An Ant Colony Optimization Algorithm to Solve the Minimum Cost Network Flow Problem with Concave Cost Functions

Marta S. R. Monteiro
Faculdade de Economia and
LIAAD-INESC Porto L.A.,
Universidade do Porto
Rua Dr. Roberto Frias,
4200-464 Porto PORTUGAL
martam@fep.up.pt

Dalila B. M. M. Fontes
Faculdade de Economia and
LIAAD-INESC Porto L.A.,
Universidade do Porto
Rua Dr. Roberto Frias,
4200-464 Porto PORTUGAL
fontes@fep.up.pt

Fernando A. C. C. Fontes
Faculdade de Engenharia
and ISR-Porto,
Universidade do Porto
Rua Dr. Roberto Frias,
4200-465 Porto PORTUGAL
faf@fe.up.pt

ABSTRACT

In this work we address the Singe-Source Uncapacitated Minimum Cost Network Flow Problem with concave cost functions. Given that this problem is of a combinatorial nature and also that the total costs are nonlinear, we propose a hybrid heuristic to solve it. In this type of algorithms one usually tries to manage two conflicting aspects of searching behaviour: exploration, the algorithm's ability to search broadly through the search space; and exploitation, the algorithm ability to search locally around good solutions that have been found previously. In our case, we use an Ant Colony Optimization algorithm to mainly deal with the exploration, and a Local Search algorithm to cope with the exploitation of the search space. Our method proves to be very efficient while solving both small and large size problem instances. The problems we have used to test the algorithm were previously solved by other authors using other population based heuristics and our algorithm was able to improve upon their results, both in terms of computing time and solution quality.

Categories and Subject Descriptors: G.2.2 [Discrete Mathematics]: Graph Theory — Network problems

General Terms: Algorithms.

Keywords: Ant Colony Optimization, Concave Costs, Hybrid, Local Search, Network Flow.

1. INTRODUCTION

The Minimum Cost Network Flow Problem (MCNFP) includes a wide range of combinatorial optimization problems. Many applications exist, for instance supply chains, logistics, production planning, communications and transportations [16, 1]

MCNFPs with linear costs are solvable in polynomial time, that is, they are considered easy to solve. In this work, we consider nonlinear costs comprising fixed costs as well as variable costs associated with the flow in each arc. Regarding the variable costs we consider both linear and nonlinear concave costs. As such, the total costs incurred when using an arc are always nonlinear and concave. When concave costs are introduced in MCNFPs, then the difficulty to solve them increases and they become NP-Hard [17]. The special case of the Single-Source Uncapacitated Minimum Cost Network Flow Problem (SSU MCNFP) with fixed-charge costs has also been proven to be NP-Hard [18, 26, 16].

Most of the work developed on concave MCNFP considers only problems with fixed-charge costs, that is functions having a fixed component and a linear routing component. Recent works on fixed-charge MCNFPs are those of [26] where the authors propose a bilinear formulation from which an exact algorithm is derived; [22] where a series of concave piecewise linear network flow problems are solved by using a dynamic slope scaling first proposed by [20]; [23] provide an exact algorithm, a branch-and-cut for such problems.

Other works considering nonlinear concave routing costs [17, 19, 29] do not include a fixed component.

As far as we are aware of, the only works considering nonlinear concave routing costs and fixed costs simultaneously are those of [7], [14, 15, 12] and [13].

Following on the work of Fontes and Gonçalves [13], in this work we propose an Ant Colony Optimization (ACO) and a Hybrid Ant Colony Optimization (HACO) algorithm to solve Single-Source Uncapacitated (SSU) Minimum Cost Network Flow Problems (MCNFP) with a fixed costs component and a variable component, both considering a linear and a nonlinear concave routing component. The HACO algorithm incorporates a Local Search (LS) procedure, based on swap moves, in order to improve the best solution found at each iteration. The results show the effectiveness and efficiency of our HACO algorithm for both small and large size problems.

2. PROBLEM DEFINITION

Consider a directed graph $G = (N, A)$, where N is a set of $n + 1$ nodes and A is a set of m arcs (i, j). A single-source minimum cost network flow problem is a problem that minimizes the total costs g_{ij} incurred with the network

GECCO'11, July 12–16, 2011, Dublin, Ireland.

while satisfying the nodes demand d_j. The commodity flows from a single source t to the n demand nodes $i \in N \setminus \{t\}$.

