XLVI CILAMCE
VITÓRIA - ES - BRAZIL - 2025
XLVI IBERO-LATIN AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGINEERING

Active Local Search Approach on Non-Delayed Solutions for the Job Shop Scheduling Problem

David Rios Betoni[1], Fábio Henrique Pereira[1]

1 *Universidade Nove de Julho – Informatics and Knowledge Management Graduate Program - PPGI/UNINOVE, Vergueiro Street 235, São Paulo, Brazil*
David.betoni@uni9.edu.br , Fabio.Pereira@uni9.edu.br

Abstract. This paper presents a method for solving the Job Shop Scheduling Problem (JSSP), a complex NP-hard problem in Operations Research with direct applications in production optimization. The approach is based on the Random-Key Genetic Algorithm (RKGA), which is recognized for balancing solution with quality and efficiency, combined with active local search. Unlike traditional works, which predominantly use complex approaches, this paper shows that a method that initially restricts the search to the space of non-delayed solutions can significantly reduce the search space. Afterward, a refinement mechanism is applied, introducing small delays to non-critical operations using local search to improve the makespan. This two-phase strategy enhances solution quality.. Experimental results on standard benchmark instances show that the refined solutions outperform those generated by conventional RKGA approaches. The findings confirm the effectiveness and efficiency of the proposed method in producing high-quality schedules with reduced makespan.

Keywords: Genetic Algorithms; Local Search; Active Solutions; Scheduling; Job Shop

Introduction

The Job Shop Scheduling Problem (JSSP) is a classic combinatorial optimization problem widely studied in Operations Research due to its industrial relevance [1, 2]. It involves determining the optimal sequence of operations for multiple jobs on a limited number of machines, aiming to minimize the total completion time (makespan). Because of its complexity and practical importance, the JSSP is classified as an NP-hard problem [2]. Efficient solutions to this problem can directly improve production sequencing, reduce idle machine time, and optimize overall productivity.

The difficulty in solving large-scale JSSP instances highlights the need for efficient metaheuristic approaches that leverage computational resources effectively [3]. In this context, metaheuristics are valuable because they guide other heuristics (computational procedures) to find "good enough" solutions efficiently, even without guaranteeing optimality [4]. Genetic Algorithms (GAs) have gained prominence due to their robust global search capability and consistent performance across diverse optimization problems [5, 6].

This work proposes the use of a Random-Key Encoded Genetic Algorithm (RKGA), an approach introduced by [7], to solve the JSSP. The method is refined by an active local search strategy, which explores a specific point of the solution obtained by RKGA to improve the final quality of results, as demonstrated in previous studies [8] for the JSSP. To assess the effectiveness of the proposed methodology, computational experiments were conducted using well-known benchmark instances from the literature [9, 10], obtained from the OR-Library provided by [9].

Job Shop Scheduling Problem (JSSP)

This section presents the formal definition of the Job Shop Scheduling Problem (JSSP). It involves sequencing a set of tasks (jobs), denoted by: $J = \{J_1, J_2, ..., J_n\}$. Each job J_j consists of an ordered sequence of operations: O_{j1}, O_{j2}, ..., O_{jm}. Each operation must be processed on a specific machine M_k, which belongs to the set of machines M, for a fixed processing time p_{jk}. The goal is to determine a feasible schedule that minimizes the makespan — the total time required to complete all jobs. The problem also considers the following constraints, which are fundamental characteristics of the classical formulation of the Job Shop Scheduling Problem [8, 9, 2]:

- Each job has a fixed path through the machines.
- No operation can be processed on more than one machine at a time.
- Each machine can only execute one operation at a time.
- Operations cannot be interrupted once they have started.

