Revista Brasileira de Computação Aplicada, April, 2020

DOI: 10.5335/rbca.v12i1.9317
Vol. 12, Nº 1, pp. 44–53
Homepage: seer.upf.br/index.php/rbca/index

ISSN 2176-6649

ORIGINAL PAPER

Routing of vehicles in the delivery/collection problems-Application of a modified Ant Colony Algorithm

Victor Hugo Resende Lima [1], Elias de Oliveira Lima[1], Hassan Sherafat[1]

[1]Federal University of Sergipe

*vhugoreslim@gmail.com; elias.sl.jr@gmail.com; sherafat.hassan@gmail.com

Received: 2019-04-09. **Revised:** 2020-03-10. **Accepted:** 2020-03-23.

Abstract

Delivering and collecting problems concern to situations where goods are delivered (or collected) in practical cases. For example, solid waste collection, postal services and snow removing. They can be modelled as the well-known Chinese Postman Problem on mixed graphs (MCPP). The MCPP is a fair model for delivering and collecting problems because its goal is to cover all links of a mixed graph with minimal cost. The objective of this paper is to develop an algorithm based on Ant Colony Optimization (ACO) and apply it to MCPP solution. The MCPP is initially converted into an equivalent Travelling Salesman Problem (TSP) and then tackled on this second instance. According to our knowledge, this approach for MCPP solution is the first one in literature.

Keywords: Arc Routing Problems; Chinese Postman Problem; Metaheuristic; Asymmetric Traveling Salesman Problem.

Resumo

Os problemas de entrega e coleta tratam das situações onde objetos são entregues (ou coletados) em casos práticos. Por exemplo, coleta de lixo sólido, serviços postais e remoção de neve. Eles podem ser modelados como o conhecido Problema do Carteiro Chinês em grafos mistos (PCCM). O PCCM é um modelo justo para problemas de entrega e coleta, pois seu objetivo é cobrir todos os *links* de um grafo misto com um custo mínimo. O objetivo deste artigo é desenvolver um algoritmo baseado na Otimização por Colônia de Formigas (OCF) e aplicá-lo para solução do PCCM. O PCCM é inicialmente convertido em um Problema do Caixeiro Viajante (PCV) equivalente e então resolvido para esta segunda instância. Segundo o nosso conhecimento, essa abordagem para solução do PCCM é a primeira na literatura.

Palavras-Chave: Problema de Roteamento de Arcos; Problema do Carteiro Chinês; Problema do Caixeiro Viajante Assimétrico; Metaheurística.

1 Introduction

Several routing problems model practical problems for cost reduction, such as, garbage collection (Golden et al., 2017), street cleaning planning (Bodin and Kursh, 1979) and ice or snow removal (Eglese, 1994). This set of problems looks for circuits, paths or subsets of circuits where all of them are paths on street mesh. They are belong to set of problems known as Arc Routing Problems (ARP).

ARP is the set of models that deal with street routing on urban networks. Mathematically, given a constraint set on the arcs, the problem concerns to find the feasible optimal solution. The optimal solution means an optimal sequence of covering on the arcs. Obligatory covering over all arcs is a arc constraint, for instance. Delivering and collecting problems such as garbage collection, production planning, postal services and accessing mutual information channels, as shown by Zhao et al. (2010), are ARP applications. For reader, Shafani and Haghani (2015) provide a brief introduction on ARP's. They report the two classes of routing

problems (Arc Routing Problems and Node Routing Problems), as well as some applications.

Within ARP, Chinese Postman Problem (CPP) translates some everyday problems into mathematical environment. CPP is defined as a problem in which a route must be created. It leaves from initial node, which must coincide with the end point of the route, passing through all the edges at a minimal cost. CPP variations are diverse, but the focus of this study is on the mixed variation, called Mixed Chinese Postman Problem (MCPP).

The MCPP is a fair model for these problems, because the urban meshes present directed and not directed streets, namely one-way and two-way streets. Regarding to MCPP, the literature does not show many solution attempts to this problem. Other problems, like the Travelling Salesman Problem (TSP), it have more attempts than MCPP. However, studies based on MCPP problems have practical advantages. For example, Sherafat (2013) reaches a traveled distance reduction of 5.8km for a collecting vehicle responsible by trash collecting in a neighborhood in Aracaju city.

The problems mentioned above (MCPP and TSP) are solvable in a non-polynomial way. That is, algorithms that guarantee the optimal solution are processed in non-polynomial time (the solution set grows in a non-polynomial way according to the increase of the variables). So, depending on problem size, solve it is not possible in a feasible processing time. Therefore, for many practical cases, the problem resolution should be done through an approximate approach, such as the metaheuristics.

We decided to solve the MCPP transforming it into a TSP instance and applying the Ant Colony Optimization (ACO) on this instance. Briefly, the ACO is a metaheuristic bio-inspired in ant behaviour for food search. They use a biological factor called pheromone for efficiency improvement in this process. Namely, our goal is to solve the MCPP instances using a polynomial transformation to TSP, through an ACO algorithm. On the best of our knowledge, this approach is the first one in literature.

This paper is structured as following: section 2 explains the two problems that are treated in this study and their variations. Section 3 shows the metaheuristics for TSP and MCPP. Section 4 formally presents the ACO and the proposed algorithm is explained in Section 5. Section 6 shows the computational results and finally, section 7 shows the paper conclusion.

