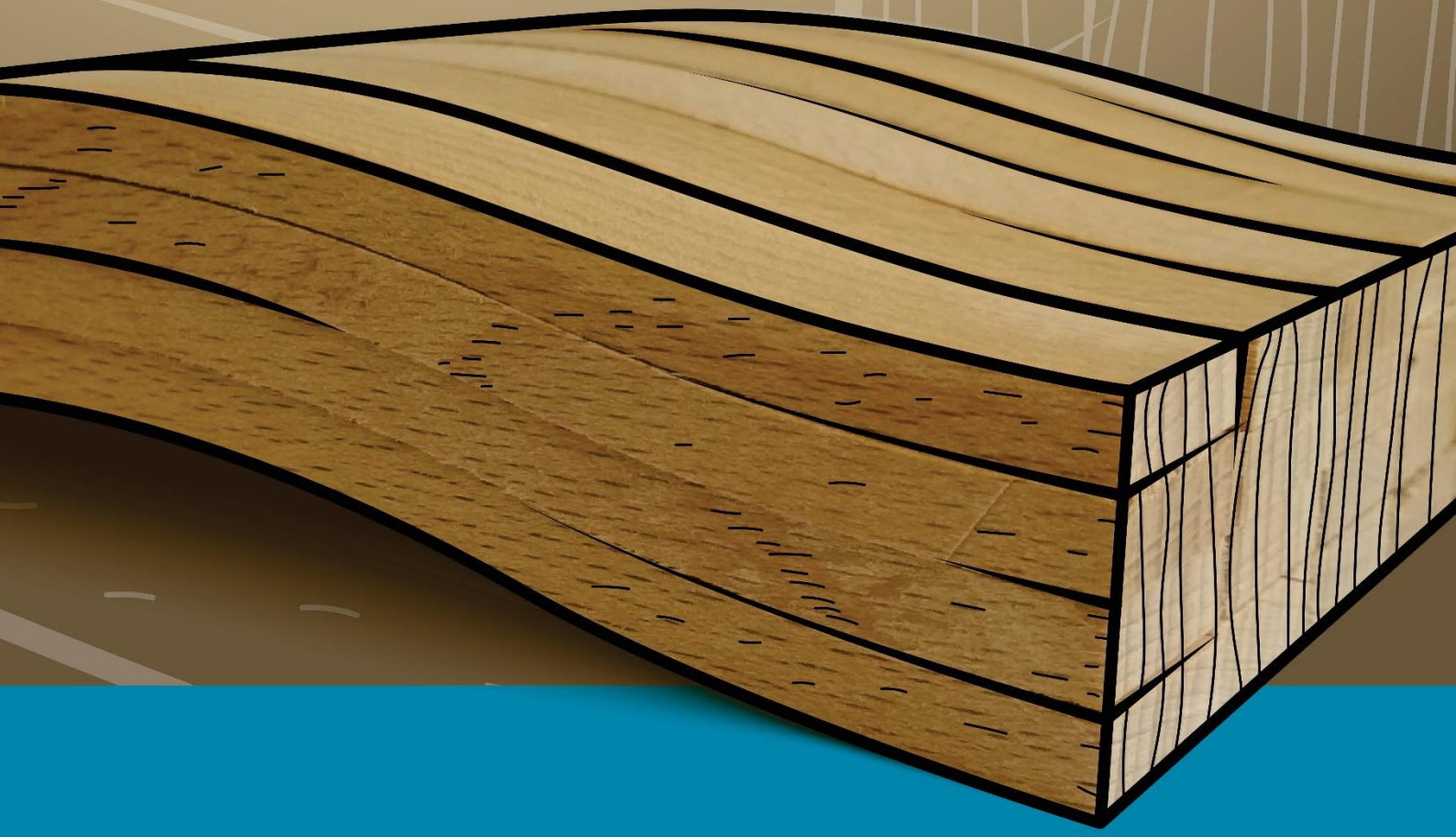
UNIVERSITY of SOPRON

FACULTY OF WOOD ENGINEERING AND CREATIVE INDUSTRIES

10th HARDWOOD Conference Proceedings

12–14 October 2022 Sopron

Editors: Róbert Németh, Christian Hansmann, Peter Rademacher, Miklós Bak, Mátyás Báder



