# A Hybrid Optimization Strategy with Low Resource Usage for Large Scale Multi-objective Problems

Wellington Rodrigo Monteiro, Gilberto Reynoso-Meza
Pontifical Catholic University of Paraná
R. Imaculada Conceição, 1155
Curitiba, Paraná, Brazil
wellington.monteiro@pucpr.edu.br
g.reynosomeza@pucpr.edu.br

***ABSTRACT:*** *The use of multi-objective approaches to solve problems in industry grew in the last years. Nevertheless, these strategies are still unused in many fields where their performance is suboptimal or when they are too complex to be implemented or even are simply unknown. One example is in the poultry industry with its particularly complex chain. In this paper, we will discuss a hybrid multi-objective approach with low computational resource usage intended for this scenario as well as other similar ones.*

**Keywords:** Multi-objective, Optimization, Many Variables, Low Resource Usage

**Received:** 12 September 2018, Revised 29 October 2018, Accepted 14 November 2018



## 1. Introduction

In corporate environments, the use of simpler tools in some situations might be favored against other tools that would generate better results. This business decision might be caused due to the lack of technical understanding of these tools or due to cost and performance reasons. One practical example shown in a previous work of the author [7] is the Production Plan algorithm used by one of the largest meat companies in the world, specifically its poultry business line.

By definition, the supply chain in industry is a complex set of operations and resources that must be extremely optimized in order to achieve its maximum potential which does include the management of upstream and downstream relationships in order to achieve an outcome which is more profitable to all the parties in the chain [2]. One of its parts is the Production Plan, which defines what should one or more plants build considering a myriad of variables–i.e. market demand, production line capacity, logistics and stock limits, suppliers constraints, raw material limits, etc. Therefore, an accurate Production Plan is a key component to maximize the potential profits.

In the meat industry the challenges are greater. Since it is livestock, at the same time the supply chain is very large and very tight [3]. The former because its production involves genetics, feeding, breeding, growth control from the current animal up to its grandparents and the latter because there are very strict sanitary (including, but not limited to the health, safety and environment) controls with the ration, water, effluents, temperature and vaccination, for example. Also, the whole animal must be pushed to the market-in the case of the chicken, for instance, even though the market might be more interested to purchase the thighs or other cuts, all the other parts must also be processed and sold somehow.

As a result, the production plan is by itself a problem composed of a large number of variables (at least 2000) [7]. This plan is usually executed as a single-objective problem (the objective being the overall profits) since it is currently able to provide results within the same day. However, it is well known that these profits differ from the real values since the reliability of the plants vary. By reliability impacts we mean both internal (e.g. different production costs between the plants, worker strikes, unscheduled maintenance, stock issues) or external (suppliers, weather) causes that are responsible to reduce the projected profits. Therefore, the company could benefit if the reliability of the production plans was known beforehand-if it was converted to a multiobjective problem (MOP) with the objectives being the expected profits and the reliability of said plans, depending on the case the analyst could choose a production plan that has less expected profits, but higher reliability rates. On the other hand, he or she could also be more aggressive and attempt higher profits, but also with higher chances of not achieving the expected value.

The work presented in [7] proved it was possible to convert the production plan of the company into a multi-objective problem. However, two problems were found: 1) there was an issue with the input data (the reliability rates of each plant), which resulted in very similar grades for all the plants and 2) the multi-objective optimization algorithm took too long (more than 24 hours in an i7 desktop with 16 GB RAM) to generate the production plans. Nevertheless, before its implementation can be greenlit by the company, additional work on both sides is required. As such, while the data issue was corrected by the plants themselves so that it can be usable, the multi-objective optimization algorithm needed to be greatly improved in order to be executed faster and with lower resource usage so that it could be used by an o-the-shelf corporate laptop and able to generate a Pareto front approximation in under one hour.

Considering this background, the proposal is to generate a multi-objective optimization (MOO) algorithm intended for problems with a large number of variables (more than 2000 since the original problem is expected to grow in complexity). As such, the main objective here is to have an algorithm that balances both the performance (i.e. low computational resource usage) and the scalability (i.e. capable of processing problems with thousands or tens of thousands of variables). On that end, a test set with similar characteristics of the real-world problem will be used to evaluate the algorithm.

