Criticality of Response Time in the usage of Metaheuristics in Industry

Silvino Fernández
ArcelorMittal
Global R&D Asturias
P.O. Box 90 - 33400 Avilés - Spain
silvino.fernandez@arcelormittal.com

Pablo Valledor
ArcelorMittal
Global R&D Asturias
P.O. Box 90 - 33400 Avilés - Spain
pablo.valledor-pellicer@arcelormittal.com

Diego Díaz
ArcelorMittal
Global R&D Asturias
P.O. Box 90 - 33400 Avilés - Spain
diego.diaz@arcelormittal.com

Eneko Malatsetxebarria
ArcelorMittal
Global R&D Asturias
P.O. Box 90 - 33400 Avilés - Spain
eneko.malatsetxebarria@arcelormittal.com

Miguel Iglesias
ArcelorMittal Global R&D Asturias
P.O. Box 90 - 33400 Avilés - Spain
miguel.iglesias@arcelormittal.com

ABSTRACT

Metaheuristics include a wide range of optimization algorithms. Some of them are very well known and with proven value, as they solve successfully many examples of combinatorial NP-hard problems. Some examples of Metaheuristics are Genetic Algorithms (GA), Simulated Annealing (SA) or Ant Colony Optimization (ACO). Our company is devoted to making steel and is the biggest steelmaker in the world. Combining several industrial processes to produce 84.6 million tones (public official data of 2015) involves huge effort. Metaheuristics are applied to different scenarios inside our operations to optimize different areas: logistics, production scheduling or resource assignment, saving costs and helping to reach operational excellence, critical for our survival in a globalized world.

Rather than obtaining the global optimal solution, the main interest of an industrial company is to have "good solutions", close to the optimal, but within a very short response time, and this latter requirement is the main difference with respect to the traditional research approach from the academic world. Production is continuous and it cannot be stopped or wait for calculations, in addition, reducing production speed implies decreasing productivity and making the facilities less competitive. Disruptions are common events, making rescheduling imperative while foremen wait for new instructions to operate. This position paper explains the problem of the time response in our industrial environment, the solutions we have investigated and some results already achieved.

General Terms

Algorithms, Experimentation.

GECCO'16 Companion, July 20-24, 2016, Denver, CO, USA

DOI: http://dx.doi.org/10.1145/2908961.2931649

Keywords

Decision Making, Manufacturing, Ant Algorithms, Swarm Intelligence, Ant Colony Optimization, Ant Colony System, Combinatorial Optimization, Algorithms, Metaheuristics, Steel Industry, Scheduling, Auto Tuning.

1. INTRODUCTION

The production of steel is a very complex process, with several stages involved in its transformation from coal and iron ore to rail or steel coils: iron making (conversion of iron ore into liquid pig iron), steelmaking (conversion of liquid pig iron into liquid steel), casting (solidification of liquid steel into semi-products: billets or slabs) and finally rolling. Under the scope of several facilities existing in the steel industry, a huge number of variants of combinatorial problems appear, requiring a quick answer to sudden production changes. An implicit hard constraint for any real problem coming from the intended use of a scheduling optimization model is the execution time. Usually it is very limited, since it must be able to provide decisions within a few minutes in order to be useful for production.

In this context, there is a wide range of examples to show the importance of the use of Metaheuristics in industrial problems. (Fernandez et al., 2014) and (Díaz et al., 2014) introduce two examples on how to use the Ant Colony Optimization algorithm (Dorigo, 1992) to optimize the scheduling and the cutting plan of steel making plants, respectively; (Fernández et al., 2006) show how to use Genetic Algorithms to optimize logistic movements of a train fleet in order to increase their productivity while covering the production necessities. These solutions provide several advantages such as objective criteria to schedule the operations independently of subjective human opinion, fast reaction under sudden incidents, avoidance of human errors, and powerful computation capacities to explore solutions.