WOOD KPLUS

Mendel University in Brno

10TH HARDWOOD CONFERENCE PROCEEDINGS

Editors: Róbert Németh, Christian Hansmann, Peter Rademacher, Miklós Bak, Mátyás Báder

UNIVERSITY OF SOPRON PRESS

SOPRON, 2022

10TH HARDWOOD CONFERENCE PROCEEDINGS

Sopron, Hungary, 12-14 October 2022

ISBN 978-963-334-446-0 (pdf)
DOI https://doi.org/10.35511/978-963-334-446-0
ISSN 2631-004X (Hardwood Conference Proceedings)

Constant Serial Editors: Róbert Németh, Miklós Bak

Cover image based on the beech specimens of Radim Rousek and Mátyás Báder by Miklós Bak, 2021

The manuscripts have been peer-reviewed by the editors and have not been subjected to linguistic revision.

In the articles, corresponding authors are marked with an asterisk (*) sign.

University of Sopron Press, 2022
Responsible for publication: Prof. Dr. Attila Fábián, rector of the University of Sopron

Long-term plant-level scheduling with uncertainty in the plywood industry

Olivér Ősz[1], Balázs Dávid[2,3], József Garab[1*], Máté Hegyháti[1]

[1] University of Sopron, Institute of Informatics and Mathematics, Hungary

[2] InnoRenew CoE, Slovenia

[3]University of Primorska, FAMNIT, Slovenia

E-mail: osz.oliver@uni-sopron.hu; balazs.david@innorenew.eu; balazs.david@famnit.upr.si; garab.jozsef@uni-sopron.hu; hegyhati.mate@uni-sopron.hu

Keywords: plywood, production, production scheduling, uncertainty

ABSTRACT

Plywood is a widely used wood-based product. Among others, plywood is used for buildings, furniture, or sports. Plywood is typically made in production plants, where the incoming material is processed through several transformation stages to satisfy the needs of the customer. Therefore, the production process is a complex procedure. Furthermore, production scheduling is a difficult and time-consuming task, as multiple orders, different machines, and production lines have to be taken into consideration simultaneously. Scheduling of production systems has been widely studied in the literature but production scheduling in the plywood industry has received little attention.

The scheduling problem of a plywood plant has been formulated through careful analysis of the selected problems in order to specify their hard and soft constraints and define their objective functions. In the case of long-term production planning, external aspects should also be taken into consideration. Unforeseen events, e.g., material shortage, changes in demands, disrupted processes, and quality issues can render the production plan infeasible. In such a case, the functionality of the system should be restored quickly, with minimal additional expenses and impact as possible. Recently, material resource availability disruptions have become a frequent issue due to epidemiological and geopolitical events affecting global supply chains. In previous work, the authors formulated Mixed-Integer Linear Programming models to minimize the cost caused by such supply disruptions in a plywood production facility. The results showed limited applicability for planning with long time horizons. Thus, the application of heuristic approaches is preferable. This study presents a Genetic Algorithm-based scheduling procedure for the plywood production process.

The aim of the proposed approach is to minimize penalties for order cancellations and deadline modifications. The operator of the facility may change the timing of each production step and swap the deadlines of orders from the same client to achieve this goal. The presented approach has been tested on numerous generated examples based on real-life industrial data from a Central European mill.

INTRODUCTION

Scheduling problems appear in almost every part of life (e.g., finding the shortest path by car to the destination or providing a robust execution plan for a manufacturing project). The goal is to assign tasks to some scarce resources and to time intervals in the most favourable way for a certain objective, while satisfying the constraints of the problem definition (Hegyháti 2015). Scheduling has an important role in the operational decision-making process in the manufacturing and service industries (Pinedo 2009).

In the frame of our research, plywood manufacturing was the focus. The plywood industry plays a significant role in hardwood utilisation and interesting scheduling problems arise at the processing of wood into final products. Several problems were investigated not only in the past (Koenigsberg 1961; Carlsson 1982) but also recently (Rikala & Sipi 2012; Mäkinen 2020; Ferretti 2021). Production planners are experienced employees in the plywood mill, who are responsible for scheduling the operation. Manual production planning is a time-consuming task that needs years of professional experience in the domain. Several customer orders, parallel processing machines (e.g., presses), and materials (e.g., peeled veneer, bought veneer) should be taken into consideration while production disruption is a daily factor. Furthermore, external factors like pandemic and energy supply disturbances can affect the production. Thus, any automated support for this kind of decisions is very beneficial.

