Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 181 (2006) 1049–1060
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

A memetic algorithm for a vehicle routing problem with backhauls

R. Tavakkoli-Moghaddam [a,*], A.R. Saremi [a], M.S. Ziaee [b]

[a] Department of Industrial Engineering, Faculty of Engineering, University of Tehran, P.O. Box 11365/4563, Tehran, Iran
[b] Faculty of Management, University of Tehran, Tehran, Iran

Abstract

This paper considers an extension of a vehicle routing problem with backhauls (VRPB). In this problem, a set of costumers are divided in two subsets consisting of linehaul and backhaul costumers. Each linehaul costumer requires delivering its demands from the depot. In addition, a specified quantity of products should be picked up from the backhaul nodes to the depot. The VRPB is a well-known NP-hard problem in strong sense and a number of algorithms are proposed for approximate solutions of such a hard problem. In this paper, a memetic algorithm (MA) is proposed which uses different local search algorithms to solve the VRPB. Exploiting power of memetic algorithm, inter and intra-route node exchanges are used as a part of this evolutionary algorithm. Extensive computational tests on some instances taken from the literature reveal the effectiveness of the proposed algorithm.

Keywords: Vehicle Routing Problem with Backhauls; Memetic algorithm; Local search

1. Introduction

This paper considers an extension of a vehicle routing problem with backhauls (VRPB). In this problem, a set of costumers are divided in two subsets consisting of linehaul and backhaul costumers. Each linehaul costumer requires delivering its demands from the depot. In addition, a specified quantity of products should be picked up from the backhaul nodes to the depot. A good example of this costumer set can be the grocery industry. An instance of this costumer partitioning is represented by the grocery industries. In this case, supermarkets and shops are the linehaul nodes and grocery suppliers are the backhaul nodes. In recent years, it is discovered that a great amount of savings have been achieved with combining the pickup/delivery context and exactly by visiting backhaul costumer along the distribution route. For example, the Interstate Commerce Commission estimated to save $160 million each year in USA grocery industries due to the introduction of backhauling [1].

* Corresponding author.
E-mail address: tavakoli@ut.ac.ir (R. Tavakkoli-Moghaddam).

0096-3003/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2006.01.059

More precisely, the VRPB can be defined as a problem of determining a set of vehicle routes visiting all costumers subject to the constraints as follows: (i) each vehicle performs just one route; (ii) for each route, the total load assigns to linehaul and backhaul node, in which should not separately exceed the vehicle capacity; (iii) one each route, backhaul nodes should be visited after all linehaul routes; and (iv) the total transportation cost should be minimized. The third condition (i.e., precedence constraint) generally puts here by the fact that in many applications, linehaul costumers have higher service priority than backhauls.

The model and algorithm presented here consider the heterogeneous fleet and associated networks can be symmetric or asymmetric. In the heterogeneous fleet, different type of vehicle with dissimilar capacities may be introduced. More over in symmetric networks, the distance between two nodes is same in two directions, whereas in asymmetric network this assumption does not hold. The VRPB is NP-hard in strong sense, since it generalizes the capacitated VRP arising when there is no backhaul node available.

2. Relevant literature

All previous researches from the literature have been considered only the homogeneous fleet version of the VRP. Toth and Vigo [2] developed a branch-and-bound algorithm in which a lower bound on the optimal solution is derived from a Lagrangean relaxation of some constraints of their linear programming (LP) formulation. Iteratively, the Lagrangean relaxation bound is further strengthened by adding valid inequalities in a cutting plane fashion. Yano et al. [3] developed a set covering based on the exact algorithm for a practical application in quality stores industry of the VRPB. In this application, each vehicle cannot deliver to (or collect from) more than a few customers. Mingozzi et al. [4] presented another LP formulation of the problem. They generated bounds by combining different heuristic procedures. An upper bound is obtained from the dual of their LP-relaxation and an exact branch-and-bound algorithm is developed based on the proposed lower and upper bound procedures.

Deif and Bodin [5] proposed a heuristic algorithm, which is called DB hereafter, is based on the Clarke–Wright VRP heuristic [6]. The Clarke–Wright algorithm starts with an infeasible solution where each node visited by a separate route. Routes are then iteratively combined by regarding the ''saving'' in terms of routing cost that can be achieved by serving two nodes in the same route rather than putting each one in the route. Deif and Bodin [5] experimentally showed that VRPB best results are obtained by delaying the insertion of backhaul nodes. They put a penalty for arcs connecting linehaul nodes to backhaul nodes. The Clarke–Wright algorithm and hence the DB algorithm do not control the number of vehicles in the final solution and certainly this solution may need more than K routes and may be infeasible. Deif and Bodin tested their algorithm on randomly generated instances with 100–300 vertices with a backhaul percentage between 10% and 50%. They tested several problems and found out the best result obtained by problems with backhaul percentage between 20% and 50%. Different extensions of the DB algorithm are proposed by Casco et al. [7]. Golden et al. [1] also dealt with the VRPB without precedence constraint and they proposed a Clarke–Wright based algorithm.

