Integration of Optimization Methods into Simulation Technology for Manufacturing via Warehouse Optimization

H. Hakalahti*. A. Ala-Huikku. T. Luomanmäki***. J. Hirvonen****.**

**Digital Factory research group, Seinäjoki University of Applied Sciences, Finland (e-mail: hannu.hakalahti@seamk.fi).*

***Digital Factory research group, Seinäjoki University of Applied Sciences, Finland (e-mail: alisa.ala-huikku@seamk.fi).*

****Digital Factory research group, Seinäjoki University of Applied Sciences, Finland (e-mail: toni.luomanmaki@seamk.fi).*

*****Digital Factory research group, Seinäjoki University of Applied Sciences, Finland (e-mail: juha.hirvonen@seamk.fi).*

Abstract: The manufacturing industry is in a strong transition towards digital, intelligent, and sustainable manufacturing. However, small and medium-sized enterprises (SME) in the manufacturing industry often lack the resources and know-how to utilize digital tools as part of their research and development (R&D) activities. Thus, there is a need for concrete examples to show the benefits of these tools. This paper discusses a demonstration of warehouse optimization where a genetic algorithm is applied to optimize pallet transfers. The simulation model of a flexible manufacturing system (FMS) cell includes a warehouse with nine Euro pallets and a stacker crane. Visual Components simulation software was used for the simulation and an external Python application for the algorithm. As a result of the optimization, the total duration of the transfers was reduced by approximately 20 seconds (8.1 %). The demonstration has been used to showcase the integration of optimization methods into simulation technology and has ignited longer-term collaboration with the local industry on the same theme.

Keywords: 3D simulation, genetic algorithms, high bay rack, optimization, discrete event simulation

1. INTRODUCTION

Simulation and optimization are useful tools to understand, design, and develop production processes. For example, simulation can be used to model and analyze processes before they are implemented in the real production environment. Production processes can be optimized based on real-time or historical data. For example, several data analysis and computation models can be used to quickly locate equipment or quality problems and improve resource efficiency in production.

The combination of simulation and optimization enables a comprehensive analysis of production processes and efficient production planning. They can save costs, reduce production risks, and enhance product quality.

Bojic et al. (2023) did a case study using discrete-event simulation (DES) and genetic algorithm (GA) to assist operational production planning and optimization in the textile industry. A simulation model was created using Tecnomatix Plant Simulation software for a textile factory that produces over 300,000 shirts per year. The simulation model considered changes in customer demand, production times, available resources, and batch sizes. GA optimization improved production efficiency and reduced work in process (WIP) inventory levels.

Ernst et al. (2017) developed an optimization tool called Adv:ProcessOptimizer for multi-objective chemical process optimization. A specific GA was customized and developed for this tool. The tool integrates established methods with new concepts that work with simulation tools like Aspen Plus and ChemCad. The effectiveness of the tool was validated by optimizing an industrial styrene process. The results showed a well-distributed Pareto front, leading to savings in investment and operating costs compared to traditional methods. This confirmed the capability of Adv-tool to improve process efficiency and its support for decision making in process design.

Howard et al. (2023) developed a method to investigate energy flexibility in process cooling systems. A case study was performed on a Danish plant that uses process cooling for canned meat production. They used a combination of multi-agent, discrete-event, and system dynamics simulations to model the process. The results showed that significant savings in operational costs and reduction in CO2 emissions can be achieved by optimizing the schedule of the refrigeration units based on forecasts of weather conditions, electricity prices and CO2 emissions. This method provided insights on how to improve the energy performance of process cooling systems in food production without compromising the product quality and the production rate, through a weeklong simulation scenario.

Xie et al. (2015) used the genetic algorithm to schedule the single overhead crane so that its transport and shuffling operations are completed in the shortest possible time. The study found that the developed genetic algorithm provided good and quick solutions to the crane scheduling problem.

