Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Concrete Delivery using a combination of GA and ACO

C.A. Silva, J.M. Faria, P. Abrantes, J.M.C. Sousa, M. Surico and D. Naso

Abstract— **The timely production and distribution of rapidly perishable goods such as concrete is a complex combinatorial optimization problem in the context of supply chain management. The problem involves several tightly interrelated scheduling and routing problems that have to be solved considering a trade-off of production and delivery costs. A hybrid meta-heuristic method combining genetic algorithms with constructive heuristics has been previously presented. This paper introduces a novel approach, by replacing the constructive heuristic with another meta-heuristic, the ant colony optimization approach. The simulation examples show that the concrete supply chain improves the performance with the novel GA-ACO algorithm.**

I. INTRODUCTION

Recently, the concrete production industry is experiencing a strategic evolution towards a supply chain business model, based on the decentralization of the concrete production activities. In general, a supply chain is a network of external partners (suppliers, warehouses and distribution centers) through which raw materials are acquired, transformed into products and delivered to customers [1]. In the concrete production case, the supply chain consists of a group of independent production centers and outsourcing delivery companies that agree to collaborate, by synchronizing their activities in order to reduce production and delivery costs.

The management of supply chains is a complex automation problem and the control and optimization of material, information and financial flows in supply chains is currently a very important research field [2]. It involves the solution of interdependent combinatorial problems such as the scheduling of production facilities or the routing of transport vehicles. It is necessary to guarantee not only that each of these problems is solved in a satisfactory way in short computation times, but also that the local optimization solutions are compliant with the global supply chain performance.

On the ready-made concrete production supply chain, it is necessary to schedule the orders from the clients among the different production centers and deliver the material to the client within strict time-windows, since the concrete is a perishable good. Presently, many companies tend to either rely on skilled operators that work out production plans based on their experience [3] or plan production operations on short time horizons, sacrificing the optimization on longer horizon to achieve a reduced risk of delayed delivery [4]. Recently, Naso *et al* proposed in [5] a hybrid meta-heuristic approach to solve the optimization of concrete-ready made supply chain. The proposed approach decomposes the global problem into two main stages: 1) scheduling of job production and loading at the production centers, optimized through a genetic algorithm (GA); 2) routing the fleet of trucks to deliver the jobs to customers, using a fast problem-specific constructive heuristic. This method was designed to achieve effective solutions for large-size instances (a supply chain with 5 production centers, servicing about 300 jobs with 50 trucks) in short computational times.

This paper presents a new approach for the second stage of the concrete supply chain decision, regarding job-to-truck assignment and the consequent truck routing. Routing problems are usually large combinatorial optimization problems, with complex constraints that cannot be efficiently handled by simple heuristics. The best known solutions for most of routing problems were obtained with meta-heuristics [6]. The Ant Colony Optimization (ACO) algorithm is one of the most recent optimization meta-heuristics that has been successfully used in complex routing problems [7]. This method provides competitive solutions when compared with other methods such as GA [6]. Therefore, this paper proposes a new hybrid GA-ACO approach for the just-in-time optimization of concrete-ready made delivery, by replacing the constructive heuristic that solves the truck assignment problem, with an ACO algorithm.

The paper proceeds as follows. Section II describes the concrete delivery problem as a combination of a scheduling and a routing problem, and includes a brief description of the GA optimization method for the first problem and the constructive heuristic for the second problem. Section III describes in detail the new ACO approach for the routing problem. The simulation results of the new method are presented and compared to the results of the previous approach in Section IV. Section V concludes the paper and describes the future research steps.

