36th European Modeling & Simulation Symposium, 027
21st International Multidisciplinary Modeling & Simulation Multiconference

2724-0029 © 2024 The Authors.
doi: 10.46354/i3m.2024.emss.027

Convergence Analysis of Genetic Algorithms on Dynamic Production Scheduling

Michael Heckmann[1,*], Bernhard Werth[1,2] and Stefan Wagner[1]

[1]Josef Ressel Center for Adaptive Optimization in Dynamic Environments, University of Applied Sciences Upper Austria, Softwarepark 11, Hagenberg, 4323, Austria
[2]Institute for Symbolic Artificial Intelligence
Johannes Kepler University Linz, Altenberger Straße 69, 4040 Linz, Austria

[*]Corresponding author. Email address: michael.heckmann@fh-hagenberg.at

Abstract

The paper analyses the convergence behavior of the open-ended relevant alleles preserving genetic algorithm (OERAPGA) in dynamic production scheduling. In a dynamic production environment, frequent changes to the scheduling problem influence the convergence behavior of the applied genetic algorithm. This study investigates the impact of the two types of changes on the optimization process: removing the first task in the current solution from the problem and randomly removing one material along with the sub-materials from the problem. The impact of the changes is tested for different intervals, affecting the optimizer problem update frequency. The research findings show that frequent and substantive changes significantly reduce the convergence rate and can potentially halt convergence. For less aggressive changes, withholding problem update information demonstrated mixed results regarding the convergence rate, but impacted the optimization quality negatively. Ultimately, it is concluded that frequent updating results in the best optimization results, even if the optimizer does not converge. This is counter-intuitive coming from static optimization.

Keywords: dynamic optimization; scheduling; genetic algorithm; convergence

1. Introduction

Dynamic production scheduling has been a relevant topic for quite some time now. It is especially interesting due to its widespread area of application throughout the industry (Johnson et al., 2022; Tang et al., 2024). During a research project concerned with dynamically scheduling the production process of a window factory, with frequent changes to the problem, the convergence behavior of the applied open-ended relevant alleles preserving genetic algorithm optimization algorithm (OERAPGA) (Karder et al., 2022) gained our interest as it did not seem to converge at an expected rate.

In general, genetic algorithms (GA) are well suited for optimization in the real world, as they are flexible enough to deal with many dynamic changes. Examples of changes to the problem are information from the machines about finished work steps resulting in the removal of these steps from the problem, the modification of existing orders, the addition of new orders, and the updates to the work times of the machines. In the optimization of static problems, the convergence, although not premature convergence, of a genetic algorithm is desired to achieve the best results (Pandey et al., 2014). Experiments were conducted to evaluate if this behavior is also desired in the case of dynamic scheduling of production processes and assess the impact on the convergence rate of varying changes to the problem.

The paper is structured into several sections. In the state-of-the-art section, works about production optimization and convergence behavior of genetic algorithms are covered. The materials and methods section provides information about the problem used for the experiments, the algorithm used, the setup of the experiments, and how the data is analyzed. The results and discussion section covers the resulting data and analysis of the experiments regarding the convergence rate and the optimization quality. Finally, the conclusion summarizes the work done and gives an outlook.

2. State of the Art

While there is quite extensive research done evaluating the performance of different optimization approaches and their adaptations to different dynamic scheduling problems providing one with valuable knowledge on what might be the best approach for the task at hand, there is in comparison much less knowledge available on dynamic optimizer behavior. The main goal of this work is to target one specific aspect of behaviors: Genetic Algorithms (here RAPGA) usually display a convergence behavior. Genetic diversity is lost and the solutions, the algorithm keeps in its active population become more similar over time. While convergence is usually a problem-independent algorithmic behavior for static optimization, we will perform this analysis specifically for the explicit optimization of a dynamic production scheduling problem There are of course approaches for dynamic scheduling like e.g. Luo et al. (2020) by optimizing on a static problem and restarting the optimizer based on events in the production facility, but for approaches that are similar to the one benchmarked in this approach conclusions to their convergence effects may be drawn.