These case studies demonstrate how the integration of simulation and optimization tools works in various manufacturing industries. They can be useful for different kinds of applications, leading to more efficient, cost-effective, and high-quality production. However, SMEs in the manufacturing industry often lack resources and know-how to utilize these tools as part of their R&D activities or even information of their existence. Concrete pilots that demonstrate solving common problems in manufacturing industry are required to increase awareness of the possibilities of new simulation and optimization tools, and their integrations.

Warehouse operations provide an ideal demonstration environment for manufacturing SMEs due to their complexity, scalability, and the critical need for efficiency. It has been estimated that approximately 55 % of warehouse operating costs are caused by picking tasks (Bartholdi and Hackman, 2019, p. 25). Thus, increasing the picking and placing efficiency can generate remarkable savings.

Genetic algorithms are often used in optimization problems due to their robust search capabilities and flexibility as shown by two of the examples mentioned earlier in this section. The interest in genetic algorithms in the field of logistics has as well increased in the recent years among the researchers, and the number of publications has doubled between the years 2016 and 2020 (Grznár et al, 2021). In warehouse optimization problems reported in the literature, the goal often is to improve a forklift route in a warehouse described by a 2D layout consisting of shelves and corridors (e.g. Avdekins and Savrasovs, 2019; Kordos et al, 2020). Grznár et al (2021) worked with a 3D simulation of a conveyor system with workers sorting the goods coming to and leaving the warehouse. However, this work did not involve the actual warehouse structure.

This paper discusses optimizing an automated warehouse of a flexible manufacturing cell by utilizing a genetic algorithm. The high-bay rack and the stacker crane of the model add complexity due to vertical space utilization and dynamic movement, which are not typically addressed in simpler, 2D warehouse models. Also, the 3D simulation provides a more realistic and comprehensive testbed for evaluating the performance of GAs. Moreover, the FMS cell as a sample environment makes the demonstration interesting and accessible for the metal industry companies that are locally abundant. Finally, the demonstration involves cooperation of simulation software and an external optimization library, which gives a good example of the expansion potential of the applicable tools.

The paper is organized as follows: Section 2 describes the main methods applied in this study, discrete event-based simulation and genetic algorithms, and the software used. Section 3 discusses the experiments performed and the results gained. The conclusion is drawn in the last section.

2. METHODS AND SOFTWARE

2.1 Discrete Event-Based Simulation

According to Banks et al (2019, p. 3) a simulation replicates the functioning of a real-world process or system as it evolves over time. Whether the simulation is conducted manually or utilizing a computer, it entails creating an artificial history of a system and observing that history to make inferences about the real system's operating characteristics. Shannon (1998, p. 1) defines simulation as a process of modeling a real system and taking experiments with the model in order to gain insights about the behavior of the real-world system and furthermore, to evaluate different operational strategies for the system.

Banks et al (2019, p. 9) categorize systems as discrete or continuous. Choi and Kang (2013, p. 8) adds quantum class to enhance the system's classification. They agree together that rarely any process is purely a certain one, but more of a combination of two or more, however, some class describes more the system behavior than the other ones. Banks et al (2019, p. 9) define the discrete system in a way where state variable(s) change only at a discrete set of points in time. Furthermore, Choi and Kang (2013, p. 9) defines the discrete-event simulation as a computer evaluation of a discrete-event dynamic system model. In the model, the operation of the simulated system is defined as a chronological sequence of events. As the pilot case was a discrete manufacturing process, the simulation method was chosen accordingly.