2 Chinese Postman and Travelling Salesman Problems

Chinese Postman Problem (CPP) was initially devised by Mei-Ko (1962) when analyzing the work of mailmen in China. Its definition is described in a graph G=(V,E), where V is the set of vertices and E the set of edges. Each edge has a positive associated cost. The goal is to find a closed circuit with lower cost, in which all edges must be visited only once. From its idealization to nowadays, CPP gained some variations. Eiselt et al. (1995) cites some of them, such as Chinese Postman with Wind (CPW), Mixed Chinese Postman Problem (MCPP) and Chinese Hierarchical Postman (CHP). CPP and its variations have several applications in everyday problems, such as vehicle routing problems, mailmen problems and the street maintenance planning (Thimbleby, 2003).

Initially, the problem was formulated considering only edges (non-oriented case) and it was solved by Edmonds and Johnson (1973) with a matching algorithm to solve the node parity of the problem. When modeled only with arcs (oriented case), it also can be solved with the same algorithm after making some adaptations that become it in a minimum flow problem. Both totally versions (oriented and non-oriented) of the problem are of polynomial order. So, they are well solved. However, the variation corresponding to the MCPP is an NP-Hard problem (Papadimitriou, 1976).

As previously explained, MCPP is a problem little addressed in the literature, when compared to other problems, such as TSP. Thus, it was decided to use an MCPP to TSP transformation, preserving the feasibility and equivalence of solutions. This type of transformation has been studied. In one brief review, we can cite three works. First, Laporte (1997) creates a general transformation, translating many ARP problems into equivalent TSP's. Posteriorly, Sherafat (2004) makes a wide transformation based on four phases. Lastly, Gordenko and Avdoshin (2017) propose a MCPP to Generalized TSP (GTSP) transformation. After this first transformation, the GTSP is transformed into an correspondent Asymmetric TSP (ATSP). In few words, the ATSP is the TSP which the edge costs in both directions are not equals. All cited procedures generate a TSP with |A|+2|E| nodes, where A is the set of arcs and E the set of edges. Namely, an arc is denoted as link that has only one direction. Otherwise, edges can be traveled in two directions. Finally, note that the TSP is also NP-Hard (Laporte, 1992). Therefore, this change just concerns computational implementation. Here we used the Sherafat (2004) transformation, but any transformation could be used.

About TSP, it is a problem that belongs to NRP class. Its worry is to find out a closed path that must pass on every node only once, with minimal cost. The mathematical formulation for TSP on directed graph is given as follows:

$$Min = \sum_{i \in V, j \in V} x_{ij}.c_{ij} \tag{1}$$

$$S.T.: \sum_{i=1}^{|V|} x_{ij} = 1, \ \forall j \in V \tag{2}$$

$$\sum_{j=1}^{|V|} x_{ij} = 1, \ \forall i \in V \tag{3}$$

$$\sum_{i \in S, j \in S} x_{ij} \leq |S| - 1, \ (S \subset V, |V| - 2 \geq |S| \geq 2) \qquad (4)$$

$$x_{ij} = \{0, 1\}, \ \forall \, i, j \in V. \qquad (5)$$

Above, V is the set of nodes, S is a subset of V and (i,j) is an arc. The objective function of the linear problem (Eq. (1)) is to minimize the total cost of the route performed. Constraints (Eqs. (2) and (3)) force all nodes to be visited only once. In addition, they guarantee that the solution is a closed circuit (initial node equal to final node), because when all nodes are visited, the returning to the initial node is occurs. Constraint (Eq. (4)) eliminates any subcircuit that can be created. Finally, last constraint refers to trivial condition to binary variables.

Some algorithms (Grötschel and Win, 1992, Sherafat, 1988, Nobert and Picard, 1996) try to solve CPP variations and achieve satisfactory results in contrast to needed computational resources. They are exact algorithms and has high computational demand for large instances, that is, the number of steps in their execution grows in a non-polynomial way. Namely, the increase of the problem variables increases the space solution of the problem in non-polynomial way. Thus, the choice by attempting based on approximate algorithms is valid, mainly for large instances. On the other hand, exact algorithms present an universe of formulations well explored and none significant advance for the reduction of computational time was reached in recent years.

Therefore, a good alternative today is the use of approximate methods, which present a good number of algorithms and approach possibilities. These methods require less computational effort and could reach the optimal solution.

3 Metaheuristics for TSP and MCPP

Approximate algorithms were very important for the development of techniques in Combinatorial Optimization. There is a division into two groups for approximate methods: heuristics and metaheuristics.

Heuristics are methods based on the deterministic choices for a set of possibilities in each step of the procedure. So, they took the best decision for current step, regarding to some approximate rule. Some applications of heuristic procedures for TSP (and some its variations) can be seen in Karabulut and Tasgetiren (2014), Mestria et al. (2013) and Monkman et al. (2008).

On the other hand, metaheuristics are methods whose decisions are made in a stochastic way, using the information of the faced problem. Furthermore, they are bio-inspired methods. The use of particular mechanisms by methods provides the possibility of escape from local optimums. This premature "prision" is a factor limiting to the efficiency of heuristics. So, metaheuristics have been the best choice for solution of routing problems. Thus, several algorithms based on metaheuristic classes have been proposed and presented good results. We can cite Genetic Algorithms with several types of crossover (Ahmed, 2010, Cheng and Gen, 1994), Ant Colony algorithms (Prakasam and Savarimuthu, 2015) and Ant Colony Algorithms with improvements to reach better location earlier, using the calculus concept (Saenphon et al., 2014).

