J Math Model Algor (2007) 6:591–613
DOI 10.1007/s10852-007-9069-2

A GA Based Heuristic for the Vehicle Routing Problem with Multiple Trips

S. Salhi · R. J. Petch

Received: 18 October 2006 / Accepted: 19 April 2007 /
Published online: 21 July 2007

Abstract A variant of the classical vehicle routing problem, where vehicles can be assigned to more than one route within a working time period, is investigated. A hybrid Genetic Algorithm, which uses a new non-binary chromosome representation and which is enhanced by a domain specific data structure, appropriate genetic operators and a scheme for chromosome evaluation, is proposed. Test problems from the literature are used to evaluate the performance of the proposed heuristic. Encouraging results are obtained.

Keywords Heuristic · Genetic algorithm · Bin-packing · Multiple trip

Mathematics Subject Classifications (2000) 90B06 · 90B35 · 90B40 · 90C59

1 Introduction

This paper addresses the vehicle routing problem (VRP) where a vehicle (driver) is not necessarily restricted to serve one route only. This is known as the vehicle routing problem with multiple trips (VRPM). In other words, the problem is characterized by vehicles and hence drivers working multiple routes or trips within a given time period. In practice, multiple trip scheduling is important since significant cost savings can be achieved if the number of vehicles and hence drivers are fully utilised. The VRPM can be used for both strategic and tactical planning. Because a reduced vehicle fleet size might be more desirable, a strategic VRPM objective accounts for

S. Salhi (✉)
Centre for Heuristic Optimisation, Kent Business School,
The University of Kent, Canterbury, UK
e-mail: s.salhi@kent.ac.uk

R. J. Petch
OR Group, London Underground, London, UK

both vehicle and scheduling costs. In this situation, there might be a trade off between higher scheduling costs and lower vehicle and driver associated costs, the latter being usually more significant. In practice, within a tactical distribution system, the fleet might be fixed to the existing setup and so the objective is reduced to minimizing scheduling costs.

This paper describes a hybrid Genetic Algorithm (GA) for the solution of this VRPM. In order to draw comparisons with the benchmarks achieved from the literature a similar objective is chosen, i.e. the minimization of the maximum overtime restriction for a prescribed number of vehicles.

The remaining of this section provides a brief review of the literature. This is followed by Section 2 that presents the main aspects of the algorithm, its motivation as well as the necessary notation. The main steps of the algorithm are described in Section 3 and the computational results are given in Section 4. We summarize our findings and highlight some research issues in the last section.

Although in practice multiple route assignment is common, there is a shortage of papers covering this feature. The first work to explicitly address multiple trips was made by Salhi [17] in the context of vehicle fleet mix. Limited to double trips, a matching scheme is used to allocate pairs of routes to vehicles within a refinement process. This problem was also tackled by Fleischmann [6] within a working paper where the author attempts to generate a solution using a one-phase algorithm, by integrating a greedy type heuristic with the need to assign routes to vehicles. Using a saving measure with respect to pairs of customers, the feasibility of the assignment of partially constructed routes to vehicles is assessed. The route assignment is achieved by using the bin-packing heuristic *best fit decreasing*, BFD (see Martello and Toth [9] for details). In this problem bins, items and corresponding weights are defined as vehicles, VRP routes and driver time required to service routes respectively. A two phase approach was proposed by Taillard et al. [21]. A set of VRP solutions is constructed from a population of routes generated using the Tabu Search (TS) heuristic of Rochat and Taillard [16] before bin-packing is used to allocate routes to vehicles. Golden et al. [8] adopted this approach to solve a similar VRPM using the minimax objective. This is a balancing problem which has a wide applicability. A constructive and improvement heuristic was proposed by Brandao and Mercer [1]. They tackled multiple trips as part of a more extensive problem involving time windows and vehicle fleet mix. Using real test data, results showed that their heuristic produced savings of 20% when compared to the manual schedule. To compare with the benchmark of Taillard et al. [21], Brandao and Mercer [2] modified their heuristic to solve the classical VRPM. Their approach is based on the nearest neighbour rule and the insertion criterion to assign customers to routes within vehicles. This process is repeated until all unrouted customers are inserted. The improvement phase attempts initially to remove overtime before reducing the routing cost within a TS framework using two types of trial moves namely insert and swap. Petch and Salhi [13] developed a multi-phase constructive heuristic which proved to be successful. The heuristic integrates the approach of Taillard et al. [21] and that of Brandao and Mercer [1] in as much as solution construction and improvement are undertaken in a VRP and VRPM environment respectively. An interesting feature of the methodology is the strategic use of existing heuristics to provide both diversity and intensification. Recently Olivera and Viera [10] put forward an interesting implementation of adaptive memory search for the VRPM. This is based on the same principle of Taillard et al. [21] with some enhancements incorporated. Initial VRP solutions are constructed by the sweep algorithm which are then enhanced by a tabu search. Several starting points are used to construct a family of solutions. The memory is then constructed by the top solutions up to a certain memory size. The routes are selected randomly from the memory and a bin packing procedure is adopted to pack the routes into vehicles while introducing some refinements based on reducing the driver overtime. The new routes are then fed back into the memory and those routes belonging to the poorest solutions are discarded. The whole process is executed for five runs. Excellent results are obtained.

2 Methodology

The power of GAs, as with other evolutionary methods, is that new solutions can be developed simultaneously and derived from several solutions, two in the case of a typical GA. For an overview of GA see Reeves [15] and Salhi [18], and for heuristic search in general see Salhi [19].

In this section we briefly describe the proposed GA approach, then we introduce the necessary notation including the objective function. We follow this by a subsection on defining our chromosome design and our step by step algorithm.

2.1 Approach

The classical GA usually requires a binary based chromosome representation. In practice it is not always possible to convert a solution to a binary representation, especially for order based problems like the VRPM. This has led to the development of new chromosome representations with unique problem specific genetic operators. A possible approach is the adaptation of the GA chromosome representation and operators developed for the TSP. Potvin [14] describes several GA implementations for the TSP but their adaptations to the VRPM are not obvious. This is primarily due to the difficulty in managing a chromosome where customers are partitioned to define the solution routes. Another approach is the partial use of the classical representation. Although the practical implementation of a binary chromosome representation is not possible in terms of codifying a unique solution, it is possible to use the classical representation in association with constructive heuristics. An example of this is a paper by Thangiah and Salhi [22] with respect to the VRP with multiple depots. Here a classical GA is used to generate a sequence of angles about the depots. This sequence, which describes a circle partition for each depot, is then used as a basis for customer clustering. A solution is produced by generating a single route for each cluster and a post-optimization is adopted to refine the solutions further. A similar structure could be used for the VRPM, with the use of bin-packing, but it can be restrictive, especially where a particular sector (partition) has a relatively high density of customers hence multiple routes share the same sector. To overcome this eventuality, we considered a variety of possible mapping techniques to generate single route clusters. The first was to approximate a route by a unique sector, with a given radius. A VRP solution could then be represented by a series of sectors, some of which may overlap. An associated solution evaluation would consist of clustering to the nearest sector boundary. Another consideration was to replace the sector by a single line approximation and then perform the same task. A weakness associated with these approaches is the need to prescribe the number of tours or routes generated within a solution as this value defines the chromosome length. Similar quality VRPM solutions can have different numbers of routes. We overcame this problem by developing a flexible non-binary chromosome structure based upon the circle partition concept used in Thangiah and Salhi [22]. Instead of fixing the chromosome length, the number of sectors can vary between solutions and more importantly each sector can generate more than one route. Subsection 2.3 describes how this is performed.