The mathematical model for the Minimum Cost Network Flow Problem can then be written as follows:

$$\text{min:} \quad \sum_{(i,j)\in A} g_{ij}(x_{ij}) \tag{1}$$

$$\text{s.t.:} \quad \sum_{\{i|(i,j)\in A\}} x_{ij} - \sum_{\{k|(j,k)\in A\}} x_{jk} = b_j, \forall_{j\in N\setminus\{t\}}, \tag{2}$$

$$x_{ij} \geq 0, \quad \forall_{(i,j)\in A}, \tag{3}$$

where x_{ij} are the decision variables representing the amount of flow routed through arc (i, j). The objective in this problem is to minimize the costs incurred with the use of the network and with the flows routed through it which are represented as given in equation (1). Constraints (2) are called the *flow conservation constraints*. The first term of these constraints represents the flow entering the node and the second term represents the flow coming out from the node. Therefore, the flow conservation constraints state that the difference between the flow going into a node and the flow coming out from a node must be the demand of the node. Constraints (3) refer to the positive nature of the variables.

We consider two types of cost functions both including a fixed cost c_{ij} which is incurred whenever arc (i, j) is used. Cost functions of Type 1 consider a linear routing cost b_{ij} per unit of flow routed through arc (i, j) as follows:

$$g_{ij}(x_{ij}) = \begin{cases} 0, & \text{if } x_{ij} = 0, \\ b_{ij} \cdot x_{ij} + c_{ij}, & \text{otherwise.} \end{cases} \tag{4}$$

Regarding cost function of Type 2 we consider a concave routing cost, in addition to the fixed cost, as follows:

$$g_{ij}(x_{ij}) = \begin{cases} 0, & \text{if } x_{ij} = 0, \\ -a_{ij} \cdot x_{ij}^2 + b_{ij} \cdot x_{ij} + c_{ij}, & \text{otherwise.} \end{cases} \tag{5}$$

Concave MCNFPs have the combinatorial property that if a finite solution exists, then there exists an optimal solution that is a vertex (extreme point) of the corresponding feasible domain (defined by the network constraints). SSU concave MCNFPs have a finite solution if and only if there exists a direct path going from the source to every demand vertex and if there are no negative cost cycles, otherwise an unbounded negative cost solution would exist. Therefore, for the SSU concave MCNFP, an extreme flow is a tree rooted at the single source spanning all demand vertices. For details and proofs see Zangwill [31].

3. METHODOLOGY

In this section we review the Ant Colony Optimization methodology and the Hybrid ACO that was developed to solve the SSU concave MCNFP.

3.1 Ant Colony Optimization

Ant Colony Optimization (ACO) principles are based on the natural behaviour of ants while searching for the shortest path between their nest and some food source. One may argue that their goal is not to find the shortest path but, in the end, and with their behaviour, that is exactly what they end up achieving. Ants have to travel several times between their nest and a food source and, while travelling, they deposit in the path a chemical substance called pheromone[1]. If a path has a large concentration of pheromone, this is probably due to its shorter length that allowed ants to travel faster, resulting in a larger number of travellers and thus of ants depositing pheromone throughout the path. When an ant, later after, has to choose between paths to follow to reach the food source, the ant will choose with higher probability the path with the largest pheromone concentration. It was the observation of this sort of communication developed by the ants that inspired Dorigo and Stützle to develop the first ant based algorithm which was called Ant System [10], which was used to solve the Travelling Salesman Problem (TSP), a well known NP-Hard problem.

The Ant System (AS) has two main phases, the construction of the tour/solution and the pheromone update. Nevertheless, other decisions have to be made before the ants can start finding a solution, such as defining the structure of the solution, deciding on the number of ants to use and the initial pheromone quantity to spread in each path.

Ants move on the network by going from one node to another. The one where it moves to is probabilistically chosen based on the pheromone quantities deposited on the arcs outgoing from the current node. Since each ant starts on the same source node, when all nodes have been passed through, a feasible solution has been achieved. After the ants have constructed their respective solutions, the pheromone trails are updated. The update is done in two ways: on the one hand pheromone values are decreased through evaporation, that is its values are decreased by a constant decay; on the other hand, pheromone values are increased for the parts of the network which are present in the best solution(s). Such increase is proportional to the solution(s) quality. The process of solution construction and pheromone updating is repeated until some stopping criterion has been reached.