Goetschalckx and Jacobs [8] proposed an algorithm, which is called SF hereafter, for VRPB with the Euclidean cost matrix. The approach is based on space filling curves. In their work linehaul and backhaul nodes are separately transformed from points in the plan to point along a line. Two sequences of nodes are then partitioned to create a feasible solution of only one type. Then, the nearest backhaul route is merged in the specified linehaul route. In addition, this approach algorithm does not guarantee the number of vehicles that do not exceed K routes. Goeschalckx and Jacobs [8] tested DB and SF algorithms on 57 Euclidean instances with 25–150 vertices, 20–50% of which are backhauls. The result presented in their paper shows that DB solutions are generally better than SF but the SF algorithm is faster for large problems.

Later Goestschalckz and Jachobs [9] presented a heuristics, which is called LHBH, for the Euclidean version of the VRP. Their algorithm was based on the extension of Fisher and Jaikumar for the general VRP, in which initially a partial solution made up of K route primitives is generated by determining k seed radials from the solution of the capacitated location–allocation problem. Then, for each such radial a route primitive is determined by selecting nodes located near the radial. Then, the sequencing procedure arranges the linehaul node according to their distance to the depot increases and the backhaul node due to their distance to the depot decreases. Then, the sequenced nodes are clustered in K routes according to vehicles' capacities by solving the generalized assignment problem for them. Ultimately, the routes can be determined due to a modified

insertion heuristic for the TSP and a post-optimization exchange procedure. Comparison of experiments on Euclidean problem instances showed that the LHBH outperforms the SF algorithm.

Toth and Vigo [10] developed a cluster-first route-second heuristic based on the K-tree approach for the VRP. Their heuristic was named TV starting possibly with infeasible initial solutions which are obtained by solving a Lagrangean relaxation of the VRPB developed in [2]. Then, it gradually tries to construct a final set of feasible routes by using some local shifts and exchanges of customers between vehicle routes with the well-known 2-Opt and 3-Opt local improvement procedures. Furthermore, Toth and Vigo [11] implemented an improved version of the above heuristic (i.e., TV) to the symmetric and asymmetric VRPB. The TV heuristic algorithm produces very competitive results in the literature for this problem.

On the heuristic side, few metaheuristic techniques have been proposed. Potvin et al. [12] designed a genetic algorithm to identify an ordering of customers that produces good routes. Tavakkoli-Moghaddam et al. [13] proposed en efficient metaheuristic method based on simulated annealing (SA) to solve a new mixed-integer linear model of a capacitated vehicle routing problem (CVRP) with split services and heterogeneous fleet. The proposed model was to minimize the fleet cost and total distance traveled in order to serve all given customers by using the maximum capacity and minimum number of vehicles. Duhamel et al. [14] designated a tabu search method for the VRP with backhaul and time windows (VRPBTW) as well as customer precedence. First, a feasible solution is constructed by an adapted version of Solomon insertion heuristic. Then, tabu search and local search and improvement algorithms are used to improve the solution.

3. Mathematical model

In contrast to the previous researches, in this paper a heterogeneous VRPB model is presented. The proposed model considers different types of vehicles in the fleet in terms of the capacity and transportation cost. A mixed-integer programming (MIP) formulation of the problem is presented below.

3.1. Notation

C_{ijk}	cost of moving from node i to node j using vehicle k
n	number of linehaul nodes
m	number of backhaul nodes
M	number of vehicles
Q_k	capacity of vehicle k
f_i	backhaul demand of node i
d_i	linehaul demand of node i

3.2. Decision variable

X_{ijk} = a binary variable which is equal to 1 if there is an arc from node i to node j using vehicle k.

3.3. Formulation

$$\min \quad z = \sum_{i=0,\ldots,n+m} \sum_{j=0,\ldots,n+m} \sum_{k=1,\ldots,M} C_{ijk} x_{ijk} \tag{1}$$

s.t.

$$\sum_{j=0,\ldots,n+m} \sum_{k=1,\ldots,M} x_{ijk} = 1, \quad i = 1,\ldots,n+m \;\; i \neq j, \tag{2}$$

$$\sum_{i=0,\ldots,n+m} \sum_{k=1,\ldots,M} x_{ijk} = 1, \quad j = 1,\ldots,n+m, \;\; j \neq i, \tag{3}$$

$$\sum_{j=0,\ldots,n+m} \sum_{k=1,\ldots,M} x_{0jk} = M, \tag{4}$$

$$\sum_{i=0,\ldots,n+m}\ \sum_{k=1,\ldots,M} x_{i0k} = M, \tag{5}$$

$$\sum_{i=1,\ldots,n}\ \sum_{j=1,\ldots,n+m} d_i x_{ijk} \leqslant Q_k, \quad k = 1,\ldots,M;\ \ j = 0,1,\ldots,n+m, \tag{6}$$

