Computación y Sistemas
ISSN: 1405-5546
computacion-y-sistemas@cic.ipn.mx
Instituto Politécnico Nacional
México

Bolufé-Röhler, Antonio; Otero Pereira, Juan Manuel; Fiol-González, Sonia
Traffic Flow Estimation Using Ant Colony Optimization Algorithms
Computación y Sistemas, vol. 18, núm. 1, enero-mayo, 2014, pp. 37-50
Instituto Politécnico Nacional
Distrito Federal, México

Traffic Flow Estimation Using Ant Colony Optimization Algorithms

Antonio Bolufé-Röhler[1], Juan Manuel Otero Pereira[1], and Sonia Fiol-González[1]

[1] Facultad de Matemática y Computación, Universidad de la Habana, Cuba

{bolufe, otero, s.fiol}@matcom.uh.cu

Abstract. Simulation and optimization of traffic flows in a city or province allow the implementation of correct developing strategies and help the decision making process when using and distributing resources such as mass transit. This estimation can be modeled as a bifurcated multi-commodity network flow problem, where the general flow distribution is dictated by Wardrop's principles. In this paper two different Ant Colony Optimization algorithms are presented for solving this problem. The proposed algorithms are tested with real-life traffic demand in the Havana city. The obtained results are compared to those provided by classical algorithms, showing that the new ant colony algorithms provide good results as well as low running times.

Keywords. Non-linear optimization, metaheuristics, traffic problem, logistics, simulation.

Algoritmos de optimización basados en colonias de hormigas para la estimación de flujos de tráfico

Resumen. La estimación de flujos de tráfico permite implementar buenas estrategias de desarrollo, a la vez que ayuda en el proceso de toma de decisiones cuando se controlan y distribuyen recursos claves como el transporte masivo. La distribución de tráfico puede ser modelada como un problema de Flujo de Costo Mínimo para Múltiples Bienes. Para su solución, la Optimización de Colonia de Hormigas provee un marco de trabajo prometedor. En la presente investigación se presentan dos nuevos algoritmos basados en Colonias de Hormigas, los mismos se aplican a instancias reales del problema de estimación de flujo en Ciudad de La Habana. Los resultados alcanzados se comparan con los provistos por algoritmos clásicos, mostrando la efectividad del método propuesto.

Palabras clave. Optimización no-lineal, metaheurísticas, problema de tráfico, logística, simulación.

1 Introduction

It is possible to simulate the behavior of a system through the optimization of a model that accurately predicts it; such simulation allows analyzing the system's responses to different events and decisions. In this case study, we are concerned with the prediction of traffic flow distribution in the Havana city. As a result of optimizing a flow problem, it will be possible to simulate different decision making strategies for such scenarios as optimum semaphore distribution and synchronization, efficient selection of mass transit routes, or infrastructure development.

For this purpose we have different maps of the city roads, the biggest of them has over 8000 edges, representing the road sections, and 2000 nodes representing the intersection between the streets. The map's nodes are partitioned into zones, representing different districts or neighborhoods of the city; an estimation of the traffic demand between these zones is also known.

The problem to solve is finding an optimal flow distribution between these zones; we formalize the optimality assumption in our case study by Wardrop's principles [1]. The flow distribution dictated by Wardrop's principles state that the journey times in all routes actually used are equal to, or less than, those which would be

Computación y Sistemas Vol. 18 No. 1, 2014 pp. 37-50
ISSN 1405-5546
DOI: 10.13053/CyS-17-4-2013-017

experienced on any unused route. Assuming that the system will lean toward such a minimum free energy status, or equilibrium, it is possible to model the flow prediction as a Minimum Cost Multi-commodity Network Flow Problem (MCMNF) with a non-linear objective function. The commodities are associated with the flow running in each direction of every pair of zones. For every commodity there exist multiple source and sink nodes, i.e., the traffic moving from a zone to another can begin at any node of the starting zone and culminate at any node of the destination zone.

The MCMNF with a non-linear function and integer flow values is a well-known NP-hard optimization problem for which some heuristic and approximation algorithms have been developed. Nevertheless, state of the art is not yet fully satisfactory when facing large real-world optimization problems. In recent years, metaheursitics have demonstrated their ability to solve this kind of complex real-world problems. In particular, the Ant Colony Optimization (ACO) metaheuristic has proved to be specially fitted to deal with the kind of dynamism that exists in similar problems, such as traffic assignment and data routing.

In this paper, two new ACO algorithms are presented for solving the MCMNF problem. Both algorithms use a specially designed data structure as *construction graph* for the ants.

This paper is organized as follows. Section 2 describes two formulations used in the literature to model the MCMNF problem. Section 3 presents a brief introduction to state of the art. In Section 4 a first approach to the problem and the *p-shortest path data structure* are presented. In Sections 5, 6 and 7, the ant algorithms and a post-optimization method are explained. The rest of the paper provides the results and conclusions of this study.

2 Mathematical Model

The MCMNF problem is defined on a directed graph $G = (V; E)$ associated to the roads network, with $|V| = n$ and $|E| = m$. Additionally to the n network nodes, other z dummy nodes are added to connect all the nodes belonging to a same zone; every ordered pair of distinct zones defines a commodity. Using the dummy nodes it is possible to associate a single source-sink pair of nodes $(s_k; t_k)$ to each of the $K = z\,(z-1)$ commodities. Thus, the problem's objective function can be modeled as in (1), with constraints (2), (3) and (4):

$$\min c(f_{ij}) = \sum_{(i,j\in E)} \frac{dist_{ij} f_{ij}}{u_{ij} - f_{ij}} \tag{1}$$

subject to

$$0 \le f_{ij} \le u_{ij} \tag{2}$$

$$\sum_{j\in V} f_{ij}^{k} = 0 \quad i \in V, i \ne s_k, t_k \ , \tag{3}$$

$$\sum_{j\in V} f_{s_k j}^{k} = \sum_{j\in V} f_{i t_k}^{k} = d_k \quad k \in K \tag{4}$$

where $dist_{ij}$, f_{ij} and u_{ij} are the distance, running flow and an upper bound of the theoretical capacity of the road section $(i\,; j)$, respectively. The traffic demand for commodity k is represented by $\mathrm{d_k}$; while $\mathrm{f_{ij}}$, and $\mathrm{f_{ij}^k}$ are the total flow and the flow for commodity k on edge $(i;\ j)$, respectively. Constraint (2) establishes a limit on the total flow that can traverse a given edge, while constraints (3) and (4) guarantee flow conservation and demand satisfaction, respectively.

