The Job-shop Scheduling Problem Considering Deterioration Effects

Diana G. Campinhoa,b, Dalila B.M.M Fontesc,d, Alexandre F.P. Ferreiraa,b

aLSRE-LCM – Laboratory of Separation and Reaction Engineering - Laboratory of Catalysis and Materials, Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal

bALiCE – Associate Laboratory in Chemical Engineering, Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal

cFaculty of Economics, University of Porto, Rua Dr. Roberto Frias, 4200-464, Porto, Portugal

dINESC TEC, Rua Dr. Roberto Frias, 4200-465, Porto, Portugal

aferreir@fe.up.pt

Abstract

This article addresses the significant issue of machine deterioration effects in job-shop scheduling problems and aims to create awareness about this topic within the industry. While previous studies have explored deteriorating effects in various configurations, research on scheduling problems in complex settings, particularly job-shop, is limited. Thus, this work focuses on the impact of deterioration in a generic job-shop scheduling problem. The study employs a biased random key genetic algorithm as a metaheuristic framework for optimization. The effectiveness of the algorithm, and its multi-population variant is demonstrated through computational experiments. Furthermore, the study investigates different deterioration functions, including linear, exponential, and sigmoid dependencies. The results highlight the importance of considering deterioration in machine scheduling to improve production efficiency, reduce costs, and mitigate potential disruptions.

**Keywords**: scheduling, job-shop, BRKGA, deterioration effects, chemical industry.

* 1. Introduction

One of the main challenges many industries face is the uncertainty of various parameters, including processing times. In addition, the presence of raw materials in a queue waiting to be processed leads to their deterioration over time. In some cases, this deterioration may result in the complete loss of the raw materials, while in other cases, they may still be salvageable but require additional processing time. This phenomenon, introduced by Gupta *et al.* (1988) and Browne *et al.* (1990), can be observed in many settings, such as steel rolling mills or fire-fighting response scheduling.

In the chemical industry, production systems often undergo changes in the physical conditions of their machines as a result of overheating, fouling, impurities, or other phenomena, leading to degradation. These changes cause a reduction in machine efficiency and an increase in processing times. Also, machine fouling results in lower production rates; hence, it takes longer to produce the necessary quantities and/or reduces heat transfer. In conversations with several companies, it has become apparent that they are unaware of the effects of deterioration, as they do not actively seek to identify the causes of failures during the production process. This lack of awareness results in production disruptions, lower productivity, and increased waste that must be discarded.

Analogous to the job-shop scheduling problem (JSP), a multi-product batch process in the chemical industry requires the production of different products (jobs), each with its own recipe and processing requirements, on a shared set of processing equipment (machines). Thus, adopting the job-shop perspective, the multi-product batch process can be effectively modeled as a scheduling problem.

Extensive research has been conducted on various configurations, mainly within the single-machine problem. Nonetheless, researchers have been exploring more complex configurations, which are more common and realistic in the manufacturing industry. In a recent review on deteriorating effects, Pei *et al.* (2022) report on works addressing the single and parallel machine and flow shop scheduling problems, but none on the JSP.

The main goal of this work is to study the effect of deterioration in a JSP, which can represent a chemical process system, and create awareness for the topic in the industry.

* 1. Methodology
     1. Biased Random Key Genetic Algorithm

The Biased Random Key Genetic Algorithm (BRKGA) is based on a general-purpose metaheuristic framework (Gonçalves et al., 2018). In this framework, the only connection to the combinatorial optimization problem is the problem-dependent portion of the algorithm. The problem-independent portion of BRKGA involves generating a set of random keys, sorting them based on their fitness, and creating new keys. The initial population is made up of vectors of random keys, with each allele being independently generated at random within the real interval . To evolve the population, a new generation of individuals must be produced. The new generation is composed of a combination of the current generation's elite individuals (group of individuals with best fitness), the offspring generated by reproduction (crossover), and the mutants. A mutant is a vector of random keys generated in the same way that the initial population was. The mechanism used to implement mating is parametrized uniform crossover. The allele of the offspring takes on the value of the elite parent with probability . This work employs a multi-population strategy, where multiple populations evolve in parallel and exchange high-quality chromosomes after a predefined number of generations. Based on the observations made by Gonçalves et al. (2012), we decided to replace the worst two elite chromosomes from each independent population with the best two chromosomes from the combined populations after every 30 generations. The algorithm evolves the population by evolving encoded solutions rather than the “real” solution. The BRKGA encodes a solution through a chromosome with alleles, where is the number of production operations required. Each allele is a real value between [0, 1] and is initially randomly generated. A solution to a JSP with four jobs (e.g., four products), each requiring three production operations, can be represented as given in Figure 1a). Elements 1 to (3) refer to the operations of job 1, elements (4) to (6) to the ones of job 2, and so on. The decoding procedure sorts the vector of random keys in ascending order, see Figure 1b), and converts the indices of the sorted vector into a sequence of repeated jobs, which is then converted into a sequence of operations by replacing (from left to right) the appearance of job by operation , as represented in Figure 1c); this way, it ensures the feasibility of the solutions. The BRKGA has a few parameters (which influence its performance) that need to be set. The values recommended for these parameters are available from Gonçalves et al. (2018). The parameters chosen were based on previous work by Rizzi et al. (2015), and Fontes et al. (2023), and were empirically selected by testing them on a subset of benchmark instances (without deterioration effects). The stop criterion is finding an optimal solution, running 100 generations without improvement, or running a total of 500 generations, whichever happens first. The parameters selected were , , , , , , and .

