somersault processes are introduced, is proposed.

2. Formulation for VRP

Given a distribution center has *m* vehicles and deliveries goods for n customers (*i.e.*, the nodes are $v_1, v_2, ..., v_n$), The location of distribution center and customers are known. Each customer has a known demand level: q_i, i=$1, 2, ..., n$. Delivery routes for vehicles are required to start and finish at the distribution center, so that all customer demands are satisfied and each customer is visited by just one vehicle .The objective is to find a set of routes that minimizes the cost (*e.g.*, the total distance traveled).

Variables used in this paper:

Q: the vehicle capacity; D: the maximum travel distance of vehicle;

c_{ij} : the distance between customer i and j; v_0 : the distribution center.

First, we define decision variables: x_{ijk} denotes whether vehicle k moves from node (or distribution center) i to j ; y_{jk} denotes whether the customer j is distributed by vehicle k.

$$x_{ijk} = \begin{cases} 1, & if \text{ vehicle k visits customer j after i} \\ 0, & else \end{cases} \qquad y_{jk} = \begin{cases} 1, & if \text{ vehicle k visits client j} \\ 0, & else \end{cases}$$

The objective function is:

$$\min z = \sum_i \sum_j \sum_k c_{ij} x_{ijk} \tag{1}$$

S.t.

$$\sum_{i=1}^{n} q_i y_{ik} \le Q, \quad k = 1, 2, ..., m ; \tag{2}$$

$$\sum_i \sum_j d_{ij} \cdot x_{ijk} \le D, \quad k = 1, 2, ..., m \tag{3}$$

$$\sum_{k=1}^{m} y_{ik} = \begin{cases} 1, & i = 1, 2, ..., n \\ m, & i = 0 \end{cases} \tag{4}$$

$$\sum_{i=0}^{n} x_{ijk} = y_{jk}, \quad j = 1, 2, ..., n; \quad k = 1, 2, ..., m \tag{5}$$

$$\sum_{j=0}^{n} x_{ijk} = y_{ik}, \quad i = 1, 2, ..., n; \quad k = 1, 2, ..., m \tag{6}$$

Equation (1) is the objective function. Eq.(2) is the capacity constraint for each vehicle. The sum over the demands of the customers within each vehicle v has to be less than or equal to the capacity of the vehicle. Eq.(3) is constraint for single maximum traveling distance, Eq.(4) makes sure that each customer is assigned to exactly one vehicle. Eq.(5)and (6) make sure that that only one vehicle arrives at each node. However, there is a lower bound on the number of vehicles, which is the smallest number of vehicles that can carry the total demand of the customers.

3. The Proposed HGMA for VRP

3.1 Framework

The proposed algorithm starts by computing an initial population, i.e. the first generation. We assume that the initial population contains *n* individuals (popsize), where *n* is an integer. A new evaluation function for the fitness value was introduced. After computing the fitness values of individuals, we select a pair of individuals (parents), and apply the crossover operator to produce two new individuals (children). Subsequently, we adopt the swapping mutation to the genotypes of the newly produced children. Then，monkey climbing and somersault processes are carried out to improve the solutions. The algorithm stops if a prespecified number of generations, which is denoted as maxgen, are reached.

The proposed algorithm framework is specified as follows:

Initialization
Generate *n* initial solutions (Pop) randomly
Repeat
    p=1
    For j =l: *n* do
        Select two parents from Pop
        Generate a new solution S_j using crossover and mutation operators;
Improve the solution S_j using monkey climbing and somersault, denoted by S_j';
Add S_j' to Pop
    End for
    p=p+ 1
Until (convergence criteria or max number of generations)

3.2 Coding and Fitness Function

A good representation (coding) of VRP solution must identify the number of vehicles [14], which customers are assigned to each vehicle and in which order they are visited. Like in most GAs for VRP, a chromosome *I(n)* simply is a sequence (permutation) *S* of n customer nodes. For instance, there are 8 customer nodes, a randomly generated chromosome is 3 1 6 8 5 4 2 7, which can be interpreted as *r*=3 feasible routes: 0-3-1-6-0, 0-8-5-4-0, and 0-2-7-0. Here, according to the capacity of vehicle and the maximum travel distance, greedy principle is applied. If no vehicle is overloaded and *r*<=m (the number of vehicles), then this chromosome is legal; otherwise, it is illegal.