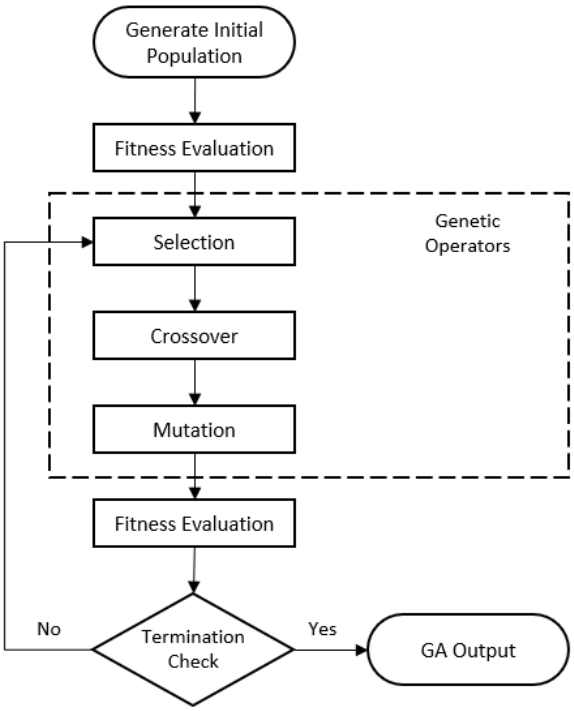

Fig. 1. Flowchart of the genetic algorithm.

2.2 Genetic Algorithms

2.2.1 Basics

A genetic algorithm developed by John Holland (Holland, 1975) is an optimization and search technique that is based on the principles of natural selection and genetics. In the genetic algorithm, a population of chromosomes evolve towards a better solution over each consecutive generation by using parent selection, crossover, and mutation. The genetic algorithms belong to the larger class of the evolutionary algorithms (EA) (Townsend, 2003).

The basic flowchart of the genetic algorithm is presented in Fig. 1.

2.2.2 Population

GA is an iterative process which starts with the creation of randomly chosen initial population of individuals (solution candidates for a given problem), which are represented by finite linear string of symbols, known as chromosome. A gene is an element in the chromosome, as shown in Fig. 2, and the allele is the value of the gene (Townsend, 2003).

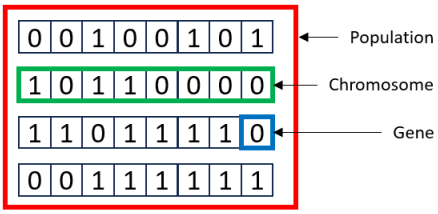

Fig. 2. Example of population, chromosome, and gene.

Determining the size of the population is important since choosing too small population might cause the GA to converge prematurely to a local minimum instead of global minimum due to lack of genetic variation in the population. On the other hand, too large population will require more computing time and thus makes the GA to run slower. The population size remains constant during the running of the genetic algorithm.

2.2.3 Genotype Representation

Genotype is a genetic composition of the chromosome (Haupt and Haupt, 2004).

A binary representation, where the chromosomes are represented as bits (a string of 1s and 0s), is the simplest and widely used representation. A floating-point representation is used for the continuous GA, and permutation representation is used for the cases where the order of genes matters (Fig. 3). Perhaps the most well-known use case of the permutation representation is the traveling salesman problem (TSP), where each city can be visited just once in some order.

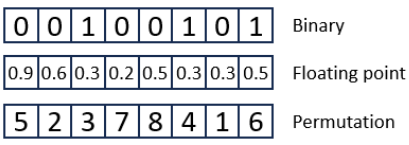

Fig. 3. Example of genotype representations in GA.

2.2.4 Fitness Function

At every evolutionary step, also known as generation, the current population is evaluated according to a fitness function set for a given problem. Because the fitness function calculates the fitness value for every chromosome on every generation, it greatly impacts the run time of the GA. Too computing heavy fitness function increases the run time of the GA (Townsend, 2003).

2.2.5 Selection

Parent selection operator selects the chromosomes in the population for reproduction. On every generation, the selected chromosomes are collected to a list known as mating pool. The better fitness value the chromosome has, the higher probability it has for being selected for the mating pool. Thus, the selection is based on the strategy of the survival-of-the-fittest (Townsend, 2003).

There are several different selection methods used in the genetic algorithms such as fitness-proportional selection, ranked selection, stochastic universal sampling, roulette wheel selection, truncation selection, and tournament selection (Townsend, 2003).

In tournament selection, a random number of individuals are selected from the population. Then, the best individual is selected from this group to be as a parent. This process is repeated until the mating pool is filled. Tournaments are often held between pairs of individuals (Goldberg & Deb, 1991).

