Ant Metaheuristics with Adapted Personalities for the Vehicle Routing Problem

Nicolas Zufferey[1] Jaime Farres[2] Rémy Glardon[2]

[1] Geneva School of Economics and Management, GSEM - University of Geneva
Blvd du Pont-d'Arve 40, 1211 Geneva 4, Switzerland (n.zufferey@unige.ch)
[2] *LGPP*, École Polytechnique Fédérale de Lausanne, Switzerland
(jaime.farres@epfl.ch, remy.glardon@epfl.ch)

Abstract. At each generation of an ant algorithm, each ant builds a solution step by step by adding an element to it. Each choice is based on the *greedy force* (short term profit or heuristic information) and the *trail* system (central memory which collects information during the search process). Usually, all the ants of the population have the same characteristics and behaviors. In contrast in this paper, a new type of ant metaheuristic is proposed. It relies on the use of ants with different personalities. Such a method has been adapted to the well-known vehicle routing problem, and the obtained average results are very encouraging. On one benchmark instance, new best results have been found.

Keywords: Evolutionary Metaheuristics, Ant Algorithms, Vehicle Routing Problem, Combinatorial Optimization

1 Introduction

As exposed in [29], modern methods for solving complex optimization problems are often divided into *exact* methods and *metaheuristic* methods. An exact method guarantees that an optimal solution is obtained in a finite amount of time. However, for a large number of applications and most real-life optimization problems, which are typically NP-hard, such methods need a prohibitive amount of time to find an optimal solution. For these difficult problems, it is preferable to quickly find a satisfying solution. If solution quality is not a dominant concern, then a simple *heuristic* can be employed, but if quality plays a critical role, then a more advanced *metaheuristic* procedure is recommended. There are mainly two classes of metaheuristics: *local search* and *population based* methods. The former type of algorithm works on a single solution (e.g., descent local search, tabu search, variable neighborhood search), whereas the latter makes a population of (pieces of) solutions evolve (e.g., genetic algorithms, ant colonies, adaptive memory algorithms). The reader interested in a recent book on metaheuristics is referred to [16].

As presented in [20, 30], in most ant algorithms, the role of each ant is to build a solution step by step. At each step, an ant adds an element to the current partial solution. Each *decision* or *move* m is based on two ingredients: the *greedy force*

$GF(m)$ (short-term profit) and the *trail* $Tr(m)$ (information obtained from other ants). The probability $p_i(m)$ that ant i chooses decision m is given by Equation (1), where α and β are parameters, and M_i is the set of admissible decisions that ant i can make.

$$p_i(m) = \frac{GF(m)^\alpha \cdot Tr(m)^\beta}{\sum\limits_{m' \in M_i} GF(m')^\alpha \cdot Tr(m')^\beta} \tag{1}$$

Let M be the set of all possible decisions. When each ant of the population has built a solution, the trails are generally updated as follows: $Tr(m) = \rho \cdot Tr(m) + \Delta Tr(m)$, $\forall m \in M$, where $0 < \rho < 1$ is a parameter representing the evaporation of the trails, which is usually close or equal to 0.9, and $\Delta Tr(m)$ is a term which reinforces the trails left on decision m by the ant population. That quantity is usually proportional to the number of times the ants have made decision m, and to the quality of the obtained solutions when decision m was made. More precisely, let N be the number of ants, then: $\Delta Tr(m) = \sum_{i=1}^{N} \Delta Tr_i(m)$, where $\Delta Tr_i(m)$ is proportional to the quality of the solution provided by ant i if it has made decision m. The pseudo-code of a classical ant method is given in Algorithm 1. A *generation* consists in performing steps (1) to (4). A stopping condition can be a maximum number of generations or a maximum time limit.

The paper is organized as follows. In Section 2, the most well-known extensions and variants of the classical ant algorithm are discussed. The vehicle routing problem (*VRP*) is presented in Section 3, where state-of-the-art metaheuristics are briefly reviewed. In Section 4, five new algorithms are proposed for the *VRP*, and the results are presented in Section 5. A conclusion is given in Section 6, where the main contributions of this paper are highlighted.