2.2 Notations and Objective Function

T:	The maximum regular travel time for a vehicle.
Q:	Vehicle carrying capacity.
NV:	The fleet size, which may be defined a priori or determined using the heuristic.
NC:	The number of customers to service.
θ_m:	The angle of the mth customer wrt the origin, $m = 1, \ldots, NC$.
n_p:	The number of chromosomes in the population (population size).
n_g:	The maximum number of generations used.
$\phi_{i,j}$	is the angle that defines the jth sector (sequence) within chromosome i.
k_i:	The number of clusters in (length of) chromosome i, $i = 1, \ldots, n_p$.
X_i	represents the ith chromosome of a given population, where $i = 1, \ldots, n_p$. (This is defined as a sequence of strictly increasing angles measured with respect to the depot, i.e., $X_i = \{\phi_{i,1}, \phi_{i,2}, \ldots, \phi_{i,k_i}\}$, where $\phi_{i,j} < \phi_{i,j+1}, \ \forall \ j = 1, \ldots, k_i - 1$).
Λ_g:	The population (i.e., set of chromosomes) at the gth generation.
P_g:	The corresponding set of solutions at the gth generation, $g = 1, \ldots, n_g$, derived from the set of chromosomes Λ_g.
$bf_{i,j}$:	The bit fitness of the jth cluster of the ith chromosome.
$NR_{i,j}$:	The number of routes in cluster j of chromosome i.
$(r_l)_{i,j}$:	The lth route of cluster j in chromosome i.
f:	The fitness function.
NC_{ij}:	The number of customers in sector $j (j = 1, \ldots, k_i)$ of chromosome i.
$\Omega_{i,j}$, $j = 1, \ldots, k_i$:	The jth cluster in chromosome i.
$q(j)$:	The demand of the jth customer, $j = 1, \ldots, NC$.
$NR(i)$:	The number of routes serviced by vehicle i, $i = 1, \ldots, NV$.
NR:	Number of routes within a solution.
S_{VRP}:	A VRP solution described as $S_{VRP} = \{r_1, \ldots, r_k, \ldots, r_{NR}\}$, where r_k is the kth route.
vs_i:	A schedule of routes assigned to vehicle i described as $vs_i = \{r_{i,1}, \ldots, r_{i,k}, \ldots, r_{i,NR(i)}\}$, where $\{r_{i,1}, \ldots, r_{i,k}, \ldots, r_{i,NR(i)}\} \subseteq S_{VRP}$.

S:	A VRPM solution described as follows$S = \{vs_1, \ldots, vs_k, \ldots, vs_{NV}\}$.
$VL(r_i)$:	The vehicle load required to service route i.
$DT(vs_i)$:	The total driver time required to service schedule i.
$DT(S)$:	The total driver time required to service solution S.
$\triangle(vs_i)$:	The total driver time infeasibility for schedule i.
$\triangle(S)$:	The total driver time infeasibility for solution S.
$\pi(S)$:	The maximum driver overtime for solution S.

The objective function is given as follows:

$$min\ \{\pi(S)\}$$

where $\pi(S)$, which refers to the maximum driver overtime for solution S, is defined as follows:

$$\pi(S) = max(0, max\{DT(vs_i) - T\}_{i=1,\ldots,NV}) \tag{1}$$

In the literature, the measure OTR_T is used to describe the overtime requirement. Computed using Eq. 2, OTR_T provides a ratio of the driver allocated the most work to T.

$$OTR_T(S) = [max\{DT(vs_i)\}_{i=1,\ldots,NV}]/T \tag{2}$$

The relationship to π is given as follows and provides an alternative to Eq. 1.

$$\pi(S) = \begin{cases} (OTR_T(S) - 1)T & \text{if } OTR_T(S) \geq 1, \\ 0 & \text{otherwise.} \end{cases}$$

Note that $OTR_T(S)$ does not define the total amount of overtime present within solution S but it provides the maximum (worst) overtime used in a given schedule. In some circumstances it may be more practical to refer to the scheduled overtime which is given by $\triangle(S)$, where $\triangle(S) = \sum_{i=1}^{NV} max(0, DT(vs_i) - T)$. As an alternative objective function, the scheduling cost for the VRPM is given in Eq. 3, where factors ϕ and p are unit costs for regular driver time and driver overtime penalty respectively.

$$C(S) = \sum_{i=1}^{NV} [\phi DT(vs_i) + p \triangle (vs_i)] \tag{3}$$

Note that when a solution S is allowed to use overtime, the legal overtime restriction needs not be violated (say a maximum of two hours per driver per day).

2.3 Chromosome Design

If a circle, with its center the depot D, is traced upon the xy plane so that it encompasses all customers, then it is apparent that each route or subset of routes from a VRPM solution can be described by a sector. This observation is the germ for the genetic approach we have adopted.

By partitioning the circle into sectors, a set of customer clusters is established by assigning customers to the sector which they occupy. Provided routes can be

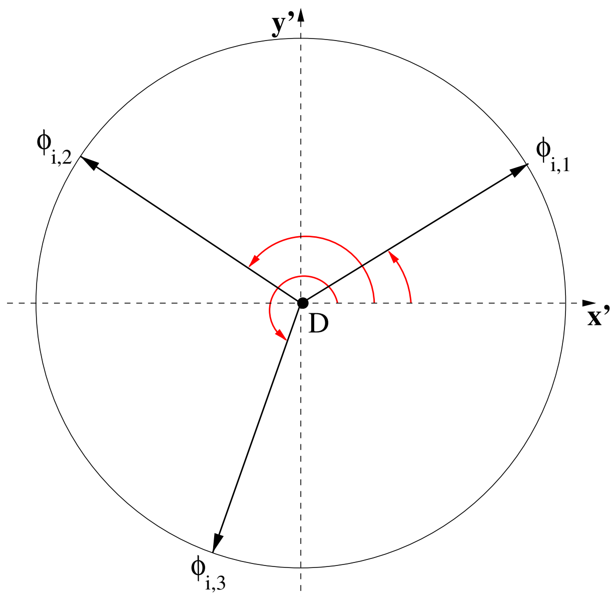

Fig. 1 Circle partition chromosome

generated from each sector, an organic encoding or chromosome is defined by describing the solutions underlying the circle partition. A partition can be described as a sequence of strictly increasing angles measured with respect to the depot, not necessarily 0. To simplify this process, for computational purposes, angles are measured in integer degrees. Furthermore, to provide a unique description, integers considered are those modulo 360. Figure 1 illustrates a chromosome representation within the plane, where $k_i = 3$.

Note that the coordinate system $(x^{'}, y^{'})$ has origin D such that $(x^{'}, y^{'}, D)$ is a translation of $(x, y, 0)$. The chromosome design, unlike the one used by [22], does not assume a sector sequence to start from a predefined position. For instance, the chromosome population is restricted as $\phi_{i,1} = 0\ \forall i$ would not allow route formation to cross the positive x' axis. The removal of such a constraint can affect the solution quality especially if customer density is large about the $x^{'}$ axis.

One may like to convert each angle measure to a binary form. This can be achieved by transforming the parameter set [0, 359] to the parameter set generated from the bit string of length $l \in \{1, 2, \ldots, 8\}$. One of the difficulties we encounter is to find a crossover operator which maintains a good schema. For this reason, we did not opt for such a binary representation. Our representation is described in Section 3.4.

2.4 The Algorithm

The main steps of our GA are given in Fig. 2. In this subsection we provide a brief description of the steps but the main explanations will be provided in the next section.

Step 1 consists of the generation of an initial population. Steps 2 and 3 are the main stages of the recursive process where new chromosomes are generated and evaluated. Step 4 is the algorithm stopping criterion related to the maximum number of generations, n_g. Although it is common to use an additional stopping criterion

- **step 1:** Generate an initial population of chromosomes Λ_1 and derive the corresponding solution set P_1 using the process as described in sections 3.1 and 3.2. Set generation counter $g = 2$.
- **step 2:** Produce a new population Λ_g using the following operators
(a) chromosome injection and cloning (see section 3.3),
(b) crossover and mutation type operators (see section 3.4).
- **step 3:** Derive the corresponding solution set P_g using the primary evaluation process as given in section 3.2) and which is enhanced by two data structures as described in subsection 3.4.5).
- **step 4:** If $g < n_g$ set $g = g + 1$ and goto step 2.
- **step 5:** Rank the obtained solutions from P_{n_g} in ascending order of their fitness f values (see Equation 5) and select the top $0.1n_p$ solutions excluding solutions with the same route configurations. Apply the improvement modules to this selected set of solutions to generate the new set P'_{n_g} as described in subsection 3.5 and choose the solution $S^* : f(S^*) = min\{f(S)\}_{S \in P'_{n_g}}$.

Fig. 2 The *GA* heuristic

based on the convergence of the current population, for simplicity the first criterion is adopted here given that new chromosomes are injected into the search to provide diversity. Step 5, which could be optional is a post optimizer. In this step, a subset of unique solutions $S \in P_{n_g}$ is improved using some refinement modules.