Normally, models are the result of the collaboration between research experts in mathematical models and optimization techniques, and the production experts who are in charge of defining the problem itself and the way of evaluating the different solutions (fitness functions). The quality and accuracy

of the model depends on this collaboration. However, in most cases, there is an implicit restriction as a consequence of the industrial scenario: response time. Facing an incident in a continuous facility that never stops means to reschedule the line-up in few minutes (or seconds) to keep the production going and to assure the levels of productivity needed to reach operational excellence, key to survive in a globalized world as it is today.

In the next sections of this position paper, we introduce our research lines to reduce the response time of our algorithms in operation to meet the requirements of the production facilities while respecting the constraints and assuring a good level of quality for the results.

2. RESPONSE TIME

We can generically define response time as the total amount of time a system takes to respond to a request for a service. In the context of metaheuristics in industry, it would be the time spent by the algorithm since the foreman has loaded the input and requested a decision. This time ***t*** depends on the parameters shown in the following equation:

$$t = t_0 + i * s * (t_s + t_e + t_u) + t_p \qquad (1)$$

Where: t_0: Setup time for the algorithm.

i: number of iterations of the search.

s: number of solutions explored per iteration.

t_s: time to create the solution.

t_e: time to evaluate the solution.

t_u: time to process the solution.

t_p: presentation of the final solution to the user.

So, it is possible to reduce the response time by means of reducing any of the parameters in equation (1). We have classified these parameters in two different sets: a first set, called Intelligent Set (IS), covering the configuration parameters of the algorithm, where we include i and s. The second set would be the Speed Set (SS), considering machine and architecture limitations, where we include t_s, t_e and t_u (we consider t0 and tp are not significant in our cases).

Acting on IS can be done directly changing (reducing) the parameters of the algorithm because they are an input. In the case of Genetic Algorithms, i would be the number of generations and s the number of individuals generated and evaluated at each generation. In ACO, they would the number of iterations and the number of ants, respectively. Decreasing these parameters will reduce the search and hence it can affect the quality of the solution, but it could help to reduce the response time significantly. To assure the same level of solution quality, it is necessary to make the algorithm *smarter* so that it can reach comparable solutions before being stopped.

On the other hand, reduction of the SS parameters is limited by the characteristics of the computers. More powerful servers reduce response time executing the instructions of the algorithm faster, but it means an investment, sometimes difficult or not possible to approve. Another alternative is to use parallelism to execute (parts of) the algorithm concurrently, gaining time.

3. ALTERNATIVES TO REDUCE RESPONSE TIME

As commented, we identify two main approaches to reduce the response time for each of the groups:

- Execute the instructions faster (SS).
- Make the search more intelligent (efficient) to obtain the same quality of the solution but faster, with less solution evaluation (IS).

3.1 Speed Set

There are two main approaches to increase the speed of the calculations: the first one (obvious) to invest in the computers where the models run, but in some contexts this is very difficult and the server where the solution is executed is not a state-of-the-art computer. Other possibility is to use parallelism techniques in the code of the models to accelerate calculations (in case of availability of a multi core machine or a high-performance computing cluster).

3.2 Intelligent Set

Regarding the number of solutions explored, there are several alternatives to reduce them, maintaining the same quality level. The main ones are:

- Improve the configuration of the algorithm
- Improve the search strategy

The configuration of the algorithm is very important to reduce the search time, but it is not easy to obtain a generic parameter set that achieves the best result for every instance. In industry, operators continuously generate instances, and results are required in a short time. Therefore, configuration analysis cannot be done on the fly, as it is too time consuming, and hence, a previous analysis is necessary to classify the instance as soon as it is created.