II. CONCRETE DELIVERY

Ready-mixed concrete (RMC) is a quickly perishable good that has to be produced on-demand and delivered at the construction sites within strict time-windows specified by customers. For strategic and logistic reasons, RMC suppliers have independent production centers (PCs) distributed on the serviced geographical area and organized as a supply network. In the most general case, a number of PCs in the network host a fleet of trucks for RMC delivery, while the

C.A. Silva, J.M. Faria, P. Abrantes, J.M.C. Sousa are with the Dept. of Mechanical Eng., GCAR/IDMEC, Instituto Superior Técnico, Technical University of Lisbon, 1049-001 Lisbon, Portugal. The work is partially supported by the Portuguese Foundation for Science and Technology (FCT) under Grant no. SFRH/BD/6366/2001 and partially supported by the project POCI/EME/59191/2004, co-sponsored by FEDER, Programa Operacional Ciência e Inovação 2010, FCT, Portugal. jmsousa@ist.utl.pt

M. Surico and D. Naso are with the Dep. Electrotechnics and Electronics, Politecnico di Bari, 70125 Bari, Italy naso@poliba.it

remaining ones do not own carriers, and explicitly rely on the other PCs for transportation. Usually, orders exceed the capacity of a single carrier and must be split into several sequential loads, hereafter called jobs.

Sequential job deliveries must necessarily be synchronized because the unload process at the customer site must be continuous (there cannot be significant pauses between the completion of a truck unload and the start of the next one) to avoid joints or other product-specific problems. The operational goals of PCs are multifaceted. Each PC aims at organizing its activities associating high resource utilization to low transportation costs and timeliness of the deliveries, the latter being particularly crucial for the characteristics of the supplied good. From a global viewpoint, operating the network involves the coordination of the various PCs, synchronizing production activities and tasks of the shared resources (trucks) in order to meet constraints and achieve production goals. Thus, scheduling the network is extremely challenging, not only for the typical combinatorial complexity that is particularly prohibitive in such large scale environments, but also for the high number of constraints deriving from the perishable nature of RMC, and for the conflicting nature of the cost and timeliness objectives.

The concrete delivery can be modeled as follows. Assume that there are $r = 1, \dots, R$ different demands to be scheduled. Each demand r is characterized by: the location of the customer, the *maximal unloading rate* UR_r, the *maximal delivery size* Mds_r, the *quantity of concrete* Q_r, the *setting time of the concrete* $Tset_r$, the *percentage of the truck* that must be empty Per_r, the *fixed waiting time* at the customer Fix_r, and the *earliest* and *latest delivery times* for accomplishing the deliveries EDT_r and LDT_r. We assume a homogeneous fleet of vehicles (same *maximum capacity* C_{max} and *average speed* V). If a demand exceeds the capacity of a single truck, it is equally divided into Z_r jobs, each of which will be delivered by a single vehicle. Jobs are defined as

$$Z_r = \frac{Q_r}{\min\{C_{max} \times (1 - Per_r), Mds_r\}}. \qquad (1)$$

At the end, the production problem can be described in terms of the different $i = 1, \dots, N$ jobs to be assigned to the different PCs, with $N = \sum_{r=1}^{R} Z_r$.

Each job can be described as a set of time parcels presented in Fig. 1. The parcels are: the *loading waiting time* at the dock LWT; the *loading* and *unloading time* LT and UT; the *source to destination travel time* and *return* SDT and DST; the *fixed waiting time* Fix and the *unloading wait time* UWT at the customer. The *starting time of loading* (STL) and *end time of unloading* (ETU) indicate the time parcel that a truck needs to deliver a job.

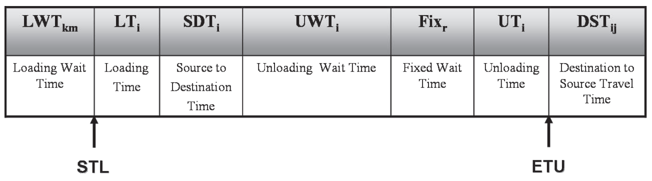


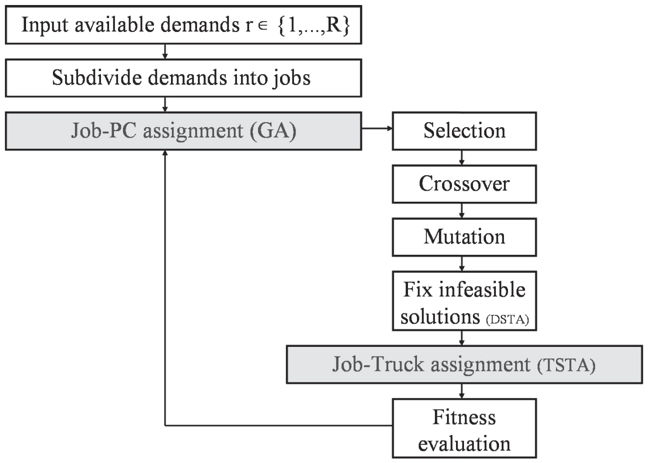
Fig. 1. Time parcels of a job.

