The International Journal of Advanced Manufacturing Technology (2024) 133:987–1010
https://doi.org/10.1007/s00170-024-13436-8

ORIGINAL ARTICLE

Manipulation order optimization in industrial pick-and-place operations: application to textile and leather industry

Francisco José Martínez-Peral[1] · Héctor Migallón[2] · Jorge Borrell-Méndez[1] · Miguel Martínez-Rach[2] · Carlos Pérez-Vidal[1]

Received: 20 February 2023 / Accepted: 14 February 2024 / Published online: 25 March 2024

Abstract
This work addresses the problem of the development of a robotic system for the picking of parts cut by a CNC machine and the optimization of the sequencing of this picking process. An automated parts collection system is optimized to reduce the time required to perform the task of both picking and the subsequent classification by the type of part. The automated picking system, which is located at the end of a cutting machine, uses a robot equipped with an additional axis to expand its working space. Therefore, in this proposal, the industrial equipment necessary to automate this process is designed and the process to be optimized is computationally modeled. In particular, three discrete optimization algorithms are analyzed, with different evolution strategies and operators, but all of them are free of specific configuration parameters. The whole process is shown in this research, from the design of the procedure to the design of the tool, the algorithm selection, and elements validation. Finally, the first steps towards its industrial implementation are presented, and the hypothesis behind this project is validated.

Keywords Sequence ordering · Pick-and-place optimization · Leather industry · Textile industry · Robotic classification

1 Introduction

1.1 Problem description

In industry, pick-and-place operations are one of the most automated tasks using robots of any topology (e.g., scara, parallel, cartesian). In the automotive industry, the food industry, PCB manufacturing, and the textile sector, robots are used to move parts, components, products, and packaging. Therefore, a small percentage improvement in the performance of this task can have a great impact on the productivity of industries, or sub-industries, in which these pick-and-place operations carry significant weight.

One of these sub-industries is the cutting of textile and leather parts using CNC cutting machines. This operation is carried out as an intermediate process in the construction of vehicle interiors, clothing manufacturing, and footwear manufacturing (to name those where it has the greatest impact at an industrial level). Parts are cut in such a way as to optimize the number of parts for a given material size (irregular leather or regular textile). This operation is called nesting and results in the disordered arrangement (in terms of position and orientation) of the pieces obtained.

Once cut, these pieces are collected by hand, following one of the two usual criteria: either all the same pieces together or all the pieces that make up a certain product together (the latter operation is known as kitting). This collection task is complex (there are many similar but not identical pieces), tedious, not very ergonomic, and of little added value.

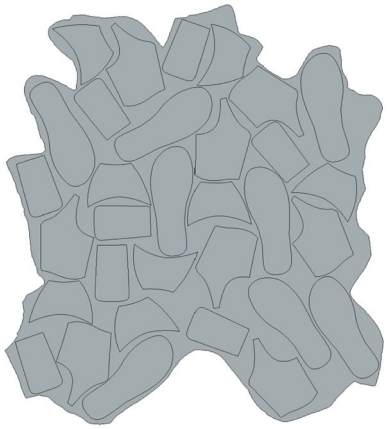
Graphically, the problem can be solved as a way to move from a set of disordered pieces (see Fig. 1a) to a set of ordered parts and a surplus of material, which is discarded (see Fig. 1b).

The paper is organized as follows: some related works are presented in the rest of this section, and the main contributions are outlined. Section 2 describes both the industrial equipment and the computational model, in addition to the

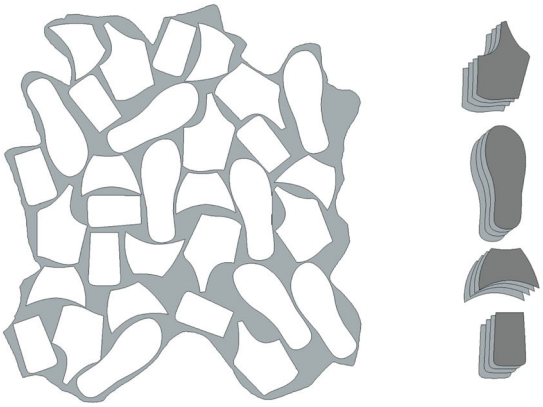
Francisco José Martínez-Peral, Héctor Migallón, Jorge Borrell-Méndez, Miguel Martínez-Rach, and Carlos Pérez-Vidal contributed equally to this work

✉ Héctor Migallón
hmigallon@umh.es

Extended author information available on the last page of the article



(a) Leather with cut pieces (initial state).



(b) Waste material and classified pieces (final state).

Fig. 1 Description of the classification process

optimization algorithms analyzed in this work. Section 3 shows the functional validation of the different parts that compose the proposed system, while Sect. 4 shows the improvements obtained in the production process, in addition to showing the feasibility of the complete proposed system. Finally, Sect. 5 presents the conclusions of this project.

1.2 Methodology

The methodology followed in this research includes the following phases:

- Definition of requirements: The goals and requirements of the project are defined. The specific task to be performed by the robot is defined, including technical, environmental, and performance constraints, as well as success criteria.
- Conceptual design: An initial design of the system is created, defining the type of robot, its relative position within the system, its kinematic and dynamic characteristics, and its extended workspace. Possible solutions and strategies for the pick-and-place task are identified.
- Development planning: Based on the previous phases and to meet the stated objectives, the following phases are developed:
  - Task planning: A detailed plan of the task to be performed by the robot is developed. The task is broken down into steps or sub-processes, and the sequences and dependencies between them are defined.
  - Path planning: The paths that the robot will follow to efficiently complete the task are determined. Factors such as the kinematics and dynamics of the robot and the choice of optimal motions are taken into account.
  - Process simulation: RobotStudio simulation software is used to validate a simplified linear model that is used recursively to minimize task execution time.
  - Gripping strategy: A multi-point gripping tool is designed and implemented to manipulate deformable textile and leather pieces. Optimization of object gripping is also considered to maximize the number of contact points between the tool and the object.
- Implementation and testing: The system is implemented according to the conceptual design. Extensive testing is then performed to verify that the robot and tool meet the requirements and perform the task effectively.
- Results evaluation: The robot's performance is evaluated based on the success criteria defined in the requirements phase. Data is recorded and compared with simulation data to determine if the task execution time has been minimized.

It should be noted that in each of these phases, iteration and feedback are critical to adapting and improving the system.

1.3 Related works

The task of sorting and classifying parts can, potentially, be optimized to try to reduce the overall production time. In this sense, there are some methods to minimize this time: parallelization of the task using several robots, use of faster robots, optimization of trajectories, choice of the sequence that requires the shortest execution time, etc.

In [1], the approach is to use an algorithm that generates an optimal trajectory for each arm performing the task by working in a coordinated way to avoid collisions, instead of using each arm sequentially.

To reduce the operation time, it has been shown that the sequencing order of the parts in the pick-and-place task is a relevant factor. In [2], an algorithm called best

uniform algorithm (BUA) was developed to obtain the optimal sequence, while other solutions are based on genetic algorithms (GA), such as those proposed by Goldberg [3] and Bessonet and Lallemand [4].

The optimization of motion planning for the pick-and-place tasks of robotic manipulators is a field that has been studied in depth. In the state of the art, several solutions to this problem are suggested. Bobrow et al. [5] suggested a phase plane analysis method to obtain a minimum time trajectory with confined torque. In [6–8], a similar problem was solved with a greedy search algorithm and convex optimization approaches. A solution is also proposed by solving the time-optimal problem, as presented in [9], based on the traveling salesman problem (TSP) with the genetic algorithm studied by Qu et al. [10]. In [11], a minimum trajectory is planned to minimize task execution time with a 2-DOF parallel robot.

In [12], a new way to mathematically model and formulate the pick-and-place operation in the food industry was presented. The Hungarian algorithm was proposed to optimize the total distance traveled by the robot to perform the task. The algorithm's capability to increase the productivity of the process was demonstrated with the aid of the algorithms used. In [13], a decision tree algorithm was proposed to minimize pick-and-place time in the shoe industry. In this case, the authors demonstrated an improvement when manipulating only five pieces.

Some researchers studying the pick-and-place problem developed algorithms to optimize different system characteristics. In [14], Ayob and Kendall developed a triple objective algorithm to optimize sequential pick-and-place tasks for soldering components on PCBs. In [1], the problem was solved with a hybrid iterated local search algorithm, and a relative optimal result was found quickly. Regarding the execution of algorithms in real time, the problem of performance evaluation and task optimization is not well studied in the literature. In [15], the problem was discussed by developing an algorithm to perform a safe pick-and-place task in real time, with image processing in mind. In [16], the authors combined the metaheuristic problem in real time, by allowing each robot to perform the assigned pick-and-place operations in real time, to maximize the throughput rate.

1.4 Main contributions

The main objective of this work is the design and validation of an automatic system for the collection of parts, from materials such as leather or other textile materials, from different cutting systems with CNC machinery. This type of system, widely used in the footwear industry, for example, is a semi-automatic system in which the cutting process is performed automatically, as well as the nesting process, i.e., the optimization of the placement of the pieces in the raw material. On the other hand, the parts picking system is still performed manually, this being a very repetitive task in which fatigue and loss of attention cause errors that are solved in the automated line proposed. The proposed system is framed within the Industry 4.0 revolution. In our proposal, we extend the current semi-automated work area with a robotic system capable of picking up parts with very different characteristics, constituting a fully automated production line.

Once the technological feasibility of the robotic system for the collection of cut pieces of leather or textile materials has been demonstrated, the optimization of the collection process is analyzed. This automatic process can be optimized by modifying the order in which the pieces are picked up, which is initially carried out in the same order in which they have been cut by the CNC cutting system. The information supplied to the robotic pick-and-place system is the one produced by the nesting procedure. Since the objective of nesting does not coincide with the optimization of the picking process, it is necessary to process this information to improve and optimize the automated pick-and-place.

To optimize the automated pick-and-place process, a low-cost computational model has been developed, from which the cost function is derived and minimized using discrete metaheuristic optimization algorithms. This low-cost computational model has been validated, both by means of computational tools with higher computational cost and by being used in a real production prototype. Three ordering algorithms are compared with non-optimized pick-and-place criteria using 50 real-world experiments from the leather industry. In particular, discrete JAYA [17], discrete teaching-learning optimization (DTLBO) [18], and discrete tree-seed algorithm (DTSA) [19] were selected to identify the improvement using metaheuristic discrete optimization algorithms.

Both the analysis of the computational complexity and the quality of the solution provided by the analyzed algorithms demonstrate the viability of the hardware and the software system as a whole application. In fact, this project has been carried out in collaboration with one of the largest European manufacturers of CNC cutting machines (Comelz). However, it is important to highlight that the system has no restrictions and can be used with any other machinery.

2 Proposed solution

Both the hardware system (explained in Sect. 2.1) and the software modeling (explained in Sect. 2.2) comprise the solution proposed in this work, as explained below.

