Expert Systems with Applications 39 (2012) 2865–2871

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Genetic algorithm with iterated local search for solving a location-routing problem

Houda Derbel[a,*], Bassem Jarboui[a], Saïd Hanafi[b], Habib Chabchoub[a]

[a] FSEGS, Route de l'aéroport km 4, Sfax 3018, Tunisia
[b] LAMIH, Université de Valenciennes et du Hainaut-Cambrésis, France

ARTICLE INFO

Keywords:
Location
Routing
Genetic algorithm
Local search

ABSTRACT

This paper deals with a location routing problem with multiple capacitated depots and one uncapacitated vehicle per depot. We seek for new methods to make location and routing decisions simultaneously and efficiently. For that purpose, we describe a genetic algorithm (GA) combined with an iterative local search (ILS). The main idea behind our hybridization is to improve the solutions generated by the GA using a ILS to intensify the search space. Numerical experiments show that our hybrid algorithm improves, for all instances, the best known solutions previously obtained by the tabu search heuristic.

1. Introduction

A Location Routing Problem (LRP) can be viewed as a combination of two difficult and inter-dependent decision-making problems, namely Location and Routing. Roughly speaking, given a set of potential facilities/depots and a set of customers, the LRP aims at efficiently servicing customers (planning routes) from a well-chosen sub-set of facilities (locating depots). The objective is to minimize the total cost of opening depots/facilities and outing to serve all customers.

When tackling such a problem, it is necessary to consider simultaneously Location and Routing given that ignoring routes when locating facilities obviously increases the cost of the distribution system and leads to sub-optimal solutions (Salhi, 1989). Generally speaking, in a complete supply chain management, Location and

Routing appear in fact to be two interrelated components of major concerns in many real-life applications. For instance, locating regional blood banks to serve hospitals (Or, 1979), or setting up newspaper/mail delivery systems (Jacobsen, 1980), or distributing goods/parcels/humanitary-care/network distribution chain (Perl, 1985, Wasner, 2004, Billionet, 2005), to cite a few, deal with location and routing problems simultaneously. Solving jointly location and routing is therefore a challenging task of both practical and fundamental interest for logistics managers and decision makers.

In its most general form, the LRP seeks to minimize the total cost by simultanously selecting a subset of potential depots and making routes by allocating customers to different depots while

* Corresponding author.

E-mail addresses: derbelhouda@yahoo.fr (H. Derbel), bassem_jarboui@yahoo.fr (B. Jarboui), Said.Hanafi@univ-valenciennes.fr (S. Hanafi), habib.chabchoub@fsegs.rnu.tn (H. Chabchoub).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.08.146

satisfying the following constraints: (i) each customer's demand is satisfied, (ii) each customer is visited by exactly one route, and is assigned to exactly one depot without exceeding vehicle or facility capacities, (iii) the length of a route and the number of vehicles are defined with a prespecified limit, (iv) each route begins and ends at the same depot. In addition, depending on the considered application context, several LRP variants with different characteristics are commonly studied in the literature. In fact, each time a new application setting is considered, a new LRP variant with additional constraints such as capacities on depots or on vehicles, length constraint on the vehicle routes, time windows or uncertainty of some data. Although, in many practical situations, a combined location-routing model is the most appropriate model for all LRP variants, location and routing are often solved separately. Only few recent works have considered this issue mainly due to the difficulty of both problems. In Laporte (1988), Laporte et al. give different types of formulations, solution algorithms and computational results on integrated LRP. A synthesis of LRP studies with a hierarchical taxonomy and classification scheme as well as different solution methodologies can also be found in Min, Jayaraman, and Srivastava (1998). More recently, a comprehensive state-of-the-art study on the huge amount of work on LRP is given in Nagy and Salhi (2007). Later in this paper, we provide the reader with a comprehensive summary of the most relevant approaches used so far in the literature to solve the integrated LRP.

In this paper, we consider a variant of LRP with capacitated depots and a single uncapacitated vehicle dispatched at each open depot such that the demand of each customer must be satisfied, each customer is served by exactly one vehicle and each route begins and ends at the same depot. To the best of our knowledge, this version is studied only by Albreda-Sambola, Diaz, and Fernandez (2005).

We propose a hybrid method, denoted (GA&ILS), that combines genetic algorithm (GA) and iterated local search (ILS) and integrates the location and routing decisions of LRP. Our GA&ILS approach coordinates between the two levels of decision making by seeking a facility configuration and the routing that corresponds to it. We conducted computational experiments to evaluate the proposed method in comparison with the result mentioned in Albreda-Sambola et al. (2005). More specifically, our approach maintains a set of solutions (a population of individuals) which is improved in many steps of the hybrid algorithm. In fact, we defined different mutation/crossover operators allowing us to generate a new generation of solutions at each phase. Depending on the quality/fitness of the new population, we then apply an ILS to a subset of generated solutions using four neighborhood structures. Generally speaking, diversification is mainly obtained through the GA while intensification is obtained through the ILS. However, the key point is that both mutation/crossover operators and ILS neighborhood structures are carefully designed in order to operate jointly on the location and the routing levels. To validate our hybrid approach, we experiment it using the same instances as Albreda-Sambola et al. (2005). For all instances, we obtain new best solutions. Additionally, our approach is adequate from a computational time point of view. In fact, as pointed in Albreda-Sambola et al. (2005), while most of the test instances cannot be solved optimally with CPLEX, our approaches outputs solutions within some milliseconds to at most some seconds for all instances.

The remainder of this paper is organized as follows. In Section 2, we recall the LRP variant considered in this paper and give a summary of LRP related works. In Section 3, we give an overview of our hybrid GA&ILS optimization framework. In Section 4, we present our computational results. Finally, in Section 5, we conclude the paper and discuss some open issues.