The concrete delivery consists of finding for each job the production center where it is produced, determining the production and delivery time windows, and finally assign that job to a truck. The objective function f to be minimized is the one presented in [5], that can be succinctly described as:

$$f = f_{transport} + f_{un/loading} + f_{outsource} \qquad (2)$$

The first term groups the transportation costs associated with the distances covered by the fleet of trucks; the second term takes into account the loading and unloading times; the third term includes the costs associated with the number of outsourced concrete demands, the number of hired trucks, and the number of extra truck driver's working hours.

Finally, the optimization of the concrete supply chain is done by splitting the supply chain problem into two consecutive optimization problems: the concrete production and the truck assignment. The first problem consists of assigning the r demands to the D production centers – this is done by a genetic algorithm and is hereafter called the *job-PC assignment problem*. After the GA has generated the different possible solutions, the truck routing problem is solved for each chromosome – this can be done either by a constructive heuristic, or using an ACO algorithm as proposed later in this paper, and is hereafter called the *job-truck assignment problem*. The cost function for each individual is evaluated and the genetic algorithm proceeds as depicted in Fig. 2



Input available demands r ∈ {1,...,R}
Subdivide demands into jobs
Job-PC assignment (GA)
Selection
Crossover
Mutation
Fix infeasible solutions (DSTA)
Job-Truck assignment (TSTA)
Fitness evaluation

Fig. 2. Concrete supply chain optimization.

A. Job-PC assignment problem

The first problem consists of assigning the demands to the PCs. This assignment is done through the GA. The chromosomes contain two parts, both containing R elements (number of demands r), as shown in Fig. 3, for an example of $R = 6$. The first part (in black) defines the assignment of demands to PCs, with each gene being an integer between 1 and D number of production centers. The second part of the chromosome (in gray) establishes the sequence in which the R requests are produced. This second part of the chromosome is a permutation between 1 and R.

Job-PC assignment						*Job production sequence*					
1	3	1	2	4	5	1	4	2	6	3	5

Fig. 3. Example chromosome for $R = 6$, with 5 PCs.

The GA implementation is straightforward. The initial population of 100 individuals is randomly created. The selection is done through the tournament method, with two individuals for each tournament. The crossover method is applied to couples of two individuals and was introduced in [5]. A single crossover point is randomly selected: if it falls on the first part of the chromosome, it performs a single point crossover on the first part of the chromosome; otherwise it performs an order-based crossover on the remaining part. For further details, see also [8]. The mutation works as the crossover method: a mutation point is chosen and the value of that gene is swapped with the value of another gene of the same part of the crossover. The algorithm runs for 200 generations.

Notice that many of the GAs solutions are not feasible, due to constraint violations. These violations include: solutions where the unloading starts before the customers specified EDT or end after $Tset$; and solutions that describe production sequences that cannot be followed. In these cases, the solutions are fixed following the depot scheduling construction heuristic (DSCA), that consists mainly on shifting the demands forward or backward in time within the same PC or, in case it is not possible, reassign the jobs to the nearest closest PC to the customer. For a detailed description, please see [5].

B. Job-Truck assignment problem

When all the GA solutions have been fixed into feasible solutions, the jobs have to be assigned to trucks. In [5], this is done through the truck scheduling construction heuristic (TSCA).

