energies

Article

Energy-Efficient Robot Configuration and Motion Planning Using Genetic Algorithm and Particle Swarm Optimization

Kazuki Nonoyama, Ziang Liu, Tomofumi Fujiwara, Md Moktadir Alam and Tatsushi Nishi *

Graduate School of Natural Science and Technology, Okayama University, 3-1-1 Tsushima-naka, Kita-ku, Okayama City 700-8530, Okayama, Japan; pws575rj@s.okayama-u.ac.jp (K.N.); liu.ziang@okayama-u.ac.jp (Z.L.); fujiwara.tomofumi@okayama-u.ac.jp (T.F.); moktadir.alam@okayama-u.ac.jp (M.M.A.)
* Correspondence: nishi.tatsushi@okayama-u.ac.jp

Abstract: The implementation of Industry 5.0 necessitates a decrease in the energy consumption of industrial robots. This research investigates energy optimization for optimal motion planning for a dual-arm industrial robot. The objective function for the energy minimization problem is stated based on the execution time and total energy consumption of the robot arm configurations in its workspace for pick-and-place operation. Firstly, the PID controller is being used to achieve the optimal parameters. The parameters of PID are then fine-tuned using metaheuristic algorithms such as Genetic Algorithms and Particle Swarm Optimization methods to create a more precise robot motion trajectory, resulting in an energy-efficient robot configuration. The results for different robot configurations were compared with both motion planning algorithms, which shows better compatibility in terms of both execution time and energy efficiency. The feasibility of the algorithms is demonstrated by conducting experiments on a dual-arm robot, named as duAro. In terms of energy efficiency, the results show that dual-arm motions can save more energy than single-arm motions for an industrial robot. Furthermore, combining the robot configuration problem with metaheuristic approaches saves energy consumption and robot execution time when compared to motion planning with PID controllers alone.

Keywords: robot motion planning; robot placement; optimization; PID; genetic algorithm; particle swarm optimization

Citation: Nonoyama, K.; Liu, Z.; Fujiwara, T.; Alam, M.M.; Nishi, T. Energy-Efficient Robot Configuration and Motion Planning Using Genetic Algorithm and Particle Swarm Optimization. *Energies* **2022**, *15*, 2074. https://doi.org/10.3390/en15062074

Academic Editors: Luigi Fortuna and Andrzej Bielecki

Received: 7 February 2022
Accepted: 9 March 2022
Published: 11 March 2022

1. Introduction

Sustainable development requires energy-efficient manufacturing. The target of doubling global energy efficiency by 2030 is one of the major objectives of the Sustainable Development Goal (SDG 7.3) [1]. According to the report of Energy Information Administration (EIA) [2], globally, the industrial sector consumes more energy than any other sector, accounting for around 54% of total supplied energy. Furthermore, in this phase of Industry 5.0, industrial robots have a broad range of applications in current production and will continue to impact smart manufacturing because of their superior repeatability, controllability, and flexibility, and may face a substantial problem due to inefficiency in energy consumption. The robot's energy consumption in the industries may be very high due to its wide applicability and global sales expansion. Because of the rising cost of energy sources and the diminishing green environment, the minimization of energy usage for robot motions is becoming an increasing concern in industries. Therefore, modeling energy-efficient robot placements are indispensable. Optimization in robot path planning has been established as one of the ways for enhancing energy efficiency in robotic systems.

Numerous approaches have been used to solve the challenges in the optimization of path planning [3], which is crucial in the model of robotics. The strategies generally seek to find a solution that maximizes the achievement of a collection of objectives while also identifying a set of the best actions given a set of limitations. Optimization techniques have a long history of handling hard-optimization challenges in industrial applications such as resource distribution, task planning, system optimization, and path planning. Some metaheuristic computational techniques, for instance, Ant Colony Optimization (ACO), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and others, have demonstrated their applicability in solving optimization issues to the path planning in robotic research [4,5]. Recently, Zaplana et al. [6] implemented a GA for balancing the workload of industrial lines, which is directly related to efficient energy consumption.

Most of these studies focus on mobile robots in the path and/or motion optimization; however, they ignore energy optimization or do not consider simultaneous robot configuration and motion planning for energy-efficient pick-and-place operations. Furthermore, if the energy efficient operation is achieved, the traveling time of robot arms is increased, and then the throughput of the multiple arms may be deteriorated. Moreover, conventional motion planning strategies are based on the layout of robot configurations and the application of the robotic system with the empirical knowledge of System Integrator (SIer). However, it has been a significant burden for Sier to provide efficient system integration knowledge for each robot configuration. Therefore, it is required to strengthen the Sier's decision-making process with optimized robot configurations and motion planning for the robot manipulators to minimize total energy consumption.

In this study, the major objective is to develop an energy-efficient workpiece transport model for the dual-arm robot for industrial applications. Dual-arm robots handle a greater variety of applications than single-arm robot manipulators, primarily due to their greater flexibility and load capacity, as well as their greater capability for cooperative operations. Dual-arm robots are not merely the combination of two single-arm robots; instead, they work together concurrently as a unit. In comparison to the single-arm manipulator, dual-arm operations increase collaboration between both arms. Environmental adaptation and coordinated control between dual-arm manipulators are critical and need more complex path planning [7]. As a result, researchers have focused their efforts on trajectory optimization and dual-arm coordination, which requires both coordinated trajectory planning and coordinated motion control at the same time.

Firstly, in this research, the robot motions are modeled and analyzed by a GA method by tuning Proportional-Integral-Differential (PID) controller gain adjustment, which can significantly improve energy efficiency. The associated torque is analyzed using four objective minimization functions in a PID-based control simulation method. The variants for objective functions are (1) number of times each joint angle surpasses the commanded angle, (2) time at which each joint angle surpasses commanded angle, (3) total execution time for the robot manipulator, and (4) total energy consumption. Nevertheless, because of the non-linear nature of the robot's joint angles, direct mathematical formulae cannot be used to estimate the optimal PID gain. Therefore, when the optimal solution is obtained by a precise optimization method, for example, the branch and bound technique, computation time for robot motion planning exponentially increases with the increasing number of joints. Because of the longer execution time, this strategy is not practicable. To address this issue, this study prioritizes obtaining a robot motion planning that may correspond to the dual-arm robot motion trajectory by using a GA method to identify the local optimum solution rather than the global optimum solution. The simulation software K-ROSET is being utilized to verify the accuracy of the robot trajectory. Analytical results using GA show that dual-arm robot motion is more energy efficient in various configurations than single-arm robot motion.

Furthermore, an empirical study is conducted to examine whether the robot configuration has any further impact on the energy consumption of the robot. The results indicate the possibility of obtaining better solutions by including the robot's configurations as decision variables. For further improvement, a metaheuristic algorithm, PSO, is applied for the solution of optimized energy usage for the dual-arm robot configurations and motion planning problem. The experimental results show that PSO performs consistently well and is stable for this problem compared with GA.

The main contribution of this study is to apply metaheuristic theories (GA and PSO algorithms) and computations to the industrial robots with a dual-arm configuration to gain energy efficiency in different robot configurations for a workpiece transportation system using a conveyor belt. As detailed in Section 2, these algorithms have been developed and applied mainly to mobile types of robots or industrial robots with a single-arm rather than industrial robots with multiple arms by many researchers. The merit of the proposed method is that it does not require to specify the placement of the robot, trajectory, and velocity planning of the robot arms in advance. Moreover, researchers applied the kinematic approach, which can be easily derived, and the relation between motion-planning (joint planning) and path-planning (cartesian planning) will be, thus, straightforward. This will largely simplify the control and thus the optimization. However, in many solutions, the kinematic model itself cannot provide the optimum motion planning. In this study, we decoupled the acting force in the joints of the robot arm to calculate the energy-efficient robot motion trajectory of the robot arm, which can be considered as another novelty of our work.

The following is a breakdown of the rest of the paper's structure. In Section 2, the most recent research on energy efficiency techniques and robot path planning will be highlighted. In Section 3, the robot motion planning problem is formulated, and GA is applied to this problem. Section 4 introduces the robot placement variables to the motion planning problem, and PSO is used to solve the optimal robot placement and motion planning problem. Section 5 presents the computational results and conducts simulation and implementation experiments on a real-world dual-arm robot. The final section summarizes the conclusions.

2. Literature Review

The energy consumption characteristics of industrial robots are increasingly being studied and predicted by robot manufacturers and research groups. Liu et al. [8] developed an energy consumption model to study power utilization and dynamic parameter identification of a robotic manipulator with a single arm. As a function of the robot joint velocities and accelerations, Paes et al. [9] proposed time and energy-optimal paths in a limited workspace. Paryanto and his research group [10] studied the energy profile and dynamic characteristics for a 6DOF robot with different speeds and payloads in an assembly system. The energy-optimal robot motion parameters inside the manufacturing work-cell of mechanical structure and control logics of industrial robots were modeled by Gadaleta et al. [11,12]. Researchers have also studied multimodal robot trajectory optimization for motion planning using a cost function [13,14], as well as extending the framework for optimized energy and maximum power consumption [14]. Researchers also presented theoretical research on the energy savings that could be realized by optimizing robot motion acceleration characteristics [15]. Their study is limited to specific motion profiles that are not affected by manufacturing cycle times. Software-simulated power data were used in several studies to identify robot dynamics and predict energy-efficient robotic motion [8,16]. Pellicciari et al. [17] suggested a numerical technique for lowering industrial robots' energy usage. The pre-scheduled robot paths are subjected to a whole-time rescaling in this method, which addresses the scheduling gap problem in manufacturing. Horla et al. obtained energy optimal trajectories for an unmanned aerial vehicle using swarm optimization methods [18].

