IJIO
(International Journal of Industrial Optimization)
ISSN 2723-3022 (online) | 2714-6006 (printed)
Vol. 5, No. 2, 118-133
https://doi.org/10.12928/ijio.v5i2.10027

Development of genetic algorithm for human-robot collaboration assembly line design

Anas Ma'ruf *, Diniarie Budhiarti

Faculty of Industrial Technology, Bandung Institute of Technology, Indonesia
* Corresponding Author: maruf@itb.ac.id

ARTICLE INFO

Article history
Received: February 4, 2024
Revised: March 5, 2024
Accepted: May 8, 2024

Keywords
Assembly line balancing;
Human-Robot Collaboration;
Genetic Algorithm;
Cycle Time.

ABSTRACT

An assembly line requires flexibility due to a shorter product life cycle. A way to increase flexibility is to utilize collaborative robots or cobots. Due to frequent product changes, redesigning an assembly line requires an efficient algorithm. This research aims to develop a genetic algorithm (GA) for solving a human-cobots assembly line design. The setup time of cobots is considered due to the flexibility of conducting multiple tasks by exchanging tools / end-effectors. The main contribution of the research is the efficient GA for solving assembly lines considering setup time. Secondly, the study proposed an upper limit parameter that enables faster computation without sacrificing the quality of the solution. The computational results showed that the algorithm could achieve an optimal solution with the number of tasks less than 35. Experiments of several data prove the proposed GA obtained solutions with an average gap of 3.83% to the optimal solution. Also, a faster computation time with an average difference of 64.66%. The proposed GA obtained a reasonable solution with fast computing time that helps improve efficiency and effectiveness in decision-making related to frequent redesigning of assembly lines.

1. Introduction

The global market development opens access to new markets where companies can experience increased demand. The ability of a company to adapt more effectively to respond to demand is one of the determining factors in business [1]. In the manufacturing industry, effective assembly line balancing can be a solution to accommodate fluctuating demand. Balancing assembly lines is crucial in improving efficiency, speeding up the production system, and reducing production costs for each unit [2]. Assembly line balancing is a form of medium to short-term decision-making and requires significant investment. Therefore, a system must be well-designed to operate more efficiently [3].

Assembly line balancing has shifted from traditional configurations, whether manual or robotic, towards more flexible and productive solutions, transitioning from mass production to mass customization. This research conducted a case study of two large electronic industries in Indonesia. The study reveals that the assembly line is redesigned every 10 to 14 days due to the fulfillment of the targeted production. It is necessary to reconfigure more frequently and promptly to achieve more flexibility in managing resources and machines and improving production efficiency. Digital technology is becoming increasingly intensive, aligning with what is now called Industry 4.0. Smart manufacturing must embrace big data and software that controls production processes and resource planning [4]. One technology that has emerged in this industrial revolution is collaborative robot

(cobot) technology [5][6]. According to Gualtieri et al. [7], considering the physical and safe workspace capabilities during the production process, collaborative robots, as cyber-physical systems, enable the implementation of human-robot collaboration.

Human-robot collaboration aims to achieve better performance in the production system [8]. Solutions in implementing human-robot collaboration (HRC) can enhance the efficiency of the assembly process by leveraging the strengths of both human skills in various aspects combined with the load capacity and repetitive capabilities of robots. Implementing HRC also improves quality performance by reducing product defects, especially for products with complex structures [9]. Additionally, from an ergonomic perspective, it can ensure human workers reduce physical labor by using robots to perform certain activities [10][11].

The cycle time is an issue in the design of assembly lines in operational problems. Cycle time is the maximum time a station on the assembly line can process a product. Within the cycle time, there is idle time, which can be caused by operators being idle after completing a task at a slower pace [12]. Companies aim to identify and eliminate behaviors that do not affect the manufacturing process to minimize cycle time. Thus, minimizing cycle time can help companies deliver products to customers faster and enhance customer satisfaction [13].

Several types of time influencing production cycle time include setup, process, queue, wait, and idle times [14]. Among these, setup time may represent a high percentage in determining cycle time, especially in assembly lines that utilize robots [15]. Setup time is needed to reconfigure the robot in the assembly process. Often, robots must change tools / end-effectors for adjustments to conduct tasks and grasp complex product geometry [16]-[18]. This characteristic makes setup time a significant consideration in this research. Additionally, considering the real needs in assembly line conditions allows for using various tools on the same robot. According to Nugraha et al. [19], each task may involve different types of tools used by robots or HRC, leading to the influence of function assignment to workstations and resources.

Research conducted using HRC includes studies by Mura & Dini [10], Nugraha et al. [19], and Yaphiar et al. [20], aiming to minimize costs. On the other hand, research by Gualtieri et al. [7], Nourmohammadi et al. [8], Weckenborg et al. [21], and Dimeny et al. [22] focuses on minimizing cycle time considering HRC resource constraints. Various methods are employed regarding solution approaches, including analytical models and metaheuristics. The literature review indicates that different aspects and objective functions related to HRC have been studied. Nevertheless, setup time and the types of tools as additional aspects to make the model closer to a realistic condition have yet to be widely considered. Among the studies mentioned, only Nugraha et al. [19] considered using tools in robots or HRC. Previous research that addressed setup time predominantly focused on robotic assembly lines rather than HRC, as seen in studies by Li et al. [23] and Janardhanan et al. [24].

Several research has been conducted applying robots to cope with flexibility. Zhang proposed a mathematical model to design and reconfigure a single-product assembly line [25]. The robot could be set at any workstation, limiting to one task at each station. Basan developed a MILP decomposition procedure for assigning multipurpose units and assembly operations [26]. The model considers redesigning the assembly line by preventing bottlenecks and balancing equipment utilization. Hashemi-Petroodi proposed the MILP model by considering future product variants [27]. The objective function is to minimize the cost of designing and reconfiguring the assembly line. Mao proposed a MILP model to assign human-robot collaboration in an assembly line [28]. The system characteristics are the same as in this research, but setup time for switching between cobot tasks is not considered. The above approach assumes reconfiguration will happen based on predicted demand changes. This research approach assumes HRC as an alternative to induce flexibility in the assembly line design [29].

Ma'ruf et al. [4] conducted HRC research, considering tool types and setup time using an analytical model. Other prominent characteristics of this research include: 1) the assembly line under study is a straight production line with a single-variant product focus, and 2) HRC can only be performed when human and robot resources are available and unassigned. However, the drawback of analytical models is that they require longer computation times to build feasible solutions and are less

effective for solving a small number of tasks compared to genetic algorithms (GA) [8]. Therefore, in improving efficiency and flexibility, an algorithm is needed to handle complex problems and its ability to adapt assembly paths with faster computing times.