In a first phase, the heuristic tries to assign the jobs produced at a given PC to the fleet of vehicles based on the same location. This set is composed by trucks that either have not left yet the base PC or have already completed some transport operations and can return to the PC at time instant $t < STL$. When both types of trucks are available, the TSCA always tries to assign the jobs to the previously used vehicles first, in order to use the minimal amount of trucks for servicing all the requests - this assignment strategy is referred to as *Shortest Idle Time*, because the truck with the smallest idle time at the PC is the one assigned first. This strategy not only minimizes the number of used trucks, but also avoids an evenly distribution of idle times, which can be useful since it allows trucks with long idle times to deliver jobs at other PCs.

At the end of this phase, there are still some unassigned jobs to trucks, usually because there are PCs that do not have its own fleet of trucks. At this stage, the heuristic assigns those jobs in order of increasing SLT, using the idle times of the used trucks or the trucks that were not yet used, sorted by completion time of the last operation. If no truck is available, a new one is hired.

The assignment of jobs to trucks is also a combinatorial problem and the described heuristic cannot efficiently search for an optimal solution in the large solution space of this problem. Therefore, this paper proposes the replacement of this heuristic by an Ant Colony Optimization algorithm.

III. Job-Truck assignment using ACO

The Ant Colony Optimization (ACO) methodology is an optimization method suited to find minimum cost paths in optimization problems described by graphs. ACO has proved to be a competitive meta-heuristic for many different types of combinatorial optimization problems [9].

The job-truck assignment problem is a routing problem in the sense that aims to minimize the routing costs, traveling as less distances and using as less trucks as possible. However, this problem can be seen as assigning tasks to resources, which makes this problem a typical job shop scheduling problem, where the machines are in this case trucks. Therefore, the ACO approach to solve this problem is based on the ACO implementation for job-shop problems proposed in [10].

The problem can be described by a graph as in Fig. 4. The

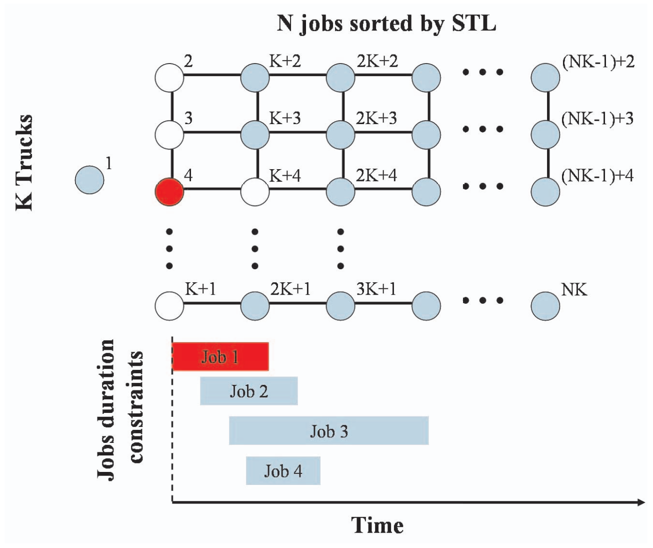


Fig. 4. Nodes model

nodes x of the graph represent all the possible assignments of jobs $i = 1, \ldots, N$ to trucks $j = 1, \ldots, K$. Each row represents one of the K trucks (including possible trucks to hire) and each column represents one of the N jobs, sorted by STL values. The total number of nodes is $N \times K + 1$, where node 1 represents a dummy source node. Figure 4 also represents the duration of the jobs.

The ACO algorithm consists usually of letting the ants wander on the graph arcs, such that each ant finds a path that describes a feasible solution. However, as described in Section I, the problem can easily reach large dimensions, and the graphs may have 15000 nodes (300 jobs $\times$ 50 trucks). Therefore, the algorithm does not consider walks on $(NK+1)\times(NK+1)$ arcs of the graph, which would make the algorithm implementation prohibitive. Instead, it only considers the ants visits to the nodes. Nevertheless, the initial number of nodes can be very high, but, with the construction of the solution, the graph size can be significantly reduced.