$$\sum_{i=n+1,\ldots,n+m}\ \sum_{j=1,\ldots,n+m} f_i x_{ijk} \leqslant Q_k, \quad k = 1,\ldots,M;\ \ j = 0,1,\ldots,n+m, \tag{7}$$

$$\sum_{i=0,1,\ldots,n+m} x_{ijk} - \sum_{l=0,1,\ldots,n+m} x_{jlk} = 0, \quad j = 1,\ldots,n+m;\ \ k = 1,\ldots,M, \tag{8}$$

$$\sum_{j=0,1,\ldots,n+m} x_{ijk} - \sum_{l=0,1,\ldots,n+m} x_{lik} = 0, \quad i = 0,1,\ldots,n+m;\ \ k = 1,\ldots,M, \tag{9}$$

$$\sum x_{ijk} \leqslant |S| - 1, \quad S \subseteq \{2,3,\ldots,n+m\}, \tag{10}$$

$$\sum_{i=n+1,\ldots,n+m}\ \sum_{j=1,\ldots,n}\ \sum_{k=1,\ldots,M} x_{ijk} = 0, \tag{11}$$

$$x_{ijk} = \{0,1\}, \quad i = 0,1,\ldots,n+m;\ \ k = 1,\ldots,M. \tag{12}$$

The objective function of the proposed model is to minimize the transportation cost. Constraints (2) and (3) ensure that from any node except depot just one route passes. Constraints (4) and (5) denote that the number of vehicles leaving the depot must equal the number of vehicles returning to the depot. Constraints (6) and (7) state that vehicle load in terms of linehaul and backhaul must not exceed its capacity. Route continuity is enforced by Constraints (8) and (9). That is each route must be served just by one vehicle. Constraint (10) avoids the model generating subtour. Finally, the precedence of linehaul nodes over backhaul nodes is stated by Constraint (11).

4. Memetic algorithm for the VRPB

Memetic Algorithms (MAs) belong to the class of evolutionary algorithms (EAs) that apply a separate local search process to refine individuals (i.e. improve their fitness by hillclimbing). These methods are inspired by models of adaptation in natural systems that combine evolutionary adaptation of populations of individuals with individual learning within a lifetime. Additionally, MAs are inspired by Dawkin's concept of a meme [15], which represents a unit of cultural evolution that can exhibit local refinement. Under different contexts and situations, MAs are also known as hybrid EAs, genetic local searchers, Baldwinian EAs, Lamarkian EAs, and the like.

From an optimization point of view, MAs are hybrid EAs that combine the global and local search by using an EA to perform exploration while the local search method performs exploitation. Combining the global and local search is a strategy used by many successful global optimization approaches. In fact, MAs have been recognized as a powerful algorithmic paradigm for evolutionary computing. In particular, the relative advantage of MAs over EAs is quite consistent on complex search spaces.

In this paper, the proposed memetic algorithm is based on a simple conceptual framework using a simple population instead of complicated structured populations. In the main loop of the algorithm, either recombination or mutation is applied that followed by a local search to assure the local optimality of the individuals in the population. Fig. 1 shows the framework of the proposed memetic algorithm.

4.1. Representation and fitness function

The representation used in the proposed MA is straightforward. Each individual in the generation maps a solution for the VRPB. Each individual is composed of a sequence of nodes in which are divided by delimiter in the sequence. An individual chromosome begins from and ends with a depot node. The delimiter used in the sequence is the depot node, "1". It is used to break the sequence in different routes serving by different vehicles. Each route consists of different nodes including linehaul and backhaul nodes in which linehaul nodes have precedence in appearing on backhaul nodes. Fig. 2 depicts an individual representation.

```
Procedure MA;
{
    for (j=1;j++; j<=popsize)
    {
        i=GenerateSolution();
        i=Local-Search(i);
        add individual i to P;
    }
    while terminate!=true
    {
        for (i=1;i++;i<= #recombination)
        {
            select two parent i_a,i_b ∈ P Randomly;
            i_c =Recombine(i_a,i_b);
            i_c =Local-Search(i_c);
            add individual i_c to P;
        }
        for (i=1;i++;i<= #mutations)
        {
            select an individual i ∈ P Randomly;
            i_m=Mutate(i);
            i_m=Local-Search(i_m);
            add individual i_m to P;
        }
        P=select(P);
        if (P converged) P=mutateAndLS(P);
    }
}
```

Fig. 1. Pseudo code of the MA structure.

1	4	3	6	7	1	5	2	8	1

Fig. 2. Representation of an individual in MA.

One of important issues in EAs is the fitness function since it evaluates the quality of solutions in the algorithm and gives the algorithm capability of making comparison between individuals in the population. In this paper, the fitness function is defined as the cost of carrying goods from a depot to linehaul costumers and bringing them back from backhaul costumers.