Metaheuristics present a vast number of algorithms. For instance, Genetic Algorithm (H. Holland, 1984), Simulated Annealing (Kirkpatrick et al., 1983) and Ant Colony Optimization (Dorigo, 1992), Teaching-Learning-Based Optimization (Rao et al., 2011), Particle Swarm Optimization (Kennedy and Eberhart, 1995), Bat Algorithm (Yang, 2010b), Harmony Search (Yang, 2010a) and Social Network Optimization (Sherafat, 2017). The state-of-art in metaheuristics for both MCPP and ATSP problems are, for the bibliographic research done here, the GRASP method tested by Corberán et al. (2002) and Genetic Algorithm by Nagata and Soler (2012), respectively.

With respect to metaheuristics for the MCPP, we listed just two works. The first is the state-of-art commented previously (Corberán et al., 2002), which use a GRASP metaheuristic, based on two phase: construction phase and local search phase. The basic procedure used by authors is to orient edges in one direction (transforming it in one arc) and to solve this oriented CPP optimally, since oriented CPP can be optimally solved. So, each edge is oriented in turn, observing the flow for the two nodes associated to edge. At end, with all edges oriented, the minimum cost flow problem associated to this directed graph is solved. In improvement phase, the operation is based on deletion of any two copies in any edge that appears more than twice in current solution with contrary orientation. With this path deletion, one new minimal cost path is performed between these two nodes. Worth remember that just some pair nodes are selected in each phase.

Jiang et al. (2010) describe a Genetic Algorithm for MCPP. They proposed mechanisms that always generate a feasible solution. The authors use a crossover operator based on swaps of one link (arc or edge) copy, effecting other operations for the maintenance of feasibility. Concerning to the mutation are used two operators. The first one deletes one path which contains only copies of links and finds another better path. The other operator makes a change in direction of an edge and creates other two paths between the nodes of edge to keep the flow balancing. Finally, there is a redundancy elimination phase for circles removing. The instances tested are small, but the proposed algorithm presents better results than a heuristic used for comparison.

Differently from MCPP, the TSP has many metaheuristics developed for its solution. For example, Osaba et al. (2016) propose a Improved Bat Algorithm for ATSP and TSP resolutions. The authors tested instances from TSPLIB (http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html) and reached results better than two basic Bat Algorithms implemented for them. In addition, comparisons to five

different metaheuristics were done and the results are comparable. In a new metaheuristic introduced, the Discrete Bacterial Memetic Evolutionary Algorithm, it was tested the TSP and TSP with Time Windows (TSPTW) instances. In few words, the procedure are not able surpass the state-of-art of heuristics for the TSP, but achieves good and more predictable results (Kóczy et al., 2017). A Discrete Symbiotic Organisms Search algorithm is proposed by Ezugwu and Adewumi (2017). Namely, this metaheurstic is based on dependency-relations between organisms in nature. It used three symbiosis phases: mutualism, commensalism and parasitism. In each one of them, one specific interrelation between organisms is made. Firstly, they made comparisons between the new algorithm and a basic Symbiotic Organisms Search algorithm. Posteriorly, four metaheuristics were compared. Instances from TSPLIB were used and the achieved results indicated that the algorithm is fairly comparable to only one algorithm. It outperforms all others methods.

Finally, Taillard and Helsgaun (2019) use the POPMUSIC metaheuristic to handle large TSP's (some instances have 10 million of customers). This metaheuristic operates based on sub-parts of the optimization problem. These optimizations are repeated until that none improvement can be found. Choong et al. (2019) proposed an Artificial Bee Colony with a Modified Choice Function. Basically, they proposed a hyper-heuristic inserted into the Artificial Bee Colony. Hyper-heuristics are methods that select other heuristics or create new heuristics for solution space searches. They use ten neighborhood searches, the majority based on inserting, reversing, swap and shuffle. The selection heuristic is updated along algorithm execution, such that intensification and diversification are dynamically balanced through scores for every heuristic. The Modified Choice Function has its efficiency proven through comparisons with "standard" Artificial Bee procedures. Finally, the algorithm is compared to ten state-of-art algorithms, the majority based on Artificial Bee and Ant Colony optimizations. The authors done Wilcoxon tests and prove that their algorithm surpass the majority of the compared algorithms.

4 Ant Colony Optimization

Ant Colony Optimization (ACO) was formulated and tested by Dorigo (1992). It is based on the ant behaviour for food search. Its differential mechanism is the communication among ants, which is done through a substance called pheromone, deposited by themselves while walk. For food search, ants explore the space around the anthill and leave pheromone along their paths. Ants that coming out from anthill after some of them find food, tends to find food more easily. This happens because the succeeded ants "mapped" the path to the food through the largest deposit of pheromone (departing and returning). This procedure is a reinforcement process over the time, until that some better path is found, "beginning" a new process.

In the first application of the ACO for the TSP, the general procedure for tours generation, according to Dorigo et al. (1996), followed the methodology below:

- Each ant decide its next step based on a probability function, which depends on the distance between the current node and the possible nodes to be visited and the pheromone quantity;
- Each ant must create a feasible solution, in other words, every node must visited only once;
- After all tours completing, the pheromone upgrade is done.

The decisions made by ants are according to a pseudo-probabilistic decision rule. This rule is denoted by Eq. (6) below.

$$P_{ij}^{k}(t) = \begin{cases} \frac{\tau_{ij}(t)^{\alpha}.\eta_{ij}^{\beta}}{\sum \tau_{ij}(t)^{\alpha}.\eta_{ij}^{\beta}}, \textit{if } j \in \textit{PossibleNodes} \\ 0, \textit{ otherwise}, \end{cases} \tag{6}$$

where, η is the inverse of distance between the two nodes, PossibleNodes set contains the nodes still not visited and α and β are the weights for pheromone and distance, respectively.