This new non-binary chromosome representation for the VRPM provides more flexibility than similar methods developed for other related routing problems. The heuristic can be classified either as a hybrid *GA*, since domain specific information is used to guide the search, or as an adaptation of Scatter Search, where new solutions are obtained through combinations of specific solution elements, see Glover [7] for details. One of our aims in this study is to see how such an evolutionary technique such as GA works for this type of routing problems.

3 Explanation of the Main Steps of the Algorithm

In the following subsections we provide some explanation for each of the steps.

3.1 Initial Population (Step 1)

A chromosome, which describes a circle partition, provides a basis for clustering and eventually route generation. A consequence of such a structure is the loss of uniqueness in terms of a corresponding solution. In other words, each sector cluster could generate many different routes and therefore each chromosome can represent many different solutions. This feature does present a weakness as it is difficult to use an existing solution or population of solutions obtained by another heuristic. Given a VRPM solution S, if we examine the routes and attempt to formulate a circle partition X which best describes S, then the solution evaluation $\Pi^{-1}(X) = S'$ is not necessarily equal to S. Nevertheless, in most cases the quality should be similar,

```
1: Set $i = 1$, compute $l_B = [\sum_{m=1}^{NC} q(m)/Q]$, and $u_B = [\frac{4}{3} l_B]$
2: Select uniformly a random number $k_i \in [l_B, u_B]$ and set $j = 1$.
3: Select uniformly a random number $\phi_{i,j} \in [0, 359]$.
4: If $j = 1$ set $X_i = \{\phi_{i,j}\}$ elseif $\phi_{i,j} \in X_{i,j}$ then goto step 3 else set $X_i = X_i \cup \{\phi_{i,j}\}$.
5: If $j = k_i$ set $i = i + 1$ and goto step 2 else set $j = j + 1$ and goto step 3.
6: Sort $X_i : \phi_{i,j} < \phi_{i,j+1} \forall j = 1, \ldots, k_i - 1$.
```

Fig. 3 Generation of the initial population

although this is not guaranteed. We chose to make use of a random initial population to avoid the problem of choosing the best partition to match a solution. This strategy also allows the heuristic to be independent and therefore enables it to produce, from such a starting point, a proposed solution to the VRPM. The algorithm developed to generate the initial chromosome population is given in Fig. 3. In this scheme two random parameter sets are generated; one to prescribe the number of sectors per chromosome k_i and the other to define each partition boundary $\phi_{i,j}$. The bounds used to prescribe k_i are based on the maximum possible number of routes which a solution could possibly require. This could be found by experience or through some runs of a simple heuristic such as the Saving Method of Clarke and Wright [4].

3.2 Primary Evaluation Process (Steps 1, 3)

This section describes the evaluation process associated with a given population. It provides not only an associated population of solutions but, more importantly, a set of fitness values which can be used to measure the quality of each chromosome. Note that this process is significantly different for subsequent generations due to the use of data structure which we developed (see Subsection 3.4.5 for details). Figure 4 outlines the evaluation process.

```
1: Construct clusters using the process as described in subsection 3.2.1.
2: Generate Cluster Routes using the saving heuristic.
3: Allocate Routes to Vehicles using the bin-packing transformation which is
   briefly described in subsection 3.2.1.
4: Compute the fitness value using the transformation of function values, as
   defined in subsection 3.2.2.
```

Fig. 4 Outline of the chromosome evaluation process

3.2.1 Clustering and Route Generation

Given a chromosome X_i a corresponding solution S_i is evaluated through a heuristic process known as cluster first-route second. The customers are clustered by assigning each one to the sector it occupies. For a given chromosome $X_i = \{\phi_{i,1}, \phi_{i,2}, \ldots, \phi_{i,k_i}\}$, the corresponding sectors are defined as follows:

$$(\phi_{i,j}, \phi_{i,j+1}]\, j = 1, \ldots, k_i - 1$$
$$[0, \phi_{i,1}] \cup (\phi_{i,k_i}, 359]\, j = k_i$$

This enables the assignment of boundary customers to be well defined, since the chromosome angle sequences do not specify which sector a boundary customer belongs to from the two choices available. Using this definition, the j^{th} cluster in chromosome i is defined by $\Omega_{i,j},\ j = 1, \ldots, k_i$. This is formed using the criterion given in expression (4), where m is a customer (i.e., $m \in \{1, \ldots, NC\}$).

$$\Omega_{i,j} = \begin{cases} m : \phi_{i,j-1} < \theta_m \leq \phi_{i,j} & \text{if } 2 \leq j \leq k_i, \\ m : 0 \leq \theta_m \leq \phi_{i,1} \text{ or } \phi_{i,k_i} < \theta_m \leq 359 & \text{otherwise.} \end{cases} \tag{4}$$

In each cluster the Saving Heuristic of Clarke and Wright [4] is used to solve a smaller VRP problem $(S_{VRP})_{i,j}$, where $(NC)_{i,j} = |\Omega_{i,j}|$. The union of routes generated from each cluster can then be used within the bin-packing heuristic, as described in Petch and Salhi [13], to provide the evaluated chromosome solution S_i.

Bin Packing Transformations – This bin packing heuristic is based on the best fit decreasing algorithm as used by Taillard et al. [21]. The idea is to assign routes to vehicles according to minimal residual driver time. Improved rescheduling is obtained by the reassignment of routes to vehicles. Refinement procedures that reallocate customers between routes are not part of this simple transformation but are an essential ingredient of the post optimiser which is used at the termination of the GA (see Subsection 3.5).

3.2.2 Fitness Value

We adopt a two-stage objective function as follows: a primary objective function is first used to find a solution which has minimum overtime requirement. If this solution involves no overtime, a secondary objective that improves the scheduling cost (driver time) is introduced. At this stage, only improving solutions which maintain feasibility, i.e. require no overtime, are considered. We address this problem by introducing a new function, given as follows, which integrates both elements of the two stage objective.

$$f_i = \xi DT(S_i) + \eta \pi(S_i) \tag{5}$$

Each element of this function, DT and π, is weighted with a priority given to the latter. We set $\xi = 1$ and $\eta = 100$, although the prescription is arbitrary provided that π is given more priority within the objective. These two values also serve as a way of approximately normalizing these two objective functions.

To convert the function value to a standard fitness value we make use of the following transformation:

$$F_i = [(\epsilon(M - f_i))^{\lambda}] \ \forall i$$

This fitness function enables the use of the roulette wheel selection process as described in Subsection 3.4.1. The constant M can be prescribed arbitrarily provided $M > Max\{f_i, i = 1, \ldots, n_p\}$. A typical method is to set M at each generation according to the value of the worst chromosome within the population. We chose to simplify the process and set the value of M from the initial population. To allow for the presence of solution deterioration in subsequent generations, we also allowed for a large deterioration in the expression of M as

$$M = 3 \times Max\{f_i, i = 1, \ldots, n_p\}$$

As $M - f^*$ can be too large, for computational reasons, we scaled down such values using ϵ as the scaling factor. In our experiments a value of 1/1000 was small enough to produce values of the order of 1–100. To discriminate between the chromosomes especially as their fitness values can be rather close, we amplified such values using a power factor, say λ. We set $\lambda = 2$ as this resolves our problem without making the F_i values excessively large.

3.3 Chromosome Injection and Cloning (Step 2a)

This section describes two simple mechanisms employed within the GA heuristic in order to maintain both solution quality and population diversity.

Other notations –

i^{gen}:	The current generation.
n^I:	The number of chromosomes to inject into the population.
n^{CI}:	The number of generations when the injection of the new chromosomes takes place.
n^c:	The number of chromosomes clones used.
n^{finish}:	The number of generations within the injection mechanism.
l_1, l_2:	The number of good chromosomes, and the number of good and mediocre (reasonable) chromosomes when both are put together respectively.
n^g, n^m, n^b:	The number of chromosomes generated from the good, mediocre and bad sections of the current population.
n^{pc}	is the number of partner choices (say $n^{pc} = 0.10n_p$).
$\Xi \subseteq \Lambda_g$	represents a set of partners (i.e., $\lvert\Xi\rvert = n^{pc}$).