Following the first Ant Algorithm, improvements were attempted and sometimes achieved by modifying and/or adding some features. Nonetheless, the most important development that followed was the description of the Ant Colony Optimization Metaheuristic (ACO) by [11].

The main difference from the basic structure of the AS algorithm is the introduction of a *Daemon*. The daemon can perform operations that use global knowledge of the solutions, thus having a very active and important role in the algorithm, in contrast to the AS algorithm where each ant was supposed to deposit pheromone in its solution despite what the other solutions were like. This is a task that has no equivalence in nature. The daemon can control the feasibility of each solution, for example, by evaporating a percentage of the pheromone quantity in its arcs as a way of penalizing such a solution. Or can give an extra pheromone quantity to the best solution found from the beginning of the run of the algorithm, or even to the best solution in the current iteration.

Another important feature, commonly used by authors in the ant algorithms that were developed ever since, is the introduction of a Local Search feature following the construction of the solutions. This is an optional feature but has been proved to be very important in the exploitation of the

[1]This behaviour is shared by several animal species and even by some plants.

search space nearby good solutions, leading almost always to better performances of the ACO.

Ant colony based algorithm have been applied to solve a broad set of problems, mainly due to their versatility: network design problems [25], assignment problems [28, 5], facility location problems [3, 8], transportation problems [21, 27], just to mention but a few.

In the past few years authors have also developed hybrid algorithms between ACO algorithms and Local Search [24], Simulated Annealing [6], Post Processing Procedures [9], and even with Genetic Algorithms as is the case of [2].

Next, we will describe the approach we have made with ACO to solve the SSU concave MCNFP.

3.2 An Ant Colony Optimization Approach

ACO algorithms are characterized by a set of decisions that have to be made regarding the construction of the solution and some of the used parameters. The first and most important decision to be made is the representation of the solution to the problem being solved, because a poor representation can lead to not so good solutions.

3.2.1 Representation of the solution

As said before, a feasible extreme solution for the SSU MCNFP with concave costs is a set of existing arcs forming a tree, i.e., a connected graph without cycles. So, each ant solution consists on a number of arcs that equals the number of demand nodes.

3.2.2 Pheromone Update

After the construction of all the solutions for iteration T the best of them is identified, and the algorithm updates the pheromones. The pheromone update is performed according to the following update function:

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \Delta\tau_{ij}. \tag{6}$$

Initially, and simulating the natural process of evaporation, the pheromone values are reduced in every existing arc $(i,j) \in A$. This is represented by the first component of equation (6), that is $(1-\rho)\cdot\tau_{ij}$, where ρ represents the pheromone evaporation rate and $\rho \in]0,1]$, and τ_{ij} is the pheromone quantity in arc (i,j). The value of the evaporation rate indicates the relative importance of the pheromone values from one iteration to the following one. If ρ takes a value near 1, then the pheromone trail will not have a lasting effect, while a small value will increase the importance of the arcs a lot longer. The second component, $\Delta\tau_{ij}$, represents the pheromone quantity to be deposited, and is given by:

$$\Delta\tau_{ij} = \begin{cases} \frac{Q}{g(S)} & \text{if } (i,j) \text{ belongs to solution S} \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where S is the best solution found at the current iteration, $g(S)$ is its corresponding cost, and Q is a positive proportionality parameter.

3.2.3 Pheromone bounds

Initially, each arc on the problem is given a small amount of pheromone,

$$\tau_{ij} = \tau_0, \forall (i,j) \in A. \tag{8}$$

At each iteration, after the pheromone update is performed a check is done to find out if its value is bounded in the interval $[\tau_{min}, \tau_{max}]$, following the work of [30]. The τ_{max} value depends on the cost of the best solution found so far G^* and on the pheromone evaporation rate ρ, equation (9).

$$\tau_{max} = \frac{1}{(1-\rho)\cdot G^*} \tag{9}$$

The τ_{min} value depends on the upper bound for the pheromone value τ_{max} and on a parameter p_{best}, equation (10),

$$\tau_{min} = \frac{\tau_{max}\cdot(1-\sqrt[n]{p_{best}})}{(\frac{n}{2}-1)\cdot\sqrt[n]{p_{best}}}. \tag{10}$$

Since both τ_{min} and τ_{max} depend on the cost of the best solution found so far, they only have to be updated each time the best solution is improved. After each pheromone update a check is made to ensure that pheromone values are within the limits. If on the one hand, some pheromone value is bellow τ_{min}, it is set to τ_{min}. On the other hand, if some pheromone value is above τ_{max}, it is set to τ_{max}.