The majority of convergence analyses in production scheduling focus nonetheless on static tasks, which of course lack the adaptability of true dynamic optimization we want to achieve and mostly focus on the issue of premature convergence or speeding up the convergence to obtain optimization results quicker as a motivation for adaptions and new approaches (Werner, 2013).

The dynamic setting is covered significantly less, only Xu et al. (2022b) and Xu et al. (2022a) offer literature examples with quantitative measurements on the convergence. They observed the convergence to improve the parent selection process by comparing the convergence of different approaches as a key factor of an optimizer's success. This analysis offers valuable results to cross-reference but does not provide insights on how significant the impact of a pre- or post-mature convergence is to the overall performance, which we want to analyze. The efforts to provide insights into the success factors of the selection process in reinforcement learning approaches like in Xu et al. (2024) can be seen as a further motivation for all state-of-the-art approaches that apply to dynamic production scheduling on a general level to investigate the inner behavior more deeply.

The combination of convergence analysis and open-ended genetic algorithms, especially for use in production scheduling does not seem to be covered sufficiently, as open-ended optimization is still a relatively new topic compared to static optimization.

3. Materials and Methods

The experiments and the following analysis consist of several aspects covered in this section. The optimization algorithm and its corresponding quality metrics are defined. Additionally, the problem instance is described. Finally, the different setups for the experiments and how the experiment results are analyzed are described.

3.1. Dynamic Production Scheduling Problem

The production environment consists of nine machines and two storage buffers as depicted in Figure 1. The processing times of the machines are listed in Table 1. The machines strictly adhere to a worktime schedule, meaning that a processing step can only start during the machine's work time and must be done before the worktime ends. Additionally, the two buffers between the machines hold ten materials each.

Within this production environment, orders are to be processed. Orders consist of several products, often consisting of several sub-materials that are processed and assembled into the final products. How and in what order to process the materials is defined in the recipe of the final product. This also means that the assignment of the production machines is not to be optimized. An order is generated by randomly picking the number of sub-materials and the corresponding number and type of processing steps for each material. Additionally, the features of each material are also randomly chosen from a set of options.

To optimize this problem, the required steps in the product recipes are seen as a sequence of tasks. Considering all recipes, this results in a sequence containing all tasks to be planned. The order in this sequence is adjusted to improve the quality of the solution.

The solutions are evaluated by applying the tasks on a simulation based on the production environment. The simulation applies the task in the order specified in the current solution. Each task is started as soon as the machine is available (based on the processing times and the

Table 1. Processing times and step types of the production machines.

Machine	Step type	Processing times
1	initial	30 sec
2	initial	25 + *size* ∗ *thickness* sec
3	initial	120 sec
21	processing	30 sec
22	processing	30 sec
23	processing	30 sec
24	processing	30 sec
41	finalizing	30 sec
42	finalizing	30 sec

