Les Cahiers du GERAD ISSN: 0711–2440

An Ant Colony Optimization Metaheuristic for the Undirected Rural Postman Problem

D. Laganà, G. Laporte, F. Mari,
R. Musmanno, O. Pisacane

G–2007–106

December 2007

An Ant Colony Optimization Metaheuristic for the Undirected Rural Postman Problem

Demetrio Laganà
Dipartimento di Elettronica, Informatica e Sistemistica
Università della Calabria, 87036 Rende (CS), Italy
lagana@hpcc.unical.it

Gilbert Laporte
GERAD and Canada Research Chair in Distribution Management
HEC Montréal, Montréal (Québec) Canada, H3T 2A7
gilbert@crt.umontreal.ca

Francesco Mari
Supercomputing Center for Computational Engineering (CESIC), NEC Italy
Ponte Pietro Bucci 22B 87036 Rende (CS), Italy
francesco.mari@cesic.neceur.com

Roberto Musmanno
Ornella Pisacane
Dipartimento di Elettronica, Informatica e Sistemistica
Università della Calabria, 87036 Rende (CS), Italy
musmanno@unical.it, opisacane@deis.unical.it

December 2007

Les Cahiers du GERAD

G–2007–106

Abstract

This paper describes a new heuristic for the well-known *Undirected Rural Postman Problem*. It consists of two steps: it first constructs a partial solution using the Ant Colony Optimization metaheuristic, and the remaining required edges are then gradually inserted. Computational results on a set of benchmark instances are presented and comparisons with alternative heuristics are performed. The optimality gap is also computed by running a branch-and-cut algorithm.

Key Words: Ant Colony Optimization, Arc Routing, Undirected Rural Postman Problem, Heuristics.

Résumé

Cet article décrit une nouvelle heuristique pour le problème du postier rural non orienté. Cette heuristique est constituée de deux étapes : dans une phase on construit une solution partielle à l'aide d'un algorithme de fourmis; dans une deuxième phase, les arêtes restantes sont graduellement insérées. Des résultats numériques sur des problèmes tests permettent de comparer l'heuristique proposée à d'autres heuristiques. L'écart d'optimalité est aussi calculé à partir des résultats obtenus par un algorithme de séparation et coupes.

Mots clés : algorithme de fourmis, tournées sur les arcs, problème du postier rural non orienté, heuristiques.

Acknowledgments: This work was partly supported by the Ministero dell'Università e della Ricerca Scientifica (MURST), the Center of Excellence on High-Performance Computing and the University of Calabria, Italy, and by the Canadian natural Sciences and Engineering Research Council under grant 39682-05. The authors also benefited from the support of the Department of Electronics, Informatics and Systems (DEIS-University of Calabria-Italy). This support is gratefully acknowledged.

1 Introduction

The *Undirected Rural Postman Problem* (URPP) is defined on an undirected graph $G = (V, E)$, where V is the vertex set, E is the edge set, $c_{ij} \geq 0$ is the cost of traversing edge $(v_i,\ v_j) \in E$, d_{ij} is the cost of a shortest chain between v_i and v_j, and $E_R \subseteq E$ is a set of required edges. Let V_R be the set of vertices incident to a required edge, and let $G_R = (V_R, E_R)$ the subgraph induced by E_R. The graph G_R consists of p disconnected subgraphs called *connected components*. Let C_i be the i^{th} connected component of the subgraph G_R, and let V_i be its vertex set. The URPP consists of determining a minimum cost tour traversing each edge of E_R at least once. An URPP feasible solution is represented by a closed sequence of vertices $S = (v_0, \ldots, v_n = v_0)$, covering all edges of E_R. A URPP partial solution is a sequence of vertices $\overline{S} = (v_0, \ldots, v_n = v_0)$, covering a subset E'_R of required edges.

The URPP was introduced by Orloff [33] and was shown to be NP-hard by Lenstra and Rinnooy Kan [31]. Some URPP applications arise in the control of plotting and drilling machines [27], and in the optimization of laser-plotter beam movements [24]. Exact algorithms for the URPP have been proposed by Christofides et al. [5], Corberán and Sanchis [6], Letchford [32], Ghiani and Laporte [26] and Fernández et al. [21]. Constructive heuristics for the URPP have been presented by Frederickson [22], and by Pearn and Wu [34], while improvement procedures have been described by Hertz, Laporte and Nanchen-Hugo [29], and Groves and van Vuuren [28].

The aim of this paper is to describe an ant colony optimization (ACO) metaheuristic for the URPP. The remainder of the paper is organized as follows. In Section 2 we introduce the ACO metaheuristic for the URPP, called ANTURPP, and we describe the procedures that have been implemented. In Section 3 we introduce F-Race, a procedure employed to set the ANTURPP parameters. Computational results follow in Section 4.

2 ANTURPP: An ant colony heuristic for the URPP

ACO is a metaheuristic introduced by Dorigo [11] and successively extended by Dorigo in [12, 13, 14, 15], Cordón et al. [7], and Dorigo and Stützle in [16, 17]. Interesting applications of ACO were described by Gambardella, Taillard and Agazzi [23] and Doerner et al. [9] for the *Vehicle Routing Problem*. Recent applications were proposed by Doerner et al. [10] and Lacomme, Prins and Tanguy [30] for the *Capacitated Arc Routing Problems*. The ACO paradigm is inspired from the behaviour of real ants. Ants build a shortest path between a food source and their nest. Initially, they explore the area surrounding their nest in a random manner. When an ant finds a source of food, it carries some of it to the nest and deposits a pheromone trail on the ground. The quantity of deposited pheromone depends on the quantity and quality of the food found and will guide other ants to the food source. This indirect communication between the ants via the pheromone trails allows them to find a shortest path between their nest and food sources. In ACO algorithms, the pheromone

trails are simulated via a parameterized probabilistic model called the *pheromone model*, which consists of a set of parameters whose values are called the *pheromone values*, and used to probabilistically generate solutions. In general the ACO optimization strategy consists of iterating the following two steps:

1. solutions are built using a pheromone model, that is, a parameterized probability distribution over the solutions space;
2. the solutions built in earlier iterations are used to modify the pheromone values so as to bias the search toward high quality solutions.

In the following, we introduce the basic procedures of ANTURPP. An ant is a simple interacting software agent capable of building a partial solution for the URPP. An anthill k is a set of b_k ants, and B is the set of anthills. For all anthills $k \in B$ and for all ants $j = 1, \ldots, b_k$, M_j^t is the set of indices of the connected components that remain to be served at the beginning of iteration t. When a connected component C_i is traversed by ant j at iteration t, i is removed from M_j^t. The idea is that an ant decides at each iteration which connected component should be served and how it should be served. These decisions are made by the ant in a probabilistic manner, taking into account the pheromone values associated with each possible way of linking two connected components.