Algorithm 1 Classical ant metaheuristic

```
While no stopping condition is met, do:
    1. for $i = 1$ to $N$, do: ant $i$ builds a solution $s_i$ step by step based on Equation (1);
    2. intensification (optional): apply a local search to some solutions of $\{s_1, \ldots, s_N\}$;
    3. update $s^\star$ (best encountered solution during the search);
    4. update the trails by the use of a subset of $\{s_1, \ldots, s_N\}$;
Output: solution $s^\star$.
```

2 Ant algorithms

As presented in [11], ant algorithms have been developed for many problems of different types. Several variants or extensions of the above Algorithm 1 can be found in the literature. Some of them are briefly discussed below.

Elitist ants trails. It is one of the first improvement of the classical ant algorithms. It biases the trail updating rule to converge faster to the most promising area of the search space. For example, at the end of each generation, only the best ants of the generation can update the trail system [3].

Pseudo-random proportional selection. For each ant, this rule is used at each iteration of the constructing process. It selects with probability q_0 the element that maximizes $GF(m)^\alpha \cdot Tr(m)^\beta$, and uses Equation (1) with probability $(1 - q_0)$. This selection rule is one of the most used and has proved to be a very easy way to regulate the balance between intensification and diversification through the parameter q_0. It was first presented in [12].

Bounded trails. This variant was first presented in the MAX-MIN ant system in [26]. It consists in having upper and lower bounds for the trail values. The lower bound avoids the algorithm to discard some solutions and ensures the asymptotic convergence, as every solution has always a probability above 0 of being generated. The upper bound avoids the algorithm to focus all its attention to a region of the search space. This mechanism has a strong diversification ability.

Candidate lists. It consists in reducing the number of possible choices to decrease the computational effort at each iteration of the constructing process. For example, only the e (parameter) elements with the best greedy forces can be chosen for a move. For instance in [13], only the e closest clients can be chosen in the construction of a solution of the traveling salesman problem.

Hyper-cube framework. Presented for the first time in [2], this technique uses weighting parameters w_s (each w_s is proportional to the quality of solution s) in the trail updating rule. It limits the trail values to interval $[0, 1]$ and it has been theoretically proved to continuously increase the expectation of the average solution quality over time.

Multiple Ant Colony System (*MACS*). It consists in several groups of ants that have their own trail system. In most *MACS*, there exists some kind of interaction or exchange of information between the different groups. A *MACS* can optimize different objective functions. In this case, each colony focuses on optimizing its own objective function and interacts with the other ones to create a global best solution. This is the case for the *VRP* with time windows, for which a *MACS* was first proposed in [14]. Having more than one trail system provides the algorithm with a very important diversification potential.

Other ant paradigms. In most ant algorithms, the role of each ant is to build a solution in a constructive way, basing each decision on the greedy force and the trails. However, different roles are possible for each individual ant, ranging from a negligible help in the decision process to a refined local search such a tabu search (e.g., [23, 28, 30, 31]).

3 Presentation of the *VRP* and literature review

The *VRP* is one of the most popular problems in combinatorial optimization because of its obvious applications in transportation. It consists in designing the route of each of the k identical vehicles with the aim of minimizing the total traveled distance f (or the total cost or the total travel time). All vehicles are initially in a depot, where each route starts and ends. Each client v (with demand $D(v)$) has to be visited once by the collection of routes. The problem is defined in an undirected graph $G = (V, E)$, where $V = \{v_0, v_1, \ldots, v_n\}$ is the vertex set and $E = \{(v_i, v_j) \mid v_i, v_j \in V, i < j\}$ is the edge set. Note that v_0 is the depot and the other vertices are clients. The following lexicographical approach is generally used: minimize k, then the total distance f. The two most well-known constraints associated with the VRP are: (1) *capacity*: each vehicle has a limited capacity Q, thus the demand of each route cannot exceed Q; (2) *autonomy*: each vehicle has a limited autonomy A, thus the total duration of each route cannot exceed A. Several extensions of the *VRP* can be found in the literature. In this paper, only the capacity constraint is considered, which is the most studied version of the *VRP*.