Literature Review

This literature review focused on approaches based on Genetic Algorithms (GAs). The research covered recent literature and was conducted in reputable databases such as Google Scholar, CAPES, SciELO, and Scopus. GAs are known for their ability to efficiently explore the solution space and handle multiple objectives in NP-hard problems. Recent literature highlights advances reported in [5, 6, 11], including:

- Hybridization of GAs with local search methods;
- Use of representations such as random keys;
- Techniques for maintaining population diversity [11];
- Application of hyper-heuristics and multi-objective fitness functions.

These strategies have contributed to improving the effectiveness of GAs in solving problems such as JSSP. The literature shows a predominance of complex representations, specialized operators, and hybrid methods such as intensive local searches or adaptive hyper-heuristics.

Moreover, when delays are incorporated into the problem, the evaluation of solutions is adapted to reflect objectives such as minimizing total tardiness, the number of delayed jobs, or weighted tardiness penalties. Combined metrics or multi-objective fitness functions have been employed to capture the impact of delays on scheduling performance.

However, there is a gap in the exploration of algorithms with simpler, more modular, and easily interpretable structures — such as those based on random keys with explicit decoding by list scheduling heuristics. Additionally, some studies such as [6] propose modified GAs with advanced crossover mechanisms and hybrid strategies, achieving promising results for complex JSSP instances.

Methodology

For the validation of the proposed method, benchmark instances FT06 and LA01 to LA07 [9] were used, ensuring the comparability of makespan results.

The proposed solution process begins with a randomly generated population of chromosomes. The chromosome length is defined by the number of decision variables that need to be represented in the problem domain [12], where each individual represents a sequence of operations encoded using Random-Keys — numerical values randomly generated between 0 and 1. This encoding ensures the feasibility of solutions and simplifies the decoding process for sequencing operations across machines [13].

The Genetic Algorithm (GA) performs a global exploration of the solution space, evolving over 120 generations, which is sufficient to explore various combinations of solutions and helps avoid premature convergence to local optima [13]. The fitness function is defined by the makespan, i.e., the total time required to complete all tasks.

The evolutionary process combines the following mechanisms:

- Tournament selection, prioritizing individuals with lower makespan;
- Elitism, which preserves the best individual from each generation [12];
- Crossover, applied to 80% of the population, generating new individuals through chromosome recombination — a rate validated in empirical studies and recent reviews as an effective choice for genetic

algorithm applications [9];

- Gaussian mutation (2%), introducing small random variations to maintain genetic diversity and prevent premature convergence [14].

After the GA concludes, if the known optimal makespan value is reached, the process terminates; otherwise, a local refinement stage called Active Local Search is applied. This stage operates on non-delayed solutions (those where operations start as early as possible without unnecessary idle times).

The local search identifies the largest idle intervals (gaps) between the end of an operation on one machine and the start of the subsequent operation on another. These intervals are used as criteria to select perturbation points, where strategic delays are introduced in non-critical operations, aiming to explore small variations in sequencing and further reduce the total makespan.

Each newly generated configuration is re-evaluated by the genetic algorithm, and improvements are accepted iteratively until no significant gains are achieved or the global optimum is reached. The two-phase, of process of the approaches are represented at the diagram 1 and diagram 2 below.