A frequent problem that most EAs suffer from that is premature convergence. This phenomenon avoids the algorithm reaching to the global optima and search entire of search space. Premature convergence can also be avoided by scaling the fitness values [16]. Scaling can be useful in later runs when the average fitness of the population has become close to the fitness of the optimal solution. Thus, the average and the best individuals of the population are almost equally likely to be chosen. Naturally, the evaluation function and scaling of fitness values work together. Several scaling methods have been introduced like rank fitness scaling, proportional fitness scaling, top fitness scaling and power law fitness scaling.

The *rank fitness scaling* method scales the raw scores based on the rank of each individual instead of its score. The rank of an individual is its position in the sorted scores: the rank of the most fittest individual is 1, the next most fittest is 2, and so on. The rank scaling function assigns the scaled values so that the scaled

value of an individual with rank n is proportional to $\frac{1}{\sqrt{n}}$. The sum of the scaled values over the entire population equals to the number of parents needed to create the next generation. Rank fitness scaling removes the effect of the spread of the raw scores.

The *top fitness scaling* method scales the top individuals equally. The number of top individuals specifies the number of individuals that are assigned the positive scaled values. Each of the individuals that produce offspring is assigned an equal scaled value, while the rest are assigned the value 0. The scaled values have the form $[0, 1/n, 1/n, 0, \ldots, 1/n, 0, 0, 1/n]$.

The *power fitness scaling* method scales the fitness value by raising it to the power of k, depending on the problem $f_i' = f_i^k$.

The *linear scaling*method scales the fitness value as follows: $f_i' = a \times f_i + b$, where a and b are chosen so that the average initial fitness and the scaled fitness are equal. The linear scaling method is quite good but it runs into problems in later iterations when some individuals have very low fitness values close to each other, resulting in negative fitness values. Also, parameters a and b depend only on the population but not on the problem.

4.2. Initial population

Generally, evolutionary algorithms generate the initial population at random. This initial population helps the algorithm to be representative from any area of search space. In this paper, we examine employing a greedy heuristic algorithm to generate the initial solution in the population. A nearest neighbor algorithm is utilized to generate a good initial population in terms of the solution quality and speed of the generation. In Section 5, results of applying this algorithm for producing the first population (or generation) are investigated.

4.3. Selection

In the proposed algorithm, selection is based on the stochastic uniform selection and tournament selection. The stochastic uniform selection lays out a line in which each parent corresponds to a section of the line of the length proportional to its scaled value. The algorithm moves along the line in steps of the equal size. At each step, the algorithm allocates a parent from the section it lands on. The first step is a uniform random number less than the step size. The tournament selection chooses each parent by choosing n players at random and then choosing the best individual out of that set to be a parent. The tournament size (or subgroup, n) must be at least 2. This size influences the selective pressure, i.e. more individuals in the subgroups increase the selection pressure on the better individuals.

4.4. Crossovers

In the research carried out on the traveling salesman problem (TSP), several crossover operators have been proposed for a permutation representation such as partial-mapped crossover (PMX), order crossover (OX), position based crossover (PBX), and order-based crossover (OBX). These operators can be viewed as an extension of two-point or multi-point crossovers of binary strings to the permutation representation. Generally, two-point crossovers yielded infeasible offspring in sense of two or more nodes may be duplicated while some missed in the sequence. The repairing procedure is usually embedded in these operators in order to fix this problem. In this paper, we benefit from the well-known TSP crossovers with slight modifications to adapt them to the VRPB.

Partial-mapped crossover (PMX) has been proposed by Goldberg and Lingle [17]. This operator can be viewed as an extension of two-point crossover for binary string using a special repairing procedure in order to fix illegitimacy caused by two-point crossover. First, PMX selects two positions in the string and substitute substrings located between these positions. Then, it determines the relation between two mapping sections and finally arranges the other nodes according to discovered relations. Fig. 3 shows how PMX works.

Order crossover (OX) can be viewed as a kind of PMX with a different in repairing procedures. First, OX selects a substring from one parent randomly and it makes a new offspring by copying the substring in the same position. Then, it arranges other nodes according to their positions in another parent. Fig. 4 illustrates a representation of OX.

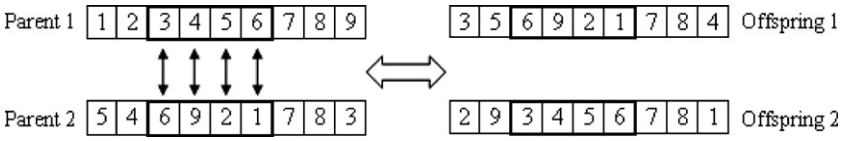

Fig. 3. PMX crossover.

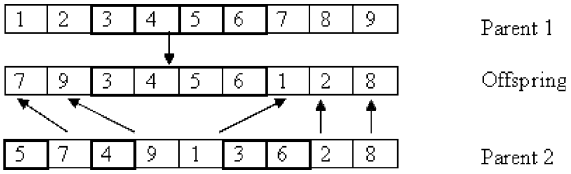

Fig. 4. OX crossover.