A few ant algorithms have been proposed in the literature for the *VRP*, but none belongs to the best *VRP* metaheuristics. The first ant algorithm for the *VRP* was presented in [4]. In the most basic version, each ant constructs a solution by choosing the next client (among the non visited ones plus the depot) to visit according to Equation (1). When the selection of a client leads to an infeasible solution, the route is closed and another route is started. The greedy force of an edge is the inverse of its length. The trail updating rule is based on an elitist technique. Some variations of this algorithm have been tested in the literature. The most successful variations are described below. The distance between two clients is not the only relevant heuristic information. In [3], a new probability rule is described and includes the *savings* s_{ij} (advantage of combining two cities i and j as consecutive elements in a tour) and the *capacity utilization* (portion of the vehicle capacity used if the next considered client is chosen). In [24], local search procedures (e.g. the well-known *2-opt* heuristic based on the *cross exchange*, the move *swap* or *reinsert*) have improved the performance of the discussed ant algorithms. A mutation operator which belongs to the genetic algorithm paradigm is introduced in [1]. With a certain probability (which is dynamically managed during the search), the algorithm selects two tours from parent solutions and exchanges two nodes (unfeasible solution are penalized but not forbidden). The resulting solution is improved with the *2-opt* heuristic. When updating the trail of an edge (i, j) with a solution s, two components are considered: the value of s and the contribution of the length of (i, j) to the tour it belongs to.

For survey papers on the *VRP*, the reader is referred to [7–9, 15, 17, 18]. Many algorithms have been developed for the *VRP*. Among them, there are some successful classical heuristics such as Clarke & Wright, Two-matching, Sweep, 1-Petal and 2-Petal, as tested in [8]. However, the best performance is achieved by metaheuristics. Four of them outperforms the others and are discussed below.

Adaptive Memory (*AM*). *AM* [25] has been proved to be a good algorithm for the *VRP* and introduces a very innovative approach. At each generation of *AM*, an offspring solution s is built route by route from a central memory M (which contains routes), then s is improved with a local search, and the resulting solution is used to update M (i.e. routes of M are replaced with routes of s).

Unified Tabu Search (*UTS*). *UTS* [10] has been proved to be a very flexible algorithm (easily adapted to variations of the *VRP*) with competitive quality and speed. *UTS* relies on a tabu search using an objective function which dynamically penalizes the constraint violations (the penalty component is likely to be increased if the last iterations violate the constraints).

Granular Tabu Search (*GTS*). *GTS* [27] has been proved to be a very balanced algorithm in terms of speed and quality. It uses a tabu search framework and relies on the use of *granular* neighborhoods to discard the edges that rarely would belong to a competitive solution. *GTS* uses a granularity threshold which is dynamically adjusted.

Active Guided Evolution Strategies (*AGES*). *AGES* [21] has been proved to be very efficient (it is probably the best *VRP* method), with a reasonable speed. *AGES* is a combination of several procedures (including local search techniques), but an important drawback is its significant number of parameters.

4 New algorithms for the *VRP*

4.1 *GR*: A greedy constructive algorithm with restarts

There are many constructing algorithms for the *VRP*, such as the *savings* algorithm [6]. Most of them are deterministic and as a consequence generate always the same solution. We propose a greedy constructing procedure *GR* with randomness, able to generate different solutions if restarted, which is the core procedure of the proposed ant metaheuristics. It works with a given number k of vehicles and it is restarted as long as a given time limit is not reached. At the end, the best generated solution is returned to the user.

GR consists in sequentially constructing each of the k routes. The procedure starts a new route R by choosing randomly an unserved client $v \in \{v_1, \ldots, v_n\}$, and creates a tour $v_0 - v - v_0$. Let $C(R)$ be the capacity of route R (defined as the vehicle capacity, minus the demands $D(R)$ of all the clients belonging to R). Then, for all the unserved clients v such that $D(v) \leq C(R)$ (called the R-*available* clients), a move $m = (v, p, R)$ can to be performed, which consists in inserting client v at position p (between two clients v_i and v_j, or between the depot v_0 and one client v_i) in route R. To do it, the greedy force $GF(v, p, R)$ is first computed for each R-available client v, for each position p of the considered route R. $GF(v, p, R)$ is defined by dividing the distance $d(v, v_0)$ between v and the depot v_0 by the augmentation $\Delta f_R(v, p)$ of the length of R if it is extended by inserting client v at position p. This greedy force is new and in contrast with the existing greedy forces proposed in the literature, it favors the insertion of clients located far away from the depot, which is likely to reduce the number of *isolated* clients (i.e. an unserved client v with a large $d(v, v_0)$ value). If too many isolated clients are left for the next routes, such routes are likely to be long. However, the consideration of $\Delta f_R(v, p)$ avoids the insertion of isolated clients located far away from the route R under construction. When the greedy force of each R-available move has been computed, all the possible choices have been identified and evaluated for the considered route R. At this moment, the *greedy force threshold* $GFT(R)$ is computed as $q_A \cdot \max_{v,p} GF(v, p, R)$. It is the largest available greedy force, multiplied by parameter q_A (tuned to 0.9) which regulates the amount of possible moves and corresponds to a candidate list technique. At the end of each iteration, the selected move m is randomly chosen among the ones whose greedy forces is above the threshold.

