Support information file for manuscript "Multi-objective matheuristic for minimization of total tardiness and energy costs in a steel industry heat treatment line"

1. Introduction

This document is divided into three sections. The first one explains how to use the shared dataset and its respective results. The second section presents preliminary tests that supported the definition of the algorithm parameters. The third section shows a comparison between the proposed matheuristic and NSGA-II.

2. Computational Experiments

We shared 24 test instances that can be used for validation and future works.

For each instance, there is a JSON file that contains the optimization parameters shown in Table 1. The JSON variable name is in the parenthesis after the description. Each file also contains the parameters grouped by temperature (GroupByTempParameters) and due date (GroupByDateParameters). They are used in the MILP runs to generate the initial solutions.

The computational experiments presented in the manuscript were all performed with the 24 instances: 30 executions of the proposed matheuristic and 30 executions of the MOGVNS algorithm initialized with a simple heuristic. We used an 8GB RAM Ubuntu 18.04 LTS server to run the tests.

The raw results from these tests are also shared as CSV data with the filenames as 'instanceX-30exec-15s.csv'. Each row of the file contains the results from one execution of the algorithm. The columns listed are:

- Instance: instance identification;
- RealSolution: the cost functions of the sequence executed in reality. The first value is the total tardiness (TT) followed by a space and the total energy cost (TEC);
- Execution: execution number (from 0 to 29);
- Initial Method: the method used to generate the initial solutions for the MOGVNS algorithm. When the value is "MILP", we are referring to the proposed matheuristic executions and when it is "Heuristics", the only MOGVNS run for comparison;

⋆

Table 1: Notation for heat treatment scheduling model parameters.

i, r	index for job.
k	index for machine.
m	number of machines (NumberMachines).
n	number of jobs (NumberJobs).
p_i	total process time of job i throughout all machines $[h]$ (ProcessTimeTotal).
p'_i	process time to enter all pieces of job i in first machine $[h]$ (ProcessTimeByJob).
p'_{ik}	process time of a piece of job i in machine k $[h]$ (ProcessTimeByJobAndMachine).
s_{irk}	setup time of job r in machine k, when job i precedes job r $[h]$ (SetupTime).
w_{ir}	waiting time to start job r, when job i precedes job r $[h]$ (WaitingTime).
v_i	total volume of gas used during execution of job i throughout all machines $[Nm^3/h]$ (VolumeGasByJob).
v_{ir}	total volume of gas used to setup all machines for job r, when job i precedes job r $[Nm^3/h]$ (VolumeGasIdleTime).
g	cost of natural gas $[\$/Nm^3]$ (CostNaturalGas).

- NumGroupsTEC (only for MILP initial method): number of jobs considered in the minimization of TEC after applying the grouping strategy;
- TimeTEC (only for MILP initial method): runtime of MILP to minimize TEC in seconds;
- GapTEC (only for MILP initial method): the gap of MILP to minimize TEC (if the solver reached the 1h maximum runtime);
- NumGroupsTT (only for MILP initial method): number of jobs considered in the minimization of TT after applying the grouping strategy;
- TimeTT (only for MILP initial method): runtime of MILP to minimize TT in seconds;
- GapTT (only for MILP initial method): the gap of MILP to minimize TT (if the solver reached the 1h maximum runtime);
- TimeTT_TEC (only for MILP initial method): runtime of MILP to minimize TEC after TT in seconds;
- GapTT_TEC (only for MILP initial method): the gap of MILP to minimize TEC after TT (if the solver reached the 1h maximum runtime);
- InitialCostFunctions: cost functions of the initial solutions found by the InitialMethod. The first value is TT followed by a space and TEC;

- Algorithm: this column only identifies the metaheuristic applied, which will always be MOGVNS-FI;
- RunTimeSec: the runtime of the MOGVNS_FI run in seconds;
- NumParetoSolutions: the number of solutions in the final approximated Pareto front (PF);
- CostFunctions: the cost functions of each solution of the final approximated Pareto (used to draw the charts). A pipe "|" separates the values from different solutions, and first, there are the total tardiness and then, separated by a space, the total energy costs. Example for a 3 solutions PF: 'sol1(TT) sol1(TEC) — sol2(TT) sol2(TEC) — sol3(TT) sol3(TEC)';
- Solutions: the final sequences given by each solution of the final approximated Pareto front. A pipe "|" separates them, and they are ordered according to the CostFunctions column.

There is an R script ("results_analysis.R" - R version 3.6.3) attached to the results data that can be executed to generate the tables and statistical analysis from the CSV files.

3. Algorithm Parametrization

Two preliminary tests were done to define the MOGVNS algorithm parameters:

1. 30 runs of the multi-objective variable neighborhood descent search algorithm (MOVND-FI) shown in the manuscript to define the best configuration of neighborhoods and see the average execution time.
2. 30 runs of a multi-objective random variable neighborhood search algorithm (Duarte et al., 2015) to define the shaking procedure.