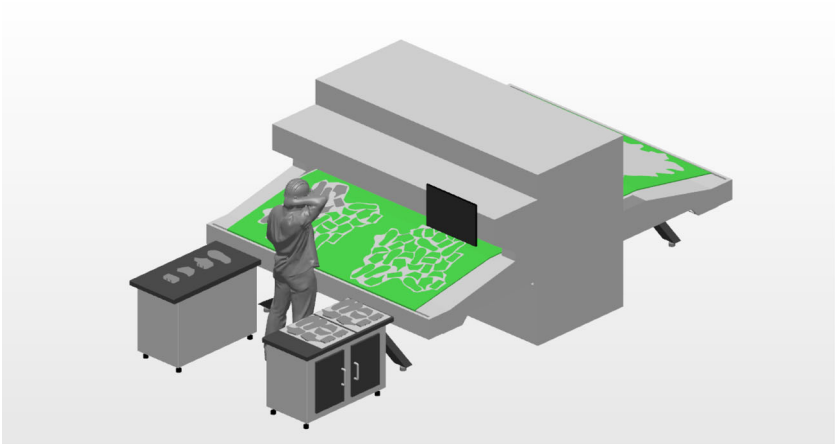


Fig. 2 Description of how the industrial process is currently performed, manually by a full dedication operator

2.1 Hardware system

2.1.1 Pick-and-place station

Currently, the picking of pieces and grouping them is a task performed manually at the end of the machine, as shown in Fig. 2. This is a tedious process that requires an operator picking parts at all times that the CNC cutting machine is in operation. In addition, an extra operator on the other side of the machine is required. This operator feeds the raw material and programs the cutting machine. Manual picking means that the pieces to be cut must be different enough to be distinguished by the operator at a glance, which makes it impossible to cut shoes of the same model with similar sizes at the same time. The solution that is usually adopted is to group only the same shoe size in each cutting process, so that there is no confusion in the grouping of pieces. The solution proposed in this project is a robotic sorting module, located in the same position as the operator was before (see Fig. 3). This module must be able to collect each of the cut pieces and classify them according to their shape.

Given that the CNC cutting machine uses a rolling mat as a transport element for the leather to be cut, the solution proposed in this project uses the same system to allow a transfer between the cutting machine and the sorting module developed. It is proposed to install an industrial or collaborative robot on this conveyor belt, mounted on a seventh axis of movement to increase its reach so that, with a vacuum-based gripper, the robot can be used to transport the leather to the cutting machine. Figure 4 shows the cutting machine at the top, where the material to be cut enters, represented in zone A of the figure. Depending on the advance of the parts, which is carried out by means of a conveyor belt or rolling mat, these reach the cutting heads which, by means of a blade or laser, generate the parts to be used in the manufacture of the final product. Zone B shows the parts already cut. The material is then transferred from the cutting machine's conveyor belt to the pick-and-place machine, where the robot is located with a

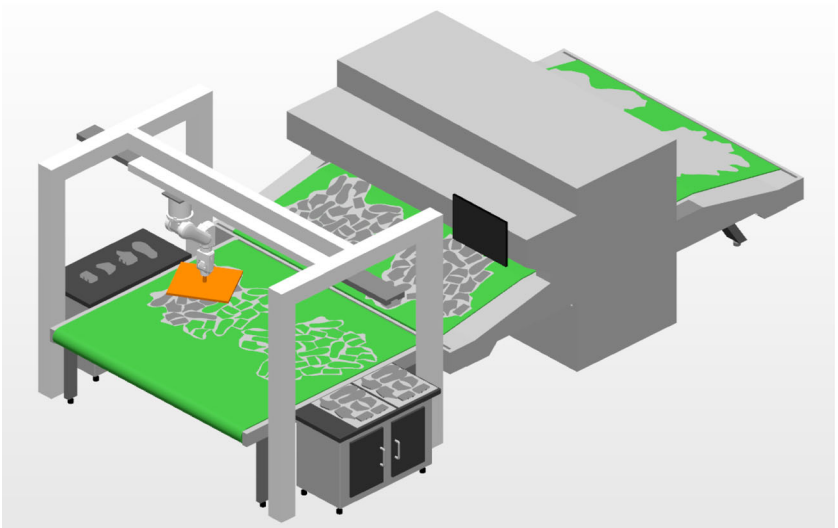


Fig. 3 Description of the proposed solution where a robot is performing the pick-and-place tasks

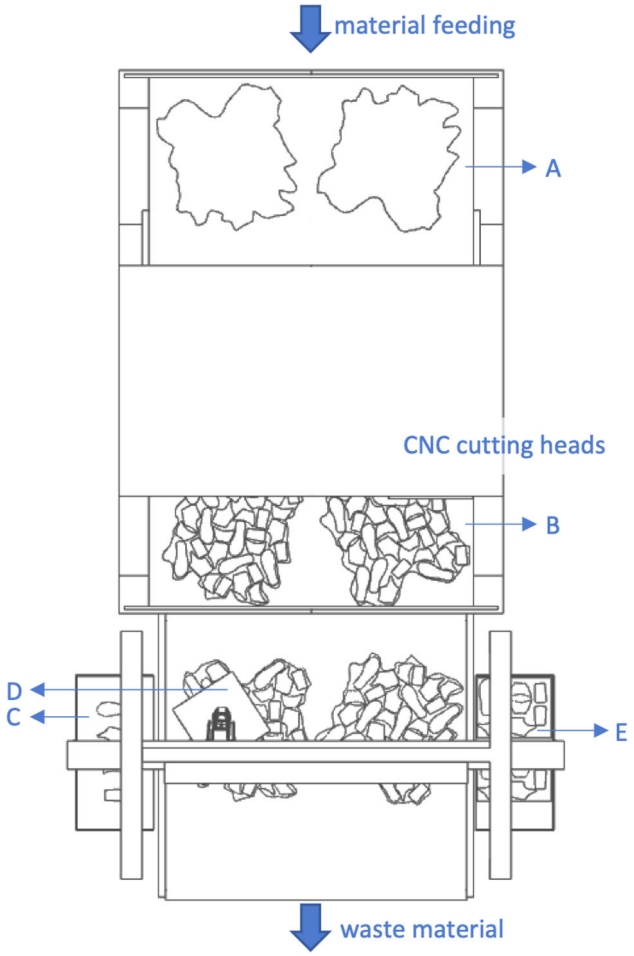


Fig. 4 Description of the industrial process showing a CNC cutting machine and the pick-and-place machine proposed in this project

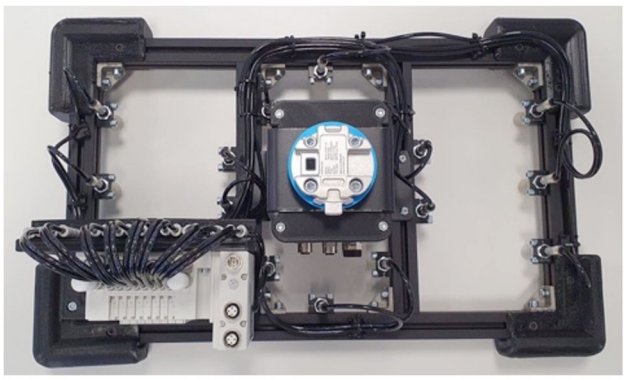
gripping tool D. The robot can sort the parts by type (zone C) or by the elements required to manufacture a product (zone E). Once all the parts have been removed and sorted, the excess material is discarded or recycled. With this solution, a single operator can feed the cutting machine and remove the sorted parts. Traditionally, two people are needed to operate the cutting machine, one at the material inlet and one at the outlet. The system proposed in this project increases the labor efficiency of factory production by introducing a system which is able to manipulate pieces, depending on their shape.

The system proposed in this work, on which the movement sequence optimization algorithms have been implemented, first consists of a CNC machine, which cuts the pieces on the leather. Once the cut has been made, the leather arrives via a conveyor belt to the robot's workspace, where the pick-and-place is to be performed. The parts are of different sizes and shapes and can be placed in any position and orientation. In the pick-and-place task, the parts are sorted according to size and shape, making a pile of the same type. It is in this step where sequence optimization algorithms are applied to minimize the time of the task, as the leather can contain a large number of parts and reducing a small amount of time for each part can greatly increase the productivity of the factory.

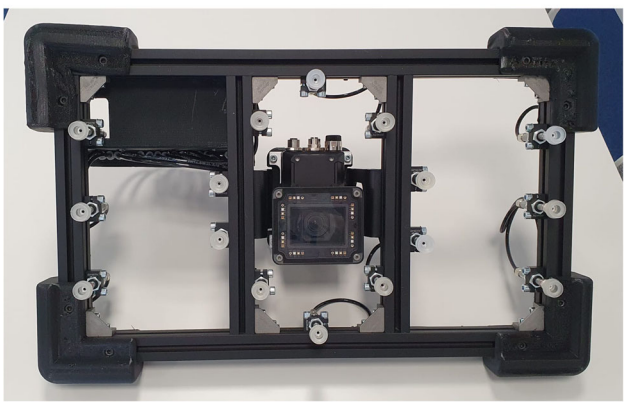
2.1.2 Multi-point vacuum gripper

The workpieces cut by the CNC table in a real production line have very different topologies and sizes. The vacuum gripping tool to be used must be configured in such a way that it can adapt to all cases. This project proposes the use of a vacuum tool based on matrix suction cups, which are activated independently and only grip the part to be handled. This gripper can be seen in Fig. 5a, where the suction cups activated by the robot are the ones matching the pieces to be picked up.

As an example, Fig. 6 shows that once a piece to be picked up is located, the robot applies a correction algorithm that modifies the position and orientation of its end



(a) Multi-point suction cups vacuum gripper.



(b) Selective activation of suction cups.

Fig. 5 Multi-point suction cups vacuum gripper designed and implemented in this project

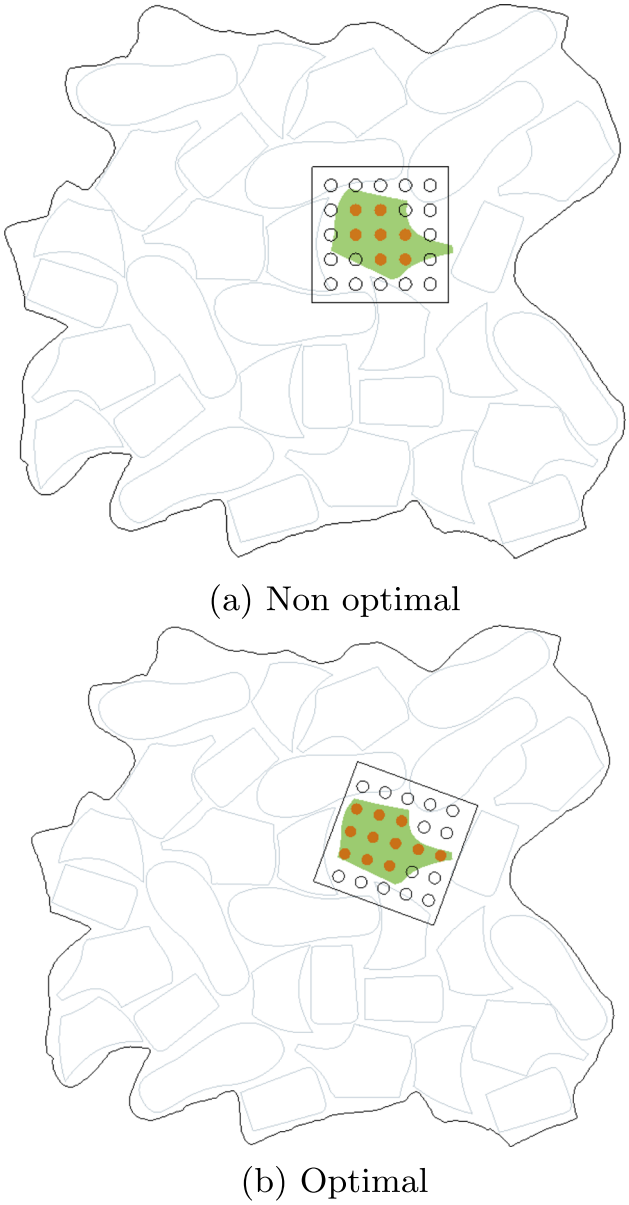


Fig. 6 Gripping of pieces

effector to maximize the number of suction cups in contact with the piece. Figure 6a shows that, with the initial configuration, only seven suction cups would be picking up the piece while, by applying the corrections developed in this project, the same piece would be picked up by 11 suction cups (see Fig. 6b), an increase of more than 50%. This simple algorithm has made it possible to increase the success of gripping non-rigid material or materials with a tendency to wrinkle during handling.

It should be noted that the final topology of the suction cups is not the square matrix shown in Fig. 6, but the irregular distribution is shown in Fig. 5b (i.e., a heterogeneous location). This has greatly increased the ability to handle particularly small or particularly large pieces.