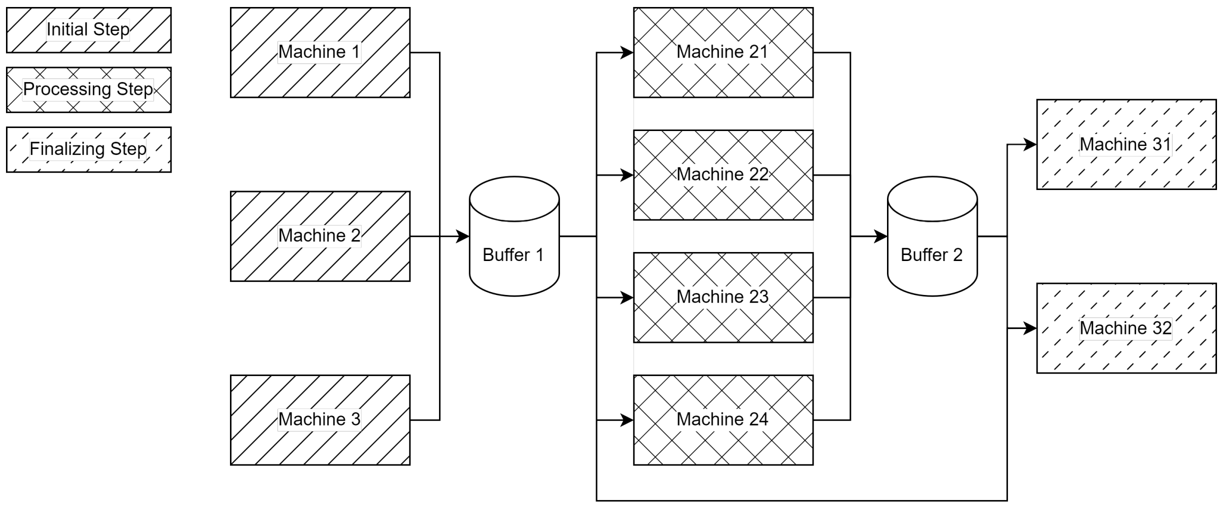

Figure 1. Production setup used for the experiments. The material is introduced into the production environment at the initial step and removed at the finalizing step. The buffers are of limited capacity and both buffers can supply materials to the finalizing step.

worktimes) and the buffer has sufficient capacity. This results in a series of actions done in the simulation from which the three quality metrics, over buffer, overdue-ness, and makespan are calculated.

3.2. Optimzation Algorithm

The genetic algorithm used for the experiments is an open-ended variant of the relevant alleles preserving genetic algorithm (RAPGA) (Affenzeller et al., 2007) which is a population-based optimizer that produces variations (offspring) of existing solutions (parents) in every iteration. The RAPGA has a varying population size based on how much of the offspring is better than its parents. The algorithm is converged when it is impossible to find better offspring through selection, crossover, and mutation. In the case of the OERAPGA, this triggers a reseed of the population, meaning that a new completely random population is initialized. To retain some information of the previous generation after reseeding, the elites from the last generation can be copied to the next population.

For the experiment, some configuring of the OERAPGA is required. The maximum effort, the number of offspring produced per generation, is set to 50. The comparison factor determines how much better a generated offspring must be to make it into the new population. It is set to 0, meaning the offspring must be better than at least one of the parents. Random selection is used to select parents to generate offspring. The mutation rate is set to 7%. As the crossover, the maximal preservative crossover (MPX) (Mühlenbein, 1992) is used. This configuration is constant for all experiment runs.

3.3. Quality Metrics

For this work, three quality metrics are used: The first metric is the maximum number of materials that would overfill the buffers at any time in the simulation. This usually happens due to poor ordering of the solution but could be unavoidable in theory, for more generalized production systems in extreme conditions. The second metric is the mean work time the materials are delayed regarding the due date specified in the order (MeanOverDueness). Finally, the last metric is the makespan, the work time required to finish all currently scheduled tasks. To make the makespan comparable across different problem sizes, it is divided by the total number of tasks resulting in the mean makespan. As this experiment does not use a multi-objective optimization, these quality metrics are ordered hierarchically so only when the first metric (OverBuffer) is equal, will the next metric (MeanOverDueness) be compared and finally if both of the first metrics are equal, the final criterium (MeanMakeSpan) is compared.

3.4. Experiment Setup

To simulate a dynamic production environment, after each generation of the OERAPGA, the problem that the algorithm is optimizing can be changed. Figure 2 shows the experiment setup with optimizer and experiment environment. For the experiments, it is possible to hold back the generational optimizer problem updates and only periodically update the problem, resulting in two problem versions, a current, external problem, and a slightly outdated optimizer problem. This gives the algorithm more time to converge. To still have comparable qualities for varying problem update rates, the solutions of a new generation are re-evaluated on the external problem after they are updated. Note that this hold-back effect may be introduced in practice inadvertently, by latency, synchronization efforts, or batching of production events.