2. Problem definition and related work

The LRP can be defined as follows. Consider $I = \{1, \ldots, n\}$ the set of customers and $J = \{1, \ldots, m\}$ the set of potential depots. Each depot $j \in J$ is characterized by a limited capacity b_j and a fixed cost f_j of establishment. Each customer $i \in I$ has a non-negative demand d_i which is known in advance and should be satisfied. Moreover, each depot is associated with a single uncapacitated vehicle. Let c_{ij}, i, $j \in I \cup J$ be the traveling cost between i and j. The LRP consists of opening a subset of depots and elaborate vehicle tours to visit the set of customer in order to minimize the total cost of location and delivery.

To solve a LRP, there is a need to solve a facility location problem (FLP) and a vehicle routing problem (VRP). The master problem (location) is directly inter-related with the subproblem (routing). If each customer is directly connected to the facility then the LRP will be a special case of a classic location problem. On the other hand, if the depots are located, the LRP is reduced to a VRP. Both subproblems, namely FLP and VRP are fielded among NP-hard problems (Cornuejols, Fisher, & Nemheuser, 1977; Karp, 1972), thus the LRP is also NP-hard. The LRP described in this paper can be seen as an extension of the capacitated VRP which is recently solved by an iterated variable neighborhood descent algorithm in Chen, Huang, and Dong (2010) and hyrbid metaheuristics in Lin, Lee, Ying, and Lee (2009).

Different exact methods were developed in the literature for solving the LRP. The first exact methods is a branch and bound algorithm given in Laporte and Norbert (1981) for the single facility LRP without tour length restrictions. For the LRP with capacitated vehicles and depots (CLRP) and a fixed number of vehicles, a branch and cut method is described in Laporte, Norbert, and Arpin (1986). That method is mainly based on the relaxation of the subtour elimination constraints (each vehicle tour must contain a depot) and the chain barring constraint (an arc connecting two depots is not allowed). Another branch and bound algorithm is also given in Laporte, Norbert, and Taillefer (1988) where an appropriate graph representation is used to transform the CLRP into an equivalent constrained assignment problem. More recently, a lower bound for the CLRP is proposed in Barreto (2004) by solving a linear programming relaxation. Interested readers are referred to Stowers and Palekar (1993) and Zografos and Samara (1989) for more related works on exact methods for the LRP.

Since the LRP is NP-hard, the exact method requires exponential CPU time in the size of the input problem, most of the research has focused on heuristic methods. A simulated annealing based decomposition approach for the multidepot LRP with capacity on both depots and vehicles (CLRP) is given in Wu, Low, and Bai (2002) and Yu, Lin, Lee, and Ting (2010). In Prins, Prodhon, and Wolfer-Calvo (2006) and Duhamel, Lacomme, Prins, and Prodhon (2009), CLRP with a homogenous and unlimited fleet is solved using mainly a greedy randomized adaptive procedure (GRASP). Many other heuristic approaches exist in the literature, the reader can for instance refer to Prins, Prodhon, Ruiz, Soriano, and Calvo (2007), Madsen (1983), Berman, Jaillet, and Simchi-Levi (1995), Min et al. (1998), Wu et al. (2002), Jacobsen and Madsen (1980), Or and Pierskalla (1979), Srikar (1983) and Salhi and Rand (1989) for a more exhaustive state-of-the-art survey on solving LRP and its variants.

In the following sections, we discuss the proposed hybrid heuristic for solving our LRP.

3. Hybrid approach

As mentioned earlier, the LRP can be viewed as an integration of two NP-hard optimization problems where each separate problem is by its own difficult to solve. In this work, we propose a hybridization of GA and ILS to jointly solve location and routing. Our hybrid approach GA&ILS is motivated by the fact that in the case of a large search space: (i) a GA may fail to converge to a global optimum since it explores too many different sub-parts of the search space and takes a long execution time, whereas (ii) a local search method may fall into a local optimum quickly. Therefore, we use ILS to refine the GA search through successive iterations and maximize the chance of convergence to an optimal solution through using various search spaces.

A high level overview of our hybrid GA&ILS algorithm is depicted in Algorithm 1 below. Before going into the technical details of our hybrid approach, let us briefly recall the basic concepts behind GA.[1] Generally speaking, the choice of GA as a building block to tackle our LRP problem is motivated by the large number of studies adopting GA in routing-like problems, e.g., (Baker & Ayechew, 2003; Berger & Barkaoui, 2004; Gen & Syarif, 2005; Ho, Ho, Ji, & Lau, 2008). GA is in fact a population-based metaheuristic which has been proved very powerful to solve many large scale problems (Holland, 1975). It is based on the natural mechanism applied to a population of individuals. Those individuals are following genetic rules to give rise to new offsprings. Those that cannot survive vanish and disappear. Similarly, GA starts with a population of solutions, then it finds better ones by applying *ge-*

netic operations over the individuals of each iteration. One can in fact think of GA as a local search applied on attributes of a set of solutions rather than attributes of a single solution. Therefore, the key issues when designing a GA is to carefully define the genetic

[1] Non expert readers can for instance refer to Holland (1975) for more details about GA.

operations over the population, namely, *selection*, *mutation*, *crossover* and *replacement*.

In Algorithm 1, we incorporate ILS into this GA general scheme. This allows us to take advantage of ILS features in order to improve the population generated by the GA and thus to complement the genetic search. Alternating GA and ILS in our hybrid approach reflects the interaction between the global search characteristic we can gain by using a GA and the ability of ILS to find local optima. In fact, while GA may produce bad (resp. good) individuals representing good (resp. bad) search spaces, ILS enables to introduce fairness when exploring different regions of the search space. More specifically, the ILS used in our approach is based on four different neighborhood structures as will be described in details later in sub Section 3.2. In the next subsection, we shall focus on the GA used in our approach. We first describe our solution representation. Thereafter, we discuss our genetic operators and the parameters used therein.