2.2 Software system

As already stated, the proposed robotic system allows automation of the pick-and-place process. In traditional production lines in the leather industry, the nesting process is optimized and the information is provided to the CNC cutting system. This information will be used in our proposal to determine the optimal pick-and-place procedure after the cutting process with the CNC machine. Nowadays, this pick-and-place process is still a manual process. In the non-optimized automated process, the order in which the cut parts are picked up is the same as the order in which they are cut, usually following a raster order. The pick-and-place process must initially be modeled and developed at the software level, and from this modeling, the cost function of the system must be derived. An optimal or near-optimal solution for this cost function, and thus for the pick-and-place process, is obtained using a discrete heuristic optimizer.

2.2.1 Simulator

A low computational cost pick-and-place process simulator has been developed to model all of the movements to be performed by the robotic system, as well as the time costs of these processes. This information is used to correctly organize the production processes and avoid idle time in the production line. On the other hand, it is used to extract the cost function used in our optimization proposal of a pick-and-place process according to both the specific process to be performed and the specific robot used.

The developed simulator faithfully represents all the movements of the robot when performing the pick-and-place process. Initially, three different movements are distinguished: pick-and-place movements, rotation movements, and translation movements.

The pick-and-place process is performed with the vacuum multi-point gripper described in Sect. 2.1.2. During this process, the robot must not be in motion, i.e., its velocity must be zero, so the time cost of this task depends on the characteristics of the hardware system used.

Regardless of the rotation angle in the picking process, all parts must be placed at the same angle. The information provided by the nesting optimization includes the rotation angle of each part, from which the rotation to be performed after the picking of each part is calculated. This rotation process, of 180° maximum, has no effective time cost, since it overlaps with the translation movement.

Finally, the translational motion is a kinematic motion characterized by a maximum velocity and acceleration. These parameters, as well as the allowed movements, depend on the hardware used. In the system developed, the simulator models a robot in which the acceleration is constant, the maximum speed is configurable, and the degrees of freedom allow minimum trajectories between any two points.

There are commercial simulators. The RobotStudio simulator, for example, offers very accurate simulations of robot movements, but this type of simulator is intended for other tasks and its computational cost is too high to be used in the optimization process performed in our proposal. This optimization process must be performed in the time interval between the availability of the cutting information, calculated by the nesting process, and the start of operations of the robotic system responsible for the pick-and-place. The optimizable cost function, that includes all optimizable time costs, is derived from the developed simulator.

2.2.2 Discrete optimization algorithms

The problem studied in this work should be considered a special case of the TSP problem (traveling sales problem). In our problem, relevant aspects of the robot arm motion, such as acceleration and velocity, must also be considered, which affect both the optimization and the simulation results. In particular, it is necessary to consider a TSP-type problem where each time one passes through one or more destinations, one has to return to the base station; in our case, one has to return after passing through each destination, and with the characteristic that there is more than one base station and the return station depends on the visited destination. This type of problem has been considered NP-hard in a large number of works for several decades, for example, in [28] dating back to 1976 and in [29] dating back to 1978. In [30], a variant of TSP closer to the work discussed in this paper, which includes the condition of eventually returning to the starting point, is also treated as NP-hard. Note that, NP-hard is a category of problems that includes problems that are at least as hard as those in NP but not necessarily in NP.

On the other hand, considering only the industrial objectives, it has been considered that an exact solution can be obtained by brute force only in the case of a process with a very limited number of parts; moreover, the industrial processes are exactly the same in each repetition. These conditions are not met in the presented work, where the number and type of parts depend on the characteristics of the raw material, mainly in terms of size and imperfections to be treated in each industrial process. Working with heuristic discrete optimization algorithms is therefore a feasible solution.

Optimization algorithms can be initially divided according to the type of problems for which they were designed. Therefore, we find algorithms that solve continuous optimization problems and algorithms that solve discrete optimization problems. The latter is the type of problem to be solved in the challenge addressed in our work. While the variables of the possible solutions of continuous optimization problems are real-valued, in discrete optimization, they can only be elements of a finite set.

Many discrete optimization problems are used in real-world applications, e.g., the traveling salesman problem (TSP) [20, 21], the graph coloring problem [22], the manufacturing cell formation problem [23], and the water pump switching problem [24]. Discrete problems of the TSP type are NP-hard complexity problems and, therefore, cannot be solved by any known method in polynomial time. When solving or obtaining an acceptable solution to categorized NP-hard optimization problems, metaheuristic algorithms obtain near-optimal solutions with a reasonable computational cost; moreover, these algorithms are not problem-dependent and are easily adaptable with simple structures.

Three of the most efficient algorithms for solving discrete optimization problems, which offer different structures and/or search strategies, have been used and adapted in order to analyze the behavior of these algorithms in solving the cost function of the system. These algorithms are the discrete tree-seed algorithm (DTSA) [19] (inspired by the tree-seed algorithm (TSA) [25]), the discrete JAYA algorithm (DJAYA) [17] (an algorithm inspired by the JAYA [26]), the continuous optimization algorithm, and the discrete teaching-learning-based optimization algorithm (DTLBO) [18] (inspired by the teaching-learning-based optimization algorithm (TLBO) [27]). These algorithms have been adapted for the pick-and-place problem under study; they, and the modifications made, are described as follows. All of them are population-based metaheuristic optimization algorithms which are free from configuration parameters; therefore, the only parameters to be set are the population size and the stopping criterion. The stopping criterion can be either the number of cost function evaluations (FE) performed or the time requirement involved in the production system.

There are common phases and procedures in all of the algorithms used. First, the initial population (see Algorithm 1)

Algorithm 1 Initial population generation.

1: **for** $m = 0$ to $popSize$ **do**
2: **for** $k = 1$ to N **do**
3: $Pop_m^k = lB^k + (uB^k - lB^k) * rand^k$
4: **end for**
5: **end for**

Algorithm 2 Swap transformation operation.

```
INPUT: oldCandidate[N]
newCandidate = oldCandidate
Obtain integer random number r1 in [1,N]
Obtain integer random number r2 in [1,N]
while r1 == r2 do
    Obtain integer random number r2 in [1,N]
end while
newCandidate[r1] = oldCandidate[r2]
newCandidate[r2] = oldCandidate[r1]
OUTPUT: newCandidate[N]
```

is created, and it should be noted that, depending on the algorithm, it is given different names, such as random permutations except for two individuals: one of them is obtained as the nearest neighbor tour, and the other one is the raster order solution, i.e., the solution used in a manual process.

The purpose of Algorithm 1 is to create a population of candidate solutions for our optimization problem. The algorithm starts by using two main parameters: the population size ($popSize$) and the number of design variables of the problem to be optimized (N). This algorithm generates an initial population of candidate solutions by randomly initializing the design variables within specified lower (lB) and upper (uB) bounds. The random numbers obtained in all cases, including this population initialization, were obtained according to a normal distribution.

Both DJAYA and DTSA use the same transformation operators (swap, shift, and symmetry) which always obtain feasible candidates, while DTLBO uses different transformation processes that may obtain non-feasible candidates.

In order to apply the swap transformation operator, initially, two different random numbers between 1 and the number of variables of the problem to be solved (N) are obtained. These two numbers indicate the two positions that are exchanged to obtain a new candidate, as shown in Algorithm 2. It should be noted that the number of variables (N) of the cost function in a pick-and-place process is the number of parts to be picked and placed afterwards.

Algorithm 3 Shift transformation operation.

```
INPUT: oldCandidate[N]
newCandidate = oldCandidate
Obtain integer random number r1 in [1,N]
Obtain integer random number r2 in [1,N]
while r1 == r2 do
    Obtain integer random number r2 in [1,N]
end while
if r1 > r2 then
    Obtain integer random number r2 in [1,N]
    temp = r1
    r1 = r2
    r2 = r1
end if
for i = r1 to r2 − 1 do
    newCandidate[i] = oldCandidate[i + 1]
end for
newCandidate[r2] = oldCandidate[r1]
OUTPUT: newCandidate[N]
```

Algorithm 4 Symmetry transformation operation.

```
INPUT: oldCandidate[N]
newCandidate = oldCandidate
Obtain integer random number r1 and r2 as shwon in lines 3–13 of Algorithm 3
Obtain integer random size ts in [1,N/2]
while r1 + rs > r2 or r2 + rs > N do
    Obtain integer random number r1 and r2 as shwon in lines 3–13 of Algorithm 3
    Obtain integer random size ts in [1,N/2]
end while
for i = 0 to rs do
    newCandidate[r1 + i] = oldCandidate[r2 + rs − i]
    newCandidate[r2 + rs − i] = oldCandidate[r1 + i]
end for
OUTPUT: newCandidate[N]
```

The shift transformation consists of shifting a block of randomly sized variables by a single position to obtain a new candidate, as shown in Algorithm 3. Algorithm 3 shows that the displacement of the block made by the shift transformation is closed with the first element and so it always generates feasible elements.

Algorithm 5 DTSA algorithm.

```
INPUT: max_FEs, N and ST
Generate the initial population Pop
Compute the objective function values (F_obj())
Set mum_FEs = N
while mum_FEs < max_FEs do
    Determine the best tree Best
    for i = 1 to N do
        Obtain random number r in [0,1]
        if r > ST then
            Create seed using swap and Best
            Create seed using shift and Best
            Create seed using symmetry and Best
            Create seed using swap and Pop_i
            Create seed using shift and Pop_i
            Create seed using symmetry and Pop_i
        else
            Select random tree Pop_r
            Create seed using swap and Pop_r
            Create seed using shift and Pop_r
            Create seed using symmetry and Pop_r
            Create seed using swap and Pop_i
            Create seed using shift and Pop_i
            Create seed using symmetry and Pop_i
        end if
        Determine the best seed B_seed
        if F_obj(B_seed) < F_obj(Pop_i) then
            Pop_i = B_seed
        end if
    end for
    mum_FEs = mum_FEs + 6 * N
end while
```

Algorithm 6 DJAYA algorithm.

```
1: INPUT: max_FEs, N, ST1 and ST2
2: Generate the initial population Pop
3: Compute the objective function values (F_obj())
4: Set mum_FEs = N
5: while mum_FEs < max_FEs do
6:    Determine the best and worst candidate (Best and Worst)
7:    for i = 1 to N do
8:       Obtain random number r in [0,1]
9:       if r <= ST1 then
10:         oldCandidate = Pop_i
11:      else
12:         if r <= ST1 then
13:            oldCandidate = Best
14:         else
15:            oldCandidate = Worst
16:         end if
17:      end if
18:      Select transformation operator by Roulette Wheel
19:      Create newCandidate
20:      if F_obj(newCandidate) < F_obj(Pop_i) then
21:         Pop_i = newCandidate
22:      end if
23:   end for
24:   mum_FEs = mum_FEs + N
25: end while
```

Finally, in the symmetry transformation, a symmetry process is performed on two blocks of equal size, avoiding any overlapping. As shown in Algorithm 4, those blocks whose size cannot be larger than half the size of the candidates are first selected, and then the symmetry process is applied, thus ensuring that feasible candidates are obtained.

It should be noted that the swap, shift, and symmetry transformation operations are used by both the DJAYA algorithm and the DTSA algorithm, which implies that the generation of new candidates does not require a feasibility check, as it is guaranteed by the procedure of these transformation processes.

