International Journal of Operations Research

International Journal of Operations Research Vol. 4, No. 4, 206–219 (2007)

Tuning the Parameters of a Memetic Algorithm to Solve Vehicle Routing Problem with Backhauls Using Design of Experiments

A. Saremi, T. Y. ElMekkawy*, and G. G. Wang

Department of Mechanical & Manufacturing Engineering, University of Manitoba, Winnipeg, MB, Canada, R3T 5V6

Received August 2006; *Revised March 2007*; *Accepted April 2007*

Abstract—Vehicle Routing Problem with Backhauls (VRPB) is an extension of the general Vehicle Routing Problem (VRP). In contrast with general VRP, VRPB considers two types of linehaul and backhaul customers. VRPB tries to find optimal routes with minimum cost in which backhaul customers are visited after linehaul customers for a fleet of heterogeneous vehicles. In this paper, a Memetic Algorithm (MA) is developed to solve the VRPB. Similar to other metaheuristic algorithms, an important issue that affects the performance of MA is the selection of components employed in the algorithm along with their parameters' values. This work examines the effect of employing different combinations of MA components and parameter values on both the algorithm's efficiency and the quality of solutions. Design of Experiments (DOE) is introduced as a systematic approach to find the best combination of these parameters' values. Analysis of variance (ANOVA) is used to analyze the main effect and interaction effects of the considered parameters. Results verified the efficacy of the proposed MA method and the systematic tuning approach for MA to solve VRPB.

Keywords—Memetic algorithm, Design of experiments, Metaheuristics, Vehicle routing problem

1. INTRODUCTION

Combinatorial optimization problems have attracted many researchers in recent decades due to their practical relevance and their considerable difficulties. Usually three different categories of approaches are used to solve these problems, namely, exact, heuristics, and metaheuristic methods. Exact methods apply analytical and mathematical approaches to solve the problems. Due to the considerable complexity of the problems, the efficacy of exact methods is limited to small scale problems. However exact methods yield analytically optimal solutions. Heuristic methods are employed to solve problems that are difficult to solve by using exact methods. They use simple intuitive techniques to make the search limited to potentially better solutions. Although heuristic methods can find a solution in a short time compared to exact methods, the analytically optimal solution is not guaranteed. To cope with the long computational time and limitation on quality of the obtained solutions, metaheuristic approaches are employed to solve combinatorial problems. Metaheuristic methods improve a solution's quality by exploring the search space while using heuristics to make the search more intelligent. Metaheuristic methods try to obtain quality solutions in a reasonable time, employing heuristic algorithms as a part of their search algorithm.

Metaheuristic methods have been applied to many research areas. In this paper, a new Memetic Algorithm (MA) will be developed to solve the Vehicle Routing Problem (VRP), a well-known combinatorial optimization problem. VRP is considered as the determination of optimal routes for fleet vehicles which are based on one depot to serve a number of customers. The optimality can be considered in terms of minimizing the total cost of transportation, minimizing the total distance of travel routes, and/or minimizing the total number of vehicles employed for serving customers. Several requirements and operational constraints confine VRP. For example, serving nodes can include deliveries along with pickups from customers, capacity of vehicles for carrying loads is limited, the length of each route should not exceed a pre-defined value, each customer should have a definite time of being served, the fleet can be composed of one or different types of vehicles, and some precedence relation can exist between customers. If the precedence is considered in VRP, then the problem becomes a Vehicle Routing Problem with Backhauls (VRPB).

The complexity of metaheuristic algorithms originates from the need of setting the values of several components and parameters within them. These parameters' values can drastically affect the performance of metaheuristic algorithms. Therefore, it is very important to develop a systematic method of tuning up these parameters for the best performance of the developed metaheuristic algorithms.

The objective of this paper is two folds. First it is to develop a Memetic algorithm (MA), as a metaheuristic algorithm, to solve the Vehicle Routing Problem with Backhauls (VRPB). Second this work is to introduce a systematic method for selecting the best combination of the employed parameters of the developed MA. Design of

* Corresponding author's email: tmekkawy@cc.umanitoba.ca

Experiments (DOE) will be used for analyzing the effect of several independent parameters on the performance of the developed MA measured by the computational time and quality of the obtained solution.

In the following sections, relevant literature concerning tuning parameters of an evolutionary algorithm by means of DOE is discussed. Problem description and the proposed Memetic Algorithm are considered in Section 3. Design of Experiments for the proposed MA algorithm is described in Section 4. Computational results are presented in Section 5. The last section is dedicated to conclusions and future research.

2. RELEVANT LITERATURE

Toth and Vigo (1997) developed a branch and bound algorithm in which a lower bound on the optimal solution is derived from their linear programming (LP) formulation with Lagrangian relaxation of constraints. Yano et al. (1987) developed a set covering approach based on the exact algorithm for a practical application of the VRPB. Deif and Bodin (1984) proposed a heuristic algorithm named DB. Deferent extensions of the DB algorithm were proposed by Casco et al. (1988). Goetschalckx and Jacobs-Blecha (1989) proposed an algorithm, which is called SF in the paper, for VRPB with the Euclidean cost matrix. The result presented in their paper showed that DB solutions were generally better than SF but the SF algorithm was faster for large scale problems. Later Goetschalckx and Jacobs-Blecha (1993) presented a heuristics, which was called LHBH, for the Euclidean version of the VRP. Their algorithm was based on the extension of Fisher and Jaikumar for the general VRP. Toth and Vigo (1996) developed a cluster-first route-second heuristic based on the K-tree approach for the VRP. Furthermore, Toth and Vigo (1999) implemented an improved version of the above heuristic to the symmetric and asymmetric VRPB which was called TV. This heuristic produced very competitive results in the literature for this problem. Duhamel and Rousseau (1997) designated a tabu search heuristic for the VRP with backhaul and time windows (VRPBTW) as well as customer precedence. Osman and Wassan (2002) published a tabu search metaheuristic which, on average, produced even better solutions than Toth and Vigo's algorithm, but required much more computing time. Zhong and Cole (2005) proposed a guided local search for the VRPB. More recently, Brandao developed a Tabu search algorithm for the same problem Brandao (2006).

In all the metaheuristic methods, it is generally recognized that different components and parameters significantly impact the performance of evolutionary algorithms. One of the impacts is seen in terms of the convergence of the algorithm toward optimum solution. Grefenstette (1986) used the Genetic Algorithm (GA) to evolve good values for crossover and mutation probabilities. Davis (1993) demonstrated that the difference between using randomly picked parameters' values rather than optimum values could easily delay the convergence. Many recent studies tried to find optimal parameter settings but there is no systematic method that specifies parameters' values and selects the type of components which can result in best performance. Thus the debate continues on if these settings are unique on each problem or they should be dynamically determined as the algorithm progresses. According to these views, Davis (1993) presented four different methods for adaptively setting the parameters to guarantee good convergence. Srinivas and Patnaik (1994) gave another adaptive GA in which low probabilities of crossover and mutation were assigned to individuals with high fitness; and high probabilities of crossover and mutation were assigned to individuals with low fitness. Bagchi and Deb (1996) proposed a DOE based approach for setting parameters of GA. Their approach also considered interaction of different parameters, e.g., probabilities of crossover and mutation. In their work, they used pilot GA runs of relatively short length of optimization in the factorial design framework. Their research demonstrated the effectiveness of the DOE approach in selecting algorithm parameters. Ruiz and Maroto (2006) and Ruiz et al. (2006) also used DOE to tune their hybrid GA, solving flowshop problems. Their algorithm was combined with different local search algorithms. Configuring such a complicated algorithm they used DOE to select the best GA component while previous works just considered DOE for setting parameter values.