At each iteration of *GR*, these three mains steps are performed for the considered route R (i.e., compute the $GF(m)$'s, then $GFT(R)$, and finally select a move m) until there is no more R-available client v. When this occurs, a new route is started by choosing randomly an unserved client. The process stops when all the clients have been served (feasible solution), or when it is not anymore possible to serve a client with one of the k vehicles (unfeasible solution).

4.2 *ANT*: An ant algorithm with two phases

In *ANT*, the role of each ant is to build a solution with a 2-phase algorithm denoted *2PH*, where a trail value is associated with each edge. In the first phase (P1), the routes are sequentially built and extended as in *GR*, whereas in the second phase (P2), the unserved clients are sequentially considered to fill any of the existing routes. In other words, (P1) works tour by tour, whereas (P2) works client by client (by order of decreasing demands, which improves the likelihood of the solution to be feasible). The transition between the two phases is one of the challenging issues. The key idea is to stop the construction of a route R in (P1) when only poor R-available insertions are possible (i.e. do not fill route R just to fill it, because these R-available clients might be much more efficiently served by other routes).

Based on Algorithm 1, each generation of *ANT* consists in the following steps: (1) construct a solution with each of the N (parameter tuned to 12) ants, using *2PH*; (2) is skipped; (3) update $s^\star$; (4) update the trails and compute a *trail threshold* TT, which is used to decide when to move from (P1) to (P2) in *2PH* (in the next generation). More precisely, the role of TT is to detect when the potential of (P1) becomes poor, in the sense that even if a client v is inserted at the best position in the considered route R (as it can be done in (P1)), it could be much better to assign v to another vehicle (as it can be done in (P2)). TT is computed as $\min(t_B \cdot T_B, t_S \cdot T_S)$, where t_B and t_S are parameters respectively tuned to 0.2 and 3, T_B is the average trail value in the best encountered solution $s^\star$ during the search, and T_S is the average trail value in the whole trail system.

The trails are updated as follows. First, for each edge (i, j), the evaporation coefficient ρ is used to set $Tr(i, j) = \rho \cdot Tr(i, j)$. Then, the N_b (parameter tuned to 4) best solutions (i.e. the *elite* solutions) of the last generation are used to reinforce the trails of the edges that appear in the elite solutions. More precisely, for each elite solution s, $Tr(i, j)$ is augmented by $(1 - \rho) \cdot w \cdot [f(s)/f^\star]$, where $f^\star$ is the value of $s^\star$, and w is a weighting parameter tuned to 0.1. One can remark that parameter ρ (tuned to 0.975) can regulate the balance between the evaporation and the reinforcement of the trails.

(P1) is derived from *GR* with the two following differences: (1) the probability of a move is proportional to its associated trail value; (2) (P1) stops when the trail of the selected move is below TT (i.e. (P1) does not only stop because of non sufficient capacity). More precisely, let $p(v, p, R)$ be the probability to insert client v at position $p = (v_i, v_j)$ in route R. Such a probability depends on the trail $Tr(v, p, R) = \max[Tr(v_i, v), Tr(v, v_j)]$. Consequently, a sequence of clients that appears to be good in the previous generations is more likely to be created. The "maximum" is used (instead of the "summation") because at that stage, many edges are going to be broken by the insertion of the next clients.

(P2) starts by calculating the greedy force $GF(v, p, R)$ of the considered client v for each position $p = (v_i, v_j)$ of each existing route R that has a sufficient remaining capacity to serve v. In contrast with (P1), the trail is computed as $Tr(v, p, R) = Tr(v_i, v) + Tr(v, v_j)$. The summation is performed because unlike in (P1), the added edges are likely to stay in the final solution. As in (P1), the probability of a move (among the ones above GFT) is proportional to its trail value. If a client v cannot be placed in any route R because $D(v) > C(R)$, the solution is unfeasible.

Because of the sequential use of the greedy forces and the trails, there is no need to use parameters α and β of Equation (1), which results in a significant reduction of the computational effort and a better overall performance, as discussed in [31]. We have now all the ingredients to summarize *2PH* in Algorithm 2.