2.2.6 Crossover

Crossover operator swaps the genetic material between two parent chromosomes to create new offspring for the next generation (Townsend, 2003).
The crossover between two good chromosomes does not necessarily create as fit or better offspring. However, because the parents are good, the probability of the offspring to be good is high, If the offspring happens to be a poor solution candidate, it will be removed from the population during the next generation.

In one-point or simple crossover, a random crossover point k is selected uniformly between 1 and the length of the parent chromosomes minus one [1, l – 1]. The genes after the crossover point k are then swapped between the parent chromosomes to create new offspring, as shown in Fig. 4 (Goldberg, 1989, p. 12).

In two-point crossover, two random crossover points are selected uniformly among the length of the parent chromosomes. The alternating segments of genes are then swapped between the parent chromosomes, as shown in Fig. 5. In general, the two-point crossover is better than one-point crossover to find solution more quickly (Townsend, 2003).

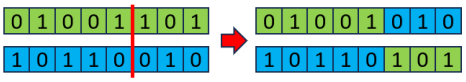

Fig. 4. Example of one-point crossover.

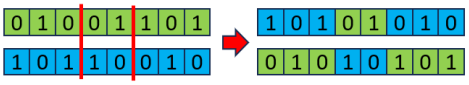

Fig. 5. Example of two-point crossover.

In uniform crossover, a randomly generated crossover mask is used to decide from which parent the offspring gets its genes. If there is a value 1 or 0 in the mask, the gene is copied from the first parent or second parent, respectively (Fig. 6). This procedure is repeated for the second offspring (Townsend, 2003).

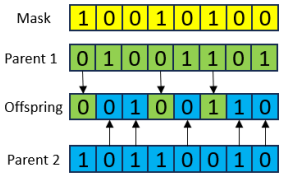

Fig. 6. Example of uniform crossover.

In permutation problems, standard crossover operators are not appropriate since each gene should be represented once and only once in the chromosome. One possible solution is to use a partially matched crossover (PMX), Under PMX, two random crossover points are chosen, and the genes are exchanged between these two points. The exchanged genes remain intact during the rest of the procedure. On the final step, the doubles (marked as yellow color in Fig. 7) are exchanged between the children to get correct permutations (Haupt and Haupt, 2004). Each child chromosome contains ordering information partially determined by each of its parents.

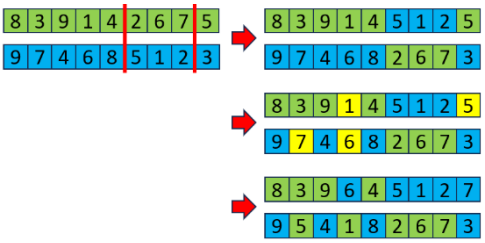

Fig. 7. Example of partially matched crossover (PMX).

Townsend (2003) gives a following summary of crossover methods: there is no more than 20 % difference in speed among the one-point, two-point, and uniform crossover. Uniform crossover or two-point crossover works better if the population is small or large, respectively, compared to the problem complexity.

2.2.7 Mutation

The mutation operator changes the value of one or more genes at randomly selected position in the chromosome. Mutation can take place at each position in the chromosome with some pre-defined probability, known as mutation rate, which is usually small. The mutation operator makes the GA to find a near optimal solution to a given problem more easily by maintaining the genetic diversity in the population (Townsend, 2003).

The most common mutation operators are binary mutation, random resetting, swap mutation, scramble mutation, and inversion mutation.

In binary mutation, the value of the one or more genes is altered with a probability equal to the mutation rate (Larranaga, 1999). For example, the value of the third gene is changed from 1 to 0 in Fig. 8.

Fig. 8. Example of binary mutation.

In swap mutation (Fig. 9), the values of two randomly selected genes are interchanged. The swap mutation is commonly used in permutation-based representations (Larranaga, 1999).

Fig. 9. Example of swap mutation.

In scramble mutation (Fig. 10), a subset of the genes on the chromosome are selected and scrambled randomly (Larranaga, 1999).

Fig. 10. Example of scramble mutation.

In inversion mutation, two random points are chosen along the length of the chromosome. The genes between these points are then inverted as shown in Figure 11 (Goldberg, 1989, p. 166).