The remainder of this document is structured as follows. The second section explains the background of the test problems that will be used as well as their rationale. The third section shows the proposal to modify and test the presented case into a MOP. Then, the following section specifically shows the technical details of the created MOP as well as its results after optimization. The fifth section presents the conclusions and the future work to be done from this document.

## 2. Background

Currently, in the meat industry some optimization solutions are both single-objective and using specialized algorithms built from scratch with the profitability in mind such as OtimixTM. However, said algorithms might not enable the industries to use two or more objectives or provide greater parameter tuning possibilities. Since such algorithms are targeted towards only one objective, the problem designer usually has only one solution as the result of the minimization/ maximization, eliminating the possibility to analyze the tradeoffs between different production plans considering two or more objectives. As a result, the production planners are required to empirically consider the differences between the plants from a reliability standpoint, leaving no possibility to compare the solutions based on this factor.

The scenario that originated this algorithm had two objectives: Profitability and reliability [7]. 2032 variables were employed from which all were integers-however, this scenario was known to be a test-therefore, more variables were expected. As such, considering these characteristics and the other business requirements, the new algorithm had to meet the following objectives:

- Be a multi-objective optimization algorithm;
- Be able to resolve problems with many variables (more than 1000, ideally with more than 15000), all integers;

• Low computer resource usage (preferably less than 1 GB RAM per 5000 variables);

• Be able to generate a Pareto front approximation (even if there is still room for improvements) in less than one hour.

Since the original data needed additional work from the teams responsible for it, the alternative was to choose easily configurable and reliable mathematical problems. The choice was the test problems in [1]-nine large-scale, multiobjective problems were considered to evaluate the proposed algorithm, each configured with 1000, 5000, 15000, 30000 and 50000 variables. All the variables are integers similar to the production plan problem–all of the problems were set to have 2 objectives, analogous to the production plan problem. These test problems are henceforth named LSMOP*n*, where n is the problem number ranging from 1 to 9 and forming a different problem. All the tests use a combination of six basic single-objective functions. These functions are the sphere function, the Schwefel's function, the Rosenbrock's function, the Rastrigin's function, the Griewank's function and the Ackley's function and, as shown in [1], implements features such as a chaos-based pseudo random number generator (to address the nonuniform grouping), a correlation matrix to keep track of the correlations between variable groups and the objective groups, different basic fitness landscape functions (to achieve the mixed separability) and also making use of a linkage function to define the variable linkages.

**3. Proposal**

As mentioned in the Section 2, this article proposes to implement an algorithm capable of resolving multi-objective problems with many variables with low resource usage. For this reason, the approach chosen was a hybrid multi-objective genetic algorithm. By hybrid, as shown in [4], it is considered to be an algorithm (in this case a genetic algorithm) enhanced with an additional local search step shortly after the selection of the individuals. The algorithm, shown in the Algorithm 1 and implemented in MATLAB c , heavily focuses on the parallelization for performance purposes. On the other hand, it attempts to overuse large matrices in order to reduce the memory footprint.

**Algorithm 1:** The proposed algorithm

**Data:** design space, objective vector

**Result:** the Pareto front approximation generate *n* random solutions;

rank the solutions by a dominance filter;

**for** *each generation until it reaches the stopping criteria* **do** store previous solutions and their objective values;

generate offspring by tournament, recombination and mutation;

join the offspring to the other solutions;

partially rank all solutions with a dominance filter;

locally improve the best solutions;

replace the best solutions with locally improved solutions;

rank all the joined solutions with a dominance filter;

prune some solutions by their crowding distances.

**end**

The local improvement algorithm (shown in Algorithm 2) has a new, optional parameter named number of random neighbors for the local search (NumberRandomNeighbors). It is used to improve the performance in scenarios where there are too many variables and this algorithm would take too long to process all of the variables.

For example, if a solution has 1000 variables and this parameter is not used, this algorithm will create 1000 new solutions based on the original solution where the first solution assigned a new random value for the first variable while keeping all the other variables intact; the second solution assigned a new random value for the second variable while keeping all the other variables

intact and so on. If 100 solutions are involved in the local search, at least 100 thousand new solutions would be created as a result. On the other hand, if a solution had 15000 variables, considering the same 100 solutions as a result 1.5 million new solutions would be created. Considering the local search would happen more than once during the algorithm execution, this method–based on the exhaustive neighborhood exploration [5] would take too long. This results in greater performance gains between the different generations.

