ACTA TECHNICA JAURINENSIS
Vol. 15, No. 3, pp. 174-187, 2022
10.14513/actatechjaur.00668

Research Article

Simulation and Genetic Algorithms to Improve the Performance of an Automated Manufacturing Line

Patrick Ruane[1,2,*], Patrick Walsh[2], John Cosgrove[2]

[1] Johnson & Johnson Vision Care
Rivers, V94 N732 Limerick, Ireland
[2] Technological University of the Shannon
Moylish, V94 EC5T, Limerick, Ireland
*e-mail: patrick.ruane@tus.ie

Submitted: 27/06/2022 Accepted: 29/08/2022 Published online: 31/08/2022

Abstract: Simulation in manufacturing is often applied in situations where conducting experiments on a real system is very difficult often because of cost or the time to carry out the experiment is too long. Optimization is the organized search for such designs and operating modes to find the best available solution from a set of feasible solutions. It determines the set of actions or elements that must be implemented to achieve an optimized manufacturing line. As a result of being able to concurrently simulate and optimize equipment processes, the understanding of how the actual production system will perform under varying conditions is achieved. The author has adopted an open-source simulation tool (JaamSim) to develop a digital model of an automated tray loader manufacturing system in the Johnson & Johnson Vision Care (JJVC) manufacturing facility. This paper demonstrates how a digital model developed using JaamSim was integrated with an author developed genetic algorithm optimization system and how both tools can be used for the optimization and development of an automated manufacturing line in the medical devices industry.

Keywords: Digital Model; Digitalization; Genetic Algorithm; JaamSim; Optimization; Simulation

I. INTRODUCTION

Digitalization in manufacturing is the conversion of information into digital format, the integration of this digital data and technologies into the manufacturing process and the use of those technologies (eg: simulation, optimization) to change a business model to provide new revenue and value-producing opportunities. Digitalization may be seen as the increased generation, analysis, and use of data to improve the efficiency of the overall manufacturing system. Digital manufacturing technologies, such as simulation models, have been considered an essential part of the continuous effort towards improving the performance of automated manufacturing equipment and processes. Optimization seeks the maximum or minimum value of an objective function corresponding to variables defined in a feasible range or space. More generally, optimization is the search of the set of variables that produces the best values of one or more objective functions while complying with multiple constraints. The purpose of optimization has been described as objective function, loss function, or cost function for minimization and utility function or fitness function for maximization [1] [2]. In this paper, it will be referred to as objective function. Simulation optimization (SO) refers to the optimization of an objective function subject to constraints, both of which can be evaluated through a stochastic simulation/digital model [3]. The term simulation optimization (SO) is an overall term for techniques used to optimize stochastic simulations. Simulation optimization involves the search for those specific settings of the input parameters to a stochastic simulation such that a target objective, which is a function of the simulation output, is either maximized or minimized [3]. Simulation techniques allow for modelling and artificially reproducing complex systems using stochastic distributions [4]. Complex simulation models may require long development times and difficult verification and validation processes and finally, simulation is not an optimization tool on its own [5]. According to [5] large Combinatorial Optimization Problems (COPs) require the use of metaheuristics to conduct an efficient search, where he proposes to combine simulation with metaheuristics to form a new class of optimization algorithms called ‘simheuristics’. These algorithms integrate simulation (in any of its

variants) into a metaheuristic-driven framework to solve complex stochastic COPs. A metaheuristic is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms [6]. The JaamSim simulation package used in this industrial case study currently has no optimization analysis capability [7]. It is thus proposed by the author to develop and integrate a metaheuristic genetic algorithm optimization engine with the JaamSim Tray Loader digital model thus enabling the optimization of this industrial case system.

An optimization problem involves searching for an optimal solution(s) x_i from a search space X, which maximize (or minimize) an objective function f(x), while satisfying a set of constraints [8]. The search space X may be composed of discrete variables (e.g., integer, categorical), continuous variables or mixed variables [9]. Metaheuristics are general algorithmic frameworks, often nature-inspired, designed to solve complex optimization problems [10]. Metaheuristics are a growing research area over the last number of years. Metaheuristics are emerging as successful alternatives to more classical approaches also for solving optimization problems that include in their mathematical formulation uncertain, stochastic, and dynamic information [10]. The Greek suffix ‘‘meta’’ used in the word metaheuristic means ‘‘beyond, in an upper level’’. Thus, metaheuristics are algorithms that combine heuristics (that are usually very problem-specific) in a more general framework. Metaheuristics are strategies that guide the search process. The goal is to efficiently explore the search space to find near–optimal solutions. Techniques which constitute metaheuristic algorithms range from simple local search procedures to complex learning processes [11]. Optimization algorithms attempt to improve solutions in each iteration, seeking to converge toward the optimal solution. After a number of iterations, the search reaches an optimal region of the feasible decision space. The best solution calculated by the algorithm at the time of termination constitutes the optimal solutions of a particular run. **Fig. 1** portrays the process of optimization by Metaheuristic and evolutionary genetic algorithms.

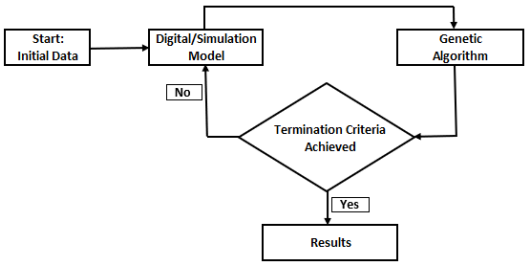


Figure 1. *Components of the Optimization System using Simulation and Genetic Algorithms*

II. GENETIC ALGORITHMS

1. Genetic Algorithm Overview

Among the meta-heuristic optimization methods, genetic algorithms have gained importance because of its capacity to find sets of optimal solutions [12]. A genetic algorithm (GA) is an 'intelligent' probabilistic search algorithm which simulates the process of evolution by taking a population of solutions and applying genetic operators in each reproduction [13]. Genetic Algorithms (GAs) are adaptive heuristic search algorithms based on the evolutionary ideas of natural selection and genetics. They are a part of evolutionary computing, a rapidly growing area of artificial intelligence. GAs are inspired by Darwin’s theory of evolution – “Survival of the fittest”. Simplicity of operation and power of effect are two of the main attractions of the GA approach [14]. Genetic algorithms are popular as they are relatively easy to implement and are used in several commercial software packages [3]. Genetic algorithms (GA) have been used for the resolution of a wide variety of combinatorial problems, due to the demonstrated success in the results it can achieve [15]. Despite the advantages of genetic algorithms, several parameter inputs are required before using this algorithm. They include waypoint, population size, crossover rate, and mutation rate. The potential GA solution to a problem is an individual which can be represented by the set of parameters. These parameters are just like a gene of a chromosome and can be represented by the string of values in binary form [16]. The fitness value is used to test the degree of goodness of the chromosome for solving a problem that is directly related to the objective value. The operators employed in a GA include selection, crossover, and mutation processes [16] [17].

The performance of the Genetic Algorithm is dependent on these parameter settings [18]. The GA method requires the algorithm to be initialized with a set of randomly generated initial values, which is known as initial population which represents a significant difference with respect to mathematical programming techniques. The initial population is then evaluated to determine which of the individuals have the best characteristics (i.e., the best values for the objective functions), allowing them to pass to the next generation (or iteration). There is a similarity between GA and those that can be observed with the natural evolution concepts. Once the population has been evaluated, the best individuals combine their genetic information between them, and a new generation is obtained. Standard GAs begins with a randomly generated population of possible solutions (individuals). The individual’s fitness is calculated

and some of them are selected as parents according to their fitness values. A new population (or generation) of possible solutions (the children's population) is produced by applying the crossover operator to the parent population and then applying the mutation operator to their offspring. The fitness value is recalculated for this new population. The iterations involving the replacement of the original generation (old individual) with a new generation (children) is repeated until the termination criteria is achieved. This whole process is shown in **Fig. 2**.