Consider that the first node chosen by the ants is node $x = 4$ in Fig. 4 (represented in black). This node represents the assignment of job 1 to truck 3. Considering the job duration, node $K+4$ (job 2 assigned to truck 3, represented in white) cannot be further assigned to the same truck, since job 2 has to be loaded before the end of the job 1. Therefore, the next job that can be assigned to truck 3, is job 3 (node $2K+4$). Notice also that each job is selected only once, thus, no more trucks can pick that specific job. For example, as soon as job 1 is assigned to truck 4, this job can no longer be assigned to truck 1 (node 2) or truck 2 (node 3) or truck K (node $K+1$). That corresponds to the elimination of all the nodes in the first column, as represented in Fig. 4, also in white. Therefore, with the selection of a job, many nodes are automatically eliminated.

A. The algorithm

The ant colony optimization algorithm constructs the solution based on the pheromone deposit information τ and the heuristic information η specific to the problem, which are, in this case, concentrated on the nodes. These matrices have dimensions $1 \times KN$ and are always defined in the interval $[0,1]$. Observe however, that in this implementation (as explained in Section III-B), the heuristic on the nodes depends on the ant, thus, each l ant has its own heuristic matrix η^l. The algorithm works as follows. Consider a colony of g ants, initially located at node 1. The probability of an ant l, with $l = 1,\ldots,g$ located in node x to choose node $x+1$ is given by

$$p_x^l(t) = \begin{cases} \frac{{\tau_x}^{\alpha}\cdot{\eta_x^l}^{\beta}}{\sum\limits_{y\notin\Gamma^l}^{KN}{\tau_y}^{\alpha}\cdot{\eta_y^l}^{\beta}} & \text{if } k \notin \Gamma^l \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where Γ^l is the tabu list of ant l, i.e. the list of nodes that cannot be visited by this ant. This list includes the list of nodes already visited, or the ones that do not satisfy the different constraints. When all the g ants have found a complete solution and scheduled all the jobs, the pheromone matrix τ is updated by

$$\tau(t+1) = \tau(t) \times (1-\rho) + \Delta\tau^q \tag{4}$$

where $\rho \in [0,1]$ expresses the pheromone evaporation phenomenon and $\Delta\tau^q$ are the pheromones deposited on the nodes x followed by the ant q that found the best solution for the cost function f defined as in (2):

$$\Delta\tau^q \begin{cases} \frac{\omega}{f^q} & \text{if node } x \text{ was used by the } q \text{ ant} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The parameter ω is a normalization weight for the cost function value. The pheromone update mechanism balances the reinforcement of good solutions with the evaporation of old solutions, in order to avoid early convergence to local optima. Since $\tau \in [0,1]$, if the cost function f solutions have a high order of magnitude, e.g. 10000, the pheromone update does not compensate the pheromone evaporation phenomena. On the other hand, if the cost function solutions have a low value, e.g. 0.001, there is hardly no evaporation and early converge will happen. To allow a balanced evaporation and pheromone reinforcement, $\omega < f^q$ must be used.

The pseudo-code of the algorithm is as follows:

```
1) INITIALIZATION:
   a) Initialization of the pheromone matrix $\tau$
   b) Initialization of the $g$ heuristic matrix $\eta^l$;
   c) Initialization of the $g$ tabu lists $\Gamma^l$;
2) FOR the number of iterations specified:
   a) Place the $g$ ants at node 1 in the tabu lists;
   b) FOR $l = 1,\ldots,g$:
      i) FOR $x = 1,\ldots,KN$
         A) Choose the next node to visit using (3);
         B) Update the solution $f^l$ of the ant;
         C) Update the tabu list $\Gamma^l$;
         D) Update the heuristic matrix $\eta^l$;
   c) Evaluate the fitness of the $g$ solutions.
3) Find $f^q = \min(f^l)$;
4) Update the $\tau$ matrix using using (4) and (5);
5) Select the best solution;
```

The pheromone matrix τ is initialized with its maximum value 1. The initialization and the update of the η^l and Γ^l matrices are described in detail in the next sections.