According to Nourmohammadi et al. [30], GA exhibits more efficient exploration performance due to their ability to solve problems in an exhaustive search space (population-based). This capability helps prevent solutions from getting stuck in local optima. Additionally, GA has been widely used for problems with various optimization criteria and constraints, providing flexible solutions [31]. For example, in research by Tjandra et al. [32], genetic algorithms could find route combinations for the Multiple Traveling Salesman Problem (MTSP). GA efficiently optimized order sequences for production planning in a study by Harale et al. [33]. Among other population-based algorithms, genetic algorithms are considered effective in solving large-scale problems and directing solutions towards optimality, making them suitable for NP-hard problems. GA has produced better solutions than Simulated Annealing in optimizing cycle time for robotic assembly line balancing problems [24].

The aim and contribution of this research is to extend the model of Ma'ruf et al. [4] by applying GA to effectively achieve a solution for an HRC assembly line having a large number of tasks. The algorithm considers the number of tools and setup time to minimize cycle time in assembly line problems. The second contribution is achieving faster computation time by limiting the solution space to converge the cycle time to the optimal value without sacrificing the quality of the solution.

The following section will discuss the method section that develops the GA model to determine task allocation and resource utilization for minimizing cycle time. The third section will discuss the results and the computational outcomes. The fourth section concludes the research findings and remarks for further development.

2. Methods

2.1. Problem statement

A collaborative robot or cobot is a solution that combines the skills, agility, and cognitive abilities of human operators with the accuracy and repetitive skills of robots in the same workspace, making it increasingly applied in the manufacturing industry [34]. Cobots are suitable for various applications encountered in the manufacturing industry [21]. Furthermore, the advantages of using cobots include being more flexible, easily reprogrammable for new tasks, mobile, and often lower costs than conventional robots [35]. Another advantage of cobots is the safety feature that allows workers to collaborate with the robot, reducing the need for robot trajectory planning or collision checking [36]-[38]. The effectiveness of the assembly line, utilizing the cobot, relies on task assignment and allocation of human workers and robots.

With various applications of cobots in the manufacturing industry, this research focuses on the assembly line balancing problem. Numerous exact and heuristic studies are continuously conducted to provide solutions for the Simple Assembly Line Balancing Problem (SALBP). SALBP is a fundamental optimization problem with a straight assembly line for a single product type, dividing the total workload of the assembly process into a series of tasks that must meet precedence constraints among several stations arranged in a serial production process [21]. The general objectives of SALBP include minimizing assignments, ensuring each assignment is designated to a station, and not violating cycle time constraints [39].

The research problem system uses a genetic algorithm to minimize cycle time in the assembly line. The development of the GA model in this study refers to the model developed by Ma'ruf et al. [4] and Weckenborg [21]. With specifications for a single product type, cycle time minimization is achieved by allocating assignments to a predetermined number of workstations and determining the resources to complete these assignments. The allocation of tasks is determined without violating the precedence diagram. The resources used consist of three types: human, robot, and human-robot collaboration. The selection of resources can choose the types of tools used and setup time if the task is performed by a robot or human-robot collaboration, thus affecting task completion time.

 10.12928/ijio.v5i2.10027

2.2. Genetic Algorithm (GA)

Genetic Algorithm (GA) was developed by John H. Holland at the University of Michigan, starting his research in 1960 and formally introduced in 1975 [40]. The concept of GA is based on Darwin's theory, where GA selects chromosomes with better fitness values from the existing population and eliminates the least fit [41]. The search technique is conducted simultaneously on several possible solutions known as the population. Each individual represents a unique solution, and the population is a set of solutions at each iteration stage. Genetic algorithms work to find high-quality individual structures within the population. The typical stages of GA [32] generally consist of 1) forming the initial population, 2) the individual selection process, 3) crossover, and 4) mutation. The research stages using GA are outlined below and illustrated in Fig. 1.

In the conducted research, GA has chromosomes with three vectors of values. These chromosomes encompass the task type, the resource used for each task, and the workstation as the task's operational location. Regarding resources, one signifies human, two denotes a robot, and three signifies human-robot collaboration. Each individual in the initially generated population, as depicted in the early stage of the diagram, will have a GA chromosome structure visualized in Table 1. These individuals will be utilized to execute the solution improvement process by employing tournament selection, crossover, mutation, and elitism to generate the best individual.

Table 1. Chromosome GA

Task	1	3	2	4	5	7	6	8	9	10
Resource	3	1	2	3	2	1	1	1	1	1
Station	1				2				3	

Stage 1: Initiating the first generation (n = 1) by determining parameter values, including the population size (P), the number of generations (N), crossover probability (P_c), and mutation probability (P_m).

Stage 2: Generating an initial feasible population using a randomly constructed algorithm that does not violate precedence constraints.

Stage 3: Calculate each individual's fitness value in the generated initial population. Based on the research's objective function to minimize cycle time, the fitness value involves minimizing the individual cycle time upper limit (CT_{up}) against the maximum station time. CT_{up} is calculated using the following formula,

$$CTup = Roundup\left(\frac{max\ total\ processing\ time\ for\ an\ individual}{available\ workstation}\right) \quad (1)$$

Stage 4: Tournament selection is performed to choose two parents for the crossover and mutation processes.

Stage 5: Generate a random number between 0 and 1. If this random number is less than or equal to the crossover probability (P_c), the crossover process is conducted on the two parents. The result of the crossover process is stored as crossover offspring, which will be used in the following process: mutation. If the random number exceeds the crossover probability (P_c), both parents are directly employed in the next stage.

Stage 6: Generate a random number between 0 and 1. If this random number is less than or equal to the mutation probability (P_m), the mutation process is carried out on the crossover offspring. The result of the mutation process is stored as mutation offspring, which will be used in the following process: elitism. If the random number exceeds the mutation probability (P_m), a review is conducted to determine whether both individuals have performed crossover.

Stage 7: If both individuals are crossover offspring (have performed the crossover process), then both individuals can be stored and proceed to the elitism stage. If both individuals are parents without the crossover process, the process will continue to the next generation (n = n+1).

Stage 8: The produced offspring will be examined to check for violations of precedence constraints, the number of stations, and the maximum tool count. If the offspring violates these

constraints, the process will continue to the next generation ($n = n + 1$). If it meets the constraints, the process will proceed to the next step.

Stage 9: Add the processed offspring that meet the constraints to the initial population.

Stage 10: Reorganize the station grouping with tasks and resources selected for each population member. Then, recalculate the fitness value for each individual in the population and the offspring generated from the crossover and mutation processes.

Stage 11: Perform elitism selection to determine whether the produced offspring are worthy of being added to the initial population, forming the improvement population.

Stage 12: If the number of generations (n) has not been fulfilled, the process can continue to the next generation. When the number of generations has been fulfilled, the individual with the lowest cycle time will be designated as the best individual after one cycle of the GA.