3.2.4 Construction of the solution

The method that is used to construct solutions for the SSU MCNFP guarantees that a solution is always feasible. All ants begin their solution construction at the source node. Initially, an ant selects an existing arc linking the source t and one of the demand nodes $i \in N \setminus \{t\}$. Then, the ant selects another arc, from the set of available arcs linking the source or one of the demand nodes already in the partial solution to another demand node not yet considered. This last step is performed until all the demand nodes are in the solution. Therefore the admissibility of the solution is guaranteed. The choice of the arc entering the solution is made using the probability function defined bellow:

$$P_{ij} = \frac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{(i,j)\in A}[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}, \tag{11}$$

where τ_{ij} is the pheromone present in arc (i,j) at the current iteration; η_{ij} is the heuristic information for the problem and is given by $\eta_{ij} = \frac{1}{c_{ij}+b_{ij}}$; $\alpha > 0$ and $\beta > 0$ are both parameters weighting the relative importance of the pheromone value and the heuristic information, respectively.

3.2.5 Algorithm

The pseudo-code for the ACO heuristic is given bellow:

```
Initialize pheromones
WHILE the no. of iterations <= maximum
  FOR each ant
    Construct solution
  END
  Identify the best solution of the iteration
  Compare with the best solution from the beginning
   and update it if necessary
  Evaporate Pheromones
  Reinforce the pheromone values in the arcs of the
   best solution of the current iteration
  FOR all arcs
    IF the pheromone values are out of the
     established bounds
```

```
    THEN modify values accordingly
  END
END
```

3.2.6 Tested Parameters

There are a few decisions regarding the values to be taken by the parameters described in the previous sections. The development of our algorithm was achieved in several phases and we had to set some values for our first experiences. In an initial phase we tested the parameter values given in Table 1. We used all the problems mentioned in section 4.1 as our training set.

Parameter	Values
α	1, 3
β	2, 3, 5
ρ	0.05, 0.1, 0.2, 0.5
Q	1, 2, 5
p_{best}	0.5, 0.05, 0.005
τ_0	1, 1000000
no. of ants	n, 2n
no. of iterations	100, 200, 1000

Table 1: Interval of values tested for the ACO parameters.

The ones with the best results are summarized in Table 2. After developing the last phase of the hybrid algorithm, we tested the parameter values once more. The results indicated that the ones chosen in the first tests were still the ones with the best average results.

Parameter	Value
α	1
β	3
ρ	0.1
Q	2
p_{best}	0.5
τ_0	1000000
no. of ants	n
no. of iterations	200

Table 2: ACO parameter configuration.

3.3 Local Search

After each iteration of the ACO heuristic has ended, and after the best solution of the iteration is found, the local search procedure takes place. The local search allows the algorithm to search a neighbour solution that might have a lower cost. For this problem a solution S' is a neighbour solution of solution S if S' is obtained from S by swapping an arc $(i, j) \in S$ by another arc $(k, j) \notin S$. Therefore, after the swap takes place node j gets its demand from node k instead of node i.

Local search is applied right after the tour construction and the best solution in the iteration is identified (see section 3.2.5). Five ants are selected from the current iteration to perform local search on their respective solutions, one is always the best for the iteration the other four are randomly selected. The pseudo-code of the local search algorithm is given by:

```
Sort solution S by ascending arc pheromone
FOR each arc (i,j) in solution S
  Sort all the arcs (k,j) that can replace arc
   (i,j) in descending order of pheromone
  FOR each arc(k, j)
    Replace arc (i,j) with the arc (k,j) with the
     largest pheromone value
    IF cost improved
      GOTO next arc in solution S
    ELSE
      Undo Replace
    END
  END
END
```

The arcs in the selected solution are sorted in ascending order of the value of their pheromone. For each of these arcs we will try to find an alternative one that improves the cost of the solution. For these we compute all the arcs that can replace the current one while maintaining the feasibility of the solution. We attempt to replace the original arc, starting with the ones with a higher pheromone value. If one of the replacements improves the cost of the solution S we proceed to the next arc in the solution S without attempting the remaining options. In the end if the solution found S' improves the cost of the original one, S, the new solution S' is the one used in the remaining of the algorithm.