Several state-of-the-art ways of modeling and planning the energy profile for robots have been presented as the methods for total energy optimization and reducing negative environmental effects. Feng et al. [19] introduced a novel numerical energy consumption model for a cyclic pick-and-place robotic operation by optimal joint placements to maximize energy saving. Ji et al. [20] devised a mechanism for fine-tuning the stiffness of the robot actuator to achieve time-energy optimal trajectory planning. Gadaleta et al. [11] introduced an offline programming-based simulation tool to automatically generate code to save energy consumption. The researcher has also adopted non-linear programming optimization approaches that can facilitate multiple-robot coordination [20,21]. They devised methods for reducing energy consumption while preserving productivity and smoothing the trajectory of the robot.

During the previous two decades, robotics scientists and engineers have highly focused on the issue of motion planning with obstacle avoidance. Path planning for robot manipulators is primarily concerned with determining the best collision-free path from a start to an end pose. Furthermore, optimizing a robot's path planning in its workspace leads to a reduction in energy consumption. Sangiovanni et al. [22] developed an ad hoc Deep Reinforcement Learning (DRL) technique for obstacle avoidance and path planning optimization for a self-configuring robot. Hovgard et al. [23] developed an optimization approach for reducing energy consumption in multi-robot systems by determining the best execution time and order of robot motions through motion parameter modification. To acquire the optimized path planning, several heuristic-based algorithms such as neural network (NN) [24], fuzzy logic (FL) [25], and nature-inspired algorithms, including GA [26], PSO [27], and ACO [28], as well as certain Artificial Potential Field Algorithm (APFA) [29,30] and some other hybrid models [31,32], are also applied. However, many of these studies do not take energy efficiency into account.

The objective of this study is to find the best energy utilization of a dual-arm robot's workpiece transporting motion. However, most of the research on energy optimization stated above does not account for workpiece transportation motion. Furthermore, rather than dual-arm robots, energy-efficient path planning algorithms are generally used in mobile robots, multi-robot systems, or single-arm robots. For dual-arm robot motion planning, Cohen et al. [5] used a heuristic technique. However, they do not incorporate motion planning optimization in their work and completely disregard energy optimization. In human–robot collaboration, Wang et al. [33] developed a cost function considering the dual-arm robot configuration to determine the most efficient path planning solutions for its human partner. Imajo et al. [34] addressed a neural network for gain tuning for the generation of a robot arm trajectory. Yan et al. [35] suggested a method for holding moving workpieces with large momentum using a dual-arm mobile robot by coordinated motion planning as well as environmental compliance control. One of the most recent studies on energy-efficient robot motion planning with dual-arm configuration, focusing on a pick-up pose controlling strategy for carrying workpieces that works while avoiding collisions, was presented by Nishi and Mori [36]. In their research, the workspace was limited to two dimensions (X and Y coordinates), and motion planning was formulated as a minimization problem in terms of workpiece transportation time and total energy utilization of the robot arm motion. To make further improvements, the research group included the Z-coordinates in the motion planning of the robot's configuration and used GA to optimize PID gain for minimal energy consumption [37]. The limitation of this previous study is that the robot placement is not considered. Our empirical results show that robot configuration has a major influence on the obtained solutions to reduce the robot's energy consumption.

Finally, in this study, the robot configuration and motion planning problems are formulated as a minimization problem. The PSO algorithm is used to solve the minimization problem. To the best of our knowledge, PSO is not used for industrial robots' energy efficiency gain tuning even though we found that this algorithm is applied for mobile types of robots [38,39]. The experiment results suggest that PSO performs consistently well and has stable performance for this problem compared with GA. This is the first study to explore the applications of metaheuristic algorithms on joint robot placement and motion planning problems considering energy efficiency. This study fills a gap in the research on energy-efficient robot optimization problems.

3. Robot's Workpiece Transportation Motion Plan

3.1. Model of Workpiece Transportation Motion

The model of work transportation operation using a dual-arm robot is shown in Figure 1. Each arm is assumed to pick-up workpieces carried by a conveyor belt from an initial robot configuration, $A_1(for\ right\ Arm)$, $A_2(for\ left\ arm)$. The conveyor belt transports the workpieces (W_k, for $k = 1, 2, \ldots\ m$) at a distance of 1.2 m apart. The workpieces are picked up at the B_n positions. The motion of the arms is predefined by the PID controller. GA is used for calculating PID gains to solve the minimization problem of the time to complete and the energy consumption for each arm during its motion so that each arm does not overshoot each desired angle. Furthermore, robot arm motions are planned by assigning motions to each workpiece transportation model that are suitable for the conveyor-belt speed (v (m/s)), derived from the obtained motions.

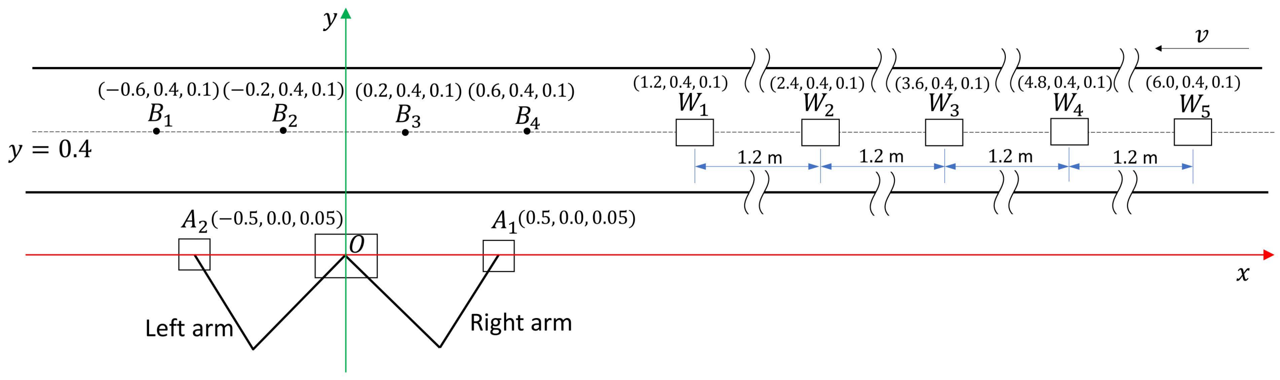

Figure 1. Workpiece transport model using a dual-arm robot.

3.2. Defining Robot Dynamics

Torque PID control law is used to represent the dynamics of each robot arm where the proportional gain is K_{Pi}, the integral gain is K_{Ii}, and differential gain is K_{Di}. The torque at the i-th joint, for $i = 1, 2, \ldots\ n$, can be defined as:

$$\tau_t = K_{Pi}\ (\theta_t - \theta_i) + K_{Ii} \int (\theta_t - \theta_i) dt - K_{Di}\omega_i \tag{1}$$

where

θ_t: Target angle of i-th joint motor (rad)
θ_i: Initial angle of i-th joint motor (rad)
ω_i: Angular velocity of joint (rad/s)
K_{Pi}, K_{Ii}, K_{Di}: Proportional, integral, differential controller gain of i-th joint motor

To identify the PID controller gain, the acting forces on the classical robot kinematic motion equation for the configuration of Figure 2 for joint $I = 1$ can be defined as follows.

The torque exerted on the first joint τ_1 is calculated by

$$\tau_1 = \tau_{m1} + F_1 L_1 \sin\theta_2 + F_2 L_1 \sin\theta_2 + F_3 L_1 \sin\theta_2 - \tau_{f1} \tag{2}$$

$$F_1 = \frac{m_2 L_2 {\omega_2}^2}{2} \tag{3}$$

$$F_2 = m_3 L_2 {\omega_2}^2 \tag{4}$$

$$F_3 = m_W L_2 {\omega_2}^2 \tag{5}$$

where

F_1, F_2, F_3: Forces on the joints and links
τ_{m1}: Motor torque of the first joint
τ_{f1}: Torque caused by frictional force exerted on the first joint
m_2, m_3: Weight of second and third link, respectively
L_1, L_2: First and second joint link length
m_w: Mass of workpiece
θ_2: Joint angle at $i = 1$

ω_2: Angular velocity at $i = 2$

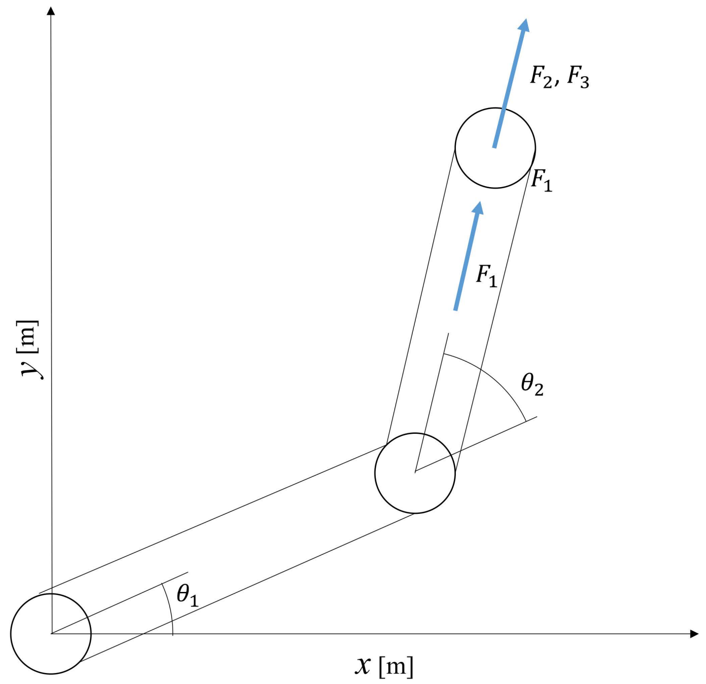

Figure 2. Forces exerted on the right manipulator.

