Improving the ADACOR^2 Supervisor Holon Scheduling Mechanism with Genetic Algorithms

José Barbosa[*,**], Paulo Leitão[*,‡], Emmanuel Adam[†,§] and Damien Trentesaux[†,**]

[*] *Polytechnic Institute of Bragança, Campus Sta Apolónia, Apartado 1134, 5301-857 Bragança, Portugal*
[†] *Univ. Lille Nord de France, F-59000 Lille, France*
[**] *UVHC, TEMPO research center, F-59313 Valenciennes, France*
[‡] *LIACC - Artificial Intelligence and Computer Science Laboratory, R. Campo Alegre 102, 4169-007 Porto, Portugal*
[§] *UVHC, LAMIH, F-59313 Valenciennes, France*

Abstract. Manufacturing companies are being pushed to their limits due to an increase of production complexity guided by a growing standards demand by the costumers. To respond properly to this, manufacturing companies must adopt innovative control architectures that are able to handle better the occurrence of disturbances at shop-floor level (e.g. workstation breakdown, orders cancellation or modification).

Additionally, the selection of a proper scheduling algorithms assumes a crucial point, in the sense that the increase of optimization levels depend on this.

This paper presents a Genetic Algorithm (GA) based technique to be embedded into the supervisor entity present at the ADACOR^2 aiming to improve the existing fast and non-optimal scheduling technique, improving the overall system processing execution. The main requirements of the GA is to be fast enough to be usable in demanding environments improving the optimization output.

The proposed algorithm is tested using a Flexible Manufacturing System using different configurations of transportation and batch sizes. Results show that despite the presented GA technique increased the optimization calculation time it performs better considering the sum of this time with the gain in the optimization output.

Keywords: Genetic Algorithm, Scheduling, Manufacturing control
PACS: 89.20.Ff

INTRODUCTION

One of the main pillars of the worlds economy is the manufacturing sector that, particularly in the recent years, has suffer a revolution from the client side, being pushed by an ever increasing demand for higher products customization, quality standards and by the decrease of the product life-cycle, just to name a few. On an internal side, and in order to face these constraints, manufacturing has seen an unprecedented process automation, leading to a production increase but also to, in some part, leaving the shop-floor vulnerable to machine failures.

A proper manufacturing control architecture assumes a crucial role in the sense that it must deal with all sort of disturbances that appear at the shop-floor as also to perform the necessary operational functions, such as process planning, scheduling and dispatching [1]. To this point, several paradigms have been proposed, being ones more monolithic and centralized while in a complete different direction others advocate the decentralization and distribution of the decision making throughout the shop-floor.

The first ones due to their intrinsic architecture, concentrate the manufacturing control functions under one single decisional node. Concentrating all the information flow regarding the production under one single node has the advantage that the decisional programs have all the available information to take the best decision when, e.g. a disturbance appears at the shop-floor. Despite this, the algorithms to handle all this information must be tailor made and usually are very complex [2]. Additionally, having this unique processing/decisional node, creates an architecture with a single point of failure, which if it fails, can create a system-wide failure, producing huge mount of losses to companies.

Decentralized manufacturing control architectures follow a completely different paradigm since it distributes the processing/decisional capacity throughout the shop-floor, enabling this way a local reaction, and thus much faster, to disturbances. Although this increase of responsiveness, this approach has the disadvantage that the reaction relies only on local information and so, not reaching the optimization levels found in the centralized architectures.

The ADACOR (Adaptive Holonic Control Architecture for Distributed Manufacturing Systems) architecture was developed [8], promoting a binary functioning state combining optimization with responsiveness. Despite of the good results achieved by the ADACOR, an evolution of it, named ADACOR^2, is being developed promoting the usage of self-organization mechanisms at the micro and macro level of the system, allowing in this way, a better response to the source of disturbances that may appear.

This paper presents a Genetic Algorithm (GA) scheduling based mechanism to be embedded into one of the ADACOR^2 entities, named Supervisor Holon, aiming to improve the first version of the scheduling algorithm, maintaining it fast enough to be usable in a real cases scenarios. The proposed GA is validated in a Flexible Manufacturing System using different batch sizes.

The rest of the paper is organized as follows: section 2 makes a brief overview of the existing heuristic based scheduling algorithms. Section 3 presents the foundations of the ADACOR^2 manufacturing control architecture and the GA based algorithm, while section 4 depicts the use case scenario and extracts the results of the GA based application. Finally the paper is round-up with the conclusions.

HEURISTIC ALGORITHMS AS SCHEDULING TECHNIQUES

The increase of complexity of manufacturing companies, namely in terms of workstation diversity, transportation system or work orders customization is pushing the scheduling algorithms to their limits. Probably the most important issue regarding the selection of an appropriate algorithm for the manufacturing world is not so much obtaining the optimal value but to reach a compromise of having real-time scheduling with the best possible, acceptable, schedule. With this, optimal algorithms, such as Mixed-Integer Linear Programming (MILP) are not suitable due to their calculation time since every possible combination must be tested.