Uma imagem com texto, captura de ecrã, número, Tipo de letra

Descrição gerada automaticamente

Figure 1. Translating a chromosome into a list of ordered operations.

* + 1. Deterioration effects

To study the influence of machine deterioration (in which the processing time depends on the machine’s running time), we consider deterioration functions of three types, namely linear, exponential, and sigmoid, see equations (1) to (3). Following the work by Wu *et al.* (2019), we consider that the processing time increases until it reaches an upper limit after which it stabilizes at the chosen value. For the sigmoid function, assuming that the processing time stabilizes at twice its processing time was acceptable. For the exponential function, empirical tests were performed to define the limit. In all equations, refers to the actual processing time of operation of job , while refers to the reference processing time of the same operation, *i.e.*, without deterioration.

|  |  |
| --- | --- |
|  | (4) |
|  | (5) |
|  | (6) |

In the above equations, the running time is expressed by the sum of the processing time of the operations since the machine has started operation. Hence, for each machine, the running time is calculated as the sum of the processing times overall operations it processes and that have been started on or before . In these equations, represents the deterioration rate of each machine, which is job independent.

The prediction of degradation rate is very complex in batch processes with multiple products. Different products require different batch recipes, such as varying raw material ratios, additives, and operating conditions, resulting in distinct deterioration processes. Consequently, predicting the deterioration rate becomes challenging and requires real-time data. Thus, in this study, for all three function types, the deterioration rates were randomly drawn from the uniform distribution . The effect was not considered in all machines to bring the study closer to a real case. We select randomly a fixed percentage of machines to be affected by deterioration (about 25%).

* 1. Results and Discussion

Tests were conducted on a computer with an Intel(R) Core (TM) i5-2400 Processor (3.10 GHz) running on the Windows 10 operating system, and the algorithm was implemented in Python® 3.11. Each instance was run 10 times.

* + 1. BRKGA efficiency

To demonstrate the effectiveness of the BRKGA to solve JSP, instances from Lawrence (1984) were considered. Table 1 shows the instances’ settings, as well as the optimal total production time (makespan) from literature (*Opt*), the best solution found, the percent optimality gap (GAP), the percentage of makespan deviation (σ), and the execution time needed to find the best solution.

Table 1. Lawrence (1984) instances and computational results for BRKGA.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Characteristics | | | BRKGA | | | |
| Instance | M-J-O |  | Best | GAP | σ% | t/s |
| la04 | 5-10-50 | 590 | 597 | 1.2 | 3.3 | 17 |
| la08 | 5-15-75 | 863 | 863 | 0.0 | 0.1 | 11 |
| la12 | 5-20-100 | 1039 | 1039 | 0.0 | 0.0 | 6 |
| la16 | 10-10-100 | 945 | 946 | 0.1 | 5.2 | 29 |
| la17 | 10-10-100 | 784 | 796 | 1.5 | 3.6 | 30 |
| la22 | 10-15-150 | 927 | 989 | 6.7 | 3.5 | 50 |
| la23 | 10-15-150 | 1032 | 1045 | 1.3 | 3.0 | 84 |
| la28 | 10-20-200 | 1216 | 1269 | 4.4 | 4.5 | 157 |
| la31 | 10-30-300 | 1784 | 1784 | 0.0 | 0.2 | 95 |
| la36 | 15-15-225 | 1268 | 1340 | 5.7 | 4.1 | 159 |
| **Mean** | | 1044.8 | 1066.8 | 2.1 | 2.7 | 64 |

Overall, the BRKGA performs well in these instances, with the mean best solution of 1066.8, which is only 2.1 % away from the mean optimal solution. The algorithm also has a relatively low standard deviation, indicating consistent performance, *i.e.*, robustness. The time required to find the best solution varies depending on the instance configuration. Then, the multi-population was implemented (BRKGA+MP), and the results obtained compared to those of Rizzi *et al.* (2015), who employed a clustering search in BRKGA (BRKGA+CS), see Table 2.