2.2.1. Initial Population

The initial population stage is utilized to generate an initial feasible population, serving as the foundation for the solution search process in this research. Initial solutions are generated by creating a task sequence by precedence constraints. It is then continued by assigning tasks to each workstation and determining the resource responsible for each task [41]. Subsequently, the initial population will be used for the evolutionary process to discover the best individual solution in the GA. The formation of individuals in the population involves determining the task sequence i that does not violate the precedence constraints. Next, the resources with a process time t are randomly chosen from the tasks in the first sequence and placed into the workstation. Another constraint is the upper bound cycle time for each station. If workstation j violates the constraint, the next workstation will be opened with $j = j + 1$. Upper bound cycle time (CT_{upP}) is calculated using the following formula:

$$CT_{upP} = Max\{Round\ up(\alpha \frac{\sum_i^m Max(t_{i1}, t_{i2}, t_{i3})}{m_j}; Max(t_{is}))\ \forall_i \in I, s = 1,2,3 \quad (2)$$

2.2.2. Tournament Selection

Tournament selection is carried out to choose two parents based on the best fitness value from randomly selected individuals, and they are then used in the crossover and mutation processes [42]. The commonly used method for this purpose is tournament selection, which involves randomly selecting several individuals and comparing the fitness values of each individual [43]. The tournament selection stage is conducted by setting a random number K with a value between 1 and the population size (P). Subsequently, the fitness values of each individual are compared, and the individual with the best fitness value, i.e., the one with the lowest cycle time, is selected and saved as the parents.

2.2.3. Crossover

Crossover is crucial to achieving optimal solutions in GAs [44]. The fundamental crossover concept involves exchanging tasks and resources between formed parents to generate new offspring. This research employs two crossover methods: one-point crossover and partially mapped crossover (PMX). The one-point crossover used in this study is a straightforward method proven to produce offspring that satisfy precedence relationships [21]. A one-point crossover is implemented by randomly selecting a crossover point with a value between 1 and the task sequence index (k) minus one (1 to k-1), as illustrated in Fig. 2. Partially mapped crossover is the most commonly used method in traveling salesman problems, production planning, and manufacturing scheduling [44]. PMX is employed for task crossover by performing segment mapping, as illustrated in Fig. 3.

 10.12928/ijio.v5i2.10027

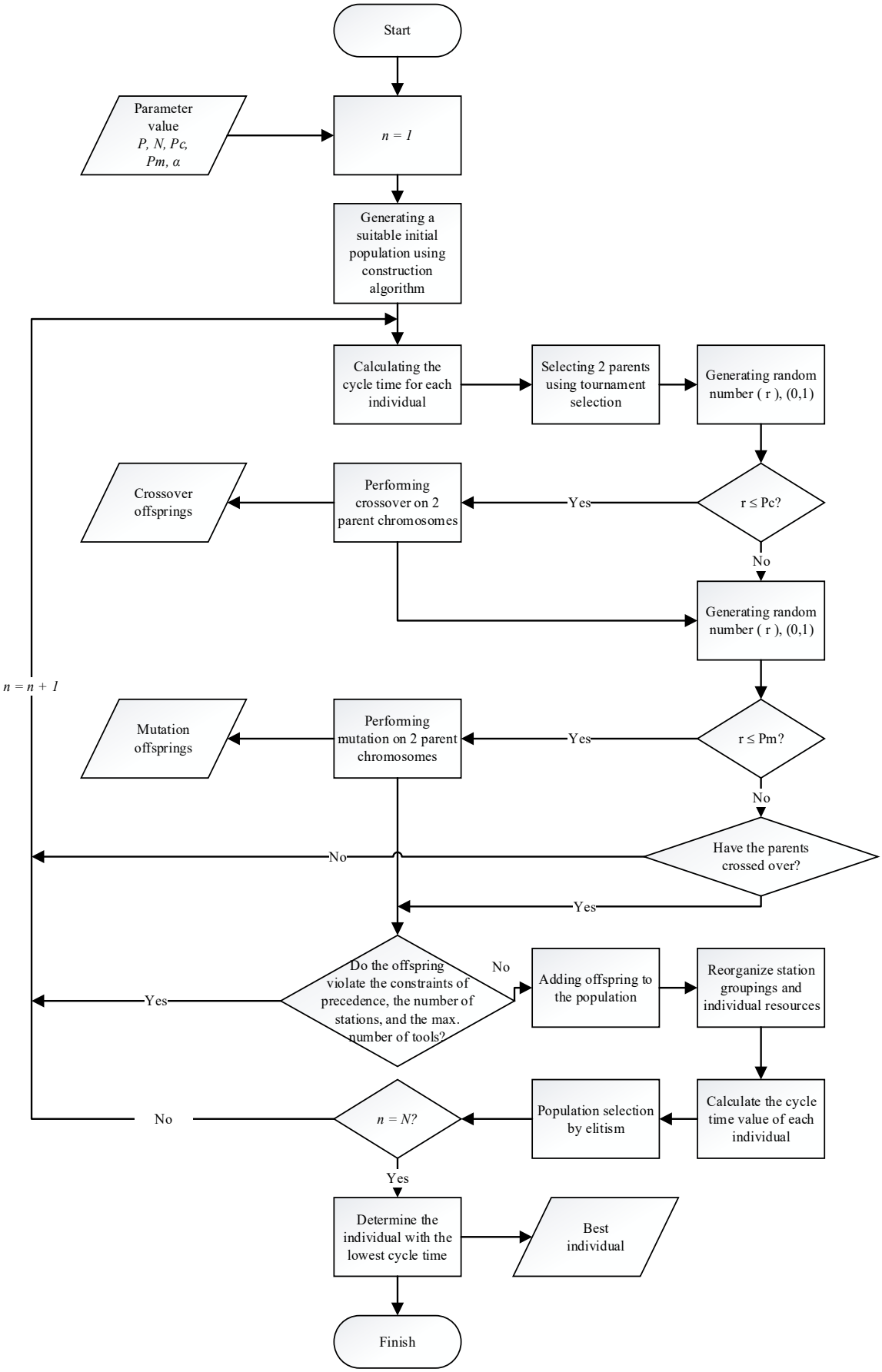

Fig. 1. Genetic algorithm flowchart

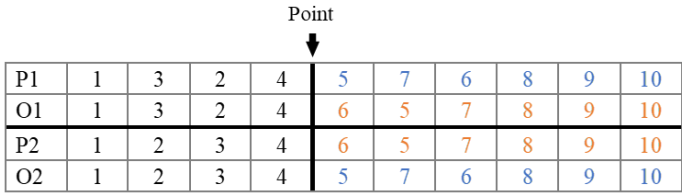

Fig. 2. One-point crossover mechanism

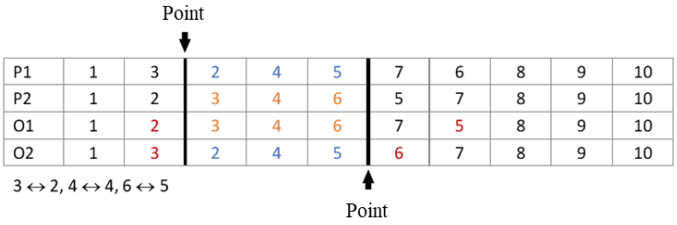

Fig. 3. PMX mechanism

2.2.4. Mutation

The mutation process involves modifying the chromosomes of parents and generating offspring. This process is carried out to enhance the diversity of the population and prevent getting stuck in local optimum solution values [41]. The mutation process must be capable of avoiding unfit solutions. Unlike crossover, mutation does not involve a recombination process. Thus, the mutation process occurs for each offspring/parent used as input. In this study, two mutation methods are applied: scramble mutation for task swapping, adapted from the research by Anwar et al. [45], and swap mutation for resource swapping, adapted from the study by Weckenborg et al. [21].