**Algorithm 2:** Local improvement

**Data:** a solution

**Result:** Neighbors initialize the list of neighbors Neighbors;

**if** *NumberRandomNeighbors* was given **then**

    select NumberRandomNeighbors random variables;

**else**

    select all the variables;

**end**

**foreach** *one of the variables selected* **do**

    copy the original solution;

    randomly modify its value according to its bounds;

    evaluate the new solution;

    add it to *Neighbors*;

**end**

replace *Neighbors* with only its anchors.

## 4. TEST

The new, proposed algorithm and the other algorithms were tested on an i7 desktop equipped with 16 GB RAM and a dedicated video card running Windows 10 Pro running MATLAB© R2015a. This desktop ran all the tests over the course of three weeks with one weekly system restart. In MATLAB©, the start and end time of each algorithm were tracked (therefore storing the time taken for each execution) as well as the Pareto front approximation found for each execution, each limited to 300 seconds. Outside MATLAB©, the Windows Performance Monitor (perfmon.exe, a known, built-in performance monitor tool in Windows) was used to track performance and memory usage in MATLAB©. As such, it was able to properly track resource usage by each algorithm. The new algorithm was tested against MATLAB©'s own gamultiobj, a built-in, optimized algorithm based on NSGA-II and a Multi-objective Differential Evolution with Spherical Pruning algorithm (sp-MODE II) [8].

### 4.1 Evaluation Methods

With the aforementioned tools both the hypervolume found for each algorithm considering the best Pareto front approximations found for them and the memory and processor usage for each algorithm were measured.

For each of the nine LSMOP problems and number of variables (1000, 5000, 15000, 30000 and 50000 variables), 51 runs were executed for each one of the three algorithm. NumberRandomNeighbors was set to 50 in the new algorithm and the initial number of random solutions set to 20 for all algorithms, limited to 200 * number of generations and a maximum of 5 generations without improvement (where an improvement is determined when the utopia had improved in at least 0.0001% for at least one objective) per run.

### 4.2 Findings

The values shown in Table 1 represent the median values for each one of the runs for each algorithm and for each problem based

| Problem | NVar | gamultiobj | sp-Mode II | new algorithm |
|---|---|---|---|---|
| LSMOP1 | 1000 | 0.998933 | 0.916951 | 0.966355 |
| LSMOP2 | 1000 | 0.531942 | 0.472204 | 0.497616 |
| LSMOP3 | 1000 | 0.999881 | 0.916340 | 0.960246 |
| LSMOP4 | 1000 | 0.562390 | 0.494451 | 0.521077 |
| LSMOP5 | 1000 | 0.997482 | 0.427428 | 0.628847 |
| LSMOP6 | 1000 | 0.999599 | 0.626870 | 0.916751 |
| LSMOP7 | 1000 | 0.999644 | 0.615992 | 0.947478 |
| LSMOP8 | 1000 | 0.995152 | 0.508990 | 0.686416 |
| LSMOP9 | 1000 | 0.991079 | 0.483426 | 0.695966 |
| LSMOP1 | 5000 | 0.990506 | 0.931265 | 0.958904 |
| LSMOP2 | 5000 | 0.512829 | 0.470381 | 0.488354 |
| LSMOP3 | 5000 | 0.996004 | 0.925004 | 0.954693 |
| LSMOP4 | 5000 | 0.545665 | 0.496839 | 0.516004 |
| LSMOP5 | 5000 | 0.995279 | 0.561639 | 0.609145 |
| LSMOP6 | 5000 | 0.947114 | 0.575706 | 0.677455 |
| LSMOP7 | 5000 | 0.999411 | 0.752861 | 0.882027 |
| LSMOP8 | 5000 | 0.993857 | 0.599694 | 0.609580 |
| LSMOP9 | 5000 | 0.958439 | 0.507450 | 0.593416 |
| LSMOP1 | 15000 | 0.979783 | - | 0.945874 |
| LSMOP2 | 15000 | 0.513371 | - | 0.484434 |
| LSMOP3 | 15000 | 0.998651 | - | 0.941039 |
| LSMOP4 | 15000 | 0.535847 | - | 0.510049 |
| LSMOP5 | 15000 | 0.974157 | - | 0.688771 |
| LSMOP6 | 15000 | 0.771705 | - | 0.585666 |
| LSMOP7 | 15000 | 0.999059 | - | 0.845907 |
| LSMOP8 | 15000 | 0.989229 | - | 0.680188 |
| LSMOP9 | 15000 | 0.990760 | - | 0.540451 |
| LSMOP1 | 30000 | - | - | 0.840306 |
| LSMOP2 | 30000 | - | - | 0.466373 |
| LSMOP3 | 30000 | - | - | 0.885842 |
| LSMOP4 | 30000 | - | - | 0.486888 |
| LSMOP5 | 30000 | - | - | 0.192226 |
| LSMOP6 | 30000 | - | - | 0.604263 |
| LSMOP7 | 30000 | - | - | 0.223234 |
| LSMOP8 | 30000 | - | - | 0.184088 |
| LSMOP9 | 30000 | - | - | 0.668866 |
| LSMOP1 | 50000 | - | - | 0.831989 |
| LSMOP2 | 50000 | - | - | 0.476713 |
| LSMOP3 | 50000 | - | - | 0.887729 |
| LSMOP4 | 50000 | - | - | 0.485269 |
| LSMOP5 | 50000 | - | - | 0.220266 |
| LSMOP6 | 50000 | - | - | 0.573581 |
| LSMOP7 | 50000 | - | - | 0.081000 |
| LSMOP8 | 50000 | - | - | 0.208853 |
| LSMOP9 | 50000 | - | - | 0.669033 |