Algorithm 1: **Hybrid GA&ILS high level overview**

```
1 $s \leftarrow 40; \mathbb{P}_a \leftarrow 0.7; \mathbb{P}_p \leftarrow 0.9; \delta \leftarrow 0.1;$ /* Global parameters */ ;
2 The GA algorithm high level code :
3     $\text{FEVAL}_{\text{best}} \leftarrow \infty;$
4     Initialize $Pop$ /* First generation of individuals*/;
5     Initialize the best fitness $\text{FEVAL}_{\text{best}}$ ;
6     while termination critrerion is not satisfied do
```

```
7         SELECT individuals $x_1$ and $x_2$ from $Pop$ following prob.
8         Apply CROSSOVER to $x_1$ and $x_2$;
9         Let $x_{\text{new}}$ be the new obtained child;
10        Apply MUTATION to $x_{\text{new}}$ ;
11        if FEVAL($x_{new}$) $< (1+\delta) \cdot$ FEVAL$_{best}$ then
12            Apply ITERATIVE LOCAL SEARCH with $x_{\text{new}}$ as an i
13        Update the best fitness FEVAL$_{\text{best}}$;
14        Apply REPLACEMENT to $Pop$;
```

3.1. Solution representation and genetic operators

3.1.1. Solution representation

A fundamental point when designing a GA is the representation of each individual in the population of solutions referred as chromosomes. In our approach, a solution x (i.e., an individual, a chromosome) is represented using two vectors $A(x)$ and $P(x)$. The first vector represents an assignment configuration that gives the set of open depots and the customers affected to each of them (location level). The second vector is a permutation vector fixing the rank of a customer on a given route (routing level).

More formally, consider an individual x and its representation $A(x) = \{a_1, a_2, \ldots, a_n\}$ and $P(x) = \{p_1, p_2, \ldots, p_n\}$. For every $\ell \in \{1, \ldots, n\}$, if $a_\ell = j \in J$ then this means that customer $\ell$is assigned to depot j. The tours can then be deduced from vector P as following. For every $\ell \in \{1, \ldots, n\}$, $p_\ell \in I$ corresponds to a client. Given two indices

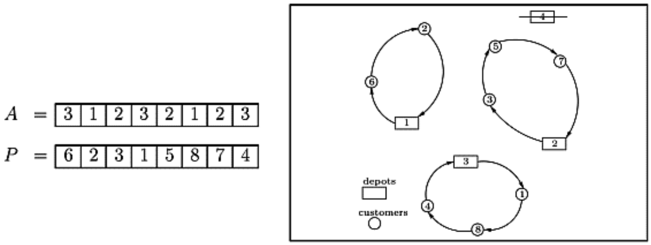


Fig. 1. An example of LRP solution representation.

ℓ and ℓ' such that $\ell < \ell'$, if customers p_ℓ and $p_{\ell'}$ are affected to the same depot, that is $a_\ell = a_{\ell'} = j \in J$, then p_ℓ is served before $p_{\ell'}$ in the route corresponding to opened depot j.

To illustrate the representation of a solution, we consider an example with $n = 8$ and $m = 3$, see Fig. 1 where on the right side we give LRP solution and on the left side its corresponding representation vectors A and P. Then, since $a_1 = a_4 = a_8 = 3$, we can deduce that customers 1, 4 and 8 are assigned to depot 3. The route serving these customers is obtained from the permutation vector P. More precisely, customer 1 is followed by customers 8 then customer 4 in vector P meaning that the route serving those customers from depot 3 starts with customer 1, then 8, and ends with customer 4. Similarly, customer 2 then customer 6 are served by depot 1 whereas customers 3, 5 and 7 are assigned to depot 2.

As we will see in next sections, this solution representation is both simple and accurate in order to capture simultaneously the routing and the location levels of our problem. This is in fact a key feature that allows us to define efficient GA operators that act jointly on customers assignment to depots and the corresponding routes.

Notice also that in line 4 of Algorithm 1, the first generation of individuals is generated randomly. More precisely, an initial individual x is initially constructed by assigning each client to one depot at random in vector A, then by generation a random permutation describing the tours for vector P.

3.1.2. Selection

During the operations of a GA process, a new generation is born by selecting some particular parents and some children. The problem is how to select good chromosomes from the population. There are many existing methods in the literature, for instance roulette wheel selection, Boltzman selection, rank selection and some others. Selecting a parent randomly or relatively to its fitness seems to be not beneficial because the difficulty of distinguishing between two chromosomes especially in minimization problems. In our approach, we follow the same selection mechanism than in Reeves (1995, 1995). More precisely, a parent is selected according to the following probability distribution:

$$\mathbb{P}([k]) = \frac{2k}{M(M+1)}$$

where $[k]$ is the kth chromosome ordered in descending order of its objective value and M is the population size. This means that the chromosome with the best objective value has a higher probability to be selected to ensure genetic operations.