2.1 Algorithm structure

The aim of this section is to present the general structure of the ANTURPP algorithm. It starts constructing an auxiliary multigraph $G^a = (N, A)$ from $G = (V, E)$, using the procedure AUXILIARYGRAPH (Section 2.2.2). Let L be the chain linking together as many as possible required edges belonging to the connected component C_i. Let L be the set of all L_i. This set is determined by the procedure CHAIN (Section 2.2.1). To each chain L_i corresponds a vertex n_i of N. The arc set A is defined as $\{(n_i, n_h) : n_i, n_h \in N, i \neq h\}$ (Section 2.2.2).

Two anthills are located at each vertex of N. An anthill is added at the end vertices of each chain $L_i (i = 1, \ldots, p)$ corresponding to node n_i of the auxiliary multigraph G^a. Therefore, the number of anthill is $|B| = 2p$. Each ant starts from its anthill and moves on G^a to construct a partial URPP solution. The movement of the ants is made in a probabilistic manner and is driven by the pheromone trails updated by an ant only after coming back to its anthill. For each anthill, when all the ants have completed their job, the minimum cost partial URPP solution found by them is selected. The best partial URPP solution is obtained by taking the best partial solution found at each anthill and choosing the least cost one. The output of ANTURPP algorithm is an upper bound obtained adding to the best partial URPP solution the remaining required edges. An anthill is added at the end vertices of each chain $L_i (i = 1, \ldots, p)$ corresponding to node n_i of the auxiliary multigraph G^a. Therefore, the number of anthill is $|B| = 2p$.

At iteration t, ant j of the k^{th} anthill $(k = 1, \ldots, 2p)$ starts from node n_i, and considers all feasible ways of connecting to another node n_h. The ant gradually builds a partial URPP

solution by moving from its anthill to all nodes of G^a, selecting at each step the next chain to be served. The movement of ant j is controlled by procedure MOVE (Section 2.2.4). At iteration t, when all nodes of N have been traversed by ant j, the algorithm returns the partial solution S_j^t. At the end of the algorithm, the set $S(k) = \{S_1, \ldots, S_{b_k}\}$, for each $k \in B$, contains all the partial URPP solutions built by each ant of the k^{th} anthill. Let $F(k) = \{F_1, \ldots, F_{b_k}\}$ be the set of related values of the URPP objective function, $F(k)_{\min} = \min\{F_j : j = 1, \ldots, b_k\}$ is defined as the minimum value of the objective function found by the ants of the k^{th} anthill. Let $\overline{S}(k)$ be the related best partial solution found by the ants of the k^{th} anthill. Then $\overline{S}$ is the best partial solution such that $\overline{F} = \min\{F(k)_{\min} : k = 1, \ldots, 2p\}$. To create a feasible URPP solution, the partial solution $\overline{S}$ is increased by adding all required edges that do not belong to any chain L_i $(i = 1, \ldots, p)$. Let $\overline{E}$ be the set of the required edges belonging to $\overline{S}$. Then the procedure *ADD* [29] returns the feasible solution whose cost $\overline{F}$ represents an upper bound for the URPP.

2.2 Description of the main procedures of ANTURPP

In this section, the procedures outlined in the previous section are described in more detail. A step-by-step description of the ANTURPP algorithm is presented in Figure 1.

1. Call CHAIN(C) (Section 2.2.1).
2. Construct the auxiliary graph G^a (Section 2.2.2).
3. Set the ACO parameters (list of the parameters) (Section 3.1).
4. Initialize
   (a) all the values of Γ to 1;
   (b) all the values of P to the initial probabilities;
   (c) all the values of $\Delta\Gamma$ to 0;
   (d) the set $\overline{S} := \emptyset$ and $\overline{F} := \infty$.
5. Construct the best partial solution $\overline{S}$. For each anthill $k \in B$ and for each ant $j = 1, \ldots, b_k$, call
   (a) MOVE (Section 2.2.4);
   (b) UPDATEPROBABILITY (Section 2.2.3).
6. Determine the best upper bound $\overline{S}$ calling, for each edge $e = (v_u, v_w) \notin \overline{E}$, ADD procedure [29].

Figure 1: Step-by-step description of the ANTURPP algorithm

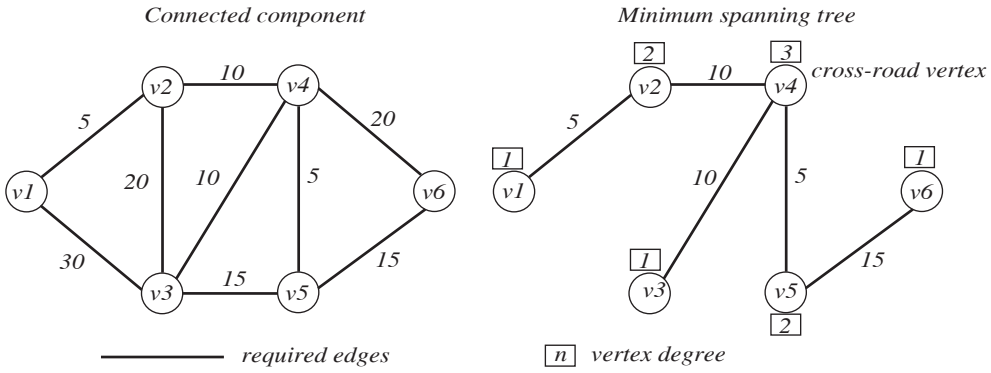


Figure 2: Construction of MST_i^t and the cross-road vertex

2.2.1 Procedure CHAIN(C)

This procedure uses as input the set C of connected components. For each connected component $C_i \in C$, the chain L_i with the maximum number of required edges is built. The *length* of a chain L_i is defined as the number of required edges it contains.

- **Step 1**: For all $v_u \in V_i$ $(i = 1, \ldots, p)$, find the minimum spanning tree MST_u^i starting from v_u. Select the minimum spanning tree MST^i $(i = 1, \ldots, p)$ with the smallest number of odd degree vertices.
- **Step 2**: For all MST^i $(i = 1, \ldots, p)$, find the set of vertices with the maximum degree, called *cross-road vertices*. For each cross-road vertex v_r the two longest disjoint chains, starting from v_r, are built (Figure 2).
- **Step 3**: For all MST^i $(i = 1, \ldots, p)$ and for each cross-road vertex v_r, merge the two disjoint longest chains in v_r and define a set I_i $(i = 1, \ldots, p)$ of chains for each connected component C_i $(i = 1, \ldots, p)$.
- **Step 4**: Select the chain $L_i \in I_i$ $(i = 1, \ldots, p)$ with the maximum length (Figure 3).