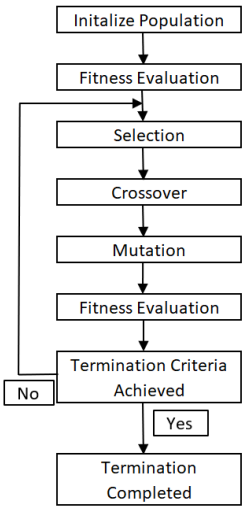


Figure 2. Genetic Algorithm Flowchart [19]

2. Elitism Strategy

A solution with a high fitness value could be replaced by a weaker solution after a crossover or mutation occurs. The process of maintaining good solutions with high fitness after a certain generation cannot be guaranteed. Hence an elitism strategy can be applied in GA to maintain a certain number of the fittest solutions for the next generation. When the next-generation population is obtained after crossover and mutation, these solutions that were maintained by elitism will replace the weaker solutions. The same number of the fittest solutions will replace the weaker solutions and be retained and utilized for the next generation [19] [20]. It has been shown that results obtained by an algorithm which uses elitism is better than the result obtained by an algorithm which doesn't use elitism [21], [22].

3. GA Parameters and Termination Strategy

The size of the population of solutions (M), the number of parents (R), the probability of crossover (PC), the probability of mutation (PM), and the termination criterion are the user defined parameters of the GA. A good choice of the parameters is related to the decision space of a particular problem, and in general the optimal parameter setting for one problem may not perform equally as well for other problems. Consequently, determining a good parameter setting often requires the execution of many time-consuming experiments. A critical factor in implementing a genetic algorithm is how to set the values for the various parameters. [23] classifies these efforts into two major forms:

1. Parameter tuning. It refers to finding good values for the parameters before the algorithm is run and then keeping these values fixed while the algorithm runs. With this method, typically one parameter is tuned at a time, which may cause some suboptimal choices, since parameters often interact in a complex way with each other. Simultaneous tuning of more parameters, however, leads to an enormous number of experiments.
2. Parameter Control. This method forms an alternative, as it amounts to starting a run with initial parameter values which are then changed during the run.

Selecting the appropriate GA parameters is regularly done based on experience with specific optimization problems. However, a reasonable method for finding suitable values for the GA parameters is to perform sensitivity analysis. This entails choosing a combination of GA parameters and running the GA several times. Other combinations of parameters are chosen, and repeated runs are made with each combination. A comparison of the optimization results obtained may lead to the best set of GA parameters. The author has used Design of Experiments to select the optimum GA parameters for the Tray Loader application.

A termination criterion is required to allow the Genetic Algorithm to end its iterations. Selecting an appropriate termination criterion has an important role on the correct convergence of the algorithm. The number of iterations, the amount of improvement of the objective function between consecutive iterations, and the run time are common termination criteria for the GA.

4. NSGA-II (Non-dominated Sorting Genetic Algorithm II)

The non-dominated sorting algorithm (NSGA), developed in 1994, was one of the first Multi Objective Evolutionary Algorithms (MOEA) [24]. NSGA differs from the standard GA in the way that the selection operator performs, with the crossover and mutation operators remaining the same. The population of solutions is ranked based on its nondomination before selection takes place. Improvements to NSGA were made to tackle issues such as high computational complexity, lack of elitism, need to specify sharing parameter and a technique was added to embed constraints into the optimization algorithm, leading to a new algorithm known as NSGA-II being introduced [25]. According to [26] [27] one of the most widely used

MOEA's that has been effective in finding the Pareto optimal solutions is the elitist NSGA-II algorithm. Both the diversity and the convergence abilities of the NSGA-II algorithm have been demonstrated by [28]. They have also shown the suitability of NSGA-II in producing an acceptable number of optimized design alternatives regarding the problem complexity and in a reasonable timeframe. A detailed review of NSGA-II optimization algorithm in machining operations was presented by [29]. They concluded that NSGA-II as part of Multi Objective Optimization Problem (MOOP) is a popular and reliable algorithm that can be used in optimizing the process parameters of multiple machine performances. Unlike the single objective optimization technique, NSGA-II simultaneously optimizes each objective without being dominated by any other solution [29]. The problem of controlling an air conditioning system using evolutionary algorithms to increase energy-saving while also considering user satisfaction was investigated [30]. They concluded that the NSGA II as an excellent algorithm for solving a multi objective optimization problem. It has also been shown that the multi-objective optimization technique NSGA-II applied to a project was efficient in searching for multiple solutions and was able to find a pareto front after a few iterations during the optimization process [31]. NSGA-II applies an elitist strategy which improves the convergence of an MOEA and avoids the loss of optimal solutions after getting them [32]. It is proposed to use the Elitist NSGA-II and develop a standalone multi objective optimization engine that will run fully integrated with the JaamSim Tray Loader digital model. The workings of the NSGA-II will now be further explained. The flowchart for NSGA-II is shown in **Fig. 3**.

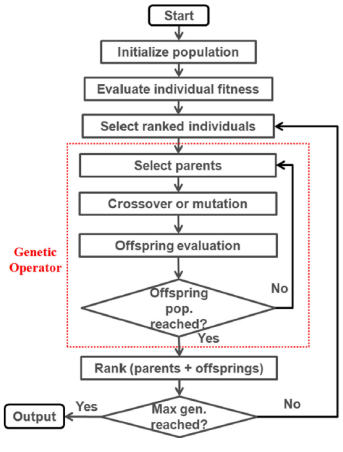
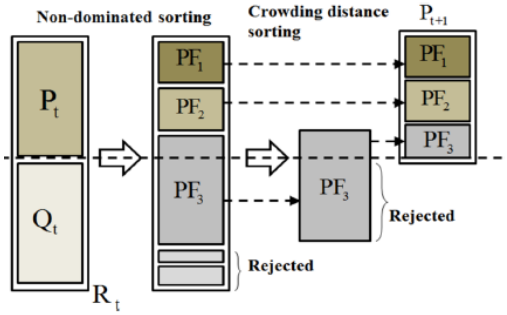


Figure 3. *NSGA-II Algorithm Flowchart* [31]

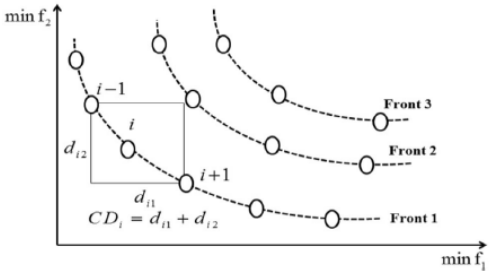
In NSGA-II parents and offspring are combined, followed by non-dominated sorting. The fitness of all individuals is assessed and chosen to be parents for the next generation. The NSGA-II Non-dominated sorting and crowding distance sorting, which is depicted in Fig 4 is then completed. P_t is the parent generation and Q_t the offspring that are both merged into R_t. The objective is to obtain a new generation P_{t+1} of the same size as the parent population P_t. Two parameters are estimated for each individual: the domination count, which provides the information of how many solutions dominate the individual, and a list of the set of



solutions that are dominated by the individual. This method splits up all solutions into different fronts. As per **Fig. 4**, $PF_{1\text{-}3}$ are the fronts that are obtained by the sorting process.