4. COMPUTATIONAL EXPERIMENTS

In this section we present the computational results obtained with the ACO and HACO heuristic described above. We also present literature results for the same problems in order to compare the performance end effectiveness of our algorithms.

4.1 Test Problems

In order to test the algorithm that was developed to solve SSU concave MCNFPs we downloaded the Euclidean test set available from [4]. The set is divided in ten groups $\{g_1, g_2, \ldots, g_{10}\}$ with different ratios between variable and fixed costs, V/F, since it has been proven in [18] that the values of such ratios are the main parameter in defining problem difficulty for fixed-charge MCNFPs. Each of these subsets has three problem instances for each problem size. Furthermore, the number of nodes considered is 10, 12, 15, 17, 19, 25, 30, 40, and 50. For the problems with 40 and 50 nodes, there are only 5 groups defined. For further details on these problems please refer to [14]. Therefore, there is a total of 240 problems to be solved for each cost function type. Each of these 240 problem instances was solved five times and the average results obtained are presented and discussed in the next section.

4.2 Computational Results

The algorithm described in this paper was implemented in Java and the computational experiments were carried out in a PC with a Pentium D at 3.20GHz and 1 GB of RAM.

The performance of the heuristics is evaluated by using two measures:

1. Time, in seconds, required to run the algorithm;
2. % Gap.

The Gap is calculated by comparing two solutions, and is given by:

$$\text{Gap}(\%) = \frac{OptS - HS}{OptS} \times 100,$$

where OptS stands for the best known solution, which is an optimum in some cases, and the HS stands for the best solution found with the heuristic in question.

We compare our results with the ones obtained with a Hybrid Genetic Algorithm (HGA) reported in [13], and with the ones obtained by the Dynamic Programming Algorithm (DP), reported in [15]. In the HGA approach the authors used the following parameter settings: 10 times the number of nodes as the population size; a crossover probability of 70%; the top 15% of chromosomes are copied to the next generation; the bottom 15% of chromosomes of the next generation are randomly generated; the fitness function is given by the cost; and finally the number of generations allowed is 100.

Table 3: Average computational results for cost function Type 1 and small size instances.

	HGA		ACO		HACO	
Size	Gap	Time	Gap	Time	Gap	Time
10	0.005	0.82	0.05	0.07	0	0.15
12	0	1.23	0.18	0.10	0	0.23
15	0	2.11	0.51	0.17	0	0.38
17	0	3.15	0.59	0.24	0	0.55
19	0	4.00	0.78	0.31	0	0.70

For the cost function Type 1 problems with 10 to 19 nodes the DP provides the optimal solution. We present results for the ACO algorithm and the Hybrid ACO and compare them with previous results in the literature. In Table 3 we have the average times spent to run the algorithms and the gaps for the HGA, the ACO and the HACO. The the gap is computed using the optimal solution reported in [15]. As we can see, although on average the ACO does not manage to reach the optimal solution for all results, but average gap remain below 1%. The computational time remains well below 1s. The average gap increases with the number of nodes in the problem never reaching 1%. Introducing the Local Search procedure in the ACO (HACO) allows to find an optimal solution in all results, an improvement over HGA that failed to reach that value for some problems. Computational times are doubled going from ACO to HACO but remain below 1s and below HGA times.

Table 4: Average computational results for cost function Type 1 and large instances.

	HGA		ACO		HACO	
Size	Gap	Time	Gap	Time	Gap	Time
25	0	9.51	1.43	0.66	0	1.36
30	0	14.61	1.60	1.16	0	2.31
40	0.005	31.67	2.54	2.83	0	5.46
50	0	59.22	2.53	5.95	0	10.75

In Table 4 we have the results obtained for large size problems with cost function of Type 1. Again the optimal solution is used for gap computation. ACO gaps and computational times increase significantly in relation to small sized problems, but computation time remains reasonably low for problem complexity. HACO manages again to find all optimal solutions outperforming HGA. HACO time continues to double ACO computational time but remains considerably below HGA. Figures 1 and 2 illustrate the comparison between HGA, ACO, and HACO times for both sets of sizes, and as it can be seen the HACO behaviour is close to a linear function, and below ACO and HGA.