The previous model is known as a node-arc formulation. An alternative way to represent the problem is the arc-path formulation. The arc-path formulation is equivalent to the previous one, but is obtained by providing for each commodity the set of all paths connecting the source node s_k with the sink node t_k. In this formulation the constraints are expressed as

$$\sum_{p\in\Phi_k} \phi_p = d_k \quad k \in K\,, \tag{5}$$

$$f_{ij} = \sum_{p\in\Phi} \delta_{ij}^{p} \phi_p \le u_{ij} \quad (i,j) \in E \tag{6}$$

where Φ is the set of all possible paths and Φ_k is those paths used to transport commodity k. The flow running over path p is represented by ϕ_p, and δ_{ij}^p is a constant equal to 1 if arc $(i; j)$ belongs to path p and 0 otherwise. In this alternative model an exhaustive enumeration of all paths for every $(s_k; t_k)$ pair is assumed. For medium or big networks this assumption is numerically intractable since the number of paths growths exponentially with the problem dimensions. Therefore, in practice many algorithms dealing with this formulation restrict the number of paths considered between every source-sink pair, or include some iterative path generation procedure [2].

3 Previous Work

Various network or transportation problems can be modeled as non-linear multi-commodity flow problems, therefore, a vast collection of algorithms can be found in the literature on this topic. If the flow values can be real numbers, then algorithms based on classical mathematical programming methods such as feasible direction, cutting planes and sub-gradient methods can be used [2, 3]. Another practical approach is to approximate the objective function by a linear or piecewise linear function [4]. A combination of metaheuristics with mathematical programming (matheuristics) has also been successfully applied to the MCMNF problem [5].

On the other hand, if the flow values are integers, then the MCMNF becomes an NP-hard problem. In such case, the Ant Colony Optimization (ACO) provides an appropriate framework to deal with this kind of complex real-life optimization problems. The non-linear MCMNF problem is characterized by an inherent dynamic structure given by the application of Wardrop's principles.

Because of the internal representation based on pheromone trails, which needs to be updated and not reconstructed as the instance and its autocatalytic ability change, ACO systems are specially fitted to deal with this kind of dynamism. This hypothesis is supported by the results of ACO algorithms applied to complex network optimization problems such as the dynamic routing wavelength assignment [6], the non-bifurcated linear MCMNF [7], and the traffic assignment problem [8].

Recent works also show that the use of ACO algorithms can be effective when reducing response time and pocket loss in computer networks [9], as well as increasing sustainability in supply chains of transportation systems [10]. A new ACO algorithm called 'Redundant Link Avoidance (RLA) algorithm' is proposed in [9]. By removing redundant links in the merged routes, this algorithm improves the results provided by ACO based multi-path routing methodology. The solutions provided by RLA allow to tackle problems such as network burst, overloading and traffic merging. In [10] an ant colony algorithm is built based on a multi-objective approach in order to find Pareto-optimal solutions. These solutions aim to increase sustainability without sacrificing economic objectives. The proposed algorithm is analyzed using a case of transportation in Europe.

The results reported in [8] show the theoretical and practical advantages obtained when applying a correctly designed ACO algorithm to traffic flow optimization and simulation problems. This paper extends those results by presenting new algorithms and insights.

4 Shortest Path Sub-Graph Structure

The first attempt to develop an ant based algorithm for the MCMNF problem was applied to the arc-node formulation, i.e., the algorithm was directly executed over the network graph. In that algorithm a number of ants were sent simultaneously at each origin node s_k with a respective sink node t_k as destination. Each ant represented a group of vehicles having the same origin and destination. However, this approach was not effective; the main reason was the inability of the ants to find short paths between the zones.

The rules determining the ants' movements were similar to the ACS algorithm [11] and the trail update were inspired by the work in [8]. The heuristic information represented such physical road attributes as length and capacity, while the pheromone trail modeled the congestion of the

roads. The heuristic information was computed *a priori* using the Dijkstra algorithm, labeling every node with the traveling time of the uncongested shortest path to every destination. The trail was updated locally and globally. As every ant moved toward its destination, the pheromone trail was decreased, with the idea of making the path less desirable to other ants due to the flow increase during the solution construction. When a solution was obtained, the trail was globally updated proportionally to the solution quality; trail evaporation was also performed.

Despite this algorithm seems to be theoretically correct, in practice several difficulties arise during its application. The ants tend to ramble before arriving to their destination, and although the flow was fairly distributed through different paths, those paths were not the shortest ones. Two main reasons explain this behavior: the first one is a relatively small difference in the heuristic information values between two nodes that have a common adjacent node. For example, for a node n which is adjacent to n_1 and n_2, the heuristic information for a specific destination k of these two nodes is quite similar. Even if the step from n to n_1 directly nears the destination zone *k* and the step from n to n_2 goes in the opposite direction, the difference between the heuristic information for n_1 and n_2 is comparatively small. This is driven by the fact that the distances between two intersections in a city are usually small in comparison to the total distance to be travelled, as illustrated in Fig. 1. When a stochastic selection rule is applied to choose the next movement for an ant located in n, then there exists a high probability that a wrong decision is made.