In this work, performance of different components of MA along with a few parameters will be considered. Then, results of the examinations will yield a good configuration of the developed MA. The main difference between this work and the other methods of algorithm calibration is the consideration of interactions, which can play a major role in affecting the algorithm performance.

3. PROBLEM DESCRIPTION AND PROPOSED MEMETIC ALGORITHM

This paper considers a special case of VRP called the Vehicle Routing Problem with Backhauls (VRPB). This problem employs different types of vehicles in the fleet and assumes precedence relationships. There are two types of customers available in this problem. The first category concerns customers that should be served first. This service includes picking up the items from those customers. This type of customers is called linehaul or forward customers since the precedence calls for serving these in the forward phase of the route. The second type of customers is called backhaul customers and will be served by the vehicles in the return to the depot. There are limitations on the capacity of each vehicle so that the total loads which are assigned to a vehicle should not exceed its capacity for each type of the customers. A good example of this problem can be the grocery industry. Being a distributor company, you have two kinds of customer, suppliers and retailer. The goods are taken from the supplier to a depot and carried from the depot to retailers. A method to carry out this process is separately assigning

vehicles for each type of customers. The second method is assigning routes to the vehicles and includes both types of the customers in the same route. Since the vehicle should start from the depot and finish their route in the depot, it would be reasonable to consider that the retailers have a higher priority over the suppliers since the loads should be delivered to the retailers first before new loads are picked up from the suppliers. The cost of transportation in this problem plays the main role in choosing the routes and assigning customers to each route. Figure 1 demonstrates a graphical representation of the problem. The problem can be mathematically considered as a graph in which the nodes represent customers and the arcs represent the transportation network and existing routes that can be used by the vehicles to reach nodes. A mathematical model of VRPB has been used in this work is presented in Appendix 1. This model has been proposed by Tavakkoli-Moghaddam et. al. (2006).

3.1 Memetic algorithm

Different metaheuristic approaches are applied in the fields of routing and scheduling, specifically to solve VRP as a challenging NP-hard problem. Genetic algorithm (GA), Tabu search (TS), and Memetic Algorithms are the most known methods in this field. 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). 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 (Dawkins, 1976), 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 particular, the relative advantage of MAs over EAs is quite consistent on complex search spaces.

The structure of the proposed Memetic Algorithm is shown in Figure 2. Modules of the algorithm and their variations will be explained in the following sections. Some of these variations and their parameters are chosen as factors in the DOE study in search of the best MA configuration for VRPB.

3.2 Representation

The MA designed for VRPB uses a string presentation for solution. Each individual in the generation maps a solution for VRPB. Individuals are composed of a sequence of nodes in which routes (sequences of nodes) are divided by delimiters in the sequence. An individual chromosome begins from and ends with a depot node which is represented by "1". Also, the delimiter used in the sequence is the depot node, "1". It is used to break the sequence in different routes which assures that each route starts and ends in the depot. Figure 3 represents a solution of two routes that visit all the nodes, 1→3→4→5→7→1 and 1→2→6→1. Each route consists of different nodes including linehaul and backhaul nodes in which, linehaul nodes have precedence in appearing than backhaul nodes. For example, assuming backhaul nodes include 5, 6, and 7, the first route is divided to a linehaul part of 3 and 4 and a backhaul part of 5 and 7. It guarantees that backhaul nodes are served after the linehaul nodes to assure there is enough room in the vehicle for backhaul customers.

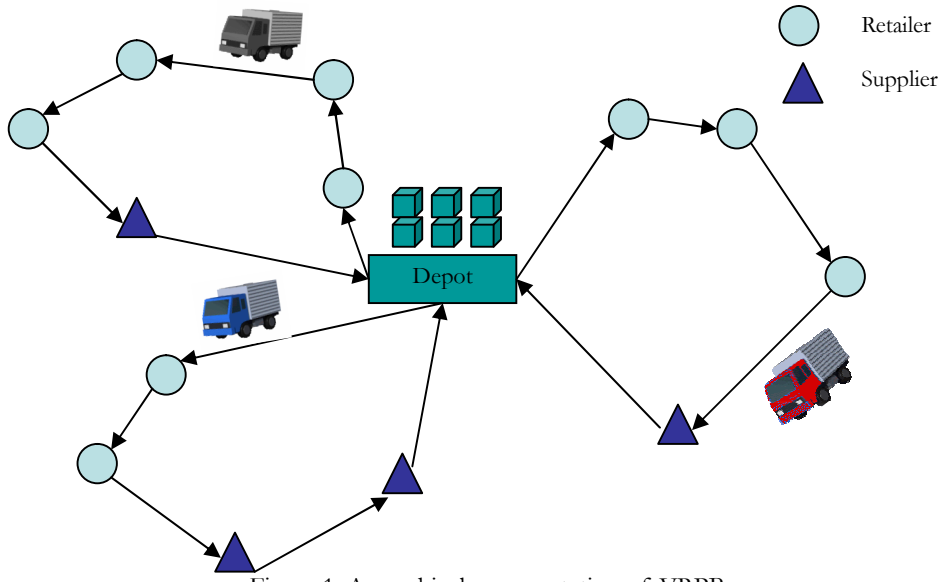

Figure 1. A graphical representation of VRPB.

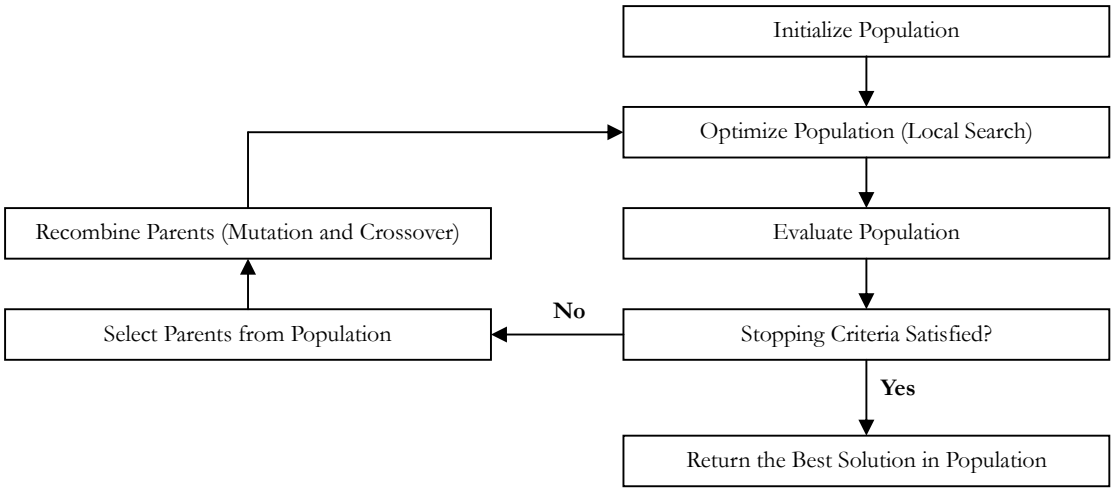

Figure 2. Framework of presented memetic algorithm.

1	3	4	5	7	1	2	6	1

Figure 3. Representation of a solution for VRPB.

