Performance Comparison of Ant Colony Algorithms for the Scheduling of Steel Production Lines

Silvino Fernández, Segundo Álvarez, Eneko Malatsetxebarria, Pablo Valledor and Diego Díaz
ArcelorMittal Global R&D
P.O. Box 90 - 33400
Avilés, Asturias, Spain
silvino.fernandez@arcelormittal.com,
segundo.alvarez-garcia@arcelormittal.com,
eneko.malatsetxebarria@arcelormittal.com,
pablo.valledor-pellicer@arcelormittal.com,
diego.diaz@arcelormittal.com

ABSTRACT

Swarm Intelligence metaheuristics, and among them Ant Colony Optimization (ACO), have been successfully applied worldwide to solve multiple examples of combinatorial NP-hard problems, giving good solutions in a reasonable period of time (an essential requirement in real life applications). Our company is dedicated to produce steel, being the biggest steelmaker in the World. The impact of a good sequencing in our daily production results is critical. Sequencing requires to take into consideration the most relevant parameters that influence along the material transformation process, being this a really difficult task to do manually or with other traditional methods. After the successful use of the Ant System (AS) algorithm to sequence several of our Galvanizing production lines, we have studied the potential benefit of moving from the classical AS algorithm to an Ant Colony System (ACS), a promising variation of the ACO metaheuristics family. In this paper, we present the work done in order to apply the ACO techniques to an industrial problem. More specifically, we exhibit a comparison between AS and ACS variants, the results obtained and the conclusions drawn from such comparison.

Categories and Subject Descriptors

G.2.1 [**Combinatorics**]: Combinatorial algorithms

Keywords

Decision Making; Manufacturing; Ant Algorithms; Swarm Intelligence; Algorithms; Scheduling; Ant Colony Optimization; Ant Colony System; Combinatorial Optimization; Steel Industry; Metaheuristics

GECCO'15 Companion, July 11–15, 2015, Madrid, Spain.

ACM ISBN 978-1-4503-3488-4/15/07..

DOI: http://dx.doi.org/10.1145/2739482.2764658

1. INTRODUCTION

Hundreds of steel coils are processed daily in our galvanizing lines. Their sequencing is critical to avoid incidents in the facility and to reduce the cost of the process. The aim is to smooth transitions from one coil to another. Otherwise, quality losses or productivity slowdowns may be induced, or even worse, a strip breakage may occur, stopping production for several hours or even for more than a day.

At our galvanizing operations, an Ant System Algorithm [3] has been developed for estimating sequencing-related losses and searching for the sequence that minimizes them [4].

An important hard constraint imposed to the model concerns execution time since it must be able to provide a schedule within a few minutes, in order to be useful to the line. This execution time constraint, together with the complexity of the cost functions that estimate the quality of the solution, result in a quite restrictive limitation on the number of schedule evaluations, a much tighter limitation than usual in systems aimed at solving combinatorial problems.

This approach is named ACO on a budget in [2], which presents a study of the different variants of the ACO algorithm to face the limitation of solving the problem with relatively few evaluations of the solutions due to execution time constraints.

Ant Colony System is one of the most famous variants of the Ant Colony Optimization algorithm. It has three main differences [5] with the original Ant Colony, which can be summarized in: the State Transition Rule, Global Pheromone Updates and Local Pheromone Updates.

2. ANALYSIS AND CONCLUSIONS

For the experimental analysis and comparison of AS and ACS, we use 45 benchmark instances from actual production data in our galvanizing lines. This benchmark consists of 3 sets of 15 instances of each size 30, 60, 90, taking size as the number of coils to schedule.

As a stopping criterion, due to the limitations imposed by our production environment, we halt after a maximum number of solutions evaluated, namely a number of 10 ants and 100 iterations (*budget* of 1000).

For a fair comparison of the algorithms, we first fine-tuned their parameters: α, β, and ρ for AS; and α, β, ρ, $q0$ for

Table 1: Optimal parameters calculated by iterated F-Race

Algorithm	α	β	ρ	$q0$
AS	1.52	2.52	0.22	-
ACS	1.75	3.69	0.14	0.43

Table 2: Number of times in which the best solution found by each ACO variant outperformed the other, out of 15 instances

Problem size	ACS best solution	AS best Solution
30	11	4
60	7	8
90	7	8

ACS. We used a set of 15 benchmarking instances, 5 for each size mentioned above, distinct from the test instances mentioned above. The ranges of possible values were $\alpha \in [1, 3]$, $\beta \in [1, 4]$, $\rho \in [0, 0.6]$, and $q0 \in [0, 1]$. We used the iterated F-Race method [1], provided by the irace [6] package for the automatic tuning process, configured to evaluate a maximum of 4800 experiments.