3.1.3. Crossover

After deciding the encoding representation of a solution, two parents are selected according to the selection method already described in Section 3.1.2 to creta a new offspring. The crossover operation tries to swap parts of two parents in the population to generate new offsprings. The crossover is made in hope that an offspring will inherit good parts of old chromosome. There are many ways to do crossover, more the crossover is specific to

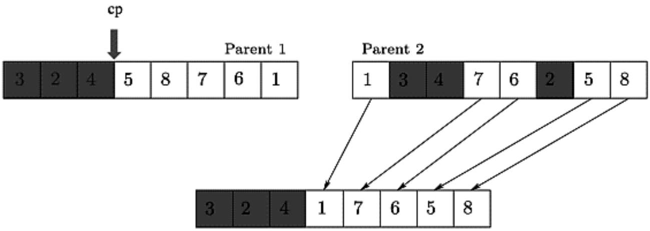


Fig. 2. Crossover operation for the permutation vector.

the problem more the GA is performant. Our crossover operator is applied both on the assignment vector *A* and the permutation (routing) vector *P*. For the first vector *A*, we use a one point crossover by randomly choosing a crossover point *cp* uniformly selected from 1 to *n*. An offspring is then obtained by appending the beginning (resp. the end) of the first parent to the end (resp. the beginning) of the second parent. As for the second vector, we follow the crossover procedure of Murata and Ishibuchi (1994). First we select one crossover point and the permutation is copied from the first parent till this point, then the second parent is scanned and if the customer is not yet in the offspring it is added. More precisely, having two parents x_1 and x_2, we construct an offspring x as follows. We randomly choose a crossover point. The first part of vector *P* representing offspring x is the same than parent x_1. The second part of $P(x)$ is completed by taking, in the same order, the element of $P(x_2)$ that were not already included in the first part.

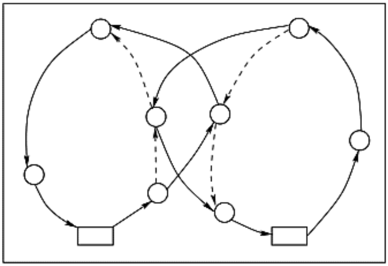
For clarity, let us consider an example with 3 depots and 8 customers and a crossover point $cp = 3$ as depicted by Fig. 2.

3.1.4. Mutation operators

When designing a GA, mutation appears to be among the most important operation to escape local optima since it preserves the diversification in the population. The mutation step takes place immediately after the crossover is performed. In our approach, we use two types of mutation, one for each vector of a chromosome. First, we randomly modify the assignment vector by removing one customer from a depot to another one belonging to the set of potential depots. This gives rise to a possibility of opening a new depot which, in turn, allows us to explore new solutions and thus to diversify the search. The second mutation applied for the permutation vector P is to insert a customer, selected at random, at a new position, chosen at random too. More precisely, given a chromosome x, we choose at random a customer p_ℓ from $P(x)$ and a new position ℓ' for this customer. The new chromosome is then obtained by swapping customer p_ℓ in position ℓ' in $P(x)$ and shifting every other customer in the range $\{\min\{\ell,\ell'\},\ldots,\max\{\ell,\ell'\}\}$ to the left or the to right depending on whether $\ell < \ell'$ or $\ell > \ell'$.

3.1.5. *Fitness function*

Comparing the quality of two individuals plays an important role in any GA. In our work, the evaluation function $\textsc{Feval}$ of an individual x is defined as the sum of two components $\textsc{Cost}(x)$ and $\textsc{Penality}(x)$, i.e., $\textsc{Feval}(x) = \textsc{Cost}(x) + \textsc{Penality}(x)$. The first component $\textsc{Cost}(x)$ is the total cost of the LRP solution represented by individual x. The second component $\textsc{Penality}(x)$ is a penalty on the violation of the capacity constraints. This penalty is used to get as close as possible to the feasible space. In fact, $\textsc{Penality}(x)$ is a well defined metric function driving the solution represented by individual x to the boundaries of the feasible space. More precisely, $\textsc{Penality}(x)$ is given by the following formula:


(a) initial solution x

$$\text{PENALITY}(x) = \sum\nolimits_{j \in J} \alpha \cdot \max\{0, D_j(x) - b_j\}$$

where:

- $D_j(x)$ is the total demand of customers assigned to depot j in solution x,
- b_j is the maximal capacity of the depot j,
- and α is a constant parameter that reflects the degree of the penalty or equivalently to what extent an unfeasible solution should be considered in the search.

3.1.6. Replacement

The replacement phase is the last phase in a GA (line 14 in Algorithm 1). It consists of maintaining the population size constant. Many existing methods are available in the literature to choose which individual must be removed from the population such as random replacement or weak parent replacement (Sivanandam & Deepa, 2008). In our algorithm, once a new offspring is created (using the GA operators and the ILS procedure), it is compared with the worst individual in the population. Then the best one is simply kept inside the population.

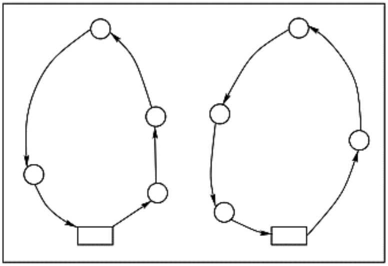
3.2. The ILS method

This section describes the ILS heuristic we have developed for our hybrid LRP approach. As stated previously, the ILS is used to mainly improve the genetic operations and to better guide the search process (line 12 of Algorithm 1). The main steps of a ILS, namely, *local search* and *perturbation* were first given in Glover (2002). The general framework of a ILS is depicted in Algorithm 2 below. Let x_0 be the initial solution for the ILS process. A local search using some neighborhoods structures is applied to x_0 to find a better solution $\hat{x}$. Once a better solution is found, perturbation is performed on $\hat{x}$ to obtain a new solution x. After that, a local search is applied again to x to obtain a new local optimum $\tilde{x}$. The solution $\tilde{x}$ replaces $\hat{x}$ only if $\tilde{x}$ is better than $\hat{x}$ is verified and the ILS process continues until a stopping criterion is met.