The trail update is the second fact that causes a poor decision making process. During the solution construction process several ants are send for every $(s_k; t_k)$ pair until the flow demand is satisfied. As these ants move from their source to their destination, the pheromone trail is locally updated and decreased. This trail update guaranties a good flow distribution among different paths, but reinforces the possibility of choosing the node n_2 instead of n_1 for an ant located at n. This may happen when the step from node n to n_1 is repeatedly selected by other ants during the same solution. On the other hand, if the local trail update is not applied, or instead of decreasing the trail is increased, then the ants' routes converge over the shortest paths between every pair of zones which leads to a poor flow distribution, augmenting the flow congestion on the used paths and therefore the time of all journeys.

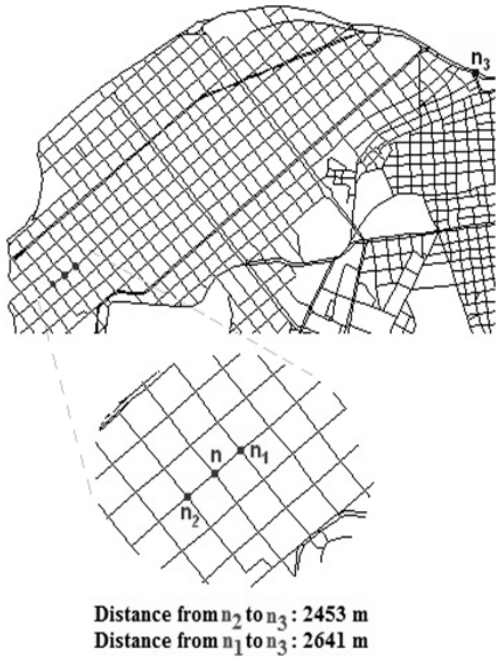

Fig. 1. Distances from adjacent street intersections (n_1 and n_2) to a destination point (n_3)

After this experience, and to overcome the arising difficulties, the ant algorithm was applied to the arc-path formulation. The basic idea is to preselect a set of promising paths between every pair of zones and establish them as the only possible routes. This approach better reflects what happens in the real life when a driver faces the task of determining a route.

From the algorithmic point of view, the arc-path formulation prevents the ants from making a wrong decision which could lead to unnecessary long tours, and at the same time allows the use of the pheromone trail as a mechanism for distributing the flow through different paths. This approach leads to better computational results, although it restricts the original solution space and the ants' ability to adapt to the environment.

To overcome these new drawbacks, a special data structure was designed to be used as a construction graph of the ant algorithm. Therefore, the ants won't move on the original network graph but on what will be called *the p-shortest path sub-graph structure*, denoted as G^p. As its name indicates, this structure is a collection of z different sub-graphs, $G^p = \{G_1^p, G_2^p, \ldots, G_z^p\}$. The sub-graph G_i^p will include those vertex and edges from the original network that belong to some of the *p*-shortest paths between any of the z zones and the zone i; p is a parameter of the algorithm that determines the number of preselected paths (presented results used $p = 20$).

Independently of the starting zone, an ant heading toward zone i will move only over the links belonging to the sub-graph G_i^p. For any pair of zone $(j\,;i)$ there will exist at least p different paths connecting zone j with the zone i in G_i^p. The maximum number of paths will depend on the amount of intersections between the *p*-shortest paths heading to i.

The *p-shortest path sub-graph structure* is a practical way to restrict the number of paths considered between every source-sink pair. At the same time using it as the construction graph allows a flexible implementation of the arc-path approach, since the path used by an ant is not selected beforehand from a fixed set of paths, but constructed as the ant moves over the correspondent G_i^p sub-graph.

4.1 Construction of the Shortest Path Sub-Graph Structure

For the construction of the G_i^psub-graphs two different methods were tested. The first method used the Eppstein algorithm to find the *p*-shortest paths between each pair of origin-destiny nodes [13]. The computational complexity of this methods is $O(m + nlog\, n + pn)$, with $n = |V|$ and $m = |E|$. The second method obtained the shortest paths by repeatedly applying the Dijkstra algorithm, after each run of which the cost of the edges belonging to the resulting paths were penalized multiplying their cost by a factor Ψ, which is a parameter. The cost function associated to the edges was $cost(i,j) = dist_{ij}/u_{ij}$ where $dist_{ij}$ is the distance and u_{ij} is the theoretical capacity of the road section (i,j). Since the Dijkstra algorithm needs to be executed p times, the computational complexity of this method is $O(p\, m\, logn)$. In both methods, once all the shortest paths toward zone i where computed (independently of the starting zone), they were merged into the G_i^p sub-graph.

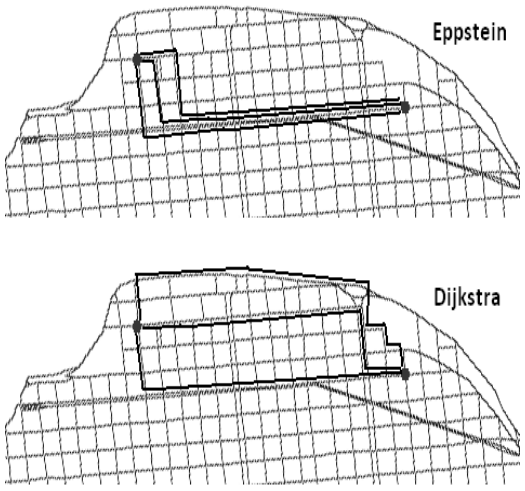

Fig. 2. Shortest paths provided by the methods based on the Eppstein and Dijkstra algorithms

In practice, the running time needed by both methods to construct the *p*-shortest path sub-graph structure were similar, however, the graph obtained using the Dijkstra algorithm leads to better solutions during optimization. This result was driven by the fact that the paths obtained through the penalized Dijkstra method were more diverse than the actual *p*-shortest paths provided by the Eppstein method. Fig shows the “shortest” paths provided by each method.

When implementing the *p*-shortest path sub-graph structure it is not necessary to actually have the k sub-graphs as independent data structure. Instead, it is enough to have, for each node, a routing table indicating the set of edges that can be followed to reach each of the destination zones. When a new path is added to a given sub-graph, it is enough to update, for each of its vertex, the set of available edges in the routing tables.