With all tours generated, the next step is pheromone updating. This phase is denoted by Eqs. (7) to (9). Therefore, the pheromone updating follows the equation below.

$$\tau_{ij}(t) := \rho.\tau_{ij}(t-1) + \Delta\tau_{ij}. \tag{7}$$

Above, ρ is the value of evaporation rate between iterations t-1 and t, τ_{ij} is the quantity of pheromone on stretch (i,j) and $\Delta\tau_{ij}$ is the value of deposit of chemical substance. This term is calculated by Eq. (8), which is:

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \tag{8}$$

where m is the total number of ants and $\Delta\tau_{ij}^{k}$ is the sum of all pheromones left on the stretch (i,j) by all m ants between the iterations t-1 and t. Finally, each single deposit is calculated following the Equation 9.

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k}, \ \forall\ (i,j) \in \textit{Tour}_k(t-1,t) \\ 0, \ \forall\ (i,j) \notin \textit{Tour}_k(t-1,t). \end{cases} \tag{9}$$

In the Eq. (9), the term Q is the deposit constant and L_k is the tour size of ant k.

There are many variations for ACO algorithms. Some of them are presented in Stüzle and Linke (2002). Among all variations, Elitist Strategy Ants System and Ranked Elitist Optimization deserve mention. In both, the best ants are encouraged to deposit more

pheromone than those considered worse (Bullnheimer et al., 1997). In a new approach, Stüzle and Hoos (2000) propose that: just the best ant deposits pheromone in each iteration; the pheromone obeys two limits (maximum and minimum) and in algorithm initialization, pheromone for all paths has its value equal to maximum limit.

Alignments between metaheuristics and other procedures has been created. These new procedures are called hybridizations. Euchi et al. (2015), for example, used the ACO with an enhancement module based on the 2-opt heuristic applied to Vehicle Routing Problem. For hybridization between metaheuristics, we can mention the work of Xiao and Jiang-qing (2012), which propose a hybridization with mutation, 2-opt and a Nearest Neighbor. Finally, Dong et al. (2012) implements a hybridization between the ACO and the Genetic Algorithm. For more recent hybridization we can cite Khan et al. (2017), Unold and Tarnawski (2016), Yan et al. (2017), Ismkhan (2017) and Mahi et al. (2015).

Treating to TSP applications, some works can be discussed. Eldem and Ülker (2017) made a Max-Min Ant System algorithm to tackle the Symmetric TSP denoted on a 3D sphere, where the nodes are on shell. They use a Genetic Algorithm and Cuckoo Search Algorithm for comparisons and both are surpassed by the proposed algorithm. In other work, Zalilah (2015) ally an Ant Colony Algorithm with hyper-heuristics. The majority are based on 2-opt and are computationally cheapest. They tested small instances (between 30 and 100 nodes) and compared their results with others seven classic algorithms. The conclusion is that, indeed, the method is comparable.

Specifically for ATSP we cited three works. Firstly, Gambardella and Dorigo (1996) present an Ant Colony System none locals searches. The instances used for tests are from TSPLIB and have up to 170 nodes. In other Ant Colony System, Dorigo and Gambardella (1997) test Symmetric and Asymmetric TSP instances. Furthermore, they use some methods for comparison. Both works use the reduced candidate list, encouraged for large instances. Finally, Bai et al. (2013) make a hybridization between exact methods and a Max-Min Ant Colony Optimization. The comparisons are done with state-of-art of some algorithms. The achieved results are clearly better than those provided by others methods, for both computational time and final solution.

4.1 Proposed algorithm

In this study we used a pure approach instead of some hybridization. This new algorithm is based on an elitist strategy, following a variant of ACO methodology. The general procedure of the algorithm is as follows:

- Ant initial positions are drawn for the quantity m of ants. In each round of draws ants are putted in different positions from the others ants until that, at round, all nodes have one ant. In this case a new round is started. For example: given G=(V,A) the graph of the problem, where V=(1,2,3,4,5) and the number of ants is equal to 10 ants, then a possible positioning order can be (1,4,5,3,2,2,4,3,1,5) and an infeasible order would be (1,2,2,4,3,5,4,3,2,1). In the infeasible positioning, the first round (5 first ants), contains two ants located in the same node (the second and third ants are located in node 2). This mechanism avoids the biased decisions;
- Routing probabilities are calculated for nodes still not visited. With the probabilities calculated, a number between 0 and 1 is drawn and the interval of the cumulative probability is matched. This drawing scheme follows the logic of roulette wheel selection, where the draw spaces are not symmetrical;
- At the end of each tour performed for all ants, one iteration is counted and the pheromone is updated. This updating is made firstly by evaporation phase, where all quantities for any arc is decreased to ρ percentage of its quantity. With each ant cost computed, the deposit constant Q is denoted as the average cost of all ants at iteration. The ant deposit is then defined as $\sigma.Q/L_k$, where σ is the elitist constant and L_k is the route cost of the ant k. So, the best ant deposits σ times more pheromone than the worst ant at iteration.

In third step of the procedure above, during the updating, a rudimentary mechanism was imposed to increase the initial diversity for the searches. During the first ten iterations, the evaporation rate was set to 90%, whereas for remain iterations its value was maintained according to the selected value.