2.2.2 Procedure AUXILIARYGRAPH(G, L)

This procedure returns the auxiliary multigraph $G^a = (N, A)$, where $N = \{n_1, \ldots, n_p\}$ is the set of nodes associated with the p longest chains of L, and A is the set of edges associated with all possible ways of reconnecting every pair of chains in L.

Let v_u^i and v_w^i be the extremes vertices of the chain L_i $(i = 1, \ldots, p)$. For each pair of chains $L_i, L_h \in L$, there are four ways of connecting them by vertices $v_u^i, v_w^i \in L_i$ and $v_u^h, v_w^h \in L_h$ (Figure 4):

1. starting from v_u^i and arriving at v_u^h by a shortest chain of length d_{uu}^{ih}(Figure 5a);
2. starting from v_u^i and arriving at v_w^h by a shortest chain of length d_{uw}^{ih} (Figure 5b);

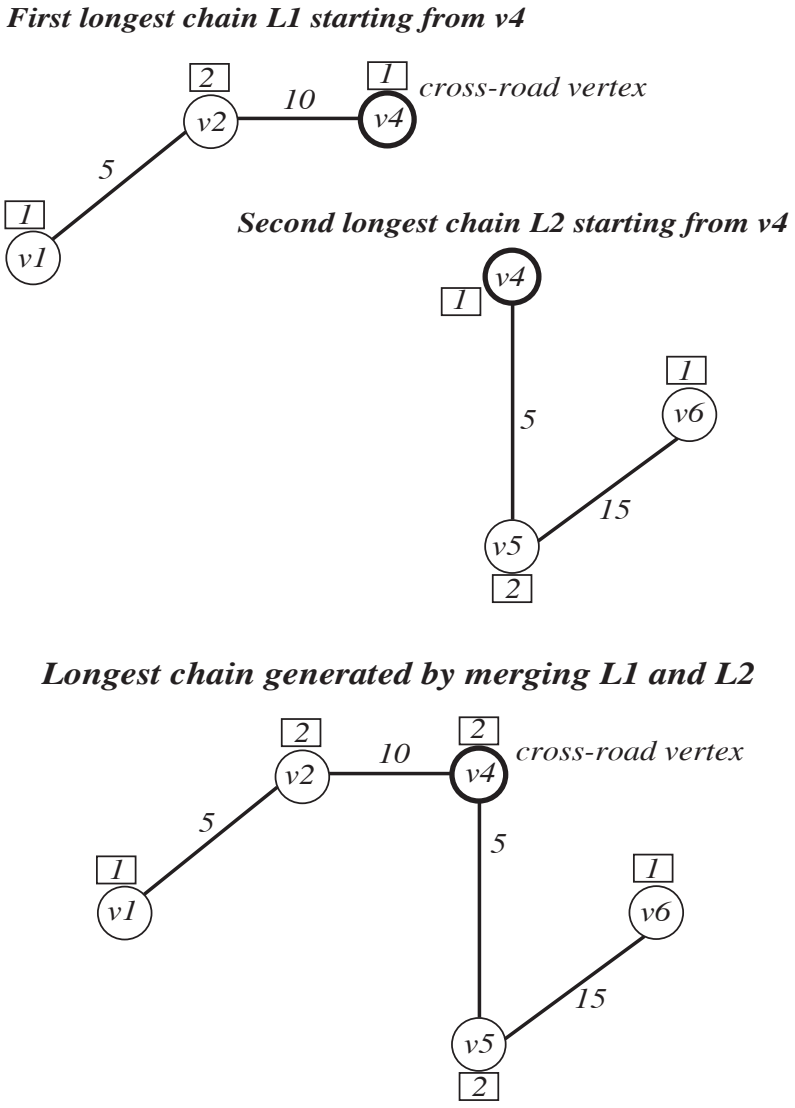


Figure 3: Longest chain L

3. starting from v_w^i and arriving at v_u^h by a shortest chain of length d_{wu}^{ih} (Figure 5c);
4. starting from v_w^i and arriving at v_w^h by a shortest chain of length d_{ww}^{ih} (Figure 5d);

Therefore, there are $2p(p-1)$ connection possibilities .

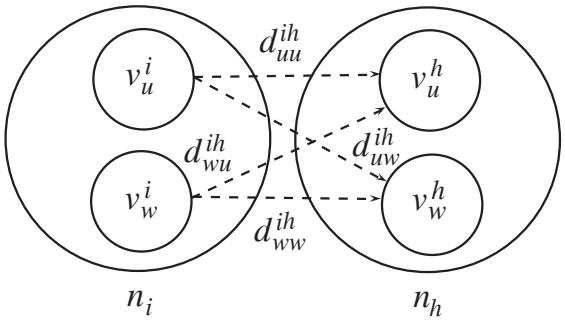
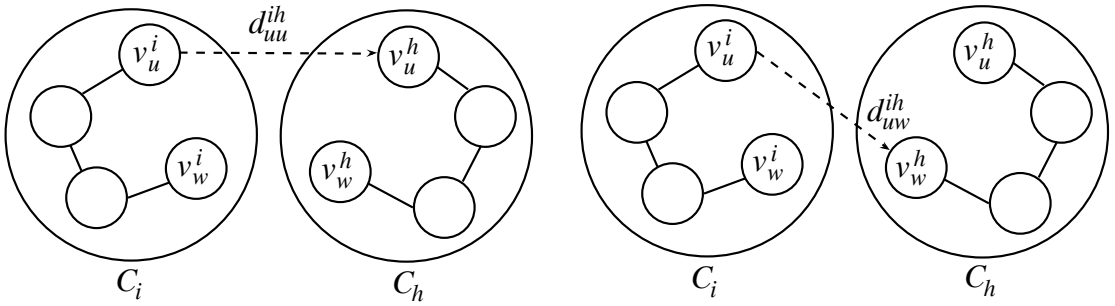


Figure 4: The auxiliary multigraph G^a



(a) First reconnection possibility

(b) Second reconnection possibility

(c) Third reconnection possibility

(d) Fourth reconnection possibility

Figure 5: Four ways of reconnecting two chains

2.2.3 Procedure UPDATEPROBABILITY($G^a, \Gamma, \Delta\Gamma, P$)

The matrix Γ contains the intensity of chains, the matrix $\Delta\Gamma$ is made up by the trails of pheromones deposited on the selected chains, and the matrix P contains the probabilities associated with each edge of G^a. This procedure returns the updated matrix $\overline{P}$.