Figure 5 provides a formulation of such mechanisms within the context of the current generation i^{gen}. Note that each mechanism requires the prescription of a set of parameters. The particular values chosen are given in the computational results section.

```
1: Define the following parameters: $n_p, n^I, n^c, n^{CI}, n^g, n^m$ and $n^b$. Rank the
   chromosomes $(X_i)_{i=1,\ldots,n_p}$ in descending order of their associated fitness value
   $(F_i)_{i=1,\ldots,n_p}$ (i.e., $F_1 \geq F_2, \ldots, \geq F_{n_p}$).

Chromosome Injection
2: If $(i^{gen}+1)$ is an integer multiple of $n^{CI}$ and $i^{gen} < n^{finish}$, then set $n^c = n^c - n^I$
   else goto step 4.
3: Generate set $(X_i^{'})_{i=1,\ldots,n^I}$ using the random population
   procedure as described in section 3.1.

Clone $n^c$ chromosomes
4:  [Best] Set $i = n^I + 1$ and $X_i^{'} = X_1$
5:  [Good] Set $i = i + 1$, generate randomly $\tau \in [2, l_1]$ and set $X_i^{'} = X_\tau$.
6:  If $i \leq n^g$ then goto step 5.
7:  [Reasonable] Set $i = i + 1$, generate $\tau \in [l_1 + 1, l_2]$ and set $X_i^{'} = X_\tau$.
8:  If $i \leq n^g + n^m$ then goto step 7.
9:  [Bad] Set $i = i + 1$, generate $\tau \in [l_1 + 1, n_p]$ and set $X_i^{'} = X_\tau$.
10: If $i \leq n^g + n^m + n^b$ then goto step 9.
```

Fig. 5 Injection and cloning at generation i^{gen}

Composition of a New Chromosome

Injection The first mechanism, which we call 'injection,' is primarily used to diversify the search process. Random chromosomes are injected into the new generation at regular generation intervals, n^{CI}, before the GA reaches a prescribed number of generations, $n^{finish} < n_g$. This stopping point was introduced to allow an element of standard convergence. Note that the time of injection and the number of injected chromosomes could be made dynamic dependent on the solution quality of the population at a given generation. For ease of implementation we adopted this simple form using regular injection.

Cloning The second mechanism, which we call 'cloning,' is the process of copying chromosomes from the current population, i^{gen}, into the new generation. Note that when injection is used, the number of chromosome clones n^c is reduced. Instead of selecting the best chromosomes, the cloning mechanism is based upon maintaining a variety of quality chromosomes. The chromosomes are ranked according to fitness and then partitioned into three groups $[1, l_1]$, $[l_1 + 1, l_2]$ and $[l_2 + 1, n_p]$. We then

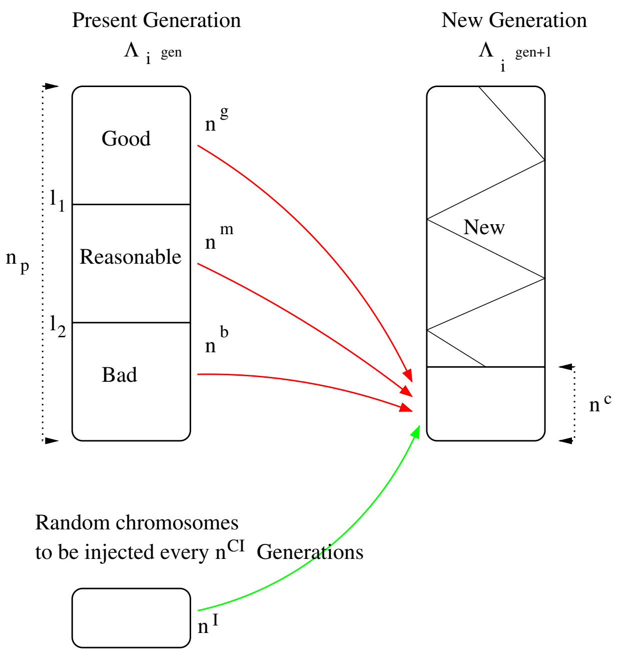

Fig. 6 New generation composition

select a proportion of chromosomes within each group, say n^g, n^m and n^b respectively. For simplicity we set $l_1 = (1/3)n_p$ and $l_2 = (2/3)n_p$. The chosen chromosomes are selected randomly as indicated in steps 5, 7 and 9 of Fig. 5. Note that the most superior chromosome is cloned systematically to retain the best solution throughout the search. Figure 6 illustrates the composition of the new chromosome set Λ_{i+1} indicating the parameters in relation to both chromosome cloning and injection.

The values of these proportions are based upon a limited preliminary experimentation and will be given in the computational result section.

3.4 Genetic Engineering (Step 2b)

This section describes the process developed to generate offspring or new chromosomes which form the bulk of the composition of a new generation. The process, referred to as 'Genetic Engineering,' consists of a crossover type operator, *Extraction*, a mutation based operator *Mutate* and a corresponding decision framework used to direct the operations. This framework enables the propagation operators to construct offspring which maintain the superior genetic code from within one of the parents.

3.4.1 Parent Selection (Parent Suitability)

Within a typical *GA* a chromosome pair, say X_i, $X_j \in \Lambda_g$ where Λ_g is the chromosome population at generation g, is needed in order to generate a new chromosome. Moreover, these parents are selected using a random process which is usually weighted according to fitness. A problem with using such a mechanism, in this context, is the formation of a chromosome pair without concern for genetic suitability. We address this problem by allowing a subset of partners, $\Xi \subseteq \Lambda_g$, to be selected so that a more attractive mate can be found for a given potential parent X_i. The definition of what is deemed attractive is given in Subsection 3.4.4. The key selection mechanism consists of the commonly used roulette wheel process.

The partner set Ξ is constructed in such a way that there is no chromosome duplication. Moreover, $|\Xi| = n^{pc}$, i.e. the number of partner choices is restricted to a subset of the population size. This value, which is considered to be a proportion of n_p, can be arbitrarily chosen but needs to be relatively small, say 10% of n_p.

3.4.2 Operator Extraction

The operator *Extraction* is the primary operator used to generate a new offspring. Given two chromosomes X_i and $X_j \in \Xi$ defined by the decision framework (see Subsection 3.4.4 for details), *Extraction* performs a variant of the two-point crossover. Unlike the usual mechanism, where genetic material is exchanged, *Extraction* does not exchange but replaces. In other words, a single offspring is produced instead of two. Furthermore, the replacement genetic material may be of different length and position within the mate chromosome X_j. *Extraction* is used to add a series of sectors, from a chosen partner X_j, into the chromosome X_i by overwriting the corresponding sectors. In other words, a sequence of angles is transplanted in the angle composition of chromosome X_i. Any angles which fall within the sequence range are removed.

Consider the following two examples where the new offspring $X^{'}$ is derived from parent X_i and its partner X_j. Note that each of the parents have their corresponding crossover points marked by [or] and that they are not necessarily at the same points.

Example 1 Let parent 1 be $X_i = \{[45, 70, 100], 150, 300\}$ and its partner $X_j = \{[20, 110], 160, 210, 290, 330\}$. The offspring is then $X^{'} = \{20, 110, 150, 300\}$. The first four sectors delimited by three angles of X_i namely 45,70 and 100 are (300-45);(45-70);(70-100) and (100-150) have been changed to three new sectors (300-20);(20-110) and (110-150) by the introduction of the two corresponding angles from X_j namely 20 and 110. Here, the sector (150,300) has not been affected. Note that the chromosome X_i has 5 clusters $(45, 70] \cup (70, 100] \cup (100, 150] \cup (150, 300] \cup (300, 45]$ whereas its offspring X' has only 4 clusters defined by $(20, 110] \cup (110, 150] \cup (150, 300] \cup (300, 20]$.