Table 1. Median hypervolume values found for each algorithm and problem. The values are relative to the utopia and nadir determined from all solutions in the Pareto front approximations found for all runs and all algorithms. The best values are in bold

on the implementation in [6]. From the hypervolume in itself the new algorithm proved incrementally better performance when the problem has more variables-in fact, starting with 15000 variables the other algorithms are unable to run these problems due to out of memory errors (as shown by their lack of median values depending on the case).

On the performance side, the memory allocation behavior was similar for the problems with the same size for all the three algorithms. The Figure 1 represents the memory usage for all the runs with 5000 variables for a given problem. First, from 11:34:49 PM to around 03:48:00 AM all the 51 runs for the new algorithm took place, followed by the 51 runs of the gamultiobj from that time to around 7:00:00 AM and later by the 51 runs of sp-MODE II. The new algorithm, as specially shown in Figure 1, used a little more than 1 GB in memory in order to achieve the results. The processor usage registered the same pattern with around 15% in overall usage for the new algorithm.

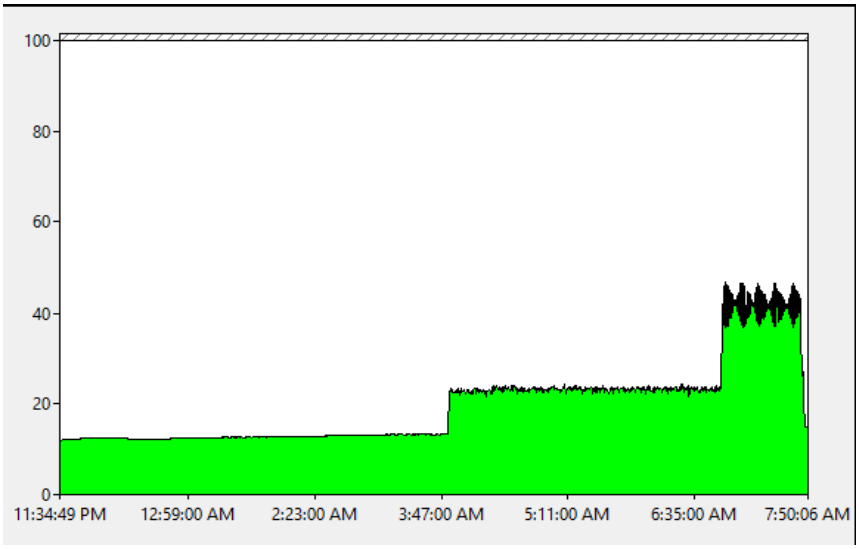


Figure 1. Results for the memory usage for three algorithms with 5000 variables. The y-axis represent the memory usage in hundreds of megabytes