The DTSA algorithm is a population-based metaheuristic optimization in which the individuals of the population or candidates are called trees. Each of these trees generates a total of six new candidates in each new generation, called "seeds." For the generation of these six new seeds (or candidates), the current tree and the best tree or a random tree are used, depending on the search tendency (ST) parameter,

Algorithm 7 Crossover transformation operation.

```
1: INPUT: individual1[N],individual2[N]
2: newCandidate = individual1
3: Obtain integer random number r_1 and r_2 as shwon in lines 3–13 of
   Algorithm 3
4: for i = r1 to r_2 do
5:    newCandidate[i] = individual2[i]
6: end for
7: OUTPUT: newCandidate[N]
```

Algorithm 8 DTLBO algorithm.

```
1: INPUT: max_FEs, N
2: Generate the initial population Pop
3: Divide the whole population in 4 subpopulation Pop_j, j in [1,4]
4: Compute the objective function values (F_obj())
5: Set mum_FEs = N
6: while mum_FEs < max_FEs do
7:    Determine the teacher of each subpopulation (Best_j) and the
   global best (Teacher)
8:    LEARNER STAGE
9:    for i = 1 to N do
10:      Set j to id of the subpopulation of i
11:      Obtain random number r in [0,1]
12:      if r < 0.25 then
13:         newCandidate = oldCandidate Θ Teacher
14:      else if r < 0.5 then
15:         newCandidate = oldCandidate Θ Best_j
16:      else if r < 0.75 then
17:         newCandidate = oldCandidate Θ Mean_j
18:      else
19:         newCandidate = Best_j Θ Mean_j
20:      end if
21:      Ensure the feasibility of newCandidate
22:      if F_obj(newCandidate) < F_obj(Pop_i) then
23:         Pop_i = newCandidate
24:      end if
25:   end for
26:   LEARNER STAGE
27:   for i = 1 to N do
28:      Obtain integer random number r in [1,N]
29:      while r == i do
30:         Obtain integer random number r in [1,N]
31:      end while
32:      newCandidate = oldCandidate Θ Pop_r
33:      if F_obj(newCandidate) < F_obj(Pop_i) then
34:         Pop_i = newCandidate
35:      end if
36:   end for
37:   mum_FEs = mum_FEs + 2 * N
38: end while
```

which can take values between 0 and 1. Once the six new candidates (or seeds) are obtained, the best of them is obtained and used as the tree (or candidate) in the new generation if it improves the current tree (or candidate). The DTSA algorithm is shown in Algorithm 5, in which it can be seen that the transformation operators are used to generate the six seeds.

The JAYA continuous algorithm introduces a modification, with respect to the vast majority of heuristic algorithms in the search procedure for the optimum. This modification consists of using both the best and the worst current elements of the population. DJAYA maintains this feature. In addition, like DTSA, it uses the concept of the search tendency parameter (ST), but, to balance the exploration and exploitation phases, it uses two search trend parameters (ST1 and ST2). Algorithm 6 shows the DJAYA algorithm, in which the transformation operators used are still the three operators described previously, and the algorithm to be used for each candidate is determined by a roulette wheel procedure (see

line 18 of Algorithm 6) that is applied after performing 10% of the function evaluations (FE).

The DTLBO algorithm, like TLBO, is based on teaching and learning processes. The DTLBO algorithm presents important changes in the skeleton of the algorithm, with respect to the two algorithms already described. DTLBO is also a population-based algorithm, and the candidates that make up this population are called learners, while the best individual of a generation is referred to as a teacher. The initial population is obtained following the same procedure as in the previous algorithms but is divided into n subpopulations. This mechanism is used to increase diversity by having n individuals acting as a teacher; the number of subpopulations was set to 4, as in [18]. Moreover, the DTLBO algorithm is a two-phase algorithm, like TLBO, with a first phase called the teaching phase and a second phase called the learning phase. The procedure conducted, which can be seen in Algorithm 8, produces candidates that may be infeasible, which makes it necessary to perform processes that make these infeasible candidates feasible. Therefore, the transformation operator used (the crossover (Θ) shown in Algorithm 7) may generate infeasible individuals. To obtain a new candidate, two individuals are used from among the old candidates, the best overall individual ($Teacher$), the best individual of the subpopulation ($Best_j$), or the individual called mean ($Mean_j$), which is calculated as the mean of the individuals of each subpopulation.

It should be noted that, especially in the exploitation phase, duplicate candidates are likely to be generated. Before replacing an old candidate with a new one, in any of the algorithms tested, a check is made to ensure that the latter no longer exists as a candidate, in which case the replacement would not be executed.

3 System analysis and validation

3.1 Simulator proposed vs physics-based simulator

As explained in Sect. 2.2.1, a simulator has been developed that fully models the motion to be performed by the robotic system in the pick-and-place process. This model, based on the intrinsic characteristics of the robotic system, which are maximum speed, acceleration, degrees of freedom, and gripping system, provides the time cost of the pick-and-place process, as well as the cost function, related to the translational movements of the robotic system to be used in the optimization process. These results should be validated. In our case, these results were compared with those provided by ABB's RobotStudio software.

The possibility of using RobotStudio for research purposes has already been analyzed due to the advantages it incorporates, in terms of kinematic analysis of the devices. In [7], an inertial sensor coupled to a robot was used to compare the data obtained with the theoretical values provided by the simulation software. Simulations of the robot operation were performed and compared with the results of a set of experiments in a physical setup. The referenced article presents simulations with an analysis of the TCP speed of the tool and analyzes the influence of different parameters on the accuracy of the execution of the movement speed. The research results presented show a considerable similarity of the simulation with the real behavior of the robots. Measurements with the inertial sensor show that the acceleration signals are characterized by high interference. Despite this, the filtered mean value gives very similar results for the simulation and real experiments. In addition, preliminary testing showed a high accuracy of the gyroscopic sensor, with only a small deviation from the mean value, which the authors propose should be analyzed in future work. For these reasons, it is considered that the conclusions reached using RobotStudio will closely reflect the behavior of the real system. In all the experiments performed, the error obtained in the simulation

Table 1 Comparison of real results with respect to the results obtained using a simplified model: velocity 0.5 m/s

Place	Pick	Real (s.)	Simplified model (s.)	Pick	Real (s.)	Simplified model (s.)
Place01	Pick01	2866	2825	Pick06	2912	2871
Place02	Pick01	2877	2839	Pick06	2930	2889
Place03	Pick01	2888	2845	Pick06	2937	2896
Place04	Pick01	2835	2796	Pick06	2860	2820
Place05	Pick01	2932	2891	Pick06	2980	2940
Place01	Pick02	2895	2854	Pick07	2879	2838
Place02	Pick02	2909	2869	Pick07	2916	2875
Place03	Pick02	2936	2896	Pick07	2909	2867
Place04	Pick02	2874	2834	Pick07	2874	2834
Place05	Pick02	2971	2932	Pick07	2940	2898
Place01	Pick03	2907	2869	Pick08	2838	2798
Place02	Pick03	2942	2903	Pick08	2872	2832
Place03	Pick03	2954	2911	Pick08	2872	2830
Place04	Pick03	2892	2850	Pick08	2827	2785
Place05	Pick03	2989	2950	Pick08	2894	2851
Place01	Pick04	2928	2886	Pick09	2888	2846
Place02	Pick04	2939	2899	Pick09	2919	2878
Place03	Pick04	2947	2906	Pick09	2924	2881
Place04	Pick04	2885	2845	Pick09	2876	2835
Place05	Pick04	2951	2913	Pick09	2952	2910
Place01	Pick05	2846	2805	Pick10	2868	2826
Place02	Pick05	2865	2823	Pick10	2895	2857
Place03	Pick05	2879	2839	Pick10	2897	2855
Place04	Pick05	2806	2763	Pick10	2862	2826
Place05	Pick05	2885	2844	Pick10	2933	2893

Table 2 Comparison of real results with respect to the results obtained using a simplified model: velocity $1m/s$

Place	Pick	Real (s.)	Simplified model (s.)	Pick	Real (s.)	Simplified model (s.)
Place01	Pick01	1508	1412	Pick06	1531	1435
Place02	Pick01	1513	1419	Pick06	1540	1445
Place03	Pick01	1519	1422	Pick06	1543	1448
Place04	Pick01	1492	1398	Pick06	1505	1410
Place05	Pick01	1541	1446	Pick06	1565	1470
Place01	Pick02	1522	1427	Pick07	1514	1419
Place02	Pick02	1529	1434	Pick07	1533	1437
Place03	Pick02	1543	1448	Pick07	1530	1433
Place04	Pick02	1512	1417	Pick07	1512	1417
Place05	Pick02	1561	1466	Pick07	1545	1449
Place01	Pick03	1529	1435	Pick08	1494	1399
Place02	Pick03	1546	1451	Pick08	1511	1416
Place03	Pick03	1552	1456	Pick08	1511	1415
Place04	Pick03	1521	1425	Pick08	1488	1393
Place05	Pick03	1569	1475	Pick08	1522	1426
Place01	Pick04	1539	1443	Pick09	1519	1423
Place02	Pick04	1545	1449	Pick09	1535	1439
Place03	Pick04	1549	1453	Pick09	1537	1441
Place04	Pick04	1517	1422	Pick09	1513	1418
Place05	Pick04	1550	1456	Pick09	1551	1455
Place01	Pick05	1498	1402	Pick10	1509	1413
Place02	Pick05	1507	1412	Pick10	1523	1429
Place03	Pick05	1514	1420	Pick10	1523	1427
Place04	Pick05	1478	1381	Pick10	1506	1413
Place05	Pick05	1518	1422	Pick10	1542	1447

Table 3 Comparison of real results with respect to results using a simplified model: velocity $1.5m/s$

Place	Pick	Real (s.)	Simplified model (s.)	Pick	Real (s.)	Simplified model (s.)
Place01	Pick01	1089	942	Pick06	1104	957
Place02	Pick01	1092	946	Pick06	1110	963
Place03	Pick01	1096	948	Pick06	1112	965
Place04	Pick01	1078	932	Pick06	1087	940
Place05	Pick01	1111	964	Pick06	1127	980
Place01	Pick02	1098	951	Pick07	1093	946
Place02	Pick02	1103	956	Pick07	1106	958
Place03	Pick02	1112	965	Pick07	1103	956
Place04	Pick02	1092	945	Pick07	1092	945
Place05	Pick02	1124	977	Pick07	1113	966
Place01	Pick03	1103	957	Pick08	1079	933
Place02	Pick03	1114	968	Pick08	1091	944
Place03	Pick03	1118	970	Pick08	1091	943
Place04	Pick03	1097	950	Pick08	1076	928
Place05	Pick03	1129	983	Pick08	1098	950
Place01	Pick04	1109	962	Pick09	1096	949
Place02	Pick04	1113	966	Pick09	1106	959
Place03	Pick04	1116	969	Pick09	1108	960
Place04	Pick04	1095	948	Pick09	1092	945
Place05	Pick04	1117	971	Pick09	1118	970
Place01	Pick05	1082	935	Pick10	1089	942
Place02	Pick05	1088	941	Pick10	1098	952
Place03	Pick05	1093	946	Pick10	1099	952
Place04	Pick05	1068	921	Pick10	1087	942
Place05	Pick05	1095	948	Pick10	1111	964

obtained with RobotStudio was less than 1%. However, the computational cost of both software tools is not comparable. It should be noted that RobotStudio is a validated and reliable software tool for these analyses, as its simulations perfectly match real movements but a fast model that accurately approximates to it is worthwhile.

As already explained, the cost function to be optimized was designed based on the modeling of the kinematic system. A commonly used simplification is based on optimizing the traveled distance, i.e., like a traditional TSP problem, instead of optimizing the most complex model of the robotic system. Tables 1, 2, and 3 show the discrepancy between the real results, which coincide with those obtained with our simulation system, and the simplification that only considers the traveled distance. In these tables, it can be seen that, for slow speeds, the discrepancy is not too important but, when the speed increases, the discrepancies are relevant. The errors due to the use of a simplified optimization model (TSP problem) can affect both the quality of the solution and the organization of the production lines, especially when higher-performance robotic systems are used. As demonstrated in Sect. 3.1, the results of the real physical system match the results obtained with our simulator.