For item 1, three neighborhood configurations were tested:

- 0: SWAP + EXCHANGE + INSERT
- 1: INSERT + EXCHANGE
- 2: EXCHANGE + INSERT

The MOVND-FI algorithm was executed to compare the neighborhood configurations, and the hypervolume of the final approximated Pareto was calculated. Figure 1 shows the results for the three configurations where it can be seen that the average value of configuration 2 is higher than the others.

From the boxplot, it is also possible to see that the MOVND-FI is robust to different neighborhood configurations since the differences in hypervolume are not so high. Figure 2 shows the results of a Tukey's test, based on an analysis of variance (ANOVA) with a 95% level of confidence, that confirms no statistical differences from the neighborhoods configuration. Even though it is not possible to affirm which one is the best, we chose the third one as the final algorithm configuration for its higher average values.

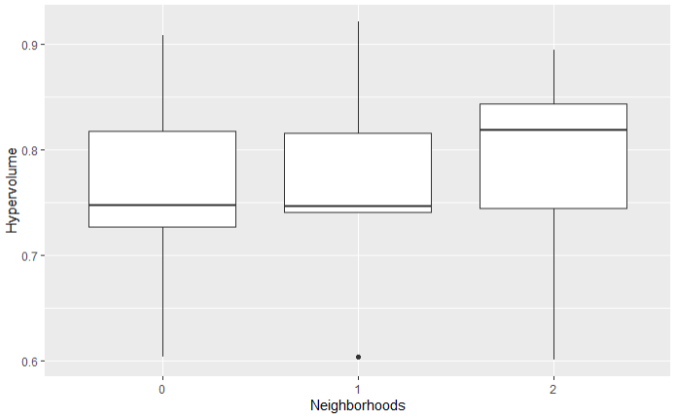

Figure 1: Comparison of the hypervolume mean values of 30 executions of the MOVND-FI with the different neighborhood configurations.

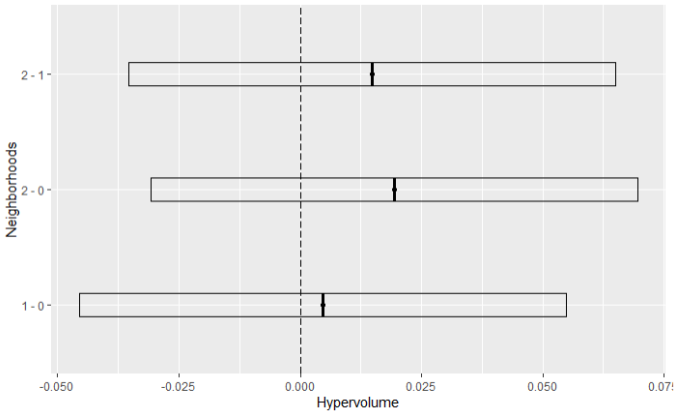

Figure 2: Tukey's test with a statistical comparison of the three neighborhood configurations.

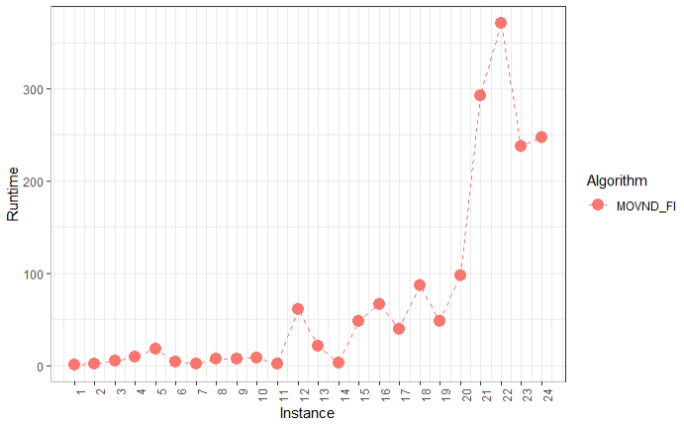

Figure 3: Average runtime in seconds for the MOVND algorithm per instance.

From the MOVND executions, it was possible to check the average runtime for each descent run. These values, presented in Figure 3, were used for the definition of the maximum execution time of the MOGVNS used in the manuscript.

For item 2, four different shaking procedures were compared:

- MORVNS-I: exchange operations with randomly selected jobs;
- MORVNS-II: insert operations with randomly selected jobs;
- MORVNS-III: an exchange and then an insert operation with randomly selected jobs;
- MORVNS-IV: randomly chosen exchange or insert operations with randomly selected jobs.

The MORVNS stoppage criteria were the maximum execution time (t_{max}). The performance metric used to compare the configurations was the hypervolume, and Figure 4 shows that the MORVNS-IV outcomes the others in terms of its average values.