The pseudo-code of the proposed algorithm is shown below:

```
1   Read all parameters;
2   Initialize the adjacency matrix for the pheromone;
3   Initialize PossibleNodes with all nodes of graph;
4   For i:=1 to m do
5    Drawn the ant k in PossibleNodes;
6    if (PossibleNodes=EmptySet) then
7     Initialize PossibleNodes again;
8   While (CurrentIteration<=T)
9    For i:=1 to m do
10    For j:=1 to N do //N is the nodes number of the TSP
11     Calculate probabilities following the Equation 6;
12     Fill the roulette with these probabilities;
13     Turn roulette;
14     Go to the node drawn;
15    Associate a number to EvaporationRate, following the rule:
16     if (CurrentIteration<10) then
17      EvaporationRate:=0.9
18     else
19      EvaporationRate:=Desired Value;
20    Deposit pheromone according to the rank of the ants;
21    Count one iteration;
```

Figure 1: Pseudocode of the new algorithm

5 Computational tests

Algorithm experiments were done in a personal computer with Intel Core i5 1.7GHz processor and 8GB of RAM desktop. Tested graphs were generated and classified into 5 groups, where each group is formed

by 4 or 5 instances of pseudo-manhanttan type. The graphs were grouped according to the number of nodes in the initial problem (MCPP). Within each of these groups, the graph features are changed. In each one of them, the percentage of arcs is varied between 0% and 100%. The average cost for these links ranged from 20-30.

Before the performance test, the algorithm underwent a calibration process of the parameters (α, β, ρ, T and m) for two graphs of different sizes. Table 1 shows the values for the five parameters, the number of nodes in the original problem and in the transformed problem. These graphs are generals, such that, they were not used for the performance test.

Table 1: Results for parameters during calibration

Number of nodes in MCPP	Number of nodes in TSP	α	β	ρ(%)	T	m
120	353	1	7	20	100	300
300	890	2	7	25	200	300

The number of ants and iterations, in the calibration instances presented higher values. However, for the performance tests, these parameters were decreased, because the computational time increased considerably. So, the computational time never exceeded 900s of processing in our tests. For each instance, 3 tests were performed, given the stochastic character of the algorithm. With the calibration of the parameters, tests were performed and the data are presented in Section 6. The values for the 3 executions and their respective average deviations are show in the paper Appendix. This table contains, from left to right, instance name (indicating the number of nodes and arcs percentage, respectively), number of nodes in TSP version, the optimal solution, the average of solutions for the three executions, the relative error between this average and the optimal solution, the parameters α, β, ρ, the number of iterations and number of ants.

The largest relative error was 10.46%. In Table 3 are presented the results grouped for the percentage arcs of instances. It can be seen that the largest relative errors were found for the mixed instances. This is trivial because the two totally versions of the MCPP are well solved, as commented earlier.

The relative error also show a dependence with the size of the graph. Concerning to the metaheuristic behavior, it promotes a good "sweep" of the path possibilities, reducing its determinism, especially with the variable evaporation module. However, to obtain this range of solutions has a computational price to be paid.

5.1 Comparisons with related works

We divided our comparisons into two, firstly about works that deal the MCCP and after that ones to TSP. In our bibliography research were identified two works in metaheuristics for MCPP and none of them are an ACO algorithm. Jiang et al. (2010) use very small instances, so comparisons with our results is unfeasible. Already Corberán et al. (2002) use a set of problems similar to ours. Table 4 contains their tests and ours.

Observing Table 4, we can conclude that GRASP is better than our algorithm, reaching very good results in short time. However, Corberán et al. (2002) deal small problems and results cannot extended for larger instances.

Now, we compare our algorithm to others based on Ant Colony Optimization. The work of Dorigo and Gambardella (1997) was selected for comparisons, because the used instances are similar to ours and it presents an basic algorithm just like us, without big increments in basic framework of ACO. From Table 5 below, we can make some conclusions.

Observing Table 5, we can affirm that our algorithm is not better than that one compared, but it produces comparable results.

6 Conclusion

This paper dealt with the Delivery/Collection Problem, mathematically translated through the Chinese Postman Problem modeled on a mixed graph. These problems framed in this class are very important for various daily activities of the society. Therefore, the objective of this study was to present a new algorithm based on Ant Colony Optimization and to test it for different instances. It was adopted an approach based on Chinese Postman Problem translation into Traveling Salesman Problem. For the best of our knowledge, this is the first ACO algorithm for this problem. Although the results are not better than others algorithms, for some instances results are clearly comparable.

For future studies, the exponential decay rate and the use of locals searches are pretensions, since the behavior of the algorithm presents a good chance of improvement in the implementation of these mechanisms.

References

Ahmed, Z. H. (2010). Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator, *International Journal of Biometrics and Bioinformatics* **3**(6): 96–105. https://doi.org/10.14569/IJACSA.2020.0110275.

Bai, J., Yang, G.-k., Chen, Y.-W., Hu, L.-S. and Pan, C.-C. (2013). A model induced max-min ant colony optimization for asymmetric traveling salesman problem, *Applied Soft Computing* **13**(3): 1365—-1375. https://doi.org/10.1016/j.asoc.2012.04.008.

Bodin, L. D. and Kursh, S. J. (1979). A detailed description of a computer system for the routing and scheduling of street sweepers, *Computers & Operations Research* **6**(4): 181–198. https://doi.org/10.1016/0305-0548(79)90002-9.