Fig. 11. Example of inversion mutation.

2.2.8 Elitism

Elitism means that the most fit chromosomes of the current generation are preserved for the next generation. Elitism prevents the population from losing its best solution due to crossover or mutation. The unwanted side effect is that there might be a super fit chromosome that causes the GA to converge prematurely (Townsend, 2003).

The elite size means the number of fit chromosomes preserved for the next generation.

2.2.9 Termination Condition

The termination condition of a genetic algorithm defines when to stop running the algorithm. Usually, the GA run is terminated when one of the following conditions is met: there is no improvement in the population for given number of consecutive generations, the maximum number of generations

is reached, or the objective function has reached a certain pre-defined value.

2.3 Visual Components

Visual Components is a 3D simulation and offline programming software that can be used for layout planning, feasibility analysis, virtual commissioning, and robot programming. The software has an extensive library of 3D models with 3000+ pre-defined and ready-to-use components including robots, conveyors, machines, resources, robot tools, factory facilities, and more. The user can also import self-made 3D models into the software (Visual Components, 2024).

Python programming language (version 2.7) can be utilized in components scripts. It is possible to add 3rd party Python package to the Visual Components, but this depends on the package. Alternatively, the user can communicate with an external application by using the TCP/IP sockets.

3. EXPERIMENTS AND RESULTS

3.1 Flexible Manufacturing Cell

There is a flexible manufacturing cell in the SeAMK's laboratory of the Machine and Production Technology. The cell consists of two Fanuc R2000iB/165F industrial robots, Kitamura HX500i machining center, workpiece positioner, and the storage system made by Fastems. The storage system has following components: high bay rack for storing Euro pallets and machining pallets, stacker crane for moving the pallets between storage shelves and workstations, material station for inserting and retrieving pallets in and out of the storage, loading station for moving the machining pallets between the stacker crane and robot, and two pallet banks next to the robots for holding the Euro pallets.

A simulation model (Figs. 12 and 13) of the FMS cell was made during one of the research and development projects. The robots, workpiece positioner, grill fences, storage selves, and Euro pallets are from the component library of the Visual Components. The stacker crane, pallet banks, loading station, material station, machining center, and machining pallet were designed in a 3D CAD software and imported into the Visual Components as STL files.

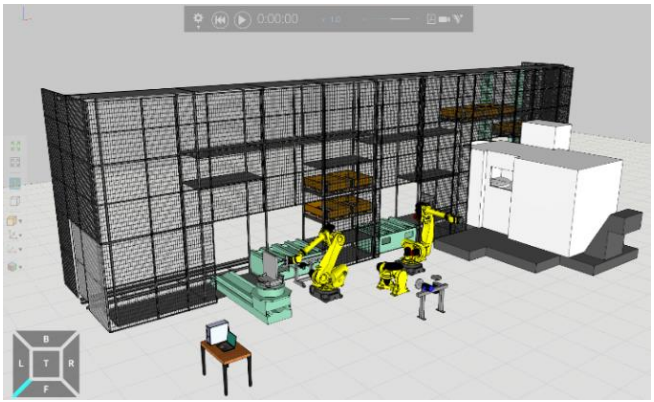

Fig. 12. Front view of the simulation model.

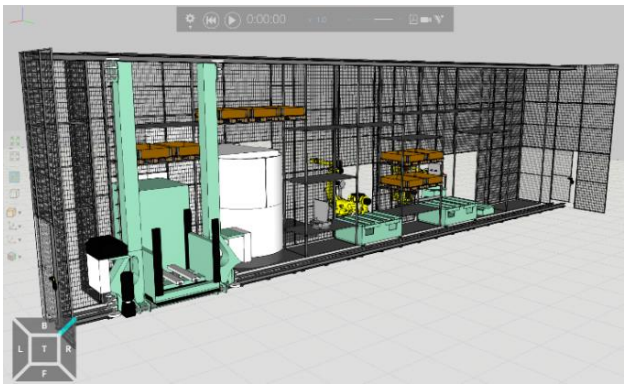

Fig. 13. Back view of the simulation model.