3.2 Analysis of the discrete optimization algorithms

To evaluate the algorithms, we used real examples of the pick-and-place problem, including ten pick jobs (named Pick01-Pick10 with 500 parts) and five different locations (named Place01-Place05, for 32 different part classes). The constant acceleration of the robot was 10 m/s2 in all of the experiments performed, while the maximum velocity took values of 0.5 m/s, 1 m/s, or 1.5 m/s.

As mentioned above, the chosen algorithms were free of configuration parameters, except for the stopping criterion and population size. The stopping criterion used in this analysis was the maximum number of evaluations of the cost function, called $\max_{FEs}$ in the previous algorithms. These heuristic algorithms require the execution of several independent runs to avoid a solution trapped in a local minimum;

Table 4 Best solution and standard deviation for DTSA, DJAYA, and DTLBO algorithms with 30 independent runs, a population size equal to 60, $\max_{FEs} = 500,000$ and velocity 0.5 m/s

Place	Pick	DTSA		DJAYA		DTLBO		Pick	DTSA		DJAYA		DTLBO	
		Best (s.)	St. Dev.	Best (s.)	St. Dev.	Best (s.)	St. Dev.		Best (s.)	St. Dev.	Best (s.)	St. Dev.	Best (s.)	St. Dev.
Place01	Pick01	2866	0.23	2873	0.31	2875	0.05	Pick06	2912	0.26	2918	0.35	2921	0.04
Place02	Pick01	2877	0.37	2885	0.57	2889	0.24	Pick06	2930	0.23	2936	0.34	2939	0.06
Place03	Pick01	2888	0.25	2893	0.27	2895	0.06	Pick06	2937	0.28	2944	0.37	2946	0.08
Place04	Pick01	2835	0.28	2842	0.31	2846	0.18	Pick06	2860	0.25	2867	0.35	2870	0.02
Place05	Pick01	2932	0.22	2939	0.37	2941	0.22	Pick06	2980	0.20	2986	0.29	2990	0.57
Place01	Pick02	2895	0.28	2901	0.37	2904	0.08	Pick07	2879	0.27	2886	0.30	2888	0.10
Place02	Pick02	2909	0.30	2916	0.36	2919	0.20	Pick07	2916	0.20	2922	0.36	2925	0.23
Place03	Pick02	2936	0.20	2944	0.36	2946	0.05	Pick07	2909	0.24	2915	0.28	2917	0.12
Place04	Pick02	2874	0.23	2882	0.40	2884	0.12	Pick07	2874	0.25	2881	0.41	2884	0.33
Place05	Pick02	2971	0.22	2979	0.44	2982	0.32	Pick07	2940	0.22	2946	0.26	2948	0.05
Place01	Pick03	2907	0.43	2916	0.48	2919	0.26	Pick08	2838	0.31	2846	0.30	2848	0.09
Place02	Pick03	2942	0.17	2950	0.49	2953	0.11	Pick08	2872	0.32	2881	0.37	2882	0.20
Place03	Pick03	2954	0.20	2960	0.29	2961	0.04	Pick08	2872	0.32	2878	0.47	2880	0.13
Place04	Pick03	2892	0.30	2898	0.42	2900	0.45	Pick08	2827	0.25	2834	0.28	2835	0.15
Place05	Pick03	2989	0.25	2996	0.39	3000	0.04	Pick08	2894	0.30	2901	0.30	2901	0.15
Place01	Pick04	2928	0.25	2934	0.27	2936	0.15	Pick09	2888	0.19	2895	0.31	2896	0.12
Place02	Pick04	2939	0.24	2946	0.37	2949	0.16	Pick09	2919	0.25	2926	0.39	2928	0.18
Place03	Pick04	2947	0.25	2955	0.33	2956	0.12	Pick09	2924	0.16	2930	0.27	2931	0.11
Place04	Pick04	2885	0.27	2892	0.39	2895	0.15	Pick09	2876	0.20	2883	0.28	2885	0.17
Place05	Pick04	2951	0.23	2959	0.38	2963	0.12	Pick09	2952	0.26	2959	0.32	2960	0.16
Place01	Pick05	2846	0.23	2852	0.27	2855	0.06	Pick10	2868	0.23	2874	0.27	2876	0.16
Place02	Pick05	2865	0.23	2871	0.38	2873	0.03	Pick10	2895	0.26	2902	0.42	2907	0.12
Place03	Pick05	2879	0.24	2886	0.41	2889	0.11	Pick10	2897	0.19	2903	0.34	2905	0.23
Place04	Pick05	2806	0.24	2811	0.27	2813	0.10	Pick10	2862	0.29	2871	0.65	2876	0.36
Place05	Pick05	2885	0.23	2892	0.42	2894	0.09	Pick10	2933	0.23	2941	0.37	2943	0.06

Table 5 Best solution and standard deviation for DTSA, DJAYA, and DTLBO algorithms with 30 independent runs, a population size equal to 60, $\max_{FEs} = 500,000$ and velocity $1m/s$

Place	Pick	DTSA		DJAYA		DTLBO		Pick	DTSA		DJAYA		DTLBO	
		Best (s.)	St. Dev.	Best (s.)	St. Dev.	Best (s.)	St. Dev.		Best (s.)	St. Dev.	Best (s.)	St. Dev.	Best (s.)	St. Dev.
Place01	Pick01	1508	0.12	1511	0.14	1512	0.03	Pick06	1531	0.11	1534	0.16	1535	0.02
Place02	Pick01	1513	0.15	1517	0.27	1519	0.09	Pick06	1540	0.13	1543	0.17	1545	0.02
Place03	Pick01	1519	0.11	1522	0.11	1522	0.03	Pick06	1543	0.15	1547	0.15	1548	0.04
Place04	Pick01	1492	0.12	1496	0.23	1498	0.04	Pick06	1505	0.11	1508	0.13	1510	0.02
Place05	Pick01	1541	0.17	1544	0.19	1546	0.09	Pick06	1565	0.11	1569	0.16	1570	0.22
Place01	Pick02	1522	0.11	1526	0.16	1527	0.04	Pick07	1514	0.11	1518	0.19	1519	0.04
Place02	Pick02	1529	0.13	1533	0.16	1534	0.09	Pick07	1533	0.14	1536	0.21	1537	0.15
Place03	Pick02	1543	0.13	1547	0.20	1548	0.04	Pick07	1530	0.13	1532	0.16	1533	0.05
Place04	Pick02	1512	0.13	1516	0.16	1517	0.08	Pick07	1512	0.15	1516	0.21	1517	0.12
Place05	Pick02	1561	0.15	1564	0.14	1566	0.16	Pick07	1545	0.10	1548	0.15	1549	0.04
Place01	Pick03	1529	0.14	1533	0.23	1535	0.13	Pick08	1494	0.11	1498	0.18	1499	0.04
Place02	Pick03	1546	0.13	1549	0.13	1551	0.07	Pick08	1511	0.15	1515	0.22	1516	0.13
Place03	Pick03	1552	0.11	1554	0.17	1556	0.02	Pick08	1511	0.13	1514	0.16	1515	0.06
Place04	Pick03	1521	0.11	1524	0.20	1525	0.23	Pick08	1488	0.14	1492	0.15	1493	0.06
Place05	Pick03	1569	0.14	1573	0.19	1575	0.04	Pick08	1522	0.13	1525	0.17	1526	0.06
Place01	Pick04	1539	0.08	1542	0.16	1543	0.08	Pick09	1519	0.13	1522	0.18	1523	0.07
Place02	Pick04	1545	0.17	1548	0.19	1549	0.10	Pick09	1535	0.13	1538	0.18	1539	0.09
Place03	Pick04	1549	0.12	1552	0.18	1553	0.07	Pick09	1537	0.14	1540	0.16	1541	0.06
Place04	Pick04	1517	0.15	1521	0.17	1522	0.11	Pick09	1513	0.13	1516	0.19	1518	0.08
Place05	Pick04	1550	0.10	1554	0.20	1556	0.05	Pick09	1551	0.14	1554	0.17	1555	0.08
Place01	Pick05	1498	0.12	1501	0.14	1502	0.03	Pick10	1509	0.15	1512	0.15	1513	0.07
Place02	Pick05	1507	0.14	1511	0.14	1512	0.02	Pick10	1523	0.13	1527	0.23	1529	0.03
Place03	Pick05	1514	0.14	1518	0.20	1520	0.06	Pick10	1523	0.14	1526	0.16	1527	0.10
Place04	Pick05	1478	0.08	1480	0.15	1481	0.04	Pick10	1506	0.14	1511	0.30	1513	0.24
Place05	Pick05	1518	0.10	1521	0.20	1522	0.04	Pick10	1542	0.12	1545	0.17	1547	0.03

Table 6 Best solution and standard deviation for DTSA, DJAYA, and DTLBO algorithms with 30 independent runs, population size equal to 60, $\max_{FEs} = 500,000$ and velocity $1.5m/s$

Place	Pick	DTSA		DJAYA		DTLBO		Pick	DTSA		DJAYA		DTLBO	
		Best (s.)	St. Dev.	Best (s.)	St. Dev.	Best (s.)	St. Dev.		Best (s.)	St. Dev.	Best (s.)	St. Dev.	Best (s.)	St. Dev.
Place01	Pick01	1089	0.08	1091	0.08	1092	0.02	Pick06	1104	0.08	1106	0.07	1107	0.03
Place02	Pick01	1092	0.09	1095	0.16	1096	0.09	Pick06	1110	0.09	1112	0.12	1113	0.02
Place03	Pick01	1096	0.10	1097	0.10	1098	0.02	Pick06	1112	0.08	1115	0.10	1115	0.02
Place04	Pick01	1078	0.10	1081	0.12	1082	0.03	Pick06	1087	0.09	1089	0.13	1090	0.01
Place05	Pick01	1111	0.07	1113	0.12	1114	0.06	Pick06	1127	0.07	1129	0.16	1130	0.16
Place01	Pick02	1098	0.08	1100	0.10	1101	0.02	Pick07	1093	0.09	1095	0.11	1096	0.03
Place02	Pick02	1103	0.11	1105	0.11	1106	0.06	Pick07	1106	0.08	1108	0.11	1108	0.08
Place03	Pick02	1112	0.09	1114	0.14	1115	0.03	Pick07	1103	0.08	1105	0.11	1106	0.04
Place04	Pick02	1092	0.10	1094	0.13	1095	0.04	Pick07	1092	0.11	1094	0.13	1095	0.11
Place05	Pick02	1124	0.09	1126	0.14	1127	0.13	Pick07	1113	0.08	1115	0.09	1116	0.02
Place01	Pick03	1103	0.08	1106	0.13	1107	0.13	Pick08	1079	0.11	1082	0.14	1083	0.03
Place02	Pick03	1114	0.07	1116	0.14	1118	0.06	Pick08	1091	0.12	1093	0.15	1094	0.08
Place03	Pick03	1118	0.09	1120	0.07	1120	0.01	Pick08	1091	0.10	1093	0.13	1093	0.03
Place04	Pick03	1097	0.07	1100	0.14	1100	0.12	Pick08	1076	0.08	1078	0.09	1078	0.03
Place05	Pick03	1129	0.09	1132	0.14	1133	0.03	Pick08	1098	0.09	1100	0.10	1100	0.04
Place01	Pick04	1109	0.07	1111	0.09	1112	0.06	Pick09	1096	0.08	1098	0.10	1099	0.04
Place02	Pick04	1113	0.09	1115	0.13	1116	0.07	Pick09	1106	0.09	1109	0.15	1109	0.06
Place03	Pick04	1116	0.06	1118	0.12	1119	0.03	Pick09	1108	0.08	1110	0.08	1110	0.04
Place04	Pick04	1095	0.07	1097	0.14	1098	0.06	Pick09	1092	0.07	1094	0.12	1095	0.05
Place05	Pick04	1117	0.09	1120	0.17	1121	0.03	Pick09	1118	0.09	1120	0.12	1120	0.04
Place01	Pick05	1082	0.10	1084	0.09	1085	0.03	Pick10	1089	0.09	1091	0.11	1092	0.05
Place02	Pick05	1088	0.09	1090	0.09	1091	0.01	Pick10	1098	0.10	1101	0.12	1102	0.06
Place03	Pick05	1093	0.08	1095	0.12	1096	0.04	Pick10	1099	0.09	1101	0.12	1102	0.08
Place04	Pick05	1068	0.09	1070	0.08	1071	0.03	Pick10	1087	0.07	1091	0.16	1092	0.15
Place05	Pick05	1095	0.08	1097	0.09	1098	0.03	Pick10	1111	0.10	1113	0.16	1114	0.02

Table 7 ANOVA results

Velocity	Sum of squares	Degrees of freedom	p-value
0.5 m/s	2338.8	2	0.4918
1.0 m/s	583.6	2	0.4931
1.5 m/s	247.4	2	0.5089

Table 9 Kruskal-Wallis results

Velocity	Sum of squares	Degrees of freedom	p-value
0.5 m/s	3209.3	2	0.4273
1.0 m/s	3212.0	2	0.4269
1.5 m/s	3026.5	2	0.4482

therefore, in order to characterize them, 30 independent runs were executed in each experiment, and the value of the best solution, the mean of these 30 solutions, and the standard deviation were obtained.