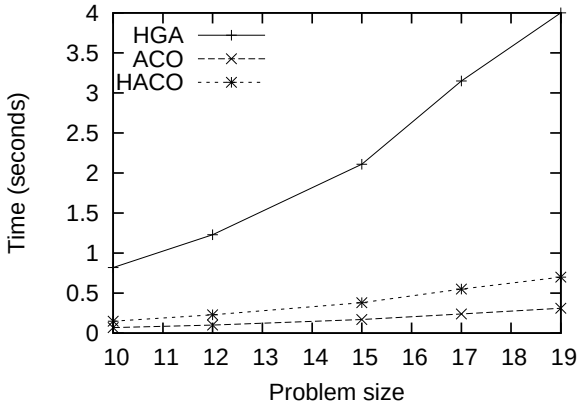

Figure 1: Computational time results obtained for small size instances and Type 1 cost functions.

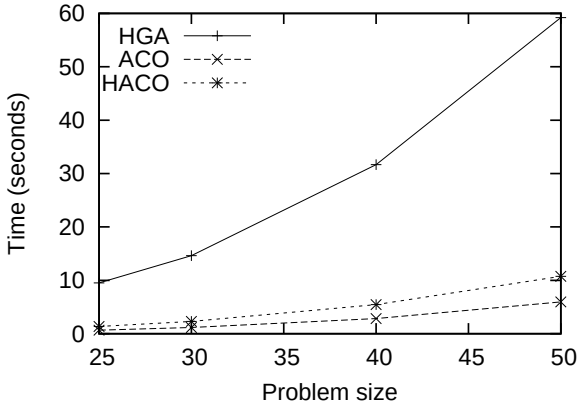

Figure 2: Computational time results obtained for large size instances and Type 1 cost functions.

In Table 5 we have the results obtained for small size instances with cost functions of Type 2 for the same algorithms. Optimal results reported in [15] are used to determine the gap. ACO performance improves in relation to Type 1 cost functions but remains unable to achieve optimum results in all instances. Both HGA and HACO find an optimal solution in all problems but HACO is at least 7 times faster than HGA, as can be seen in Figure 3.

Large size instances with Type 2 cost functions results are presented in Table 6. For these problems optimal solutions are not available, thus the quality of the solutions is measured by a ratio computed as a percentage using the upper bounds(UB) obtained in [14]. HGA and ACO are compared

Table 5: Average computational results for cost function Type 2 and small size instances.

	HGA		ACO		HACO	
Size	Gap	Time	Gap	Time	Gap	Time
10	0	0.90	0.01	0.09	0	0.16
12	0	1.42	0.12	0.15	0	0.24
15	0	2.50	0.49	0.21	0	0.40
17	0	3.74	0.17	0.29	0	0.58
19	0	4.63	0.38	0.37	0	0.74

with UB and HACO is compared with UB, HGA and ACO. HGA and HACO achieve the same results improving on UB for three sets of sizes. Again the computational time, see Figure 4, is considerably smaller for HACO in relation to HGA.

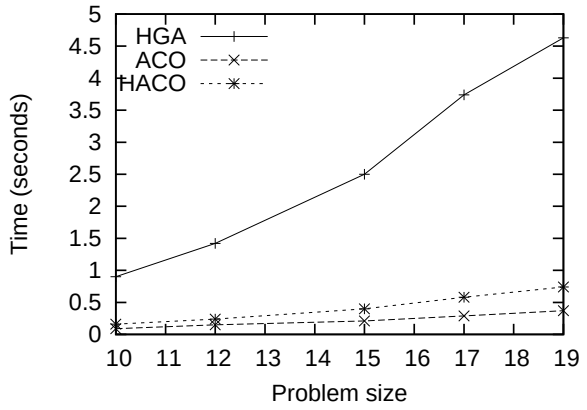

Figure 3: Computational time results obtained for small size instances and Type 2 cost.

In the previous tables we did not provide the minimum, maximum, and the standard deviation values for the HACO approach gap because they were always zero, and for the HGA approach because we did not have the necessary data. Nonetheless, Table 7 gives these results for the ACO approach. Although the average gap for ACO was never zero, the algorithm reached a zero minimum gap value for all but one size problem.