Figure 4. *NGSA-II Ranking Procedure [25]*

All individuals are compared with each other. The first front will comprise only solutions with a domination count of 0. From there, the algorithm continues going individual by individual through all sets of solutions that have a domination count of 0 to form the first front. The individuals from this 1st front are removed from the list, and the remaining individuals now compared to each other with the 2nd front obtained by selecting individuals with a new domination count of 0. After this process, all the individuals that have a



domination count of zero, excluding the first front solutions, will form the second front. The procedure is continued until the last front is obtained as can be seen in **Fig. 5**.

Figure 5. *Solution Pareto fronts and Crowding Distance Estimation*

From Fig 5 all solutions in PF1 and PF2 are taken forward to the new population P_{t+1}. Some solutions from PF_3 are taken forward to P_{t+1}, while the remainder is rejected. The solutions that are taken forward from PF_3 is based on the crowding distance calculation, with the lesser crowded distance individual being chosen to form the total in population P_{t+1} [25]. The crowding distance is a number that determines how closely other solutions are surrounding an individual. Figure 5 shows the calculation of the crowding distance of solution i.

The crowding distance is an estimate of the size of the largest cuboid enclosing solution i without including any other solution [30]. The nearest neighbours are used to calculate the average distance between the closest solutions of the same front. A higher value of crowding distance gives a lesser crowded region and vice versa [25].

III. Development of Tray Loader Digital Model

5. Overview of The Tray Loader Digital Model

A digital model of an industrial system (**Fig. 6**) known as a Tray Loading System was developed using JaamSim software.

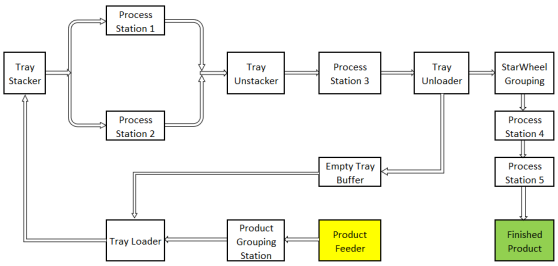


Figure 6. *Automated Tray Loading System Industrial Case*

This system consists of individual product (p) that arrives from an upstream line to a product feeder at defined arrival times. These are then grouped into multiples of 10. The group of products are then loaded into empty plastic trays that can hold up to 660 parts. Once filled the plastic tray moves at a defined cycle time to a tray stacker. The tray stacker accumulates the filled trays into groups of 30. This group of 30 trays then undergoes a batch process in either Process station 1 or 2 under defined conditions. Upon completion of this batch process, the trays of product leave Process Station 1 or 2, where a tray unstacking operation takes place. Each individual tray of product undergoes a further process step (Process Station 3), again under defined conditions. Once a tray is finished at Process Station 3, the product is removed from the tray at the Tray Unloading station and is then passed to the Star Wheel grouping station, where the product is now grouped into batches of 30. These groups are then passed to Process Station 4 and 5 for the final finishing process. The empty trays from the tray unloading station, are returned to the empty tray buffer and finally back to the tray loader operation, to repeat the overall process. The digital model developed, will simulate this whole operation, considering the following 5 points:

1. Entities (units of Product) per arrival.
2. Service times for process stations, travel times for conveyors
3. Probability distributions for reliability and repair of stations.
4. Conditions for process stations to process and pass product to the next station.
5. Queue size and location.

6. Verification of the Tray Loader Digital Model

A detailed verification process was undertaken on the Tray Loader digital model following the Logical/mathematical verification, program/code verification steps outlined by [33] and the detailed knowledge of the author of the actual tray loading system. All the Tray Loader Objects, Service Times, Steps, Thresholds, Maintenance conditions and Threshold condition logic were all verified and confirmed to be correct to how the actual line operates. A detailed verification checklist was completed on the Tray Loader digital model. As part of the digital model verification process it was important to verify that the product flow into and out of the various simulation objects (as seen from the JaamSim GUI) are identical to what occurs on the tray loader line. This verification process allowed any additions or changes to the simulation logic to be corrected, verified, and visualized immediately. It was through the ongoing and iterative model verification and the testing process during model development, that a realistic model of the actual dynamic interactions was developed and fine-tuned. During this phase of model verification, the weak points of the system were discovered and corrected. It is extremely advantageous to find these early-stage simulation bugs, thus allowing a well-tested and robust system to be developed.

7. Validation of the Tray Loader Digital Model

The approach taken for developing the Tray Loader digital model followed the steps described by [34]. Step 5 of this approach deals with confirming that the programmed model is valid. The model is run using the standard basic settings from the actual tray loader system. The simulation model output data for the system was compared with the comparable output data collected from the actual system. This is called results validation. If the results are consistent with how the system should operate, then the simulation model is said to have face validity. Sensitivity analyses is performed on the programmed model to see which factors have the greatest impact on the performance measures and, thus, must be modelled carefully [34]. According to [35], validation is concerned with determining whether the conceptual digital model (as opposed to the computer program) is an accurate representation of the system under study. [35] outlines the

following three (3) steps to validate a simulation model.

1. Obtaining real-world data from the actual system.
2. Tests for comparing simulated and real data (namely graphical, Schruben-Turing or t tests).
3. Sensitivity analysis (using statistical design of experiments with associated regression analysis).

The above approach was used to validate the Tray Loader digital model, see section 3 for more detail. Actual Tray Loader system data was collected from the historian database for all the relevant process stations used in the digital model. The data collected included input feed rate, yield, throughput and uptime per minute for each process station. Excel macros were then developed to calculate the equipment reliability metrics namely: Mean Time Between Failures (MTBF) and Mean Time to Repair (MTTR) for each of the process stations using the uptime/minute data. The Input feed rate, yield, output data and the MTBF/MTTR for each process station was analysed, outliers removed, and distributions determined along with the distribution parameters. Minitab is used to analyse all the data obtained. Minitab is a statistical analysis software that assists in the analysis of data collected from any process and provides a simple, effective way to input the data, manipulate that data and statistically analyse it.

IV. Development of The NSGA-II Optimization Engine

A closed loop digital model and optimization engine is proposed by the author as shown in **Fig. 7**. An NSGA-II optimization engine is integrated with the Tray Loader JaamSim digital model and the following four (4) elements being executed automatically until an optimized solution is obtained:

1. Digital Model inputs parameters updated.
2. Simulation runs executed and monitored
3. Digital Model outputs collected.
4. Optimization analysis completed and new parameter settings recommended.

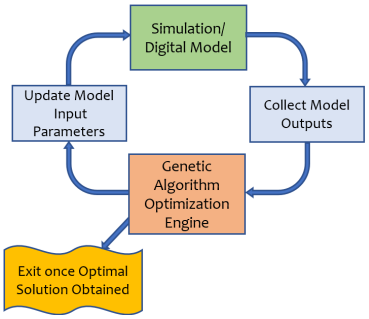


Figure 7. *Closed Loop Optimization Engine*

The overall optimization system developed by the author allows the user to specify the objectives to be optimized from an excel file. **Table 1** shows an example of two (2) objectives to be minimized along with two (2) objectives to be maximized (station throughputs). This file is used to configure the optimization problem along with the associated objectives to be either maximised or minimised.