The resolution of disruptions that occur to existing production schedules can happen in a variety of ways depending on their nature (Vieira et al. 2003). While disruptions that arise during the execution of a preliminary schedule must be addressed by a quick, small modification of the schedule as soon as possible to restore operational order, others, such as future material shortages can be foreseen earlier, and the remaining schedule can be reworked completely.
This paper introduces a long-term production planning problem for a plywood manufacturing plant, where the incoming orders cannot be satisfied necessarily on time, due to the shortage of incoming raw material deliveries. In addition to producing a feasible production schedule, decisions also have to be made about either cancelling certain orders or modifying their deadline by swapping them with ones that would be due later. The problem tackled in this paper was studied by the authors in a previous work (Őszet al. 2022), where exact solution approaches were implemented. However, these approaches were only tractable on small- and medium-sized problems, hence the motivation for the solution method developed in this paper.
The outline of the paper is the following. First, the plywood scheduling problem is introduced, and the constraints of the arising optimization problem are defined. Then, a Genetic Algorithm-based (GA) solution method is introduced for the proposed problem. Finally, computational results of this method are shown on randomly generated instance sets.

PROBLEM DEFINITION

Plywood manufacturing has two major stages: veneer production and plywood production. Both stages have several process steps, the key ones that were considered are shown in Fig. 1.

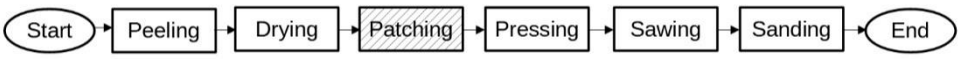


Figure 1: Simplified process flow of plywood production

Each order is completed by the sequence of 5 or 6 steps: peeling, drying, patching, pressing, sawing, and sanding. Patching is only required for certain subset of orders. Each step has dedicated machines with given throughput, and it is assumed that the storage space is sufficiently large between each stage, thus, an Unlimited Intermediate Storage policy can be considered.
Orders are placed by clients and entail a wood type, a quantity, and a deadline. Ideally, all the orders can be completed, and the mill accepts orders until they can be fit into its schedule. Recently, however, the supply of logs became unreliable. Thus, situations often occur, when the delayed supply shipments render the timely completion of the already accepted orders impossible.
Cancelling an order has major repercussions, however, there is often a less drastic option available: a client may allow to swap the delivery deadline of its orders for a certain (much smaller) fee. After these decisions, the operator has the freedom to time and allocate each of the production steps of accepted orders to the available machines.

The planner has the flexibility to decide:

- Whether an order is cancelled or not.
- Assignment of orders to agreed deadlines from the same client.
- Assignment of the steps of accepted orders to available machines.
- Timing of the steps of accepted orders.

The constraints of the problem are as follows:

- Each step of all the accepted orders must be completed in the correct order.
- A machine may not work on two orders at the same time.
- The completion of an order can only start when the required amount of wood logs is available.
- An accepted order must be completed before its assigned deadline.
- At any given time, the number of operators required by the machines cannot exceed the total number of operators at the facility.

The objective of the optimization problem is to minimize the total cost of order cancellations and deadline reassignments.

PROPOSED APPROACH

This paper suggests a Genetic Algorithm (GA) based approach. Genetic algorithms have been applied widely to various optimization problems, including project scheduling (Hartmann 1998; Hartmann 2001), various timetabling problems (Ross et al. 2003), and optimal job scheduling (Chang et al. 2005; Pezzella et al. 2008). Due to the nature of population-based evolutionary algorithms, the optimality of the best-reported solution cannot be guaranteed usually. However, it was concluded in the aforementioned work (Ősz et al. 2022) that exact approaches are only suitable for small and medium problems. For long-term scheduling, only heuristic approaches can be considered.
While a GA-based algorithm has many configuration parameters, its key parts are the chromosome representation, mutation and crossover operations, and the fitness function. The following subsections briefly introduce these parts of the approach.