Every solution has a fitness value assigned to it, which measures its quality. It is rather straight forward to select a suitable fittness value for VRP, where the quality of solution is based on the total cost of travelling for all vehicles.

Usually, no infeasible solutions were allowed even though the operators were able to generate such solutions. But it can often be profitable to allow infeasible solutions during the computation process. Expanding the search space over the infeasible region does often enable the search for the optimal solution, particularly when dealing with non-convex feasible search spaces, as Figure 1 shows. The infeasible solution is much closer to global optimal solution than the feasible solution. It contains more valuable information.

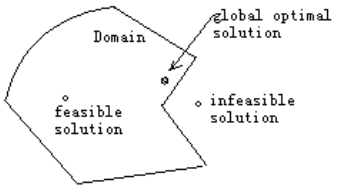

Figure 1. Feasible and Infeasible Solution

The fitness value is made capable of handling infeasible solutions by adding a penalty term depending on how much the constraint are violated. The penalty is supposed to be insignificant at the early iterations, allowing infeasible solutions, and to force the final solution to be feasible in the end. In this paper, a new evaluation function for the fitness value is then developed. It is illustrated as follows.

$$f_x = \frac{M}{D + \alpha \cdot \frac{it}{IT} \sum_r (\max(0,\ totload_r - cap) \times L + 10 \times L \times \max(0, r - m))} \quad (7)$$

where:

M: a large enough positive number; *L: the maximum travel distance* of the vehicles *;*

cap: the capacity of vehicles, $totload_r$: the total load of vehicle r;

it : the current iteration, *IT*: the total number of iterations,

α: constant, *D* : the total distance vehicles traveled of the corresponding chromosome;

The size of α determines the effect of penalty, i.e. a large α increases the influence of penalty term on the performance and a small one decreases the effect. Contrast with the capacity constraint, the number of vehicles is more prohibited to violate, i.e. the penalty is more. Apparently from the above formula, the penalty value for feasible solutions is zero.

3.3 Crossover

Crossover is a probabilistic process that exchanges information between parents for generating offsprings. It leads to an effective combination of partial solutions in other chromosomes and speeds up the search procedure. To preventing premature convergence of the GA to a local optimum, the crossover probability p_c varies adaptively. When the iteration of population is less than $0.1*N_{cmax}$, the crossover probability is the constant, or the crossover probability is as follows.

$$p_c = \begin{cases} k_1 & , \quad f_c \le f_{avg} \\ \dfrac{k_1 (f_{\max} - f_c)}{f_{\max} - f_{avg}}, & f_c > f_{avg} \end{cases} \quad (8)$$

where k_1($0< k_1<1$) is a constant, $f_{\max}$ is the maximum fitness value of the population; f_{avg} is the average fitness value of the population and f_c is the larger fitness value of the solutions to be crossed.

Let the two parent solutions be P1=(1 2 3 4 5 6 7 8 9), P2 =(4 5 2 1 8 7 6 9 3). There are many various crossover operators developed for permutation chromosomes [15]. In this paper, procedures of crossover are shown as follows.

(1) Select two positions along the string uniformly random, for example P1=3, P2=7, see Figure2 (a).

(2) Randomly scramble the order of the genes in two substrings. It is repeated for 10 times, see Figure2 (b).

(3) Exchange two substrings between parents to produce children, see Figure2 (c) and (d).

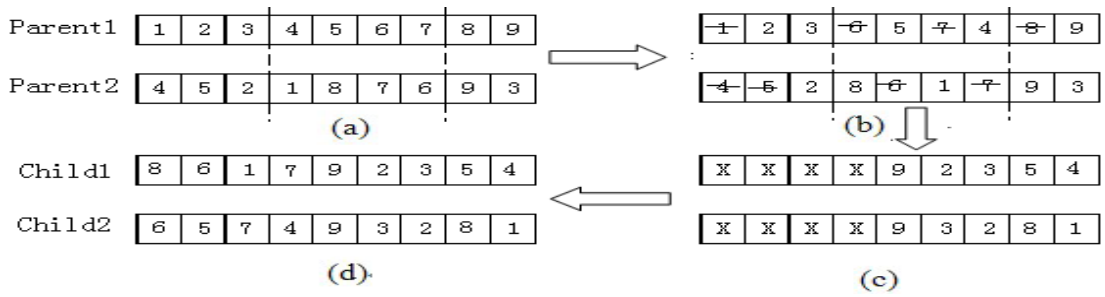

Figure 2. Crossover Operators

After crossover, each offspring and parent (*i.e.,* Parent1,Parent2,Child1, Child 2) is evaluated in terms of fitness value mentioned before. Two solutions with the better fitness value (offspring or parent) are preserved, labeled C1,C2.