Processes	Objective
Max Trays Used	Minimize
P_Feeder	Maximize
Process4	Maximize
P_Feeder Util	Minimize

Table 1. *Optimization Objectives*

Entity_Name	Parameter	Value	Optim_Space
P_Feeder	InterArrivalTime	0.90 s	-10 to 10
P_Feeder	Downtime Duration (MTTR)	2.00 Min	-10 to 10
Tray_Pack	Downtime Duration (MTTR)	2.00 Min	-10 to 10
Process4	ServiceTime	2.3 s	-10 to 10
Process5	ServiceTime	2.3 s	-10 to 10
P_Feeder_Yield	Weibull Location	0.5573	-10 to 10
Empty_Tray_Stacker	Capacity	90	-15 to 15

Another excel file is set-up to store all the entities/workstations names along with their associated base parameter values, see **Table 2** for a sample of some configuration settings for the Tray Loader Simulation model.

Table 2. *Simulation Model and Optimization Parameters*

The settings in the Optim_Space column are used by the NSGA-II optimization engine. As an example, referring to **Table 2**, JaamSim is configured with an entity generator called P_Feeder. The base InterArrivalTime for this generator is 0.90sec. When performing an optimization analysis, the InterArrivalTime for this entity can be changed within a space of 0.90 sec ±10% in increments of 1%. Likewise, the Tray Loader JaamSim model uses a resource called Empty_Tray_Stacker, with a base setting of 90 units. The optimization space for this parameter is 90 ± 15% in increments of 1%. The user can select which parameters, the base setting for that parameter and if required the optimization space for that parameter to be used by NSGA-II.

Reviewing the optimization space for the 7 factors in Table 2, there is in excess of 2.6 Billion combinations of different factor settings that the Tray Loader line can be operated to. It is impossible to run all of those combinations using the Tray Loader digital model, hence the need to use optimization approaches to determine a particular setting for each of the 7 factors that results in an optimum solution to the required objective(s).

Python code was developed that integrates the excel input configuration files with both the JaamSim Tray Loader digital model and the NSGA-II optimization engine. The overall structure and Python code that was written to integrate the NSGA-II and the JaamSim Tray Loader digital model to form an optimization system followed a modular format. This modular format followed the ten (10) rules and two (2) best practices for code development highlighted by [36]. The overall system architecture is shown in **Fig. 8**. This architecture gives a high-level overview of how the optimization system was developed with the main optimization system being controlled by the module called *invoke_Simw* (see purple box in **Fig. 8**). The main function module called *invoke_simw* then calls other blocks (red boxes in **Fig. 8**) forming the main spine of the Tray Loader Optimization system. All the function modules are written using the python programming language. The four (4) main blocks of the system include:

1. Main controlling function module called *invoke_simw*
2. Input Data Pre-processing function block that calls several sub function modules.
3. Overall GA and JaamSim Optimization Loop function block calling several sub function modules.
4. Output data file post processing block calling several sub function modules.

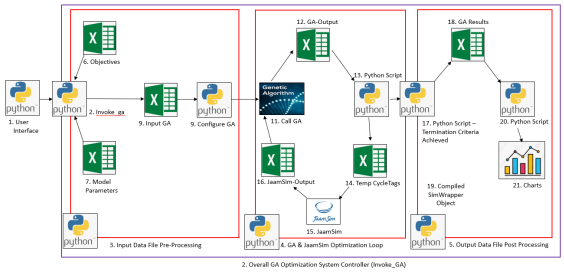


Figure 8. *OPTIM-GA Program and Data Structure*

Tray Loader digital model parameters are passed to the 'invoke_simw' function. The invoke_simw function (**Fig. 8**), then schedules the calling of all the various functions and methods required to execute all the tasks in the three (3) red boxes. When all input data pre-processing is completed, the NSGA-II and JaamSim Optimization loop (**Fig. 8**) is activated where simulation runs are completed using the tray loader digital model. Output results from each simulation run is then analyzed by NSGA-II optimization engine and any associated changes to the digital model input parameters based on the requirements of the objective function are then made. This process is repeated until the termination criteria is achieved thus producing an optimal solution. The tray loader termination criteria is reviewed in section 8 below.

Once NSGA-II optimization has terminated the program returns to the calling function '*invoke_simw*'. At this point the function 'Output Data File Post Processing' **Fig. 8** is called. This block of code prepares the results from the optimization study for review and graphing. The data is also saved to a csv file to allow the user to further analyze the data with statistical packages (eg: Minitab ©) to support any decisions in relation to possible design changes to the tray loading system. A significant number of Python libraries associated with optimization have been developed recently, however, only a few of them support optimization of multiple objectives at a time [37]. As such, pymoo (python multi-objective optimization) which is a library of multi-objective optimization tools was developed in Python [37]. There are several different algorithm implementations in "pymoo" examples include GA and NSGA-II to name a few. These NSGA-II pymoo library of optimization routines were used in the development of the overall Tray Loader NSGA-II optimization system. A plug-in library called pymoo, Ver 0.5.0 was then installed into the Thonny IDE to enable multi objective optimization in Python (**Fig. 9**).

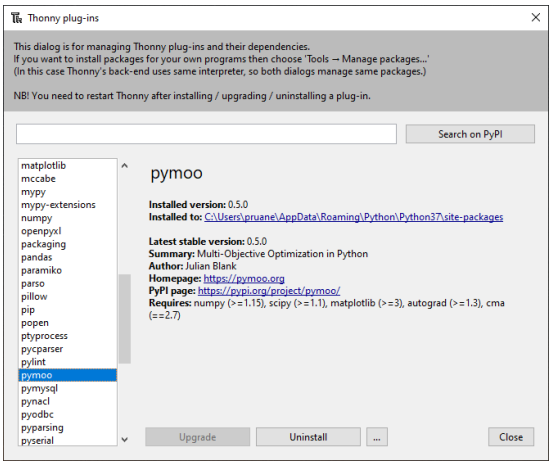


Figure 9. *Pymoo Library in the Thonny IDE*

A list of the additional Python libraries installed into the Thonny IDE are given in **Table 3**. These libraries are required to allow the developed python code for the Tray Loader optimization to run without errors.

Table 3. *Python Libraries installed into the Thonny IDE*

Number	Library Name	Installed Version	Summary Description
1	beautifulsoup4	4.10.0	Screen-scraping library
2	bitstring	3.1.7	Simple construction, analysis and modification of binary data.
3	bs4	0.0.1	Dummy package for Beautiful Soup
4	KDE-diffusion	1.0.3	Kernel density estimation via diffusion in 1d and 2d
5	lxml	4.6.4	Powerful and Pythonic XML processing library combining libxml2/libxslt with the ElementTree API.
6	matplotlib	3.3.3	Python plotting package
7	numpy	1.19.5	NumPy is the fundamental package for array computing with Python.
8	openpyxl	3.0.9	A Python library to read/write Excel 2010 xlsx/xlsm files
9	pandas	1.2.0	Powerful data structures for data analysis, time series, and statistics
10	pymoo	0.5.0	Multi-Objective Optimization in Python
11	thonny	3.3.13	Python IDE for beginners
12	scipy	1.7.1	SciPy: Scientific Library for Python
13	seaborn	0.11.2	seaborn: statistical data visualization

8. NSGA-II Termination Criteria for the Tray Loader Optimization Problem

Whenever an optimization algorithm is executed, it needs to be determined at each iteration whether the optimization run shall be continued or not. Many different ways exist of how to decide when an optimization run should be terminated. Running the algorithm not long enough can lead to unsatisfactory results and running it too long might waste function evaluations, time and thus computational resources. Pymoo have developed several termination criterion for both single and multi-objective optimization. The Tray Loader termination criteria uses the standard 'Termination' function which was imported by python from pymoo. Actual code is given below:

```
from pymoo.core.termination import Termination
```