3.3 Fitness function

The fitness function is an important part of an evolutionary algorithm since it evaluates the goodness of each solution in the population, based on which individuals are selected for reproduction, forming the next generation of the population. In this paper, the fitness function is defined as the total cost of serving all nodes through the determined routes. The total cost consists of the sum of each vehicle's traveling cost. The traveling cost of each vehicle, C_k, is defined by Eq. (1).

$$C_k = m_k \sum d_{ij} x_{ij} \qquad \forall i, j \tag{1}$$

where m_k is the vehicle cost multiplier; d_{ij} is the traveling cost for the distance between node i and j; and x_{ij} determines if the route includes the arc connecting i to j.

A common problem that population-based algorithms have is the premature convergence. The combinatorial nature of VRPB prevents an algorithm from searching the entire search space, therefore there is no guarantee of the global optimum solution. For the proposed algorithm, this problem escalades in later runs when the average fitness of population approaches the best individual's fitness. In this situation, the average and best individuals of the population are almost equally likely to be chosen and the search progresses slowly. The solution of this problem is to use fitness scaling techniques. Fitness scaling modifies the scores of individuals to make the discrimination possible through the new values. Different types of scaling are employed in this work for this problem such as the rank fitness scaling, proportional fitness scaling, top fitness scaling, and power law fitness scaling.

3.4 Initial population

Evolutionary algorithms benefit from a randomly generated population as their initial population. The randomness in generating the first population assures that the selected solutions are uniformly chosen from the search space. An alternative approach is to use a constructive algorithm to develop the initial population. Some of these approaches are explained in the following:

Randomly generated population: Using randomly generated population, some consideration should be noticed. Since the problem forces limitations on the capacity of vehicles and urges for precedence of linehaul nodes on backhaul nodes, some repairing procedures becomes necessary. First to fix the inconsistency problem with capacity constraints, the nodes that are included in each route should be considered along with their demand. Also in order to rearrange the chosen nodes and make them in the order, a procedure is employed to change the order of nodes so that all the backhaul nodes are placed after the linehaul node while keeping their initial arrangement in the randomly generated solutions.

Nearest neighborhood greedy method: A greedy constructive algorithm is also used to generate the initial population in the proposed MA. This algorithm first selects a linehaul node randomly. Then it considers the first vehicle. The second node is the closest linehaul node to the current node. Following nodes are selected according to their distance to the current node considering linehaul capacity of the first vehicle. When there is no excessive room for another linehaul node, the algorithm selects the closest backhaul node as the current node. The procedure continues till reaches the backhaul capacity limit. The next step is selecting the next vehicle and filling with nodes satisfying the mentioned criteria. In this work, the nearest neighbor method is used to generate the initial population.

3.5 Selection

Selection plays an important role in MA. It is used to determine which individuals are allowed to reproduce and develop the next generation. According to the natural

evolution, the fittest members of population have the right to reproduce because of their fitness to environment so that they are more probable to survive. However, in the EAs different methods are used to discriminate between different individuals such as the roulette wheel selection, uniform selection, tournament selection, and rank selection.

In this paper, two methods are utilized; namely, tournament and roulette wheel selection. Tournament selection runs a "tournament" among a few individuals chosen at random from the population and selects the winner (the one with the best fitness) for crossover. It chooses k (the tournament size) individuals from the population at random. Then the best individual from the tournament is chosen with a probability p. The second individual is chosen with the probability $p(1-p)$. The third best individual is selected with the probability $p(1-p)^2$, and so on.

In the roulette wheel selection, like other selection methods, possible solutions are evaluated by the fitness function. In this method, the fitness level is used to determine a probability of selection for each solution. Candidate solutions with a higher fitness will more likely be chosen. As an advantage of this selection method, there is a chance that some weaker solutions survive the selection process. Although a solution may be weak, it may include some components which could prove useful following the recombination process.

3.6 Evolutionary operators

Mutation and crossover operators are dependant on the coding of the candidate solutions of the optimization problem. Operators for sequential coding as used in GA and MA will be discussed in this section. Most of operators used in this work are borrowed from the literature on the traveling salesman problem (TSP) with some modification to comply with VRPB assumptions. It goes back to the nature of VRP which can be considered as a generalized TSP problem. Each route in VRP can be seen as an ordinary TSP problem so it is natural to expect good TSP evolutionary operators that can be applied to VRP problems.

3.6.1 Crossover

In GA and MA, the crossover is a genetic operator used to vary the composition of chromosomes from one generation to the next. In the research on TSP, several crossover operators have been proposed such as the 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, the two-point crossovers yield infeasible offspring because 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. This work employs these well-known TSP crossover operators with slight modifications. The repairing procedure has the main responsibility to adapt solutions to VRPB assumptions. Thus, when a solution is generated by crossover, a feasibility check is performed to investigate solution regarding its feasibility. In case that any infeasibility exists in the solution, the repairing procedure adjusts the solution.

Partial-mapped crossover (PMX) has been proposed by Goldberg and Lingle (1985). 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.

Order crossover (OX) can be viewed as an extension of the PMX with a difference 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.

Position based crossover (PBX) was proposed by Syswerda (1985). It is essentially a kind of uniform crossover. It also can be viewed as a kind of OX in which the nodes are selected inconsecutively. First, PBX selects a set of positions randomly then copies them to the offspring and finally puts other nodes according to their positions in the second parent.

Order-based crossover (OBX) was also proposed by Syswerda (1985). In this crossover, a set of positions is randomly selected and the order of cities in the selected positions in one parent is imposed on the corresponding alleles in the other parent. The difference between OX and OBX is that selected positions in OBX are separated from each other.

3.6.2 Mutation

The classic example of a mutation operator involves a probability that a randomly chosen bit in a genetic sequence will be changed from its original state according to a pre-determined mutation rate. The purpose of mutation in GAs is to allow the algorithm to avoid being trapped in local minima by preventing the population of chromosomes from becoming too similar to each other. The bits in a chromosome are often deemed independent. When evolutionary algorithms are employed to solve combinatorial optimization, however, the mutation rate concept is not directly applicable. It is because in these problems a chromosome is composed of a sequence of nodes, jobs or other elements that are dependent on each other. Therefore for these problems such as the VRPB in this work, mutation operators are often designed as an exchange procedure which randomly but systematically changes the position of an element in the sequence. The conventional mutation rate parameter is therefore not explicitly used. Mutation used in this paper is originated from mutation used for TSP with slight modifications. The

repairing procedure is the main procedure in the algorithm which takes after nonconformities and repairs them.

Inversion mutation: In this mutation, two positions are chosen randomly, which determines a substring. Then this substring is inverted. The inverted substring then replaces the original substring in the original string.

Insertion mutation: The operator first selects a random position in the sequence and then inserts the content in the position to a different, randomly-chosen, position.

Displacement mutation: The displacement mutation selects a substring 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. Therefore in this work, only the inversion and displacement mutation schemes are used in the experiments for MA tuning.

3.7 Local search algorithms

Local search (LS) can be used to improve VRPB 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 change the route structure of a whole solution. In general, improvement heuristics are characterized by a certain type of basic move to alter the current solution. Different local search algorithms are 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 paper employs three local search methods as follows: 2-Opt, Adjacency improvement, and 1-Move. All LS methods used in this paper make inter-route adjustment. In the proposed local searches, 2-Opt and 1-Move generate the solutions obeying the VRPB assumptions and preserve feasibility of solutions. However, in Adjacency improvement, the generated solutions are prone to be infeasible and need to be tailored to VRPB solutions using repairing procedure.