5 Ant Colony Algorithm for Multi-commodity Problems

The Ant Colony Algorithm for Multi-commodity problems (ACAM) is a simple ant based algorithm for the MCMNF problem (Fig. 3) In ACAM, artificial ants move on the *p-shortest path sub-graph structure* introduced in the previous section. The algorithm exploits artificial pheromone trails associated with each node i of the sub-graph network. The pheromone values denoted as τ_{ijd} represent the learned desirability to move from node i to node j with a destination zone d. At each node, an heuristic information value η_{ij} is also available, it characterizes the convenience to move from node i to j based on problem specific knowledge associated to the $(i; j)$ edge. This heuristic information is determined by physical road attributes and is computed *a priori* according to $\eta_{ij} = u_{ij}/dist_{ij}$.

ACAM is inspired by the Ant Colony System [11]. The algorithm iterates until some termination conditions are met and at every iteration a number ω of solutions are sequentially build by a colony of ants. After the iteration is concluded, the ω_b best solutions are used to update the trail proportionally to the solution quality. The values ω and ω_b are parameters of the algorithm with $1 < \omega_b < \omega$. The algorithm proceeds by creating a list of ants associated to an origin-destination pair $(s_k; t_k)$ and with an assigned flow cargo equal to θ which is a parameter of the algorithm (presented results used $\theta = 5$). The sum of all ants' cargo must cover the entire traffic demand between all zones. The ants are then randomly chosen and sent from their origin zone toward their destination. When the ant arrives, the trail of the traversed edges is updated using Equation 7, where $0 < \rho < 1$ is the pheromone evaporation rate (presented results used $\rho = 0.9$). The goal of this pheromone decrease is to diversify the paths used to deliver the flow between every pair of zones by making an already traversed edge less desirable. After a solution is obtained, the pheromone trail is reset to its original value at the beginning of the current iteration:

```
Algorithm ACAM (G, p, θ, ω, ω_b, ξ, ρ, flows)
G^p = Shortest_paths_subgrapgh(G, p)
while not StopCondition
  for i=1 to ω
    ants = CreateAnts(flows, θ)
    foreach ant in ants
      while NotArrived(ant)
        ant.Move(G^p, pheromone, heu_info, ξ)
      endwhile
      pheromone = LocalUpdate(ρ)
    endforeach
    solution_cost_i = Cost(solution_i)
    pheromone = ResetPheromone()
  endfor
  foreach j e ω_b
    pheromone = GlobalUpdate(solution_cost_i, ρ)
  endforeach
endwhile
return best_solution
```

Fig. 3. Pseudo-code of the Ant Colony Algorithm for Multi-commodity problems (ACAM)

$$\tau_{ijd} = (1 - \rho)\tau_{ijd} \ . \tag{7}$$

When an iteration ends, the ω_b best obtained solutions are used for a global trail update applying (8), where $S_{\omega b}$ is the set of best solutions at the current iteration; f^s_{ijd} is the flow running from node *i* to *j*, with destination zone d, in solution *s*; φ_s and φ_{best} are the costs of solution s and the best solution found so far, respectively:

$$\tau_{ijd} = \frac{\sum_{s \in S_{\omega b}} f^s_{ijd} \frac{\varphi_{best}}{\varphi_s}}{\mu_{ijd}} \ , \tag{8}$$

$$\mu_{ijd} = \sum_{s \in S_{\omega b}} \sum_{j \in N^d_i} f^s_{ijd} \frac{\varphi_{best}}{\varphi_s} \ . \tag{9}$$

The factor μ_{ijd} weights the deposit of trail, to make it proportional to the solution quality. This term guaranties that the pheromone trails remain smaller than 1, and is calculated using Equation 9, where N^d_i represents the neighbor nodes of *i* in G^p_d:

$$p_{ij}^{d} = \frac{\xi\, \tau_{ijd} + (1-\xi)\eta_{ij}}{\sum_{l \in N_i^d} \xi\, \tau_{ild} + (1-\xi)\eta_{il}} \,. \qquad (10)$$

At the construction stage, the list of created ants is cleared as they are randomly selected and sent from their assigned origin to their destination. During the journey the ants construct their route by applying a probabilistic action choice rule every time they are at an intersection with multiple links to choose from. Because of moving in G_d^p, there may exist nodes where only one edge is available when aiming for a destination d. If that is not the case, then an ant located at node i and leading to zone d moves to node j with probability p_{ij}^d, calculated as in (10). The parameter ξ allows balancing the influence between the pheromone trial and the heuristic information ($0 < \xi < 1$, presented results used $\xi = 0.8$). This rule is similar to the one used by the ANTS algorithm [12] and provided the best results in our experimental evaluation of different selection rules.

6 Ant Commodity Routing Algorithm

Although ACAM provides good computational results, its general design can be further improved by hybridizing it with techniques from other heuristic and approximation algorithms. The Ant Commodity Routing Algorithm (ACRA) in Fig. 4 has similarities with some well-known routing algorithms. The general operation of ACRA is strongly related with the approximation algorithm of Awerbuch and Leighton [14]. In ACRA, as in the approximation algorithm, the flow is pumped continuously into the network, until a given amount is reached. This flow accumulation allows modeling and locating the bottlenecks, and avoiding them on the final solution. Another point of reference is the AntNet algorithm [15], where the information about the network status, gathered by the ants through their journeys, is stored and updated on-line on node-local models.

The ACRA algorithm works by continuously sending ants with a flow cargo of θ, until some stop conditions are met (presented results used $\theta = 5$). The commodities are randomly selected with a probability proportional to their demand. Besides the pheromone trail, the model of the flow distribution generated by the ants is also stored; on the other hand, no heuristic information is used. After an ant arrives, it backtracks over the followed path to update the pheromone trail and the flow distribution, then another commodity is selected and the next ant is sent.