2.2.5. Elitism

Elitism is a selection method used to preserve the best individual for the next generation without any modification [46]. This process compares the fitness value of the worst individual in the population with the previously generated offspring. Elitism prevents the best individual from undergoing the reproductive process, allowing it to pass to the next generation without modification. This is done to ensure that the quality of the best individual remains in the population and is preserved during the evolutionary process.

2.2.6. Parameter Setting

One crucial aspect of performing GAs is selecting the appropriate parameter values to ensure the quality of the generated solutions. Experimental design (DOE) has been employed to determine the parameter combinations in this study. The method utilized for experimental design in this research is the complete factorial design. Full factorial design is commonly used for independent variables (X) greater than 2. Additionally, it explores all possible combinations of variables and their levels, allowing for a comprehensive understanding of the effects of each variable X and their interactions. There are four factors or variables (X) used in this study: population size (P), number of generations or iterations (N), crossover probability (P_c), and mutation probability (P_m). The experimental design is conducted using two levels: low and high. Determining parameter values for low and high levels is based on the number of tasks in the tested dataset and references used. Replication is also performed to reduce measurement errors and obtain more accurate results. In this experiment, replication is conducted three times. The parameter values used in the experimental design are shown in Table 2. If the level value of the population size and the number of generations or iterations is lower than the predetermined level, the solution will not find a near-optimal result. Meanwhile, if the level value exceeds Table 2, finding a solution will require a relatively longer computation time.

In this research, a parameter α is developed, which serves as a multiplier weight on the upper bound of cycle time. The upper bound of cycle time functions to determine additions at the workstations.

10.12928/ijio.v5i2.10027

The value of parameter α ranges from 1.0. It decreases gradually by 0.05 differences so that the resulting upper bound of cycle time will not exceed the initial upper bound of cycle time. With this implementation, the upper bound of cycle time incorporated into the algorithm tends to approach the optimal solution. Thus, it is expected to assist the algorithm in finding more efficient solutions.

Table 2. The levels of factors

Parameter	Small Data		Large Data	
	Low	High	Low	High
Population Size (P)	20	110	225	400
Number of Generation/Iteration (N)	100	200	210	300
Crossover Probability (P_c)	0.5	0.9	0.5	0.9
Mutation Probability (P_m)	0.01	0.2	0.01	0.2

3. Results and Discussion

In this research, several previous research datasets are utilized as inputs for testing the developed algorithm. The data to be processed comes from various references, each with multiple tasks. The data used in this study is shown in Table 3. From the table providing information about the research data, several supporting data will be employed, such as the number of tasks, processing time for each task for each resource used, the number of equipment types used, and the precedence diagram. The precedence diagram in previous research data is employed to identify the paths, relationships, and dependencies among tasks. Fig. 4 illustrates an example of a precedence diagram. Table 4 presents the processing times and equipment requirements used in this study, sourced from Weckenborg's data.

Table 3. Secondary data set

Category	Secondary Data	Number of Tasks	Number of Tools	Reference
Small Data	Weckenborg	10	4	Weckenborg, et al. [21]
	Su & Lu	17	0	Su & Lu [47]
	Nugraha 25	25	4	Nugraha, et al. [19]
	Nugraha 35	35	4	Nugraha, et al. [19]
Large Data	Nugraha 45	45	3	Nugraha, et al. [19]
	Kim	61	0	Kim, et al. [48]

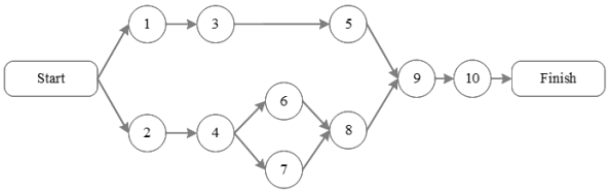

Fig. 4. Sample of precedence diagram

Table 4. Sample of task data details

Task	Process Time (m)			Tools	
	Human	Robot	HRC	Robot	HRC
1	8	M	6	2	1
2	7	10	5	3	3
3	6	M	M	1	4
4	4	M	3		2
5	5	11	4		
6	6	M	M		
7	5	11	4		
8	4	M	M		
9	7	M	5		
10	5	11	4		

3.1. Result of DOE

A complete factorial design is employed to identify the optimal parameter values and the significance of each parameter. In Minitab, four factors with two levels result in 16 combinations. Replication is conducted three times. Therefore, the total number of runs is 16 x 3 = 48 times. In performing the experimental design, ANOVA tests significant differences between group means or treatments [49]. ANOVA allows for determining whether these factors significantly impact the objective values and evaluating interactions among the factors. Using Minitab, the results of ANOVA and interactions between factors can be identified. Table 5 is the result of significance level ANOVA for small data and large data parameters.

Table 5. Result of ANOVA

Parameter	P-Value	
	Small Data	Large Data
P	0.000	0.000
N	0.003	0.000
P_c	0.567	0.011
P_m	0.061	0.143
$P*N$	0.243	0.375
$P* P_c$	0.556	0.375
$P* P_m$	0.556	0.766
$N* P_c$	0.243	0.766
$N* P_m$	0.023	0.766
$P_c * P_m$	0.556	0.766

In the small data category, the results indicate that the population size (P), the number of iterations (N), the interaction between the number of iterations (N), and the mutation probability (P_m) significantly affect the cycle time. In the large data category, the results show that the population size (P), the number of iterations (N), and the crossover probability (P_c) significantly influence the cycle time. The combinations of parameter values for the two data categories can be seen in.