B. The heuristic matrix η

The objective of the heuristic matrix η^l, is to provide a greedy guidance to each ant in order to improve the solution. In our implementation, each node of the matrix is initialized with one of three different constants η_O, η_N, η_D, all defined in $\eta = [0,1]$, that describe different truck states:

1) *IF node corresponds to a hired truck, then* $\eta_x^l = \eta_O$;
2) *ELSE IF node corresponds to jobs produced at the PC were the truck is placed, then* $\eta_x^l = \eta_D$;
3) *ELSE,* $\eta_x^l = \eta_N$;

During the optimization process, it is important to observe the *Shortest Idle Time* heuristic, that assigns the jobs to vehicles that have been previously used, in order to use the minimal amount of trucks for servicing all the requests (see Section II-B and [5]). This information has to be updated during the solution construction by each l ant, through constant η_T. Therefore, each ant updates its η^l matrix according to the rule:

1) After a truck has been assigned, all the nodes that correspond to jobs produced in the same PC of the added job are set to $\eta_x^l = \eta_T$.

In order to respect the heuristics relative importance, the constants must respect the inequalities

$$\eta_O << \eta_N < \eta_D < \eta_T. \tag{6}$$

C. The Tabu matrix Γ

The Tabu list Γ^l describes all the nodes that cannot be visited by ant l. The update of this matrix follows the next rules (imposed by the constraints):

1) Elimination of the nodes concerning the assignment of other trucks to the selected job (column elimination);
2) Elimination of the nodes representing jobs that cannot be attended by the selected truck, because they start before the truck has finished the actual job (partial row elimination).

IV. EXPERIMENTAL RESULTS

The new truck assignment algorithm was tested using two different test instances, with increasing complexity:

- Instance T_1 consists of $R = 5$ demands to be produced in 2 different PCs, that generate 16 jobs, with 1 truck located at PC_1 and 5 trucks at PC_2;
- Instance T_2 has a total of 10 demands to be produced in 2 different PCs, that generate 169 jobs. The trucks are distributed as for Instance T_1.

In order to compare directly the constructive heuristic (TSTA) performance with the ACO implementation, the ACO is used only with the best GA solution. That also saves a lot of computational time, since the computational effort of the ACO meta-heuristic is much higher than the computational effort of the TSTA heuristic. The results refer to the best solution found by each methodology, for both test instances.

A. ACO parameters sensitivity analysis

The proposed ACO algorithm has many parameters that have to be set: α, β, ρ; the heuristic constants η_O, η_N, η_D and η_T; the cost function normalization factor ω; and the number of g ants, in terms of $\%$ of total number of nodes. To find the best parameter set for the ACO algorithm, we performed a sensitivity analysis: an initial parameters setting was defined and then different values for each parameter were tried at a time. The initial set was: $\alpha = 1, \beta = 1, \rho = 0.1$, $\eta_N = 0.5, \eta_O = 0.01, \eta_D = 0.7, \eta_T = 0.8$, $\omega = 500, \omega/f^q \approx 0.1$ and $g = 5\%$. The η_N value, the visibility of nodes of jobs produced in PCs with no trucks, is fix for all the analysis. The remaining constants were tuned using this value and rule (6). All the tests were performed on instance T_1. The sensitivity analysis is presented in Table I, in terms of cost function f as in (2), for each parameter variation.

For the α and β parameters, it is clear that the algorithm performs better if $\alpha < \beta$: in the row varying α (with $\beta = 1$), the best results are achieved for $\alpha < 1$; when β varies (with $\alpha = 1$), the best results are achieved for $\beta > 1$. Observe however that when $\alpha \ll \beta$, the optimization performance degrades, since in this case, the algorithm is following mostly the heuristic information.

The evaporation coefficient ρ analysis shows that it must be low (0.1), but if it is to low, the algorithm suffers from early convergence to a local optima.

For the heuristic matrix constant analysis, and assuming that $\eta_N = 0.5$, it is obvious that the η_O value must be very low, i.e. the nodes corresponding to outsourced trucks must be avoided by the ants. Notice further that for $\eta_O = 0.7$, i.e. $\eta_O > \eta_N$ the optimization result is extremely bad. For constant $\eta_N = 0.5$, it is also visible that the best values are obtained when $\eta_D = 0.7$, i.e. $\eta_N < \eta_D < \eta_T$. Finally, for constant $\eta_N = 0.5$, the best result of the algorithm are obtained for $\eta_T \geq \eta_D$. All these results confirm the heuristic relative importance rule defined in (6).