According to [38] the most interesting stopping criterion is to use objective space change to decide whether to terminate the algorithm. This termination criteria uses a simple and efficient procedure to determine whether to stop the optimization or not. This termination procedure is called 'MultiObjectiveSpaceToleranceTermination', and is imported from pymoo as given by the actual code below:

```
from pymoo.util.termination.f_tol import MultiObjectiveSpaceToleranceTermination
```

This termination procedure 'MultiObjectiveSpaceToleranceTermination' is then configured with various termination parameters and assigned to the 'termination' attribute with python code as given below:

```
# NSGA-II Tray Loader termination criteria

termination = MultiObjectiveSpaceToleranceTermination(tol=0.0025, n_last= min(30, n_max_gen), nth_gen= min(5, patience), n_max_gen= n_max_gen, n_max_evals=None)
```

The five (5) termination parameters [38] above are described as follows:

1. tol = This is the average threshold tolerance in the objective space. If the value is below this bound (0.25% from above), the algorithm is terminated.
2. n_last = To make the termination criterion more robust, this parameter specifies the last *n* generations to review and then takes the maximum from this number of generations.
3. nth_gen = Defines whenever the termination criterion is calculated by default, or every nth generation. In the example above, nth_gen is the minimum of 5 or the patience value.
4. n_max_gen = Furthermore, the number of generations executed by the algorithm can be used for termination. For some optimization problems, the termination criterion might not be reached, thus, an upper bound for generations can be defined to stop in this case.
5. n_max_evals = Lastly, the number of function evaluations can be used for termination. In the example above, this is not used as can be seen when this variable is set to `None`.

9. NSGA-II parameter tuning for the Tray Loader Application

The key to a successful implementation of Genetic Algorithms primarily depends on the efficient crossover and mutation search operators to guide the system toward a global optimum [39]. The values of GA parameters greatly determine whether the GA will find a near-optimum solution and whether it will find such a solution efficiently in a timely manner. Choosing the right parameter values can be a time-consuming task where the computer specifications can play a significant factor in how long it takes to obtain both the GA optimum parameters and determining the optimum solution to the problem itself [23]. According to [40], GAs are not easy to use because they require parameter tunings in order to achieve the desirable solutions. The task of tuning GA parameters has been proven to be far from trivial

due to the complex interactions among the parameters. In the research carried out by [41], parameter setting for MOOP using evolutionary algorithms (MOEAs) is crucial for finding the best performance of the algorithm. These parameters are very sensitive in driving the algorithms to the best performance and finding the good results. Design of experiments (DoE) methods offer practical approaches to tune the parameters effectively [42]. It has been shown that the internal parameters of NSGA-II can be tuned using the Design of Experiments (DoE) procedure to enhance the quality of the results for the synthesis optimization of a four-bar mechanism [43]. The six (6) operating parameters of the NSGA-II algorithm which need to be set for the Tray Loader optimization application are as follows:

1. Population size.
2. # of Offspring.
3. Crossover Probability
4. Crossover Distribution Index
5. Mutation Probability
6. Mutation Distribution Index

These parameters affect the capability of the algorithm to achieve the optimum objective results and computing time to reach these results. According to [44] population size can be decided by experience and usually between 50 and 160. If the population size is too small, then it can be difficult to get an optimum solution, whereas, if it's too large then the convergence time can be long. A recommended range of parameter settings is given in **Table 4** to achieve optimum GA performance [41], [42], [44].

Table 4. *GA Parameter Settings*

GA Parameter Name	Identifier	Range
Population Size	P	30 - 100
Crossover Probability	P_C	0.2 - 0.8
Mutation Probability	P_M	0.001 - 0.1

The mutation distribution index (η_m) and the crossover distribution index (η_c) are typically set in the range of 10 – 40 [45]. A large crossover distribution index (η_c) gives a higher probability for creating near parent solutions and a small crossover distribution index (η_c) allows distant solutions to be selected as children solutions [46]. The parameters with the associated levels for each parameter that are used in the NSGA-II algorithm are given in **Table 5**.

Table 5. *NSGA-II Parameters and Levels*

Factor #	Factor Name	Symbol	# of Levels	High Setting	Centre Setting	Low Setting
1	Population Size	P	2	100	65	30
2	# of Offspring	P_O	2	0.80	0.55	0.30
3	Crossover Probability	P_C	2	0.80	0.55	0.30
4	Crossover Distribution Index	η_c	2	40	25	10
5	Mutation Probability	P_M	2	0.10	0.05	0.01
6	Mutation Distribution Index	η_m	2	40	25	10

A ½ fractional DoE was chosen for tuning the Tray Loader NSGA-II optimization parameters as the resolution provided was sufficient to analyse the data. A total of 33 runs is required for the experiment (32 ½ fraction runs and 1 centre level run). Each experiment was run for a maximum of 30 generations based on previous optimization experiments carried out by the author during the development of this optimization system. Increasing the number of generations, significantly increases the time required to run each experiment. The max *P_Feeder* output and max *Process4* output was recorded across the total population for each of the thirty (30) generations. Analysis of Variance (ANOVA) and response optimization of the NSGA-II parameters is completed using Minitab in order to maximise both the *P_Feeder* and *Process4* outputs. The results are shown in **Table 6**.

Table 6. *Tray Loader NSGA-II Parameter Optimization*

Solution

Solution	Population Size	No of Offspring	Crossover Probability	Crossover Distribution Index	Mutation Probability	Mutation Distribution Index
1	1	1	-1	-1	-1	1

Solution	Process 4 Result Fit	P_Feeder Result Fit	Composite Desirability
1	441228	461766	0.870449

Based on this analysis, the Tray Loader NSGA-II optimization system is configured with the parameter values as shown in **Table 7**.

Table 7. *Tray Loader NSGA-II Parameter Values*

Parameter #	ParameterName	Symbol	Setting Level	Setting Value
1	Population Size	P	1	100
2	# of Offspring	P_O	1	0.80
3	Crossover Probability	P_C	-1	0.30
4	Crossover Distribution Index	η_c	-1	10
5	Mutation Probability	P_M	-1	0.01
6	Mutation Distribution Index	η_m	1	40

V. Results from the Tray Loader Digital Model and NSGA-II Optimization Engine

All simulation/optimization runs were completed using a HP ZBook Firefly 15 G7 2Z4F7UC laptop running an Intel(R) Core(TM) i7-10810U CPU @ 1.61 GHz processor and 64GB of RAM. The single objective optimization run (Maximize P_Feeder output) was executed 10 times as recommended by [47] [48]. The results of the 10-run experiment is given in Table 8.

Table 8. *Simulation Model and Optimization Parameters*

Run Number	Total # of Generations	# of Generations to Best Sol'n	Total Execution Time (Mins)	Time to Optimal Sol'n (Mins)	P_Feeder Mean	P_Feeder Max	P_Feeder Min	P_Feeder STD
1	50	21	481	202	459,575	462,391	457,550	1,322
2	50	50	470	470	459,345	461,870	458,117	1,076
3	50	40	544	435	458,955	462,741	457,305	1,367
4	50	50	460	460	454,725	461,899	424,117	1,182
5	25	8	262	84	457,000	461,870	455,286	1,539
6	45	26	473	273	458,702	462,685	456,726	1,432
7	30	14	278	130	457,656	462,741	455,715	1,519
8	50	35	518	363	458,204	461,654	456,585	1,230
9	45	26	442	255	459,158	462,391	457,623	1,050
10	25	9	236	85	457,868	462,741	455,828	1,593
Average of All Runs	42	28	416	276	458,119	462,298	453,485	1,331

As can be seen from **Table 8**, the *P_Feeder* Mean, Max, Min and Standard Deviation is calculated across the 50 generations for each run using the NSGA-II optimization. The average *P_Feeder* (max) across the 10 runs using NSGA-II was 462,298 units. The overall *P_Feeder* maximum output across the 10 runs using NSGA-II optimization was 462,741 achieved on runs 3, 7 and 10. Run #1 was analysed in additional detail, as the results of this particular run produced results that were close to the overall average of the 10 runs completed. Analyzing the data collected from Run #1, the *P_Feeder* maximum, minimum, average and standard deviation is calculated for each of the 50 generations and plotted using Minitab©. **Fig. 10** shows how all the individual solutions within the population of 100 solutions for each of the 50 generations are converging closer to the *P_Feeder* maximum value of 462,391 which was achieved on generation #22.