The experiment environment also has some configurations to discuss. An experiment consists of 2000 generations of the GA and the optimization, problem update cycle

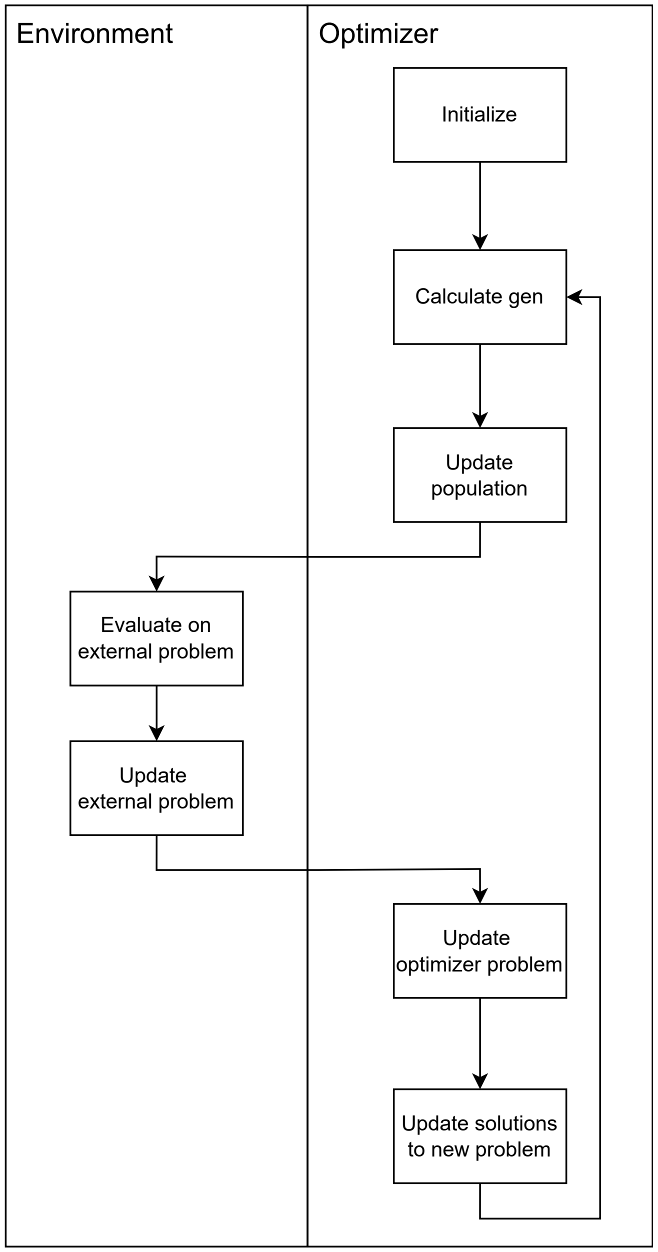

Figure 2. The experiment setup is split into its optimizer and environment tasks. This is an open-ended process, but the experiments are stopped after a predefined number of cycles.

is terminated when reached. As changes to the problem result in the removal of tasks from the problem, the problem size would reach zero if no new orders are supplied. For this, a new order is generated when less than 100 tasks are being optimized. This results in an implicit change to the problem and a fluctuating problem size throughout the experiment, which is also visible in the quality metrics.

For this work, two kinds of changes are considered. As in a production environment, the proposed schedule is processed from the start, resulting in the removal of tasks from the front of the schedule. This is the first kind of change examined. As the problem is changed based on the outcome of the optimizer, the quality metrics are not directly comparable. The values $\{1, 2, 5, 10, 20, 50\}$ for generations per change were tested and were chosen to examine a range of problem change intensities. The lowest change frequency of 50 generations per change, is roughly the convergence rate without any changes.

To also have an experiment with comparable quality metrics, the second kind of change randomly removes one final material and corresponding tasks from the problem. This approach results in identical problems regardless of the optimizer outcome. Compared to removing just the first task in the schedule, removing a whole material is a more significant change in the problem. The values $\{1, 2, 5, 10, 20, 50\}$ for generations per change were tested.