2-Opt: A 2-Opt move includes removing two edges and reconnecting the two resulting paths in a different way to obtain a new solution. Among all pairs of edges whose 2-Opt exchange decreases the length, the pair that gives the shortest tour is selected. This procedure is then iterated until no such pair of edges is found.

Adjacency improvement: Adjacency improvement is a very simple and effective algorithm. It tries to exchange two adjacent nodes in the sequence to improve the solution. If the exchange makes any improvement it will be implemented and the algorithm continues searching the rest of sequence for any potential changes. The developed algorithm is presented below. This local search method is efficient as only the difference $\Delta f = f(s) - f(s')$ between two adjacent nodes, s and s', are calculated. The calculation of Δf takes time $O(1)$, while the calculation of a new f takes $O(n)$. In other words, the LS algorithm is able to visit n solutions of the search space within the same time as that the traditional EA evaluates a single solution.

1-Move: The 1-Move approach is similar to the heuristic operators (1, 0) and (0, 1) modified by Zhong and Cole (2005). The 1-Move deletes one node from a route and inserts it into another route. The evaluation of 1-Move is decided by the cost of arcs and capacity violations caused by 1-Move.

3.8 Repairing procedure

After performing any operation including crossover, mutation, or local search, a feasibility check is performed on the solutions. If the solution is infeasible, a repairing procedure is called. The repairing procedure adjusts the infeasible solutions to get a feasible solution with minimum changes.

The repairing procedure considers two important issues in feasibility of each solution. First, the procedure determines each vehicle route in such a way that the capacity of vehicle is respected. The second important issue is the order of linehaul and backhaul nodes in the route of each vehicle. By the definition of VRPB problem it is advised that the occurrence of the backhaul node should be after the linehaul node. Thus, the repairing procedure has a routine to accommodate this need. Figure 4 describes the repairing procedure.

S_l = Linehaul subsequence.
S_b = Backhaul subsequence.
S_v = Sequence of nodes which is forming for current vehicle.
Seq_{fin} = The final solution which includes the sequence of nodes (routes) for the vehicles.
cl_i = Linehaul capacity of vehicle i
cb_i = Backhaul capacity of vehicle i
VS_{ij} = Node j of the sequence of nodes which determines the route for vehicle i
$ncapl_j$ = Linehaul demand of jth node in the sequence
$ncapb_j$ = Backhaul demand of jth node in the sequence
nv = Number of available vehicles
nc = Current node
nc_l = Current node linehaul demand
nc_b = Current node backhaul demand
Rej = Rejection counter
Rej_{max} = Maximum number of allowed rejections
Seq_{inp} = Input solution which is composed of the infeasible sequence of nodes for all the vehicles.

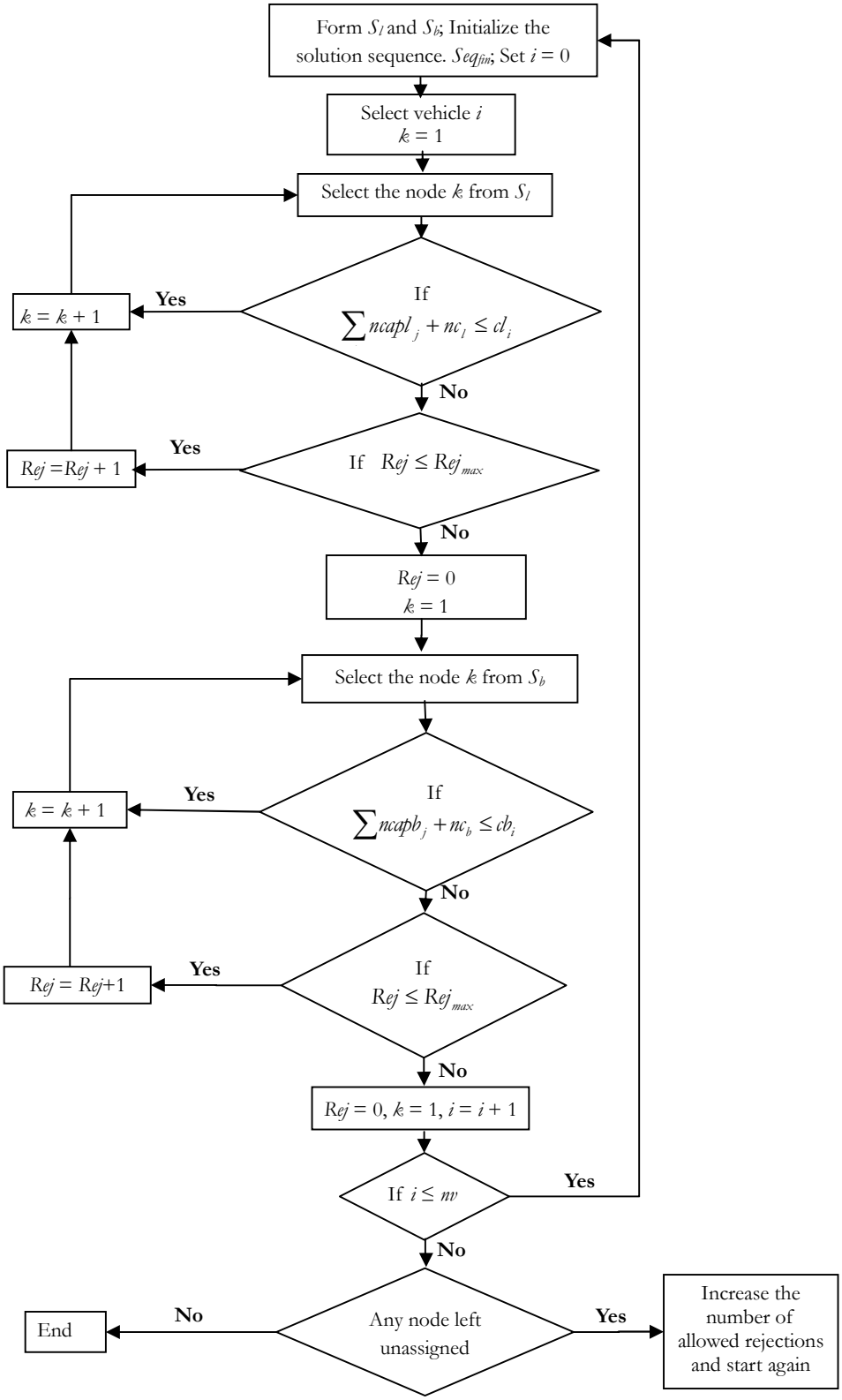

Figure 4. Flowchart of repairing procedure.

4. DESIGN OF EXPERIMENTS

Design of experiments is an effective approach for evaluating the effect of multiple factors on a process performance. Its efficiency and effectiveness in the analysis of multi-factor cause-response relationship has been studied rigorously (Montgomery, 1997) and its associated data analysis approach (ANOVA) is widely used (Cochran and Cox, 1957). Therefore, this work applies DOE on developed MA to identify the factors that might have significant effect on its performance. Moreover, an important consideration is the interaction between studied factors. Two factors are said to interact if the observed effect of one factor depends on the other factors. Approaches rather than DOE usually ignore this interaction effect. In addition, rarely in evolutionary

algorithms, the efficacy of an algorithm is dominated by a single factor. DOE can find the effect of individual factors, as well as it can reveal any significant interaction among factors.