The torque caused by frictional force exerted on each joint τ_{fi} is written as

$$\tau_{fi} = \tau_{vi} + \tau_{si} + \tau_{ci} + \tau_{di} \tag{6}$$

where

τ_{vi}: viscous friction torque at *i*-th joint
τ_{si}: static friction torque at *i*-th joint
τ_{ci}: coulomb friction torque at *i*-th joint
τ_{di}: rag friction torque at *i*-th joint

Figure 3 depicts the active forces (F_4 to F_7) on the second joint ($i = 2$). The torque for the second joint τ_2 is stated by the following equation:

$$\tau_2 = \tau_{m2} - \frac{F_4 L_2}{2}\sin\left(\theta_1 + \theta_2 - \tan^{-1}\frac{y_{g2}}{x_{g2}}\right) - (F_5 + F_6)L_2\sin\left(\theta_1 + \theta_2 - \tan^{-1}\frac{y_{g3}}{x_{g3}}\right) - \frac{F_7 L_2}{2}\cos\left(\theta_1 + \theta_2 - \tan^{-1}\frac{y_{g2}}{x_{g2}}\right) - \tau_{f2} \tag{7}$$

where

(x_{g2}, y_{g2}): center of gravity position for second link
(x_{g3}, y_{g3}): center of gravity position for third link
τ_{m2}: joint torque at $i = 2$
τ_{f2}: frictional torque at $i = 2$
θ_1: joint angle at $i = 1$

The center of gravity (x_{g2}, y_{g2}) for the robot arm's second link can be expressed as

$$x_{g2} = L_1\cos\theta_1 + \frac{L_2}{2}\cos(\theta_1 + \theta_2) \tag{8}$$

$$y_{g2} = L_1\sin\theta_1 + \frac{L_2}{2}\sin(\theta_1 + \theta_2) \tag{9}$$

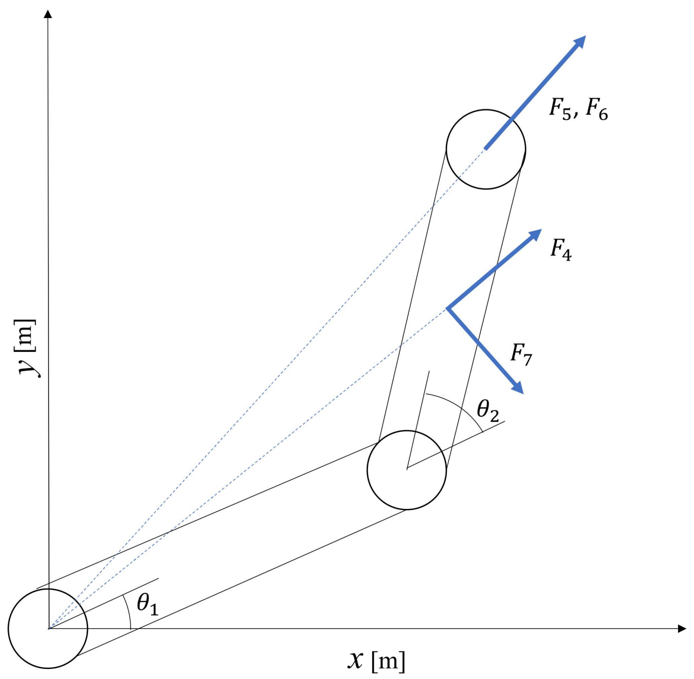

Figure 3. Force exerted on the right arm of the dual arm robot.

The centrifugal force (F_4) of the robot arm's second link caused by the angle θ_1, with an angular velocity of ω_1 derived as

$$F_4 = m_2\sqrt{{x_{g2}}^2 + {y_{g2}}^2}{\omega_1}^2 \tag{10}$$

Similarly, the centrifugal force, F5 and F6 derived as:

$$F_5 = m_3\sqrt{{x_{g3}}^2 + {y_{g3}}^2}{\omega_1}^2 \tag{11}$$

$$F_6 = m_W\sqrt{{x_{g3}}^2 + {y_{g3}}^2}{\omega_1}^2 \tag{12}$$

The force acting on the second link (F_7) due to the motion trajectory of the first link can be calculated as:

$$F_7 = m_2\sqrt{{x_{g2}}^2 + {y_{g2}}^2}\alpha_1 \tag{13}$$

The coordinate for the center of gravity for the third link is (x_{g3}, y_{g3}) and obtained as:

$$x_{g3} = L_1\cos\theta_1 + L_2\cos(\theta_1 + \theta_2) \tag{14}$$

$$y_{g3} = L_1\sin\theta_1 + L_2\sin(\theta_1 + \theta_2) \tag{15}$$

In contrast to the first and second joints, the third joint of the dual armed manipulator operates in a vertical orientation. It is tough to express such orientation as an angle because of this configuration. As a result, in this experiment, we infer that a piece of virtual gear at the end of the second link moves the third link. The third joint's angle is treated as the gear's rotation angle. The third link's movement can thus be treated as an angle, much like the first and second joints. Figure 4 depicts a virtual gear setup for the third link. Assuming r_3 is the radius of the virtual gear and l_3 is the displacement length of the third link, then the third joint (i = 3) torque is calculated as:

$$\tau_3 = \tau_{m3} - 0.05(m_3 + m_w)g - \tau_{f3} \tag{16}$$

where

$$\theta_3 = \frac{l_3}{r_3}. \tag{17}$$

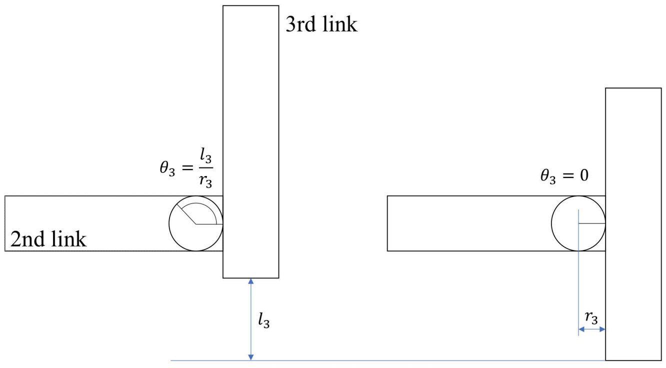

Figure 4. The third link of the robot.

From the classical equation of rotational motion $\tau_i = I_i\alpha_i$ for the moment of inertia I_i, at joint i with angular acceleration of α_i. If the angular velocity of the rotational motion changes from ω_{0i} to ω_i in time interval of Δt, then angular acceleration is defined as follows:

$$\frac{\omega_i - \omega_{0i}}{\Delta t} = \alpha_i \tag{18}$$

Furthermore, for change in angle from an initial value of θ_{0i} to a final value of θ_i

$$\frac{\theta_i - \theta_{0i}}{\Delta t} = \omega_i \tag{19}$$

The i-th joint angle of the robot arm is represented by the following equations.

$$\omega_i = \frac{\tau_i}{I_i}\Delta t + \omega_{0i} \tag{20}$$

$$\theta_i = \frac{\tau_i}{I_i}\Delta t^2 + \omega_{0i}\Delta t + \theta_{0i} \tag{21}$$

A robot arm's joint parameters, such as command angle, joint velocity, payload, and so on, must all be defined within a certain range. If the range is crossed, the maximum and minimum values must be restored. Under the above model description, we can simulate detailed kinematic dynamics of the dual armed robot arm evaluating energy consumption. The PID control simulation is run from the initial state, computing the motion trajectory at each sufficiently small-time interval Δt, until the simulation's termination states are met.

The termination states for the simulation are described as follows.

State 1. During pick-up, the distances between the robot's end-effector and the workpieces are sufficiently small (within 0.01 m).

State 2. The movements of each joint are stopped (the absolute values of angular velocity and acceleration of each joint are within 0.1 rad/s and within 0.1 rad/s^2, respectively).

State 3. The value of the evaluation function reaches a threshold, which is set in advance.

State 1 comes from the arm robot simulation setting of Kawasaki Heavy Industries. The Kawasaki Heavy Industries simulation software, K-ROSET, does not work well when the error tolerance is set as less than 0.01 m. In a unified way, the PID simulation was also conducted.