Table 2: Computational results for the parameters for the proposed algorithm

Instance	Number of nodes in TSP	Optimum solution	Best solution found	Execution time	Relative error (%)	α	β	ρ(%)	T	m
N100A0	498	9095	9134	118.714	0.43	1	7	20	50	100
N100A25	438	9487	9926	86.408	4.63	1	7	20	50	100
N100A75	311	9256	9346	47.687	0.97	1	7	20	50	100
N100A100	250	9712	9936	30.890	2.31	1	7	20	50	100
N200A0	996	18454	18920	415.120	2.53	2	7	25	50	100
N200A25	872	18419	19608	248.968	6.46	2	7	25	50	75
N200A50	747	18464	19620	178.585	6.26	2	7	25	50	75
N200A75	625	19063	20722	121.520	8.70	1	7	20	50	75
N200A100	500	19235	19855	102.072	3.22	1	7	20	50	100
N300A0	1500	27482	28501	748.099	3.71	2	7	25	50	80
N300A25	1311	27641	29613	427.505	7.13	2	7	25	50	60
N300A50	1124	27971	30106	316.406	7.63	2	7	25	50	60
N300A75	936	27971	29718	366.765	6.25	2	7	25	50	100
N300A100	749	29149	30645	234.609	5.13	2	7	25	50	100
N400A0	2000	37000	38905	717.406	5.15	2	7	25	40	50
N400A25	1746	37160	40169	630.922	8.10	2	7	25	50	50
N400A50	1497	37275	40486	743.651	8.61	2	7	25	50	80
N400A75	1250	38676	41070	517.770	6.19	2	7	25	50	80
N400A100	998	39419	41008	329.239	4.03	2	7	25	50	80
N500A0	2496	46163	48357	840.583	4.75	2	7	25	40	40
N500A25	2182	46111	49932	803.057	8.29	2	7	25	40	50
N500A50	1874	46659	51538	739.166	10.46	2	7	25	50	50
N500A75	1561	47953	51732	825.026	7.88	2	7	25	50	80
N500A100	1248	48802	51278	514.139	5.07	2	7	25	50	80

Table 3: Mean errors given the percentage of arcs

Group of instances	Average error %
A0	3.31
A25	6.92
A50	8.24
A75	6.00
A100	3.95

Bullnheimer, B., Hartl, R. F. and Strauß, C. (1997). A new rank based version of the ant system - a computational study, *Central European Journal for Operations Research and Economics* **7**(1): 25–38.

Cheng, R. and Gen, M. (1994). Crossover on intensive search and traveling salesman problem, *Computers & Industrial Engineering* **27**(1-4): 485–488. https://doi.org/10.1016/0360-8352(94)90340-9.

Choong, S. S., Wong, L.-P. and Lim, C. P. (2019). An artificial bee colony algorithm with a modified choice function for the traveling salesman problem, *Swarm and Evolutionary Computation* **44**: 622–635. https://doi.org/10.1016/j.swevo.2018.08.004.

Corberán, A., Martí, R. and Sanchis, J. M. (2002). A grasp heuristic for the mixed chinese postman problem, *European Journal of Operational Research* **142**(1): 70–80. https://doi.org/10.1016/S0377-2217(01)00296-X.

Dong, G., Guo, W. W. and Kevin, T. (2012). Solving the traveling salesman problem using cooperative genetic ant systems, *Expert Systems with Applications* **39**(5): 5006–5011. https://doi.org/10.1016/j.eswa.2011.10.012.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms*, PhD thesis, Politecnico di Milano, Milano, Italy.

Dorigo, M. and Gambardella, L. M. (1997). Ant colonies for the travelling salesman problem, *Bio Systems* **43**: 73–81.

Dorigo, M., Maniezzo, V. and Colorni, A. (1996). Ant system: Optimization by a colony cooperating agents, *IEEE Transactions on Systems, Man, and Cybernetics-Part B* **26**(1): 29–41. https://doi.org/10.1109/3477.484436.

Edmonds, J. and Johnson, E. L. (1973). Matching, euler tours and the chinese postman, *Mathematical Programming* **5**(1): 88–124. https://doi.org/10.1007/BF01580113.

Eglese, R. W. (1994). Routeing winter gritting vehicles, *Discrete Applied Mathematics* **48**(3): 231–244. https://doi.org/10.1016/0166-218X(92)00003-5.

Eiselt, H. A., Gendreau, M. and Laporte, G. (1995). Arc routing problems, part i: The chinese postman problem, *Operations Research* **43**(2): 231–242. https://doi.org/10.1287/opre.43.2.231.

Eldem, H. and Ülker, E. (2017). The application of ant colony optimization in the solution of 3d traveling salesman problem on a sphere, *Engineering Science and Technology, an International Journal* **20**(4): 1242–1248. https://doi.org/10.1016/j.jestch.2017.08.005.

Table 4: Corberán et al. (2002) comparisons.

GRASP				ACO			
Links	Arcs (%)	% Dev.	Time	Links	Arcs (%)	% Dev.	Time
	30	0.42	6.2		-	-	-
132	50	0.60	2.8	-	-	-	-
	70	0.66	1.2		-	-	-
	30	1.34	89.4		25	4.63	86.4
257	50	1.36	25.9	250	50	-	-
	70	0.69	7.4		75	0.97	47.7
	30	2.04	812.5		25	6.46	249.0
510	50	2.06	209.6	499	50	6.26	178.6
	70	0.90	47.3		75	8.70	121.5

Table 5: Dorigo and Gambardella (1997) comparisons.

ACS		ACO	
Num. of Arcs	% Dev.	Num. of Arcs	% Dev.
198	0.68	250	2.31
442	0.96	498	0.43
532	1.67	500	3.22
778	2.37	749	5.13
1577	4.11/4.31	1500	3.71

Euchi, J., Yassine, A. and Chabchoub, H. (2015). The dynamic vehicle routing problem: Solution with hybrid metaheuristic approach, *Swarm and Evolutionary Computation* **21**: 41–53. https://doi.org/10.1016/j.swevo.2014.12.003.

Ezugwu, A. E.-S. and Adewumi, A. O. (2017). Discrete symbiotic organisms search algorithm for travelling salesman problem, *Expert Systems with Applications* **87**: 70–78. https://doi.org/10.1016/j.eswa.2017.06.007.