Table 1 shows the optimal configurations derived by the iterated F-Race method.

All the code has been written in $C\#$, compiled using Microsoft Visual Studio .NET 2010, and run on an Intel Xeon X5675 CPU 3.07 GHz with 16 GB of RAM, under a Windows 7 operating system.

We compare ACS and AS in terms of best, worst, median and standard deviation of the objective function for 30 independent runs of each algorithm on each instance. Figure 1 shows the results for the size 30 problem.

Additionally, we apply the two-tailed Wilcoxon signed-rank test to assess the significance of the algorithms comparison [7]. At the default 5% significance level, the test fails to reject a null hypothesis of zero median in the difference.

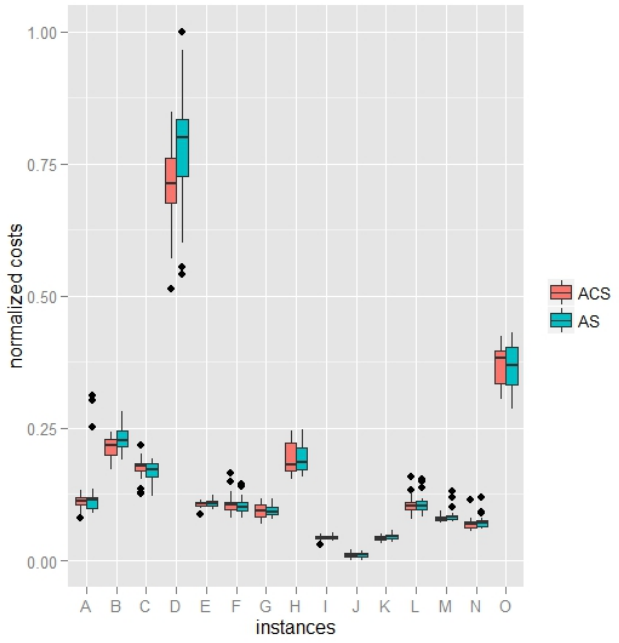

Figure 1: Normalized costs for different instances of size 30

Table 2 compares the results considering only the best solution found by each algorithm per size of the problems. Sets of size 60 and 90 have a similar performance, whereas for 30 coil sets a clear advantage is found for ACS.

Based on an industrial benchmarking dataset built from production data, we can conclude that for our context there is no significant difference between AS and ACS in medium and large problems (60 and 90 coils), although for small size problems (30 coils) ACS outperforms AS.

From the practical point of view of our industrial scope, these results suggest evolving our current solution from AS to ACS in order to benefit from the better results in small size problems.

3. REFERENCES

[1] P. Balaprakash, M. Birattari, and T. Stützle. Improvement strategies for the f-race algorithm: Sampling design and iterative refinement. In *Hybrid Metaheuristics*, pages 108–122. Springer, 2007.

[2] L. P. Cáceres, M. López-Ibáñez, and T. Stützle. Ant colony optimization on a budget of 1000. In *Swarm Intelligence - 9th International Conference, ANTS 2014, Brussels, Belgium, September 10-12, 2014. Proceedings*, pages 50–61, 2014.

[3] M. Dorigo and L. M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on*, 1(1):53–66, 1997.

[4] S. Fernández, S. Alvarez, D. Díaz, M. Iglesias, and B. Ena. Scheduling a galvanizing line by ant colony optimization. In *Swarm Intelligence*, pages 146–157, Brussels, 2014. Springer.

[5] A. Kaveh and S. Talatahari. A novel heuristic optimization method: charged system search. *Acta Mechanica*, 213(3-4):267–289, 2010.

[6] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. The irace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.

[7] W.-l. Xiang, M.-q. An, Y.-z. Li, R.-c. He, and J.-f. Zhang. An improved global-best harmony search algorithm for faster optimization. *Expert Systems with Applications*, 41(13):5788–5803, 2014.