Chromosome representation

In the proposed approach, each solution is represented by a single, homogeneous array of real numbers, *g*. The length of a vector is the total number of production steps. Technically, *g* does not encode the solution itself. Rather, it represents a blueprint from which a feasible schedule is generated deterministically within the fitness function. Each value within the array, *g[s]* encodes a lot of information about the production step *s:*

- The integral part represents the index of the assigned machine.
- The fractional part of *g[s]* indicates a priority among activities. Smaller means earlier.
- Cancellation of a step and thus its order is encoded in digits for the hundredths, thousandths, etc.

Fig. 2 illustrates this representation. The τ parameter for cancellation is a configuration parameter of the algorithm, like the population size, mutation/crossover rate, etc.

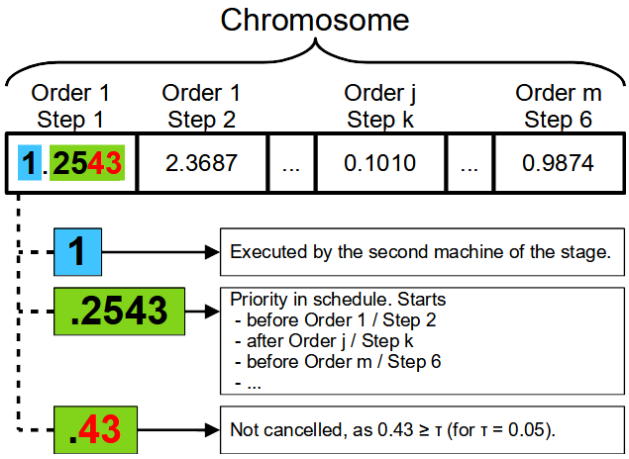


Figure 2: Explanation of chromosome representation

The only decision not encoded in a chromosome is the swapping of deadlines. That decision only affects the cost, not the schedule generation and it is handled in the fitness function.

Fitness function

The role of the fitness function is to provide a goodness value for a solution represented by a chromosome. In the proposed approach, the fitness function is more sophisticated, as it tries to fix infeasible solutions and finds the best suitable deadline assignment of orders. The schematic behaviour of the fitness function can be described as follows:

```
best := ∞
for all deadline assignments:
  while True:
    if cost(g, assignment) < best:
      g.preprocess()
      try:
            s := schedule(g, assignment)
            best := cost(g, assignment)
            break
      except deadline violation of order o:
            g.remove(o)
return s,best
```

For each possible deadline assignment that have a promising cost, a preprocessing step avoids recipe infeasible step priorities and excludes steps of a cancelled order. Then, a priority-based scheduling procedure tries to schedule the accepted orders. If the deadline of any of those gets violated, the scheduling is restarted with that order removed. If a feasible schedule is found, the solution is kept. At the end, the cost of the best solution is returned.

The schedule subroutine

The role of the `schedule` subroutine is to time each production step. Its schematic logic is as follows:

```
steps := g.get_ordered_active_steps()
timing := {}
for all s in steps:
    timing[s] := max (
        earliest_start_by_recipe(s, timing),
        earliest_start_by_operators(s, timing),
        earliest_start_by_machine(s, timing),
        earliest_start_by_log_shipments(s, timing)
    )
    if timing[s] > assignment.get_deadline(s.order):
        raise Deadline_violation(s.order)
return timing
```

First, the subroutine returns a sequence of steps for not cancelled orders, ordered by their priorities (fractional parts of the chromosome). The starting times of steps will be non-decreasing in this order. There are 4 circumstances that provide a lower limit on the starting of a step:

- the finishing time of the previous step in the order
- the earliest time when a sufficient number of operators are available
- the earliest time when the assigned unit becomes available
- if the step is peeling, the earliest time, when a sufficient amount of wood logs is available

The maximum of these 4 lower limits is the earliest time when the step can start. If that timing violates the assigned deadlines, an exception is raised.

Chromosome preprocessing

First, each chromosome gets preprocessed to avoid trivially infeasible solutions. If the fractional part (priority) of a step is lower than that of one of its predecessors, the resulting schedule cannot be feasible. Thus, the fractional part of the subsequent step is set to a slightly greater value than that of the predecessor step.

Moreover, if the cancellation of a step is indicated by the chromosome values, all steps of the order should be removed from consideration.

Mutation and crossover

Since the chromosome is an array of real numbers, where each number has a valid interpretation, very simple mutations and crossover functions are available.