Additionally, the interval when the optimizer is updated with the current external problem is also evaluated with the values $\{1, 2, 10, 20, 50, 100, \infty\}$. The number given represents the interval in generations. Longer update intervals result in longer durations a new change is held back. This results in more undisturbed optimization time for the optimizer. To have a comparison with an undisturbed optimizer the setting for no problem update (∞) was added.

3.5. Analysis of Experiment Data

The experiments resulted in per-generation data regarding the optimization, containing information about the external problem size, optimizer problem size, selection pressure, quality metrics, and the experiment parameters. Primarily important are the selection pressure and the quality metrics. The selection pressure is the average ratio of how many children were created to produce a successful child over a generation. Figure 3 shows the selection pressure during the optimization run. The number of generations between convergences is calculated to compare the convergence behavior and quality of different parameter configurations.

Focusing on the selection pressure, each run is split into convergences by starting a new convergence after the selection pressure reaches the maximum value. The various line colors in Figure 3 depict the different convergences. With the convergences identified, the average convergence rate for a run is calculated by averaging the maximum generation number for each convergence. This maximum generation number is calculated for each convergence individually, thus starting at zero for each convergence. Since the experiments run a fixed number of generations, the last convergence run is likely not completed, thus skewing the results. To remedy this, the final convergence run is discarded. This average convergence rate is comparable to other runs with different parameterizations.

Furthermore, the optimization quality is examined to assess the effect of varying experiment parameters. This optimization quality is the best found quality from the population evaluated on the current, external problem.

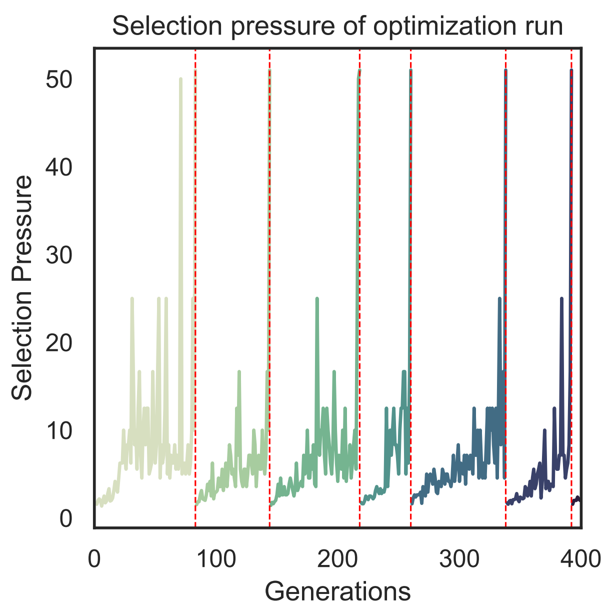

Figure 3. The selection pressure of an open-ended optimization run. The red lines mark the generations where the population converged and thus has reached the limit of selection pressure. Each convergence after is visualized by a different color.

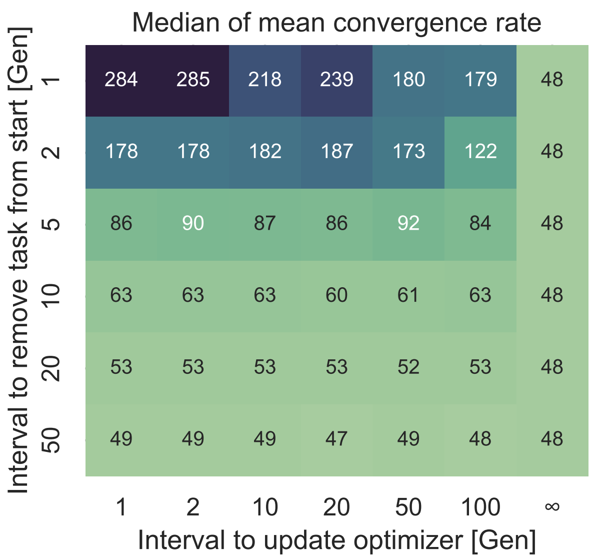

Figure 4. A heatmap showing the median of the mean convergence rate of repeated runs. The y-axis shows the varying interval of removing the first element in the solution from the external problem. The x-axis shows the optimizer problem update interval with ∞ meaning no updates beyond the initial setup.