The simulation model of the FMS cell works as its real-world counterpart. The model does not, however, work together with the Fastems Manufacturing Management Software (MMS) that is used to control the real FMS cell. Instead, the simulation model relies on the component scripts to function properly.

3.2 Goal

The goal of the experiment was to utilize the genetic algorithm in the simulation model of the flexible manufacturing cell to minimize the time it takes to reshuffle the warehouse. In other words, the goal was to find the best permutation of Euro pallet transfers so that the stacker crane could move the pallets in the least amount of time. This kind of storage reshuffle process is common in the warehouses.

As the number of transfers increases, the number of different combinations of transfers also increases. For example, in the case of just nine transfers, the number of different combinations is 9! = 362 880.

Although the genetic algorithm can explore multiple solution candidates in parallel, it is not possible to compute the duration of every possible combination of transfers in a reasonable time frame. With the genetic algorithm, however, there is no need to go through every possible combination of transfers.

3.3 Pallet Transfers

The storage reshuffle process consists of nine transfers of Euro pallets. The transfers are randomly generated to eliminate the human bias for choosing transfers that one knows will benefit strongly from the optimization.
In the simulation model, the transfer of the Euro pallet is represented by a dictionary which contains the pallet ID, stock keeping unit (SKU) of the pallet, and source and destination shelf positions. For example, {"pid": 1, "sku": "epallet", "src": 3, "dst": 20}, represents the transfer of the Euro pallet with id = 1 from a shelf position 3 to a shelf position 20. In the GA, these pallet transfers are represented as integers from 1 to 9, so when the fitness values are computed, one needs to decode the genes of the chromosomes into dictionaries.

In this experiment, the fitness function computes the total duration of the nine pallet transfers. Because the goal is to find the minimum duration, a shorter duration yields to smaller fitness value.

To speed up the computation of the fitness function, the transfer time from any given shelf position to any other shelf position was measured programmatically by computing the time difference between the moment when the pallet was picked up from the shelf and the moment when the pallet was placed onto the shelf. The transfer times were then hard coded into the component script of the stacker crane.

The process of measuring the transfer times was quite a tedious task. Alternatively, one could have made the simulation model so that the fitness value of the chromosome is computed by simulation. Based on some testing, the computation of fitness values by simulation took at least 30 minutes so it was decided to use hard coded transfer times instead.

The Euro pallet components are created to the source positions of the pallet transfers when the user starts the simulation.

3.4 Utilizing the GA in the Simulation Model

The Python code for the genetic algorithm was written into the PythonScript object of the stacker crane component in the simulation model. The genetic algorithm was implemented as a Python class whose input arguments are the direction of the optimization as an integer (0 for minimum and 1 for maximum), genes as a list of pallet IDs from 1 to 9, size of the population as integer, number of generations as integer, fitness function as a callable function, size of elite parents in the population as integer, and the mutation rate as a float.
The minimum direction was used since the goal was to find the order of the transfers in which the transfers are performed as quickly as possible, i.e. in minimum time. Based on some testing the population size was set to 1000 since with larger sizes the simulation ran significantly longer times without any major reduction in total transfer times of the pallets. The number of generations was set to 40 and the elite size to 10. According to Townsend (2003, p. 43) the probability of mutation is set to be inversely proportional to the size of the chromosome. Since each chromosome has 9 genes, the mutation rate was set to 11 % (1/9 = 0.111). The parents of the next generation were selected using a tournament selection of 3 chromosomes. Because the goal was to find the best permutation of pallet transfers, PMX was used for the crossover operator. Finally, the swap mutation was used for the mutation operator. Below is a list of the values of the operators and input arguments of the GA.

- Direction: minimum
- Genes: [1, 2, 3, 4, 5, 6, 7, 8, 9]
- Population size: 1000
- Number of generations: 40
- Elite size: 10
- Mutation rate: 11 %
- Selection: tournament of 3 chromosomes
- Crossover: partially matched crossover (PMX)
- Mutation: swap.