Table 6. Result parameters setting

Data	Number of Tasks	*P*	*N*	*Pc*	*Pm*	*α*
Weckenborg	10	110	200	0.5	0.01	0.70
Su & Lu	17	110	200	0.5	0.01	0.70
Nugraha 25	25	110	200	0.5	0.01	0.70
Nugraha 35	35	400	300	0.9	0.01	0.80
Nugraha 45	45	400	300	0.9	0.01	0.80
Kim	61	400	300	0.9	0.01	0.80

The population size (P) refers to the number of feasible solution individuals that do not violate constraints for performing crossover and mutation processes in each generation or iteration. The higher the population size, the more possibilities of task and resource combinations that can be processed. The number of iterations (N) is the number of generations or iterations needed in the solution search process. Also, it serves as a stopping rule in the development of the GA. Generations in the GA involve selection, crossover, and mutation to produce a better population. A higher number of generations can provide a more significant opportunity for individual improvement. The research aligns with the study by Zhang & Chen [50], stating that a larger population size and a higher number of generations can explore many possibilities for better individual solutions. The significant crossover probability effect in the large data category may be due to the larger and more complex solution space, allowing for better solution exploration.

3.2. Computation Result

The Weckenborg data is used as one of the case examples in the development of the GA. The Weckenborg data has a task count of 10, with a maximum number of workstations being 3, 1 available robot, and a maximum number of tool types at one workstation being 4, resulting in an upper bound

 10.12928/ijio.v5i2.10027

cycle time of 26 minutes. Validation is carried out throughout the GA development model. The best solutions generated are ensured not to violate existing constraint functions and have values consistent with the computational results of the reference model by Ma'ruf et al. [4]. The computational results can be seen in Table, with a visualization of the precedence diagram in Fig. 5 and a Gantt chart in Fig. 6.

Table 7. Computation result of Weckenborg's data solution

Workstation	Task	Resource	Tools	Process Time (m)	Start time (m)	Finish Time (m)	Station Time (m)
1	1	H	tools 3 tools 2	8	0	8	17
	3	H		6	8	14	
	2	R		10	0	10	
	4	HRC		3	14	17	
2	5	H		5	0	5	16
	6	H		6	5	11	
	7	H		5	11	16	
3	8	H		4	0	4	16
	9	H		7	4	11	
	10	H		5	11	16	

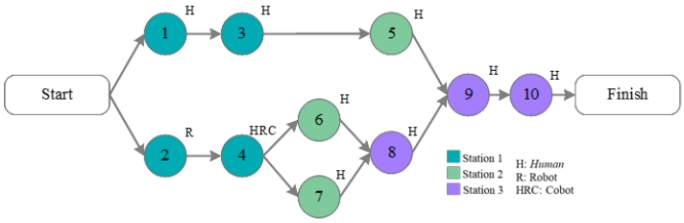

Fig. 5. Precedence diagram of Weckenborg's data solution

The results of the improvement solutions indicate that they do not violate constraint functions, including precedence constraints, the number of workstations, the number of robots used, and the number of tool types used. Based on the computational process for the improvement solutions, the objective function values match those of the reference model by Ma'ruf et al. [4]. Therefore, it can be concluded that GA development with the programming code is correct and aligns with the reference model.

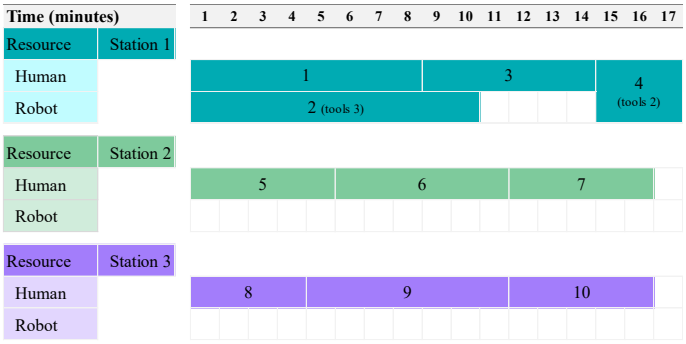

Fig. 6. Gantt Chart of Weckenborg's data solution

In this study, two methods in the crossover process have been adopted: a one-point crossover and a partially mapped crossover. Data collection was conducted 200 times using Weckenborg's data to observe the percentage usage of the crossover methods. The performance of each crossover method was evaluated by ensuring that the obtained results did not violate the constraints in the algorithm. Subsequently, the performance of each method was compared. The results of the comparison of crossover methods show that the performance of the one-point crossover method, with a passing percentage of 82%, is better than the partially mapped crossover method, which has a passing rate of

58%, for the ALB-HRC case studied. This is because the steps in the partially mapped crossover method, such as segment mapping, are more likely to violate precedence constraints.

3.3. Comparison between MIP and GA

The computational results obtained using GA with Python software will be compared with those obtained using the analytical method. Analytical computations were performed using a solver application with a computation time limit of 24 hours (86400 seconds), following the research conducted by Ma'ruf et al. [4] and Nugraha et al. [19]. GA computations were replicated three times, and the objective value used to compare GA and the analytical method is the minimum objective value obtained from all computations. The comparison of objective values between the analytical process and GA can be seen in Fig. 7. Meanwhile, the comparison of computation times between the analytical method and GA can be seen in Fig. 8.

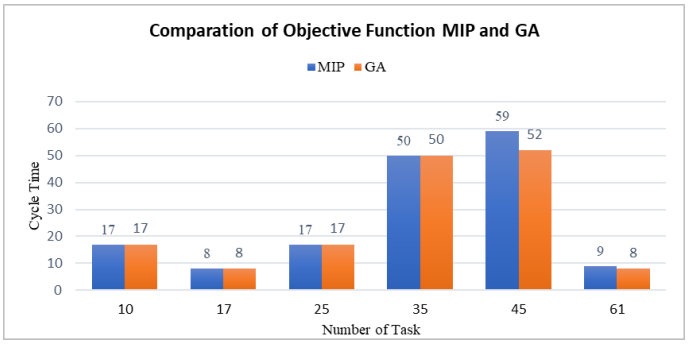

Fig. 7. Comparison of objective value between MIP and GA

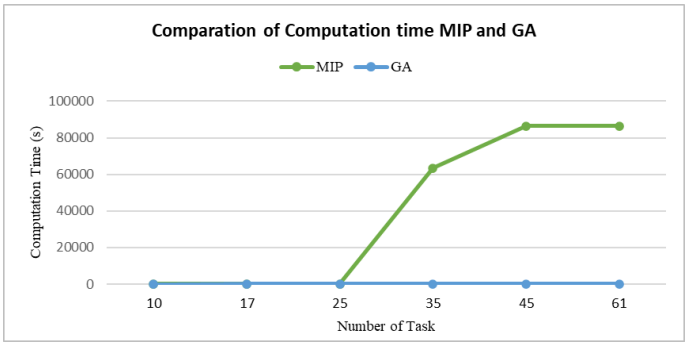

Fig. 8. Comparison of time computation between MIP and GA

The computational results consist of two aspects to be examined: a comparison of the objective values in this study, cycle time, and computation time. For reference, the objective values in the analytical method for the number of tasks 45 and 61 are solutions obtained from computation limited to 24 hours or 86,400 seconds. Based on the comparison graph, it can be seen that for small data consisting of Weckenborg, Su & Lu, and Nugraha 25 data, the solutions obtained from the GA computations can achieve optimal solutions corresponding to those from the analytical method. Optimal solutions matching the analytical method are also obtained for Nugraha 35 in the large data group. Meanwhile, for Nugraha 45 and Kim data, GA can produce objective values close to better optimal solutions than the analytical method. Based on the comparison of solutions obtained from computations using the analytical method and GA, the average difference in objective values is 3.83%.