State 2 is set because the next sequence of the robot arm motion is assumed to start with zero acceleration and velocity. Therefore, the final state of the arm should be zero in acceleration and velocity.

State 3 is necessary because the PID gains randomly generated by the simulation can take zero or negative values. If all the gains are set to zero, then the robot arms do not move and never reach a goal. Furthermore, the evaluation function stated in the following Equation (22) indicates that time t value will increase even during robot arm resting, so this state should be added to terminate the simulation.

The flowchart in Figure 5 defines the termination states for the PID simulation. For the calculation of individual gains, GA and PSO algorithms are used. Section 3.3 describes the optimization techniques of PID gain tuning using GA. Moreover, Section 4.2 defines the objective function for individual gains using PSO. At the termination of the simulation, the objective function's performance is assessed. For robot arm motions that satisfy state 3, the value of the objective function is defined to an adequately larger range irrespective of the significance of the objective function at the termination of the simulation.

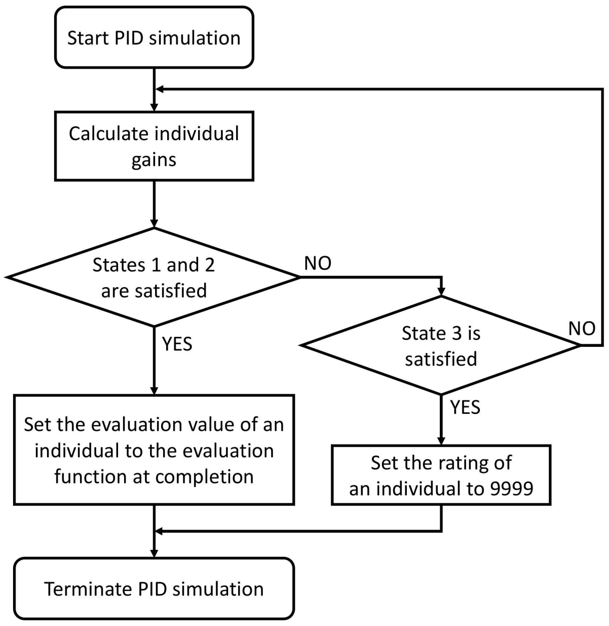
Start PID simulation
Calculate individual gains
States 1 and 2 are satisfied
NO
YES
State 3 is satisfied
NO
YES
Set the evaluation value of an individual to the evaluation function at completion
Set the rating of an individual to 9999
Terminate PID simulation

Figure 5. PID simulation flowchart.

3.3. Genetic Algorithm for Optimization of PID Gain Tuning

The GA method is widely used as an evolutionary computation technique simulating the process of evolution. It takes much computational effort to find the optimal solution for solving a complex optimization problem, including physical simulations. The enumerative search from the vast number of combinations is unrealistic. Therefore, this study aims to find a local optimal solution instead of a global optimal solution considering the computation time. Our designed GA for finding an energy-efficient operation is described below. The objective is to find a better solution by repeating the search a predetermined number of times. The objective function is represented by Equation (22).

$$\min_{\{\tau_{mi}\}} (t + aN + bM + cE + dP) \tag{22}$$

where a, b, c, d are the weighting factors in the objective function. t is the total completion time of pick-and-place motion, N represents the number of overshoots. M represents time in seconds when the current angle deviates from the target angle. P represents the penalty for violating conflicts between the trajectory of the other arm. Those terms are obtained in the simulation. The calculation method of energy consumption E is from reference [40] in this study. Although, in our previous study [36], the energy consumption was obtained from the acceleration of the hand position of the robot arm.

$$E_i = \int \left\{ \tau_{mi}\omega_i + R_i \left(\frac{\tau_{mi}}{k_{ti}} \right)^2 \right\} dt \tag{23}$$

E_i: Energy consumed by the i-th joint
R_i: Terminal resistance of the i-th joint
k_{ti}: Torque constant of the i-th joint

Joints 1 and 2 have a maximum energy consumption of 80 W, while joint 3 has a maximum energy consumption of 50 W [41]. When the integral calculation of Equation (21) is conducted in the range of $t = [0, 1]$, the terminal resistance and torque constants of the first and the second joints can be evaluated by Equation (24) assuming that the motor torque and angular velocity are at their maximum values.

$$\frac{R_i}{{k_{ti}}^2} = \frac{80 - 15\pi}{900} \tag{24}$$

where R_i is the terminal resistance and k_{ti} is the torque constant. Similarly, for the third joint:

$$\frac{R_3}{{k_{t3}}^2} = \frac{50 - 7.5\pi}{225} \tag{25}$$

Therefore, energy consumption equation for each robot arm joint is defined as:

$$E = \sum_{i=1}^{3} E_i \tag{26}$$

The steps in the proposed GA for the PID gain optimization technique are as follows.

3.3.1. Initialization

The number of individuals and parameters are set and the individuals for decision variables are encoded by binary encoding of real values. We have set the population sizes of GA to 200, 300, and 400, and compared the performance of these GA variants. The experimental results suggest there were not many differences in the final solutions. We set the number of individuals as 300 because 200 individuals did not work well in the preliminary experiments.

3.3.2. Generation of Initial Individuals

Initial individuals are generated by using random numbers on uniform distribution. The proportional gains K_{P1}, K_{P2}, K_{P3}, differential gains K_{D1}, K_{D2}, K_{D3}, and integral gains K_{I3} are initialized by random numbers in the range $[-5, 30]$. The individuals' generations are repeated for the number of individuals.

3.3.3. Evaluation

The value of the objective function is obtained for each individual included in the generation by executing the PID simulation, which is explained in the previous section.

3.3.4. Selection

Selection of the next iteration is conducted by the following idea. A set of individuals are randomly selected from the current generation and their fitness values are used to select the individuals to be left for the next generation. The probability P_i for individual i is calculated by Equation (27).

$$P_i = \frac{\frac{1}{O_i}}{\sum_{j=1}^{3} \frac{1}{O_j}} \tag{27}$$

Each selected individual is assigned index j (j = 1, 2, 3) for each joint, and the evaluation value of that individual is O_j, and the probability of being left to the next generation is P_i. An elite preserving strategy is also adopted in the selection process. The top three individuals are elite, and they are not subject to crossover and mutation.

3.3.5. Crossing Over

In the crossing over, two individuals are chosen from the two groups, which are divided, and a random number is generated in the range of [0, 1], and the crossover is performed when it is smaller than the set crossover rate. A random number with an integer value in the range of $\left[1,\ 2^6 - 1\right]$ is generated to determine the genes to be exchanged between the two individuals. If the remainder of 2 of the random number value is 1, $K_{P1},\ K_{P2}$ exchange is performed extensively, otherwise K_{P3}, $K_{D1},\ K_{D2},\ K_{D3}$, and K_{I3} exchange is conducted when the remainder of 2, 4, 8, 16, 32 for the random value is an odd number. The gene in the digit with the binary value of 1 is exchanged into 0.

3.3.6. Mutation

Random numbers are generated in the range of [0, 1] for each next-generation individual, and mutation is performed when the mutation rate is smaller than the set mutation rate. When mutations occur, random values are entered in $K_{P1},\ K_{P2},\ K_{P3},\ K_{D1},\ K_{D2}, K_{D3},\ K_{I3}$. However, if the specified number of generations is exceeded, a value close to the near-optimal solution is generated at that point.

3.3.7. Update Generation

After updating the population of individuals, Steps 2 to 6 are repeated for a specified number of generations.

The setting of initial solutions does not depend on the selection of initial solutions because the set of initial solutions are generated from random numbers. The GA iterations are repeated 10 times in the numerical experiments. Therefore, there is little effect on the selection of the initial solution and the selected parameters from preliminary experiments.

When finding a solution by a genetic algorithm, different behaviors can be obtained from the same algorithm by changing the weighting factors of the objective function. If the value of the energy consumption weight is made small, time-oriented operations can be obtained, and if it is made large, energy-saving-oriented operations can be obtained. In Section 5, numerical experiments are performed based on the simulation and genetic algorithms.

4. Optimal Robot Placement and Motion Planning

4.1. Problem Statement

In the previous section, GA was used to optimize the robot motion planning problem. Evidence from our experimental results suggests that the robot placement plays a crucial role in the energy-efficient robot optimization problems. Changing the robot placement may have a major influence on the optimal results. Therefore, by combining the robot placement problem and the motion planning problem, it will be possible to find better solutions.

The aim of this section is to optimize the joint robot placement and motion planning problem by metaheuristic algorithms. The objective function of the robot placement and motion planning problem is shown in Equation (22). In addition, x and y represent the robot placement variables. The optimization algorithm simultaneously searches the optimal robot motion and the robot placement. The robot can be moved in a two-dimensional space, as shown in Figure 6.