In recent years, heuristic optimization algorithms are taking much attention from the research community. Of particular interest are the bio-inspired algorithms that are being widely used as the optimization mechanisms to solve problems in diverse branches of society, namely in Engineering or Mathematics [3].

Ants individually don't possess the necessary intelligence to develop the complex behaviour [4] displayed when they are building their house of while searching for food. As example, during food foraging, ants leave their nest and start by going in a random walk until they found their destination. Once the food source if found, ants come back to their nest laying down a chemical substance known as pheromone. This scent can then be sensed by other ants that, if the intensity is appellative enough, can decide to follow it. This process, by being repeated numerous times, creates several possible paths to different (or the same) food source. The combination of the number of ants that walk the path with the natural evaporation process that the pheromones suffers (e.g. due to wind) eliminates the less used paths, which tend to be either the longest ones or with poor food. This natural process was translated into the Ant Colony Optimization algorithm [5].

Particle Swarm Optimization (PSO) gets inspiration on the social behaviour of bird flocks and fish schools [6] and is a population-based stochastic optimization technique. The algorithm starts by initializing a population of random solutions. An iterative process is then started, where potential solutions, called particles, fly through the problem space, following the current optimum particles. As the swarm iterates, the fitness of the overall best solution improves (i.e., decreases for minimization problem).

Genetic Algorithms (GA), derived from natural evolution, are based on a population of abstract representations of candidate solutions to an optimization problem. GA use evolutionary operators such as inheritance, mutation, selection and crossover as means to operate the candidates solutions [7].

AN EVOLUTIVE MANUFACTURING CONTROL ARCHITECTURE

The ADACOR manufacturing control architecture defines 4 types of entities (or holons [8]), namely the Product Holon (PH), Task Holon (TH), Operational Holon (OH) and Supervisor Holon (SH).

Every product that the manufacturing company can produce is mapped into a PH that has all the knowledge necessary to produce it. THs are created, as many as work orders being produced at the shop-floor, by the PH, which passes to them the processing plan needed to be executed. The execution of the process plan is then of the responsibility of every TH, which has negotiation capabilities in order to fulfil this. Every resource is mapped with an OH, which is responsible to manage its own internal agenda, by negotiating with the THs. Finally, the SH is a non-physical

entity that is responsible to introduce optimization to the system. As an example, it is responsible to collect all the work-orders being processed and to generate a most optimized schedules to their OHs.

Briefly, in normal conditions, i.e. without disturbances, the system is in a hierarchical structure, where the SH dispatches all the schedules to their OH. On contrary, when a disturbance appears, the system switches into a more heterarchical structure, being now the TH responsible to directly negotiate the processing allocation with the OH. Once the disturbance phase is over, the system switches back to its hierarchical form, assuming the SH once more the role of schedule optimization [8].

Despite the good results achieved by the ADACOR architecture, it's missing the truly possibility to really have an architecture that is able to evolve with the system constraints. In this way, the ADACOR architecture is enriched by acting at two levels. First, by acting at the micro-level, one allows the holons to dynamically change their behaviour. With this, the holons can select the best behaviour to respond better to a given situation, e.g. a TH can select a different allocation mechanism or a OH can change its operational parameters. This is know in ADACOR^2 as behavioural self-organization [9].

Secondly, a drastic change can be applied by re-arranging the relation between the holons. This arrangement appears at a macro-level and is know as structural self-organization ([10]). In practice the result of this re-arrangement can be the clustering of a set of OH under a SH or the grouping of a set of TH to gain negotiation power in the architecture.

In such architecture, the SH assumes a crucial role in the sense that it can introduce schedule optimization, increasing the throughput of the system, and so its profitability. Since the first version of scheduling mechanism was based on a simple but non-optimal algorithm, the challenge was to develop a new one which was able to achieve better optimization results, namely the *Cmax*, without compromising the sum of calculation speed with the output result.

For this purpose, the SH was enriched with a GA based algorithm. The algorithm pseudo-code, shown in Algorithm 1, requires as input the set of work orders to be manufactured and the available workstations.

Algorithm 1 Genetic Algorithm pseudo-code

```
Require: workOrders, workStations
Ensure: Scheduling of work orders to the workstations
1: procedure GA(workOrders, workStations, population)
2:     InitialPopulationGeneration();          ▷ Generates random schedule allocation
3:     n ← population
4:     for i = 0 to n do
5:         addRealTimeToSchedules();           ▷ Adds real time to schedule
6:         orderSolutionsByFitness();
7:         CrossOver();
8:     end for
9:     addRealTimeToSchedules();
10:    orderSolutionsByFitness();
11: end procedure
```

The process starts by generating a set of random scheduling solutions of size *population* each one having already the allocation of the work orders to workstations. Having this set of initial possible solutions, the algorithm will start a iterative process that starts discarding the worst half solutions, followed by a set of crossover operations that will scramble allocated work orders from two random sets of solutions. A random selection of the allocated work orders within these solutions is also used to crossover. After repeating this process *n* times, the best solution will be selected as the one to be dispatched to the shop-floor.