One objective of this paper is to show the potential efficacy of DOE as a general tool for parameterization of algorithms, regardless of being MA or simulated annealing. It should also be noted that specific parameter values obtained from applying DOE are normally context specific. This is because the value of parameters including the type of operators and values of probabilities for operators may interact in a complex manner with the structure of the solution space. In this work, DOE is used as a methodology for tuning parameters and components to get the best performance from the proposed MA for VRPB. Analysis of variance (ANOVA) will be used as a tool to study effects of different factors on the performance of the developed algorithm.

4.1 Factors affecting the performance of an MA

In spite of the fact that many parameters affect the performance of an algorithm, it is natural to focus on those which are expected to have a significant effect. Thus, the first step is to identify these parameters and components. The considered factors within this research along with their selected levels are shown in Table 1 and are discussed in the following.

Crossovers type and crossover rate: Considering crossover type as an effective factor, we introduce four levels for this factor. These levels are PMX, OBX, PBX and OX that are explained in Section 3.6.1. According to our experience it is predicted that factors such as the crossover rate have an important effect on the algorithm. Basically, the crossover rate determines the probability of performing crossover on the parent to generate the next generation. The main impact of this factor can be seen on the convergence of population and obviously on the computational time. Four different levels that are selected for this factor are 0.2, 0.4, 0.6 and 0.8.

Mutation type: Two different mutation operators, i.e., displacement mutation and inversion mutation, are introduced in the algorithm and they are treated as two levels of the mutation factor in the experiment.

Local search: Local searches in MA assure that a population is only composed of local optimum solutions. In GA, evolutionary operators are responsible for finding the optimum solution in the algorithm, because of which these operators could become very complex. For example for a routing problem, if we want to use a simple GA, it is inevitable to employ very complex crossover which can survive route patterns and can avoid disturbance of routes inherited from parents. In contrast, MA evolutionary operators are only responsible for introducing promising regions to local searches. Then it is the local searches' job to find local optima in the region. Hence, the structure of evolutionary operators in the MAs can be simpler and this modification is compensated by involving different local searches in the algorithm. Local searches affect the quality of solution and computational time of the MA algorithm. In this study it is important to find out which local search can generate better solutions in less computational time. Since local searches are the most computation demanding part of this algorithm, it is predicted that this factor has significant effect on computational time. There are three local searches available in the algorithm. Three levels of this local search factor are 2-Opt, 1-Move, and adjacency improvement methods.

Selection methods: Selection methods concern the quality of selecting individuals for reproduction. There are different methods for performing selection in evolutionary algorithms. Different methods could lead to different results. Therefore it is important to understand which type of selection is of benefit for the current problem. In this paper two selection methods are considered and will be investigated as levels of this selection factor, shown in Table 1.

Table 1. Factors and their levels

Factors	*Levels*
Crossover type	-Partial-mapped crossover -Order crossover -Position based crossover -Order-based crossover
Crossover rate	0.2, 0.4, 0.6, and 0.8
Mutation type	-Displacement mutation -Inversion mutation
Local search type	-Adjacency improvement -1-Move -2-Opt
Selection method	-Tournament -Roulette wheel

5. ANALYSIS OF RESULTS

This section presents the result of experiments performed on the developed MA equipped with different combinations of factors' levels. Two sets of experiments are carried out to examine the effect of different factors on the performance of the algorithm for each test case. The first set of experiments deals with the performance of algorithm in terms of solution quality; while the second investigates the factors which play a significant role in the computational time of the algorithm. Experiments are performed on an Intel Pentium 4 2.4 MHz PC with 512 MB of RAM. Test cases used in the experiments are borrowed from the literature on VRPB Toth and Vigo (1999). Three test cases used for the experiment differ in the size of VRPB. The first test case includes 21 nodes using three vehicles. The percentage of backhaul nodes in the problem is 50%. The second test case considers 30

Table 2. Describes the parameter settings for first part of experiments

Parameter	Eli 21_50	Eli 30_80	Eli 51_80
Generations No.	500	500	500
Population size	63	63	63
Fitness scaling	Top scaling	Top scaling	Top scaling
Elite No.	6	6	6

Table 3. Describes the parameter settings for second part of experiments

Parameter	Eli 21_50	Eli 30_80	Eli 51_80	Eli 75_80
Generations No.	500	500	500	500
Population size	110	200	300	400
Fitness scaling	Top scaling	Top scaling	Top scaling	Exponential
Stall generation number	150	200	200	200
Elite No.	20	20	40	40

Table 4. ANOVA results concerning solution quality

Source	Eli 21_50		Eli 30_80		Eli 51_80	
	F-ratio	*P*-value	*F*-ratio	*P*-value	*F*-ratio	*P*-value
A: Crossover	4.79	0.003	8.81	0	**68.53**	0
B: Mutation	**2575.69**	0	**915.98**	0	0.49	0.484
C: Crossover rate	23.52	0	23.55	0	**38.28**	0
D: Local search	**281.31**	0	**29.75**	0	**79.61**	0
E: Selection	28.39	0	1.63	0.202	17.07	0
AB	6.81	0	3.39	0.018	2.3	0.076
AC	2.52	0.008	0.94	0.486	5.02	0
AD	4.26	0	2.64	0.016	13.18	0
AE	1.15	0.327	2.79	0.04	1.11	0.344
BC	26.14	0	13.78	0	7.54	0
BD	**90.35**	0	**193.55**	0	27.7	0
BE	34.66	0	14.77	0	0.19	0.664
CD	5.84	0	1.77	0.102	21.2	0
CE	0.93	0.427	1.4	0.243	5.72	0.001
ED	22.31	0	2.87	0.058	8.28	0

nodes and employs 4 vehicles to serve the customers. The ratio of backhaul nodes is 80% in this problem. The third problem includes 51 nodes and employs 6 vehicles to serve customers. Backhaul nodes compose 80% of all nodes. The MA is executed with MATLAB Version 14; and the Minitab 14 software is used for performing analysis of variance. Each experiment contains four replications for each of 192 different combinations according to levels which are considered for each factor. In the first part, the effect of different factors which we assume that are significant is investigated. The other parameters which are used in the experiments but are not investigated are presented in Table 2. These parameters include generation, population size, fitness scaling, and number of elite individuals in each generation.

In the second part of this section, the values which are recognized as the best values for significant parameters are used to evaluate performance of the Memetic algorithm with the other well known algorithms from the literature. For this comparison, the other selected parameters' values are shown in Table 3.

5.1 Analysis of variance: Solution quality

This section will first analyze the main effects of crossover type, crossover rate, mutation type, local search type, and selection type, as well as their interactions, on the quality of obtained solution. Totally, for each test case 768 experiments are carried out (192 for a full factorial design times 4 replications). Table 4 shows the ANOVA result from the experiments.

Table 4 includes the test case name (e.g. eli21_50 means 21 nodes with 50% backhaul nodes), *F*-ratio and *P*-value of ANOVA. According to the results, for the first test case, the mutation type is the most effective parameter in the algorithm, followed by the local search type. Interaction of the two factors is of importance too. Therefore, different combinations of local searches and mutations lead to dissimilar performances of the algorithm.

As the number of nodes in the problem grows and the problem size increases, for the second problem the situation is different. Mutation is still the most important factor in the performance of the algorithm; however the interaction between mutation type and local search bears more significance than before and seems to be more important than the local search type due to its higher F-ratio. The next important factor is the local search type

used in the algorithm.

The third class of experiments uses a test case with 51 nodes. The situation is different from the first experiment. Crossover in this class plays the most important role instead of mutation which was previously the most important factor for smaller problems. The local search type is still the second important factor. Interaction of the crossover and local search type is also of importance.