Example 2 Consider the same X_i and X_j as in example 1 but with a crossover point at angle 45 and 300 for the former and 110 and 290 for the latter. Let parent 1 be $X_i = \{45], 70, 100, 150, [300\}$ and its partner $X_j = \{20, 110], 160, 210, [290, 330\}$. For instance, in this case the removal of the sector $(300, 45]$ followed by the addition of sectors $(20, 110] \cup (290, 330]$ produces a new offspring $X^{'} = \{20, 110, 150, 290, 330\}$

3.4.3 Operator Mutate

Mutate is the mutation type operator used within the GA. As with the mutation operators developed for the binary representation, a mutation rate is used to test each bit for mutation. In our case, a bit string $\phi_{i,j}$ represents an angle and therefore a bit could technically mutate in such a way that $\phi^{'}_j \in [0, 359]$. We restrict this range

Fig. 7 *Mutate* implementation of chromosome i with k_i clusters

```
1: Set $\xi = 1/k_i$ and $j = 0$.
2: Select random number $\tau_1 \in [0, 1]$
3: If $j = k_i$ set $X' = \phi'_j : \phi'_j < \phi'_{j+1}$
   $\forall j = 1, \dots, k_i$ and stop; else set $j = j + 1$.
4: If $\tau_1 > \xi$ set $\phi'_j = \phi_{i,j}$ and goto step 2.
5: If $j = 1$ select random number $\tau_2 \in [0, \phi_{i,1})$, elseif
   $j = k_i$ select random number $\tau_2 \in (\phi_{i,k_{i-1}}, 359]$,
   else select random number $\tau_2 \in (\phi'_{j-1}, \phi'_{j+1}]$.
6: Set $\phi'_j = \tau_2$ and goto 2.
```

to the bounds of the local sectors. This was done so that the sequence of angles is not distorted and sector length for the chromosome considered for mutation is also maintained. Figure 7 outlines the implementation of the operator *Mutate*. Note that the mutation probability ξ is defined uniquely for the chromosome considered for mutation, dependent upon the number of bits or sectors.

3.4.4 Decision Framework (Chromosomes Selection)

The decision framework is the name given to the process of establishing how and if an operator, particularly *Extraction*, is implemented. Whereas traditional crossover type operators are initiated without domain specific information, the decision framework does allow such information into the picture. The motivation for developing this framework is the reluctance to use a random crossover mechanism in the traditional sense. Although such operators are extremely powerful, their use within this particular chromosome design is inappropriate. This is because vital sections of genetic code, i.e. sector sequences, can be corrupted once a random crossover is performed.

Bit Fitness The decision framework makes use of a bit measure. The concept of bit measure can then identify sectors or sequences of sectors that are both good and bad. This can be achieved in a number of ways and here we present a measure which relates to vehicle under utilization, see expression (6). The bit value representing a given sector, say j, in a given chromosome, say i, refers to the sum of the empty vehicle capacity for each route, say l, generated from that sector.

$$bf_{i,j} = \begin{cases} \sum_{l=1}^{(NR)_{i,j}} [Q - VL((r_l)_{i,j})] & \text{if } NC_{i,j} > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

This measure is important since it identifies sectors producing routes which are not efficient. If a route is not efficient, it could be that there is only a few customers within the sector and therefore a reasonable strategy would be to increase the sector size. Note that in the case where a sector produces more than one route another possible solution would be to reduce the sector size. For simplicity we chose to limit the choice to that of sector expansion. Another possible bit fitness measure could be related to the overtime restriction π. This can be defined by the driver allocated the most work.

Once the parent selection process had been carried out, we have the parent chromosome X_i and a subset of potential mate chromosomes Ξ, such that $X_i \neq X_j \in \Xi$. We use a sector replacement based on the bit fitness as defined in Eq. 6. Figure 8 outlines the decision framework for any such replacement with respect to the parent X_i and mate $X_j \in \Xi$. The framework uses probability acceptance p_e and p_m for the operators *Extraction* and *Mutate* respectively.

Initially, the worst sector in terms of vehicle load under utilization, i.e. maximum chromosome bit measure, is identified for replacement. Note that the prescription of

Application of the mutation type operator

1: [Mutate X_i]
Select a random variable $\tau_1 \in [0, 1]$. If $\tau_1 > p_e$ use the operator *mutate* to produce a single new chromosome $X^{'}$ and stop, else continue.

Application of the crossover type operator

2: [Identify the angle boundaries (l, u) of the worst case sector]
Select $s : bf_{i,s} = max\{bf_{i,l}\}_{l=1,\dots,k_i}$ and choose $\phi_u = \phi_{i,s}$.
If $s > 1$ set $\phi_l = \phi_{i,s-1}$, else set $\phi_l = \phi_{i,k_i}$.

3: [Identify the replacement sectors and compute the bit measure]
For each chromosome , say X_j select
min $m : \phi_{j,m} > \phi_u$ and set $m_u = m$, and max $m : \phi_{j,m} < \phi_l$ and set $m_l = m$.
If $m_u < m_l$ set $\alpha_j = \sum_{m=1}^{m_u} bf_{j,m} + \sum_{m=m_l}^{k_j} bf_{j,m}$
and $\beta_j = \phi_{j,m_u} + 360 - \phi_{j,m_l}$,
else set $\alpha_j = \sum_{m=m_l}^{m_u} bf_{j,m}$ and $\beta_j = \phi_{j,m_u} - \phi_{j,m_l}$.

4: [Select the best replacement sectors and extract]
Construct $B = \{j \text{ such that } \alpha_j < bf_{i,s}\}$, and select $j^* : \beta_{j^*} = Min_{j \in B}\{\beta_j\}$. If $|B| = 0$ select $j^* : \beta_{j^*} = Min\{\beta_j\}\ \forall j$. Perform the operator *Extraction* by replacing the sector defined by ϕ_{j^*,m_u} and ϕ_{j^*,m_l}

Application of the mutation type operator

5: [Mutate a child produced by crossover $X^{'}$]
Select a random variable $\tau_2 \in [0, 1]$. If $\tau_2 \leq p_m$ use the operator *Mutate* to produce a second new chromosome $X^{''}$ based on the child found by crossover $X^{'}$, else stop.

Fig. 8 Decision framework

the associated sector boundaries, ϕ_u and ϕ_l, takes into account the removal of the sector which crosses the positive $x^{'}$ axis.

In Fig. 8, steps 3, 4 and 5 determine which sequence of sectors from which chromosome mate X_j is a candidate for replacement. For each chromosome X_j, the associated sector sequence, such that the removed sector is expanded from both sides, is identified. The algorithm then simply selects the most improved replacement in terms of bit measure. Note that this process can yield the removal of more than one sector, since the new sector could violate local sectors.

3.4.5 Enhancing Efficiency using Two Data Structures

Once a new offspring chromosome has been constructed, using the operator *Extraction*, the next step is to derive the associated solution and to measure its fitness. This derivation makes use of two independent data structures to provide a more efficient implementation of the evaluation process outlined in Fig. 4.

(1) The first consists of data storage with respect to the routes developed within each sector of the chromosome. This information is extremely important since many sectors, and therefore routes generated, will remain unchanged from those of its parents. Since each sector cluster can be viewed as an independent VRP, we can adapt the storage data structure, described in Petch and Salhi [13], and in Osman and Salhi [11].
(2) The second data structure is used to assist the cluster process. When a series of sectors are transplanted into a chromosome, two new sectors are formed. Figure 9 illustrates that these sectors are local to the sector series insertion.

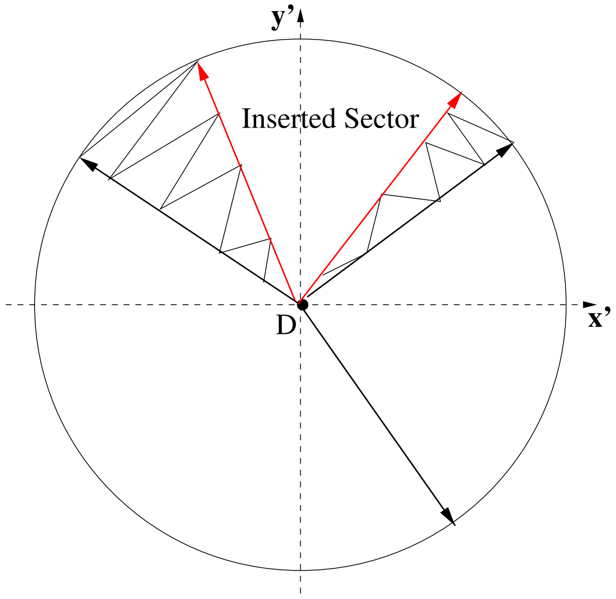

Fig. 9 Localized new sectors

A simple data structure is developed to examine whether all customers fall within a given sector $\Omega_{i,j}$ or not. The structure consists of a matrix $\Theta_{n,o}$ which holds lists of customers $m \in \{1, \ldots, NC\}$ which are characterized by their integer angle θ_m. If ρ_n is the cardinality of the list for angle n, $n = 0, \ldots, 359$, then if $\rho_n > 0$ then $o = 1, \ldots, \rho_n$. To access the customers with a sector defined by, say, $\Omega_{i,j} = \theta_m : \phi_{i,j} < \theta_m \leq \phi_{i,j+1}$ only values ranging from $m = \phi_{i,j} + 1$ to $\phi_{i,j+1}$ are considered. Since the process of clustering is undergone several times, the advantage of such a data structure becomes more and more significant. Using these data structures, route allocation to vehicles and fitness value computation are achieved using steps 3 and 4 of the primary evaluation process as outlined in Fig. 4.