EXPERIMENTAL RESULTS

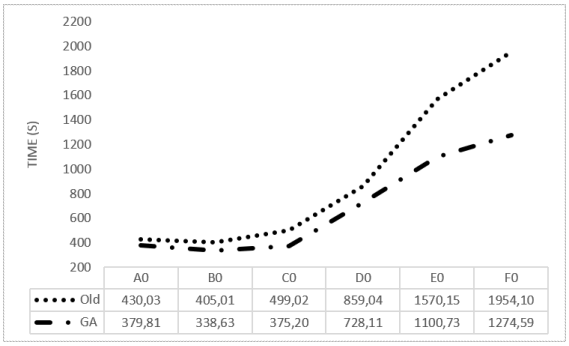
The case study used to test the presented work is based on the AIP-PRIMECA Flexible Manufacturing System (FMS) located at the Université de Valenciennes et du Hainaut-Cambrésis and the products produce herein. The FMS is composed by a shuttle transport conveyor system and 7 workstations, which are categorized into the loading/unloading station, an automated inspection unit, several skill defined operations and a manual recovery. This FMS is able to produce a set of products, namely the letters B, E, L, T, A, I and P, being necessary the execution of a set of operations to its completion [11].

A set of production scenarios and system configurations were designed to benchmark manufacturing control architectures or scheduling algorithms [11] and the designed scenarios involve variations on the batch sizes, shuttles number or constraints on the workstation buffer capacity.

A proper test for the designed GA algorithm, to assess the calculation speed and the output results, imposes the comparison with the previous scheduling algorithm [8] and the use of different batch sizes (in this case the scenarios ranging from A0 to F0 of [11]). Given this, only the *population* parameter is still missing in order to fully characterize

the input data for the algorithm. In the present case, a value of 6 was used, meaning that initially 6 scheduling solutions are generated and that the algorithm runs iteratively 6 times.

The experimental results of running the existing scheduling algorithm (named "old" in the legend) and the GA approach are shown in Figure 1.



	A0	B0	C0	D0	E0	F0
Old	430,03	405,01	499,02	859,04	1570,15	1954,10
GA	379,81	338,63	375,20	728,11	1100,73	1274,59

FIGURE 1. Calculation time plus the output results

It is possible to analyse that for all the testing scenarios, the GA approach obtains better results. As example, for the scenario C0, despithe the existing scheduling algorithm needs 17ms to compute and the GA 11195ms, the GA overall time, considering calculation time with the output result, improves the previous scheduling by 24,81%. Additionally, it is still possible to observe that as the batch order increases, the GA improvement also rises, being of 34,77% for scenario F0.

CONCLUSION

This paper uses a GA to improve the supervisor holon of the ADACOR^2 manufacturing control architecture. This assumes a crucial role in the sense that high level entities are able to introduce schedule optimization at the shop-floor.

Experimental results have shown that even with a simple version of the GA it is still possible to increase deeply the actual scheduling algorithm.

Future work will be devoted to incorporate a dedicated scheduling tool in the SH. Tools such IBM ILOG or the Choco API are good candidates for this integration, being the last one fully compliant with Java. With these it is expected to greatly improve the GA calculation speed and the GA results.

REFERENCES

1. P. Leitão, ADAptive holonic COntrol aRchitecture for distributed manufacturing systems, Ph.D. Thesis, University of Porto (2004).
2. C. Anderson, J. Bartholdi, Centralized versus decentralized control in manufacturing: lessons from social insects, *Complexity and Complex Systems in Industry*, pp. 91-105(2000).
3. P. Leitão, J. Barbosa, and D. Trentesaux, Bio-inspired multi-agent systems for reconfigurable manufacturing systems, *Engineering Applications of Artificial Intelligence*, Vol. 25, pp.934-944 (2012).
4. P. Miller, The Genius of Swarms, *National Geographic* (2007).
5. M. Dorigo, Optimization, Learning and Natural Algorithms (in Italian), Ph.D. Thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy (1992).
6. R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, Proceedings of the Sixth International Symposium on Micromachine and Human Science, pp. 39-43 (1995).
7. Holland, John H, Hidden order: how adaptation builds complexity, Addison-Wesley (1995).
8. P. Leitão, F. Restivo, ADACOR: A Holonic Architecture for Agile and Adaptive Manufacturing Control, *Computers in Industry*, Vol. 57(2), pp.121-130 (2006).
9. J. Barbosa, P. Leitao, E. Adam, D. Trentesaux, Self-Organized Holonic Multi-agent Manufacturing System: The Behavioural Perspective, IEEE International Conference on Systems, Man, and Cybernetics, pp. 3829-3834 (2013).
10. J. Barbosa, P. Leitao, E. Adam, D. Trentesaux, Structural Self-organized Holonic Multi-Agent Manufacturing Systems, Industrial Applications of Holonic and Multi-Agent Systems, pp. 59-70 (2013).
11. D. Trentesaux, C. Pach, Cyrille, A. Abdelghani, Y. Sallez, T. Berger, T. Bonte, P. Leitão, J. Barbosa, Benchmarking flexible job-shop scheduling and control systems, *Control Engineering Practice*,Vol. 21, pp.1204-1225 (2013).