To select the best values for factors and to obtain best quality solutions, it is necessary to examine the average of each treatment. As the problem is a minimization problem, the level should be chosen that leads to the minimum average. For example for the first test case, Figure 5 shows the average results for each significant factor. As seen, displacement mutation and 2-Opt can be selected as the best choice for the type of mutation and local search, respectively, in the algorithm for this problem. Also, the interaction of mutation and local search type has a significant effect on the performance of the algorithm. Figure 6 verifies that selecting the displacement mutation along with 2-Opt local search leads to the best solution quality for the first test case. As seen from the ANOVA results, for solution quality of small scale problems, the mutation and local search play a main role. This is due to the ability of displacement mutation in introducing new regions of search space to the algorithm. This helps the algorithm to escape from local minima and explore for better solutions. The local search has significant impact since it uses the present solutions to find optimum solutions. Additionally, applying displacement mutation along with 2-Opt leads to very good results in the first test case through their ability to complement each other in terms of finding more promising regions and finding the local optimum in the designated region. In the second test case as the size of problem increases, the 1-Move local search shows better results. When the problem size becomes larger, the interaction of the local search and crossover becomes more important. For the third case, the crossover is the most important factor while the mutation is not of importance. As the size of the test case in the third problem is almost twice of the second test case, the number of nodes in each route grows substantially. Mutation is no longer capable of introducing rich regions of search space and thus the crossover operators such as OBX play a more important role in finding better solutions.

5.2 Analysis of variance: Computational time

For the study of computational time, factors involved in this experiment are as the previous study, i.e., the crossover type, crossover rate, mutation type, local search type, and selection type as shown in Table 1. Three test cases are chosen and for each case 768 experiments are performed. In this experiment a full factorial design is used and factors and two-factor interactions are reflected in the result. Table 5 shows the ANOVA results.

Results for the three test cases show consistently that the local search type, crossover rate, and crossover type have main impacts on the computational time. It is predictable that the local searches have the most significant effect on the computational time since they are invoked after every change takes place in the population and demands major computational efforts to find local optimum solutions for each individual in the population. Also, the crossover rate has a direct relationship with the computational time since it determines the amount of crossover operation as the crossover is a time consuming operator. In addition, crossover operators by themselves are accountable for a considerable portion of computations since different crossover operators have different computational demand.

To select the best values, averages of levels of each significant factor are considered. For example for the first test case, the 1-Move method is selected as the most time saving local search. Also it is obvious that the smaller crossover rate, the less time is needed for computation. For the crossover type, Figure 7 shows that the PBX crossover leads to the least calculation time on average.

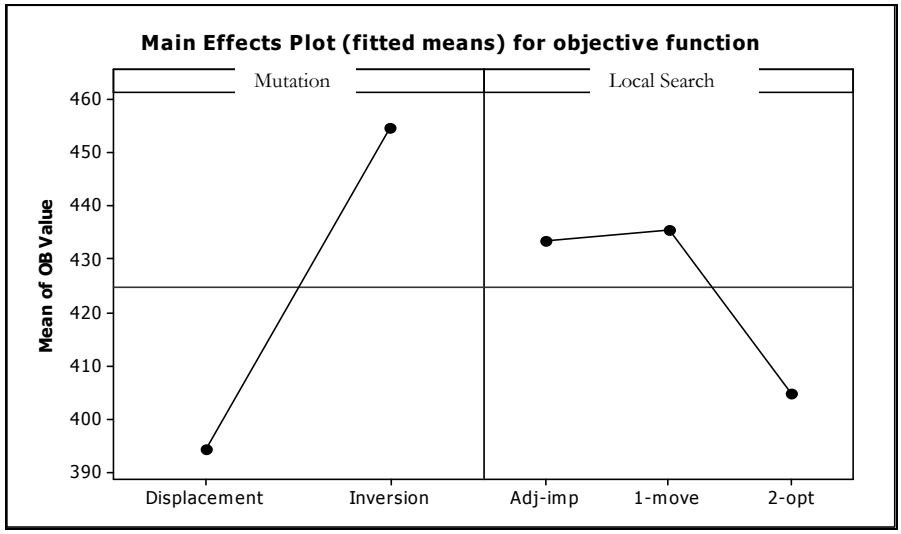

Figure 5. Different level averages for mutation and local search for the first test case.

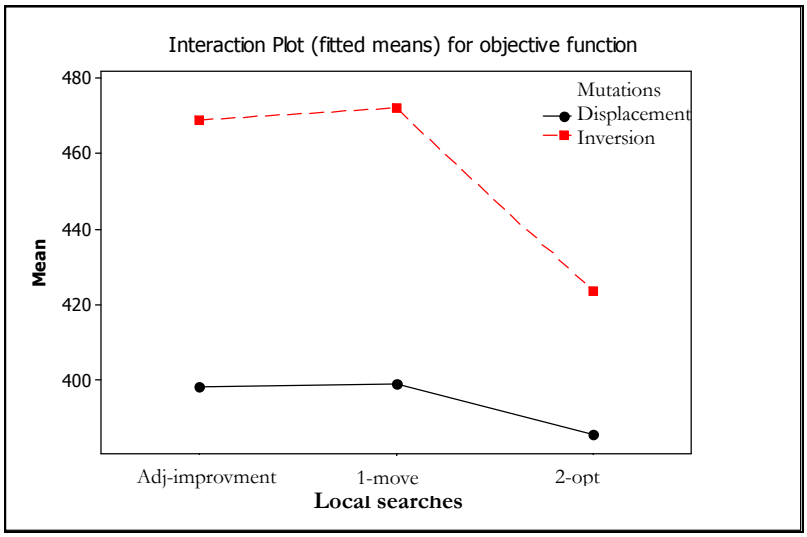

Figure 6. Level averages in interaction of local searches and mutations for the first test case.

Table 5. ANOVA results for the computational time experiments

Source	Eli21_50		Eli30_80		Eli51-80	
	F-ratio	*P*-value	*F*-ratio	*P*-value	*F*-ratio	*P*-value
A: Crossover	**78.8**	0	**90.45**	0	**453.47**	0
B: Mutation	1.25	0.265	0.48	0.489	0.29	0.589
C: Crossover rate	**288.61**	0	**302.32**	0	**1006.27**	0
D: Local search	**3809.72**	0	**15065.26**	0	**669947.7**	0
E: Selection	3.09	0.079	2.17	0.141	12.09	0.001
AB	0.93	0.427	0.71	0.544	0.69	0.556
AC	5.49	0	7.45	0	27.24	0
AD	1.25	0.281	1.22	0.295	2.47	0.023
AE	0.63	0.597	1.11	0.345	0.71	0.544
BC	0.01	0.998	0.6	0.616	0.88	0.451
BD	3.32	0.037	1.49	0.226	1.97	0.14
BE	0.02	0.892	2.11	0.147	3.88	0.049
CD	0.39	0.884	1.42	0.203	0.93	0.47
CE	0.76	0.519	0.32	0.81	2.52	0.057
ED	0.44	0.643	1.84	0.16	1.75	0.174