3.5 Improvement Modules (Step 5)

We make use of a set of refinement modules developed by Petch and Salhi [13] to provide post optimisation. This is a composite heuristic that consists of five modules which are briefly given below.

Meiosis – divides a route to generate two new tours, similar to route split in Salhi and Rand [20].

VRP-partition – consists of a set of VRP refinement modules which are used to improve the particular assignment of routes for a given driver based on bin-packing ideas.

Donate, Exchange and Donate–exchange – these three refinements consist of a variety of customer reallocation transformations. These include the idea of inserting customers from one route to another, swapping customers between routes, among others. In addition, new routes can also be originally constructed by providing dummy or empty routes for each driver.

4 Computational Experience

The GA heuristic was tested using the adapted VRP data sets taken from Christofides et al. [3] namely CMT1 to CMT5, and CMT11 and CMT12 and those by Fisher [5] such as F71 and F134. The number of instances generated from each problem is given in Table 1 under the heading 'NC,' totalling 104 individual

Table 1 The number of infeasible solutions found by the proposed GA and the other benchmarks

BP	NC	# instances	GA	TLG	BM	PS	OV
$CMT1$	50	8	3	3	2	4	2
$CMT2$	75	14	6	3	2	2	1
$CMT3$	100	12	4	4	1	2	0
$CMT4$	150	16	8	4	2	3	1
$CMT5$	199	20	11	1	2	6	0
$CMT11$	120	10	5	1	1	2	0
$CMT12$	100	12	2	4	3	1	1
$F71$	71	6	3	3	2	2	1
$F134$	134	6	0	0	0	0	0
Overall		104	42	23	15	22	6

instances. The associated algorithms were coded in Fortran 90 and executed on a Ultra Enterprise 450 dual processor at 300 MHz. The performance of the proposed heuristic is assessed empirically by a comparison to the benchmarks found by Taillard et al. [21], Brandao and Mercer [2], Petch and Salhi [13], and Olivera and Viera [10]. These are referred as (TLG), (BM), (PS) and (OV) respectively. The following parameter prescriptions were set to generate computational results. These can be categorized according to their association as follows:

– *General type parameters:* $n_p = 100$, $n_g = 100$ and $n^{finish} = (0.8)n_g$
– *Cloning:* $n^c = (0.2)n_p$, $n^g = (0.5)n^c$, $n^m = (0.3)n^c$ and $n^b = (0.2)n^c$
– *Injection:* $n^I = (0.5)n^c$, $n^{CI} = (0.1)n_g$
– *Genetic Engineering:* $p_e = 0.8$, $p_m = 0.2$ and $n^{pc} = (0.2)n_p$

The values chosen were based on a limited experimentation and therefore more suitable choices may be possible.

Figure 10 illustrates a typical convergence behaviour observed for the GA. The graph, which is based on the base problem $CMT5$ where $NC = 200$, measures the best solution, in terms of OTR, found thus far at each generation $1, \ldots, n_g$. This is accompanied by the average OTR found within the given generation population set Λ. Note that despite the use of chromosome injection, the average solution quality has a similar path as the most superior member of the population. This behaviour is evidence of population convergence. It can be observed that although we used 100 generations, in most cases a good solution is found before 50 generations have elapsed and hence this information could be used to cut down on the number of

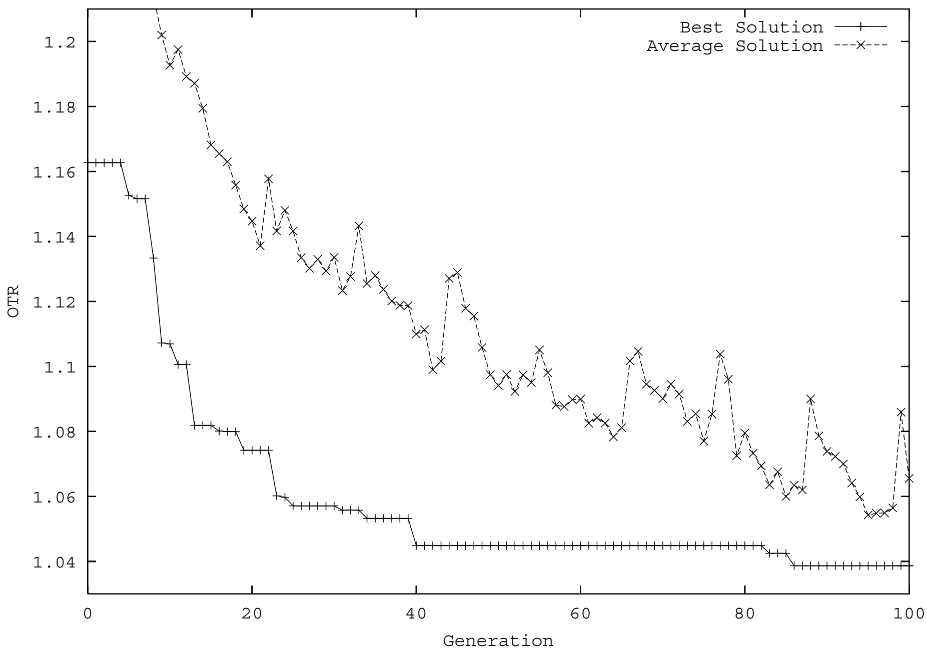

Fig. 10 Typical behaviour of the GA (use of $CMT5$, where $T = 678$)

Table 2 Average CPU time (s) of the proposed GA and the other benchmarks

BP	NC	# instances	GA	TLG	BM	PS	OV^+
$CMT1$	50	8	15.89	300	150	108	16
$CMT2$	75	14	30.01	420	300	330	29
$CMT3$	100	12	70.02	1,440	600	828	27
$CMT4$	150	16	206.42	3,060	1,500	984	68
$CMT5$	199	20	483.56	3,960	3,750	2,454	125
$CMT11$	120	10	1,131.61	2,700	1,500	2,430	28
$CMT12$	100	12	45.31	1,380	600	120	27
$F71$	71	6	93.31	1,560	150	258	13
$F134$	134	6	584.20	4,500	4,800	810	31

$^+$Refers to the time of the best run only.

generations if necessary. Such a reduction in computing time could obviously be used to improve further the results by experimenting with other parameter values or simply by applying more intensive post optimisation.

When comparing against the benchmarks we do so with respect to overtime restriction only as this is the measure which is recorded in the literature. The average overtime restriction π for the GA is 1.52% compared to 0.86, 0.53, 0.72 and 0.09% for benchmarks TLG, BM, PS and OV respectively. Moreover, if we consider the number of instances tested where no overtime was required, the GA produces 61% compared to 78% from TLG, 86% from BM, 78% from PS and 94% from OV. The number of infeasible solutions reported by each of the five methods is given in Table 1.

Though in general, the GA does not perform as well as the benchmarks, it does produce several equal quality results in certain base problems. In addition, when the GA produces a solution requiring overtime the corresponding overtime restriction is usually relatively small, approximately less than 4% of T. This small margin could easily be reversed if more intensive refinements are used. Note also that in this paper we also report, see Table 3 in the Appendix, the detailed results for each instance whether the OTR value is smaller or larger than 1. In our view, such information is useful to other researchers who like to use these results as benchmarks since in the published literature only those instances whose OTR value is larger than 1 are given but not the others (the previous authors considered solutions with an OTR value less than 1 to be feasible and hence did not report them).