Computation time, referring to the total time an algorithm needs to solve a problem, is a crucial aspect of this research. A comparison of computation time was conducted between the analytical method and GA. The computation process in the analytical method was limited to 24 hours or 86,400 seconds. Based on the comparison of computation time between the two methods, it can be observed that the computation time with GA is relatively much faster than the analytical method, with an average difference in computation time of 64.66%. The significant difference is particularly evident in the large data group: Nugraha 35, Nugraha 45, and Kim. The disparity in computation time between

10.12928/ijio.v5i2.10027

the analytical method and GA increases with the growing number of tasks in the data group and the complexity of the relationships between tasks. The α parameter helps the algorithm produce an optimal solution on the Weckenborg data set and obtain near-optimal results on the Kim data set. By omitting the α parameter, the Weckenborg and Kim data sets produce worse solutions. Thus, it requires more iterations and a longer computation time. The proposed α parameter can help the algorithm become more efficient, with an average increase in computational efficiency of 39%.

GA effectively optimizes the cycle time in the ALBP-HRC case based on the experiments that were conducted. The ability of the genetic algorithm to produce optimal solutions in the tested datasets can be attributed to its population-based solution search, enabling it to explore the solution space widely and locate potential solutions in various regions of the search space. Additionally, successfully executing critical stages contributes to GA finding optimal solutions. The genetic algorithm's capability to broadly explore the solution space is supported by several crucial stages: generating the initial population, conducting tournament selection, crossover, mutation, and elitism. The initial population allows the algorithm to explore various combinations in the search for individuals. Tournament selection enables potentially productive individuals to generate offspring, resulting in diverse solutions within the population. Crossover can create new alternative solutions, complemented by the mutation process, to prevent potential solutions from being trapped in local optima. Finally, elitism is employed to evaluate the fitness value of solutions and ensure that promising solutions are retained within the population.

3.4. Managerial Implication

This research assumes the reachability and accessibility of the HRC in each station. If the HRC is assigned to a particular workstation, a layout design should be considered. Other boundaries, such as the shop-floor boundary and non-overlapping resource footprints between human operator and cobot, should also be considered [51], [52]. As Duan states, it is also essential to consider the handover between the human operator and the robot from the anthropometric point of view [53].

Cobot is known to be small in size, increasing the moveability and reconfigurability but limited in payload. An extra safety consideration should be considered for heavy or oversized parts [54]. Another practical issue is the downtime of the assembly line during the reconfiguration process when new product variants or technologies are introduced into the assembly line system [55]. Even though the cobot is claimed to be agile, reconfiguring the whole assembly line should also be considered during the planning horizon.

4. Conclusion

This research proposes a GA for the ALBP-HRC problem, considering several tools and setups. The GA can provide optimal solutions on tasks less than 35 and an average gap of 3.83% relative to the optimal solution. The computation time is faster than that of the analytical method, with an average computation time difference of 64.66%. The identified significant parameters for minimizing cycle time in small and large dataset categories are the number of iterations (N) and population size (P) to enhance the likelihood of obtaining better solutions. Companies can consider assembly lines utilizing HRC applied in the industrial sector. Deploying resources with HRC can work alongside workers to handle complex or monotonous tasks, thus facilitating and accelerating production. With tools / end-effectors, HRC can be dynamic and expandable to perform various tasks on the assembly floor, enabling rapid and effective changes in the production process. The use of HRC on assembly lines is highly suitable for industries seeking to achieve high efficiency, quality, and flexibility in the production process. The fast computation times can help improve efficiency and effectiveness in decision-making related to frequent redesigning of assembly. Future research will be conducted for ALBP-HRC, where multiple product variants are assembled in a single-line assembly.

Author Contribution: All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.

Funding: This research received no external funding

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] F. Lestari, "Perancangan Lintas Perakitan pada Product Family berdasarkan Common Subassembly," *SNTIKI III*, pp. 430–436, 2011, [Online]. Available: https://ejournal.uin-suska.ac.id/

[2] M. Eghtesadifard, M. Khalifeh, and M. Khorram, "A systematic review of research themes and hot topics in assembly line balancing through the Web of Science within 1990-2017," *Computers & Industrial Engineering*, 2019, doi: 10.1016/j.cie.2019.106182.

[3] N. H. Kamarudin and M. F. F. A. Rashid, "Assembly line balancing with resource constraints using new rank-based crossovers," *Journal of Physics: Conference Series*, vol. 908, 2017, doi: 10.1088/1742-6596/908/1/012059.

[4] A. Ma'ruf, C. Nugraha, and A. S. Tarigan, "The Development of Human-Robot Collaborative Assembly Line Model by Considering Availability of Robots , Tools , and Setup Time," *Jurnal Ilmiah Teknik Industri*, pp. 319–327, 2022, doi: 10.23917/jiti.v21i2.19619.

[5] M. Kheirabadi, S. Keivanpour, Y. A. Chinniah, and J. M. Frayret, "Human-robot collaboration in assembly line balancing problems: Review and research gaps," *Computers & Industrial Engineering*, vol. 186, Dec. 2023, doi: 10.1016/j.cie.2023.109737.

[6] T. Kiyokawa *et al.*, "Difficulty and complexity definitions for assembly task allocation and assignment in human–robot collaborations: A review," *Robotics and Computer-Integrated Manufacturing*, vol. 84. Elsevier Ltd, Dec. 01, 2023. doi: 10.1016/j.rcim.2023.102598.

[7] L. Gualtieri, E. Rauh, and R. Vidoni, "Methodology for the definition of the optimal assembly cycle and calculation of the optimized assembly cycle time in human-robot collaborative assembly," *International Journal of Advanced Manufacturing Technology*, 2021, doi: 10.1007/s00170-021-06653-y.

[8] A. Nourmohammadi, M. Fathi, and A. H. C. Ng, "Balancing and scheduling assembly lines with human-robot collaboration tasks," *Computers & Operations Research*, vol. 140, Apr. 2022, doi: 10.1016/j.cor.2021.105674.

[9] S. Puttero, E. Verna, G. Genta, and M. Galetto, "Towards the modelling of defect generation in human-robot collaborative assembly," in *Procedia CIRP*, Elsevier B.V., 2023, pp. 247–252. doi: 10.1016/j.procir.2023.06.043.

[10] M. D. Mura and G. Dini, "CIRP Annals - Manufacturing Technology Designing assembly lines with humans and collaborative robots : A genetic approach," *CIRP Annals - Manufacturing Technology*, vol. 68, no. 1, pp. 1–4, 2019, doi: 10.1016/j.cirp.2019.04.006.