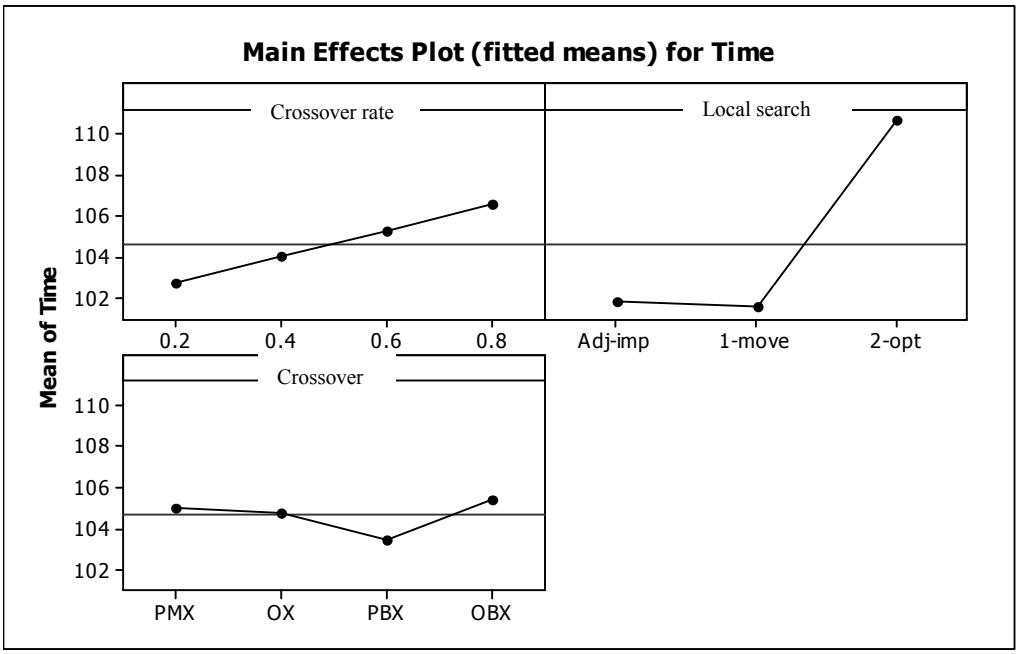

Figure 7. Different levels of effective factors in computational time for the first test case.

Table 6. Summary of results

Test case	Solution quality		Computational time	
	Important factors	Selected component	Important factors	Selected component
Eli 22_50	Mutation, Local Search, and their interaction	Displacement, mutation , and 2-Opt	Local search, Crossover rate, and type	Crossover rate = 0.2, PBX and 1-Move
Eli 30_80	Interaction of local search and mutation, Mutation and local search	1-Move and displacement mutation	Local search, Crossover rate and type	Crossover rate = 0.2, PBX and 1-Move
Eli 51_80	Crossover, Crossover rate and local Search	Crossover rate = 0.6, OBX and 2-Opt	Local search, Crossover rate and type	Crossover rate = 0.2, PBX and 1-Move

Table 7. Comparison of tuned MA and mathematical programming

Name	Number of nodes	Number of backhaul nodes	MA		Mathematical programming		Gap (%)
			Solution	Time (s)	Solution	Time (s)	
Eli16	16	6	112393	116.40	112385	65	0.01
Eli17	17	6	168952	121.09	168944	2619	0.00
Eli22_1	22	7	160617	169.63	157066	12897	2.26
Eli22_2	22	4	170439	340.47	167613	225297	1.69

Table 8. Comparison of the tuned MA with other heuristics for VRPB

Name	*N*	*m*	*k*	DB		SF		TV		Brandao		MA	
				Sol.	Ratio (%)	Sol.	Ratio (%)	Sol.	Ratio (%)	Sol.	Ratio (%)	Sol.	Ratio (%)
Eil22_50	11	10	3	429	115.63	492	132.61	**371**	100.00	371	100	384.95	103.76
Eil30_80	24	5	3	586	114.36	778	151.83	522	101.87	514	100.30	**512.43**	100.00
Eil51_80	40	10	4	655	115.93	703	124.42	**574**	101.59	565	100	612.13	108.34
Eil76_66	50	25	7	907	118.1	1057	137.63	780	101.56	**768**	100	831	108.20
Average				644.25	116.	757.5	136.62	561.75	101.26	554.5	100.08	593.5775	105.80

Parameters and their values are recommended based on the ANOVA results for both the solution quality and computation time, which are summarized in Table 6.

To set the configurations in such away that yields an acceptable solution quality in reasonable computational time, a rule of thumb can be derived from Table 6. In case that there is no contradiction among the settings for both criteria, these settings can be directly employed. For example, in the test case Eli 30_80, the solution quality asks for the use of 1-Move local search and displacement mutation while computational time asks for employing PBX crossover with the rate of 0.2 and using 1-Move, which is also demanded by the solution quality. Therefore for this case, the PBX crossover with a rate of 0.2, 1-Move mutation, and displacement mutation can be used for both quality and time criteria. However, in case that there is a contradiction in recommendations for these criteria, whichever criterion has more importance, the settings for that criterion will be used. For example, in the test case Eli 22_50, assuming the computation time is more of concern, for the local search method, 1-Move will be applied rather than the 2-Opt method to favor the computation criterion.

5.3 Comparison of tuned MA with mathematical programming and heuristic methods

To demonstrate the efficacy of the developed MA after parameter tuning, it is inevitable to compare its performance with other methods. Thus, the mathematical model presented in the appendix is used to solve some test cases. Test cases for this comparison include 4 different problems with size of 16, 17, 22 and 22 nodes in which two last problems differ in the percentage of backhaul nodes. These test cases are used since it is very difficult to solve problems with more than 22 nodes by means of the mathematical programming approach due to the huge amount of computational time needed for solution.

The mathematical model for each test case is solved using Lingo 6.0 software. Table 7 presents the results of comparison. Results show that there is a gap (less than 2.5%) between the analytical optimal solution and results obtained by the tuned MA. The MA, however, is more efficient than the mathematical programming methods in larger problems since it is not possible to solve such problems in a reasonable time using exact methods. The computational time for mathematical programming in the last test case is about 63 hours while the tuned MA takes only 340.47 seconds.

The tuned MA is also compared with heuristic approaches in the literature. Four heuristic methods from the literature are considered in this study. These are DB, SF and TV that are implemented by Toth and Vigo (1999) on Intel 486/33. Brandao (2006) also used the same test cases to evaluate its tabu search algorithm. As the computing hardware is different, it lacks of a common ground for computation time comparison and thus only the solution

quality is compared. Table 8 presents the result of comparing MA algorithm with other four metaheuristics that are available in the literature. Table 8 shows the problem specification and solution characteristics, where n, m, and k represent the number of nodes, number of backhaul nodes, and number of vehicles, respectively. It also includes the solutions and the ratio between the current solutions to the best known solution for the test cases.

As it can be seen from Table 8, the tuned MA obtains better solutions as compared to DB and SF and that it is comparable with the TV and Brandao algorithms on the test cases. In all test cases, MA outperforms DB and SF and in one of the four test cases MA performs better than TV and Brandao's algorithms.

6. CONCLUSION

In solving real world combinatorial problems, exact solution methods are not suitable since they need a long computational time. Metaheuristic approaches are complex, demanding for good parameterization. There seems lack of a systematic approach in the literature for tuning parameters of metaheuristic approaches. In this work, a memetic algorithm is presented for solving VRPB. This approach satisfactorily solves the test cases with the state-of-the-art solution quality and much greater efficiency as compared with the mathematical programming approach. The authors also presented a systematic study based on Design of Experiments in tuning the memetic algorithm to the highest performance, which is defined in terms of solution quality and computational time. Extensive experiments are performed and satisfactory results obtained. The developed MA method is applicable to other operational research problems. The systematic tuning approach should help the design and refinement of other metaheuristic optimization algorithms.