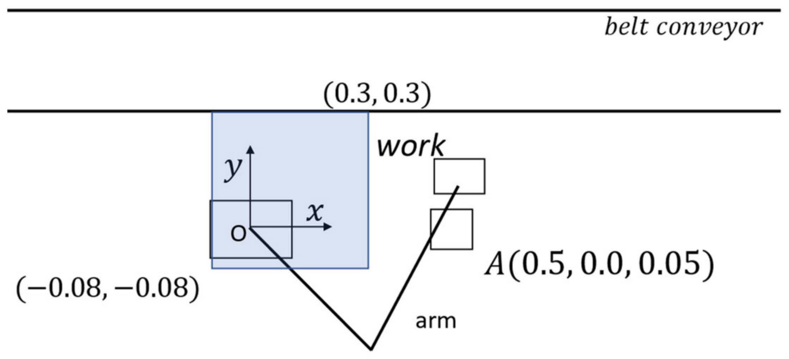

Figure 6. Optimal robot placement and motion planning problem.

4.2. Particle Swarm Optimization

Particle swarm optimization (PSO) is a population-based metaheuristic algorithm proposed by Kennedy and Eberhart [42]. The PSO algorithm is one of the most widely used metaheuristics because of its good performance for many optimization problems [43,44]. The PSO algorithm searches the solution space by changing the positions of particles. The positions of particles represent the possible solutions for an optimization problem. To date, PSO has been successfully applied to various real-world problems. However, the present research explores, for the first time, the usefulness of PSO in the robot placement and motion planning problem.

In this study, the PSO algorithm is used to solve the robot placement and motion planning problem while considering the energy-efficiency. For this problem, the position of a particle consists of nine variables that are proportional gains $K_{P1},\ K_{P2},\ K_{P3}$, differential gains $K_{D1},\ K_{D2},\ K_{D3}$, integral gains K_{I3}, and robot placement variables x and y. These variables can be represented by a vector $p_i = \{K_{P1},\ K_{P2},\ K_{P3},\ K_{D1},\ K_{D2},\ K_{D3}, K_{I3}, x, y\}$. Let $p_{i,t}$ be the position of particle i at generation t, where i is the index of particles ($i = 1, 2, \ldots, NP$), and NP is the population size.

During the computation process, the first seven parameters are set within the range $[-5, 30]$, and the robot placement variables take values in the interval of $[-0.08, 0.3]$. Therefore, the following ranges are set for $p_{i,t}$: $p_{\min} = (-5, -5, -5, -5, -5, -5, -5, -0.08, -0.08)$, and $p_{\max} = (30, 30, 30, 30, 30, 30, 30, 0.3, 0.3)$.

The first step in PSO is initialization. The initial population $p_{i,0}$ is created by the following equation:

$$p_{i,0} = p_{\min} + \text{rand}(p_{\max} - p_{\min}) \tag{28}$$

where rand is a random vector that is uniformly distributed in the range $[0, 1]^9$.

After the initialization, the objective value of $p_{i,0}$ $(i = 1, 2, \ldots, NP)$ is evaluated. Then, the best solution in the swarm g^*, and the personal best solution of particle i, p_i^* are generated. For $i = 1, 2, \ldots, NP$, let $g^* = \text{argmin} f(p_{i,0})$, and $p_i^* = p_{i,0}$.

At each generation, the velocity v_i and position p_i of ith particle are updated as follows:

$$v_{i,t+1} = \omega v_{i,t} + c_1 r_1 (g^* - p_{i,t}) + c_2 r_2 (p_i^* - p_{i,t}) \tag{29}$$

$$p_{i,t+1} = p_{i,t} + v_{i,t+1} \tag{30}$$

where ω is the inertia factor that takes values within the range $[0, 1]$. The parameter c_1 and c_2 are the accelerate coefficients. These two parameters often take values around 2. Two random vectors r_1 and r_2 take values in the interval of $[0, 1]$.

The position $p_{i,t}$ should always be in the bounds of $p_{\min}$ and $p_{\max}$. To keep the variables in this range, the feasibility of the generated position $p_{i,t+1}$ is examined. A variable is set to the most closed bound value when it is larger than $p_{\max}$ or lower than $p_{\min}$. This repair strategy is as follows:

$$p_{i,t+1} = \min(p_{\max},\ p_{i,t+1}) \tag{31}$$

$$p_{i,t+1} = \max(p_{\min},\ p_{i,t+1}) \tag{32}$$

The same repair strategy is used to the velocity $v_{i,t+1}$. However, the ranges of velocity are set as

$$v_{\max} = 0.5(p_{\max} - p_{\min}) \tag{33}$$

$$v_{\min} = -v_{\max} \tag{34}$$

At the end of each generation, the personal best solution p_i^* is updated if the new position $p_{i,t+1}$ has better fitness than it. Moreover, after updating the positions for all the particles, if the fitness of the best $p_{i,t+1}$ is better than $f(g^*)$ for $i = 1, 2, \ldots, NP$, g^* is updated to p_i^*.

The pseudo code of PSO is shown in Algorithm 1:

Algorithm 1. Particle swarm algorithm

```
Initialize positions $p_{i,0}$, $i = 1, 2, \ldots, NP$ //Equation (28)
Set $t = 0$
Evaluate the objective values $f(p_{i,0})$, $i = 1, 2, \ldots, NP$
Set $f(g^*) = \min\{f(p_{i,0})\}$, $g^* = \text{argmin} f(p_{i,0})$, $i = 1, 2, \ldots, NP$
Set $p_i^* = p_{i,0}$, $i = 1, 2, \ldots, NP$
while (stop condition is not reached) do:
  for $i = 1 : NP$
    Generate $v_{i,t+1}$ and $p_{i,t+1}$ //Equations (29) and (30)
    Repair $v_{i,t+1}$ and $p_{i,t+1}$ if it is not feasible //Equations (31) and (32)
    Evaluate the objective value $f(p_{i,t+1})$
if $f(p_{i,t+1}) < f(p_i^*)$
      $p_i^* = p_{i,t+1}$, $f(p_i^*) = f(p_{i,t+1})$
end if
  end for
if $\min(f(p_i^*)) < f(g^*)$, $i = 1, 2, \ldots, NP$
    $g^* = p_i^*$, $f(g^*) = \min(f(p_i^*))$
end if
  $t = t + 1$
end while
```

PSO has four important parameters, including the inertia factor, two accelerate coefficients and the population size. This study uses a common parameter setting for the first three parameters: $\omega = 0.5$, $c_1 = 2$, and $c_2 = 2$. It is a well-known fact that the population size has a crucial impact on the performance of metaheuristic algorithms [45]. In the next section, we examine the performance of six PSO variants under different population sizes that range from 20 up to 200 particles.

5. Results

5.1. Robot Motion Planning

5.1.1. Experimental Conditions

We consider a pick-and-place motion with cyclic operations from an initial position to a workpiece acquisition point. The values of the weighting factors a and b are fixed at 20, and the value for c is varied to 0, 0.1, 1, and 10, to normalize the values of each term of the objective function. There are 32 types of motions obtained, and these motions are assigned in such a way that the total energy consumption for the conveyor-belt speed is reduced.

5.1.2. Experimental Results

Table 1 illustrates the average total execution time and total energy consumption during a pick-and-place operation for various analytical weights for the GA method. The robot operation duration grows steadily as the computational weight increases, while the energy consumption reduces. In comparison to the case in which the weighting factors changed from 0 to 10, the total completion time increased by 1.7 times, while the energy consumption can be lowered by 0.2 times. By merging these algorithms to the robot motion plan for different robot configurations, the total energy consumption for the pick-and-place operation can be reduced.

Table 1. Computational results for the GA method under various weights.

Computational Weight for GA	Average Execution Time (s)	Total Energy Consumption, *E* (W)
0	320,125	525,625
0.1	390,375	2,174,625
1	51,575	16,405
10	54,775	1,143,625

Figure 7 depicts the changes in total energy consumption for single-arm and dual-arm robots when the conveyor-belt speed was varied during the experiment. According to the numerical study of right-arm motion, one cycle of pick-and-place operation takes about 5.00 s to complete. As a result, the conveyor-belt speed should be set below 0.24 m/s to pick up the workpieces when the spacing between them was 1.2 m. In comparison to using only the right arm, using dual-arms with increasing conveyor-belt speed results in lower total energy consumption. However, as the conveyor-belt speed decreased, the difference in total energy consumption shrank. When the conveyor-belt speed is below 0.17 m/s, the energy consumption with only the right arm is lower than the dual-arm operation. The most likely reason for this is that the conveyor belt's speed is low enough that the right arm can choose an action that requires less energy, whereas the dual-arm action uses more energy to maintain the arm's posture.

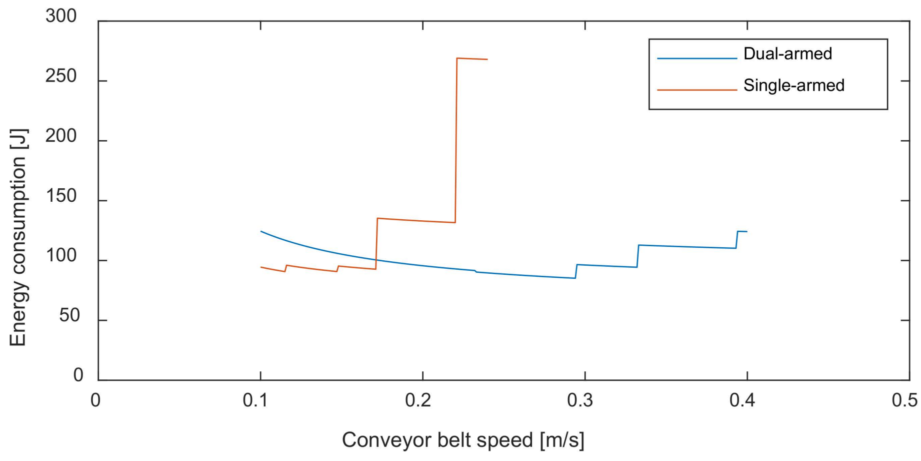

Figure 7. Total energy consumption for single and dual-arm robot pick-and-place operation with different conveyor-belt speeds.