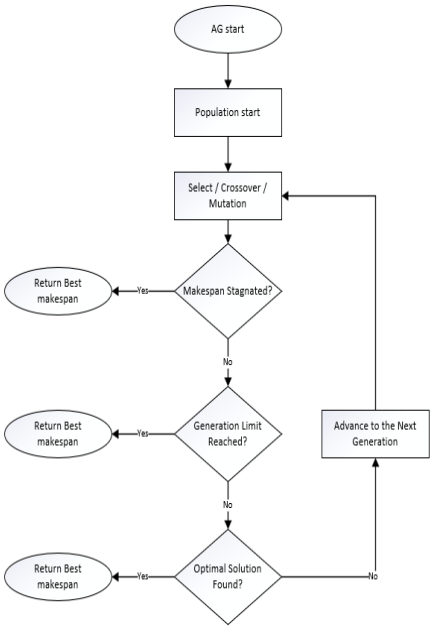


Diagram 1: Phase 1 - Global Search with RKGA

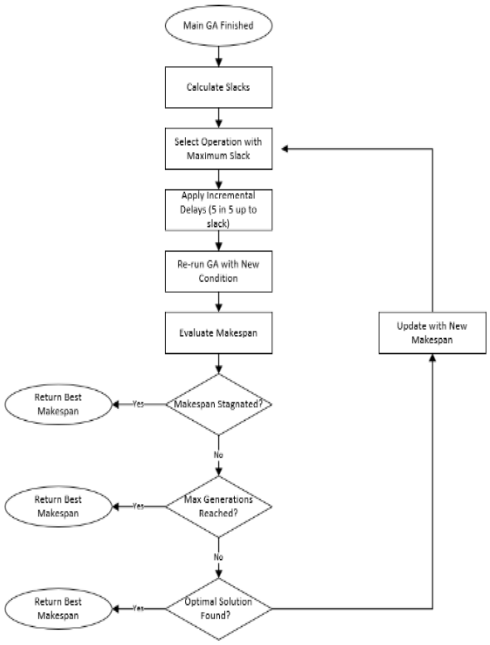


Diagram 2: Phase 2 - Local Search Refinement

The method was implemented in Python and executed on a system equipped with an Intel i7 processor and 32 GB of RAM, similar to other comparative studies on GA computational performance [2]. Performance was evaluated based on two main metrics: makespan reduction and execution time.

Figure 1 illustrates a typical JSSP solution represented by a Gantt chart, with an arrow indicating the exact point where the local search strategy is applied — that is, between jobs that exhibit the largest spacing between machines. In this example, the final makespan obtained was 57, and after applying the refinement stage, the makespan obtained was 55, and the order of the jobs was realocated, according demonstrated at Figure 2.

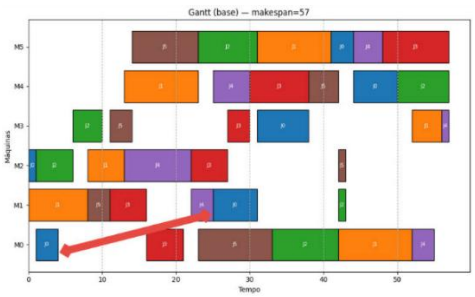


Figure 1: RKGA without local search approach

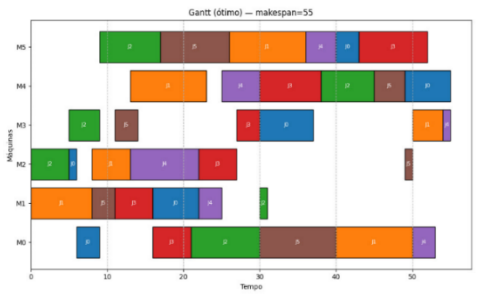


Figure 2: RKGA with local search approach

Results

The proposed approach for solving the JSSP was evaluated through computational experiments using benchmark instances from the literature (FT06 and LA01–LA07). Preliminary results demonstrated at table 1, shows the algorithm's effectiveness in reducing makespan, with improvements ranging from 0% to 23.03%, and demonstrated robust convergence behavior across all instances, successfully achieving optimal or near-optimal makespans in the majority of cases.
The algorithm performed especially well on instances (FT06, LA01, LA05, LA6 and LA7), consistently achieving the optimum, while in the others instances (LA02–LA04) exhibited slightly higher variability and runtime.