The ω normalization weight shows that the ratio ω/f must be low. As soon as the pheromone reinforcement mechanism is too high, the algorithm converges to sub-optimal solutions.

The total number of ants g analysis shows that it must represent at least 25% of the total number of nodes. It also shows that the use of more ants does not improve the algorithms performance, while the computational effort is much higher.

The final parameters set is: $\alpha = 0.5$, $\beta = 1.0$; $\rho = 0.1$, $\eta_O = 0.008$, $\eta_N = 0.5$, $\eta_D = 0.6$, $\eta_T = 1.0$; $g = 25\%$. For instance T_1, $\omega = 500$ and for instance T_2, $\omega = 10000$. The total number of iterations is 200.

TABLE I
SENSITIVITY ANALYSIS

α	0.1	0.5	0.8	1.5	2
f	7855	3655	5140	6975	8440
β	0.1	0.5	1.5	2	4
f	16280	5240	3760	3625	3760
ρ	0.01	0.05	0.1	0.2	0.3
f	8890	4005	3625	8454	8565
η_O	0.008	0.1	0.7		
f	3625	5060	21980		
η_D	0.1	0.6	0.7	1	
f	7015	4255	3655	4190	
η_T	0.1	0.7	0.8		
f	11310	3790	3625		
ω/f	0.1	0.15	0.2		
f	3655	3790	5090		
g	5%	10%	25%	100%	
f	5240	4255	3625	3625	

TABLE II
COMPARISON RESULTS: INSTANCE T_1

Method	TSCA heuristic	ACO
Best f	6700	3625
Time [s]	1	133.03

B. Instance T_1

The results for instance T_1 are presented in Table II. The ACO algorithm is able to find a solution that represents almost a 50% reduction of costs, when compared to the solution proposed by the heuristic method. As expected, the major drawback is that the computational time required by the ACO algorithm is much higher.

Figure 5 shows further that the best solution is achieved around iteration 100, which means that in practice the ACO algorithm needs half of the computational time indicated in Table II. Further, it also shows that the ants first solutions of the ACO algorithm is worse than the heuristic. This also means that if the ACO algorithm uses explicitly the heuristic solution, the computational effort can be reduced.

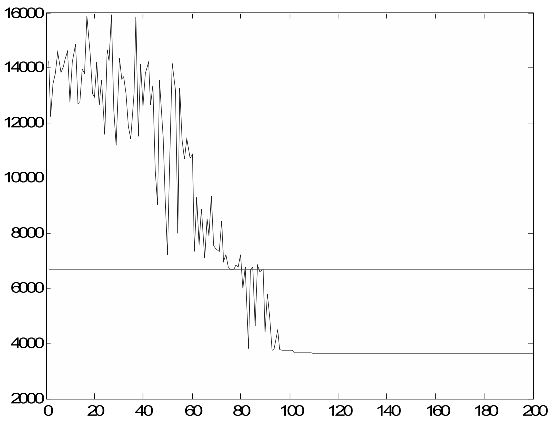


Fig. 5. Instance T_1

TABLE III
COMPARISON RESULTS: INSTANCE T_2

Method	TSCA heuristic	ACO
Best f	101195	81945
Time [s]	12	28950

C. Instance T_2

The results for test instance T_2 are similar. Although the problem complexity increases significantly, the ACO algorithm is able to find a solution that is 20% better than the solution found by the heuristic. Notice however that this reduction is in absolute value much higher than the 50% reduction obtained for instance T_1. This means that for large problems, even if the improvement is not high in relative terms, it represents large cost saves in absolute terms. Again, the results show that the ACO algorithm computational effort drastically increases with the optimization problem size. However, as shown in Fig. 6, if the ACO algorithm used the solution provided by the heuristic, a lot of computational effort could be saved.