Algorithm 2: ILS general stucture

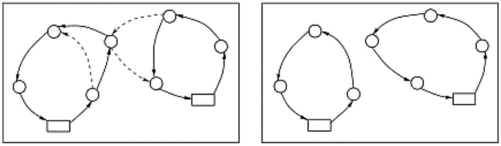
```
1 Let $x_0$ be an initial solution;
2 $\hat{x} \leftarrow$ LOCAL SEARCH($x_0$);
3 repeat
4     $x \leftarrow$ PERTURBATION($\hat{x}$);
5     $\tilde{x} \leftarrow$ LOCAL SEARCH($x$);
6     if FEVAL($\hat{x}$) $<$ FEVAL($\tilde{x}$) then $\hat{x} \leftarrow \tilde{x}$;
7 until Termination condition is met ;
```

In the sequel, we detail the local search followed by the perturbation mechanism used in our ILS process.



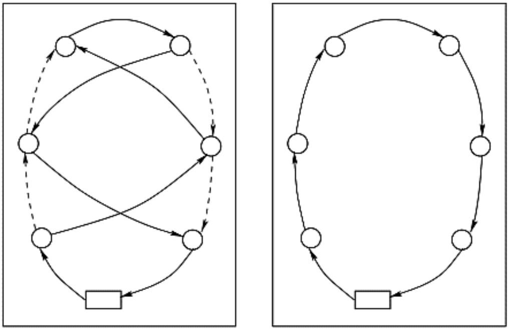
(b) neighboring solution in $\mathcal{N}1(x)$

Fig. 3. Neighborhood $\mathcal{N}1$.



(a) initial solution x (b) neighboring solution in $\mathcal{N}2(x)$

Fig. 4. Neighborhood $\mathcal{N}2$.



(a) initial solution x (b) neighboring solution in $\mathcal{N}3(x)$

Fig. 5. Neighborhood $\mathcal{N}3$.

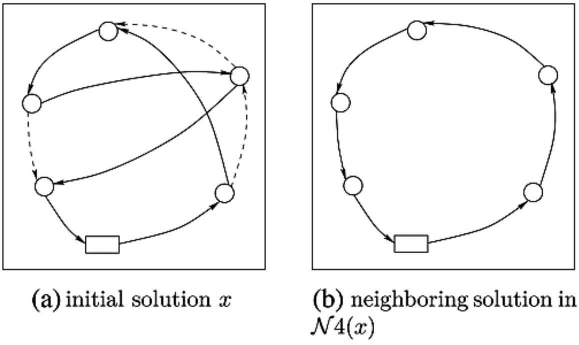


Fig. 6. Neighborhood $\mathcal{N}4$.

3.2.1. Local search

A local search is a classical optimization method that consists of generating a local optimal solution by exploring the neighborhood of a given initial solution. One of the main ingredients when designing a local search is the choice of the neighborhood structure. To make it simple, a neighborhood is constructed by modifying some components of a given solution to create a new neighboring solution. In other words, the neighborhood defines the way to reach a new solution within few modifications. Identifying an effective neighborhood structure allowing us to efficiently move from one solution to another is extremely important. However, a local optimal solution in a neighborhood structure is not necessary a local optimal in another neighborhood structure. For this reason, the use of several neighborhood structures help guiding the local search to converge less rapidly to a local optimum. In addition, for our simultaneous routing and location problem, it is of special interest to consider different neighborhoods, each one reflecting a different characteristic of the problem. In the ILS used to complement the GA in our hybrid approach, we use four neighborhoods including insertion move and swap move, namely $\mathcal{N}1, \mathcal{N}2, \mathcal{N}3$ and $\mathcal{N}4$, which are described in the following.

Neighborhood $\mathcal{N}1$ and $\mathcal{N}2$ are performed between two different routes with an attempt to improve the solution. More precisely, as shown in Fig. 3, for neighborhood $\mathcal{N}1$, the swap move is performed by randomly selecting two customers assigned to two different depots and interchanging them. The corresponding routes are of course modified but neither the number nor the order of customers in those initial routes is changed. As for neighborhood $\mathcal{N}2$, the insertion move is carried out by choosing one customer from one route and inserting it into another route (see Fig. 4).

Neighborhoods $\mathcal{N}3$ and $\mathcal{N}4$ allows us to focus on new solutions by applying swap and insertion moves inside the same route. More precisely, for neighborhood $\mathcal{N}3$, we consider one route and swap the positions of two customers inside that route as shown in Fig. 5. As for the neighborhood $\mathcal{N}4$, we insert a customer between two other customers in the same route as shown in Fig. 6.

Having this neighborhood structures in mind, the local search process follows the general scheme of Algorithm 3 below. We first start by applying a classical local search according to neighborhood structure $\mathcal{N}1$. More precisely, given an initial solution x, we sequentially improve this solution by choosing the first incumbent neighbor improving x with respect to $\mathcal{N}1$ and so on until computing a solution x_1 that cannot be improved no more. We obtain a local optimum x_1 that is given as the starting point for the new local search with respect to $\mathcal{N}2$. We proceed in the same way with neighborhood $\mathcal{N}3$ and $\mathcal{N}4$. This process is repeated until a local optimum of the four structures of neighborhood is reached.