3.4 Mutation

Similarly, the mutation probability pm varies adaptively. When the iteration of population is less than 0.1**Ncmax*, the mutation probability is the constant, or the mutation probability is as follows.

$$p_m = \begin{cases} k_2 & , \quad f_m \le f_{avg} \\ \dfrac{k_2(f_{\max} - f_m)}{f_{\max} - f_{avg}}, & f_m > f_{avg} \end{cases} \tag{9}$$

where k2(0< k2<1) is a constant, $f_{\max}$ is the maximum fitness value of the population; f_{avg} is the average fitness value of the population and f_m is the larger fitness value of the solutions to be crossed.

Inversion [16] is a widely used mutation operator. It selects two random genes along the chromosome and reverses the segment between these two genes. It is particularly well-suited for all the problems that naturally admit a permutation representation in which adjacency among elements plays an important role [21]. In this paper, inversion mutation is adopted. Procedures are shown as follows.

Random mutation Points: P1=2, P2=6.

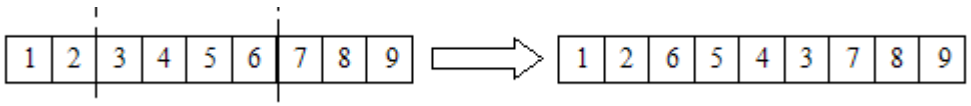

Figure 3. Inversion Mutation Operators

The reverse operation is repeated n/10 times.

3.5 Climbing and Somersault Search

Monkey algorithm (MA) is suitable for solving optimization problems of continuous variables. However, when the problem contains discrete variables, the pseudo-gradient can't give the descent direction of the objective function and it lead to invalid climb process. So, the algorithm can't find the optimum solution. Here , a novel climb process is designed for VRP with discrete variables.

(1) Climbing process

1) Randomly generate a vector $\Delta x_i = (\Delta x_{i1}, \Delta x_{i2} ..... \Delta x_{in})$, $\Delta x_{ij} \in$[-a,a] , the parameter a (a > 0), called the step length of climb process.

2)Then, calculate $f(x_i + \Delta x_i)$ and $f(x_i - \Delta x_i)$, if $f(x_i + \Delta x_i) < f(x_i - \Delta x_i)$, and $f(x_i + \Delta x_i) < f(x_i)$, set $x_i = x_i + \Delta x_i$; if $f(x_i - \Delta x_i) < f(x_i + \Delta x_i)$,and

$f(x_i - \Delta x_i) < f(x)$, set $x_i = x_i - \Delta x_i$.

3) Repeat steps 1) to 2) until there is little change on the values of objective function in the neighborhood iterations .

(2) Somersault process

Somersault process aims to find out new searching domains. the barycentre of all current positions (solution) is selected as a pivot, then the monkeys will somersault along the direction pointing to the pivot.

1) Given the somersault interval [c, d] (determined by specific situations); randomly generate a real number $\beta \in [c, d]$;

2) Calculate $p_i = \frac{1}{m}\sum_{j=1}^{m} x_{ij}$, p is called the somersault pivot. Set

$$x_i^{''} = x_i + round(\beta \mid p - x_i \mid), \quad \text{i=1,2,…..m}$$

3) Set $x_i = x_i^{''}$, repeat steps 1) and 2) until a feasible solution is found.

The flow chart for the proposed algorithm is denoted in Figure 4.

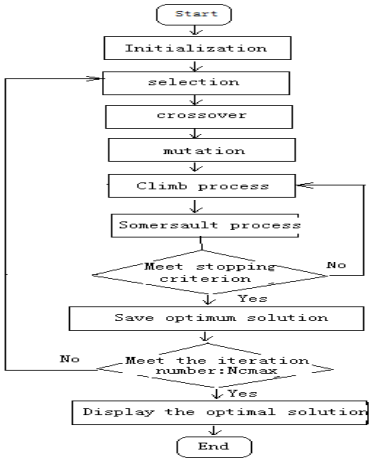

Figure 4. Flow Chart for HGA

4. Experimental Analysis

Here，the proposed HGA was applied to solve benchmark test problem (1987) with the aim of evaluating its performance against other methods. Data of the problems, bounds, optimal solution, *etc.*, are taken from the website:

http://www.branchandcut.org/VRP/data.