Table 7: Minimum, maximum and standard deviation values for the ACO gap and cost functions Type 1 and Type 2.

	Type 1			Type 2		
Size	Min	Max	StDev	Min	Max	StDev
10	0	0.87	0.17	0	0.89	0.10
12	0	2.74	0.48	0	2.04	0.32
15	0	4.64	0.93	0	6.88	1.16
17	0	5.04	0.98	0	1.67	0.35
19	0	5.03	1.15	0	3.59	0.67
25	0	10.28	1.72	0	3.81	0.97
30	0	9.33	1.85	0	7.47	1.65
40	0.20	5.18	1.05	0	5.80	1.20
50	0	9.33	2.36	0	11.48	2.63

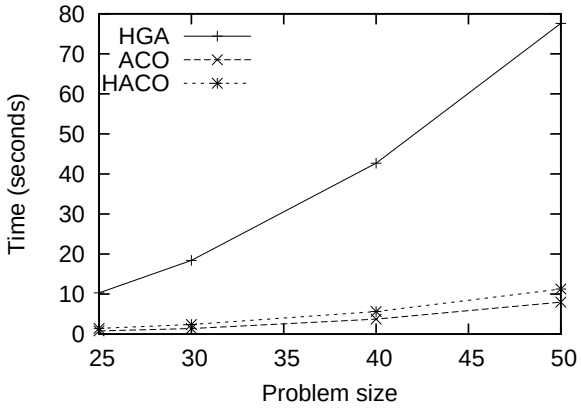

Figure 4: Computational time results obtained for large size instances and Type 2 cost.

5. CONCLUSIONS

In this work we develop a Hybrid algorithm based on Ant Colony Optimization and on Local Search, to solve the single-source uncapacitated minimum cost network flow problem with concave cost functions. The cost functions are of two types, fixed-charge costs and quadratic concave costs. We have solved both small size and large size problems, ranging from 10 to 50 nodes. We compare our results with the ones in literature and our algorithm proved to be very efficient. The solutions obtained were always as good or better than the ones obtained by HGA [13].

Furthermore, we have been able to find an optimal solution for every problem that was solved.

The quality of the results obtained has encouraged us to extend the scope of application of our HACO to other network flow problems in future work.

6. ACKNOWLEDGEMENTS

The financial support by FCT POCI and FEDER, through FCT Projects PTDC/EEA-CRO/116014/2009 and PTDC/EGE-GES/099741/2008 is gratefully acknowledged.

7. REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, and M. Reddy. Applications of network optimization. In *Network Models, volume 7 of Handbooks in Operations Research and Management Science*, pages 1–83, 1995.

[2] F. Altiparmak and I. Karaoglan. A genetic ant colony optimization approach for concave cost transportation problems. In *IEEE Congress on Evolutionary Computation, 2007. CEC 2007.*, pages 1685 –1692, 2007.

[3] A. Baykasoglu, T. Dereli, and I. Sabuncu. An ant colony algorithm for solving budget constrained and unconstrained dynamic facility layout problems. *Omega*, 34:385–396, 2006.

[4] J. Beasley. Or-library. http://www.brunel.ac.uk/deps/ma/research/jeb/orlib/netflowccinfo.html, 2010.

[5] E. M. Bernardino, A. M. Bernardino, J. M. Sánchez-Pérez, J. A. Gómez-Pulido, and M. A. Vega-Rodríguez. A hybrid ant colony optimization algorithm for solving the terminal assignment problem. In *IJCCI 2009 - International Joint Conference on Computational Intelligence*, 2009.

[6] L. Bouhafs, A. Hajjam, and A. Koukam. A combination of simulated annealing and ant colony system for the capacitated location-routing problem. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 409–416, 2006.

Table 6: Average computational results for cost function Type 2 and large size instances.

	HGA		ACO		HACO			
Size	HGA/UB	Time	ACO/UB	Time	HACO/UB	HACO/HGA	HACO/ACO	Time
25	100.72	10.28	102.04	0.79	100.72	100.00	98.71	1.44
30	99.13	18.39	100.77	1.37	99.13	100.00	98.42	2.38
40	99.90	42.70	102.20	3.72	99.90	100.00	97.86	5.62
50	99.94	77.62	101.94	7.99	99.94	100.00	98.07	11.26

[7] R. E. Burkard, H. Dollani, and P. T. Thach. Linear approximations in a dynamic programming approach for the uncapacitated single-source minimum concave cost network flow problem in acyclic networks. *Journal of Global Optimization*, 19:121–139, February 2001.