Algorithm 3: Local search using neighborhoods $\mathcal{N}1$, $\mathcal{N}2$, $\mathcal{N}3$ and $\mathcal{N}4$

```
input : $x$ : an initial solution
1 $x_1 \leftarrow$ First Improvement on $x$ using neighborhood $\mathcal{N}1$ ;
2 $x_2 \leftarrow$ First Improvement on $x_1$ using neighborhood $\mathcal{N}2$;
3 $x_3 \leftarrow$ First Improvement on $x_2$ using neighborhood $\mathcal{N}3$;
4 $x_4 \leftarrow$ First Improvement on $x_3$ using neighborhood $\mathcal{N}4$;
5 if Feval($x_4$) $<$ Feval($x_1$) then
6     $x \leftarrow x_4$;
7     Go to Line 1;
```

3.2.2. Perturbation criterion

In the above described neighborhoods, our local moves concern only open depots. In fact, we can reasonably argue that a local move to a closed depot can deteriorate the evaluation function due to the high fixed cost of opening a depot. However, we are also interested in opening a closed depot since this gives us the opportunity to deal with a different type of solutions. This is precisely the aim of the perturbation mechanism of Algorithm 2 (line 4). In fact, we modify the current local optima $\hat{x}$ computed by the local

Table 1
Percent deviations to the lower bound and CPU time. The best gap values are shown in bold.

Instances		p						m						g					
		ILS		GA&ILS		T.S		ILS		GA&ILS		T.S		ILS		GA&ILS		T.S	
		%gap	Time	%gap	Time	%gap	Time	%gap	Time	%gap	Time	%gap	Time	%gap	Time	%gap	Time	%gap	Time
*S*1	*a*	**0**	0.01	**0**	0.02	0.35	5.37	**0**	0.02	0	0.03	0.6	5.45	0.01	0.01	**0**	0.02	0.06	5.43
	b	**0**	0.00	**0**	0.02	0.04	4.89	**0**	0.01	**0**	0.02	0.35	5.10	**0**	0.01	**0**	0.02	0.04	5.04
	c	0.00	0.00	**0**	0.02	0.24	3.97	**0**	0.01	0	0.03	0.04	4.19	0	0.01	**0**	0.02	0.33	4.23
*S*2	*a*	**10.42**	0.41	**10.42**	0.40	10.79	22.7	7.66	0.83	**7.65**	0.92	7.97	24.94	**2.85**	2.28	**2.85**	0.64	2.94	25.91
	b	**11.58**	0.10	**11.58**	0.21	12.30	23.4	**8.88**	0.32	**8.88**	1.05	9.29	23.09	**4.02**	0.96	**4.02**	0.36	4.40	23.88
	c	**13.63**	0.12	**13.63**	0.22	13.69	17.47	**10.34**	0.90	**10.34**	1.00	10.44	17.42	**4.03**	0.2	**4.03**	0.55	4.08	17.37
*S*3	*a*	10.88	5.47	**10.28**	6.07	11.31	70.18	**6.28**	7.00	6.48	4.25	7.41	71.73	2.27	5.48	**2.26**	8.44	2.50	70.32
	b	11.35	4.95	**11.20**	4.44	12.33	62.78	**8.69**	3.66	8.78	7.62	10.30	64.88	3.21	2.36	**3.16**	2.55	3.79	65.53
	c	**13.26**	2.99	**13.26**	2.64	14.20	45.76	9.75	5.51	**9.55**	5.75	11.38	45.39	**3.75**	4.18	**3.75**	2.19	4.33	45.22
*M*2	*a*	21.75	1.90	**21.64**	4.71	22.01	90.55	14.15	2.40	**14.10**	5.28	14.90	91.14	6.08	3.18	**6.01**	7.30	6.13	90.72
	b	**27.37**	0.71	27.38	2.03	28.63	58.89	**17.17**	4.90	17.30	3.74	17.52	63.19	8.39	5.04	**8.30**	6.26	8.83	55.6
	c	29.60	2.85	**29.32**	2.43	30.71	40.30	**20.94**	1.67	21.02	3.67	22.98	38.13	**9.94**	4.54	**9.94**	4.87	11.54	35.60
*M*3	*a*	20.34	7.63	**19.43**	14.61	20.63	240.62	**12.79**	12.98	12.98	15.35	14.16	241.96	**6.63**	17.71	6.75	17.23	6.44	239.20
	b	**24.75**	4.82	24.82	18.07	26.65	139.08	16.97	6.58	**16.23**	9.33	17.45	163.97	**6.97**	11.42	8.18	15.23	8.77	133.81
	c	**26.98**	5.44	27.10	5.93	28.62	97.96	19.67	6.69										

19.65	10.87	19.36	96.21	**9.11**	10.17	9.16	15.04	9.89	98.01

%gap: the average deviation of solution value relative to the lower bound.
Time: running time over ten instances.

search in Algorithm 2 in order to obtain an intermediate solution x' and avoid stopping at this local optimum. Each time a perturbation is made throughout the ILS procedure, a new solution is created. We perturb the solution in the following way: we select a open depot at random and we remove the customers already assigned to another depot either opened or closed in an arbitrary way. This kind of move generates a new kind of solutions by opening/closing some depots.

4. Computational results

The proposed heuristic described in the previous section was coded in C language. The experiments are performed on a desktop PC Intel Pentium IV, windows XP, 3.2 GHz processor and 1 GB memory.

4.1. Test instances

We tested our algorithm on benchmark instances generated by Albreda-Sambola et al. (2005). There are five sets: *S*1, *S*2 and *S*3 with 5 facilities and respectively 10, 20 and 30 customers whereas *M*2 and *M*3 represent instances with 10 facilities and 20 or 30 customers. Each instance *i* is named "$pcr_1r_2u_i$" where "*pc*" indicates the set of instances we are dealing with and hence the number of depots and customers as previously described. The number r_1 corresponds to the ratio between the total demand of the customers and the total capacity depots which gives the number of open facilities a priori in the optimal solution. Three values of r_1 are considered which are 0.3, 0.5 and 0.7, and are represented by "a", "b" and "c", respectively. The value of $r_2 u$ is proportional to the fixed cost of opening a depot. The factor r_2 ranges in $\{1.5, 3, 9\}$ and is represented by $\{p, m, g\}$ whereas $u = r(u = d)$ if it is uniformly distributed in $[-1.1, 1.1]$ (respectively in $[-1.01, 1.01]$). For more information about the considered instances, see Albreda-Sambola et al. (2005).