Algorithm 2 *2PH*: The two-phases algorithm associated with each ant

```
(P1) While there is a free vehicle, do:
  1. select a free vehicle;
  2. select randomly an unserved client $v$ and build the route $R = v_0 - v - v_0$;
  3. while there is at least a $R$-available client $v$, do:
     (a) compute $GF(v, p, R)$ for each $R$-available client $v$;
     (b) compute $GFT(R)$;
     (c) select a move $m = (v, p, R)$ (based on their trail values) among the moves such
         that $GF(v, p, R) \geq GFT(R)$;
     (d) if $Tr(m) < TT$, STOP (select another free vehicle, if any);
(P2) For each unserved client $v$ (ordered by decreasing demand), do:
  1. if there is no route $R$ such that $D(v) \leq C(R) - D(R)$, STOP (unfeasible solution);
  2. for each route R such that $D(v) \leq C(R) - D(R)$, compute $GF(v, p, R)$ and
     $Tr(v, p, R)$;
  3. compute $GFT$ (over all the possible moves and routes at that step);
  4. select a move $m = (v, p, R)$ (based on their trail values) among the moves such
     that $GF(v, p, R) \geq GFT(R)$;
Output: feasible/unfeasible VRP solution.
```

4.3 *AL*: *ANT* enhanced with local search techniques

Often, in order to get competitive results, it is unavoidable to apply a local search method (e.g., a descent method, tabu search) to the solutions provided by the classical constructive ants [31]. Widely used neighborhood structures for the *VRP* are: (1) the forward and backward *Or-exchange* [22], where the neighborhood structure consists in moving a client from a route to another; (2) the *2-opt* [19], where a move consists in removing two edges of a route and rebuilding the solution by creating a different pair of edges. *2-opt* appears to be very efficient while combined with the forward and backward *Or-exchange* (in one configuration of the algorithm proposed in [21], these are the only three local search procedures used). Moreover, the *2-opt* local search is the most used in literature for the *VRP* because of its simplicity, its speed and its capacity to improve the solutions by making intra-route improvements.

At the end of each generation, before updating the trail system (i.e. at step (2) of Algorithm 1), the following local search techniques are sequentially applied to the elite solutions: the *2-opt*, the forward *Or-exchange*, the backward *Or-exchange*. This sequence of three local search procedures is restarted until no more improvement is encountered by any of the procedure.

4.4 *ALM*: *AL* enhanced with a central memory

ALM is derived from *AL* by adding an intensification component at the beginning of *2PH*, before (P1). This component (P0) consists in copying some of the routes of $s^\star$ when generating a solution s with the considered ant. More precisely, each route R of $s^\star$ has a probability $p(R)$ to be copied in s, which depends on two elements: the *saturation* $Sat(R)$ of R and the mutual *attractiveness* $Att(R)$ of the clients belonging to R. The probabilities $p(R)$ are updated at each generation, after $s^\star$ has been updated (i.e. at the end of step (3) of Algorithm 1).

On the one hand, the saturation $Sat(R)$ of a route R is defined as $D(R)/C(R)$. The larger it is, the better R is filled (which favors the likelihood of a solution to be feasible). On the other hand, let M be a central memory containing the elite solutions of the M_b (parameter tuned to 10) previous generations. For a given client v, we define $f_M(v)$ as the average length of the routes in M which serve v. In addition, $f_R(s^\star)$ is the length of R in $s^\star$. The attractiveness $Att(R)$ of the clients belonging to route R of $s^\star$ can now be defined as $\prod_{v\in R} f_R(s^\star)/f_M(v)$. The larger it is, the better are likely to be the solutions which group together the clients of R. Finally, probability $p(R)$ is computed as in Equation (2), where q_M is a parameter (tuned to 0.4) which can regulate the influence of $s^\star$ on the solution s generated by the involved ant. Note that if $p(R)$ exceeds 1, we simply set $p(R) = 1$.

$$p(R) = q_M \cdot Sat(R) \cdot Att(R) = q_M \cdot \frac{D(R)}{C(R)} \cdot \prod_{v\in R} \frac{f_R(s^\star)}{f_M(v)} \qquad (2)$$

4.5 *ALMP*: *ALM* with different ant personalities

The idea of *ALMP* is to assign a specific personality to each of the N ants of the population. The personality intervenes anytime the ant makes a decision, which consists in selecting a move among the ones above GFT. Four ant personalities are proposed: *Normal Ants* (*NA*), *Follower Ants* (*FA*), *Moody Ants* (*MA*) and

Innovative Ants (*IA*). These characteristics are likely to belong to any group of individuals working together to reach a common goal. In order to work with a well-balanced ant society, we propose to use $N/4$ ants of each personality (remind that N was tuned to 12).