4. Results and Discussion

The experiments are analyzed in two aspects. The first one is the impact of problem changes on the convergence rate. The second aspect is the resulting optimization quality for the two described problem change types and various optimizer problem update frequencies.

4.1. Convergence Rate

For both types of problem changes, the mean convergence rate is given as generations per convergence. For more stable results, each parameter configuration was run ten times, and the median of the mean convergence rate was shown.

Firstly, just removing the first task in the current solution from the problem, shown in Figure 4 is discussed. Looking at the y-axis it is clear that changing the problem significantly impacts the convergence rate. Compared to not updating the optimizer problem at all, as seen in the rightmost column, updating the optimizer problem increases the convergence rate about six times. Furthermore, there appears to be some separation between problem variations that change so fast, that the algorithm has severe trouble converging, and slower problem changes that barely impact convergence speed. Holding back changes, shown on the x-axis seems to improve the convergence rate only in a minor way, only showing a significant effect when the optimizer converges faster than the update interval. This is likely due to the increasing divergence between the external problem and the optimizer problem. Holding back the changes longer, results in collecting more changes for one large change, which seems to disrupt the

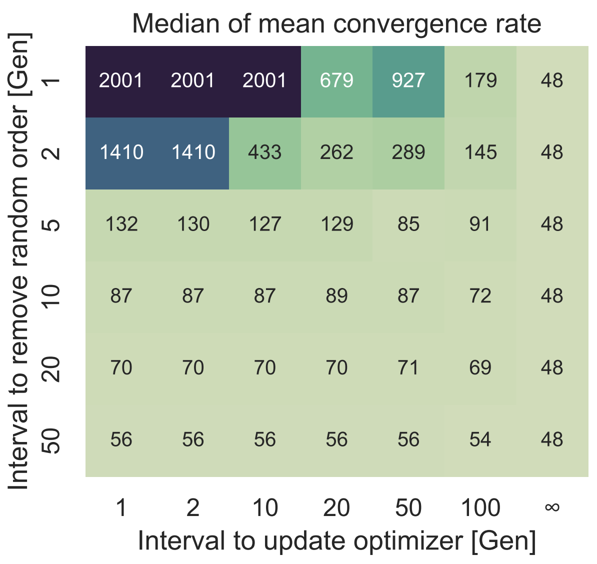

Figure 5. A heatmap showing the median of the mean convergence rate of repeated runs. The y-axis shows the varying interval of removing a random material from the external problem. The x-axis shows the optimizer problem update interval with ∞ meaning no updates beyond the initial setup.

optimizer nearly as much, as just updating immediately.

Figure 5 shows the experiments with random removal of one material. A similar trend is visible along the x-axis, the external update interval. The changes are more severe, so by updating the external problem each generation and updating the optimizer problem up to ten generation intervals, the optimizer does not converge within the 2000 generation limit in the median case. With the already larger change type, holding back the changes seems to affect the convergence rate positively.

4.2. Optimization Quality

Only experiments with the random removal of materials are considered to evaluate the optimization quality, as removing tasks from the problem based on the current solution would result in different problem states for all runs, changing the quality. Figure 6 shows the optimizer problem size and the first two quality metrics, for a range of optimizer problem updated interval values. Other values for generations per change parameter show a slowed down but similar picture, thus they were not visualized.

The over-buffer data in the figure clearly shows the effect of holding back the external problem changes. For update intervals of one, two, and ten generations the interval of the optimization problem update is still small enough to keep the drift not too far from the external problem, thus all are reaching the optimum of zero. Starting the update interval of 20 generations per update, the solutions optimized on the outdated optimizer problem do not match the quality of optimization run with a higher update frequency.