The results with respect to CPU time, measured in seconds, are given in Table 2.

The results show that the GA is relatively quicker than the first three benchmarks except the OV heuristic which seems to be much faster but the OV reported time is based on the best run only and not over all the five runs (in other words such times need to be approximated by a factor of 5). However, such a comparison may not be strictly informative due to the different computer power of the machines used. It is also worth noting that the efficiency of the implementation of this GA is partially due to the important data structure designed in this work for generating offsprings. The use of the post-optimisation also contributed to improving the solutions though not significantly, see Petch [12] for more details. At the time of the revision of this paper we were informed about the recent work of Olivera and Viera [10] in which newer results were found. We revised this paper accordingly.

5 Conclusion and Possible Further Research

In this work we examine the use of a population-based heuristic such as GA to solve the VRP with multiple trips. In our view this is the first attempt that uses an evolutionary type method contrarily of the other which are mainly either tabu search-based or multi-level type heuristics. Using the test problems, we found that the GA generates a significant number of reasonable quality solutions in a short time period when compared to most benchmarks from the literature. Although, on average, the present implementation of our GA does not outperform the best benchmarks, the proposed heuristic produced reasonably good results in most instances. In addition, this GA has strong potential for future development, both in terms of the creation of new chromosomes and the coding structure.

In this study, we introduced a number of parameters which, in some ways, make the results sensitive to changes. However, it will be interesting to understand the practical meaning of each of these parameters and assign appropriate values accordingly. In other words, the values of some of the parameters would be set a priory whereas the values for the remaining ones could be obtained dynamically. It will also be interesting to assess the effect of each of the modules and especially the impact of the post optimisation by conducting additional experiments. In the proposed GA implementation, we inject the new chromosomes randomly and in a periodic manner. An investigation to determine dynamically the time for using the injection process as well as the number, type (random, high quality, etc.) and frequency could be worth pursuing. It may also be useful to explore other ways of clustering rather than limiting the search to the sectors as proposed here. This can be critical as the routes may not necessarily belong to well defined and restricted sectors as attempted in this study.

Acknowledgements The authors would like to thank the referees for their constructive comments that improved both the content as well as the presentation of the paper.

Appendix

Detailed Computational Results

The following series of Tables provides a comprehensive set of results of the OTR for the GA heuristic. The Table columns are described as follows:

BP:	Base problem tested taken from Christofides et al. [3] and Fisher [5].
T:	The maximum regular travel time for a vehicle.
NC:	The number of customers.
NV:	The fleet size required.
C:	Scheduling cost including overtime.
OTR:	The ratio of the driver allocated the most work to T.
GA, TLG, PS and OV	refer to the present GA heuristic, the heuristic by Taillard et al. [21], the heuristic by Brandao and Mercer [2], the heuristic by Petch and Salhi [13] and the heuristic by Olivera and Viera [10] respectively.

Table 3 The OTR Results of the GA and other benchmarks

BP	T	NV	C	GA	TLG	BM	PS	OV
$CMT1$	551	1	546.28	0.991	≤ 1	≤ 1	≤ 1	≤ 1
$(NC = 50)$	275	2	552.76	1.003	≤ 1	≤ 1	1.004	≤ 1
	184	3	586.32	1.030	1.115	1.041	1.026	1.024
	138	4	632.54	1.056	1.027	1.027	1.085	1.027
	577	1	547.14	0.948	≤ 1	≤ 1	≤ 1	≤ 1
	289	2	549.42	0.955	≤ 1	≤ 1	≤ 1	≤ 1
	192	3	560.26	0.989	1.050	≤ 1	≤ 1	≤ 1
	144	4	566.86	0.996	≤ 1	≤ 1	1.031	≤ 1
$CMT2$	877	1	869.06	0.991	≤ 1	≤ 1	≤ 1	≤ 1
$(NC = 75)$	439	2	865.48	0.987	≤ 1	≤ 1	≤ 1	≤ 1
	292	3	877.07	1.003	≤ 1	≤ 1	≤ 1	≤ 1
	219	4	856.77	0.991	≤ 1	≤ 1	≤ 1	≤ 1
	175	5	904.98	1.022	≤ 1	≤ 1	≤ 1	≤ 1
	146	6	1,021.29	1.068	1.032	1.031	1.019	≤ 1
	125	7	1,056.34	1.102	1.073	1.088	1.064	1.009
	919	1	869.73	0.946	≤ 1	≤ 1	≤ 1	≤ 1
	459	2	881.50	0.963	≤ 1	≤ 1	≤ 1	≤ 1
	306	3	869.11	0.968	≤ 1	≤ 1	≤ 1	≤ 1
	230	4	880.90	0.972	≤ 1	≤ 1	≤ 1	≤ 1
	184	5	883.29	0.997	≤ 1	≤ 1	≤ 1	≤ 1
	153	6	914.25	1.024	≤ 1	≤ 1	≤ 1	≤ 1
	131	7	966.43	1.036	1.023	≤ 1	≤ 1	≤ 1
$CMT3$	867	1	845.33	0.975	≤ 1	≤ 1	≤ 1	≤ 1
$(NC = 100)$	434	2	850.65	0.987	≤ 1	≤ 1	≤ 1	≤ 1
	289	3	872.73	1.006	≤ 1	≤ 1	≤ 1	≤ 1
	217	4	874.80	1.011	≤ 1	≤ 1	≤ 1	≤ 1
	173	5	964.79	1.056	1.062	≤ 1	1.52	≤ 1
	145	6	927.87	1.050	1.032	1.003	1.001	≤ 1
	909	1	845.33	0.930	≤ 1	≤ 1	≤ 1	≤ 1
	454	2	872.11	0.962	≤ 1	≤ 1	≤ 1	≤ 1
	303	3	869.48	0.966	≤ 1	≤ 1	≤ 1	≤ 1
	227	4	878.00	0.982	≤ 1	≤ 1	≤ 1	≤ 1
	182	5	901.30	0.999	1.010	≤ 1	≤ 1	≤ 1
	151	6	861.76	0.998	1.012	≤ 1	≤ 1	≤ 1
$CMT4$	1,080	1	1,064.06	0.985	≤ 1	≤ 1	≤ 1	≤ 1
$(NC = 150)$	540	2	1,065.86	0.987	≤ 1	≤ 1	≤ 1	≤ 1
	360	3	1,103.50	1.009	≤ 1	≤ 1	≤ 1	≤ 1
	270	4	1,149.41	1.031	≤ 1	≤ 1	≤ 1	≤ 1
	216	5	1,216.00	1.054	≤ 1	≤ 1	≤ 1	≤ 1
	180	6	1,179.41	1.036	≤ 1	≤ 1	≤ 1	≤ 1
	154	7	1,312.08	1.090	1.033	1.071	1.072	1.002
	134	8	1,284.01	1.100	1.075	1.031	1.058	≤ 1
	1,131	1	1,088.93	0.963	≤ 1	≤ 1	≤ 1	≤ 1
	566	2	1,070.50	0.954	≤ 1	≤ 1	≤ 1	≤ 1
	377	3	1,077.24	0.969	≤ 1	≤ 1	≤ 1	≤ 1
	283	4	1,119.05	0.995	≤ 1	≤ 1	≤ 1	≤ 1
	226	5	1,085.38	0.972	≤ 1	≤ 1	≤ 1	≤ 1
	189	6	1,112.03	0.985	≤ 1	≤ 1	≤ 1	≤ 1
	162	7	1,211.54	1.032	1.010	≤ 1	1.005	≤ 1

Table 3 (continued)