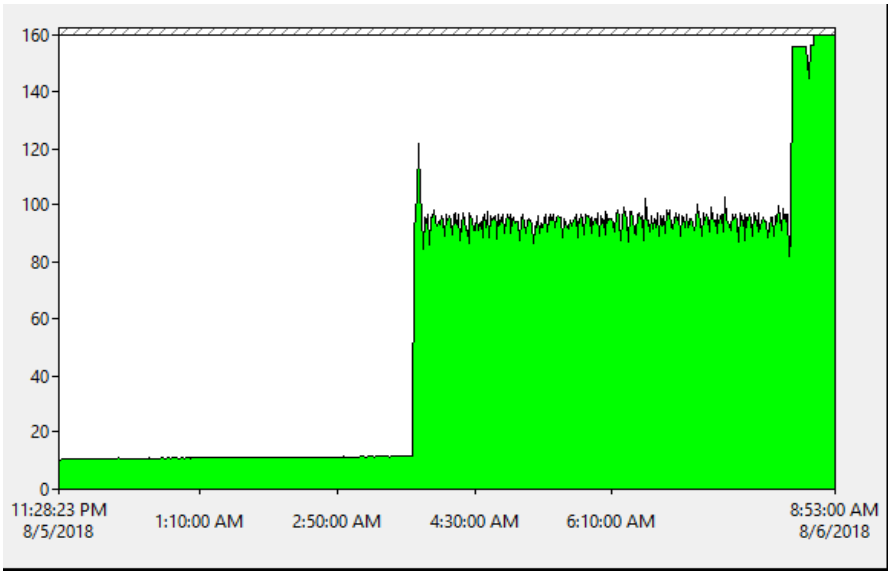


Figure 2. Results for the memory usage for three algorithms with 15000 variables. The y-axis represent the memory usage in hundreds of megabytes

Furthermore, the memory usage grew considerably when changing the problem size to 15000 variables, as seen in an attempt recorded in the Figure 2. While the new algorithm, with its runs recorded between 11:28:23 PM to around 3:30:00 AM kept the memory usage around 1 GB in memory, gamultiobj (executed from that time up to around 8:00:00 AM) registered around 10 GB in usage. sp-MODE II, on the other hand, attempted to allocate more than 16 GB (as registered by the last spike to the right), culminating in an out-of-memory error. This behavior happened again with 30000 and 50000 variables, but with gamultiobj as well. The processor usage also showed the same behavior from an usage standpoint. With 15000 variables the new algorithm used around 25% from the processor, peaking in around 35% with 50000 variables, depending on the problem.