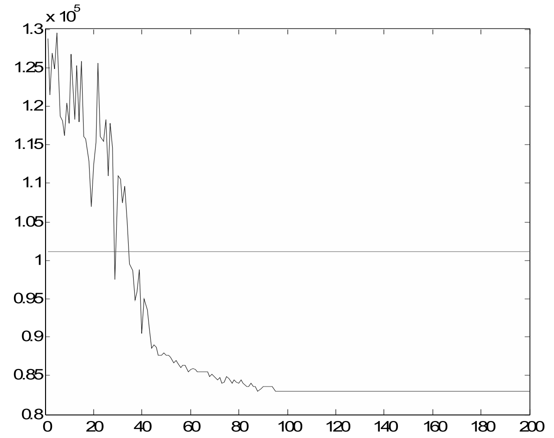


Fig. 6. Instance T_2

V. CONCLUSIONS AND FUTURE WORK

This paper introduced a new combination of GA and ACO algorithms for the optimization of a concrete delivery problem. This problem can be seen as a composition of two problems, the job-production center assignment and the job-truck assignment combinatorial problems. The problem had been solved in [5], through an hybrid GA-heuristic approach. The approach proposed in this paper is an evolution of that algorithm, by replacing the heuristic that solves the job-truck assignment problem by an ACO algorithm.

ACO algorithms have proven in the last decade that they are competitive meta-heuristics for optimization problems that can be modeled in a graph environment. In this problem, the replacement of the heuristic with the ACO algorithm led to significant improvements of the final solution, which means in practice a large reduction of the concrete delivery costs. However, this improvement was obtained with a drastic increase of the computational time. Nevertheless, as observed, the computational cost of the ACO algorithm can be reduced, which is already under development.

The future research direction is the comparison of the GA implementation for the job-PC problem with an ACO implementation. The objective is to optimize the concrete supply chain using the distributed optimization paradigm proposed in [11], which obtained good results for other supply chains examples.

REFERENCES

[1] M. Barbuceanu and M. Fox, "Coordinating multiple agents in the supply chain," in *Proceedings of the Fifth Workshops on Enabling Technology for Collaborative Enterprises, WET ICE'96*. IEEE Computer Society Press, 1996, pp. 134–141.

[2] N. Viswanadham, "The past, present, and future of supply-chain automation," *IEEE Robotics & Automation Magazine*, vol. 9, no. 2, pp. 48–56, 2002.

[3] N. F. Matsatsinis, "Towards a decision support system for the ready concrete distribution system: A case of a greek company," *European Journal of Operational Research*, vol. 152, no. 2, pp. 487–299, 2004.

[4] I. Tommelein and A. Li, "Just-in-time concrete delivery: Mapping alternatives for vertical supply chain integration," in *Proceedings of the Seventh Annual Conference of the International Group for Lean Construction IGLC-7*, 1999, pp. 97–108.

[5] D. Naso, M. Surico, B. Turchiano, and U. Kaymac, "Just-in-time production and delivery in supply chains: a hybrid evolutionary approach," in *Proceedings of the IEEE SMC 2004, International Conference on Systems, Man and Cybernetics*, 2004, pp. 1932–1937.

[6] O. Bäysy, "Efficient local search algorithms for the vehicle routing problem with time windows," in *Proceedings of MIC'2001 - 4^{th} Metaheuristics International Conference*, 2001.

[7] L. M. Gambardella, Éric Taillard, and G. Agazzi, "MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. McGraw-Hill, 1999, pp. 63–76.

[8] A. Nearchou, "The effect of various operators on the genetic search for large scheduling problems," *International Journal of Production Economics*, vol. 88, no. 2, pp. 191–203, 2004.

[9] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA: MIT Press/Bradford Books, 2004.

[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: Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[11] C. Silva, J. Sousa, T. Runkler, and J. S. da Costa, "A multi-agent approach for supply chain management using ant colony optimization," in *Proceedings of the IEEE SMC 2004, International Conference on Systems, Man and Cybernetics*, 2004, pp. 1938–1943.