[8] C.-H. Chen and C.-J. Ting. Combining lagrangian heuristic and ant colony system to solve the single source capacitated facility location problem. *Transportation Research Part E: Logistics and Transportation Review*, 44(6):1099 – 1122, 2008.

[9] B. Crawford and C. Castro. Integrating lookahead and post processing procedures with ACO for solving set partitioning and covering problems. In *ICAISC*, pages 1082–1090, 2006.

[10] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 26(1):29–41, 1996.

[11] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.

[12] D. B. Fontes, E. Hadjiconstantinou, and N. Christofides. A branch-and-bound algorithm for concave network flow problems. *Journal of Global Optimization*, 34:127–155, January 2006.

[13] D. B. M. M. Fontes and J. F. Gonçalves. Heuristic solutions for general concave minimum cost network flow problems. *Networks*, 50:67–76, 2007.

[14] D. B. M. M. Fontes, E. Hadjiconstantinou, and N. Christofides. Upper bounds for single-source uncapacitated concave minimum-cost network flow problems. *Networks*, 41(4):221–228, 2003.

[15] D. B. M. M. Fontes, E. Hadjiconstantinou, and N. Christofides. A new dynamic programming approach for single-source uncapacitated concave minimum cost network flow problems. *European Journal of Operational Research*, 174:1205–1219, 2006.

[16] J. Geunes and P. Pardalos. *Supply Chain Optimization*. Springer, Berlin, 2005.

[17] G. Guisewite and P. Pardalos. Algorithms for the single-source uncapacitated minimum concave-cost network flow problem. *Journal of Global Optimization*, 3:245–265, 1991.

[18] D. Hochbaum and A. Segev. Analysis of a flow problem with fixed charges. *Networks*, 19:291–312, 1989.

[19] R. Horst and N. V. Thoai. An integer concave minimization approach for the minimum concave cost capacitated flow problem on networks. *OR Spectrum*, 20:47–53, 1998.

[20] D. Kim and P. M. Pardalos. A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. *Networks*, 35:216–222, 1999.

[21] R. Musa, J.-P. Arnaout, and H. Jung. Ant colony optimization algorithm to solve for the transportation problem of cross-docking network. *Computers & Industrial Engineering*, 59(1):85 – 92, 2010.

[22] A. Nahapetyan and P. Pardalos. Adaptive dynamic cost updating procedure for solving fixed charge network flow problems. *Computational Optimization and Applications*, 39:37–50, 2008.

[23] F. Ortega and L. A. Wolsey. A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem. *Networks*, 41:143–158, May 2003.

[24] H. D. Pour and M. Nosraty. Solving the facility layout and location problem by ant-colony optimization-meta heuristic. *International Journal of Production Research*, 44:5187–5196, 2006.

[25] E. Rappos and E. Hadjiconstantinou. An ant colony heuristic for the design of two-edge connected flow networks. In *ANTS Workshop*, pages 270–277, 2004.

[26] S. Rebennack, A. Nahapetyan, and P. Pardalos. Bilinear modeling solution approach for fixed charge network flow problems. *Optimization Letters*, 3:347–355, 2009.

[27] L. Santos, J. Coutinho-Rodrigues, and J. R. Current. An improved ant colony optimization based algorithm for the capacitated arc routing problem. *Transportation Research Part B: Methodological*, 44(2):246 – 266, 2010.

[28] S. J. Shyu, B. M. T. Lin, and T.-S. Hsiao. Ant colony optimization for the cell assignment problem in PCS networks. *Computers and Operations Research*, 33(6):1713–1740, 2006.

[29] D. K. Smith and G. A. Walters. An evolutionary approach for finding optimal trees in undirected networks. *European Journal of Operational Research*, 120(3):593 – 602, 2000.

[30] T. Stützle and H. Hoos. MAX-MIN ant system and local search for the traveling salesman problem. In *IEEE International Conference On Evolutionary Computation (ICEC'97)*, pages 309–314, Piscataway,NJ, 1997. IEEE Press.

[31] W. Zangwill. Minimum concave cost flows in certain networks. *Management Science*, 14:429–450, 1968.