Ant Colony Optimization (ACO) is a family of algorithms inspired in the way the ants look for food in nature and they transmit the information about the success of the search to the following ants. The first version was called Ant System. Since then, many variants have been developed showing better performance depending on the problem. The normal way of evaluating performance is to measure and compare the quality of the solutions for a specific problem in a limited time period. This time period is a key factor to determine whether a variant is suitable, because it must meet the time requirements of the context. (Cáceres, López-Ibánez, and Stützle, 2014) present a study to determine the best ACO variant for a problem under the limitation of 1000 evaluations (in the end, a computation time limit). Following this approach, we have evaluated the different variants on one of our problems. Results are shown in the following sections.

4. AUTO CONFIGURATION OF PARAMETERS

Finding the most appropriate parameter settings for an optimization problem and for a given set of heterogeneous instances is critical to obtain good results in a short (valid) calculation time in industry. There are several approaches to calculate a suitable configuration. Irace is a software package developed by (López-Ibánez et al., 2011). Irace implements the Iterated F-Race method, generating different candidates of

parameter configurations for a specific algorithm and then performing a race of those candidates on distinct problem instances, that is, all candidates are run on a predefined number of instances before the first statistical test is performed to remove the worst candidates. After that, the test is performed each time after all remaining candidates have been run on the next instance until the iteration budget is exhausted or only a few candidates remain. Usually, multiple races are performed containing the previous best candidates and newly generated ones until the total budget is exhausted. As results Irace provides the optimal candidate configurations proposed by the iterated F-Race method.

We carried out an experiment, classifying the instances of one of our models to optimize the scheduling of production facilities. The model sequences the coils according to two criteria to minimize the objective function: costs (losses) of the transitions between coils, and constraints (large penalties), when there is not a possible transition that make the foremen to introduce transition coils in the middle to make it feasible. The classification criteria in our experiment was to study the characteristics of the steel coils (items) to produce in each sequence (instance), analyzing the average and standard deviation of each parameter. By means of a correlation matrix and hierarchical cluster analysis, we have classified the instances into 12 different clusters using these variables as criteria; they would be the input for Irace in order to obtain the best configuration for each one. For each cluster, we run IRace to obtain the best configuration for an ACO algorithm (**α, β, ρ**). Afterwards, we compare the results for a set of instances of each cluster versus the results of the algorithm for the same instances with a default configuration (α=1, β=2 and ρ=0.2, respectively).

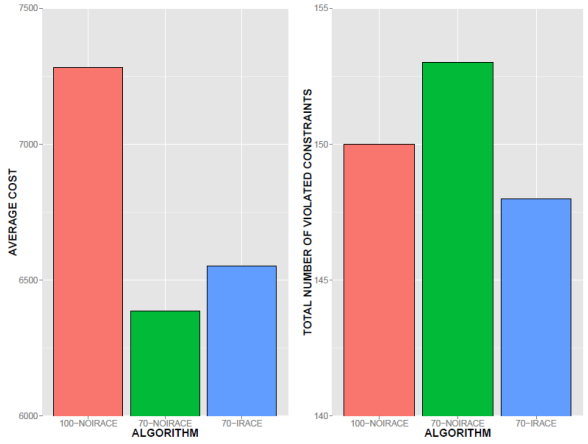


Figure 1: Impact of Clustering in the Results of the model.

Figure 1 shows the comparison among the three tests done. 100-NOIRACE is the test with default parameter settings and the normal solution evaluations. 70-NOIRACE uses the same parameters, but only 70% of the evaluations. Finally, 70-IRACE uses the parameters fine-tuned with Irace after the classification of the instance in a representative cluster according to these parameters with 70% of the evaluations. The chart on the right shows the number of hard constraints violated by the solutions, when large penalty is invoked (their reduction is the first objective in the optimization). The one on the left shows the average of the rank cost of the solutions. As shown, the experiments with parameters pre-calculated by Irace improve significantly the quality of the results (note that reducing hard constraints justifies higher costs). This could allow a reduction of the number of evaluations and thus the calculation time.

It would also be possible to combine this approach with the selection of the best ACO strategy as shown in the previous section, selecting for each cluster the best configuration and the best strategy. For this study, we used Ant System.