5.2. Robot Placement and Motion Planning

5.2.1. Illustrative Example

An empirical study is conducted to investigate the impacts of different robot placements on the energy consumption and completion time of a robot. The initial robot placements are set to the robot's initial platform position as the origin and from the coordinates B_4 on the conveyor belt to the arm's initial position A_1. The computational results with different robot configurations are presented in Table 2.

Table 2. Computational results with different robot configurations.

Robot Configurations (x [**m**], y [**m**])	Total Execution Time t (s)	Total Energy Consumption E (W)
(−0.053, −0.004)	3.685	8.841
(0.148, −0.080)	5.085	15.422
(0.140, −0.042)	5.185	11.830
(0.082, −0.021)	5.752	16.116
(0.131, −0.008)	6.318	15.670

The results from Table 2 suggest that the robot configurations have a major impact on the energy consumption and completion time of a robot. In detail, the robot placement (−0.053, −0.004) has the lowest energy consumption at 8.841, while the robot placement (0.082, −0.021) has the highest energy consumption at 16.116. Thus, setting the robot placement to different values may result in obtaining better results. However, to do this, many trial-and-error computations are required, and the computations are time-consuming. Therefore, in the next section, GA and PSO are applied to solve the robot configuration and motion planning problem.

Overall, the findings clearly indicate the possibility of obtaining better solutions by considering the robot placements as decision variables.

5.2.2. Comparison Results

In this study, the joint robot placement and motion planning problem is solved by two metaheuristic algorithms, the GA algorithm and the PSO algorithm. The maximum number of fitness evaluations $MaxFEs$ is set to 5000. All the algorithms are coded using Python and executed on a computer with AMD Ryzen 9 3900X CPU (3.8 GHz), 16 GB RAM and a Windows 10 operating system.

The parameter setting may have a major influence on the performance of metaheuristic algorithms. To find the appropriate parameter setting of GA and PSO for the robot placement and motion planning problem, the performance of GA and PSO under different parameter settings are investigated. The population sizes of GA and PSO are set so that there would be approximately 2000 simulations in each condition. GA and PSO run 10 times with each parameter setting.

The parameter settings for GA and PSO are shown in Table 3, where *ind* represents the number of individuals, *gen* the number of generations, *mutation* the mutation rate, and *cross* the crossover rate for GA described in Section 3.3, while ω represents the inertia factor, c_1 and c_2 the accelerate coefficients, and NP the population size for PSO described in Section 4.2.

Table 3. The parameter settings for GA and PSO.

Algorithm	Parameter Settings
GA1	*ind* = 100, *gen* = 30, *mutation* = 0.3, *cross* = 0.5
GA2	*ind* = 100, *gen* = 30, *mutation* = 0.4, *cross* = 0.4
GA3	*ind* = 100, *gen* = 30, *mutation* = 0.5, *cross* = 0.3
GA4	*ind* = 200, *gen* = 20, *mutation* = 0.1, *cross* = 0.4
GA5	*ind* = 200, *gen* = 50, *mutation* = 0.1, *cross* = 0.1
GA6	*ind* = 300, *gen* = 10, *mutation* = 0.4, *cross* = 0.4
GA7	*ind* = 300, *gen* = 30, *mutation* = 0.1, *cross* = 0.1

Table 3. *Cont.*

Algorithm	Parameter Settings
PSO1	$\omega = 0.5, c_1 = 2, c_2 = 2,\ NP = 20$
PSO2	$\omega = 0.5, c_1 = 2, c_2 = 2,\ NP = 40$
PSO3	$\omega = 0.5, c_1 = 2, c_2 = 2,\ NP = 60$
PSO4	$\omega = 0.5, c_1 = 2, c_2 = 2,\ NP = 80$
PSO5	$\omega = 0.5, c_1 = 2, c_2 = 2,\ NP = 100$
PSO6	$\omega = 0.5, c_1 = 2, c_2 = 2,\ NP = 200$

Figure 8 and Tables 4 and 5 show the calculation results of the objective function (22), which is the weighted sum of total completion time, number of overshoots, time in seconds when the current angle deviates from the target angle, penalty, and energy consumption. Boxplots of the objective values are obtained by GA and PSO for solving the robot placement and motion planning problem are shown in Figure 8. The interquartile ranges of the PSO algorithms are smaller than GA, meaning that PSO has robust performance. Furthermore, the upper quartiles of the PSO algorithms are lower than GA, which indicates that PSO has better performance.

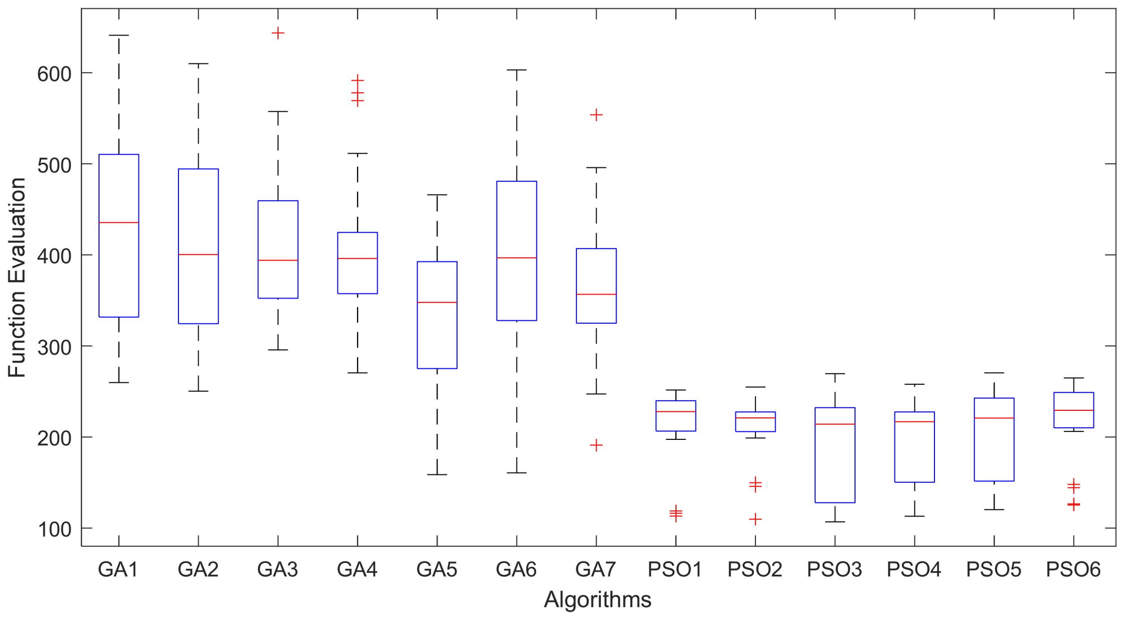

Figure 8. Boxplots of GA and PSO for solving the robot placement and motion planning problem.

Table 4. The best, worst, and mean results of GA.

	GA1	GA2	GA3	GA4	GA5	GA6	GA7
Best	6,411,957	6,101,113	6,437,138	5,914,767	4,660,399	6,031,033	5,539,134
Worst	2,598,877	2,503,144	2,956,093	2,704,759	1,587,137	1,606,871	1,910,518
Mean	4,287,444	4,089,252	410,142	4,121,686	3,311,333	3,944,367	366,246

Table 5. The best, worst, and mean results of PSO.

	PSO1	PSO2	PSO3	PSO4	PSO5	PSO6
Best	2,517,318	2,549,859	2,695,012	2,579,943	2,704,407	2,649,485
Worst	1,131,703	1,098,057	1,068,871	1,130,499	1,202,886	1,254,347
Mean	2,120,034	210,798	1,938,292	1,984,438	2,067,629	2,169,012

The best, worst, mean results of GA and PSO are shown in Tables 4 and 5, respectively. Furthermore, the worst results obtained by PSO are still better than the best results by GA3 and GA4. Overall, PSO performs consistently well and is stable for this problem compared with GA. As shown in Table 5, PSO3 provides the best mean result for the robot placement and motion planning problem.

To further evaluate the results obtained by PSO3, we investigated this result under different weight settings, as shown in Table 6, and compared it with the results obtained by GA in Table 7. The comparison results indicate that our proposed method can save 16% implementation time and 18% energy on average.

Table 6. Execution time and energy consumption applying the PSO3 algorithm.

Computational Weight	Robot Configuration $(x\ [\mathbf{m}], y\ [\mathbf{m}])$	Average Execution Time t (s)	Total Energy Consumption E (W)
10	$(-0.053, -0.004)$	3685	8841
1	$(-0.053, -0.004)$	2852	10,031
0.1	$(-0.053, -0.004)$	267	16,101
0	$(-0.053, -0.004)$	2502	32,730

Table 7. Execution time and energy consumption applying the GA algorithm.