Position based crossover (PBX) is proposed by Syswerda [18]. It is essentially a kind of the uniform crossover. It also can be viewed as a kind of OX in which the cities are selected inconsecutively. First, PBX selects a set of positions randomly then copies them to the offspring and finally it puts other nodes according to their positions in the second parent. Fig. 5 depicts PBX.

Order-based crossover (OBX) is proposed by Syswerda [18]. It is a slight modification of PBX in which the order of nodes in the selected position in the first parent is imposed by the second parent.

4.5. Mutation

In this paper, the proposed MA uses different types of mutations. A number of these operators have previously been applied to the TSP such as inversion mutation, insertion mutation, displacement mutation, and reciprocal exchange mutation using different mutation rates. According to the experimental results, the displacement mutation is the most efficient performance among the others. The inversion mutation also produces better results after the displacement mutation. The displacement mutation selects a subtour at random and inserts it in a random position. Moreover, the insertion mutation can be viewed as the displacement mutation in which the substring contains only one node. Fig. 6 shows how the displacement mutation works.

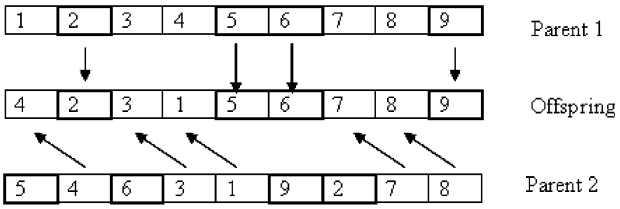

Fig. 5. PBX crossover.

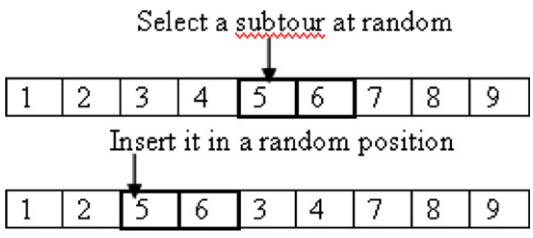

Fig. 6. Displacement mutation.

4.6. Local search

Tavakkoli-Moghaddam et al. [19] proposed a hybrid simulated annealing (SA) based on the nearest neighborhood (NN) algorithm to solve a linear-integer model of the VRP independent of the route length to minimize the heterogeneous fleet cost and maximize the capacity utilization. This NN can be considered as a local search method. The experience of the last few years has shown that combining simulated annealing or genetic algorithms (GAs) with local search (LS) algorithms is required to effectively solve the VRP. The LS can be used to improve VRP solutions in two ways. They can either be improvement heuristics for the TSP that are applied to only one route at a time or multi-route improvement methods that exploit the route structure of a whole solution. Different local search algorithms introduced in this paper. These algorithms can be divided in terms of the computational time and range of adjustment that they make (i.e. intra or inter-route adjustment). This work employs four local search methods as follows: 2-Opt, adjacency improvement, 1-move, and 1-exchange. The first three methods can be considered as the fast LS while 1-exchange may be judged as the slow LS. All LS used in this paper is an inter-route operator and make changes in more than on route.

4.6.1. 2-Opt

Most local search heuristics for the TSP and VRP can be described in a similar way as Lin's λ-Opt algorithm. The algorithm removes λ edges from the tour and the remaining segments are reconnected in every other possible way. If a portable reconnection is found (i.e. the first or the best), then it is implemented. The process is repeated until no further improvement can be made and thus a locally optimal tour has been obtained. The most famous LS methods are the simple 2-Opt and 3-Opt algorithms ($\lambda = 2$ and $\lambda = 3$). The 2-Opt algorithm removes two edges from a tour and reconnects the resulting two subtours in the other possible way.

4.6.2. Adjacency improvement

Adjacency improvement is a very simple and effective algorithm. It tries to substitute to adjacent nodes in the sequence to make saving and to improve the solution. A pseudo code of this algorithm is presented in Fig. 7.

In this local search, one important advantage of applying a local search is apparent. It is that the configuration space can be searched very efficiently instead of calculating the objective value of each neighbor. It is sufficient to calculate the difference $\Delta f = f(s) - f(s')$ and to test whether Δf is less than zero. The calculation of Δf takes time O(1), while the calculation of f takes O(n). In other words, the LS algorithm is able to visit n solutions of the search space within the same time. The traditional EA evaluates a single solution.

4.6.3. 1-Move

1-Move is the same heuristic as operators $(1,0)$ and $(0,1)$ proposed by Chiang and Russell [20]. The 1-move procedure effectively improves the quality and feasibility of poor solutions. 1-Move deletes one node from a

```
Adjacency Improvement

For (I=2; (length (Solution)-2); I++)
{
    FittBeforChange=Dis(Solution(I-1),Solution(I))+Dis(Solution(I+1), Solution(I+2));
    FittAfterChange=Dis(Solution(I-1),Solution(I+1))+Dis(Solution(I), Solution(I+2));
    If (FittBeForChange>FittAftterChange)
    {
        Swap (Solution (I), Solution (I+1));
        Saving=(FittBeForChange-FittAftterChange);
    }
}
```

Fig. 7. Pseudo code of the adjacency improvement algorithm.

route and inserts it into another route. The evaluation of 1-move is decided by the length of the arc and capacity violations change caused by 1-move. The neighborhood of 1-move is all the customers and any potential customer positions for all pairs of routes in the current solution. A repairing procedure is added to the algorithm and makes it compatible to the VRPB.