BP	T	NV	C	GA	TLG	BM	PS	OV
	141	8	1,332.05	1.076	1.029	≤ 1	≤ 1	≤ 1
$CMT5$	1,356	1	1,347.34	0.994	≤ 1	≤ 1	≤ 1	≤ 1
$(NC = 199)$	678	2	1,346.63	0.994	≤ 1	≤ 1	≤ 1	≤ 1
	452	3	1,418.62	1.006	≤ 1	≤ 1	≤ 1	≤ 1
	339	4	1,451.94	1.029	≤ 1	≤ 1	≤ 1	≤ 1
	271	5	1,460.75	1.030	≤ 1	≤ 1	1.007	≤ 1
	226	6	1,476.94	1.036	≤ 1	≤ 1	≤ 1	≤ 1
	194	7	1,427.52	1.023	≤ 1	≤ 1	1.008	≤ 1
	170	8	1,465.44	1.038	≤ 1	≤ 1	1.015	≤ 1
	151	9	1,477.54	1.047	≤ 1	1.056	1.024	≤ 1
	136	10	1,602.40	1.076	1.024	1.051	1.064	≤ 1
	1, 421	1	1,340.44	0.943	≤ 1	≤ 1	≤ 1	≤ 1
	710	2	1,399.65	0.987	≤ 1	≤ 1	≤ 1	≤ 1
	474	3	1,409.37	0.993	≤ 1	≤ 1	≤ 1	≤ 1
	355	4	1,397.60	0.987	≤ 1	≤ 1	≤ 1	≤ 1
	284	5	1,411.19	0.996	≤ 1	≤ 1	≤ 1	≤ 1
	237	6	1,377.07	0.981	≤ 1	≤ 1	≤ 1	≤ 1
	203	7	1,394.73	0.999	≤ 1	≤ 1	≤ 1	≤ 1
	178	8	1,416.27	1.001	≤ 1	≤ 1	≤ 1	≤ 1
	158	9	1,440.64	1.013	≤ 1	≤ 1	≤ 1	≤ 1
	142	10	1,506.95	1.034	≤ 1	≤ 1	1.018	≤ 1
$CMT11$	1,094	1	1,088.26	0.995	≤ 1	≤ 1	≤ 1	≤ 1
$(NC = 120)$	547	2	1,139.70	1.014	≤ 1	≤ 1	≤ 1	≤ 1
	365	3	1,117.96	1.009	≤ 1	≤ 1	≤ 1	≤ 1
	274	4	1,206.41	1.052	1.020	1.011	1.052	≤ 1
	219	5	1,670.45	1.202	≤ 1	≤ 1	1.037	≤ 1
	1,146	1	1,088.26	0.950	≤ 1	≤ 1	≤ 1	≤ 1
	573	2	1,110.10	0.975	≤ 1	≤ 1	≤ 1	≤ 1
	382	3	1,088.56	0.973	≤ 1	≤ 1	≤ 1	≤ 1
	287	4	1,141.62	1.001	≤ 1	≤ 1	≤ 1	≤ 1
	229	5	1,092.95	0.992	≤ 1	≤ 1	≤ 1	≤ 1
$CMT12$	861	1	819.97	0.952	≤ 1	≤ 1	≤ 1	≤ 1
$(NC = 100)$	430	2	821.33	0.956	≤ 1	≤ 1	≤ 1	≤ 1
	287	3	826.98	0.977	≤ 1	≤ 1	≤ 1	≤ 1
	215	4	824.57	0.991	≤ 1	1.012	≤ 1	≤ 1
	172	5	869.45	1.015	1.050	1.036	≤ 1	≤ 1
	143	6	898.88	1.029	1.064	1.072	1.029	1.014
	902	1	819.97	0.909	≤ 1	≤ 1	≤ 1	≤ 1
	451	2	829.54	0.936	≤ 1	≤ 1	≤ 1	≤ 1
	301	3	851.16	0.956	≤ 1	≤ 1	≤ 1	≤ 1
	225	4	821.53	0.958	≤ 1	≤ 1	≤ 1	≤ 1
	180	5	833.85	0.984	1.003	≤ 1	≤ 1	≤ 1
	150	6	855.36	0.982	1.014	≤ 1	≤ 1	≤ 1
$F71$	254	1	254.22	1.000	≤ 1	≤ 1	≤ 1	≤ 1
$(NC = 71)$	127	2	266.13	1.020	1.031	1.011	1.020	≤ 1
	85	3	266.85	1.020	1.075	1.011	1.020	1.020
	266	1	254.07	0.955	≤ 1	≤ 1	≤ 1	≤ 1
	133	2	254.07	0.979	≤ 1	≤ 1	≤ 1	≤ 1
	89	3	256.53	0.985	1.027	≤ 1	≤ 1	≤ 1

Table 3 (continued)

BP	T	NV	C	GA	TLG	BM	PS	OV
$F134$	1, 221	1	1,190.21	0.975	≤ 1	≤ 1	≤ 1	≤ 1
($NC = 134$)	611	2	1,194.24	0.990	≤ 1	≤ 1	≤ 1	≤ 1
	407	3	1,199.86	0.990	≤ 1	≤ 1	≤ 1	≤ 1
	1, 279	1	1,183.00	0.925	≤ 1	≤ 1	≤ 1	≤ 1
	640	2	1,199.64	0.940	≤ 1	≤ 1	≤ 1	≤ 1
	426	3	1,215.43	0.962	≤ 1	≤ 1	≤ 1	≤ 1

References

1. Brandao, J., Mercer, A.: A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. Eur. J. Oper. Res. **100**, 180–191 (1997)
2. Brandao, J., Mercer, A.: The multi-trip vehicle routing problem. J. Oper. Res. Soc. **49**, 799–805 (1998)
3. Christofides, N., Mingozzi, A., Toth, P.: The vehicle routing problem. In: Combinatorial Optimization, pp. 313–338. Wiley, Chichester (1979)
4. Clarke, G., Wright, J.: Scheduling of vehicles for a central depot to a number of delivery points. Oper. Res. **12**, 568–581 (1964)
5. Fisher, M.: Optimal solution of vehicle routing problems using minimum k-trees. Oper. Res. **42**(4), 626–646 (1994)
6. Fleischmann, B.: The vehicle routing problem with multiple use of vehicles. Working paper, Fachbereich Wirschaftswissenschaften, Universitat Hamburg (1990)
7. Glover, F.: A template for scatter search and path relinking. In: Artificial Evolution. Lecture Notes in Computer Science, pp. 13–54. Springer, Berlin Heidelberg New York (1998)
8. Golden, B., Laporte, G., Taillard, E.: An adaptive memory heuristic for a class of vehicle routing problems with minmax objective. Comput. Oper. Res. **24**, 445–452 (1997)
9. Martello, S., Toth, P.: Knapsack Problems. Wiley, Chichester (1990)
10. Olivera, A., Viera, O.: Adaptive memory progarmming for the vehicle routing problem with multiple trips. Comput. Oper. Res. **34**, 28–47 (2007)
11. Osman, I., Salhi, S.: Local search strategies for the mix fleet routing problem. In: Rayward-Smith, V.J., et al. (eds.) Modern Heuristic Search Methods, chap. 8, pp. 131–154. Wiley, Chichester (1996)
12. Petch, R.: Constructive and population based heuristics for the vehicle routing problem with multiple trips. Ph.D. thesis. University of Birmingham, UK (2001)
13. Petch, R., Salhi, S.: A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. Discrete Appl. Math. **133**, 69–92 (2004)
14. Potvin, J.: Genetic algorithms for the traveling salesman problem. Ann. Oper. Res. **63**, 339–370 (1996)
15. Reeves, C. (ed.): Modern Heuristic Techniques for Combinatorial Problems. Blackwell, Oxford, UK (1995)
16. Rochat, Y., Taillard, E.: Probabilistic diversification and intensification in local search for vehicle routing. Heuristics **1**, 147–167 (1995)
17. Salhi, S.: The integration of routing into the location-allocation and vehicle composition problems. Ph.D. thesis, University of Lancaster, pp. 198–208 (1987)
18. Salhi, S.: Heuristic search methods. In: Marcoulides, G. (ed.) Modern Methods for Business Research, chap. 6. Lawrence Erlbaum, London (1998)
19. Salhi, S.: Heuristic search methods: the science of tomorrow. In: Salhi, S. (ed.) Keynote Papers at OR48, pp. 39–58. Operational Research Society, Bath (2006)
20. Salhi, S., Rand, G.: Incorporating vehicle routing into the vehicle fleet composition problem. Eur. J. Oper. Res. **66**, 313–330 (1993)
21. Taillard, E., Laporte, G., Gendreau, M.: Vehicle routing with multiple use of vehicles. J. Oper. Res. Soc. **47**, 1065–1070 (1996)
22. Thangiah, S., Salhi, S.: Genetic clustering: an dadptive heuristic for the multi depot vehicle routing problem. Appl. Artif. Intell. **15**, 361–383 (2001)