Table 2. Computational results for BRKGA+MP and comparation with literature values.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BRKGA+MP | | | | BRKGA+CS | | | |
| Inst. | Best | GAP | σ | t/s | Best | GAP | σ | t/s |
| la04 | 590 | 0.0 | 1.9 | 36 | 590 | 0.0 | 7.8 | 14 |
| la08 | 863 | 0.0 | 0.0 | 8 | 863 | 0.0 | 5.4 | 6 |
| la12 | 1039 | 0.0 | 0.0 | 7 | 1039 | 0.0 | 6.1 | 6 |
| la16 | 946 | 0.1 | 4.8 | 49 | 946 | 0.1 | 10.8 | 22 |
| la17 | 785 | 0.1 | 1.7 | 50 | 784 | 0.0 | 8.8 | 47 |
| la22 | 954 | 2.9 | 3.3 | 74 | 942 | 1.6 | 12.0 | 25 |
| la23 | 1032 | 0.0 | 2.3 | 93 | 1032 | 0.0 | 8.1 | 53 |
| la28 | 1241 | 2.1 | 4.5 | 152 | 1258 | 3.5 | 10.9 | 11 |
| la31 | 1784 | 0.0 | 0.2 | 101 | 1784 | 0.0 | 4.5 | 67 |
| la36 | 1315 | 3.7 | 3.5 | 85 | 1315 | 3.7 | 10.3 | 75 |
| **Mean** | 1054.9 | 0.9 | 2.2 | 66 | 1055.3 | 0.9 | 8.5 | 33 |

The results show that BRKGA and BRKGA+MP performed well, but the BRKGA+MP consistently provided better solutions. This variant achieved results that were either optimal or very close to optimality. The BRKGA+CS variant performed slightly worse than BRKGA+MP, with a slightly higher mean best solution and a significantly higher standard deviation. Nevertheless, it required less time than the BRKGA+MP variant. The computational time required by BRKGA, and its variants was generally moderate, with most instances solved within a few minutes or less. However, the time required to find the best solution was higher for BRKGA+MP, which was expected since this variant involves multiple populations, increasing the complexity and time required to find a solution. Based on the results obtained, BRKGA+MP is better in terms of solution quality, consistency, and time, and therefore, it was used to account for deterioration.

* + 1. Machine deterioration

Firstly, for the best solutions found by BRKGA+MP, *i.e.*, disregarding the deterioration, the makespan was recalculated by considering the deterioration afterward, that is, without optimizing the “real” processing time (w/out opt.). Then, the instances were solved again, considering processing times that incorporate the deterioration (w/ opt.). The results of both cases are provided in Table 3, as well as the computational time required for the optimization, and the makespan improvement percentage. In all cases, optimization addressing deterioration proved to be better than considering it after solving the problem, as expected.

Table 3. Computational results for linear, exponential, and sigmoid machine deterioration.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Linear | | | | Exponential | | | | Sigmoid | | | |
|  | W/out opt. | W/ opt. | s | /% | W/out opt. | W/ opt. | /s | /% | W/out opt. | W/ opt. | /s | /% |
| la04 | 719.9 | 658.2 | 17 | 9.4 | 783.5 | 748.5 | 3 | 4.7 | 1043.3 | 1008.3 | 5 | 3.5 |
| la08 | 871.4 | 863.0 | 14 | 1.0 | 879.5 | 863.0 | 16 | 1.9 | 1444.2 | 1405.6 | 24 | 2.7 |
| la12 | 4407.6 | 3438.8 | 69 | 28.2 | 2699.0 | 2481.9 | 1 | 8.7 | 2051.9 | 1989.0 | 0 | 3.2 |
| la16 | 1116.6 | 1060.3 | 108 | 5.3 | 1339.0 | 1156.2 | 20 | 15.8 | 1357.5 | 1186.0 | 24 | 14.5 |
| la17 | 1314.6 | 1197.7 | 73 | 9.8 | 1362.3 | 1269.3 | 17 | 7.3 | 1221.0 | 1128.0 | 27 | 8.2 |
| la22 | 2346.8 | 1844.1 | 72 | 27.3 | 1546.9 | 1541.9 | 6 | 0.3 | 1708.6 | 1501.1 | 52 | 13.8 |
| la23 | 2643.3 | 2191.8 | 136 | 20.6 | 2313.4 | 2162.4 | 7 | 7.0 | 1714.0 | 1567.0 | 35 | 9.4 |
| la28 | 3174.0 | 2772.5 | 115 | 14.5 | 2089.8 | 2015.3 | 43 | 3.7 | 2225.8 | 2099.8 | 120 | 6.0 |
| la31 | 16774.9 | 8699.1 | 235 | 92.8 | 3472.9 | 3187.1 | 11 | 9.0 | 2865.8 | 2717.0 | 70 | 5.5 |
| la36 | 1817.7 | 1514.8 | 194 | 20.0 | 1547.3 | 1386.0 | 232 | 11.6 | 1855.0 | 1556.7 | 204 | 19.2 |
| **Mean** | 3518.7 | 2424.1 | 103 | 22.9 | 1803.4 | 1681.2 | 36 | 7.0 | 1748.7 | 1615.9 | 56 | 8.6 |