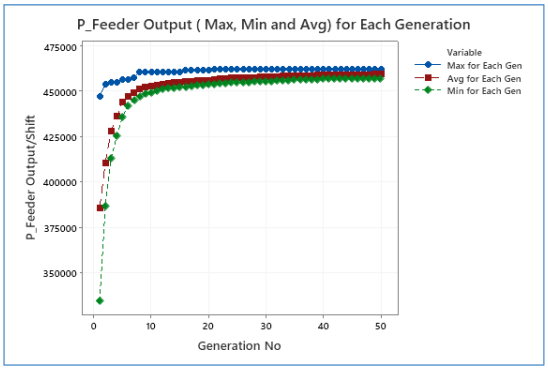


Figure 10. *NSGA-II Optimization of P_Feeder Output/Shift*

The maximum *P_Feeder* output remained unchanged for the remaining 28 generations of the experiment. The standard deviation of the *P_Feeder* output within the population of 100 solutions for each generation is plotted and can be seen in **Fig. 11**.

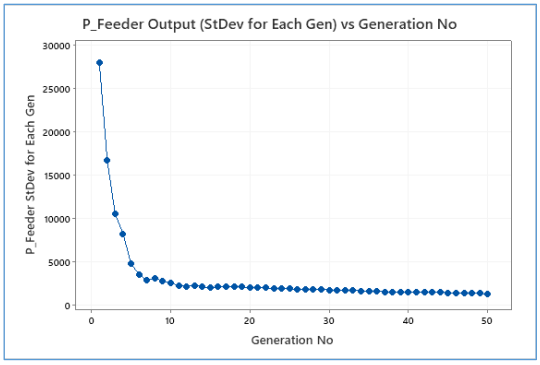


Figure 11. *NSGA-II optimization of P_Feeder Standard Deviation*

Fig. 11 shows that as the solutions are generated for each generation, the spread is reducing indicating that all of the solutions are progressively getting closer to the optimum *P_Feeder* max value of 462,391 and the optimization procedure can be terminated. It can be seen from **Fig. 10** and **11** that the NSGA-II algorithm has converged after approximately 22 generations, at which point the fitness value function (max *P_Feeder* Output) was unchanged and the standard deviation of P_Feeder output of all the solutions within each generation decreasing slightly. To reduce the optimization computation time, the maximum number of generations could be reduced from 50 to approx. 30. This value of 30 was selected (greater than 22), with the aim of avoiding an early termination of the algorithm before the max *P_Feeder* output was obtained. The solution developed for run #7 (**Table 8**) with an overall *P_Feeder* max of 462,741 units using the Tray Loader JaamSim digital model and the NSGA-II optimization engine is shown in **Table 9**.

Table 9. *Tray Loader Optimized Digital Model Parameters*

			Optimized Tray Loader Digital Model Parameters						
Digital Model Used	Optimization Engine Type	P_Feeder (Max)	P_Feeder_Inter Arrival Time	P_Feeder_Yield (Location)	P_Feeder_Down Time_ Duration	Process4_Service Time (Sec)	Tray_Pack_Down Time_ Duration	Process5_Service Time (Sec)	Empty_Tray_Stacker_ Capacity Count
Tray Loader JaamSim	NSGA-II	462,741	81	0.613	2.04	2.41	2.18	2.50	94

A two (2) objective optimization problem (maximize the P_Feeder and minimize the Empty Tray Buffer capacity) was designed and tested using the tray loader digital model and NSGA-II optimization engine. As with the single objective optimization problem, the same Tray Loader simulation model, model parameters and optimization parameter space was used for this study (See Table 2), with results given in **Table 10**.

Table 10. *Two Objective Optimization problem of Tray Loader System using NSGA-II Optimization*

	NSGA-II Optimization (Population size = 100, Analysis of last Generation)							
Run Number	Total # of Generations	# of Generations to Best Sol'n	Total Execution Time (Mins)	Time to Optimal Sol'n (Mins)	P_Feeder Min	P_Feeder Max	Avg_Trays_Used (Min)	Avg_Trays_Used (Max)
1	50	31	453	281	304,943	461,707	61	84
2	40	22	355	195	311,928	461,592	63	84
3	30	11	432	158	308,997	461,650	63	84
4	50	34	434	295	308,967	461,917	62	81
5	50	17	438	149	303,533	462,391	61	84
6	35	18	301	155	316,285	460,309	64	85
7	45	28	435	271	315,424	462,741	63	81
8	35	13	350	130	305,053	460,750	63	95
9	50	38	464	353	312,300	461,273	63	84
10	35	11	355	112	305,756	460,683	62	84
Average of All Runs	42	22	402	210	309,319	461,501	63	85

A pareto front is a set of nondominated solutions, being chosen as optimal, if no objective can be improved without sacrificing at least one other objective [8]. The pareto front is an excellent visualization to show the interaction of each objective has on the other. A pareto front (*Empty Tray Buffer* Capacity vs *P_Feeder* Output/Shift) was generated using all the data gathered from the 2 objective SimWrapper Optimization runs. See **Fig. 12** for the pareto front.

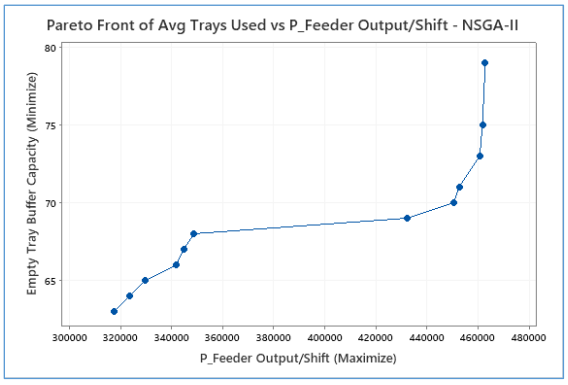


Figure 12. *Tray Loader 2 Objective Optimization Pareto Front.*

As can be seen from **Fig. 12**, the optimum solution is where the Empty Tray count (Buffer Capacity) is approx. 79 trays, thus producing a stable *P_Feeder* output of ~ 462,741. Increasing the Empty Tray buffer beyond 79 trays, has no impact on the *P_Feeder* output/shift. Since the optimization problem is to minimize Empty Tray Buffer and maximize *P_Feeder* output, the factor setting providing the solution of 79 trays and *P_Feeder* output of 462,741 is selected.