The quality of the solution first obtained by each algorithm was analyzed in terms of the best solution and the standard deviation of the 30 solutions obtained in each experiment. These results are shown in Tables 4, 5, and 6, for 0.5 m/s, 1 m/s, or 1.5 m/s, respectively. The three algorithms provide satisfactory solutions, the DTSA being the one that obtained the best results and the least good results being provided by the DTLBO algorithm. This conclusion holds when varying both the population size and the stopping criterion, i.e., the maximum number of function evaluations ($\max_{FEs}$).

In order to perform a comparative analysis of the proposed algorithms, an inferential statistical analysis is proposed to statistically determine whether there are significant differences in the behavior of the algorithms analyzed. For this purpose, a parametric method will be used, in particular the ANOVA method, in which case normality and homoscedasticity in the data are assumed, as well as a non-parametric method, in particular the Kruskal-Wallis method, which is suitable when normality or homoscedasticity, or both, are not assumed.

In both cases, the goal is to analyze whether the results allow us to reject the null hypothesis, that is, to refute the conclusion that there are no significant differences between the algorithms. The results obtained with both the ANOVA and Kruskal-Wallis methods can be subjected to a post hoc test, that is, a multiple comparison test to determine the difference between the different groups; for this analysis, the Tukey-Kramer test, also known as the honestly significant Tukey (HSD) method, was used. In both cases, the results obtained are statistically significant if the null hypothesis can be rejected.

The results of applying the ANOVA method to the results of the "Best" column of Tables 4, 5, and 6 are shown in Table 7, while the corresponding results of the Tukey-Kramer comparison analysis are shown in Table 8. The results presented in Table 8 show that the DTSA algorithm is superior to DJAYA and DTLBO and that the DJAYA algorithm is superior to DTLBO, but these results cannot be considered statistically significant because they do not reject the null hypothesis, since the p-value is greater than 0.05 in all cases. The results obtained using the Kruskal-Wallis method are shown in Table 9, and the corresponding results of the Tukey-Kramer comparison analysis are shown in Table 10. The conclusions of the results shown in these two tables are analogous to the previous case, although the DTSA algorithm is dominant, the null hypothesis cannot be rejected. The fact that the null hypothesis cannot be rejected means that the analyses performed do not indicate that the three algorithms behave significantly differently. Note that, as already mentioned, they all give good results and the differences between the proposed solutions are not very significant; in fact, the maximum difference between DTLBO and DTSA is 0.45%.

Table 8 Tukey-Kramer analysis based on ANOVA results

Velocity	Comparison	Mean differences	p-value
0.5 m/s	DTSA - DJAYA	−25.9	0.6706
	DTSA - DTLBO	−28.3	0.4829
	DJAYA - DTLBO	−21.4	0.9520
1.0 m/s	DTSA - DJAYA	−12.9	0.6819
	DTSA - DTLBO	−14.2	0.4805
	DJAYA - DTLBO	−10.8	0.9449
1.5 m/s	DTSA - DJAYA	−8.5	0.6893
	DTSA - DTLBO	−9.4	0.4981
	DJAYA - DTLBO	−7.1	0.9504

Table 10 Tukey-Kramer analysis based on Kruskal-Wallis results

Velocity	Comparison	Mean differences	p-value
0.5 m/s	DTSA - DJAYA	−28.2	0.6424
	DTSA - DTLBO	−31.4	0.4131
	DJAYA - DTLBO	−23.6	0.9267
1.0 m/s	DTSA - DJAYA	−28.0	0.6555
	DTSA - DTLBO	−31.4	0.4091
	DJAYA - DTLBO	−23.8	0.9158
1.5 m/s	DTSA - DJAYA	−27.8	0.6677
	DTSA - DTLBO	−31.1	0.4315
	DJAYA - DTLBO	−23.7	0.9235

Table 11 Analysis of the best solution obtained by the DTSA, DJAYA, and DTLBO algorithms using the Friedman rank test at velocities 0.5 m/s, 1.0 m/s, and 1.5 m/s

Velocity	Mean rank			*p*-value
	DTSA	DJAYA	DTLBO	
0.5 m/s	1.00	2.01	2.99	2.47E-22
1.0 m/s	1.00	2.01	2.99	2.47E-22
1.5 m/s	1.00	2.10	2.90	1.58E-21

Since the previous tests show which of the three algorithms analyzed is the prevalent one, but the statistical significance is not guaranteed because the null hypothesis could not be refuted, it was decided to use the Friedman rank test method, which does not analyze the results obtained, i.e., the results of Tables 4, 5, and 6 are not analyzed directly; these data are preprocessed in order to assign ranks according to the quality of the solution obtained, in our case rank 1 (the best), rank 2, and rank 3 (the worst), and if several algorithms obtain the same solution, they are assigned the average of the ranks they would occupy without the ties. Therefore, this non-parametric inferential statistical test, the Friedman rank test, is applied to these ranks calculated according to the solution, but not to the solution itself, with the objective still being to reject the null hypothesis, i.e., to determine that there is a difference between the groups analyzed.

Table 11 shows the results of the Friedman rank test applied to the best solution for all the velocities analyzed (0.5 m/s, 1.0 m/s, and 1.5 m/s). In this table, we can see that the DTSA algorithm always obtains the best solution, since the average rank is equal to 1, while DJAYA is almost always the second-best algorithm. In addition, the null hypothesis can be rejected since the *p*-value is much lower than 0.05.

Therefore, this analysis also confirms, but with statistically reliable results, that the conclusions drawn are correct and that the DTSA algorithm is the dominant one according to the quality of the solution provided.

We performed an analysis similar to the previous one, based on the Friedman rank test, on the value of the standard deviation of the 30 results obtained in each experiment; the results shown in Table 12 indicate that for all velocities, the best algorithm in terms of standard deviation is DTLBO, followed by the DTSA algorithm. This result could call into question the choice of the predominant algorithm in real-time situations that prevent the 30 scheduled runs from being performed. To clarify this issue, in Table 13, we compare the best result obtained by the DTLBO algorithm of the 30 runs of each experiment with the worst result obtained by the DTSA algorithm of the 30 runs of each experiment. This table shows that the worst DTSA solution always improves the best DTLBO solution, confirming the superiority of the DTSA algorithm.

Table 12 Analysis of the standard deviation obtained by the DTSA, DJAYA, and DTLBO algorithms using the Friedman rank test at velocities 0.5 m/s, 1.0 m/s, and 1.5 m/s

Velocity	Mean rank			*p*-value
	DTSA	DJAYA	DTLBO	
0.5 m/s	1.88	2.93	1.19	1.45E-17
1.0 m/s	1.96	2.87	1.17	6.30E-17
1.5 m/s	2.00	2.85	1.15	3.06E-17

The algorithms were also analyzed in terms of computational cost; the results of the computational cost and the mean value of the 30 independent solutions of the results of Table 4 are shown in Table 14, and the computing platform used is described in Sect. 4. These results also show the predominance of the DTSA algorithm, regarding the quality of the solutions, when comparing the "Mean of solutions" value among the algorithms, row by row. On the other hand, the fastest algorithm is the JAYA algorithm, while the TLBO is the most computationally expensive, mainly due to the processes of converting the non-feasible candidates into feasible ones. This processing was not necessary in the other two algorithms. Although any of the three analyzed algorithms can be used effectively in the proposed system, the DTSA algorithm was the one used in the rest of the experiments because it offered both the most balanced performance and the best solutions.

4 Experimental results

This section describes the analysis of the improvements that our proposal introduces into current industrial processes. Computational experiments were performed using one computing resource, out of the twenty available in the Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz processor; the computing system had 256GB of RAM installed. Both the development of the simulator and the experimental system were developed in C language, and the GCC compiler version 8.5.0 was used.

4.1 Software system performance analysis

Tables 15, 16, and 17 show the pick-and-place time cost of the solution obtained by our proposal using the DTSA algorithm, the time cost of the procedure using the same order as the cutting process, i.e., non-optimized order, both using the same robotized system. These tables also show the percentage improvement of the solution obtained by the proposed system over the non-optimized solution, i.e., the same picking sequence as in the cutting procedure.

These results demonstrate the efficiency of the proposed system and that the automation of the pick-and-place process

Table 13 Comparison of best DTLBO solution vs worst DTSA solution for 30 independent runs, population size equal to 60, $\max_{FEs} = 500,000$ and velocities 0.5 m/s, 1.0 m/s, and 1.5 m/s

Place	Pick	0.5 m/s		1.0 m/s		1.5 m/s		Pick	0.5 m/s		1.0 m/s		1.5 m/s	
Place	Pick	Best DTLBO	Worst DTSA	Best DTLBO	Worst DTSA	Best DTLBO	Worst DTSA	Pick	Best DTLBO	Worst DTSA	Best DTLBO	Worst DTSA	Best DTLBO	Worst DTSA
Place01	Pick01	2874	2866	1512	1508	1091	1089	Pick06	2921	2912	1535	1531	1107	1104
Place02	Pick01	2888	2877	1519	1513	1096	1092	Pick06	2939	2930	1544	1540	1113	1110
Place03	Pick01	2895	2888	1522	1519	1098	1096	Pick06	2946	2937	1548	1543	1115	1112
Place04	Pick01	2845	2835	1498	1493	1082	1078	Pick06	2870	2860	1510	1505	1090	1087
Place05	Pick01	2941	2932	1545	1541	1114	1111	Pick06	2989	2980	1570	1565	1130	1127
Place01	Pick02	2903	2895	1527	1522	1101	1098	Pick07	2888	2879	1519	1514	1096	1093
Place02	Pick02	2918	2909	1534	1529	1106	1103	Pick07	2924	2916	1537	1533	1108	1105
Place03	Pick02	2946	2937	1548	1543	1115	1112	Pick07	2917	2909	1533	1530	1106	1103
Place04	Pick02	2884	2875	1517	1512	1095	1091	Pick07	2883	2875	1517	1512	1094	1092
Place05	Pick02	2981	2971	1565	1560	1127	1124	Pick07	2948	2940	1549	1545	1116	1113
Place01	Pick03	2919	2908	1534	1529	1106	1103	Pick08	2848	2838	1499	1494	1083	1079
Place02	Pick03	2952	2942	1551	1546	1117	1114	Pick08	2882	2872	1516	1511	1094	1091
Place03	Pick03	2961	2954	1556	1552	1120	1118	Pick08	2880	2872	1515	1511	1093	1091
Place04	Pick03	2899	2891	1525	1521	1100	1097	Pick08	2835	2827	1492	1488	1078	1076
Place05	Pick03	3000	2989	1575	1569	1133	1130	Pick08	2901	2894	1526	1522	1100	1098
Place01	Pick04	2935	2928	1543	1539	1112	1109	Pick09	2896	2888	1523	1519	1099	1096
Place02	Pick04	2948	2939	1549	1544	1116	1113	Pick09	2928	2919	1539	1534	1109	1106
Place03	Pick04	2956	2947	1553	1549	1119	1116	Pick09	2931	2924	1540	1537	1110	1108
Place04	Pick04	2894	2885	1522	1517	1098	1095	Pick09	2885	2876	1517	1513	1095	1092
Place05	Pick04	2962	2951	1556	1550	1121	1117	Pick09	2960	2952	1555	1551	1120	1117
Place01	Pick05	2855	2846	1502	1498	1085	1082	Pick10	2876	2868	1513	1509	1092	1089
Place02	Pick05	2873	2864	1512	1507	1091	1088	Pick10	2907	2895	1529	1523	1102	1098
Place03	Pick05	2889	2879	1519	1514	1096	1093	Pick10	2905	2896	1527	1523	1102	1099
Place04	Pick05	2812	2805	1481	1478	1071	1068	Pick10	2875	2862	1512	1506	1092	1087
Place05	Pick05	2894	2886	1522	1518	1098	1095	Pick10	2943	2933	1546	1542	1114	1111