Let $m = \sum_{k \in B} b_k$ be the total number of ants. At the beginning of each iteration t, an ant j leaves from each anthill k to build a partial solution S_j^t. The ants coming from the two anthills corresponding to the same node $n_i \in N$ visit in the opposite direction the shortest chain L_i of C_i $(i = 1, \ldots, p)$ represented by this node. At the end of iteration t, the ants have constructed $2p$ partial URPP solutions S_j^t $(j = 1, \ldots, 2p)$. They then deposit pheromones on the edge of G^a chosen to build the partial solutions S_j^t $(j = 1, \ldots, 2p)$. On the other edges the pheromone evaporates, and their probability of being selected by the next ants thus decreases. A chain connecting two nodes n_i and n_h of G^a corresponds to an edge e=$(n_i, n_h)$$\in A$. The updating of the pheromone value on edge e is determined by each ant between iterations t and $t+1$. Let $\tau^e(t+1)$ be the *intensity* of the chain generated at the beginning of iteration $t+1$:

$$\tau^e(t+1) = \rho\tau^e(t) + \Delta\tau^e(t, t+1), \tag{1}$$

where ρ is the *evaporation coefficient*, and the values $\Delta\tau^e(t, t+1)$ represent the quantity of pheromones deposited on edge e by the ant at the end of iteration t. The values $\Delta\tau^e(t, t+1)$ are stored in the matrix $\Delta\Gamma$. Let $\tau^e(0)$ be the intensity of the chain associated with edge e at iteration $t = 0$, whose value can be arbitrarily set to 1 in our experiments. The values of $\tau^e(t)$ are stored in a matrix Γ. The ant j, being on a node n_i at iteration t, reads the indices of vertices to be visited from the memory list M_j^t. If M_j^t is not empty, the probability of visiting the other nodes is distributed as follows. Let d^e be the length of the shortest chain associated to the edge e and let $\eta^e = 1/d^e$ be the *visibility of the chain*. The transition probability from vertex $n_i \in N$ to vertex $n_h \in N$ is defined as:

$$p^e(t) = \frac{[\tau^e(t)]^\alpha [\eta^e(t)]^\beta}{\sum_{w \in A: w=(n_i,n_v), v \in M_j^t} [\tau^w(t)]^\alpha [\eta^w(t)]^\beta}. \tag{2}$$

Equation (2) is used if $t \geq 1$, when $2p$ ants have already computed their partial solution. For the ants that leave from the anthills located at node n_i at iteration $t = 0$, the distribution of the transition probability is computed using a formula that attributes a larger probability to serve the closest nodes than the others. It is called the *greedy initial probability*. Therefore, we can write:

$$p^e(0) = \frac{\sum_{w \in A: w=(n_i,n_v), v \in M_j^0} d^w - d^e}{(\sum_{w \in A: w=(n_i,n_v), v \in M_j^0} d^v)(2|M_j^0| - 1)}. \tag{3}$$

It is worth nothing that

$$\sum_{w \in A: w=(n_i,n_v), v \in M_j^0} p^w(0) = 1. \tag{4}$$

The values of the probabilities are stored in the matrix P.

2.2.4 Procedure MOVE($G^a, j, k, M_j^t, S_j^t, \Delta\Gamma$)

This procedure returns the next anthill at node $n_h \in N$, the new memory list M_j^t of ant j, the new current partial URPP solution S_j^t at iteration t generated by ant j starting from anthill k, and the updated matrix $\Delta\Gamma$.

We apply the ACO to connect all vertices of the auxiliary multigraph G^a . This way, a cycle that crosses all connected components of G through the associated longest chains (Figure 6) is generated. This cycle represents a partial URPP solution because some required edges are not included in it. Procedure MOVE implements the single decision that ant j has to make every time it is at node $n_i \in N$ and has to serve another node $n_h \in N$, with $h \in M_j^t$. We observe that the new memory list M_j^t is obtained by eliminating the index corresponding to node n_h. This procedure is called iteratively as long as a partial URPP solution has not been built. Otherwise, M_j^t is empty and the ant j can go back to its anthill.

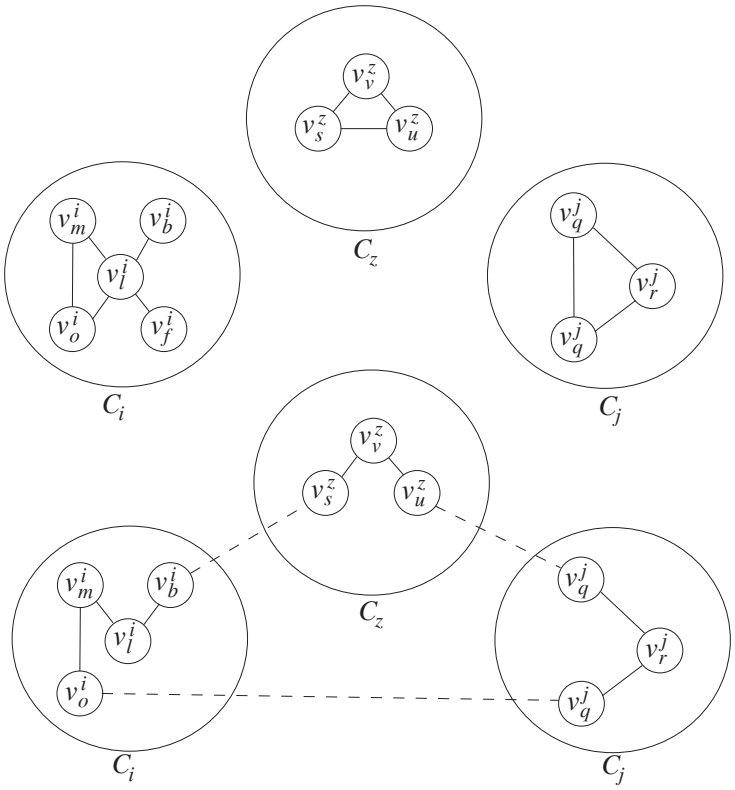


Figure 6: Cycle connecting the longest chains correspondent to the connected components C_i, C_j and C_z

At the beginning the ant is located at its anthill. We have positioned two anthills at each node $n_i \in N$, one for each way of serving the longest chain L_i: from v_u^i to v_w^i and vice versa. The ant is initially forced to serve the required edge corresponding to its anthill in a given direction. For example, suppose that ant j is initially positioned at node $n_i \in N$, corresponding to the longest chain L_i, and suppose it is forced to serve L_i from v_u^i to v_w^i. The ant has to make the following decisions: which longest chain L_h should be served next time, then which node of G^a should be reached next. If the ant decides to serve the longest chain L_h with the shortest chain from v_w^i to v_u^h, it will have to move from v_u^h to v_w^h (Figure 7). Otherwise, if it serves L_h with the shortest chain from v_w^i to v_w^h , it will have to move from v_w^h to v_u^h (Figure 8).