The best result was obtained with pop_size= 30, the out iteratation: maxgen = 200, and the inner iteratation (climbing and somersault process), threshold value: e=0.001 ($i.e., \mid f(x_i) - f(x_{i-1}) \mid < 0.001$).

In the mutation, the reverse operator is repeated n/10 times for qualified chromosome, where n is the number of nodes. Penalty weight for illegal route is denoted in (7). For each instance, the algorithm has been run 10 times. The results of simulations are presented in Table 1. Here, instances are solved by three methods: the proposed algorithm (HGA), general Genetic Algorithm (GA) and Ant Colony System (ACS). The experiment results (see Table 1) indicate that the proposed algorithm has high reproducibility for solving VRPs.

In most instances, the proposed algorithm precedes general Genetic Algorithm and ACS. However, in the instances of A-n63-k9, A-n63-k10 and E-n30-k3, the proposed algorithm is overtaken by ACS mainly because of improper parameters setting, such as the probability of crossover and mutation, the step length of climb process, the somersault

interval and others.

Table 1. Comparisons between Different Methods

Problems instance	Genetic algorithm	AntColony System(ACS)	The propsed algorithm	Problems instance	Genetic algorithm	AntColony System	The propsed algorithm
A-n33-k5	697	680	675	E-n30-k3	545.35	524. 97	538.95
A-n37-k5	724	709	694	E-n33-k4	885.61	871. 15	858.72
A-n37-k6	1012	980	985	E-n51-k5	584.64	578.49	538.41
A-n54-k7	1309	1227	1197	E-n76-k7	731. 48	723. 29	701.28
A-n63-k9	1746	1670	1714	E-n76-k10	900.26	888.04	853.05
A-n63-k10	1452	1387	1410	E-n76-k14	1279.35	1108.59	1069.95

Figure 5(a), (b), (c) and (d) show interation comparison of three methods: the propsed algorithm(HGA), general Genetic Algorithm and ACS.

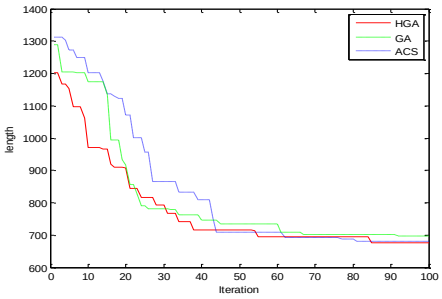

Figure 5(a). Interation Comparison of Different Methods (A-n33-k5)

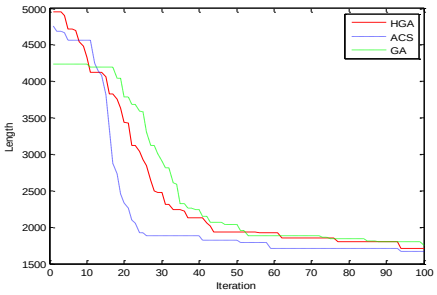

Figure 5(b). Interation Comparison of Different Methods (A-n63-k9)

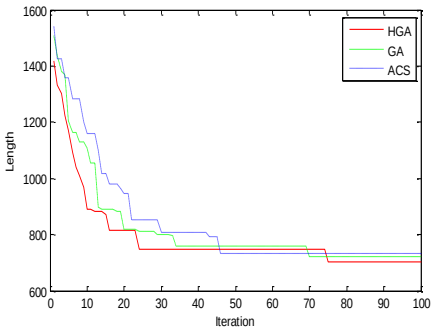

Figure 5(c). Interation Comparison of Different Methods (E-n76-k7)

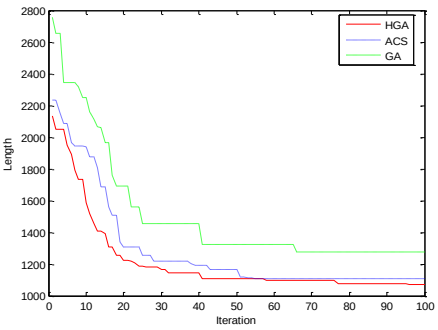

Figure 5(d). Interation Comparison of Different Methods (E-n76-k14)

In the process of calculation，infeasible solutions were allowed to develop the optimum. For the instance of A-n33-k5, in the solution: V_1:1-3-21-33-14-9-8-27-5-1; V_2:1-6-28-26-31-11-13-1; V_3:1-25-7-20-15-22-12-1; V_4:1-30-4-10-18-17-16-23-1; V_5:1-2-32-19-29-24-1, two trucks are overloaded respectively by 1 and 11 percent, i.e. V_1:1-3-21-33-14-9-8-27-5-1,loading rate :101%; V_4: 1-30-4-10-18-17-16-23-1, loading rate :111%. So, it is an infeasible solution. However, the infeasible solutions include more valuable information. They are remained more in the evolution to guarantee the validity of iteration and result in the optimum.