The pheromone trail value is associated to every node of G_d^p; in ACRA τ_{ijd} represents an estimation of the travelling time to zone d following the $(i; j)$ edge, given the current flow distribution. The trail is updated as an ant backtracks its path, and it is computed as $\tau_{ijd} = \delta_{jd} - \delta_{ij}$, where δ_{jd} is the time needed for the ant to travel from vertex *j* to the destination *d*, and δ_{ij} is the cost of traversing the edge $(i; j)$ as follows:

$$\delta_{ij} = \frac{dist_{ij}\, f_{ij}}{u_{ij} - f_{ij}} \,. \qquad (11)$$

To update the pheromone trail, it is necessary to store the flow values distributed by the ants. These values are directly associated to the edges of the original network *G*, making no distinction between flows of different destinations. Every time an ant backtracks, it increases the flow values by θ.To keep the flow values over the network at realistic levels, i.e., similar to those present in a solution of the problem, a flow evaporation process is implemented. Initially, a number of $\left\lfloor \frac{t_f}{\theta} \right\rfloor$ iterations are run without evaporation, allowing the flow to accumulate to the desired level; t_f is the total flow to be distributed among all the commodities. Once the desired level is achieved after each iteration, the flow evaporation process is applied to every edge. The evaporation rate guaranties that the flow values over the entire network remain stable and approximately equal to t_f.

In practice, for efficiency reasons, the evaporation is only performed on the edges backtracked by the ant. It is done before the flow increases and is calculated by (12), where $\Delta\iota_{ij}$ is the number of iterations since the last flow evaporation at the edge $(i; j)$. If it is the first time

the flow is evaporated on this edge, then $\Delta\iota_{ij}$ is the difference between the current iteration and $\left\lfloor \frac{t_f}{\theta} \right\rfloor$:

$$f_{ij} = \max\left(f_{ij} - \Delta\iota_{ij}\frac{f_{ij}}{t_f}\theta, 0\right). \quad (12)$$

The selection rule used by ACRA's ants during their movement is quite simple, an ant heading to zone d and situated at a node i chooses the next node j with a probability given by

$$p_{ij}^{d} = \frac{1}{\tau_{ijd}\sum_{l\in N_i^d}\frac{1}{\tau_{ild}}} \quad if\ j \in N_i^d\,. \quad (13)$$

The algorithm continuously iterates until some termination criteria are met, such as time limit or the total amount of flow. The pheromone trail laid by the ants is then used to construct a solution. The solution construction procedure follows the same principles as at the previous phase, generates and sends ants with a fixed amount of flow, using (11) as the selection rule. In this phase, the number of ants and their associated commodity exactly covers all the traffic demand of the problem. During the solution construction no pheromone or flow update is done.

7 Post-Optimization

The computational results presented in the next section show the good performance obtained by ACRA in our study. However, the performance of the algorithm strongly depends on a correct construction of the G^pstructure. The use of G^pas the construction graph in ACRA and ACAM restricts the solution space to a set of pre-selected paths. The global optimum of the original problem does not necessarily belong to this new restricted subspace. As standard ACO algorithms, ACRA and ACAM do not even guarantee the final solution to be a local optimum. It is possible to overcome these disadvantages designing a post-optimization method not restricted by the G^p structure on its search.

The post-optimization can be implemented using a standard local search (hill climbing) method. This local search method starts on a solution provided by ACRA and as a neighborhood structure uses a function that considers all the feasible solution space, i.e., the set of all possible paths which connect a couple of zones. Consequently, the final solution provided by the post-optimization procedure will be a local optimum of the original (not restricted) problem.

A convenient way to define the neighborhood function is to consider as neighbors of a given solution all those solutions that can be obtained by transferring a fixed amount of θ_{po} flow units between two routes of the same good solution

```
Algorithm ACRA (G, p, θ, ρ)
G^p = Shortest_paths_subgrapgh(G, p)
while not StopCondition
  (origin, destiny) = SelectGood(flows)
  ant = CreateAnt(origin, θ)
    while NotArrived(ant)
        ant.Move(G^p, pheromone)
    endwhile
    if (current_it > ⌊t_f/θ⌋)  EvaporateFlow(G^p )
  endforeach
  pheromone = UpdatePheromone(G^p)
endwhile
solution = BuildSolution(pheromone)
return solution
```

Fig. 4. Pseudo-code of the Ant Commodity Routing Algorithm (ACRA)

```
Algorithm ACRA_PO (solution, θ_po)
while Solution Improves
  foreach good
    worst = WorstActivePath()
    best = MinimumCostPath()
    new_solution = TransferFlow(worst, best, θ_po)
  endforeach
endwhile
return new_solution
```

Fig. 5. Post-optimization method for improving solutions provided by the ACRA algorithm

(presented results used $\theta_{po} = 2$). The source route, from which the flow is extracted, must be an active path. It means a path with a running amount of flow bigger than θ_{po}. On the other hand, the destination route can be any path connecting the origin-destiny nodes of a given good solution, with enough capacity to assimilate the θ_{po} units of flow without violating the capacity constraints.

The general working of the post-optimization method is very simple, see Fig. 5. On every iteration for each good solution, the worst active path and the minimum cost path are determined. Then a flow transfer is initiated from the first to the second one. This operation is executed incrementally: moving θ_{po} units of flow, evaluating the new solution and if this solution is better than the previous one, transferring another θ_{po} units of the flow. This procedure stops if all the flow from the active path has been transferred, the maximum capacity on any of the destiny path's arcs has been reached, or the new solution doesn't improve the previous one. The worst active path can be directly attained from ACRA solution, while the minimum cost path is determined applying the Dijkstra algorithm over the entire graph *G*. When evaluating the edges cost on the Dijkstra algorithm, the objective function (1) should be used, considering one unit as the minimum flow running on every arc to avoid a division by zero. The combination of ACRA with this post-optimization method will be denoted as ACRA_PO.