The sequence of operations performed at iteration t is described as follows:

1. Ant j constructs the feasible decisions set D_j^t (Figure 9).
2. Ant j computes the probabilities assigned to each feasible case using the UPDATE-PROBABILITY procedure (Section 2.2.3).
3. Ant j randomly decides to cross the shortest chain $\{v_u^i, \ldots, v_w^h\}$, and updates the memory list $M_j^t = M_j^{t-1} \setminus \{h\}$ and the current partial solution $\overline{S}_j^t = S_j^{t-1} \bigcup \{v_u^i, \ldots, v_w^h\}$. Let d_{ih} be the length of a shortest chain between v_u^i and v_w^h and let Q_1 be a constant quantity of pheromone deposited by ant j on the shortest chain chosen at iteration t, to go from v_u^i to v_w^h.

Let $e = (n_i, n_h) \in A$ be the edge associated to the shortest chain $\{v_u^i, \ldots, v_w^h\}$ on G^a, then it is possible to define

$$\Delta\tau^e(t, t+1) = \begin{cases} Q_1/d_{ih} & \text{if the ant } j \text{ goes from } n_i \text{ to } n_h \text{ between } t \text{ and } t+1 \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

According to (5), the pheromone intensity increases on the shortest chain when ant j goes from n_i to n_h and is inversely proportional to d_{ih}. Let $\overline{\Delta\Gamma}$ be the updated matrix $\Delta\Gamma$ such that the element corresponding to the edge $e = (n_i, n_h)$ is

$$\overline{\Delta\Gamma}^e = \Delta\Gamma^e + \Delta\tau^e(t, t+1). \tag{6}$$

3 ACO parameters

As described in Section 2.2.3, in order to initialize the ANTURPP algorithm, it is necessary to define parameters ρ, α, β and b_k, the number of ants associated to the k^{th} anthill. In our computational study we have tested following feasible ranges for the previous parameters: $b_k \in \overline{B} = \{10, 15, 20, 25, 30, 35, 40, 45, 50, 55\}$, $\rho \in \overline{R} = \{0.5, 0.6, 0.7, 0.8\}$, $\alpha \in \overline{A} = \{1, 1.5, 2, 2.5, 3\}$, and $\beta \in \overline{H} = \{1, 1.5, 2, 2.5, 3\}$.

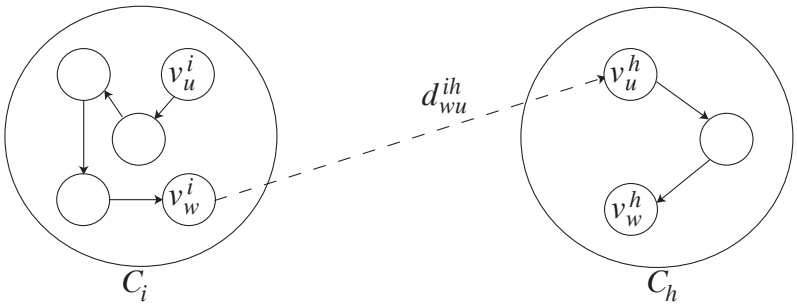


Figure 7: Serving the longest chain L_h with the shortest chain from v_{wi} to v_{uh}

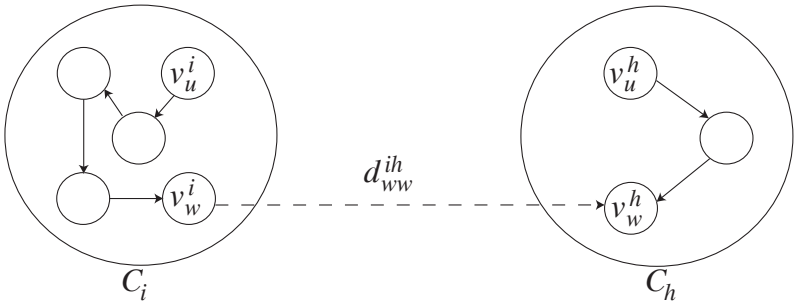


Figure 8: Serving the longest chain L_h with the shortest chain from v_{wi} to v_{wh}

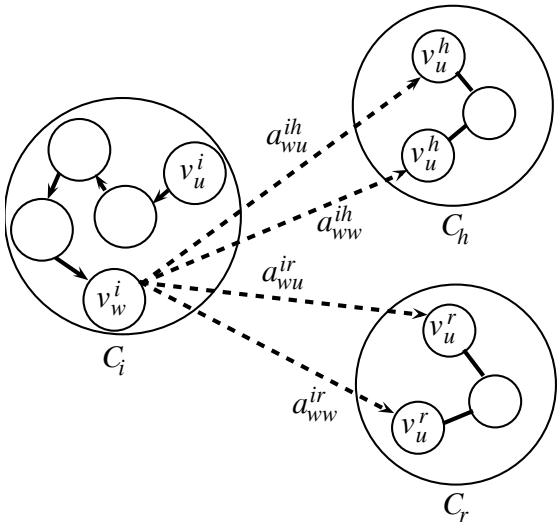


Figure 9: The set of feasible decisions D_j^t for the ant j at iteration t

3.1 F-Race for the ANTURPP algorithm

The F-Race procedure is used to find an optimal setting for the parameters of a meta-heuristic approach to a combinatorial optimization (CO) problem [2]. Let $\theta = \{\theta_1, \ldots, \theta_n\}$ be the set of all possible parameters configurations, $I = \{I_1, \ldots, I_m\}$ the set of the benchmark instances for CO, and $\chi(m, n)$ a matrix whose number of rows is equal to m and the number of columns is equal to n. The heuristic algorithm Υ, used to solve CO runs for a fixed time T on each couple (I_i,θ_j), where $I_i \in I$ and $\theta_j \in \theta$. Each element $\chi_{I_i\theta_j}$ is the best solution found by Υ at time T on instance I_i, using configuration θ_j. The F-Race procedure is able to find the optimal setting for the parameters using a statistical test, starting from data matrix χ.

In our specific case, the set of all possible configurations has cardinality $n = |\theta| = |\overline{B}| \times |\overline{R}| \times |\overline{A}| \times |\overline{H}| = 10 \times 4 \times 5 \times 5 = 1000$, and algorithm Υ is the ANTURPP.

4 Computational results

ANTURPP was implemented in C++ and was executed on a processor Itanium (1 GHz) NEC TX-7. It was tested on two sets of the URPP benchmark instances introduced by Ghiani and Laporte [26]: Type A graphs whose vertices are randomly generated in a plane. To ensure they are connected a specific test is used. In practice, E_R is always disconnected in these graphs. Type C graphs whose vertex degrees are equal to 4 and E_R is disconnected. We have tested our algorithm on 29 instances of type A and 20 of type C.