[11] L. Gualtieri, F. Fraboni, M. De Marchi, and E. Rauch, "Development and evaluation of design guidelines for cognitive ergonomics in human-robot collaborative assembly systems," *Applied Ergonomics*, vol. 104, Oct. 2022, doi: 10.1016/j.apergo.2022.103807.

[12] A. Scholl, *Balancing and Sequencing of Assembly Lines*. Darmstadt: Physica-Verlag Heidelberg, 1999, doi: 10.1007/978-3-662-11223-6.

[13] O. Bagaria, "Importance of Cycle time Reduction for Productivity Improvement," *JETIR*, vol. 6, no. 4, pp. 802–805, 2019, [Online]. Available: https://www.jetir.org/papers/JETIREO06173.pdf

[14] R. Chase, F. R. Jacobs, and N. Aquilano, *Operations Management for Competitive Advantage*, 11th ed. New York: The McGraw-Hill Companies, Inc., 2006.

[15] C. Andres, C. Miralles, and R. Pastor, "Balancing and scheduling tasks in assembly lines with sequence-dependent setup times," *European Journal of Operational Research*, vol. 187, pp. 1212–1223, 2008, doi: 10.1016/j.ejor.2006.07.044.

[16] D. Andronas, S. Xythalis, P. Karagiannis, G. Michalos, and S. Makris, "Robot gripper with high speed, in-hand object manipulation capabilities," in *Procedia CIRP*, Elsevier B.V., 2020, pp. 482–486. doi: 10.1016/j.procir.2020.08.007.

[17] Z. Y. Deng, L. W. Kang, H. H. Chiang, and H. C. Li, "Integration of Robotic Vision and Automatic Tool Changer Based on Sequential Motion Primitive for Performing Assembly Tasks," in *IFAC-PapersOnLine*, Elsevier B.V., Jul. 2023, pp. 5320–5325. doi: 10.1016/j.ifacol.2023.10.175.

[18] B. Zhang, Y. Xie, J. Zhou, K. Wang, and Z. Zhang, "State-of-the-art robotic grippers, grasping and control strategies, as well as their applications in agricultural robots: A review," *Computers and Electronics in Agriculture*, vol. 177. Elsevier B.V., Oct. 01, 2020. doi: 10.1016/j.compag.2020.105694.

[19] R. C. Nugraha, A. Ma'ruf, A. C. Nugraha, and A. H. Halim, "A mixed-integer linear programming formulation for assembly line balancing problem with human- robot shared tasks," *Journal of Physics: Conference Series*, 2020, doi: 10.1088/1742-6596/1858/1/012021.

[20] A. Yaphiar, Susanto; Nugraha, Cahyadi; Ma'ruf, "Mixed Model Assembly Line Balancing for Human-Robot Shared Tasks," *iMEC-APCOMS*, pp. 245–252, 2019, doi: 10.1007/978-981-15-0950-6_38.

[21] C. Weckenborg, K. Kieckha¨fer, C. Muller, M. Grunewald, and T. S. Splenger, "Balancing of assembly lines with collaborative robots," *Business Research*, pp. 93–132, 2020, doi: 10.1007/s40685-019-0101-y.

[22] I. Dimény, T. Koltai, C. Sepe, T. Murino, V. Gallina, and T. Komenda, "MILP model to decrease the MILP model to decrease the number of workers MILP model to decrease the number of workers in in assembly assembly lines lines with collaboration of workers in assembly lines with human-robot MILP model to decrease the number col," in *IFAC PapersOnLine*, Elsevier Ltd, 2021, pp. 169–174. doi: 10.1016/j.ifacol.2021.08.019.

[23] Z. Li, M. N. Janardhanan, Q. Tang, and S. G. Ponnambalam, "Model and metaheuristics for robotic two-sided assembly line balancing problems with setup times," *Swarm and Evolutionary Computation*, vol. 50, no. October 2018, 2019, doi: 10.1016/j.swevo.2019.100567.

[24] M. N. Janardhanan, Z. Li, G. Bocewicz, Z. Banaszak, and P. Nielsen, "Metaheuristic algorithms for balancing robotic assembly lines with sequence-dependent robot setup times," *Applied Mathematical Modelling*, vol. 65, pp. 256–270, 2019, doi: 10.1016/j.apm.2018.08.016.

[25] C. Zhang, J. Dou, and P. Wang, "Configuration design of reconfigurable single-product robotic assembly line for capacity scalability," *Computers & Industrial Engineering*, vol. 185, Nov. 2023, doi: 10.1016/j.cie.2023.109682.

[26] N. P. Basán, M. E. Cóccola, A. García del Valle, and C. A. Méndez, "Scheduling of flexible manufacturing plants with redesign options: A MILP-based decomposition algorithm and case studies," *Computers & Chemical Engineering*, vol. 136, May 2020, doi: 10.1016/j.compchemeng.2020.106777.

[27] S. E. Hashemi-Petroodi, S. Thevenin, and A. Dolgui, "Mixed-Model Assembly Line Design with New Product Variants in Production Generations," in *IFAC-PapersOnLine*, Elsevier B.V., 2022, pp. 25–30. doi: 10.1016/j.ifacol.2022.09.363.

[28] Z. Mao, Y. Sun, K. Fang, D. Huang, and J. Zhang, "Model and metaheuristic for human–robot collaboration assembly line worker assignment and balancing problem," *Computers & Operations Research*, vol. 165, p. 106605, May 2024, doi: 10.1016/j.cor.2024.106605.

[29] L. Rozo *et al.*, "The e-Bike motor assembly: Towards advanced robotic manipulation for flexible manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 85, Feb. 2024, doi: 10.1016/j.rcim.2023.102637.

[30] A. Nourmohammadi, M. Fathi, A. H. C. Ng, and E. Mahmoodi, "A genetic algorithm for heterogenous human-robot collaboration assembly line balancing problems," in *Procedia CIRP*, Elsevier B.V., 2022, pp. 1444–1448. doi: 10.1016/j.procir.2022.05.172.

[31] S.-G. Liao, Y.-B. Zhang, C.-Y. Sang, and H. Liu, "A Genetic Algorithm for Balancing and Sequencing of Mixed-Model Two-Sided Assembly Line with Unpaced Synchronous Transfer," *Applied Soft Computing*, vol. 146, 2023, doi: 10.1016/j.asoc.2023.110638.

[32] S. S. Tjandra, F. Setiawan, and H. Salsabila, "Jurnal Optimasi Sistem Industri Application

of Genetic Algorithms to Solve MTSP Problems with Priority (Case Study at the Jakarta Street Lighting Service),” *Jurnal Optimasi Sistem Industri*, vol. 21, pp. 75–86, 2022, doi: 10.25077/josi.v21.n2.p75-86.2022.