4.2. Parameter setting

We use the following parameters in our experimentation: the genetic parameters are a population size $M = 40$; additionally, our mutation operations are parametrized with two probability parameters $\mathbb{P}_a$ and $\mathbb{P}_p$ as depicted in Algorithm 1. More clearly, mutation on vector A (resp. vector P) is applied according to a Bernoulli probability distribution with parameter $\mathbb{P}_a = 0.7$ (resp. $\mathbb{P}_p = 0.9$). These two parameters were in fact carefully tuned to enhance our mutation phase. When ILS is used with genetic algorithm, δ is fixed to 0.1 and the algorithm was terminated whenever there was no improvement in 100 successive iterations. The maximal time in seconds is fixed to $n * s$ where n is the number of costumers. Within the scope of our work, we force our algorithm to visit only feasible solutions by fixing a large value of the parameter $\alpha = 1000$ in the PENALITY(x).

4.3. Comparative study

In this section, a computational study is carried out to compare our approach with best known solutions. According to the computational experiments, our algorithm outperforms the results giving by Albreda-Sambola et al. (2005) which uses tabu search. Table 1 summarizes the results given by Albreda-Sambola et al. (2005) and our proposed method for the test problems. We use the following notations: %gap, the average deviation of solution value to the lower bound and Time, the running time over ten instances. The average percent deviations relative to the lower bound is computed as:

$$\frac{1}{|C(j,k,l)|}\sum_{i\in C(j,k,l)}\frac{Heu_i - LB_i}{LB_i}\times 100$$

for each group of instances $C(j,k,l)$, $j \in \{S1,S2,S3,M2,M3\}$, $k \in \{a,b,c\}$, $l \in \{p,m,g\}$, with LB_i is the lower bound, Heu_i the best value of the solution obtained either with ILS or T.S and $|C(j,k,l)|$ is the number of instances belonging to one group.

We further do comparative results based on t-test to enrich our comparative study. We study the performance of our algorithm using a t-test evaluation. The t-test is able to achieve a comparison

Table 2
Comparative results based on t-test.

H_0	H_1	t-value	p-value
ILS = T.S	ILS < T.S	−8.72	0.00
GA&ILS = T.S	GA&ILS < T.S	−9.83	0.00
GA&ILS = T.S	GA&ILS < T.S	−1.069	0.142

between the averages of two samples of observations. Let α_1, α_2 be the average deviations of two algorithms 1 and 2 respectively. The tested hypotheses are:

$$\begin{cases} H_0: & \alpha_1 = \alpha_2 \\ H_1: & \alpha_1 < \alpha_2 \end{cases}$$

The hypothesis H_0 implies that the percent deviations of the two algorithms are similar whereas the hypothesis H_1 shows that the average deviation of the solutions value of Algorithm 1 is lower than Algorithm 2. The t-test values comparing TS with ILS and GA&ILS and ILS with the hybrid GA&ILS are given in Table 2. We can observe from the Table 1 that our approach is able to find optimal solution for the *S*1 pool of instances and that our method outperforms the results of Albreda-Sambola et al. (2005) in term of total cost. For all instances of the group *S*1, the hybrid (GA&ILS) converges to optimality while ILS fails at only one instance to achieve the optimal solution ($S1agd_3$). However, the percent deviation of T.S reaches over 2% for the instance ($S1bmd_3$). Besides, we denote that T.S solves the problem to optimality with up to only 30 nodes. Therefore, for small instances, the (GA&ILS) not only gives better solutions than ILS and T.S but also gives all the optimal solutions. For group *S*2, (GA&ILS) and ILS are equal only for the ten instances of type *S*2*am*. The gap of (GA&ILS) decreases slightly compared with ILS otherwise both methods are better than TS. Besides, only on the largest instances, those of the last set *M*3, the ILS is better than TS for all types of instances and is better than the hybrid (GA&ILS) except for the instances of type *M*3*am*, *M*3*bp* and *M*3*cp*. In addition, the table shows that the average time required for computing the solution value relative to T.S is much greater than those of computing with ILS and (GA&ILS) especially for small instances. The best value on each row is indicated in boldface. In addition, after performing a t-test over the 450 instances of our problem for ILS, (GA&ILS) and TS as mentioned in Table 2, we observe that both ILS and hybrid (GA&ILS) dominate the TS with an error risk close to 0 whereas the hybrid (GA&ILS) dominates ILS with an error risk of 15%.

5. Conclusion

This paper introduces a hybrid approach that combines a GA with an ILS to solve the LRP efficiently. Our hybridization is based on an ILS using four neighborhood structures and allowing us to improve each of the generation outputted by a GA. Unlike most of existing common approaches, we pay a special interest in solving jointly the location and routing levels of our LRP. In fact, we have carefully designed our genetic operators and neighborhood structures in order to be able to take into account both of these levels simultaneously. The proposed GA&ILS algorithm was tested on five problem sets existing in the literature. The computational results that we have conducted show that our approach allows to obtain significant improvements over existing results. In fact, not only the solutions obtained by our hybrid GA&ILS algorithm but also their computational requirements, for all instances, are better than the best previously known solutions obtained using a TS heuristic. The results of the comparative study are very encouraging and suggest to apply the variable neighborhood search (VNS) for the LRP with one single route per capacitated open depot.