Regardless of the deterioration function under consideration, optimization addressing deterioration proved to be better than considering it after solving the problem. The optimization proved to be more efficient in the linear case, in which the makespan improved, on average, about 23 %. In this case, the deterioration had the highest impact on makespan, increasing by 234 %. Hence, the optimization eliminated about 10 % of the imposed increase. The improvement in the linear deterioration is as high as 92.8 %, for instance la31, which experienced the highest increase in the makespan. This case proves, again, the extreme importance of taking deterioration into account in industrial processes. In the presence of an exponential deterioration, an improvement of about 7 %, on average, was obtained. In this case, by optimizing, considering the deterioration effects, about 10 % of the imposed increase on the makespan was reduced. In the sigmoid case, the average improvement amounts to 9 %. Still, it was able to mitigate the deterioration effects since about 13 % of the deterioration impact can be avoided. However, this case was the case where deterioration caused a lower increase in makespan, only increasing by about 65 %. In this case, despite operational reorganization, the process maintains a fixed reference processing time, *i.e.*, the machines take longer to process the operations regardless of the operations sequence. The results prove that optimizing machine deterioration can improve production time across all instances. Future research should focus on integrating maintenance and exploring alternative decoders to improve the algorithm’s efficiency and the quality of the obtained solutions.

* 1. Conclusions

This study investigates the JSP with machine deterioration and its impact on production productivity. Companies should be made aware to consider machine deterioration in the decision process. We propose a multi-population BRKGA and demonstrate its efficiency and effectiveness in solving JSP instances. In particular, the algorithm consistently outperformed the clustering search in terms of solution quality and solution robustness, although requiring some additional computational time. The multi-population BRKGA was also used to solve JSP instances with machine deterioration. Although machine deterioration impacts the makespan, one can successfully mitigate such impact by using a proper strategy, as reported results. We have considered linear, exponential, and sigmoid time deterioration functions. Our solution approach was able to (almost) eliminate the impact of deterioration on the makespan. This study highlights the importance of addressing deterioration in industrial processes. Optimization techniques that account for deterioration can reduce costs, minimize disruptions, and enhance sustainability in industrial operations.

Acknowledgements

This work was financially supported by UIDB/50020/2020 and UIDP/50020/2020 (LSRE-LCM), and LA/P/0045/2020 (ALiCE), funded by national funds through FCT/MCTES (PIDDAC).

References

Browne, S., & Yechiali, U. (1990). Scheduling Deteriorating Jobs on a Single Processor. Operations Research*,* 38, 495-498.

Fontes, D. B. M. M., Homayouni, S. M., & Fernandes, J. C. (2023). Energy-efficient job shop scheduling problem with transport resources considering speed adjustable resources. International Journal of Production Research, 1-24.

Gonçalves, J. F., & Resende, M. G. C. (2012). A parallel multi-population biased random-key genetic algorithm for a container loading problem. Computers & Operations Research*,* 39, 179-190.

Gonçalves, J. F., & Resende, M. G. C. (2018). Random-Key Genetic Algorithms. In R. Martí, P. M. Pardalos & M. G. C. Resende (Eds.), Handbook of Heuristics (pp. 703-715). Cham: Springer International Publishing.

Gupta, J. N. D., & Gupta, S. K. (1988). Single facility scheduling with nonlinear processing times. Computers & Industrial Engineering*,* 14, 387-393.

Lawrence, S. (1984). Resouce constrained project scheduling: An experimental investigation of heuristic scheduling techniques (Supplement). In Graduate School of Industrial Administration, Carnegie-Mellon University.

Pei, J., Zhou, Y., Yan, P., & Pardalos, P. M. (2022). A concise guide to scheduling with learning and deteriorating effects. International Journal of Production Research, 1-22.

Rizzi, M. M., Chaves, A. A., Senne, E. L. F., & Lorena, L. A. N. (2015). Metaheurística híbrida aplicada ao problema de job shop. In Simpósio Brasileiro de Pesquisa Operacional (pp. 1735-1744). Porto de Galinhas, Pernambuco.

Wu, X., Shen, X., & Li, C. (2019). The flexible job-shop scheduling problem considering deterioration effect and energy consumption simultaneously. Computers & Industrial Engineering*,* 135, 1004-1024.