For each instance, we compared the ANTURPP heuristic with the classical Frederickson algorithm [22] and with the recent constructive insertion heuristic [25]. Moreover, we compared our solutions with those generated by the application of the 2-OPT procedure [29] to the solutions provided by Frederickson and the insertion heuristics. Finally, we compared our solutions with the optimal ones provided by a branch-and-cut algorithm [26]. Computational results are presented in Tables 1 and 2.

The column headings are as follows:

- ***OBJ***: objective function value. In the case of ANTURPP, it is the best value found over ten simulation runs;
- ***SEC***: computational time (in seconds) corresponding to the best solution identified by ANTURPP;
- ***Fred***: solution provided by the Frederickson heuristic [22];
- ***Ins***: solution value given by the Ghiani, Laganá and Musmanno heuristic [25];
- ***Fred2-OPT***: solution value provided by the Frederickson constructive algorithm [22], followed by the 2-OPT improvement procedure of Hertz, Laporte and Nanchen-Hugo [29];

- ***Ins2-OPT***: solution value given by the constructive algorithm of Ghiani, Laganá and Musmanno [25] followed by the 2-OPT improvement procedure of Hertz, Laporte and Nanchen-Hugo [29];
- ***ANTURPP***: the result provided by the ANTURPP algorithm;
- ***BC***: optimal solution value provided by the branch-and-cut algorithm of Ghiani and Laporte [26];
- ***R1***: best solution value provided by the ANTURPP heuristic divided the value provided by the ***Frederickson + 2-OPT*** heuristic;
- ***R2***: best solution value provided by the ANTURPP heuristic divided the value provided by the ***Ins+ 2-OPT*** heuristic;
- ***DEV1*** (%): percentage gap between the best value provided by ANTURPP heuristic and the optimal value.

Our results (Tables 1, 2 and 3) show that we obtain the best performance on eight type A instances out of 29, on 14 type C instances out of 20 and, finally, on 13 Christofides instances out of 23. Regarding the type A instances, we can see that the deviation among the ANTURPP upper bound and the optimal objective value does not exceed 1.67% for

Table 1: Computational results for type A instances

Instance Name	$\|V\|$	$\|R\|$	Fred		Ins		Fred2-OPT		Ins2-OPT		ANTURPP		BC	R1	R2	DEV1
A0500001	50	18	6497	0,09	6379	0,55	**5471**	21,87	**5471**	33,01	5501	0,76	5471	1,01	1,01	0,55
A0500002	50	17	6031	0,08	5903	0,56	**5208**	8,68	5255	13,68	5314	0,85	5164	1,02	1,01	2,90
A0500003	50	13	4689	0,08	4147	0,56	4141	11,20	4141	11,80	**4141**	0,39	4141	1,00	1,00	0,00
A0500004	50	22	6822	0,08	6613	0,39	**6064**	13,02	**6064**	25,93	6349	0,36	6064	1,05	1,05	4,70
A0500005	50	24	7495	0,09	6927	0,81	6905	33,68	6927	20,44	**6846**	2,05	6846	0,99	0,99	0,00
A0800001	80	34	7761	0,11	7818	0,92	7400	42,37	**6924**	99,36	6962	5,02	6911	0,94	1,01	0,74
A0800002	80	34	7029	0,13	7110	0,67	**6639**	45,61	6641	90,35	6820	2,58	6554	1,03	1,03	4,06
A0800003	80	41	10370	0,13	9694	1,36	9231	149,67	9361	91,23	**8814**	6,62	8772	0,95	0,94	0,48
A0800004	80	24	6901	0,09	7157	0,73	**6726**	22,76	7021	37,68	6975	1,97	6726	1,04	0,99	3,70
A0800005	80	20	6101	0,09	5384	0,99	5281	22,31	**5259**	44,98	5263	1,42	5247	1,00	1,00	0,30
A1500001	150	57	10543	0,20	9499	2,34	9270	193,18	**8682**	546,45	9067	65,12	8664	0,98	1,04	4,65
A1500002	150	148	25812	2,4	26961	20,54	25288	3221,27	**25068**	1791,05	27543	41,63	24664	1,09	1,10	11,67
A1500003	150	73	12309	0,28	11590	2,33	11516	260,68	**11098**	446,32	11565	97,72	10898	1,00	1,04	6,12
A1500004	150	46	8853	0,20	8520	1,88	7916	279,79	8158	219,43	**7848**	26,01	7691	0,99	0,96	2,04
A1500005	150	48	9813	0,2	9577	1,97	**8701**	557,87	8745	567,54	8780	29,99	8650	1,01	1,00	1,50
A2000001	200	82	10680	0,39	11444	2,50	**10441**	562,77	10756	778,46	10447	112,20	10092	1,00	0,97	3,52
A2000002	200	108	15726	0,53	16848	2,25	14429	1459,15	**14279**	1238,48	14389	127,13	13823	1,00	1,01	4,09
A2000003	200	104	13568	0,52	14381	4,70	**12763**	1740,32	12839	889,29	13137	170,85	12426	1,03	1,02	5,72
A2000004	200	77	11026	0,42	10947	2,08	9337	1281,58	10027	353,67	**9311**	70,51	9173	1,00	0,93	1,50
A2000005	200	91	13184	0,50	13316	2,69	11726	1824,62	11918	960,02	**11255**	99,16	10635	0,96	0,94	5,83
A2500001	250	105	13562	0,61	13328	4,32	12199	1720,49	**12099**	2782,75	12560	300,60	11864	1,03	1,04	5,87
A2500002	250	69	9893	0,58	9830	3,67	**8961**	654,51	9280	1212,81	9333	119,50	8906	1,04	1,01	4,79
A2500003	250	102	13659	0,63	13866	3,82	12344	1221,66	**12139**	3183,37	12247	391,23	11633	0,99	1,01	5,28
A2500004	250	146	17153	5,89	17006	7,8	**15986**	1826,8	16057	3545,77	16440	311,171	15505	1,03	1,02	6,03
A2500005	250	94	12711	0,59	12798	3,87	**11536**	1065,81	11897	2308,96	11611	359,051	11087	1,01	0,98	4,73
A3000001	300	99	13615	0,95	13663	4,82	12343	1549,66	12202	5462,95	**12140**	586,79	11610	0,98	0,99	4,57
A3000002	300	78	11368	0,92	10602	4,98	9609	1109,44	9348	3010,08	**9311**	321,418	8914	0,97	1,00	4,45
A3000003	300	144	16536	1,11	18205	4,89	**15641**	3168,64	15819	9170,91	15881	1055	15253	1,02	1,00	4,12
A3000005	300	116	15256	1,04	14034	4,73	13235	2101,46	**13197**	1190,22	13616	647	13115	1,03	1,03	3,82