Computational Weight	Average Execution Time t (s)	Total Energy Consumption E (W)
10	3.89	14.83
1	4.22	14.76
0.1	3.07	16.24
0	2.77	36.75

5.3. Simulation and Implementation

The pick-and-place operation of a dual-arm robot, named duAro, is used to pick workpieces with the conveyor-belt speed of v = 0.2 m/s, as shown in Figure 9. To check the motion trajectory and simulate, Kawasaki's offline robot simulation program K-ROSET is employed. After verifying that the generated motion trajectory is likely workable on the actual robot using K-ROSET, the proposed method is implemented on the actual robot duAro. Figure 10 illustrates the actual motion trajectory for the duAro robot and the simulated trajectory.

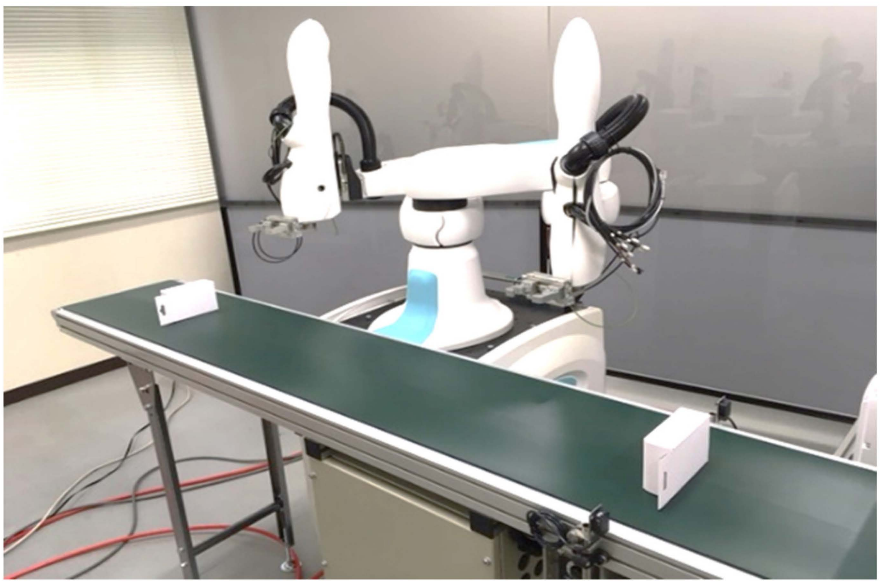

Figure 9. Experimental dual-arm robot system in pick-and-place operation.

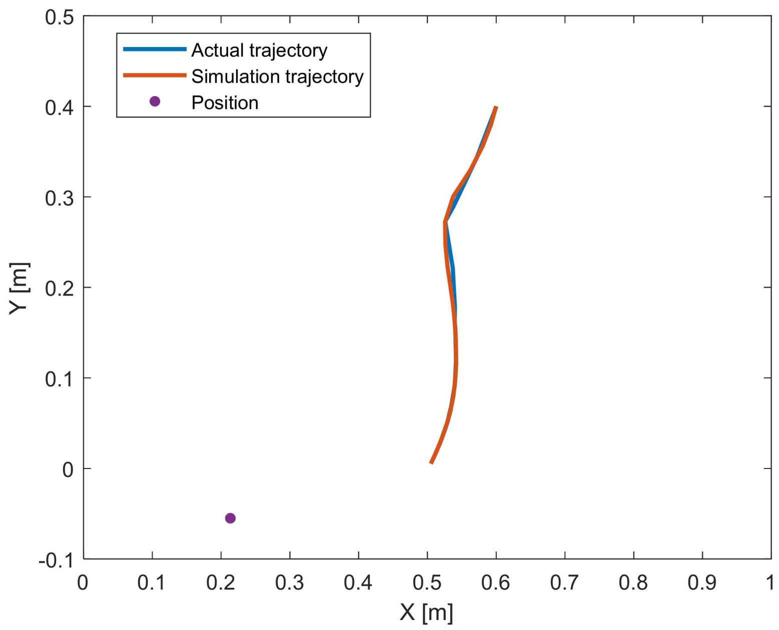

Figure 10. Trajectories of the proposed method implemented in the PID simulation and the actual robot duAro. Position indicates the origin of the robot arm (the first joint).

As seen in Figure 10, there is a slight difference of up to around 7 mm between the simulation trajectory and the actual trajectory; however, the actual robot duAro was almost able to achieve the desired trajectory. The slight difference is caused by the actual friction force and the discretization error of the trajectory.

There are also some limitations to our proposed method:

(1) The proposed method is only applied to a SCARA robot.
(2) The workpieces are assumed to be at the center of the conveyor belt.
(3) Each workpiece is set at a constant interval.
(4) The conveyor-belt speed is constant.

A vision system will be necessary to detect and grasp roughly set (dynamic) workpieces on the conveyor belt.

Therefore, the future work will be to address those issues above and to apply our proposed method to more complicated scenarios.

6. Conclusions

This study set out to develop a model for the joint robot placement and motion planning problem and evaluate the performance of two metaheuristic algorithms on this problem. First, the robot motion planning problem is formulated, and the optimized PID gains are obtained by GA. The objective is to minimize the total execution time and energy utilization of a dual-arm robot for the workpiece transportation process. Then, the PSO algorithm is applied to optimize the robot configuration as well as motion simultaneously. Finally, the proposed approach is implemented in a real-world dual-arm robot duAro. The simulation and implementation results suggest that our proposed motion planning approach is effective, and it can be verified on the K-ROSET simulation environment. The findings of the experiments reveal that dual-arm motions can save more energy than single-arm motions in terms of energy efficiency. Furthermore, when compared to motion planning with PID controllers alone, combining the robot configuration problem with the metaheuristic method can save 18% of energy usage and 16% of execution time. The future work will be to apply our proposed method to more complicated scenarios, including other types of industrial robots and dynamically set workpieces.

Author Contributions: Conceptualization, K.N. and T.N.; methodology, K.N. and T.N.; software, K.N. and Z.L.; validation, K.N., T.F. and T.N.; formal analysis, K.N.; investigation, K.N.; resources, T.N.; data curation, K.N.; writing—original draft preparation, K.N., Z.L., T.F., M.M.A. and T.N.; writing—review and editing, K.N., Z.L., T.F., M.M.A. and T.N.; visualization, K.N.; supervision, T.N.; project administration, T.N.; funding acquisition, T.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research is subsidized by New Energy and Industrial Technology Development Organization (NEDO) under a project JPNP20016.

Acknowledgments: This research is subsidized by New Energy and Industrial Technology Development Organization (NEDO) under a project JPNP20016. This paper is one of the achievements of join research with ROBOT Industrial Basic Technology Collaborative Innovation Partnership and the ROBOT Industrial Basic Technology Collaborative Innovation Partnership a joint copyright of this paper.