8 Parameter Tuning

A correct parameter tuning is fundamental when using metaheuristics; the performance, as well as the search behavior of the algorithm may strongly vary depending on the parameter settings. In the current research an exhaustive tuning process was performed using a hyperheuristic approach [16], i.e., a metaheuristic was used for finding the optimum parameter configuration of another metaheuristic. A simple genetic algorithm was used as a higher level heuristic algorithm for the tuning process. The genetic algorithm used a binary representation of the problem with a standard implementation of a one-point crossover, uniform mutation and tournament selection genetic operators [17]. Table 1 presents the parameters that were tuned, their brief description and the recommended values.

Table 1. Brief parameter description and recommended values

Parameter	Description	Values
p	Number of shortest paths found for each good solution when building the shortest path sub-graph structure. Larger values increase exploration; smaller values promote exploration of the search space.	[10, 20]
Ψ	Penalization applied to the length of edges when using the Dijkstra algorithm for building the shortest path sub-graph structure. Larger values lead to more disjoint paths.	[1.1, 1.2]
ω	Amount of solutions build on each iteration of the ACAM algorithm	12
ω_b	Amount of solutions selected for the pheromone update on each iteration of the ACAM algorithm.	4
ρ	Pheromone evaporation coefficient for the ACAM algorithm.	0.15
ξ	Balance between heuristic information and pheromone trail in ACAM.	0.8
τ_0	Pheromone initial value in the ACAM algorithm.	0.15
θ	Units of flow assigned to each ant in the ACAM and ACRA algorithms.	2

9 Computational Results

Six different instances of the traffic flow problem in the Havana city were used to analyze the computational results of the designed algorithms. Each instance corresponds to a different partial map of the city. Table 2 summarizes the main characteristics of each map: the number of nodes and edges on the graph, the amount of zones (neighborhoods) and the traffic flow to be distributed.

Table 2. Instances of the MCMNF problem

	Vertex	Edges	Zones	Flow
Map 1	474	1740	4	2670
Map 2	474	1740	7	7320
Map 3	1762	6426	6	8790
Map 4	1762	6426	12	12970
Map 5	2460	8846	12	10830
Map 6	2460	8846	20	10048

The results of the ACAM, ACRA and ACRA_PO algorithms were initially compared with those obtained by the MIP solver provided with the *COIN-OR* tool [18]. The solver worked on a relaxed model of the original problem, modeled by a piecewise linear approximation of the non-linear objective function. To create the piecewise linear function, the capacity of each edge is divided into *s* intervals of equal size, where *s* is a parameter of the model. Fig. illustrates an approximation of the original objective function with $s = 4$.

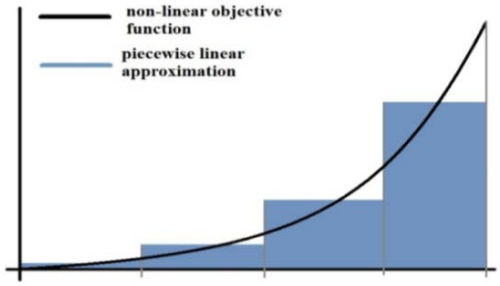

Fig. 6. Piecewise linear approximation of the non-linear objective function

Table 3 shows the average solutions found by each algorithm after 100 independent runs. The reported values for each instance are the costs of the objective function, i.e., the total travelling time of all the vehicles involved in the system. For the MIP solver, the numbers of parts used to approximate the objective function are given. It can be noticed that ACRA clearly outperforms ACAM; however, when compared with the MIP solver, both algorithms exhibit a good performance.

These results support the hypothesis that ACO metaheuristics provide an appropriate framework for solving complex real-life routing optimization problems. The effectiveness of the post-optimization method and the convenience of combining simple local search with ACO are also confirmed; the ACRA_PO algorithm provides the best solution for every instance.

Table 3. Average results of ACAM, ACRA and ACRA_PO and COIN-OR's MIP solver

Map	ACAM	ACRA	ACRA_PO	MIP Solver				
				s = 2	s = 4	s = 6	s = 8	s = 10
Map 1	7.28e+03	7.14e+03	7.13e+03	1.49e+04	1.09e+04	9.92e+03	9.90e+03	9.82e+03
Map 2	6.16e+03	5.08e+03	5.03e+03	9.04e+03	7.93e+03	7.25e+03	6.43e+03	6.15e+03
Map 3	3.81e+04	3.59e+04	3.52e+04	5.14e+04	4.42e+04	4.54e+04	4.47e+04	4.37e+04
Map 4	3.43e+04	3.18e+04	3.12e+04	4.53e+03	3.85e+03	3.69e+03	3.61e+03	3.54e+03
Map 5	4.67e+04	4.39e+04	4.34e+04	---	---	---	---	---
Map 6	4.10e+04	4.01e+04	3.96e+04	---	---	---	---	---

Table 4. Running time of each algorithm (seconds)

Map	ACAM	ACRA	ACRA_PO	MIP Solver				
				s = 2	s = 4	s = 6	s = 8	s = 10
Map 1	2.00e+00	2.00e+00	2.20e+00	2.04e+00	4.17e+00	5.29e+00	7.03e+00	1.38e+01
Map 2	2.00e+00	2.00e+00	3.41e+00	5.56e+02	5.77e+02	6.29e+02	5.06e+02	4.58e+02
Map 3	2.00e+00	2.00e+00	6.12e+00	1.36e+03	5.34e+03	1.04e+04	1.71e+04	1.89e+04
Map 4	2.00e+00	2.00e+00	6.20e+00	1.63e+05	1.93e+05	2.37e+05	2.57e+05	2.60e+05
Map 5	2.00e+00	2.00e+00	5.45e+00	---	---	---	---	---
Map 6	2.00e+00	2.00e+00	1.01e+01	---	---	---	---	---

Table 4 shows the time required for every algorithm to find the best solutions. ACAM and ACRA were executed with a fixed time budget of 3 seconds; on the other hand, the values for post-optimization (ACRA_PO) represent the additional time needed by the local search method. Given a good quality of the initial solutions provided by ACRA, this time is relatively short. Another relevant data is the execution time required by the exact method to solve the problem. As it can be noticed, this time increases exponentially with respect to the dimensions of each instance making it impossible to solve Instances 5 and 6 using a piecewise linear approximation of the objective function.