5. Conclusion and Future Work

Evolutionary algorithms have been widely applied to resolve complex optimization problems. However, single evolutionary algorithm is difficult to find satisfactory solution. The resolution effectiveness decreases with the increase of nodes. Hybrid algorithm are popular, which outperforms single evolutionary algorithm in many cases. In this paper, we have combined genetic algorithm (GA) with monkey algorithm(MA), and designed the computer program .The proposed hybrid algorithm(HGMA) is very successful when applied to VRPs and problems that can be naturally coded as ordered lists and there is no standard GA for manipulating . Monkey climbing and somersault process play an important role in improving quality of solution. The proposed algorithm is also robust and effective.

Acknowledgements

This paper is supported by the National Natural Science Foundation of China (No. 50904032) and the Education Department of Liaoning province science and technology research project（No.L2010177）.

References

[1] G. Dantizig and J. Ramser, "The Truck Dispatching Problem", Management Science, **(1959)**, pp. 81-89.
[2] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing", Journal of Heuristics, **(1996)**, pp. 25-30.
[3] G. Laporte, "The Vehicle Routing Problem: an Overview of Exact and Approximate Algorithms. European Journal of Operational Research, vol. 59, **(1992)**, pp. 345-348.
[4] F. Gloyer, "Future Paths for Integer Programming and Links to Artificial Intelligence", Computers and Operations Research, vol. 13, **(1986)**, pp. 533-549.
[5] N. Metropolis, A. Rosenbluth, *et al.*, "Equation of state calculations by fast computing machines", Journal of Chemical Physics, vol. 21, **(1953)**, pp. 1087-1092.
[6] T. J. Ai and V. Kachitvichyanukul, "A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery", Computers &Operations Research, no. 36, **(2009)**, pp. 1693–1702.
[7] Y. Gajpal and P. Abad, "An ant colony system (ACS) for vehicle routing problem with simultaneous delivery and pickup", Computers & Operations Research, no. 36, **(2009)**, pp. 3215–3223.
[8] E. E. Zachariadis, C. D. Tarantilis and C. T. Kiranoudis, "An adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries", European Journal of Operational Research, **(2010)**, pp 401–411.
[9] J. H. Holland, "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor, **(1975)**.
[10] H. M. Shieh and M. D. May, "Solving the capacitated clustering problem with genetic algorithms", Journal of the Chinese Institute of Industrial Engineers, no. 18, **(2001)**, pp. 1-12.
[11] J. S. Chen, J. C. H. Pan and C. M. Lin, "A hybrid genetic algorithm for the re-entrant flow-shop scheduling problem", Expert systems with Applications, no. 34, **(2008)**, pp. 570-577.
[12] R. Q. Zhao and W. S. Tang, "Monkey algorithm for global numerical optimization", Journal of Uncertain Systoms, vol. 2, no. 3, **(2008)**, pp. 165-176.
[13] S. M. Sait and H. Youssef, "Iterative Computer Algorithms with Application in Engineering: Solving Combinatorial Optimization Problems, chapter 3", IEEE Computer Society, **(1999)**.
[14] B. M. Baker and M. A. Ayechew, "A genetic algorithm for the vehicle routing problem", Computers and Operations Research, vol. 30, no. 5, **(2003)**, pp. 787-800.
[15] I. Oliver, D. Smith and J. Holland, "A study of permutation crossover operators on the traveling salesman problem", Proceedings of the Second International Conference on Genetic Algorithms, **(1987)**, pp. 224–230.
[16] E. Abido, *et al.*, "Precedence-Preserving GAs Operators for Scheduling Problems with Activities' Start Times Encoding", Journal of Computing in Civil Engineering, vol. 24, no. 4, **(2001)**, pp. 345-356.