4.6.4. 1-Exchange

The 1-exchange heuristic deletes two customers from two different routes and inserts each customer into the best position in the other route. 1-Exchange first deletes the two customers, x and y, from the first and second route respectively. By having done that, customers x and y search for a good position in the second and first routes respectively. The best combination of these two insertions (if the result has a better objective function value than the original configuration) is the result of 1-exchange intensifying the local search. The neighborhood of 1-exchange is larger than that of 1-move since each of the customers needs to find its best position in the other route.

5. Experiments and computational results

This section presents experimental results. Three experiments are carried out to test the performance of the proposed memetic algorithm (MA). The first experiment evaluates the effectiveness of using the nearest neighbor algorithm in order to generate the initial population. The second experiment examines the performance of MA in comparison with the mathematical programming method. Following tests investigate in functioning MA related to different heuristic algorithms that propose to solve the VRPB taken from the literature.

5.1. Memetic algorithm implementation

The proposed memetic algorithm is programmed in the Microsoft Visual C++ 6 and executed on a Pentium 4 2.8 MHz pc. The performance of the proposed algorithm is tested on two sets of instances. First, a set is made up according to Goeschalckx and Jacobs [8] to generate test cases. Nodes coordinates are uniformly distributed in the interval $[0, 2400]$ for the x values, and in the interval $[0, 32{,}000]$ for the y values. The depot is located centrally in $(12{,}000, 16{,}000)$. The costs of arcs are defined as the Euclidean distances between nodes. Nodes demands are generated from the normal distribution with the mean value 500 and standard deviation 200. Vehicle capacities are defined so that with different sizes serve the nodes' demand and fleet in this set of problems is heterogeneous. The second set of problems is taken from Toth and Vigo work [11] to compare the proposed algorithm with their approach and two other heuristic methods (DB and SF) which are implemented by them that can be compared with TV. This set contains 15 test cases which have different ratio of backhaul nodes. The problems dimension range between 21 an 101 with linehaul percentage equal to 50%, 66%, and 80%.

5.2. Computational results

The mathematical model proposed in Section 3 is implemented by the Lingo 6 software package. The associated results are obtained by running the model on test problems. Table 1 reports the results obtained by the proposed MA and the mathematical programming method on the first set of instances. For each problem, we give the problem name, number of backhaul nodes (m), problem size (n), and number of vehicles. For each solution method, Table 1 presents the following output:

1. The solution value.
2. The percentage ratio of gap respect to optimal solution value obtained from mathematical programming approach.
3. The computation time expressed in seconds.

Table 1 shows that MA performs as good as the mathematical programming in term of the solution quality. MA solves such a hard problem in the maximum gap of 2.26%. The greater effectiveness of MA can be also

Table 1
Test problems generated according to Goetschalckx and Jacobs method

Name	*m*	*n*	*K*	MA		Lingo		Gap (%)
				Time	Solution	Time	Solution	
Eli 14	5	14	2	42	119,106	0	117,141	1.68
Eli 16	6	16	2	37	112,393	20	112,385	0.01
Eli 17	6	17	2	46	168,952	627	168,944	0.00
Eli 22	7	22	3	98	160,617	8056	157,066	2.26
Eli 22	4	22	3	108	170,439	135,420	167,613	1.69
Eli 51	15	51	6	135	319,946	–	–	–
Eli 101	50	101	8	165	574,873	–	–	–
Average				***66.2***	***14,6301.4***	***28,824.6***	***14,4629.8***	***1.13***

seen by noting that over all the instances, the average percentage gap is 1.13%, while the solution time is very slight in contrast with the computational time reported by the mathematical programming. The solution time for the mathematical approach as expected before rises drastically when the problem size increases. This phenomenon results in disability of the approach to solve problems with more than 22 nodes.

The second experiment examines the effect of using the nearest neighbor (NN) algorithm to generate the initial population on the performance of MA. This experiment employs the Eli 51 test problem. In first three runs, the initial population is generated at random while in the next three runs the NN algorithm is used. The results for the second experiment are illustrated in Table 2. These results verify that applying the nearest neighbor algorithm in the proposed MA improves its performance in terms of the solution quality and solution time. As showed in Fig. 8, the proposed MA with the NN algorithm starts the evolution from more fit generation which speeds the evolution process and better solution will be achieved. It is worth noting that the use of the NN algorithm also affects the diversity of the population and improves the MA performance.