NA corresponds to the average ant personality as presented in Subsection 4.2. *NA* selects a move proportionally to its trail value.

FA corresponds to the personality that strictly follows what others have done previously. *FA* always selects the move with the largest trail value. This behavior aims at intensifying the search.

MA corresponds to *NA* with probability $(1 - p_{MA})$, but with a probability p_{MA} (parameter tuned to 0.4), it changes its mood and starts behaving apparently against the goal. *MA* selects a move proportionally to the trail values with probability $(1 - p_{MA})$, and inverse-proportionally to the trail values with probability p_{MA}. This behavior aims at strongly diversifying the search.

IA corresponds to the personality that tends to behave in an unusual way, but with the intention to reach the goal. *IA* corresponds to *FA* with probability $(1 - p_{IA})$ (intensification role), but with a probability p_{IA} (parameter tuned to 0.2), it changes its mood and make a random decision (diversification role). *IA* selects the move with the largest trail values with probability $(1 - p_{IA})$, and randomly with probability p_{IA}. For this personality, the value of parameter q_A which appears in $GFT = q_A \cdot \max_{v,p} GF(v, p, R)$ is lower than usual (it is tuned to 0.8 instead of 0.9), which means that the number of possible choices is larger than for the other personalities, which favors the exploration of new solutions.

5 Results

The algorithms have been coded in C++ and compiled by Microsoft Visual Studio 2013. The tests have been run in a Windows 7 PC with an Intel Core2 Quad Q9400 of 2.66GHz and 4MB of RAM in 32-bit but only using one of the 4 processors of the PC. To make the results comparable to other results obtained with other computers, a GFlops test has been performed with the software LinX 0.6.5. The obtained result is 9.1 GFlops. For each proposed algorithm, the stopping condition is $5 \cdot n$ seconds, where n is the number of clients of the considered instance. The results are averaged over 9 runs (as 9 processors were available). The considered instances are all the benchmark instances from [5, 17] which do not have the autonomy constraint. More precisely, the instances are 1 to 5, 11 and 12 from [5], and 9 to 20 from [17]. For each instance, the smallest number k of vehicles and the best-known solution value $f^\star$ are taken from [21].

The results are provided in Table 1. The five first columns indicate respectively: the instance name (starting with a "C" if from [5], and with a "G" if from [17]), the number n of clients, the number k of vehicles, the instance saturation Sat computed as the total demand divided by the total capacity of the vehicles, the time $t^\star$ (in seconds) to get the best known value $f^\star$ (obtained from [21]). Column 6 indicates on the one hand the average percentage gap between GR and $f^\star$, and on the other hand the average computing time (in brackets) needed to get the best results of *GR*. Columns 7 to 10 provide the same information, but for *ANT*, *AL*, *ALM* and *ALMP*, respectively. The times (indicated in seconds) are all re-scaled according to the above mentioned computer (based on the corresponding GFlops performance), so that they can be fairly compared. Average results are given in the last line. Remind that for the *VRP*, minimizing k is more important than minimizing the total traveled distance f. In this respect C14 is of particular interest, as the newly proposed algorithms are able to generate solutions with $k = 29$ vehicles instead of $k = 30$ (as it is the case in the existing literature). This indicates that the proposed algorithm *2PH* has a strong ability to find feasible solutions with large values of Sat. For this reason, C14 is not considered to compute the average results in the last line of Table 1.

From Table 1, it can be concluded that every ingredient (i.e., a trail system, local search procedures, a central memory, and various personalities) successively added to *GR* to derive *ALMP* is useful, as the average percentage gap is reduced step by step from 12.5% to 11% to 7.5% to 3.7% to 3.3%. Other experiments, which are not detailed here, confirm this statement: it was observed that each ingredient improves significantly the solution values even if other time limits are used (ranging from n to $5 \cdot n$ seconds). In addition, it was also observed that *ALMP* with the proposed mix of personalities is better than if only one personality is used (i.e. there is no personality which outperforms the proposed mix of personalities).