Instance	Size (J×M)	Initial Makespan (RKGA Range)	Known Optimal Makespan	Final Makespan Range	Improvement (%)	Avg. Execution Time (s)	Analysis (Optimum Reached?)
FT06	6×6	57–63	55	55	≈3.51–12.7	10.02	Yes (Best: 55 - 10/10 runs)
LA01	10×5	678–817	666	666	≈1.77–18.48	176.14	Yes (Best: 666 - 10/10 runs)
LA02	10×5	714–851	658	655–677	≈8.26–23.03	1113.25	Yes (Best: 655 - 1/10 runs)
LA03	10×5	630–732	615	615–626	≈3.49–16.94	840.01	Yes (Best: 615 - 1/10 runs)
LA04	10×5	640–730	598	602–623	≈5.94–17.53	1038.59	No (Best: 602)
LA05	10×5	593–692	593	593	≈0.0–14.31	2.22	Yes (Best: 593 - 10/10 runs)
LA06	15×5	975–1110	926	926	≈5.03–16.58	140.66	Yes (Best: 926 -10/10 runs)
LA07	15×5	934–1085	890	890	≈4.71–17.97	753.74	Yes (Best: 890 - 10/10 runs)

Table 1 - Evaluation Results

Conclusion

This work presented a hybrid approach that combines a Random Key Genetic Algorithm (RKGA) and an active local search strategy to address the JSSP. Using this method of generating a solution through a genetic algorithm applying targeted local refinements, the method obtained promising results in benchmark instances. The approach proved competent of reducing the makespan by up to 23.03% and achieving known optimal solutions in several cases, validating its effectiveness. Future research could explore multi-objective refinements and compare them with other metaheuristics, such as Iterated Local Search (ILS), Simulated Annealing (SA), Particle Swarm Optimization (PSO).

Acknowledgements. The authors acknowledge Capes (grant PROSUP) and Uninove for the scholarship awarded to the first author.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors or has the permission of the owners to be included here.

References

[1] M. L. Pinedo, Scheduling: Theory, Algorithms, and Systems, 5th ed. Springer, 2016.
[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 4th ed. Rio de Janeiro: LTC, 2022.
[3] J. Bordignon, L. C. Santos Junior, and F. H. Pereira, "Comparative performance study of GPU vs. CPU implementations of genetic algorithms in optimization problems," ENIC Uninove Conference, São Paulo, Brazil, 2016.
[4] K. Sörensen and F. Glover, "Metaheuristics," Encyclopedia of Operations Research and Management Science, pp. 960–970, 2013.
[5] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," Multimedia Tools and Applications, vol. 79, pp. 26859–26881, 2020.
[6] M. S. Viana, O. M. Junior, and R. C. Contreras, "A modified genetic algorithm with local search strategies and multi-crossover operator for the job shop scheduling problem," Sensors, vol. 20, no. 18, 2020.
[7] J. Bean, "Genetic algorithms and random keys for sequencing and optimization," ORSA Journal on Computing, vol. 6, no. 2, pp. 154–160, 1994.
[8] A. F. C. Rosa and F. H. Pereira, "An intensification approach based on fitness landscape characteristics for the job shop scheduling problem," Journal of Combinatorial Optimization, vol. 47, pp. 77–89, 2024.
[9] D. C. Mattfeld and R. J. M. Vaessens, "Job shop benchmark instances for optimization testing," OR-Library, 1996. Available: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/jobshopinfo.html
[10] H. Fisher and G. L. Thompson, "Probabilistic learning combinations of local job-shop scheduling rules," in Industrial Scheduling, Englewood Cliffs, NJ: Prentice-Hall, 1963, pp. 225–251.
[11] O. H. Constantino and C. Segura, "A parallel memetic algorithm with explicit management of diversity for the job shop scheduling problem," Applied Intelligence, vol. 51, pp. 5799–5816, 2021.
[12] A. E. Eiben and J. E. Smith, Introduction to Evolutionary Computing, 2nd ed. Springer, 2015.
[13] J. Gonçalves, J. Mendes, and M. G. C. Resende, "A hybrid genetic algorithm for the job shop scheduling problem," European Journal of Operational Research, vol. 167, no. 1, pp. 77–95, 2005.
[14] O. Bell, "Applications of Gaussian mutation for self-adaptation in evolutionary genetic algorithms," arXiv preprint, arXiv:2201.00285, 2022.