Table 14 Computational cost and mean of solutions for DTSA, DJAYA, and DTLBO algorithms with 30 independent runs, a population size equal to 60, $\max_{FEs} = 500,000$ and velocity 0.5 m/s

Place	Pick	DTSA		DJAYA		DTLBO		Pick	DTSA		DJAYA		DTLBO	
		Computational Cost (s.)	Mean of solutions	Computational Cost (s.)	Mean of solutions	Computational Cost (s.)	Mean of solutions		Computational Cost (s.)	Mean of solutions	Computational Cost (s.)	Mean of solutions	Computational Cost (s.)	Mean of solutions
Place01	Pick01	495.4	2866	463.3	2872	503.2	2875	Pick06	462.6	2912	465.9	2918	517.4	2921
Place02	Pick01	495.0	2877	460.7	2885	525.6	2889	Pick06	462.3	2930	463.9	2936	511.1	2939
Place03	Pick01	495.2	2888	463.3	2893	520.4	2895	Pick06	460.1	2937	459.2	2943	525.8	2946
Place04	Pick01	495.9	2835	468.2	2842	517.3	2846	Pick06	461.5	2860	467.5	2867	527.4	2870
Place05	Pick01	495.4	2932	467.1	2938	533.3	2941	Pick06	461.7	2980	458.4	2987	527.6	2990
Place01	Pick02	496.9	2895	467.9	2901	516.9	2904	Pick07	460.8	2879	466.2	2886	526.2	2888
Place02	Pick02	496.2	2909	465.5	2915	525.4	2918	Pick07	460.8	2916	462.9	2922	522.0	2925
Place03	Pick02	497.2	2937	461.3	2943	526.6	2946	Pick07	460.3	2909	465.2	2915	534.6	2917
Place04	Pick02	496.5	2875	464.8	2881	531.3	2884	Pick07	462.0	2875	463.6	2881	502.9	2884
Place05	Pick02	496.6	2971	465.2	2978	525.6	2982	Pick07	461.7	2940	462.2	2946	520.9	2948
Place01	Pick03	496.0	2908	463.8	2916	521.9	2920	Pick08	462.7	2838	465.8	2846	513.2	2848
Place02	Pick03	496.3	2942	460.8	2949	522.7	2953	Pick08	462.7	2872	460.6	2879	522.5	2883
Place03	Pick03	496.3	2954	466.9	2959	529.4	2961	Pick08	462.9	2872	465.1	2878	528.7	2880
Place04	Pick03	496.0	2891	463.9	2898	508.8	2900	Pick08	414.6	2827	461.5	2833	520.6	2835
Place05	Pick03	496.2	2989	463.2	2996	523.8	3000	Pick08	410.2	2894	466.2	2900	531.3	2901
Place01	Pick04	496.1	2928	460.0	2933	507.4	2936	Pick09	410.8	2888	466.3	2894	522.9	2896
Place02	Pick04	496.3	2939	464.0	2945	509.8	2949	Pick09	410.6	2919	462.5	2925	529.0	2928
Place03	Pick04	496.0	2947	461.0	2954	527.6	2957	Pick09	410.8	2924	463.4	2929	535.8	2931
Place04	Pick04	496.2	2885	462.9	2892	534.3	2895	Pick09	410.0	2876	465.6	2882	526.0	2885
Place05	Pick04	496.2	2951	461.7	2958	514.1	2963	Pick09	409.2	2952	462.5	2958	519.6	2960
Place01	Pick05	495.1	2846	465.5	2852	518.1	2855	Pick10	409.2	2868	469.9	2874	528.5	2876
Place02	Pick05	489.0	2864	462.9	2871	518.6	2873	Pick10	409.0	2895	462.4	2902	512.7	2907
Place03	Pick05	462.6	2879	459.0	2886	531.6	2889	Pick10	407.7	2896	457.1	2903	524.6	2905
Place04	Pick05	462.6	2805	464.3	2811	512.2	2813	Pick10	408.0	2862	462.7	2871	523.4	2875
Place05	Pick05	462.9	2886	462.6	2891	511.6	2894	Pick10	408.5	2933	464.2	2940	520.2	2943

Table 15 Improvement of our proposal with respect to the non-optimized order with 30 independent runs, a population size equal to 60, $\max_{FEs} = 500,000$ and velocity 0.5 m/s

Place	Pick	Proposal (s.)	Non-opt. (s.)	Improv.(%)	Pick	Proposal (s.)	Non-opt. (s.)	Improv.(%)
Place01	Pick01	2866	3593	20.2%	Pick06	2912	3589	18.9%
Place02	Pick01	2877	3582	19.7%	Pick06	2930	3599	18.6%
Place03	Pick01	2888	3645	20.8%	Pick06	2937	3635	19.2%
Place04	Pick01	2835	3545	20.0%	Pick06	2860	3532	19.0%
Place05	Pick01	2932	3658	19.8%	Pick06	2980	3658	18.5%
Place01	Pick02	2895	3575	19.0%	Pick07	2879	3576	19.5%
Place02	Pick02	2909	3587	18.9%	Pick07	2916	3603	19.1%
Place03	Pick02	2936	3643	19.4%	Pick07	2909	3633	19.9%
Place04	Pick02	2874	3557	19.2%	Pick07	2874	3561	19.3%
Place05	Pick02	2971	3653	18.7%	Pick07	2940	3651	19.5%
Place01	Pick03	2907	3600	19.2%	Pick08	2838	3544	19.9%
Place02	Pick03	2942	3629	18.9%	Pick08	2872	3563	19.4%
Place03	Pick03	2954	3672	19.6%	Pick08	2872	3595	20.1%
Place04	Pick03	2892	3577	19.2%	Pick08	2827	3528	19.9%
Place05	Pick03	2989	3677	18.7%	Pick08	2894	3591	19.4%
Place01	Pick04	2928	3590	18.5%	Pick09	2888	3555	18.8%
Place02	Pick04	2939	3592	18.2%	Pick09	2919	3587	18.6%
Place03	Pick04	2947	3639	19.0%	Pick09	2924	3629	19.4%
Place04	Pick04	2885	3534	18.4%	Pick09	2876	3554	19.1%
Place05	Pick04	2951	3626	18.6%	Pick09	2952	3634	18.8%
Place01	Pick05	2846	3540	19.6%	Pick10	2868	3555	19.3%
Place02	Pick05	2865	3528	18.8%	Pick10	2895	3579	19.1%
Place03	Pick05	2879	3578	19.5%	Pick10	2897	3605	19.7%
Place04	Pick05	2806	3492	19.7%	Pick10	2862	3550	19.4%
Place05	Pick05	2885	3574	19.3%	Pick10	2933	3614	18.8%

Table 16 Improvement of our proposal with respect to the non-optimized order with 30 independent runs, a population size equal to 60, $\max_{FEs} = 500,000$ and velocity 1 m/s

Place	Pick	Proposal (s.)	Non-opt. (s.)	Improv.(%)	Pick	Proposal (s.)	Non-opt. (s.)	Improv.(%)
Place01	Pick01	1508	1871	19.4%	Pick06	1531	1870	18.1%
Place02	Pick01	1513	1866	18.9%	Pick06	1540	1875	17.9%
Place03	Pick01	1519	1897	20.0%	Pick06	1543	1892	18.4%
Place04	Pick01	1492	1848	19.2%	Pick06	1505	1841	18.3%
Place05	Pick01	1541	1904	19.1%	Pick06	1565	1904	17.8%
Place01	Pick02	1522	1862	18.3%	Pick07	1514	1863	18.7%
Place02	Pick02	1529	1869	18.2%	Pick07	1533	1876	18.3%
Place03	Pick02	1543	1896	18.6%	Pick07	1530	1891	19.1%
Place04	Pick02	1512	1854	18.4%	Pick07	1512	1856	18.5%
Place05	Pick02	1561	1902	17.9%	Pick07	1545	1900	18.7%
Place01	Pick03	1529	1875	18.4%	Pick08	1494	1847	19.1%
Place02	Pick03	1546	1889	18.2%	Pick08	1511	1857	18.6%

Table 16 continued

Place	Pick	Proposal (s.)	Non-opt. (s.)	Improv.(%)	Pick	Proposal (s.)	Non-opt. (s.)	Improv.(%)
Place03	Pick03	1552	1911	18.8%	Pick08	1511	1872	19.3%
Place04	Pick03	1521	1864	18.4%	Pick08	1488	1839	19.1%
Place05	Pick03	1569	1913	18.0%	Pick08	1522	1871	18.6%
Place01	Pick04	1539	1870	17.7%	Pick09	1519	1852	18.0%
Place02	Pick04	1545	1871	17.4%	Pick09	1535	1869	17.9%
Place03	Pick04	1549	1894	18.3%	Pick09	1537	1890	18.7%
Place04	Pick04	1517	1842	17.6%	Pick09	1513	1852	18.3%
Place05	Pick04	1550	1888	17.9%	Pick09	1551	1892	18.0%
Place01	Pick05	1498	1845	18.8%	Pick10	1509	1853	18.6%
Place02	Pick05	1507	1839	18.1%	Pick10	1523	1864	18.3%
Place03	Pick05	1514	1864	18.8%	Pick10	1523	1878	18.9%
Place04	Pick05	1478	1821	18.9%	Pick10	1506	1850	18.6%
Place05	Pick05	1518	1862	18.5%	Pick10	1542	1882	18.1%

Table 17 Improvement of our proposal with respect to the non-optimized order with 30 independent runs, a population size equal to 60, $\max_{FEs} = 500,000$ and velocity 1.5 m/s

Place	Pick	Proposal (s.)	Non-opt. (s.)	Improv.(%)	Pick	Proposal (s.)	Non-opt. (s.)	Improv.(%)
Place01	Pick01	1089	1331	18.2%	Pick06	1104	1330	17.0%
Place02	Pick01	1092	1327	17.7%	Pick06	1110	1333	16.7%
Place03	Pick01	1096	1348	18.7%	Pick06	1112	1345	17.3%
Place04	Pick01	1078	1315	18.0%	Pick06	1087	1311	17.1%
Place05	Pick01	1111	1353	17.9%	Pick06	1127	1353	16.7%
Place01	Pick02	1098	1325	17.1%	Pick07	1093	1325	17.5%
Place02	Pick02	1103	1329	17.0%	Pick07	1106	1334	17.1%
Place03	Pick02	1112	1348	17.5%	Pick07	1103	1344	17.9%
Place04	Pick02	1092	1319	17.3%	Pick07	1092	1320	17.3%
Place05	Pick02	1124	1351	16.8%	Pick07	1113	1350	17.5%
Place01	Pick03	1103	1333	17.3%	Pick08	1079	1315	17.9%
Place02	Pick03	1114	1343	17.0%	Pick08	1091	1321	17.4%
Place03	Pick03	1118	1357	17.6%	Pick08	1091	1332	18.1%
Place04	Pick03	1097	1326	17.2%	Pick08	1076	1309	17.9%
Place05	Pick03	1129	1359	16.9%	Pick08	1098	1330	17.5%
Place01	Pick04	1109	1330	16.6%	Pick09	1096	1318	16.9%
Place02	Pick04	1113	1331	16.4%	Pick09	1106	1329	16.8%
Place03	Pick04	1116	1346	17.1%	Pick09	1108	1343	17.5%
Place04	Pick04	1095	1311	16.5%	Pick09	1092	1318	17.1%
Place05	Pick04	1117	1342	16.8%	Pick09	1118	1345	16.9%
Place01	Pick05	1082	1313	17.6%	Pick10	1089	1318	17.4%
Place02	Pick05	1088	1309	16.9%	Pick10	1098	1326	17.2%
Place03	Pick05	1093	1326	17.6%	Pick10	1099	1335	17.7%
Place04	Pick05	1068	1297	17.7%	Pick10	1087	1317	17.4%
Place05	Pick05	1095	1325	17.3%	Pick10	1111	1338	16.9%

Fig. 7 Frames extracted from the video sequence of the implementation

with a robotic system requires the simulation and optimization of this system, in order to efficiently exploit the economic investment made in the automation of the production lines.