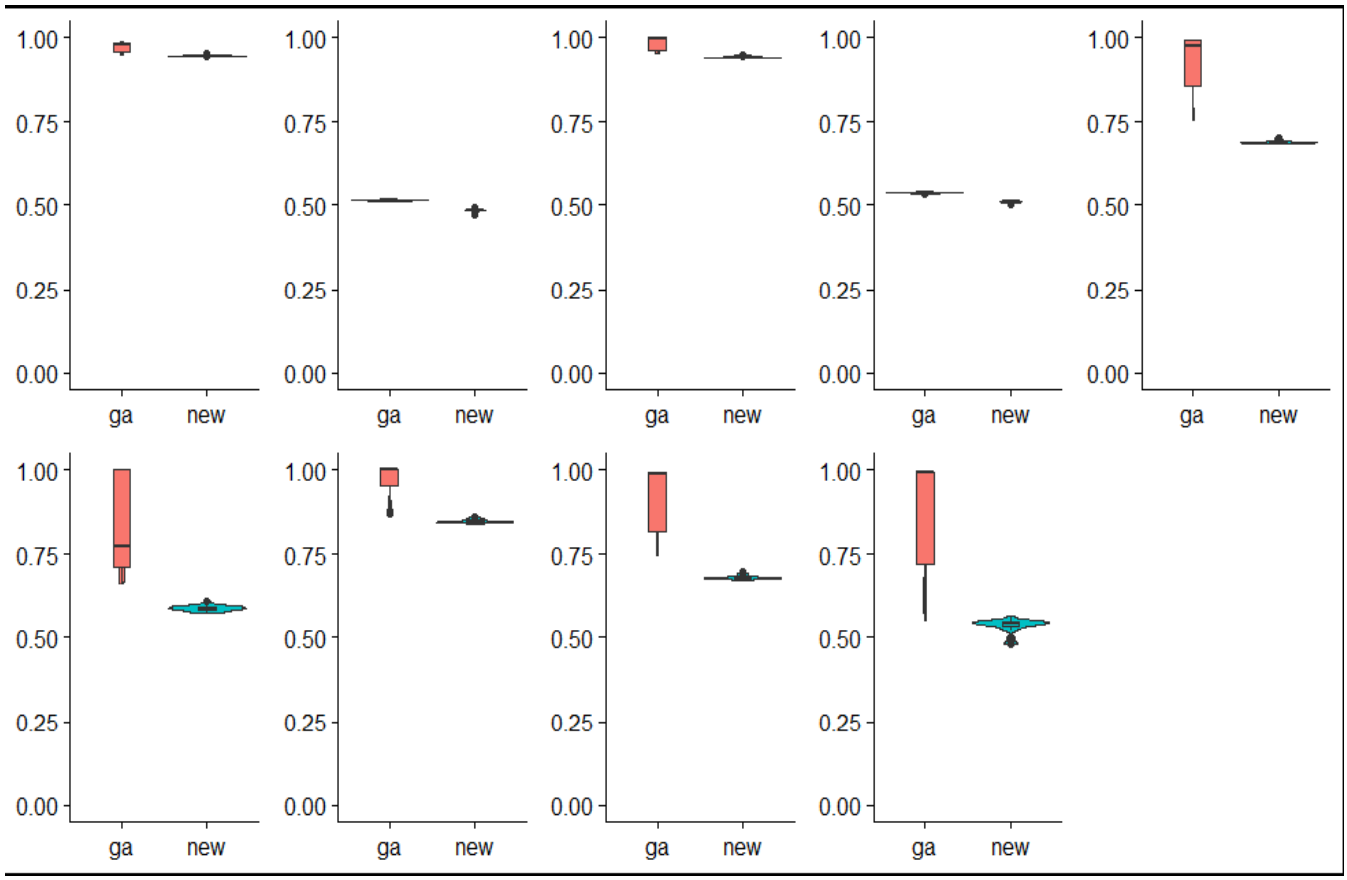


Figure 3. Hypervolume distribution for all the problems for 15000 variables. The y-axis represents the hypervolume values, where *ga* represents the runs with gamultiobj and *new* with the new algorithm

## 5. Conclusions

The proposed algorithm showed that it is better suited for problems with a large number of variables. As of the time of writing this paper the company still could not make a new, real-world case available for tests. On the other hand, the new algorithm can also be implemented in other complex scenarios where, for example, the objectives are both the maximum profits for a given distributed production plan and the energy and steam consumption reduction in the plants. The energy and steam consumption variables would result in an estimated multi-objective problem with around 40000 variables since each industrial boiler, production line and work shift could also be accounted.


## Acknowledgments

The authors would like to thank the Pontical Catholic University of Paranà for supporting the research. This study was nanced in part by the Coordenacao de Aperfeicoamento de Pessoal de Nivel Superior - Brasil (CAPES) - Finance Code 001 and the PQ2|304066/2016-8 grant.

**References**

[1] Cheng, R., Jin, Y., Olhofer, M., Sendho, B. (2016). Test problems for large-scale multiobjective and many-objective optimization. *IEEE Transactions on Cybernetics*.

[2] Christopher, M. (2016). Logistics & supply chain management. Pearson UK.

[3] Flanders, F., Gillespie, J. R. (2015). Modern livestock & Poultry production. Cengage Learning.

[4] Kelner, V., Capitanescu, F., Leonard, O., Wehenkel, L. (2008). A hybrid optimization technique coupling an evolutionary and a local search algorithm. *Journal of Computational and Applied Mathematics*, 215 (2) 448-456.

[5] Liefooghe, A., Humeau, J., Mesmoudi, S., Jourdan, L., Talbi, E.-G. (2012). On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems. *Journal of Heuristics*, 18 (2) 317-352.

[6] MATHWORKS. Hypervolume approximation - matlab central, 2015.

[7] Monteiro, W. R., Reynoso-Meza, G. (2017). A multi-criteria based approach for the production distribution in the poultry industry. *In*: 24th ABCM International Congress of Mechanical Engineering. *Brazilian Society of Mechanical Sciences and Engineering*.

[8] Reynoso-Meza, G., Sanchis, J., Blasco, X., Martnez, M. (2010). Design of continuous controllers using a multiobjective differential evolution algorithm with spherical pruning. *In:* European Conference on the Applications of Evolutionary Computation, p. 532-541. Springer.