The mutation first determines the number of changes to carry out, as a random integer between 1 and 5. Then randomly selects this many *g[s]* values from the chromosome and changes it by a random real value from the [-1,1] interval. If the result is negative or higher than the number of suitable machines, a modulo operation is used to normalize the value.

For crossover, a classic 1-point crossover operator is used. An index *s* is chosen randomly, then lower indexed values are copied from one parent, and the other values are copied from the other parent.

Algorithm parameters

The genetic algorithm has parameters that can be configured to finetune its behaviour. The population size can be adjusted based on the complexity of the fitness function, the available computational power, and the time limit. The crossover rate sets the ratio of solutions created by crossover of parents from the previous generation. Similarly, the mutation rate sets the ratio of solutions where the mutation operation is applied on them. Based on preliminary tests, crossover rate was set to 0.7, and mutation rate to 0.4.

Elitism can also be used to carry the best solutions to the next generation without altering them, thereby guaranteeing that the objective does not degrade over the evolution. The best 10% of the solutions were copied over this way in each generation.

Several stopping criteria can be used, which can also be configured through parameters. We used 2 criteria for stopping: if the best objective value did not improve for 100 generations, or the change of the average objective value of the generation is below a threshold for 50 generations.

EMPIRICAL TESTS & RESULTS

The described method has been implemented and ran on several randomly generated test cases. An open-source C++ genetic algorithm library, openGA (Mohammadi et al. 2017) was used for the efficient multithreaded implementation of the high-level algorithm, and the fitness function and genetic operators were implemented as described in the previous section. To validate the efficiency of the proposed method, its solutions were compared to those obtained by the MILP (Mixed-Integer Linear Programming) models presented in (Ősz et al. 2022).

Test setup

The tests were carried out on a laptop with an Intel i7-8750H 2.20 GHz CPU and 16 GB RAM. For solving the MILP model developed in our earlier work, Gurobi 9.5 was used. The population size for the GA was set to 2000.

Test instances were randomly generated with different parameter settings. The plant parameters such as the equipment inventory and the number of operators were the same for each case, only the parameters of clients, their orders, and the raw material deliveries were varied: the number of clients and orders, deadline, quantity, wood type, cancellation and swapping penalty, and incoming delivery times and quantities.

Smaller instances were generated for evaluation purposes, as these can be solved to optimality in a reasonable time by the MILP models developed earlier. This dataset includes 20 cases with 3 jobs belonging to 1 client, and 20 cases with 4 jobs with 2-2 belonging to 2 clients. These cases consider a 7-day-long time horizon.

Long-term scenarios were generated for a 30-day time period, in 4 datasets containing 10/10/12/15 jobs, belonging to 4/5/5/10 clients, with 10 instances in each dataset.

Results

From the 40 smaller test instances, the proposed GA was able to find the optimal solution in 36 cases. In 3 cases, it cancelled more orders than necessary, and in 1 case, it found the correct combination of orders but reported a solution with unnecessary swaps of delivery deadlines. For problems with this small size, exact approaches are more suitable, but this test showed that the GA metaheuristic can find optimal solutions in many cases. Some instances proved to be very difficult despite their size, as the MILP solver did not finish within 10 minutes in 4 cases and took more than 1 minute in 3 other cases, while the GA finished within 2-6 seconds consistently. The best solution was usually found in the first 10 generations, and the remainder of the runtime was spent on trying to improve it before a stopping criterion was met.

Fig. 3 presents the test results for the 4 long-term datasets. The MILP solver was stopped when the 10-minute time limit was met, and the best solution found until then was reported. It could only prove optimality for 2 instances within the time limit (599 s and 571 s), in both cases a 0-cost solution was found. For every other case, the GA solver returned a significantly better solution in a much shorter time. The difference between solution times of the datasets is caused by the number of possible deadline permutations. As delivery times can only be swapped if the orders belong to the same client, increasing the number of orders per client results in a higher computational load for building and evaluating schedules. If the number of permutations becomes too large, the proposed fitness function is not suitable. This shortcoming is a potential area for further research.