Gambardella, L. and Dorigo, M. (1996). Solving symmetric and asymmetric tsp by ant colonies, pp. 622–627. https://doi.org/10.1109/ICEC.1996.542672.

Golden, B., Nossack, J., Pesch, E. and Zhang, R. (2017). Routing problems with time dependencies or how different are trash collection or newspaper delivery from street sweeping or winter gritting?, *Procedia Engineering* **182**: 235–240. https://doi.org/10.1016/j.proeng.2017.03.174.

Gordenko, M. and Avdoshin, S. (2017). The mixed chinese postman problem, *Proceedings of the Institute for System Programming of the RAS* **29**(04): 107–122. https://doi.org/10.15514/ISPRAS-2017-29(4)-7.

Grötschel, M. and Win, Z. (1992). A cutting plane algorithm for the windy postman problem, *Mathematical Programming* **55**(1-3): 339–358. https://doi.org/10.1007/BF01581206.

H. Holland, J. (1984). *Genetic Algorithms and Adaptation*, pp. 317–333. https://doi.org/10.1007/978-1-4684-8941-5_21.

Ismkhan, H. (2017). Effective heuristics for ant colony optimization to handle large-scale problems, *Swarm and Evolutionary Computation* **32**: 140–149. https://doi.org/10.1016/j.swevo.2016.06.006.

Jiang, H., Kang, L., Zhang, S. and Zhu, F. (2010). Genetic algorithm for mixed chinese postman problem, *Advances in Computation and Intelligence*, pp. 193–199. https://doi.org/10.1007/978-3-642-16493-4_20.

Karabulut, K. and Tasgetiren, M. F. (2014). A variable iterated greedy algorithm for the traveling salesman problem with time windows, *Information Sciences* **279**: 383–395. https://doi.org/10.1016/j.ins.2014.03.127.

Kóczy, L. T., Földesi, P. and Tüű-Szabó, B. (2017). Enhanced discrete bacterial memetic evolutionary algorithm - an efficacious metaheuristic for the traveling salesman optimization, *Information Sciences* pp. 389–400. https://doi.org/10.1016/j.ins.2017.09.069.

Kennedy, J. and Eberhart, R. (1995). *Particle Swarm Optimization*, Vol. 4, pp. 1942–1948. https://doi.org/10.1109/ICNN.1995.488968.

Khan, I., Maiti, M. K. and Maiti, M. (2017). Coordinating particle swarm optimization, ant colony optimization and k-opt algorithm for traveling salesman problem, pp. 103–119. https://doi.org/10.1007/978-981-10-4642-1_10.

Kirkpatrick, S., Gelatt Jr., C. D. and Vecchi, M. P. (1983). Optimization by simulated annealing, *Science (New York, N.Y.)* **220**: 671–80. https://doi.org/10.1126/science.220.4598.671.

Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms, *European Journal of Operational Research* **59**(2): 231–247. https://doi.org/10.1016/0377-2217(92)90138-Y.

Laporte, G. (1997). Modeling and solving several classes of arc routing problems as traveling salesman problems, *Comput. Oper. Res.* **24**(11): 1057–1061. https://doi.org/10.1016/S0305-0548(97)00013-0.

Mahi, M., Baykan, m. K. and Kodaz, H. (2015). A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for travelling salesman problem, *Applied Soft Computing* **30**: 484–490. https://doi.org/10.1016/j.asoc.2015.01.068.

Mei-Ko, K. (1962). Graphic programming using odd or even points, *Chinese Mathematics* **1**: 273–277.

Mestria, M., Ochi, L. S. and Lima Martins, S. d. (2013). Grasp with path relinking for the symmetric euclidean clustered traveling salesman problem, *Computers & Operations Research* **40**(12): 3218–3229. https://doi.org/10.1016/j.cor.2012.10.001.

Monkman, S. K., Morrice, D. J. and Bard, J. F. (2008). A production scheduling heuristic for an electronics manufacturer with sequence-dependent setup costs, *European Journal of Operational Research* **187**(3): 1100–1114. https://doi.org/10.1016/j.ejor.2006.06.063.

Nagata, Y. and Soler, D. (2012). A new genetic algorithm for the asymmetric traveling salesman problem, *Expert Systems with Applications* **39**(10): 8947–8953. https://doi.org/10.1016/j.eswa.2012.02.029.

Nobert, Y. and Picard, J.-C. (1996). An optimal algorithm for the mixed chinese postman problem, *Networks* **27**(2): 95–108. https://doi.org/10.1002/(SICI)1097-0037(199603)27:2<97::AID-NET1>3.0.CO;2-8.

Osaba, E., Yang, X.-S., Diaz, F., Lopez-Garcia, P. and Carballedo, R. (2016). An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems, *Engineering Applications of Artificial Intelligence* **48**: 59–71. https://doi.org/10.1016/j.engappai.2015.10.006.

Papadimitriou, C. H. (1976). On the complexity of edge traversing, *Journal of Association Computing Machinery* **23**(3): 544–554. https://doi.org/10.1145/321958.321974.

Prakasam, A. and Savarimuthu, N. (2015). Metaheuristic algorithms and polynomial turing reductions: A case study based on ant colony optimization, *Procedia Computer Science* **46**: 388–395. https://doi.org/10.1016/j.procs.2015.02.035.

Rao, R. V., Savsani, V. J. and Vakharia, D. P. (2011). Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems, *Computer-Aided Design* **43**: 303–315. https://doi.org/10.1016/j.cad.2010.12.015.

Saenphon, T., Phimoltares, S. and Lursinsap, C. (2014). Combining new fast opposite gradient search with ant colony optimization for solving travelling salesman problem, *Engineering Applications of Artificial Intelligence* **35**: 324–334. https://doi.org/10.1016/j.engappai.2014.06.026.