Table 2: Computational results for type C instances

Instance Name	$\|V\|$	$\|R\|$	Fred		Ins		Fred2-OPT		Ins2-OPT		ANTURPP		BC	R1	R2	DEV1
C0500301	50	24	7437	0,09	6801	0,56	6371	25,06	**6043**	63,82	6271	0,76	6043	0,98	1,04	3,77
C0500302	50	30	6708	0,08	7508	0,64	6362	50,27	6362	84,75	**6362**	0,91	6362	1,00	1,00	0,00
C0500303	50	18	5280	0,09	5716	0,64	4443	21,14	4443	94,36	**4443**	0,56	4443	1,00	1,00	0,00
C0500304	50	22	6895	0,09	5725	0,74	5725	17,49	5725	19,39	**5725**	0,92	5725	1,00	1,00	0,00
C0500305	50	22	7517	0,09	6742	0,81	6283	34,20	6283	43,83	**6283**	1,82	6283	1,00	1,00	0,00
C1000301	100	61	10074	0,17	10621	1,27	8995	233,84	9158	515,81	**8974**	11,69	8873	1,00	0,98	1,14
C1000302	100	45	10906	0,13	10297	1,75	8721	200,31	8721	389,53	**8656**	14,84	8578	0,99	0,99	0,91
C1000303	100	60	10874	0,14	11593	1,84	**10184**	241,41	10319	235,74	10194	31,72	9955	1,00	0,99	2,40
C1000304	100	57	10971	0,13	11618	2,13	**9878**	148,24	10289	547,17	10056	37,32	9593	1,02	0,98	4,83
C1000305	100	56	9642	0,13	9444	1,88	8931	236,12	8741	462,53	**8645**	39,16	8492	0,97	0,99	1,80
C1500301	150	75	11159	0,88	10903	6,97	10175	1277,51	10207	1567,89	**9845**	103,56	9762	0,97	0,96	0,85
C1500302	150	86	13665	0,88	14129	7,80	12178	859,80	12456	2638,18	**11812**	112,46	11808	0,97	0,95	0,03
C1500303	150	82	11185	0,82	11936	7,14	9850	961,36	10109	614,45	**9808**	76,24	9624	1,00	0,97	1,91
C1500304	150	75	13619	0,93	13739	7,41	**11565**	1297,28	11685	2220,91	11724	96,89	11326	1,01	1,00	3,51
C1500305	150	88	12758	1,04	13254	6,76	11803	1339,3	11669	2827,34	**11475**	117,97	11353	0,97	0,98	1,07
C2000301	200	89	12577	1,38	12674	9,78	11203	3247,25	11183	2106,94	**10857**	152,86	10818	0,97	0,97	0,36
C2000302	200	116	14372	1,6	14851	9,01	13442	2129,52	13753	3128,83	**12959**	227,72	12736	0,96	0,94	1,75
C2000303	200	107	15524	1,38	16369	9,88	**13490**	5175,85	14122	3621,13	13864	334,97	13330	1,03	0,98	4,01
C2000304	200	113	14109	1,43	14358	8,63	**12666**	6736,62	13026	3453,44	12930	282,327	12417	1,02	0,99	4,13
C2000305	200	108	12922	1,48	12768	10,32	11681	3099,50	12407	3556,87	**11567**	298,13	11511	0,99	0,93	0,49

Table 3: Computational results for Christofides instances

Instance Name	$\|V\|$	$\|R\|$	Fred		Ins		Fred2-OPT		Ins2-OPT		ANTURPP		BC	R1	R2	DEV1
Chr1	11	7	**76**	0	79	0,39	**76**	1,69	**76**	2,05	79	0,04	76	1,04	1,04	3,95
Chr2	14	12	155	0	153	0,00	152	3,08	153	2,50	**152**	0,30	152	1,00	0,99	0,00
Chr3	28	26	105	0	107	0,41	**102**	9,05	107	10,28	117	0,14	102	1,15	1,09	14,71
Chr4	17	22	**84**	0	89	0,30	**84**	6,39	86	9,70	86	0,08	84	1,02	1,00	2,38
Chr5	20	16	130	0	127	0,47	124	9,30	124	5,03	**124**	0,11	124	1,00	1,00	0,00
Chr6	24	20	107	0	121	0,66	102	16,81	104	12,91	**102**	1,42	102	1,00	0,98	0,00
Chr7	23	24	**130**	0	137	0,30	**130**	9,20	**130**	14,42	132	0,10	130	1,02	1,02	1,54
Chr8	17	24	**122**	0	123	0,20	**122**	7,70	**122**	7,09	125	0,06	122	1,02	1,02	2,46
Chr9	14	14	**83**	0	84	0,30	**83**	3,45	**83**	3,75	84	0,04	83	1,01	1,01	1,20
Chr10	12	10	80	0	91	0,39	80	2922,00	80	5,36	**80**	0,17	80	1,00	1,00	0,00
Chr11	9	7	26	0	23	0,30	23	0,77	23	1,09	**23**	0,01	23	1,00	1,00	0,00
Chr12	7	5	22	0	19	0,30	19	0,61	19	0,67	**19**	0,01	19	1,00	1,00	0,00
Chr13	7	4	35	0	35	0,30	35	0,49	35	0,69	**35**	0,01	35	1,00	1,00	0,00
Chr14	28	31	207	0	218	0,56	202	33,27	202	44,67	**202**	0,48	202	1,00	1,00	0,00
Chr15	26	19	445	0	459	0,00	441	23,22	441	17,49	**441**	0,51	441	1,00	1,00	0,00
Chr16	31	34	215	0	207	0,66	205	30,48	203	16,67	**203**	0,74	203	0,99	1,00	0,00
Chr17	19	17	116	0	117	0,48	112	7,44	112	7,06	**112**	0,15	112	1,00	1,00	0,00
Chr19	33	29	274	0	274	0,66	**257**	29,73	274	15,52	258	0,55	257	1,00	0,94	0,39
Chr20	50	63	402	0	408	0,67	400	58,97	400	103,89	**398**	2,07	398	1,00	1,00	0,00
Chr21	49	67	372	0	393	0,58	366	77,31	378	58,86	**366**	1,82	366	1,00	0,97	0,00
Chr22	50	74	633	0	669	0,59	622	190,63	647	151,70	626	2,54	621	1,01	0,97	0,81
Chr23	50	78	479	0	489	0,58	**475**	78,33	480	139,09	484	3,29	475	1,02	1,01	1,89
Chr24	41	55	411	0	413	0,66	**405**	60,16	**405**	50,56	417	1,64	405	1,03	1,03	2,96

type A instances, and 5.17% for type C instances. For both instance types, our algorithm requires a very low CPU time to identify the final solution compared with other ones. On the Christofides instances, we can see that, except for the instance Chr3, the maximum

deviation among the ANTURPP solution and the optimal value is about 3.95%. The instance Chr3 is the only one in which the optimality gap is high. After examining the results related to this instance, we can deduce that the high optimality gap is due to a *stagnation of the search*. The reason could be related to the structure of this instance and to the use of an ACO algorithm which, contrary to a MAx Min Ant System (MMAS, [35]), does not consider a limit on the amount of pheromone. In other words, this high gap could be related to the fact that the best solution, found by the ants, with a high optimality gap, has a high pheromone concentration. This concentration could tend to increase itself and could force the ants to build the same solution. It is important to observe that this problem occurs only on this instance, because, with the other, the ANTURPP is able to find good solutions.