[33] N. Harale, S. Thomassey, and X. Zeng, “Dynamic small-series fashion order allocation and supplier selection : a ga-topsis-based model,” *International Journal of Industrial Optimization*, vol. 4, no. 2, pp. 82–102, 2023, doi: 10.12928/ijio.v4i2.7640.

[34] P. Segura, O. Lobato-calleros, A. Ramírez-serrano, and I. Soria, “Human-robot collaborative systems : Structural components for current manufacturing applications,” *Advances in Industrial and Manufacturing Engineering*, vol. 3, 2021, doi: 10.1016/j.aime.2021.100060.

[35] N. Berx, W. Decre, and L. Pintelon, “Examining the Role of Safety in the Low Adoption Rate of Collaborative Robots,” in *Procedia CIRP*, Elsevier B.V., 2022, pp. 51–57. doi: 10.1016/j.procir.2022.02.154.

[36] K. Merckaert, B. Convens, M. M. Nicotra, and B. Vanderborght, “Real-time constraint-based planning and control of robotic manipulators for safe human–robot collaboration,” Robotics and Computer-Integrated Manufacturing, vol. 87, Jun. 2024, doi: 10.1016/j.rcim.2023.102711.

[37] K. Katsampiris-Salgado *et al.*, “Collision detection for collaborative assembly operations on high-payload robots,” *Robotics and Computer-Integrated Manufacturing*, vol. 87, Jun. 2024, doi: 10.1016/j.rcim.2023.102708.

[38] J. Shu, W. Li, and Y. Gao, “Collision-free trajectory planning for robotic assembly of lightweight structures,” *Automation in Construction*, vol. 142, Oct. 2022, doi: 10.1016/j.autcon.2022.104520.

[39] N. Boysen, P. Schulze, and A. Scholl, “Assembly line balancing : What happened in the last fifteen years?,” *European Journal of Operational Research,* vol. 301, pp. 797–814, 2022, doi: 10.1016/j.ejor.2021.11.043.

[40] D. M. Utama, L. R. Ardiansyah, and A. K. Garside, “Penjadwalan Flow shop Untuk Meminimasi Total Tardiness Menggunakan Algoritma Cross Entropy – Algoritma Genetika,” *Jurnal Optimasi Sistem Industri*, vol. 2, pp. 133–141, 2019, doi: 10.25077/josi.v18.n2.p133-141.2019.

[41] J. C. Chen, Y. Chen, T. Chen, and Y. Kuo, “Applying two-phase adaptive genetic algorithm to solve multi-model assembly line balancing problems in TFT – LCD module process,” *Journal of Manufacturing Systems*, vol. 52, no. May, pp. 86–99, 2019, doi: 10.1016/j.jmsy.2019.05.009.

[42] D. Shukla, A.; Pandey, H. M.; Mehrotra, “Comparative Review of Selection Comparative Review of Selection,” *2015 1st International Conference on Futuristic trend in Computational Analysis and Knowledge Management*. 2015. doi: 10.1109/ABLAZE.2015.7154916.

[43] D. Chakraborti, P. Biswas, and B. B. Pal, “FGP Approach for Solving Fractional Multiobjective Decision Making Problems using GA with Tournament Selection and Arithmetic Crossover,” *Procedia Technology*, vol. 10, pp. 505–514, 2013, doi: 10.1016/j.protcy.2013.12.389.

[44] A. K. Pachuau, Joseph L.; Roy, Arnab; Saha, “An Overview of Crossover Techniques in Genetic Algorithm,” *Proceedings of CoMSO: Modeling, Simulation Modeling, Simulation*. pp. 581–598, 2020. doi: 10.1007/978-981-15-9829-6_46.

[45] S. M. Anwar, A. M. Ali, and M. A. Awad, “Single Model Assembly Line Balancing Using Enhanced Genetic Algorithm,” *Saudi Journal of Engineering and Technology*, no. December 2019, pp. 494–501, 2020, doi: 10.36348/sjet.2019.v04i12.003.

[46] S. L. Yadav and A. Sohal, “Comparative Study of Different Selection Techniques in Genetic Algorithm,” *International Journal of Engineering Science and Mathematics*, vol. 6, no. 3, pp. 174–180, 2017, [Online]. Available: https://ijesm.co.in/uploads/68/3180_pdf.pdf

[47] Y. Su, Ping; Lu, “Combining Genetic Algorithm and Simulation for the Mixed-model Assembly Line Balancing Problem,” *Third International Conference on Natural*

Computation, no. Icnc, pp. 174–180, 2007, doi: 10.1109/ICNC.2007.306.

[48] Y. J. Kim, Y. K.; Lee, S. Y.; Kim, "A Genetic Algorithm for Improving the Workload Smoothness in Mixed Model Assembly Lines.," *Journal of the Korean Institute of Industrial Engineers*, pp. 515–532, 1997. Available: https://koreascience.kr/article/JAKO199729464079022.page

[49] B. Durakovic, "Design of Experiments Application , Concepts , Examples : State of the Art," *Periodicals of Engineering and Natural Sciences*, vol. 5, pp. 421–439, 2021, doi: 10.21533/pen.v5i3.145.

[50] Y. Zhang and R. Chen, "Energy-efficient scheduling of imprecise mixed-criticality real-time tasks based on genetic algorithm," *Journal of Systems Architecture*, vol. 143, 2023, doi: 10.1016/j.sysarc.2023.102980.

[51] C. Seeber, M. Albus, M. Fechter, A. Neb, and S. I. Yoshida, "Automated 2D Layout Design of Assembly Line Workstations through Physical Principles," in *Procedia CIRP*, Elsevier B.V., 2021, pp. 1197–1202. doi: 10.1016/j.procir.2021.11.201.

[52] B. Su, S. H. Jung, L. Lu, H. Wang, L. Qing, and X. Xu, "Exploring the impact of human-robot interaction on workers' mental stress in collaborative assembly tasks," *Applied Ergonomics*, vol. 116, Apr. 2024, doi: 10.1016/j.apergo.2024.104224.

[53] H. Duan, Y. Yang, D. Li, and P. Wang, "Human-robot object handover: Recent progress and future direction," *Biomimetic Intelligence and Robotics*, p. 100145, Feb. 2024, doi: 10.1016/j.birob.2024.100145.

[54] V. Gopinath, K. Johansen, M. Derelöv, Å. Gustafsson, and S. Axelsson, "Safe Collaborative Assembly on a Continuously Moving Line with Large Industrial Robots," *Robotics and Computer-Integrated Manufacturing*, vol. 67, Feb. 2021, doi: 10.1016/j.rcim.2020.102048.

[55] M. Eswaran *et al.*, "Optimal layout planning for human robot collaborative assembly systems and visualization through immersive technologies," *Expert Systems with Applications*, vol. 241, May 2024, doi: 10.1016/j.eswa.2023.122465.