VI. Conclusion

As manufacturing capital equipment is expensive, it is necessary that the equipment once in operation is reliable and delivers to the business plan targets. Simulation along with an optimization system is an invaluable tool to confirm that an automated manufacturing line can produce to the required business objectives before and after it goes into operation. Implementing the actual changes to equipment to improve reliability can be both time consuming and expensive. Simulation in conjunction with optimization can be used to verify these improvements before the equipment is modified. These technologies form the basis of an overall digital manufacturing system that enables the optimization of a manufacturing line during the line design stage or when the line is put into operation. The use of this technology gives a deeper understanding of what can occur on the manufacturing line when it is running. A simulation model when combined with optimization engine, can be used to identify problems before they occur and aid in the selection of optimum parameters to run the line before it is fully designed or built. Digital model and optimization technologies supports other Industry 4.0 technologies such as predictive maintenance, OEE improvement, waste reduction, improve batch changeover times and to improve product quality [49]. It allows for efficient design and development, linking 3D models with simulation and emulation of equipment control code. In addition, having a digital model enables virtual line analysis, removing the physical restraints of expert engineers having to be on your location [50]. The author has demonstrated how the development of digital model can be validated and subsequently used as part of an optimization system which is then used for the study of equipment design, maintenance and reliability of an automated production line in the medical devices industry.

Acknowledgement

The authors express their sincere gratitude to Johnson & Johnson Vision Care and Technological University of the Shannon for providing me with the tools, time and support necessary to allow me to pursue this very important research in the field of simulation and optimization of advanced automated manufacturing equipment.

Author Contributions

Patrick Ruane: Conceptualization, development, Experimentation, Analysis, Writing and Editing.

Patrick Walsh: Supervision & Review.

John Cosgrove: Supervision & Review.

Disclosure Statement

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

ORCID

Patrick Ruane http://orcid.org/0000-0002-6685-1663

REFERENCES

[1] W. Erwin Diewert, "The New Palgrave Dictionary of Economics," Palgrave Macmillan UK, 27 April 2017. [Online]. https://link.springer.com/referenceworkentry/10.1057/978-1-349-95121-5_659-2.

[2] S. Boyd and L. Vandenberghe, Convex Optimization, vol. 7, Cambridge University Press, 2009.

[3] S. Amaran, N. Sahinidis, B. Sharda and S. Bury, "Simulation Optimization: A Review of Algorithms and Applications," *Annals of Operations Research,* vol. 240, pp. 351-380, 2016. https://doi.org/10.1007/s10479-015-2019-x

[4] R. Nance and R. Sargent, "Perspectives on the Evolution of Simulation," *Operations Research,* vol. 50, no. 1, pp. 161-172, 2002. https://doi.org/10.1287/opre.50.1.161.17790

[5] A. Juan, J. Faulin, S. Grasman, M. Rabe and G. Figueira, "A review of Simheuristics: Extending Metaheuristics to deal with Stochastic Combinatorial Optimization Problems," *Operations Research Perspectives,* vol. 2, pp. 62-72, 2015. https://doi.org/10.1016/j.orp.2015.03.001

[6] K. Sörensen and F. Glover, "Metaheuristics," In: Gass, S.I., Fu, M.C. (eds) Encyclopedia of Operations Research and Management Science. Springer, Boston, MA (2013) pp. 960-970. https://doi.org/10.1007/978-1-4419-1153-7_1167

[7] King, D.H, and H.S Harrison. 2013. "Open Source Simulation Software "JaamSim"." Proceedings of the 2013 Winter Simulation Conference. Washington, DC, USA. https://doi.org/10.1109/WSC.2013.6721593

[8] N. Gunantara and Q. Ai, "A review of multi-objective optimization: Methods and its applications.," *Cogent Engineering,* vol. 5, no. 1, pp. 1 - 16, 2018. https://doi.org/10.1080/23311916.2018.1502242

[9] J. Pelamatti, L. Brevault, M. Balesdent, E. Talbi and Y. Guerin, "Efficient global optimization of constrained mixed variable problems.," *Journal of Global Optimization,* vol. 73, no. 3, pp. 583-613, 2019. https://doi.org/10.1007/s10898-018-0715-1

[10] L. Bianchi, M. Dorigo and L. Gambardella, "A Survey on Metaheuristics for Stochastic Combinatorial Optimization," 2009. https://doi.org/10.1007/s11047-008-9098-4

[11] C. Blum and A. Roli, "Metaheuristics in Combinatorial Optimization Overview and Conceptual Comparison," *ACM Computing Surveys,* vol. 35, no. 3, pp. 268-308, 2003. https://doi.org/10.1145/937503.937505

[12] F. Castro, C. Gutierrez-Antonio, A. Briones-Ramirez and J. Herandez, "Genetic Algorithms: A tool for Optimizing Intensified Distillation Sequences," *Genetic Algorithms: Advances in Research and Applications,* pp. 1-17, 2017.

[13] P. Chu and J. Beasley, "A Genetic Algorithm for the Generalised Assignment Problem," *Computers & Operations Research,* vol. 24, no. 1, pp. 17-23, 1997. https://doi.org/10.1016/S0305-0548(96)00032-9

[14] D. Goldberg and J. Holland, "Genetic Algorithms and Machine Learning," *Machine Learning,* vol. 3, no. 2-3, pp. 95-99, 1988. https://doi.org/10.1023/A:1022602019183

[15] M. Dalle Mura and G. Dini, "A Multi-Objective Software Tool for Manual Assembly Line Balancing using a Genetic Algorithm," *CIRP Journal of Manufacturing Science and Technology,* vol. 19, pp. 72-83, 2017. http://dx.doi.org/10.1016/j.cirpj.2017.06.002

[16] R. Haupt and S. Haupt, Practical Genetic Algorithms, New York, NY, USA: John Wiley & Sons, 2004. https://doi.org/10.1002/0471671746

[17] M. Hamza, H. Yap and I. A. Choudhury, "Genetic Algorithm and Particle Swarm Optimization Based Cascade Interval Type 2 Fuzzy PD Controller for Rotary Inverted Pendulum System.," *Mathematical Problems in Engineering,* vol. 2015, pp. 1 - 15, 2015. https://doi.org/10.1155/2015/695965

[18] P. Rajendran and K. Yit Yok, "The Optimization of a Genetic Algorithm for Unmanned Aerial Vehicle Path Planning," *Genetic Algorithms: Advances in Research and Applications,* pp. 19 - 32, 2017.

[19] K. Kok, P. Rajendran, R. Rainis, I. Wan and M. Wan Mohd, "Investigation on selection

schemes and population sizes for genetic algorithm in unmanned aerial vehicle path planning.," *International Symposium on Technology Management & Emerging Technologies (ISTMET),* pp. 6 - 10, 2015. https://doi.org/10.1109/ISTMET.2015.7358990

[20] K. Yit Kok, P. Rajendran, R. R and W. Mohd Muhiyuddin Wan Ibrahim, "Investigation on Selection Schemes and Population Sizes for Genetic Algorithm in Unmanned Aerial Vehicle Path Planning," in *2015 International Symposium on Technology Management and Emerging Technologies (ISTMET)*, Langkawi, Malaysia, 2015. https://doi.org/10.1109/ISTMET.2015.7358990

[21] C. Grosan and M. Oltean, "The Role of Elitism in Multiobjective Optimization with Evolutionary Algorithms," *Acta Universitatis Apulensis* 5 (2003) pp. 83-90.