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

References

1. Sustainable Development Goals (SDG 7). Available online: https://unric.org/en/sdg-7/ (accessed on 14 December 2021).
2. Conti, J.; Holtberg, P.; Diefenderfer, J.; LaRose, A.; Turnure, J.T.; Westfall, L. *International Energy Outlook 2016 with Projections to 2040*; DOE/EIA-0484(2016); USDOE Energy Information Administration: Washington, DC, USA, 2016; p. 1296780.
3. Madridano, Á.; Al-Kaff, A.; Martín, D.; De la Escalera, A. Trajectory Planning for Multi-Robot Systems: Methods and Applications. *Expert Syst. Appl.* **2021**, *173*, 114660. [CrossRef]
4. Fong, S.; Deb, S.; Chaudhary, A. A Review of Metaheuristics in Robotics. *Comput. Electr. Eng.* **2015**, *43*, 278–291. [CrossRef]
5. Cohen, B.; Chitta, S.; Likhachev, M. Single- and Dual-Arm Motion Planning with Heuristic Search. *Int. J. Robot. Res.* **2014**, *33*, 305–320. [CrossRef]
6. Zaplana, I.; Cepolina, E.; Faieta, F.; Lucia, O.; Gagliardi, R.; Baizid, K.; D'Imperio, M.; Cannella, F. A Novel Strategy for Balancing the Workload of Industrial Lines Based on a Genetic Algorithm. In Proceedings of the 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 8–11 September 2020; pp. 785–792.
7. Zhang, H.; Zha, W.; Xu, X.; Zhu, Y. Optimization Control of Cooperative Trajectory towards Dual Arms Based on Time-Varying Constrained Output State. *Complexity* **2021**, *2021*, 5338134. [CrossRef]
8. Liu, A.; Liu, H.; Yao, B.; Xu, W.; Yang, M. Energy Consumption Modeling of Industrial Robot Based on Simulated Power Data and Parameter Identification. *Adv. Mech. Eng.* **2018**, *10*, 1687814018773852. [CrossRef]
9. Paes, K.; Dewulf, W.; Elst, K.V.; Kellens, K.; Slaets, P. Energy Efficient Trajectories for an Industrial ABB Robot. *Procedia CIRP* **2014**, *15*, 105–110. [CrossRef]
10. Paryanto; Brossog, M.; Kohl, J.; Merhof, J.; Spreng, S.; Franke, J. Energy Consumption and Dynamic Behavior Analysis of a Six-Axis Industrial Robot in an Assembly System. *Procedia CIRP* **2014**, *23*, 131–136. [CrossRef]
11. Gadaleta, M.; Berselli, G.; Pellicciari, M. Energy-Optimal Layout Design of Robotic Work Cells: Potential Assessment on an Industrial Case Study. *Robot. Comput.-Integr. Manuf.* **2017**, *47*, 102–111. [CrossRef]
12. Gadaleta, M.; Pellicciari, M.; Berselli, G. Optimization of the Energy Consumption of Industrial Robots for Automatic Code Generation. *Robot. Comput.-Integr. Manuf.* **2019**, *57*, 452–464. [CrossRef]
13. Osa, T. Multimodal Trajectory Optimization for Motion Planning. *Int. J. Robot. Res.* **2020**, *39*, 983–1001. [CrossRef]
14. Riazi, S.; Wigström, O.; Bengtsson, K.; Lennartson, B. Energy and Peak Power Optimization of Time-Bounded Robot Trajectories. *IEEE Trans. Autom. Sci. Eng.* **2017**, *14*, 646–657. [CrossRef]
15. Pastras, G.; Fysikopoulos, A.; Chryssolouris, G. A Theoretical Investigation on the Potential Energy Savings by Optimization of the Robotic Motion Profiles. *Robot. Comput.-Integr. Manuf.* **2019**, *58*, 55–68. [CrossRef]
16. Paryanto; Hetzner, A.; Brossog, M.; Franke, J. A Dynamic Simulation Model of Industrial Robots for Energy Examination Purpose. *Appl. Mech. Mater.* **2015**, *805*, 223–230. [CrossRef]
17. Pellicciari, M.; Berselli, G.; Leali, F.; Vergnano, A. A Method for Reducing the Energy Consumption of Pick-and-Place Industrial Robots. *Mechatronics* **2013**, *23*, 326–334. [CrossRef]
18. Horla, D.; Cieślak, J. On Obtaining Energy-Optimal Trajectories for Landing of UAVs. *Energies* **2020**, *13*, 2062. [CrossRef]
19. Feng, Y.; Ji, Z.; Gao, Y.; Zheng, H.; Tan, J. An Energy-Saving Optimization Method for Cyclic Pick-and-Place Tasks Based on Flexible Joint Configurations. *Robot. Comput.-Integr. Manuf.* **2021**, *67*, 102037. [CrossRef]
20. Ji, C.; Kong, M.; Li, R. Time-Energy Optimal Trajectory Planning for Variable Stiffness Actuated Robot. *IEEE Access* **2019**, *7*, 14366–14377. [CrossRef]
21. Glorieux, E.; Riazi, S.; Lennartson, B. Productivity/Energy Optimisation of Trajectories and Coordination for Cyclic Multi-Robot Systems. *Robot. Comput.-Integr. Manuf.* **2018**, *49*, 152–161. [CrossRef]
22. Sangiovanni, B.; Incremona, G.P.; Piastra, M.; Ferrara, A. Self-Configuring Robot Path Planning with Obstacle Avoidance via Deep Reinforcement Learning. *IEEE Control Syst. Lett.* **2021**, *5*, 397–402. [CrossRef]

23. Hovgard, M.; Lennartson, B.; Bengtsson, K. Applied Energy Optimization of Multi-Robot Systems through Motion Parameter Tuning. *CIRP J. Manuf. Sci. Technol.* **2021**, *35*, 422–430. [CrossRef]
24. Wang, J.; Liu, J.; Chen, W.; Chi, W.; Meng, M.Q.-H. Robot Path Planning via Neural-Networks-Driven Prediction. *IEEE Trans. Artif. Intell.* **2021**, 1. [CrossRef]
25. Patle, B.K.; Jha, A.; Pandey, A.; Gudadhe, N.; Kashyap, S.K. The Optimized Path for a Mobile Robot Using Fuzzy Decision Function. *Mater. Today Proc.* **2019**, *18*, 3575–3581. [CrossRef]
26. Jang, G.; Cho, S.-B. Optimal Trajectory Path Generation for Jointed Structure of Excavator Using Genetic Algorithm. In Proceedings of the 2019 IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 10–13 June 2019; pp. 1953–1959.
27. Landa-Torres, I.; Manjarres, D.; Bilbao, S. Metaheuristic Algorithm for Optimal Swarm Robotic Parameter Configuration in Time-Variant Plume Detection. In *EngOpt 2018 Proceedings of the 6th International Conference on Engineering Optimization, Lisboa, Portugal, 17–19 September 2018*; Rodrigues, H.C., Herskovits, J., Mota Soares, C.M., Araújo, A.L., Guedes, J.M., Folgado, J.O., Moleiro, F., Madeira, J.F.A., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 959–969, ISBN 978-3-319-97772-0.
28. Cong, Y.Z.; Ponnambalam, S.G. Mobile Robot Path Planning Using Ant Colony Optimization. In Proceedings of the 2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Singapore, 14–17 July 2009; pp. 851–856.
29. Wang, P.; Gao, S.; Li, L.; Sun, B.; Cheng, S. Obstacle Avoidance Path Planning Design for Autonomous Driving Vehicles Based on an Improved Artificial Potential Field Algorithm. *Energies* **2019**, *12*, 2342. [CrossRef]
30. Szczepanski, R.; Bereit, A.; Tarczewski, T. Efficient Local Path Planning Algorithm Using Artificial Potential Field Supported by Augmented Reality. *Energies* **2021**, *14*, 6642. [CrossRef]
31. Quan, Y.; Ouyang, H.; Zhang, C.; Li, S.; Gao, L.-Q. Mobile Robot Dynamic Path Planning Based on Self-Adaptive Harmony Search Algorithm and Morphin Algorithm. *IEEE Access* **2021**, *9*, 102758–102769. [CrossRef]
32. Liu, S.Q.; Kozan, E. A Hybrid Metaheuristic Algorithm to Optimise a Real-World Robotic Cell. *Comput. Oper. Res.* **2017**, *84*, 188–194. [CrossRef]
33. Wang, W.; Chen, Y.; Jia, Y. Evaluation and Optimization of Dual-Arm Robot Path Planning for Human–Robot Collaborative Tasks in Smart Manufacturing Contexts. *ASME Lett. Dyn. Syst. Control* **2020**, *1*, 011012. [CrossRef]
34. Imajo, S.; Konishi, M.; Nishi, T.; Imai, J. Application of a Neural Network to the Generation of a Robot Arm Trajectory. *Artif. Life Robot.* **2005**, *9*, 107–111. [CrossRef]
35. Yan, L.; Yang, Y.; Wenfu, X.; Vijayakumar, S. Dual-arm Coordinnated Motion Planning and Compliance Control for Capturing Moving Objects with Large Momentum. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 2901–2908.
36. Nishi, T.; Mori, Y. Energy Efficient Motion Planning of Dual-Armed Robots with Pickup Point Determination for Transportation Tasks. In Proceedings of the 2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Bangkok, Thailand, 16–19 December 2018; pp. 1401–1405.
37. Nonoyama, K.; Nishi, T. Every-Efficient Motion Planning for Dual-Armed Robot by Pid Gain Optimization with Genetic Algorithm. In Proceedings of the 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), Lyon, France, 23–27 August 2021; pp. 1155–1160.
38. Das, P.K.; Behera, H.S.; Panigrahi, B.K. Intelligent-Based Multi-Robot Path Planning Inspired by Improved Classical Q-Learning and Improved Particle Swarm Optimization with Perturbed Velocity. *Eng. Sci. Technol. Int. J.* **2016**, *19*, 651–669. [CrossRef]
39. Contreras-Cruz, M.A.; Ayala-Ramirez, V.; Hernandez-Belmonte, U.H. Mobile Robot Path Planning Using Artificial Bee Colony and Evolutionary Programming. *Appl. Soft Comput.* **2015**, *30*, 319–328. [CrossRef]
40. Plooij, M.; Wisse, M.A. Novel Spring Mechanism to Reduce Energy Consumption of Robotic Arms. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Algarve, Portugal, 7–12 October 2012; pp. 2901–2908.
41. Kawasaki Heavy Industries, Ltd. Standard Specifications of DuAro. Available online: https://robotics.kawasaki.com/userAssets1/productPDF/duAro1_WD002NLF61052-E.pdf (accessed on 7 January 2022).
42. Kennedy, J.; Eberhart, R.C. Particle Swarm Optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
43. Liu, Z.; Nishi, T. Multipopulation Ensemble Particle Swarm Optimizer for Engineering Design Problems. *Math. Probl. Eng.* **2020**, *2020*, 1450985. [CrossRef]
44. Liu, Z.; Nishi, T. Strategy Dynamics Particle Swarm Optimizer. *Inf. Sci.* **2022**, *582*, 665–703. [CrossRef]
45. Piotrowski, A.P.; Napiorkowski, J.J.; Piotrowska, A.E. Population Size in Particle Swarm Optimization. *Swarm Evol. Comput.* **2020**, *58*, 100718. [CrossRef]