3.5 Results

The simulation was run 50 times with and without the genetic algorithm to see how much the genetic algorithm reduces the total transfer time.

When the order of the transfers was optimized with the genetic algorithm, the total duration of the transfers was reduced by circa 20.5 seconds on average. This represents about an 8.1 % reduction in transfer time. The time reduction achieved with the GA in each run is shown in Fig. 14.

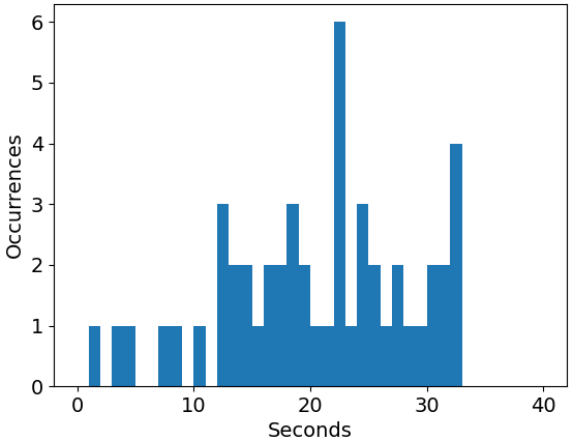

Fig. 14. Histogram of the time reduction achieved with the GA.

It was noticed that the fitness value did not improve that much between the first and last generation of the genetic algorithm. Figure 15 shows a typical progress of the fitness value during the generations of the GA.

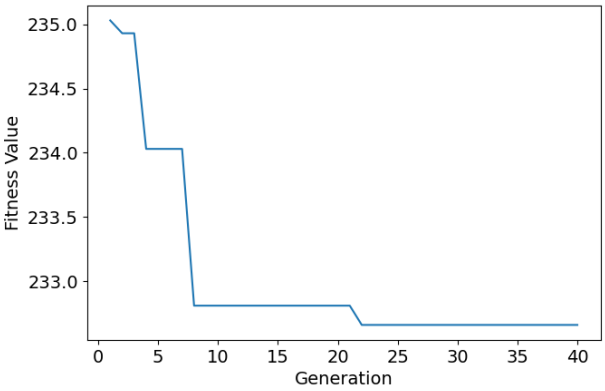

Fig. 15. The progress of the fitness value.

In many cases, the fitness value, i.e. the duration of the transfers, improved mere few seconds. The different combinations of input parameters of the genetic algorithm were tested but the results remained the same.

One possible explanation is the small population size which covers about 0.002 % of all possible solutions (362 880). Increasing the population size would, however, make the simulation to run significantly longer. Thus, the population size of 1000 was chosen.

Other optimization methods than genetic algorithm were not tested in this experiment so it is hard to say if some of them could perform better than GA.

4. CONCLUSIONS

Manufacturing systems are complex environments with many interacting parts and variables in dynamic processess. These complexities often lead to challenges that are difficult to solve with traditional methods, especially as the demand for efficiency grows. To address these issues, combining optimization techniques with 3D simulation has proven to be highly effective. This approach allows us to explore a wide range of potential solutions, helping to find the best ways to improve system performance. By integrating these methods, we can make manufacturing processes more efficient and better equipped to handle the complexities of modern production.

This paper showed how to use an optimization method together with a simulation tool to find a solution that saves time and resources in production. The total time saving of 20 seconds (8.1 %) that was achieved in the demonstration is a significant improvement to the process. The automated warehouse that was utilized as the testbed is a practical and familiar example of a suitable scale for SMEs and thus makes the demonstration more illustrative and easier to catch. The 3D environment is a descriptive surrounding that facilitates showing the process in practice and makes the simulation more realistic. In addition, the high-bay rack and the stacker crane increase complexity because they involve using vertical space and dynamic movement, aspects usually not considered in basic, two-dimensional warehouse models.