Our future work will include comparison of other calibrating techniques such as adaptive parameter setting with the presented approach, using the same approach to tune parameters of other metaheuristics like Tabu Search and simulated annealing.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their constructive and useful comments that significantly improved the quality and presentation of this paper.

REFERENCES

1. Bagchi, T.P. and Deb, K. (1996). Calibration of GA parameters: The design of experiments approach. *Computer Science and Informatic*, 26(4): 46-56.
2. Brandao, J. (2006). A new tabu search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research*, 173(2): 540-555.
3. Casco, O., Golden, L., and Wasil, A. (1988). Vehicle routing with back hauls: Models, algorithms, and case studies. In: B.L. Golden and A.A. Assad (Eds.), *Vehicle Routing: Methods and Studies*, Elsevier, Amsterdam, North Holland, pp. 127-147.
4. Cochran, W.G. and Cox, G.M. (1957). *Experimental Designs*, Wiley, New York.
5. Davis, L. (1993). *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York.
6. Dawkins, R. (1976). *The Selfish Gene*, Oxford University Press, New York.
7. Deif, I. and Bodin, L. (1984). Extension of the Clarke and Wright algorithm for solving the vehicle routing problem with backhauling. *Proceedings of Babson Conference on Software Uses in Transportation and Logistic Management*, Babson Park, MA, pp. 75-96.
8. Duhamel C. and Rousseau, J. (1997). A tabu search heuristic for the vehicle routing problem with backhauls and time windows. *Transportation Science*, 3(1): 49-59.
9. Goeschalckx, M. and Jacobs-Blecha, C. (1989). The vehicle routing problem with backhauls. *European Journal of Operational Research*, 42: 39-51.
10. Goetschalckx, M. and Jacobs-Blecha, C. (1993). The vehicle routing problem with backhauls: Properties and solution algorithms. Technical report, Georgia Institute of Technology.
11. Goldberg, D.E. and Lingle, R. (1985). *Alleles, Loci and the Traveling Salesman Problem*. Lawrence Eribaum Associates, Mahwah, NJ.
12. Grefenstette, J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1): 122-128.
13. Montgomery, D.C. (1997). *Design and Analysis of Experiments*, Wiley, New York.
14. Osman, I. and Wassan, N. (2002). A reactive tabu search for the vehicle routing problem with back-hauls. *Journal of Scheduling*, 5(4): 263-285.
15. Ruiz, R. and Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, 169(3): 781-800.
16. Ruiz, R., Maroto, C., and Alcaraz, J. (2006). Two new robust genetic algorithms for the flowshop scheduling problem. *Omega*, 34(5): 461-476.
17. Srinivas, M. and Patnaik, L. (1994). Adaptive probabilities of crossovers and mutations in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4): 656-667.
18. Syswerda, G. (1985). Uniform crossover in genetic algorithms. *Proceedings of the 3rd International Conference on Genetic Algorithms*, J.D. Schaffer (Ed.), San Francisco, CA, pp. 2-9.
19. Tavakkoli-Moghaddam, R., Saremi, A.R., and Ziaee, M. S. (2006). A memetic algorithm for a vehicle routing problem with backhauls. *Applied Mathematics and Computation*, 181(2): 1049-1060.
20. Toth, P. and Vigo, D. (1996). A heuristic algorithm for the vehicle routing problem with backhauls. In: L. Bianco, and P. Toth (Eds), *Advanced Methods in Transportation Analysis*, Springer-Verlag, Berlin-

Heidelberg.
21. Toth, P. and Vigo, D. (1997). An exact algorithm for the vehicle routing problem with backhauls. *Transportation Science*, 31(4): 372-385.
22. Toth, P. and Vigo, D. (1999). A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls. *European Journal of Operational Research*, 113(3): 528-543.
23. Yano, C.A., Chan, T.J., Richter, L., Murty, K.G., and McGettigan, D. (1987). Vehicle routing at quality stores. *Interfaces*, 17(2): 52-63.
24. Zhong, Y. and Cole, M.H. (2005). A vehicle routing problem with backhauls and time windows: A guided local search solution. *Transportation Research Part E: Logistics and Transportation Review*, 41(2): 131-144.

APPENDIX: MATHEMATICAL FORMULATION OF THE VEHICLE ROUTING PROBLEM WITH BACKHAULS (VRPB)

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

$$\min z = \sum_{i=0,...,n+m} \sum_{j=0,...,n+m} \sum_{k=1,...,M} C_{ijk} x_{ijk} \quad \text{(A-1)}$$

Subject to:

$$\sum_{j=0,...,n+m} \sum_{k=1,...,M} x_{ijk} = 1,\ i = 1,...,n+m,\quad i \neq j \quad \text{(A-2)}$$

$$\sum_{i=0,...,n+m} \sum_{k=1,...,M} x_{ijk} = 1,\ j = 1,...,n+m,\ j \neq i \quad \text{(A-3)}$$

$$\sum_{j=0,...,n+m} \sum_{k=1,...,M} x_{0jk} = M \quad \text{(A-4)}$$

$$\sum_{i=0,...,n+m} \sum_{k=1,...,M} x_{i0k} = M \quad \text{(A-5)}$$

$$\sum_{i=1,...,n} \sum_{j=1,...,n+m} d_i x_{ijk} \leq Q_k,\ k = 1,...,M,\quad j = 0,1,...,n+m \quad \text{(A-6)}$$

$$\sum_{i=n+1,...,n+m} \sum_{j=1,...,n+m} f_i x_{ijk} \leq Q_k,\ k = 1,...,M,\quad j = 0,1,...,n+m \quad \text{(A-7)}$$

$$\sum_{i=0,1,...,n+m} x_{ijk} - \sum_{l=0,1,...,n+m} x_{jlk} = 0,\ j = 1,...,n+m,\quad k = 1,...,M \quad \text{(A-8)}$$

$$\sum_{j=0,1,...,n+m} x_{ijk} - \sum_{l=0,1,...,n+m} x_{lik} = 0,\quad i = 0,1,...,n+m,\quad k = 1,...,M \quad \text{(A-9)}$$

$$\sum x_{ijk} \leq |s| - 1,\ s \subseteq \{2,3,...,n+m\} \quad \text{(A-10)}$$

$$\sum_{i=n+1,...,n+m} \sum_{j=1,...,n} \sum_{k=1,...,M} x_{ijk} = 0 \quad \text{(A-11)}$$

$$x_{iik} = 0,\ i = 0,1,...,n+m,\quad k = 1,...,M,\quad x_{ijk} = \{0,1\} \quad \text{(A-12)}$$

The objective function (A-1) tries to minimize the transportation cost, which is the summation of the cost of all arcs in a route multiplied by the vehicle multiplier. Constraints (A-2) and (A-3) specify that just one route passes from any node except the depot. Constraints (A-4) and (A-5) ensure that the number of vehicles leaving the depot must be the same as the number of vehicles returning to the depot. Constraints (A-6) and (A-7) stipulate that the vehicle load in terms of linehaul and backhaul must not exceed the vehicle capacity. Continuity of routes is enforced by constraints (A-8) and (A-9), i.e., each route must be served just by one vehicle. Constraint (A-10) prevents the model generating sub-tours. Finally, the precedence of linehaul nodes over backhaul nodes is ensured by constraint (A-11).