The number of generations varied between 112 and 266 for each dataset, but remained under 200 in most cases. The reason for stopping was the same in each case, that the best solution did not improve over 100 generations. Outstanding generation numbers and higher solution times for some cases were the results of the algorithm finding improving solutions after seemingly getting stuck in a local optimum. The most extreme example of that is instance 4 of dataset 1, where 266 generations were explored. The changes in the objective values over these generations are shown in Fig. 4. Such behaviour of the search is the reason behind the high values of the stopping criteria. Mutation and crossover operations can help the search in escaping from local optima.

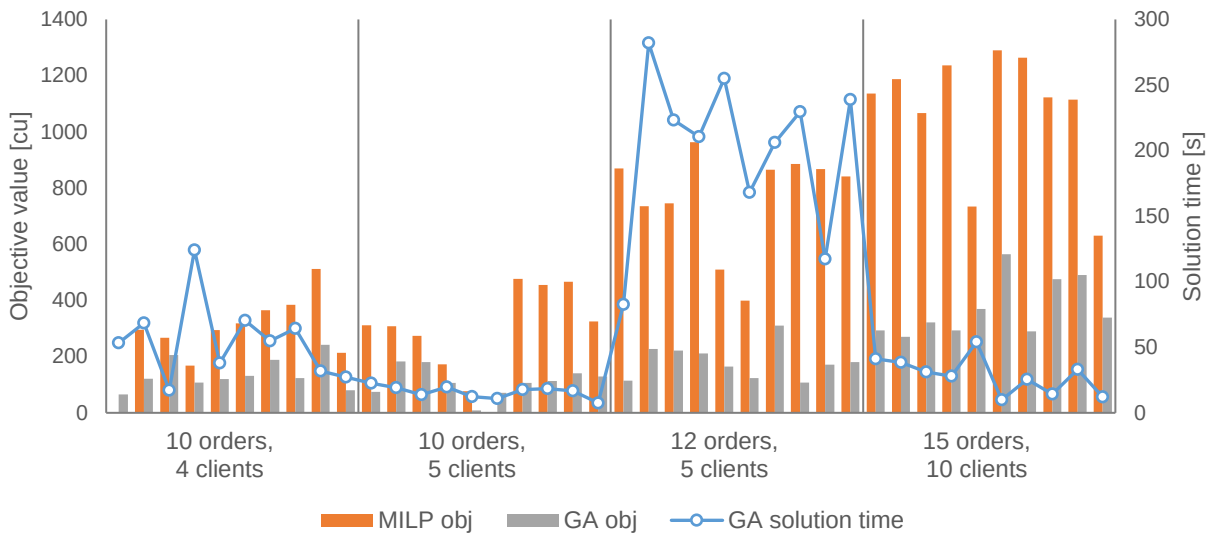


Figure 3: Results for long-term test cases

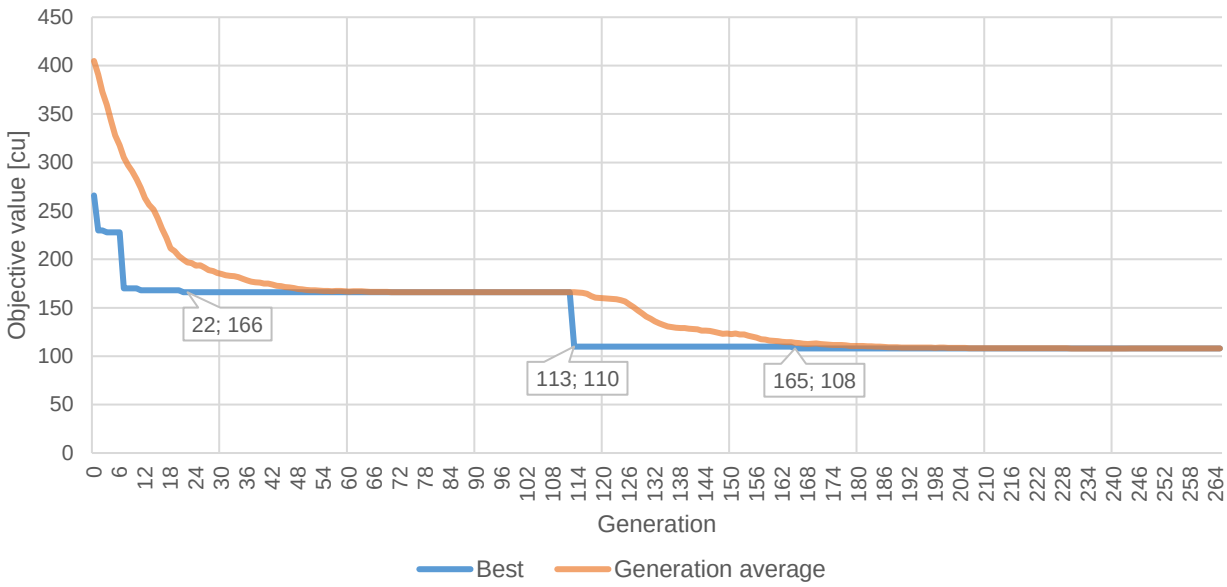


Figure 4: Improvement of the objective over generations

CONCLUSIONS

This paper studied the scheduling problem of a plywood plant under resource shortage, which might prevent the timely fulfilment of existing orders. To resolve this, cancellation of orders or the exchange of their deadlines was allowed as a scheduling decision. In order to acquire good quality schedules in a short time, a Genetic Algorithm was proposed for the solution of the problem. The efficiency of this algorithm was tested on randomly generated test instances, and the quality of the solutions was compared to ones given by MILP models under a time limit. The proposed GA managed to find better quality solutions than the MILP model in most of the cases in a significantly shorter running time.

ACKNOWLEDGEMENTS

This research was supported by the National Research, Development and Innovation Office, NKFIH grant no. 129178. Balázs Dávid gratefully acknowledges the European Commission for funding the InnoRenew CoE project (Grant Agreement #739574) under the Horizon2020 Widespread-Teaming program, and the Republic of Slovenia (Investment funding of the Republic of Slovenia and the European Union of the European Regional Development Fund). He is also grateful for the support of the Slovenian National Research Agency (ARRS) through grants N1-0093, N1-0223 and V4-2124, and the University of Primorska postdoc grant No. 2991-5/2021