5. VARIANTS TO MAKE THE SEARCH SMARTER

In (Fernández et al., 2015) we presented our first analysis of the potential improvements in terms of response time (and solution quality) by means of using different approaches in the strategy of the search of an Ant Colony Optimization. In these experiments, we proved for a specific model that even though the results can be better in some cases, the improvement was not very significant. In any case, we implemented all the variants to perform further analyses of the well-known ACO algorithms. Figures 2 and 3 show the results of each variant on instances of 60 and 90 steel coils to schedule.

Figure 2: Performance of different ACO algorithms for several instances of 60 coils.

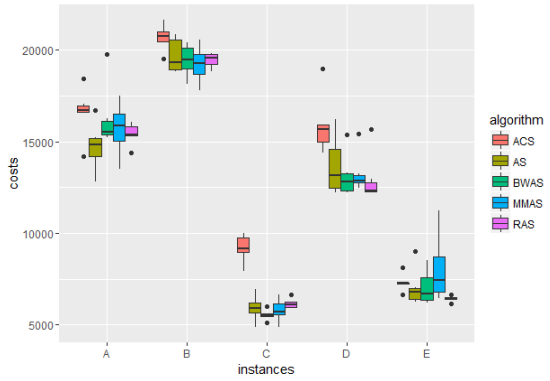
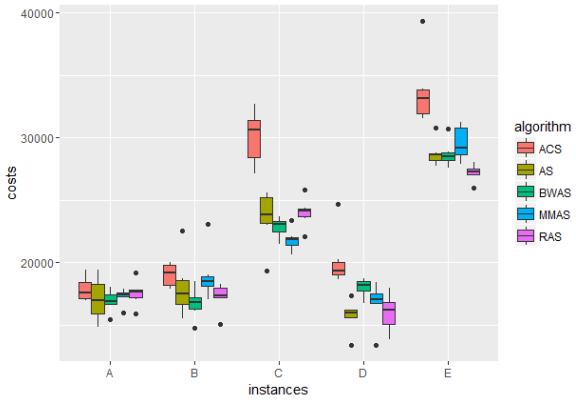


Figure 3: Performance of different ACO algorithms for several instances of 90 coils.



The chart above shows clear variability in the relative performance of the variants with the problem instance. Although ACS appears to consistently underperform, it improves over the others in terms of constraint fulfilment for instances D and E with 60 coils, for example (the number of violated constraints is not shown for clarity). Therefore, we conclude that the algorithm

selection should be a part of the tuning phase as it depends on a given instance.

6. PARALELLISM TO ACCELERATE THE CALCULATION

Parallelism is an obvious choice to attain computation speed-up for ACO algorithms, as the solution building and evaluation phases of an ant are both independent and concurrent to that of other ants within an iteration. Many approaches to exploit parallelism at every level, from GPU acceleration of cost function calculation to multi-colony variants have been proposed; see (Janson, Merkle, and Middendorf, 2005) for an in-depth survey.

For our tests, we followed the multithreaded approach where the execution of the ants is performed by a thread pool, splitting the ants in each iteration evenly across threads. Since the algorithm is only partially parallel (the comparison of solutions and the pheromone update occur sequentially between iterations), the expected speed-up must be sublinear in the number of threads; additionally, the synchronization effects of the iterations and the distribution of ants across threads should further reduce the improvements.

We tried out a range of number of concurrent threads on two different platforms (4- and 24-core machine, respectively), and for two different implementations (the built-in thread pool in the .NET standard library, and one of our own), always using as a reference a single-threaded version on the same platform.

As main conclusions, there is a significant speed gain to be obtained from the concurrent execution of the ants, but the marginal gain from each extra thread decreases, and actually becomes negative as we approach the number of available physical or logical threads in the machine (this result might differ in cases where the ants read from files or databases, since then the system can interweave their execution). The implementation of the thread pool can have some impact, but it is minor compared to the overall effect, and difficult to predict; in our case, each implementation had cases where it performed both better and worse than the other, and preliminary analyses point to garbage collection behavior as the main cause.