References

Albreda-Sambola, M., Diaz, J., & Fernandez, E. (2005). A compact model and tight bounds for a combined location-routing problem. *Computers and Operations Research, 32*(3), 407–428.

Baker, B., & Ayechew, M. (2003). A genetic algorithm for the vehicle routing problem. *Computers and Operations Research, 30*(5), 787–800.

Barreto, S. (2004). *Análise e modelização de problemas de localização-distribuição (analysis and modelling of location-routing problems)*. Ph.D. thesis, University of Aveiro, campus universitÃrio de Santiago, Portugal.

Berger, J., & Barkaoui, M. (2004). A parallel hybrid genetic algorithm for the vehicle routing problem with time windows. *Computers and Operation Research, 31*(12), 2037–2053.

Berman, O., Jaillet, P., & Simchi-Levi, D. (1995). *Facility location: A survey of applications and methods*. Springer [Chap. Location-routing problems with uncertainty, pp. 427–452].

Chen, P., Huang, H., & Dong, X. (2010). Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem. *Expert Systems with Applications, 37*, 1620–1627.

Cornuejols, G., Fisher, M., & Nemheuser, G. (1977). Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms. *Management Science, 23*.

Duhamel, C., Lacomme, P., Prins, C., & Prodhon, C. (2009). A graspxels approach for the capacitated location-routing problem. *Computers and Operations Research*.

Gen, M., & Syarif, A. (2005). Hybrid genetic algorithm for multi-time period production/distribution planning. *Computers and Industrial Engineering, 48*, 799–809.

Glover, F., & Kochenberger, G. (2002). *Handbook of metaheuristics*. Boston: Kluwer [Chap. Iterated Local Search, pp. 321–353].

Ho, W., Ho, G., Ji, P., & Lau, H. (2008). A hybrid genetic algorithm for the multi-depot vehicle routing problem. *Engineering Applications of Artificial Intelligence, 76*, 548–557.

Holland, J. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.

Jacobsen, S., & Madsen, O. (1980). A comparative study of heuristics for a two-level routing-location problem. *European Journal of Operational Research, 5*, 378–387.

Karp, R. (1972). *Complexity of computer computations*. New York: Plenum Press [Chap. Reducibility among combinatorial problems, pp. 85–104].

Laporte, G. (1988). Vehicle routing: methods and studies. Golden BL, Assad AA, North-Holland, Amsterdam [Chap. Location-Routing Problems, pp. 163–197].

Laporte, G., & Norbert, Y. (1981). An exact algorithm for minimizing routing and operating costs in depot location. *European Journal of Operational Research, 6*, 224–226.

Laporte, G., Norbert, Y., & Arpin, D. (1986). An exact algorithm for solving a capacitated location-routing problem. *Annals of Operations Research, 6*, 293–310.

Laporte, G., Norbert, Y., & Taillefer, S. (1988). Solving a family of multi-depot vehicle routing and location-routing problems. *Transportation Science, 22*, 161–167.

Lin, S., Lee, Z., Ying, K., & Lee, C. (2009). Applying hybrid meta-heuristics for capacitated vehicle routing problem. *Expert Systems with Applications, 36*, 1505–1512.

Madsen, O. (1983). Methods for solving combined two level location-routing problems of realistic dimensions. *European Journal of Operational Research, 12*, 295–301.

Min, H., Jayaraman, V., & Srivastava, R. (1998). Combined location-routing problems: A synthesis and future research directions. *European Journal of Operational Research, 108*, 1–15.

Murata, T., & Ishibuchi, H. (1994). Performance evaluation of genetic algorithms for flowshop scheduling problems. In: *International conference on evolutionary computation* (pp. 812–817).

Nagy, G., & Salhi, S. (2007). Location-routing: issues, models and methods. *European Journal of Operational Research, 177*, 649–672.

Or, I., & Pierskalla, W. (1979). A transportation location-allocation model for regional blood banking. *AIIE Transactions, 11*, 86–95.

Prins, C., Prodhon, C., Ruiz, A., Soriano, P., & Calvo, R. W. (2007). Solving the capacitated location-routing problem by a cooperative Lagrangean relaxation-granular tabu search heuristic. *Transportation Science, 41*, 470–483.

Prins, C., Prodhon, C., & Wolfer-Calvo, R. (2006). Solving the capacitated location-routing problem by a grasp complemented by a learning process and a path relinking. *4OR – A Quarterly Journal of Operations Research, 4*, 221–238.

Reeves, C. (1995). A genetic algorithm for flowshop sequencing. *Computers and Operations Research, 22*, 5–13.

Reeves, C. (1995). *Modern heuristic techniques for combinatorial problems*. McGraw-Hill Book Company Inc..

Salhi, S., & Rand, G. (1989). The effect of ignoring routes when locating depots. *European Journal of Operational Research, 39*, 150–156.

Sivanandam, S., & Deepa, S. (2008). *Introduction to genetic algorithms*. Berlin Heidelberg: Springer-Verlag.

Srikar, B. R. S. (1983). Solution methodology for the location-routing problem. In: *The ORSA/TIMS conference*.

Stowers, C., & Palekar, U. (1993). Location models with routing considerations for a single obnoxious facility. *Transportation Science, 27*, 350–362.

Wu, T., Low, C., & Bai, J. (2002). Heuristic solutions to multi-depot location-routing problems. *Computers and Operations Research, 29*, 1393–1415.

Yu, V., Lin, S., Lee, W., & Ting, C. (2010). A simulated annealing heuristic for the capacitated location routing problem. *Computers and Industrial Engineering, 58*, 288–299.

Zografos, K., & Samara, S. (1989). Combined location-routing model for hazardous waste transportation and disposal. *Transportation Research Record, 1245*, 52–59.