The demonstration has been presented in one regional technology event and several smaller workshops for selected SMEs from the manufacturing industry. The reception has been good, and a closer collaboration on the same theme has started with one of the companies. Thus, the methods presented in this paper will be applied in other warehouses in real manufacturing surroundings shortly. Furthermore, the longer-term aim is to promote the integration possibilities of optimization methods and simulation technology from the product level to the development of production-level solutions. This supports the green transition as unnecessary work can be eliminated and processes can be streamlined.

ACKNOWLEDGEMENTS

This paper was written as a part of the project OPLITE (A80151), and the funding from the Regional Council of South Ostrobothnia is greatly appreciated.

REFERENCES

Avdekins, A. and Savrasovs, M. (2019). Making warehouse logistics smart by effective placement strategy based on genetic algorithms. Transport and Telecommunication Journal 20(4):318-324. doi:10.2478/ttj-2019-0026.

Banks, J. (2005). Discrete-event system simulation. 4th ed. Upper Saddle River, N.J: Pearson Prentice Hall.

Bartholdi, J.J. and Hackman, S.T. (2019). Warehouse and Distribution Science, Georgia Institute of Tehcnology. https://www.warehouse-science.com/book/index.html (accessed on 22 May 2024).

Bojic, S., Maslaric, M., Mircetic, D., Nikolicic, S., and Todorovic, V. (2023). Simulation and Genetic Algorithm-based approach for multi-objective optimization of production planning: A case study in industry. Advances in Production Engineering & Management, 18(2), 250-262. doi: 10.14743/apem2023.2.471.

Choi, B. K. and Kang, D. (2013). Modeling and simulation of discrete-event systems. Hoboken, N.J.: John Wiley & Sons Inc.

Ernst, P., Zimmermann, K., and Fieg, G. (2017). Multi-Objective Optimization-Tool for the Universal Application in Chemical Process Design. Chemical Engineering & Technology, 40(10), 1867–1875. doi: 10.1002/ceat.201600734.

Goldberg, D. and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms, in Foundations of Genetic Algorithms, pp. 69-93, Morgan Kaufmann, San Francisco, Calif.

Goldberg, D. (1989). Genetic Algorithms in Search, Optimization & Machine Learning. Addison Wesley.

Grznár, P., Krajčovič, M., Gola, A., Dulina, Ľ., Furmannová, B., Mozol, Š., Plinta, D., Burganová, N., Danilczuk, W., and Svitek, R. (2021). The Use of a Genetic Algorithm for Sorting Warehouse Optimisation. Processes, 9(7), 1197. doi: 10.3390/pr9071197

Haupt, R. L. and Haupt, S. E. Practical Genetic Algorithms, Second Edition. John Wiley & Sons, Inc. (2004).

Holland, J. Adaptation in Natural and Artificial Systems. MI: University of Michigan Press. (1975).

Howard, D. A., Jørgensen, B. N., and Ma, Z. (2023). Multi-Method Simulation and Multi-Objective Optimization for Energy-Flexibility-Potential Assessment of Food-Production Process Cooling. Energies, 16(3), 1514. doi: 10.3390/en16031514.

Kordos, M., Boryczko, J., Blachnik, M., and Golak, S. (2020). Optimization of Warehouse Operations with Genetic Algorithms. Applied Sciences, 10(14), 4817. doi: 10.3390/app10144817

Larranaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., and Dizdarevic, S. (1999). Genetic algorithms for the travelling salesman problem: A review of representations and operators. Artificial intelligence review, 13, 129-170.

Shannon, R. E. Introduction to the art and science of simulation. 1998 Winter Simulation Conference. Proceedings (Cat. No.98CH36274), Washington, DC, USA, (1998), pp. 7-14 vol.1, doi: 10.1109/WSC.1998.744892.

Townsend, A. A. R. (2003). Genetic Algorithm–A Tutorial. Av.: www-course. cs. york. ac. uk/evo/SupportingDocs/TutorialGAs. pdf, 8.

Visual Components. (2024). Visual Components - 3D manufacturing simulation software. https://www.visualcomponents.com/

Xie, X., Zheng, Y., Li, Y. (2015). Genetic Algorithm and Its Performance Analysis for Scheduling a Single Crane. doi: 10.1155/2015/618436