Table 1. Results on well-known benchmark instances

1	2	3	4	5	6	7	8	9	10
Inst.	n	k	Sat	$t^\star$	GR	ANT	AL	ALM	$ALMP$
C01	50	5	97.1%	1	2.4% [82s]	1.4% [130s]	0.3% [88s]	0.9% [25s]	1.3% [46s]
C02	75	10	97.4%	22	11.3% [187s]	8.7% [177s]	3.9% [121s]	1.8% [98s]	1.4% [124s]
C03	100	8	91.1%	4	13.9% [280s]	9.0% [299s]	1.6% [251s]	0.4% [176s]	0.5% [211s]
C04	150	12	93.1%	41	19.9% [338s]	15.1% [229s]	4.4% [491s]	2.0% [364s]	1.7% [630s]
C05	199	16	99.6%	8640	20.0% [289s]	20.2% [575s]	13.0% [446s]	6.2% [721s]	5.5% [746s]
C11	120	7	98.2%	4	12.0% [233s]	6.3% [232s]	4.3% [287s]	4.0% [155s]	0.9% [309s]
C12	100	10	90.5%	1	10.6% [322s]	7.6% [227s]	1.2% [223s]	0.0% [92s]	0.0% [58s]
G09	255	14	95.9%	1441	11.7% [671s]	9.8% [736s]	8.1% [757s]	2.9% [562s]	3.1% [1006s]
G10	323	16	95.0%	300	13.2% [716s]	11.9% [848s]	8.8% [1081s]	4.2% [1353s]	4.1% [1492s]
G11	399	18	94.3%	1763	14.6% [1009s]	13.2% [1249s]	9.7% [1270s]	5.3% [1716s]	5.1% [1833s]
G12	483	19	98.4%	2591	12.6% [1573s]	12.4% [1496s]	11.8% [793s]	5.9% [2161s]	4.8% [2324s]
G13	252	26	96.7%	1600	10.4% [539s]	9.9% [607s]	8.4% [472s]	3.0% [1061s]	3.2% [886s]
G14	*320*	*29*	*98.9%*	*N/A*	*1255 [791s]*	*1261 [867s]*	*1245 [775s]*	*1170 [1510s]*	*1173 [1516s]*
G15	396	33	97.7%	110	12.0% [985s]	11.9% [1075s]	10.2% [849s]	5.8% [1743s]	5.3% [1899s]
G16	480	37	96.7%	3200	14.1% [1429s]	13.3% [1168s]	11.1% [1419s]	7.3% [1957s]	6.2% [2136s]
G17	240	22	98.2%	121	10.0% [638s]	9.7% [495s]	6.6% [271s]	1.7% [994s]	1.7% [1064s]
G18	300	27	100.0%	600	10.0% [663s]	10.5% [648s]	9.0% [904s]	3.6% [1256s]	3.6% [1312s]
G19	360	33	98.2%	93	12.9% [979s]	12.9% [916s]	10.1% [640s]	5.0% [1619s]	5.2% [1747s]
G20	420	38	99.5%	920	12.9% [1620s]	13.4% [988s]	11.7% [1312s]	6.0% [2026s]	5.8% [1964s]
Avg.					**12.50%**	**11.00%**	**7.50%**	**3.70%**	**3.30%**

6 Conclusion

In this paper, the vehicle routing problem is tackled with new approaches. The main contributions are the following: (1) five new algorithms are developed for the *VRP*; (2) a new type of ant algorithm is designed (relying on the use of various personalities of ants), which can be applied to any combinatorial optimization problem; (3) a new type of greedy force is proposed for the *VRP*, which, in contrast with the existing ant methods for the *VRP*, favors clients located far away from the depot (this augments the likelihood of a solution to be feasible); (4) the performance of the best proposed metaheuristic is very encouraging, as the average percentage gap is 3% with respect to the best-known results, and new best results are provided for one benchmark instance. Future works include the adaptation of the ant algorithm with personalities to other problems.