In a statistical analysis, using ANOVA and Tukey's test with a 95% level of confidence, MORVNS-IV is also better than the others, as shown in Figure 5. The MORVNS algorithm is more sensitive to these shaking configurations because it only relies on these random movements. Despite that, it allowed us to verify the efficiency of the MORVNS-IV shaking procedure, that was chosen for the final MOGVNS algorithm.

4. Comparison with NSGA-II

The manuscript focuses on the solution of a real industrial scheduling case. Even though the problem formulation was based on classical no-wait flow shop problems, our mathematical model and approaches are problem-specific. Since the formulation is unique, it is not possible to make direct comparisons with other researches. Therefore,

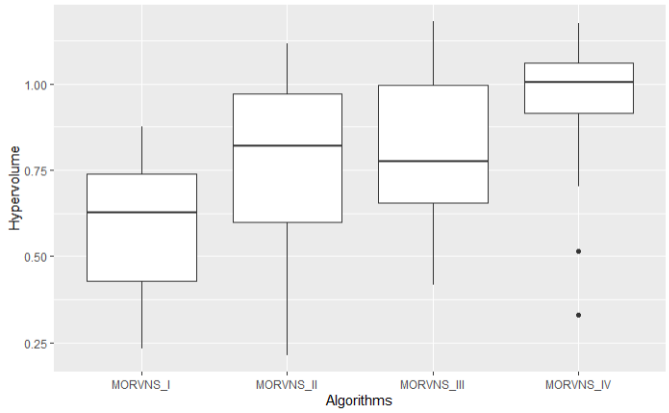

Figure 4: Comparison of the hypervolume mean values of 30 executions of the MORVNS with four different shaking procedures.

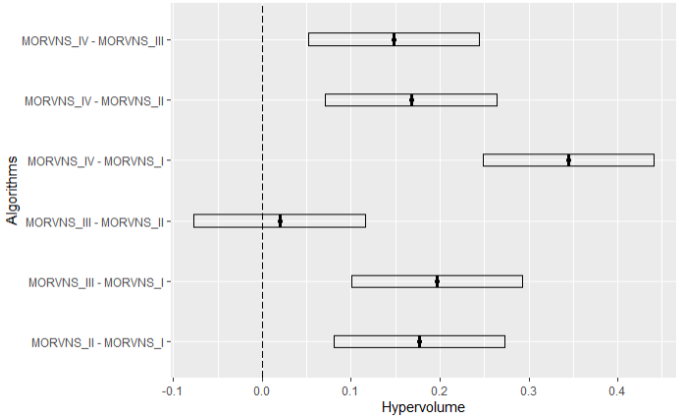

Figure 5: Tukey's test with a statistical comparison of the four shaking procedures.

to validate the performance of the proposed matheuristic, we compare it with the well-known NSGA-II (Deb et al., 2002). The latest is considered a state-of-the-art multi-objective evolutionary algorithm widely used to derive approximate Pareto front of multi-objective optimization problems.

The parameters of NSGA-II were set as follows: population size of 50, order-based crossover operation with a rate of 0.5, mutation operations based on the MOGVNS shaking neighborhood (size 3) and rate of 0.8, and maximum execution time of $15 * n$, with n being the instance number of jobs. Similar comparisons can be found in Chen et al. (2019); Wu & Che (2019).

The order-based crossover (OBX) is commonly used in scheduling problems (Kellegöz et al., 2008) and can be considered as a set of insert and exchange operations, being comparable to the MOGVNS descent search neighborhoods.

The initial population of the NSGA-II was randomly generated and two variations were tested. One with all 50 solutions being randomly selected, and another with 48 random solutions and 2 from the MILP execution (same used in the matheuristic). We will call the first algorithm only NSGA-II and the second the NSGA-II-MILP.

The proposed matheuristic and the two variants of the NSGA-II algorithm were executed 30 times for each of the 24 test instances. The normalized hypervolume of the final approximated Pareto front was calculated for each execution. Figure 6 shows the instances mean hypervolume per algorithm. The red series represent the matheuristic (MOGVNS-MILP), the blue the NSGA-II, and the green the NSGA-II with the MILP solutions in the initial population (NSGA-II-MILP).