The second quality, the mean over dueness, seems not to be optimized at all, as the optimizer cannot optimize both the over buffer and over dueness at the rate of changes tested in the experiment. On a static problem, the optimizer takes hundreds of generations to optimize this second quality criterium.

Overall it does not seem like holding back updates improves the optimization quality of the OERAPGA. Due to the dynamic nature of the problem, holding back changes only leads to optimizing a past state of the problem that does not transfer well to the current external problem.

5. Conclusions

This work evaluates the impact of two types of changes to the dynamic scheduling problem optimized by OERAPGA with and without holding back updates to the optimizer. Changes with high frequency drastically reduce the convergence rate, even stopping convergence altogether when the changes are severe enough. For less aggressive

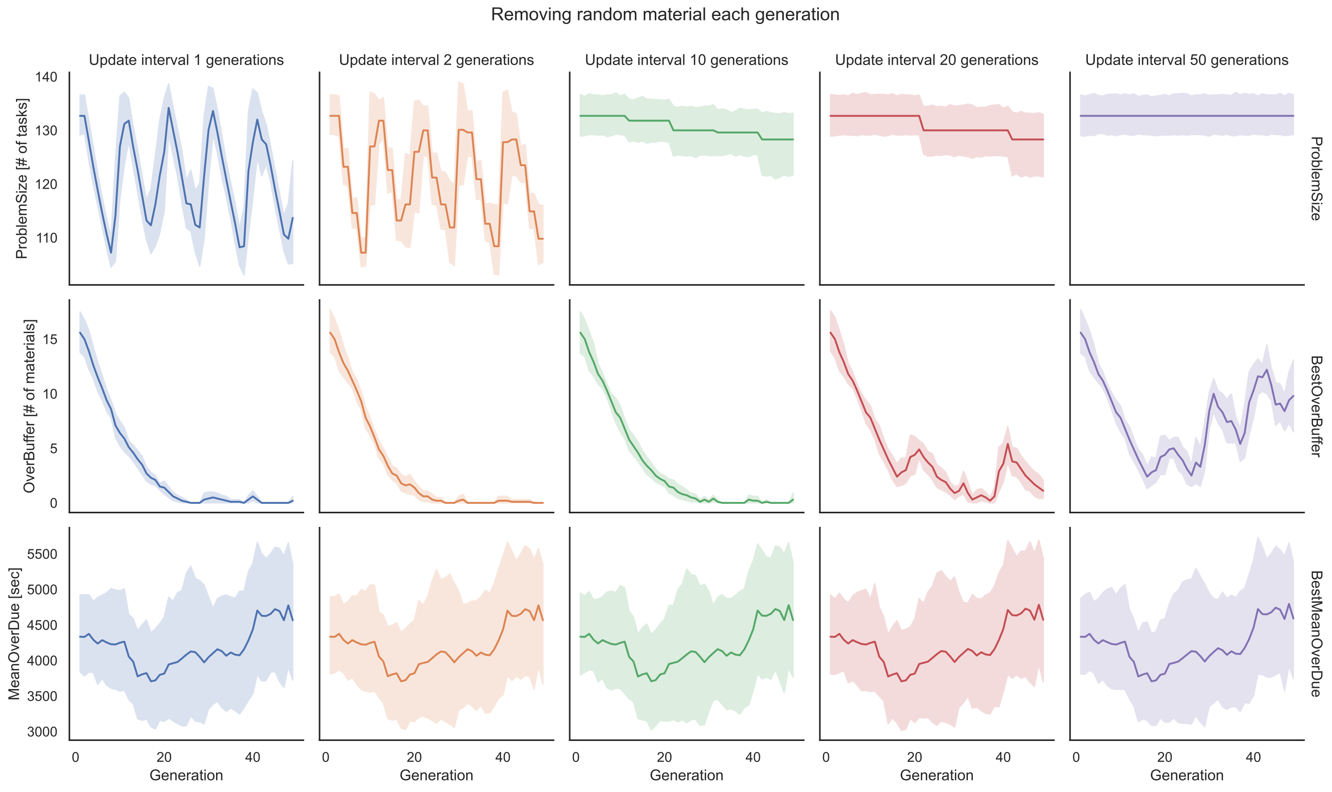

Figure 6. Lineplot showing the problem size and two of the three optimization criteria, for a set of experiment runs where one material is removed each generation. The first 50 generations are shown, so there is no elite from a previous convergence. Each column shows a different optimizer problem update interval. For the sake of legibility, not all parameter settings are visualized.