9.1 Comparison with other Ant Colony Algorithms

Previous work, such as the one developed by Maniezzo *et al.*, has successfully applied ACO algorithms to the Traffic Assignment Problem (ACO_TAP). Due to big similarities between the TAP and the MCMNF problems, it is straightforward to adapt and apply the algorithm proposed in [8] to the current optimization problem. Another well-known and highly effective ACO algorithm is the Max-Min ACO, developed by Stützle and Hoos [19]. A standard implementation of the Max-Min ACO algorithm was also used to solve the traffic estimation problem.

Both algorithms (ACO_TAP and Max-Min ACO) were used to solve the six instances of the MCMNF problem in the Havana city. On each case, the algorithms were applied using the original graph with all the edges and nodes (G), as well as the p-shortest path sub-graph structure (G^p). The average solutions found after 100 independent runs are presented in Table 5 together with the results reported by ACAM, ACRA and ACRA_PO. All algorithms had a

Table 5. Results provided by different ACO algorithms

Map	ACAM	ACRA	ACRA_PO	ACO_TAP		MAX-MIN ACO	
				G	G^p	G	G^p
Map 1	7.28e+03	7.14e+03	7.13e+03	9.29e+03	7.35e+03	1.09e+04	7.22e+03
Map 2	6.16e+03	5.08e+03	5.03e+03	7.04e+03	6.25e+03	9.43e+03	5.15e+03
Map 3	3.81e+04	3.59e+04	3.52e+04	4.04e+04	3.74e+04	4.27e+04	3.67e+04
Map 4	3.43e+04	3.18e+04	3.12e+04	3.63e+04	3.29e+04	3.51e+04	3.14e+04
Map 5	4.67e+04	4.39e+04	4.34e+04	5.10e+04	4.61e+04	5.12e+04	4.35e+04
Map 6	4.10e+04	4.01e+04	3.96e+04	4.40e+04	4.07e+04	4.39e+04	4.01e+04

maximum time of 3 seconds as the stopping condition. As it can be noticed, there is a clear difference in performance when using the original graph and the *p*-shortest path sub-graph structure. The use of G^p strongly improves the performance of both algorithms. It is also important to notice that although the results of all ACO algorithms are similar, ACRA_PO provides the best results on every instance.

9.2 Convergence Analysis

Different heuristics have usually different convergence speeds. Fig. 7 shows a plot of the best solution found by ACAM and ACRA (using Map 5). This comparison between the convergence rate of ACAM and ACRA reveals that ACAM has a slower convergence process before reaching a stagnation phase. On the other hand, although ACRA provides worst solutions during the first iterations, the quality of the best solution found quickly improves before reaching stability.

The convergence curve for ACRA shows that the quality of the solution built using the congestion model slightly decreases after a given point. This diminishment is a direct consequence of small fluctuations in the optimal pheromone matrix due to the stochastic nature of the process used to model traffic congestions. A simple way to overcome this difficulty is by building a final solution regularly using the current pheromone matrix. A better and more efficient strategy would be to stop the algorithm once the optimality point has been reached.

The main difficulty with this approach is to determine the correct moment to stop the algorithm. An empirical study showed that a recommendable stopping criterion can be provided by Equation 14, where n_i is the optimal number of iterations to be executed, t_f is the total amount of flow that has to be distributed in a given problem instance, and ϑ is the parameter that determines the units of flow assigned to each ant:

$$n_i = 3\left\lfloor \frac{t_f}{\vartheta} \right\rfloor . \tag{14}$$

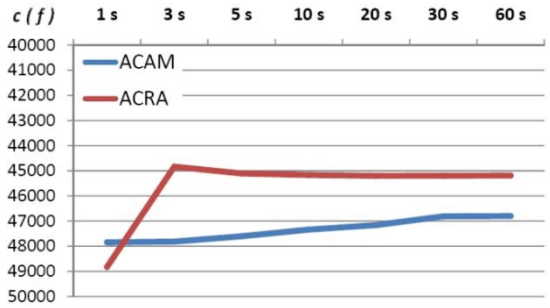

Fig. 7. Convergence rate of the ACAM and ACRA algorithms

9.3 Traffic Simulation

A traffic simulation has been implemented using the flow distribution provided by ACRA [20]. On this discrete event simulation, the movements and interactions of small groups of cars during their journey through the city were accurately predicted. The simulation is based on events such as the departure and arrival of groups of cars with the same origin and destiny, changes of lights on semaphores, waiting and crossing of roads intersections, among others. By using an adequate estimation of traffic demands and the flow distribution provided by ACRA, these events allow recreating the dynamic of the traffic behavior in a city.

The validation of the obtained data against real-world information shows the feasibility of this approach when gathering crucial information about city traffic. Examples of statistical information that can be obtained through the simulation are the waiting time of vehicles on roads intersections with and without a semaphore, the most used routes and their congestion levels, and the evolution of the traffic flow during the day. This useful information can be vital when distributing resources such as mass transit, planning future infrastructure developments, and synchronizing semaphores.

Fig. 8 shows an example of the overall flow distribution for Instance 6, where 20 zones were considered. It can be noticed how congestion is correctly handled when paths followed by different drivers are plotted. Different choices were made for going from one same origin to one same

Fig. 8. Traffic flow distribution on a partial map of the Havana city

destination. This ensures that no faster alternative way is available (Wardrop's first principle).

10 Conclusions

This paper presents two new ant algorithms for the MCMNF problem and their application to the case study of distributing traffic flows in the Havana city. Computational results have shown that ACO algorithms provide good results with low running times, especially when solving complex real-life routing optimization problems. The convenience of hybridizing ACO algorithms with techniques of other heuristic and approximation algorithms as well as with simple local search method has been also confirmed by the good results obtained using ACRA and ACRA_PO.