Table 2
Computational results of using the nearest neighbor algorithm in the proposed MA

Test	Nearest neighbor		Random generation	
Time	Solution	Time	Solution	Time
1	622	228	**630**	**549**
2	608	220	671	456
3	**601**	**283**	716	195
Average	***610.3***	***243.6***	***672.3***	***400***

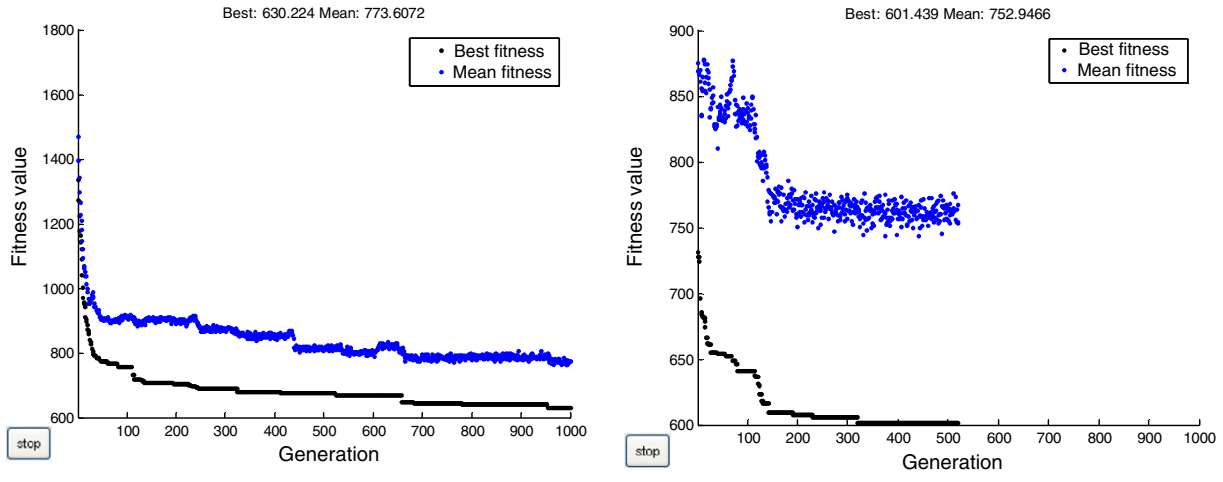

Fig. 8. Comparison of MA with the NN (left side) and MA without the NN (right side).

Table 3
Computational results, in which test problems are taken from Toth and Vigo [11]

Name	n	m	K	DB		SF		TV		MA	
				Sol.	Ratio (%)	Sol.	Ratio (%)	Sol.	Ratio (%)	Sol.	Ratio (%)
Eil22_50	11	10	3	429	115.63	492	132.61	**371**	100.00	373	100.54
Eil22_66	14	7	3	424	115.85	493	134.70	**366**	100.00	367	100.27
Eil22_80	17	4	3	375	100.00	526	140.27	**375**	100.00	378	100.80
Eil30_50	15	14	3	577	115.17	728	145.31	**501**	100.00	508	101.40
Eil30_66	20	9	3	594	110.20	802	148.79	**539**	100.00	551	102.23
Eil30_80	24	5	3	586	115.58	778	153.45	522	102.96	**507**	100.00
Eil51_50	25	25	3	669	119.04	696	123.84	**562**	100.00	637	113.35
Eil51_66	34	16	4	641	115.91	656	118.63	**553**	100.00	610	110.31
Eil51_80	40	10	4	655	114.11	703	122.47	**574**	100.00	601	104.70
Eil76_50	37	38	6	840	111.11	922	121.96	**756**	100.00	906	119.84
Eil76_66	50	25	7	907	116.28	1057	135.51	**780**	100.00	831	106.54
Eil76_80	60	15	9	913	105.67	972	112.50	883	102.20	**864**	100.00
Eil101_50	50	50	5	913	107.16	1021	119.84	**852**	100.00	995	116.78
Eil101_66	67	33	6	955	110.02	1125	129.61	**868**	100.00	1025	118.09
Eil101_80	80	20	7	956	106.22	1104	122.67	**900**	100.00	973	108.11
Average				***695.60***	***111.86***	***805.00***	***130.81***	***626.80***	***100.34***	***675.06***	***106.86***

The third experiment examines the proposed MA with other heuristics which are available for the homogenous VRPB. This experiment considers Toth and Vigo test cases. In this test, all vehicles have the same capacity and cost on the same arc. Table 3 shows the result of this test. For each problem, we give the problem name, number of backhaul nodes (m), problem size (n), and number of vehicles. For each solution method, this table also gives the solution and its percentage ratio respect to the best known solution value obtained from other heuristic algorithms.

This experiment shows that the proposed MA generally out performs the DB and SF algorithms. Moreover, the MA is able to compete the TV algorithm in terms of the solution quality while in some test problems the MA presents better solutions than the TV. Over all instances, the average percentage ratio of 106% shows that the MA has great effectiveness. However, the results explain weaknesses in large-sized test cases for the proposed MA in comparison with the TV algorithm.