changes to the problem, holding back the information is not as effective as for more severe changes. Furthermore, the quality of the optimization was also evaluated. For the OERAPGA, updating the optimizer problem as fast as possible, resulted in the best optimization results. Indicating the counter-intuitive proposition, that convergence plays a less significant role in dynamic optimization performance. The initial assumption, that the population should converge, does not seem to be valid. Due to the dynamic nature of this particular case, problems regarding premature convergence were not relevant as opposed to other other papers on the topic.

To better understand the GA behavior on dynamic problems, more dynamic problems should be used for further experiments. Other problems could be interesting as the types of changes and the associated quality metrics might behave differently regarding the convergence rate and the resulting quality. Testing different open-ended optimization algorithms also seems like a good idea as the behavior of the RAPGA may not apply to other algorithms.

6. Acknowledgements

The financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development and the Christian Doppler Research Association is gratefully acknowledged.

References

Affenzeller, M., Wagner, S., and Winkler, S. (2007). Self-adaptive population size adjustment for genetic algorithms. In Moreno Díaz, R., Pichler, F., and Quesada Arencibia, A., editors, *Computer Aided Systems Theory – EUROCAST 2007*, pages 820–828, Berlin, Heidelberg. Springer Berlin Heidelberg.

Johnson, D., Chen, G., and Lu, Y. (2022). Multi-agent reinforcement learning for real-time dynamic production scheduling in a robot assembly cell. *IEEE Robotics and Automation Letters*, 7(3):7684–7691.

Karder, J., Werth, B., Beham, A., Wagner, S., and Affenzeller, M. (2022). Analysis and handling of dynamic problem changes in open-ended optimization. In *International Conference on Computer Aided Systems Theory*, pages 61–68. Springer.

Luo, J., El Baz, D., Xue, R., and Hu, J. (2020). Solving the dynamic energy aware job shop scheduling problem with the heterogeneous parallel genetic algorithm. *Future Generation Computer Systems*, 108:119–134.

Mühlenbein, H. (1992). Parallel genetic algorithms in combinatorial optimization. In BALCI, O., SHARDA, R., and ZENIOS, S. A., editors, *Computer Science and Operations Research*, pages 441–453. Pergamon, Amsterdam.

Pandey, H. M., Chaudhary, A., and Mehrotra, D. (2014). A comparative review of approaches to prevent premature convergence in ga. *Applied Soft Computing*, 24:1047–1077.

Tang, H., Xiao, Y., Zhang, W., Lei, D., Wang, J., and Xu, T. (2024). A dql-nsga-iii algorithm for solving the flexible job shop dynamic scheduling problem. *Expert Systems with Applications*, 237:121723.

Werner, F. (2013). *A Survey of Genetic Algorithms for Shop Scheduling Problems*, pages 161 – 222. Nova Science Publishers, Incorporated.

Xu, M., Mei, Y., Zhang, F., and Zhang, M. (2022a). Genetic programming with cluster selection for dynamic flexible job shop scheduling. In *2022 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8.

Xu, M., Mei, Y., Zhang, F., and Zhang, M. (2022b). Genetic programming with diverse partner selection for dynamic flexible job shop scheduling. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '22, page 615–618, New York, NY, USA. Association for Computing Machinery.

Xu, M., Mei, Y., Zhang, F., and Zhang, M. (2024). Genetic programming and reinforcement learning on learning heuristics for dynamic scheduling: A preliminary comparison. *IEEE Computational Intelligence Magazine*, 19(2):18–33.