Therefore the ANTURPP algorithm obtains, within a reasonable computational time, a tight upper bound, in particular with the instances defined on a large number of required edges. Moreover, we have shown that the proposed method outperforms all existing URPP algorithms in terms of computational times.

References

[1] A.A. Assad and B.L. Golden. Arc routing methods and applications. In *Network Routing*, C.L. Monma M.O. Ball, T.L. Magnanti and G.L. Nemhauser, editors, Volume 8, *Handbook of Operations Research and Management Science*, North-Holland, Amsterdam, pages 375–483, 1995.

[2] M. Birattari. *The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective*. Ph.D. thesis, Université Libre de Bruxelles, 2004.

[3] M. Birattari, T. Stützle, L. Paquete, and L. Varrentrapp. A racing algorithm for configuring metaheuristics. In W.B. Langdon, E. Cant-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18, 2002.

[4] B. Bullnheimer, G. Kotsis, and C. Strauss. A new rank-based version of the ant system: A computational study. *Central European Journal for Operations Research and Economics*, 7:25–38, 1999.

[5] N. Christofides, V. Campos, A. Corberán, and E. Mota. An algorithm for the rural postman problem. *Imperial College Report IC.O.R.81.5*, 1981.

[6] A. Corberán and J. M. Sanchis. A polyhedral approach to the rural postman problem. *European Journal of Operational Research*, 79:95–114, 1994.

[7] O. Cordón, I. Fernández de Viana, F. Herrera, and L. Moreno. A new ACO model integrating evolutionary computation concepts: The best-worst ant system. In M. Dorigo, M. Middendorf, and T. Stützle, editors, *Proceedings of ANTS2000-From Ant Colonies to Artificial Ants*, pages 22–29, 2000.

[8] O. Cordón, F. Herrera, and T. Stützle. A review on the ant colony optimization metaheuristic: Basis, models and new trends. *Mathware and Soft Computing*, 9(2-3):141–175, 2002.

[9] K.F. Doerner, M. Gronalt, R.F. Hartl, M. Reimann, C. Strauss, and M. Stummer. Savings ants for the vehicle routing problem. In E. Hart M. Middendorf G.R. Raidl S. Cagnoni, J. Gottlieb, editors, *Applications of Evolutionary Computing*, pages 11–20. Springer-Verlag, Berlin, 2002.

[10] K.F. Doerner, R.F. Hartl, V. Maniezzo, and M. Reimann. Applying ant colony optimization to the capacitated arc routing problem. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, and F. Mondada, editors, *Ant Colony Optimization and Swarm Intelligence: 4th International Workshop, ANTS 2004*, pages 420–421. Springer-Verlag, Berlin, 2004.

[11] M. Dorigo. *Optimization, Learning and Natural Algorithms.* Ph.D. thesis, Politecnico di Milano, 1992.

[12] M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, Englewood Dliffs, NJ, 1999.

[13] M. Dorigo, G. Di Caro, and L.M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5:137–172, 1999.

[14] M. Dorigo and L.M. Gambardella. Ant colony system: A cooperative learning approach to the travelling salesman problem. *Transaction on Evolutionary Computation*, 5:53–66, 1997.

[15] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics*, Part B, 26:1–13, 1996.

[16] M. Dorigo and T. Stützle. The ant colony optimization metaheuristic: Algorithms, applications and advances. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of International Series in Operations Research and Management Science, pages 251–285. 2002.

[17] M. Dorigo and T. Stützle. *Ant colony optimization.* The MIT Press,Cambridge, MA, 2004.

[18] J. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards*, 69:125–130, 1965.

[19] H.A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, part 1: The Chinese postman problem. *Operations Research*, 43:231–242, 1995a.

[20] H.A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, part 2: The rural postman problem. *Operations Research*, 43:399–414, 1995b.

[21] E. Fernández, R. Garfinkel, O. Meza, and M. Ortega. On the undirected rural postman problem: Tight bounds based on a new formulation. *Operations Research*, 51:281–291, 2003.

[22] G.N. Frederickson. Approximation algorithms for some postman problems. *SIAM Journal on Computing*, 7:178–193, 1979.

[23] L. M. Gambardella, E. Taillard, and G. Agazzi. Macs-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 63–76. McGraw-Hill, Englewood Cliffs, NJ, 1999.

[24] G. Ghiani and G. Improta. The laser-plotter beam routing problem. *Journal of the Operational Research Society*, 52:945–951, 2001.

[25] G. Ghiani, D. Laganá, and R. Musmanno. A constructive heuristic for the undirected rural postman problem. *Computers & Operations Research*, 33:3450–3457, 2006.

[26] G. Ghiani and G. Laporte. A branch-and-cut algorithm for the undirected rural postman problem. *Mathematical Programming*, 87:467–481, 2000.

[27] M. Grötschel, M. Jünger, and G. Reinelt. Optimal control of plotting and drilling machines: A case study... *Operations Research*, 35:61–84, 1991.

[28] G.W. Groves and J.H. van Vunsen. Efficient heuristics for the rural postman problem. *ORiON*, 21:33–51, 2005.

[29] A. Hertz, G. Laporte, and P. Nanchen-Hugo. Improvement procedures for the undirected rural postman problem... *INFORMS Journal on Computing*, 11:53–62, 1999.

[30] P. Lacomme, C. Prins, and A. Tanguy. First competitive ant colony scheme for the CARP. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, and T. Stützle, editors, *Ant Colony Optimization and Swarm Intelligence: 4th International Workshop, ANTS 2004*, pages 426–427. Springer-Verlag, Berlin, 2004.

[31] J. K. Lenstra and A. H. G. Rinnooy Kan. On general routing problems. *Networks*, 6:273–280, 1976.

[32] A. N. Letchford. New inequalities for the general routing problem. *European Journal of Operational Research*, 96:317–322, 1997.

[33] C.S. Orloff. A fundamental problem in vehicle routing. *Networks*, 4:35–64, 1974.

[34] W. L. Pearn and T. C. Wu. Algorithms for the rural postman problem. *Computers & Operations Research*, 22:819–828, 1995.

[35] T. Stützle and H. H. Hoos. Max-min ant system. *Future Generation Computer Systems*, 16:889–914, 2000.