References

1. Y. Bin, Y. Zhong-Zhen, and Y. Baozhen. An improved ant colony optimization for vehicle routing problem. *European Journal of Operational Research*, 196:171–176, 2009.
2. C. Blum and M. Dorigo. The hyper-cube framework for ant colony optimization. *IEEE Trans Syst Man Cybernet Part B*, 34 (2):1161 – 1172, 2004.
3. B. Bullnheimer and R. F. Hartl anc C. Strauss. A new rank-based version of the Ant System: A computational study. *Central European Journal for Operations Research and Economics*, 7 (1):25 – 38, 1999.
4. B. Bullnheimer, R. F. Hartl, and C. Strauss. An improved Ant System algorithm for the Vehicle Routing Problem. *Annals of Operations Research*, 89:319 – 328., 1997.
5. N. Christofides, A. Mingozzi, and P. Toth. *Combinatorial Optimization*, chapter The vehicle routing problem, pages 315 – 338. 1979.
6. G. Clarke and J. R. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12 (4):568 – 581, 1964.
7. J.-F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, and J.-S. Sormany. *Logistics Systems: Design and Optimization*, chapter New Heuristics for the Vehicle Routing Problem, pages 270–297. Springer, 2005.
8. J.-F. Cordeau, M. Gendreau, G. Laporte, J.-Y. Potvin, and F. Semet. A Guide to Vehicle Routing Heuristics. *Journal of the Operational Research Society*, 53 (5):512–522, 2002.
9. J.-F. Cordeau and G. Laporte. *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*, chapter Tabu search heuristics for the vehicle routing problem, pages 145–163. Kluwer, Boston, 2004.
10. J.-F. Cordeau, G. Laporte, and A. Mercier. A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows. *Journal of the Operational Research Society*, 52:928–936, 2001.
11. M. Dorigo, M. Birattari, and T. Stuetzle. Ant colony optimization – artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*, 1 (4):28–39, 2006.

12. M. Dorigo and L.M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.
13. M. Dorigo and T. Stuetzle. *Handbook of Metaheuristics*, volume 57, chapter The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances, pages 251–285. In F. Glover and G. Kochenberger (Eds), 2003.
14. L. M. Gambardella, E. Taillard, and G. Agazzi. *New Ideas in Optimization*, chapter MACS-VRPTW: A Multiple Ant Colony System for vehicle routing problems with time windows, pages 63 – 76. London, UK, McGraw-Hill, 1999.
15. M. Gendreau, G. Laporte, and J.-Y. Potvin. *The Vehicle Routing Problem*, chapter Metaheuristics for the VRP, pages 129–154. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 2002.
16. M. Gendreau and J.-Y. Potvin. *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*. Springer, 2010.
17. B. L. Golden, E. A. Wasil, J. P. Kelly, and I.-M. Chao. *Fleet Management and Logistics*, chapter Metaheuristics in vehicle routing, pages 33–56. Kluwer, Boston, 1998.
18. G. Laporte and F. Semet. *The Vehicle Routing Problem*, chapter Classical heuristics for the capacitated VRP, pages 109–128. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 2002.
19. S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44:2245–2269, 1965.
20. L. Luyet, S. Varone, and N. Zufferey. An Ant Algorithm for the Steiner Tree Problem in Graphs. *Lecture Notes in Computer Science*, 4448:42 – 51, 2007.
21. D. Mester and O. Braysy. Active-guided evolution strategies for large-scale capacitated vehicle routing problems. *Computers & Operations Research*, 34 (10):2964 – 2975, 2007.
22. I. Or. *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking.* PhD thesis, Nortwester University, USA, 1976.
23. M. Plumettaz, D. Schindl, and N. Zufferey. Ant local search and its efficient adaptation to graph colouring. *Journal of the Operational Research Society*, 61:819 – 826, 2010.
24. M. Riemann, K. Doerner, and R. F. Hartl. Analyzing a Unified Ant System for the VRP and Some of its Variants. *Applications of Evolutionary Computing*, 2611:300–310, 2003.
25. Y. Rochat and E. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1:147–167, 1995.
26. T. Stuetzle and H. Hoos. MAX-MIN Ant System. *Future Generation Computer Systems*, 16 (9):889 – 914, 2000.
27. P. Toth and D. Vigo. The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. *INFORMS Journal on Computing*, 15 (4):333 – 346, 2003.
28. N. Zufferey. *Heuristiques pour les Problèmes de la Coloration des Sommets d'un Graphe et d'Affectation de Fréquences avec Polarités*. PhD thesis, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland, 2002.
29. N. Zufferey. Metaheuristics: some Principles for an Efficient Design. *Computer Technology and Applications*, 3 (6):446 – 462, 2012.
30. N. Zufferey. Optimization by ant algorithms: Possible roles for an individual ant. *Optimization Letters*, 6 (5):963 – 973, 2012.
31. N. Zufferey. Design and Classification of Ant Metaheuristics. In *Proceedings of the 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 339 – 343, 2014.