REFERENCES

Carlsson, C. (1982). Tackling an MCDM-problem with the help of some results. European Journal of Operational Research, 10(3), 270-281.

Chang, P., Chen, S., & Lin, K. (2005, 10). Two-phase sub population genetic algorithm for parallel machine-scheduling problem. Expert Systems with Applications, 29(3), 705-712.

Ferretti, I. (2021). Optimization of the use of biomass residues in the poplar plywood sector. Procedia Computer Science, 180, 714-726.

Hartmann, S. (1998, 10). A competitive genetic algorithm for resource-constrained project scheduling. Naval Research Logistics, 45(7), 733-750.

Hartmann, S. (2001). Project Scheduling with Multiple Modes: A Genetic Algorithm. Annals of Operations Research, 102(1/4), 111-135.

Hegyháti, M. (2015). Extensions of the S-graph Framework (Ph.D. Thesis). Veszprém: University of Veszprém.

Koenigsberg, E. (1961). Some industrial applications of linear programming. Journal of the Operational Research Society, 12(2), 105-114.

Mäkinen, S. (2020). Flow shop scheduling of multi phase plywood production with paralell machines (M.Sc. Thesis). Espoo: Aalto University.

Mohammadi, A., Asadi, H., Mohamed, S., Nelson, K., & Nahavandi, S. (2017). OpenGA, a C++ Genetic Algorithm Library. 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (pp. 2051-2056). IEEE.

Ősz, O., Garab, J., Hegyháti, M., & Dávid, B. (2022). Mitigating Supply Chain Disruptions in Plywood Production by Rescheduling. Submitted to: Central European Journal of Operations Research.

Pezzella, F., Morganti, G., & Ciaschetti, G. (2008, 10). A genetic algorithm for the Flexible Job-shop Scheduling Problem. Computers & Operations Research, 35(10), 3202-3212.

Pinedo, M. L. (2009). Planning and Scheduling in Manufacturing Services. New York: Springer.

Rikala, J., & Sipi, M. (2012). Research and utilization of domestic hardwood species in Finland. Lővér Print, The 5th Conference on Hardwood Research and Utilisation in Europe, pp. 313-319. Sopron.

Ross, P., Hart, E., & Corne, D. (2003). Genetic Algorithms and Timetabling.

Vieira, G., Herrmann, J., & Lin, E. (2003). Rescheduling Manufacturing Systems: A Framework of Strategies, Policies, and Methods. Journal of Scheduling, 6(1), 39-62.