Although the computational results on various instances of the MCMNF problem with real networks suggest that ACRA and ACRA_PO can be used to efficiently determine flow distributions in a city, there is still room for future improvements. One of the main issues to be improved is the strong effect that some parameters have on the performance of the algorithms, especially those parameters which are related to the construction of the G^p structure. As stated earlier, the post-optimization method helps to partially overcome this difficulty; however, a great amount of time is still needed to correctly tune the parameters.

The general performance of the algorithms and the high standard deviation values in particular could also be improved in future works. Good results may be obtained by the interoperation of metaheuristics and mathematic programming; a promising trend known as *matheuristics* has shown to be very effective on solving hard real-life optimization problems. The hybridization between different metaheuristics is also an alternative for future developments.

To compare the performance of ACRA and ACRA_PO to other state of the art ACO algorithms, future work will adapt these algorithms to the particularities of other flow distribution problems, such as those presented in [9] and [10].

References

1. **Wardrop, J.G. (1952).** *Some theoretical aspects of road traffic research*. Palo Alto: Institution of Civil Engineers.
2. **Fratta, L., Gerla, M. & Kleinrock, L. (1973).** The flow deviation method: An approach to store-and-forward communication network design. *Networks*, 3(2), 97–133.
3. **Goffin, J., Gondzio, J., & Vial, R. (1997).** Solving nonlinear multicommodity flow problems by the analytic center cutting plane method. *Mathematical Programming*, 76(1), 131–154.
4. **Gabrel, V., Knippel, A., & Minoux, M. (1999).** Exact Solution of Multicommodity Network Optimization Problems with General Step Cost Functions. *Operations Research Letters*, 25(1), 15–23.
5. **Boschetti, M.A., Maniezzo, V., Roffilli, M., & Röhler A.B. (2009).** Matheuristics: Optimization, Simulation and Control. *Hybrid Metaheuristics. Lecture Notes in Computer Science*, 5818, 171–177.
6. **Garlick, R.M. & Barr, R.S. (2002).** Dynamic wavelength routing in WDM networks via ant colony optimization. *Ant Algorithms*, *Lecture Notes in Computer Science*, 2463, 250–255.
7. **Walkowiak, K. (2005).** Ant algorithm for flow assignment in connection-oriented networks. *International Journal on Applied Mathematics and Computer Science*, 15(2), 205–220.
8. **Maniezzo, V., Roffilli, M., Gabrielli, R., Guidazzi, A., Otero, M., & Trujillo, R. (2008).** Regional

Traffic Assignment by ACO. *Ant Colony Optimization and Swarm Intelligence, Lecture Notes in computer Science*, 5217, 409–410.

9. **Mohan, C. & Baskaran, R. (2010).** Improving network performance using ACO based redundant link avoidance algorithm. *International Journal of Computer Science Issues*, 7(3), 27–35.
10. **Sawadogo, M. & Anciaux, D. (2012).** Sustainable supply chain by intermodal itinerary planning: a multiobjective ant colony approach. *International Journal of Agile Systems and Management*, 5(3), 235–266.
11. **Dorigo, M. & Gambardella, L.M. (1997).** Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transaction on Evolutionary Computation*, 1(1), 53–66.
12. **Maniezzo, V. (1999).** Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS Journal of Computing*, 11(4), 358–369.
13. **Eppstein, D. (1998).** Finding the k Shortest Paths. *SIAM Journal on Computing*, 28(2), 652–673.
14. **Awerbuch, B. & Leighton, T. (1993).** A Simple local-control approximation algorithm for multicommodity flow. *34th Annual Symposium on Foundations of Computer Science*, Palo Alto, CA, 459–468.
15. **Di Caro, G. & Dorigo, M. (1998).** AntNet: distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research,* 9(1), 317–365.
16. **Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., & Schulenburg, S. (2003).** Hyper-heuristics: An emerging direction in modern search technology. *Handbook of Metaheuristics*, *International Series in Operations Research & Management Science*, 57, 457–474.
17. **Goldberg, D.E. (1989).** *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, Mass.: Addison-Wesley Pub. Co.
18. *COIN-OR Foundation*. (s.f.). Retrieved from *http://www.coin-or.org/foundation.html.*
19. **Stützle, T. & Hoos, H.H. (2000).** MAX-MIN Ant System. *Future Generation Computer Systems*, 16(8), 88–914.
20. **Fernández, C.A. & Bolufé-Röhler, A. (2011).** MapTools, Herramienta para la Simulación y Visualización de Problemas de Tráfico. *XXV JCJuvenil ICIMAF.*

Antonio Bolufé Röhler graduated in Computer Science from the University of Havana in 2007. In 2010 he received the M.Sc. degree in Mathematical Sciences and is currently a Ph.D. student. He holds an Assistant Professor position at the Faculty of Mathematics and Computer Sciences, where he teaches metaheuristics, simulation, logic and computer history. His main research area is application of metaheuristics for optimization, machine learning and computational linguistic analysis.

Juan Manuel Otero Pereira is a Bachelor in Mathematics from the University of Havana (1978) and a Ph.D. in Mathematics from the Humboldt University of Berlin (1987). He is Professor Titular at the Department of Applied Mathematics, the University of Havana, and Invited Professor of several Latin American universities. His main research area is the design of metaheuristics for vehicle routing problems, clustering and automatic classification.

Sonia Fiol González received her B.Sc. degree in Computer Science at the University of Havana in 2012. Currently, she is a Master student at the Faculty of Mathematics and Computer Science of the University of Havana. She works at Simpro, an enterprise dedicated to the development of simulators and virtual reality devices and leads Simpro's Research Lab at the Havana University. Her main research experience and interests are in the fields of databases and simulation.

Article received on 01/05/2013, accepted on 27/06/2013.