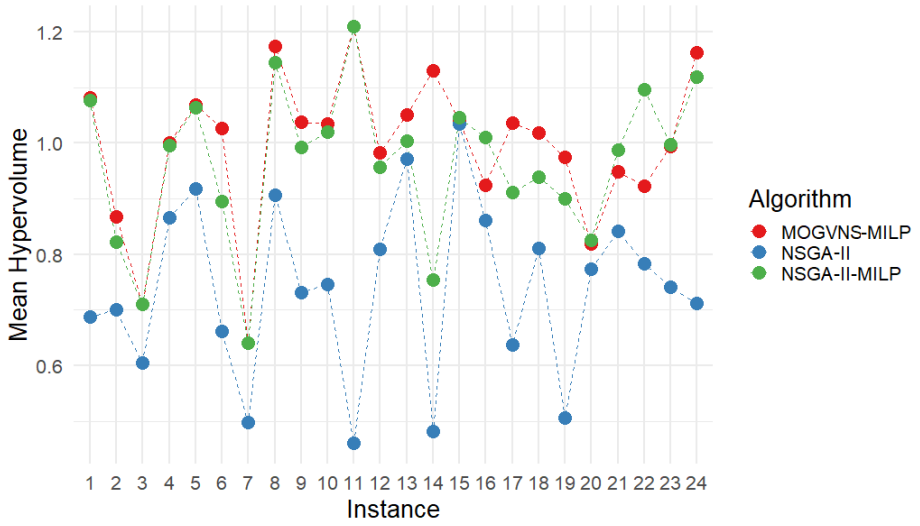

Figure 6: Comparison of the mean hypervolume calculated over 30 executions of the multi-objective matheuristic (MOGVNS-FI-MILP), NSGA-II, and NSGA-II with the MILP solutions in the first population set (NSGAII-MILP).

From this exploratory analysis, it is possible to see that the mean hypervolume of NSGA-II is smaller than the proposed matheuristic for all the instances. When we introduce the two solutions generated by mathematical programming in the NSGA-II

initial population, we observe an improvement in the mean hypervolumes reached. Even though, the method proposed in the manuscript is superior than the NSGA-II-MILP, in terms of hypervolume, for 15 instances.

Figure 7 shows a statistical comparison of the three algorithms based on Tukey's test with a 95% level of confidence. The analysis shows that the proposed matheuristic is statistically superior to the NSGA-II in terms of hypervolume. It can also be observed that the MOGVNS-MILP and the NSGA-II-MILP are statistically incomparable.

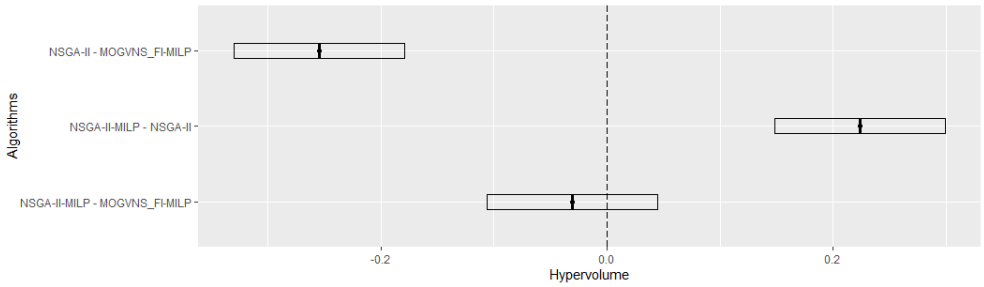

Figure 7: Tukey's test with a statistical comparison between the proposed matheuristic, NSGA-II, and NSGA-II with the two MILP initial solutions in the first generation set.

The conclusion from this experiment highlights the importance of the mathematical programming initial solutions in the convergence and diversity of the final approximated Pareto fronts found by a subsequent metaheuristic. It also shows that the MOGVNS-FI performance is comparable to the NSGA-II, considered a state-of-the-art framework to solve multi-objective optimization problems. Future works intend to explore problem-specific neighborhoods to improve MOGVNS performance.

References

Chen, J.-f., Wang, L., & Peng, Z.-p. (2019). A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling. *Swarm and Evolutionary Computation*, *50*, 100557. http://www.sciencedirect.com/science/article/pii/S2210650219302652. https://doi.org/10.1016/j.swevo.2019.100557.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*, 182–197. https://doi.org/10.1109/4235.996017.

Duarte, A., Pantrigo, J. J., Pardo, E. G., & Mladenovic, N. (2015). Multi-objective variable neighborhood search: an application to combinatorial optimization problems. *Journal of Global Optimization*, *63*, 515–536. https://doi.org/10.1007/s10898-014-0213-z.

Kellegöz, T., Toklu, B., & Wilson, J. (2008). Comparing efficiencies of genetic crossover operators for one machine total weighted tardiness problem. *Applied Mathematics and Computation*, *199*, 590–598. http://www.sciencedirect.com/science/article/pii/S009630030701034X. https://doi.org/10.1016/j.amc.2007.10.013.

Wu, X., & Che, A. (2019). Energy-efficient no-wait permutation flow shop scheduling by adaptive multi-objective variable neighborhood search. *Omega*, (pp. 102–117). http://www.sciencedirect.com/science/article/pii/S030504831930194X. https://doi.org/10.1016/j.omega.2019.102117.