6. Conclusion

In this paper, we have presented a memetic algorithm (MA) to solve the vehicle routing problem with backhaul (VRPB) and heterogeneous VRPB. The proposed algorithm used a greedy heuristic method to generate initial solutions in order to improve its performance in terms of the solution quality and computational time. The proposed MA employs different types of evolutionary operators such as PMX, OX, PBX, OBX, and several mutations which have already been applied for the traveling salesman problem (TSP). The proposed MA benefits from four inter-route local search algorithms generating local minimum populations. The MA performance is tested in two experiments. The MA has generated the competitive result in comparison with the mathematical programming approach. The computational time in the MA was reasonable while this value for the mathematical programming approach rises drastically in large-sized problems. The computational results of the third experiment stated that The MA outperformed better than the DB and SF algorithms, whereas it obtained suitable results in relation to the TV algorithm.

Acknowledgements

The authors would like to acknowledge the Iran National Science Foundation (INSF) for the financial support of this work. We would also thank the scholars who recommended and helped through the preparation of this paper.

References

[1] B.L. Golden, E. Baker, J. Alfaro, J. Schaffer, The vehicle routing problem with backhauling: two approaches. in: R. Hammesfahr (Eds.), Proceedings of the XXI Annual Meeting of S.F. TIMS, Myrtle Beach, SC, 1985, pp. 90–92.
[2] P. Toth, D. Vigo, An exact algorithm for the vehicle routing problem with backhauls, Transportation Science 31 (4) (1997) 372–385.
[3] C.A. Yano, T.J. Chan, L. Richter, T. Culter, K.J. Murty, D. McGettigan, Vehicle routing at quality stores, Interfaces 17 (2) (1987) 52–63.
[4] A. Mingozzi, S. Giorgi, R. Baldacci, An exact method for the vehicle routing problem with backhauls, Transportation Science 33 (1996) 315–329.
[5] I. Deif, L. Bodin, Extension of the Clarke and Wright algorithm for solving the vehicle routing problem with backhauling, in: A. Kidder (Ed.), Proceedings of the Babson Conference on Software Uses in Transportation and Logistic Management, Babson Park, 1984, pp. 75–96.
[6] G. Clarke, J.W. Wright, Scheduling of vehicles from a central depot to a number of delivery points, Operations Research 12 (1964) 568–581.
[7] O. Casco, L. Golden, A. Wasil, Vehicle routing with back hauls: Models, algorithms, and case studies, in: L. Golden, A.A. Asssad (Eds.), Vehicle routing: methods and studies, 1988, pp. 127–147.
[8] M. Goeschalckx, B. Jacobs, The vehicle routing problem with backhauls, European Journal of Operational Research 42 (1989) 39–51.
[9] M. Goestschalckz, B. Jachobs, The vehicle routing problem with backhauls: properties and solution algorithms, MHRC-TR-88-13, Georgia Institute of Technology, 1993.
[10] P. Toth, D. Vigo, A heuristic algorithm for the vehicle routing problem with backhauls, in: L. Bianco, P. Toth (Eds.), Transportation Analysis, 1996, pp. 585–608.
[11] P. Toth, D. Vigo, A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls, European Journal of Operation Research 113 (1999) 528–543.
[12] J. Potvin, C. Duhamel, F. Guertin, A genetic algorithm for vehicle routing with backhauling, Applied Intelligence 6 (1996) 345–355.
[13] R. Tavakkoli-Moghaddam, N. Safaei, M.M.O. Kah, M. Rabbani, A new capacitated vehicle routing problem with split service for minimizing fleet cost by simulated annealing, Journal of the Franklin Institute, in press, doi:10.1016/j.jfranklin.2005.12.002.
[14] C. Duhamel, J. Potvin, J. Rousseau, A tabu search heuristic for the vehicle routing problem with backhauls and time windows, Transportation Science 31 (1) (1997) 49–59.
[15] R. Dawkins, The Selfish Gene, Oxford University Press, 1976.
[16] Z. Michalewicz, Genetic Algorithm + Data Structures = Evolution Programs, third ed., Springer Verlag, 1996.
[17] D.E. Goldberg, R. Lingle, Alleles, Loci and the Traveling Salesman Problem, Lawrence Eribaum Associates, Mahwah, NJ, 1985.
[18] G. Syswerda, Uniform crossover in genetic algorithms, in: J.D. Schaffer (Ed.), Proceeding of the 3rd International Conference on Genetic Algorithms, 1989, pp. 2–9.
[19] R. Tavakkoli-Moghaddam, N. Safaei, Y. Gholipour, Capacitated vehicle routing problems (VRP) with the independent route length by hybrid simulated annealing, Applied Mathematics and Computation, in press, doi:10.1016/j.amc.2005.09.040.
[20] W. Chiang, R. Russell, A reactive tabu search metaheuristic for the vehicle routing problem with time windows, INFORMS Journal on Computing 9 (4) (1997) 417–430.