Shafani, A. and Haghani, A. (2015). Generalized maximum benefit multiple chinese postman problem, *Transportation Research Part C: Emerging Technologies* **55**: 261–272. https://doi.org/10.1016/j.trc.2015.01.017.

Sherafat, H. (1988). Uma solução para o problema do carteiro chinês misto, *Anais do IV CLAIO-XXI SBPO*, Rio de Janeiro, Brasil, pp. 157–170.

Sherafat, H. (2004). *Algoritmos Heurísticos de Cobertura de Arcos*, Doutorado em engenharia de produção, Universidade Federal de Santa Catarina, Santa Catarina, Brasil.

Sherafat, H. (2013). Sistema construtor de circuitos e sua aplicaÇÃo na roteirizaÇÃo de coleta de lixo domiciliar, *Revista GEINTEC* **3**(5): 544–554. https://doi.org/10.7198/S2237-0722013000050027.

Sherafat, H. (2017). Social network optimization a new methaheuristic for general optimization problems, *Revista GEINTEC - Gestão, Inovação e Tecnologias* **7**(4): 4123–4130.

Stüzle, T. and Hoos, H. H. (2000). Max-min ant system, *Future Generation Computer Systems* **16**(8): 889–914. https://doi.org/10.1016/S0167-739X(00)00043-1.

Stüzle, T. and Linke, S. (2002). Experiments with variants of ant algorithms, *Mathware & soft computing* **9**(2-3): 1–15.

Taillard, r. D. and Helsgaun, K. (2019). Popmusic for the travelling salesman problem, *European Journal of Operational Research* **272**(2): 420–429. https://doi.org/10.1016/j.ejor.2018.06.039.

Thimbleby, H. W. (2003). The directed chinese postman problem, *Journal of Software - Practice and Experience* **33**(11): 1081–1096. https://doi.org/10.1002/spe.540.

Unold, O. and Tarnawski, R. (2016). Cultural ant colony optimization on gpus for travelling salesman problem, Vol. 10122, pp. 317–329. https://doi.org/10.1007/978-3-319-51469-7_27.

Xiao, Z. and Jiang-qing, W. (2012). Hybrid ant algorithm and applications for vehicle routing problem, *Physics Procedia* **25**: 1892–1899. https://doi.org/10.1016/j.phpro.2012.03.327.

Yan, Y., Sohn, H.-s. and Reyes, G. (2017). A modified ant system to achieve better balance between intensification and diversification for the traveling salesman problem, *Applied Soft Computing* **60**: 256–267. https://doi.org/10.1016/j.asoc.2017.06.049.

Yang, X.-S. (2010a). Harmony search as a metaheuristic algorithm, *Studies in Computational Intelligence* **191**: 1–14. https://doi.org/10.1007/978-3-642-00185-7_1.

Yang, X.-S. (2010b). A new metaheuristic bat-inspired algorithm, *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)* **284**: 65–74. https://doi.org/10.1007/978-3-642-12538-6_6.

Zalilah, A. A. (2015). Ant colony hyper-heuristics for travelling salesman problem, *Procedia Computer Science* **76**: 534–538. https://doi.org/10.1016/j.procs.2015.12.333.

Zhao, N., Wu, Z., Zhao, Y. and Quan, T. (2010). Ant colony optimization algorithm with mutation mechanism and its applications, *Expert Systems with Applications* **37**(7): 4805–4810. https://doi.org/10.1016/j.eswa.2009.12.035.

Appendix

Appendix: General data

Instance	Execution1	Time Execution1	Execution2	Time Execution2	Execution3	Time Execution3	Execution Deviation	Time Execution Deviation
N100A0	9145	121.985	9134	112.063	9186	122.094	27	5.760
N100A25	10138	86.725	9926	86.860	10077	85.640	109	0.669
N100A75	9374	43.625	9346	49.828	9541	49.610	105	3.520
N100A100	9962	30.969	9975	32.250	9936	29.453	20	1.400
N200A0	18995	418.110	19009	416.953	18920	410.297	48	4.217
N200A25	19634	250.812	19723	243.250	19608	252.844	60	5.056
N200A50	20019	177.953	19620	178.992	19813	178.812	200	0.556
N200A75	20722	120.687	20850	125.719	20857	118.156	76	3.850
N200A100	19855	100.890	19948	101.125	19932	104.203	50	1.849
N300A0	28509	750.062	28501	748.641	28501	745.594	10	2.283
N300A25	29613	429.156	29723	427.563	29627	425.797	60	1.680
N300A50	30106	315.875	30183	313.688	30240	319.656	67	3.020
N300A75	29906	365.781	29718	368.297	29838	366.219	95	1.344
N300A100	30836	234.469	30654	234.047	30645	235.313	108	0.645
N400A0	38905	750.281	39075	701.843	38905	700.094	90	28.484
N400A25	40210	630.547	40420	630.438	40169	631.781	135	0.746
N400A50	40960	742.937	40486	743.953	40486	744.063	254	0.621
N400A75	41070	514.390	41133	524.656	41070	514.266	338	5.963
N400A100	41280	328.359	41511	328.594	41008	330.765	252	1.326
N500A0	48372	833.984	48865	858.922	48357	828.844	289	16.008
N500A25	50257	803.375	50589	813.469	49932	792.328	329	10.574
N500A50	51538	740.922	51684	748.156	51538	728.422	82	9.983
N500A75	52035	807.000	51732	845.719	51732	822.360	152	19.497
N500A100	51590	513.422	51278	513.964	51278	515.031	161	0.819