[22] G. Guariso and M. Sangiorgio, "Improving the Performance of Multiobjective Genetic Algorithms: An Elitism-Based Approach," *Information,* 11 (12) pp. 1 - 14, 2020. https://doi.org/10.3390/info11120587

[23] A. Eiben, R. Hinterding and Z. Michalewicz, "Parameter Control in Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation,* vol. 3, no. 2, pp. 124-141, 1999. https://doi.org/10.1109/4235.771166

[24] N. Srinivas and K. Deb, "Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms," *Evolutionary Computation,* p. 22 1– 248, 1994. https://doi.org/10.1162/evco.1994.2.3.221

[25] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation,* vol. 6, no. 2, pp. 182 - 197, 07 August 2002. https://doi.org/10.1109/4235.996017

[26] T. Goel, R. Vaidyanathan, R. Haftka, W. Shyy, N. Queipo and K. Tucker, "Response Surface Approximation of Pareto Optimal Front in Multi-Objective Optimization," *Computer Methods in Applied Mechanics and Engineering,* vol. 196, no. 4, pp. 879 - 893, 2007. https://doi.org/10.1016/j.cma.2006.07.010

[27] C. Leung and H. Lau, "Simulation-based optimization for material handling systems in manufacturing and distribution industries," *Wireless Networks,* p. 4839 – 4860, 2020. https://doi.org/10.1007/s11276-018-1894-x

[28] N. Mahammed, M. B. S, O. A and M. Fahsi, "Evolutionary Business Process Optimization using a Multiple-Criteria Decision Analysis method," in *International Conference on Computer, Information and Telecommunication Systems (CITS)*, 2018. https://doi.org/10.1109/CITS.2018.8440166

[29] Y. Yusoff, M. Salihin Ngadiman and A. Mohd Zain, "Overview of NSGA-II for Optimizing Machining Process Parameters," *Procedia Engineering 15,* p. 3978 – 3983, 2011. https://doi.org/10.1016/j.proeng.2011.08.745

30] A. Ruiz, S. Martínez, J. Rocha, J. Villanueva, J. Menchaca, M. Berrones, M. Flores and A. Pineda, "Assessing a Multi-Objective Genetic Algorithm with a Simulated Environment for Energy-Saving of Air Conditioning Systems with User Preferences," *Symmetry 2021,* vol. 13, no. 2, 20 Febuary 2021. https://doi.org/10.3390/sym13020344

[31] Y. Chang, Z. Bouzarkouna and D. Devegowda, "Multi Objective Optimization for Rapid and Robust Optimal Oil Field Development Under Geological Uncertainty," *Computational Geosciences,* vol. 19, p. 933 – 950, 2015. https://doi.org/10.1007/s10596-015-9507-6

[32] H. Kumar and S. Yadav, "Hybrid NSGA-II Based Decision Making in Fuzzy Multi Objective Reliability Optimization Problem," *SN Applied Sciences,* 2019. https://doi.org/10.1007/s42452-019-1512-2

[33] P. K. Davis, "Generalizing Concepts and Methods of Verification, Validation and Accreditation (VV&A) for Military Simulations," *National Defense Research Institute,* pp. 5-6, 1992.

[34] A. M. Law, "How to Build Valid and Credible Simulation Models," 2019 Winter Simulation Conference (WSC), 2019, pp. 1402-1414. https://doi.org/10.1109/WSC40007.2019.9004789

[35] J. P. Kleijnen, "Theory and Methodology of Verification and validation of simulation models," *European Journal of Operational Research,* vol. 82, pp. 145-162, 1995.

[36] H. Hunter-Zinck, A. de Siqueira, V. Vásquez, R. Barnes and C. Martinez, "Ten Simple Rules on Writing Clean and Reliable Open-Source Scientific Software," *PLOS Computational*

Biology, pp. 1 - 9, 2021. https://doi.org/10.1371/journal.pcbi.1009481

[37] J. Blank and K. Deb, "Pymoo - Multi-objective Optimization in Python," *IEEE Access,* vol. 8, pp. 89497 - 89509, 2020. https://doi.org/10.1109/ACCESS.2020.2990567

[38] Blank, J, and K Deb. 2020. "A Running Performance Metric and Termination Criterion for Evaluating Multi and Many Objective Optimization Algorithms." 2020 IEEE Congress on Evolutionary Computation (CEC) pp. 1 - 9. https://doi.org/10.1109/CEC48606.2020.9185546

[39] Lim, S.M, A.B Sultan, N Sulaiman, A Mustapha, and K.Y Leong. 2017. "Crossover and Mutation Operators of Genetic Algorithms." *International Journal of Machine Learning and Computing* 7 (1). https://doi.org/10.18178/ijmlc.2017.7.1.611

40] Duc Tran, Khoa. 2005. "Elitist non-dominated sorting GA-II (NSGA-II) as a parameter-less multi-objective genetic algorithm." Proceedings. IEEE Southeast Conference. Ft. Lauderdale, FL, USA. pp. 359 - 367. https://doi.org/10.1109/SECON.2005.1423273

[41] Samsuri, S, R Ahmad, M Zakaria, and M Zain. 2019. "Parameter Tuning for Comparing Multi-Objective Evolutionary Algorithms Applied to System Identification Problems." Proc. of the 2019 IEEE 6th International Conference on Smart Instrumentation, Measurement and Applications. Kuala Lumpur, Malaysia. https://doi.org/10.1109/ICSIMA47653.2019.9057333

[42] Arin, A, G Rabadi, and R Unal. 2011. "Comparative studies on design of experiments for tuning parameters in a genetic algorithm for a scheduling problem." International Journal of Experimental Design and Process Optimisation 102-124. https://doi.org/10.1504/IJEDPO.2011.040262

[43] Badduri, J, R.A Srivatsan, Kumar G.S, and S Bandyopadhyay. 2012. "Coupler-Curve Synthesis of a Planar Four-Bar Mechanism Using NSGA-II." Asia-Pacific Conference on Simulated Evolution and Learning. 460-469. https://doi.org/10.1007/978-3-642-34859-4_46

[44] Cao, Z, and Z Zhang. 2010. "Parameter Settings of Genetic Algorithm Based on Multi-Factor Analysis of Variance." 2010 Fourth International Conference on Genetic and Evolutionary Computing. Shenzhen, China. 305 - 307. https://doi.org/10.1109/ICGEC.2010.82

[45] Deb, K. 2011. "Multi-Objective Optimization Using Evolutionary Algorithms." Department of Mechanical Engineering, Indian Institute of Technology Kanpur, Kanpur, PIN 208016, India, 1 - 24. Accessed January 10, 2022. https://www.egr.msu.edu/~kdeb/papers/k2011003.pdf

[46] Deb, K, and H Beyer. 2001. "Self Adaptive Genetic Algorithms with Simulated Binary Crossover." Evolutionary Computation 9 (2): 197 - 221. https://doi.org/10.1162/106365601750190406

[47] M. Jeong, J. H. Choi and B. H. Koh, "Performance evaluation of modified genetic and swarm-based optimization algorithms," *Structural Control and Health Monitoring,* p. 878–889, 2013. https://doi.org/10.1002/stc.507

[48] J. Shen and Y. Zhu, "Chance-Constrained Model for Uncertain Job Shop Scheduling Problem," *Soft Computing - A Fusion of Foundations, Methodologies & Applications.,* vol. 20, no. 6, pp. 2383-2391, June 2016. https://doi.org/10.1007/s00500-015-1647-z

[49] G. Shao, S. Jain, C. Laroque, L. H. Lee, P. Lendermann and O. Rose, "Digital Twin for Smart Manufacturing: The Simulation Aspect," 2019 Winter Simulation Conference (WSC), 2019, pp. 2085-2098. https://doi.org/10.1109/WSC40007.2019.9004659

[50] Q. Qi, F. Tao, T. Hu, N. Anwer, A. Liu, Y. Wei and L. Wang, "Enabling Technologies and Tools for Digital Twin," *Journal of Manufacturing Systems,* vol. 58, pp. 3-21, 2021.