We expect to improve on these results by removing the synchronization: continuously running ants on every thread and updating the pheromone matrix without stopping them; we expect the erroneous readings to be infrequent enough so as to not distort the solution quality significantly.

7. CONCLUSIONS AND FUTURE WORK

Academia and industry present different approaches when they devote efforts to investigate. In an ideal world, they should converge at a point where industry could apply the research achieved in the laboratory and take advantage of it. Benefits are innumerable, such as: increase productivity with a better job scheduling, gas emissions reduction by optimizing logistic movements or reducing the impact of a bottleneck resource through optimization of its assignment. But the transition from lab to the production plants requires solving some hard restrictions that are not presented in the academic environment: one critical constraint is response time. Under some conditions, the calculation must be delivered in a few minutes (even seconds) whereas the search space of the feasible solutions can have more combinations than there are drops in the oceans.

In this position paper we have introduced the problem of the response time in industrial problems where metaheuristics play a key role. The alternatives presented are mainly focused in two alternatives: parallelism and making the search smarter, with automatic auto-configuration of the algorithms parameters and algorithm selection, using different search strategies. Some promising results were presented that guide the next steps towards further research on each topic.

8. REFERENCES

[1] Cáceres, L. Pérez, M. López-Ibánez, and T. Stützle. 2014. “Ant Colony Optimization on a Budget of 1000.” http://iridia.ulb.ac.be/IridiaTrSeries/rev/IridiaTr2014-009r001.pdf.

[2] Díaz, Diego, Pablo Valledor, Paula Areces, Jorge Rodil, and Montserrat Suárez. 2014. “An ACO Algorithm to Solve an Extended Cutting Stock Problem for Scrap Minimization in a Bar Mill.” In *Swarm Intelligence*, edited by Marco Dorigo, Mauro Birattari, Simon Garnier, Heiko Hamann, Marco Montes de Oca, Christine Solnon, and Thomas Stützle, 13–24. Lecture Notes in Computer Science 8667. Springer International Publishing. http://link.springer.com/chapter/10.1007/978-3-319-09952-1_2.

[3] Dorigo, Marco. 1992. “Optimization, Learning and Natural Algorithms.” *Ph. D. Thesis, Politecnico Di Milano, Italy*. http://ci.nii.ac.jp/naid/10016599043/.

[4] Fernández Alzueta, S., S. Álvarez, E. Malatsetxebarria, P. Valledor, and D. Díaz. 2015. “Performance Comparison of Ant Colony Algorithms for the Scheduling of Steel Production Lines.” In *Genetic and Evolutionary Computation Conference*, 1387–88. ACM. http://dl.acm.org/citation.cfm?id=2764658.

[5] Fernández Alzueta, Silvino, Diego Díaz Fidalgo, Tatiana Manso Nuño, and Montserrat Suarez Rodríguez. 2006. “Optimization Techniques to Improve the Management of a Distribution Fleet in the Steel Industry.”

[6] Fernandez, Silvino, Segundo Alvarez, Diego Díaz, Miguel Iglesias, and Borja Ena. 2014. “Scheduling a Galvanizing Line by Ant Colony Optimization.” In *Swarm Intelligence*, 146–57. Brussels: Springer. http://link.springer.com/chapter/10.1007/978-3-319-09952-1_13.

[7] Janson, Stefan, Daniel Merkle, and Martin Middendorf. 2005. “Parallel Ant Colony Algorithms.” *Parallel Metaheuristics: A New Class of Algorithms* 47: 171.

[8] López-Ibánez, Manuel, Jérémie Dubois-Lacoste, Thomas Stützle, and Mauro Birattari. 2011. “The Irace Package, Iterated Race for Automatic Algorithm Configuration.” *IRIDIA, Université Libre de Bruxelles, Belgium, Tech. Rep. TR/IRIDIA/2011-004*.