As explained in Sect. 2.2.2, the optimization algorithms were designed by introducing two solutions in the initial population that, although not optimal, allowed improvement of the heuristic search process. Moreover, in the actual implementation of the system, the stopping criterion used was the available computation time instead of the number of function evaluations. This time depends on the production line and is the time at which the nesting information provided to the CNC cutting machine can be accessed, in particular, as well as the processing and transport times of the material through the production line. It is guaranteed that the system will always improve results with respect to using the same picking sequence as in the cutting procedure and that the automation introduced will not generate undesirable downtime on the production line.

4.2 Analysis of the entire system operation

To validate the data obtained in the simulation and as an initial part of the real implementation of the system, a simplified setup was assembled to enable testing. The robot used in this case was a Universal Robot UR5. This robot was selected for safety reasons because, in the development tests, it allowed the elimination of safety barriers, as it is a collaborative robot. Figure 7 shows a sequence of frames from the video available at this site https://youtu.be/GJxV3TP_N2E (accessed on 08 February 2023). The size of the leather and the number of pieces used in this case were reduced, since the seventh axis has not been implemented and the robot's workspace is

smaller. In any case, it can be seen that the operation of the different hardware and software modules developed works satisfactorily.

The system developed is part of a multidisciplinary project whose objective is to design an optimal industrial process, both in terms of automated manipulation of the textile pieces considered, and in terms of accelerating this manipulation to the maximum by means of computational techniques. From an academic point of view, this work has shown the necessity of a detailed analysis of the computational problem, avoiding simplifications that lead to non-optimal solutions and that show divergences in the results obtained from simulations. From the practitioner's point of view, it is shown that investments in R&D&I lead to cost savings in many aspects, such as savings in raw materials, energy, and/or industrial infrastructure. The work presented is an example of this saving, which amortizes the investment made and can be used as a reference in the field of industrial engineering, both in the field of machinery and tools and in the optimization of processes through simulation and computational optimization. In addition, these industrial savings are necessary to increase the productivity of manufacturing processes, with the aim of bringing factories closer to consumers. Deglobalization offers clear benefits in terms of reducing transportation costs and time to market, but it also presents a challenge in terms of increasing the productivity of manufacturing systems.

5 Conclusions

Two main conclusions were reached in this project. On the one hand, the system can be automated, which allows the elimination of a workstation, thus increasing the competitiveness of the production line. Tasks can be performed by one operator and one robot instead of two operators. Currently, the CNC machine is fed to process shoe parts of the same shoe size for each leather piece since; for some parts, the difference in size among parts of different shoe sizes is so small that operators end up making mistakes in the classification, usually caused by fatigue. Introducing pieces of different shoe sizes in the same leather piece forces the operator to pay more attention and, therefore, slows down the process. Therefore, automation in the pick-and-place stage brings several benefits. Firstly, a reduction in waste materials can be achieved because of the possibility of including parts of different sizes in the same hide to be cut, which can optimize the nesting process. Secondly, the time reduction at the pick-and-place stage allows us to consider a new design of the cutting plant so that the same pick-and-place station can be fed by different cutting machines.

On the other hand, as we demonstrate in this work, the use of a discrete optimization algorithm allows us to get the appropriate sequence to perform the pick-and-place operation in a way that reduces operation time and allows us to benefit from the advantages mentioned above. Specifically, although the use of the other algorithms analyzed also introduces a substantial improvement, the DTSA algorithm was selected because it provides pick-and-place sequences with an improvement of between 15 and 20%, with respect to performing the same operation following a raster order sequence.

In this work we have considered industrial processes with not extremely demanding time requirements; these requirements are determined by the previous processes in the production line. However, in very powerful industrial plants, these time requirements may not be met, which is an important limitation to be solved in the future. Therefore, if the available time slot does not allow to obtain an acceptable near-optimal solution, the computational process should be accelerated by means of parallelization techniques, initially using the same existing computational resources, i.e., the available multicore systems.

If a higher speed of part processing per unit time is required, one solution is to increase the number of robot arms performing the pick-and-place operation. It should be noted, however, that in this case, a shared workspace for several robots has not been considered, since no collision avoidance techniques between robot arms have been integrated. This is one of the main limitations of the research presented, since if a higher processing speed of parts per unit of time is required, it would be necessary to increase the space required for processing; one of the future research lines is to adapt the algorithms to avoid such an increase in the space required in the production line.

As a future work, using techniques based on the acquisition and analysis of hyperspectral images, we intend to extend this work to the field of pick-and-place, after real-time sorting, of industrial solid waste. In this case, the industrial pick-and-place process will still exist, but we will be dealing with a continuous production model rather than discrete raw materials, which will have to be computationally discretized to maximize the speed of the industrial process.

Acknowledgements The authors want to thank Simplicity COMELZ España http://comelz.es (accessed on 29 January 2023) and DESINOPE https://desinope.com (accessed on 29 January 2023) for providing ideas and collaborating with this project. This work would not have been possible without the help of CFZ Cobots https://cfzcobots.com (accessed on 29 January 2023), which provided the software, hardware, and technical support necessary to make this project possible.

Author Contributions All authors contributed to the study conception, design, development, material preparation, data collection, results analysis, and writing of the article and its revision.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This publication is part of the project CPP2021-008593, funded by MCIN/AEI/10.13039/501100011033 and by the European Union-NextGenerationEU/PRTR. This research was

also partially supported by the Spanish Ministry of Science and Innovation and the Research State Agency under Grant PID2021-123627OB-C55 co-financed by FEDER funds (MCIN/AEI/FEDER/UE).

Availability of data and materials Test data set available at http://atc.umh.es/gatcom/Place5Pick10_2022.zip.

Code availability Available upon request to the corresponding author.

Declarations

Ethics approval Not applicable

Consent to participate Not applicable

Consent for publication All the authors have read and agreed to the published version of the manuscript.

Conflict of interest The authors declare no competing interests

References

1. Gao J, Zhu X, Liu A, Meng Q, Zhang R (2018) An iterated hybrid local search algorithm for pick-and-place sequence optimization. Symmetry 10(633). https://doi.org/10.3390/sym10110633
2. Alazzam AR (2018) Using BUA algorithm to solve a sequential pick and place problem. In: 2018 international conference on information and computer technologies (ICICT). pp 144–149. https://doi.org/10.1109/INFOCT.2018.8356858
3. Golberg DE (1989) Genetic algorithms in search, optimization, and machine learning
4. Bessonnet G, Lallemand JP (1990) Optimal trajectories of robot arms minimizing constrained actuators and travelling time. In: Proceedings IEEE International conference on robotics and automation vol 1. pp 112–117. https://doi.org/10.1109/ROBOT.1990.125956
5. Bobrow JE, Dubowsky S, Gibson JS (1985) Time-optimal control of robotic manipulators along specified paths. Int J Robot Res 4(3):3–17. https://doi.org/10.1177/027836498500400301
6. Zhang K, Yuan CM, Gao XS, Li H (2012) A greedy algorithm for feedrate planning of CNC machines along curved tool paths with confined jerk. Robot Comput Integr Manuf 28(4):472–483. https://doi.org/10.1016/j.rcim.2012.02.006
7. Zhang Q, Li SR (2013) Efficient computation of smooth minimum time trajectory for CNC machining. Int J Adv Manuf Technol 68(1):683–692. https://doi.org/10.1007/s00170-013-4790-7
8. Zhang Q, Li SR, Gao XS (2013) Practical smooth minimum time trajectory planning for path following robotic manipulators. In: 2013 American control conference. pp 2778–2783. https://doi.org/10.1109/ACC.2013.6580255
9. Zhang Q, Zhao MY (2016) Minimum time path planning of robotic manipulator in drilling/spot welding tasks. J Computat Des Eng 3(2):132–139. https://doi.org/10.1016/j.jcde.2015.10.004
10. Qu L, Sun R (1999) A synergetic approach to genetic algorithms for solving traveling salesman problem. Inf Sci 117(3):267–283. https://doi.org/10.1016/S0020-0255(99)00026-2
11. Huang T, Wang PF, Mei JP, Zhao XM, Chetwynd DG (2007) Time minimum trajectory planning of a 2-DOF translational parallel robot for pick-and-place operations. CIRP Ann 56(1):365–368. https://doi.org/10.1016/j.cirp.2007.05.085
12. Aminzadeh V, Wurdemann H, Dai JS, Reed J, Purnell G (2010) A new algorithm for pick-and-place operation. Ind Robot Int J 37(6):527–531. https://doi.org/10.1108/01439911011081678
13. Borrell Méndez J, Perez-Vidal C, Segura Heras JV, Pérez-Hernández JJ (2020) Robotic pick-and-place time optimization: application to footwear production. IEEE Access 8:209428–209440. https://doi.org/10.1109/ACCESS.2020.3037145
14. Ayob M, Kendall G (2005) A triple objective function with a Chebychev dynamic pick-and-place point specification approach to optimise the surface mount placement machine. Eur J Oper Res 164(3):609–626. https://doi.org/10.1016/j.ejor.2003.09.034. Recent Advances in Scheduling in Computer and manufacturing Systems
15. Gecks T, Henrich D (2005) Human-robot cooperation: safe pick-and-place operations. In: ROMAN 2005. IEEE International workshop on robot and human interactive communication, 2005. pp 549–554. https://doi.org/10.1109/ROMAN.2005.1513837
16. Daoud S, Chehade H, Yalaoui F, Amodeo L (2014) Efficient metaheuristics for pick and place robotic systems optimization. J Intell Manuf 25:27–41. https://doi.org/10.1007/s10845-012-0668-z
17. Gunduz M, Aslan M (2021) DJAYA: a discrete Jaya algorithm for solving traveling salesman problem. Appl Soft Comput 105. https://doi.org/10.1016/j.asoc.2021.107275
18. Wu L, Zoua F, Chen D (2017) Discrete teaching-learning-based optimization algorithm for traveling salesman problems. MATEC Web Conf 128:02022. https://doi.org/10.1051/matecconf/201712802022
19. Cinar AC, Korkmaz S, Kiran MS (2020) A discrete tree-seed algorithm for solving symmetric traveling salesman problem. Eng Sci Technol Int J 23(4):879–890. https://doi.org/10.1016/j.jestch.2019.11.005
20. Mahi M, Baykan ÖK, Kodaz H (2015) A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem. Appl Soft Comput 30:484–490. https://doi.org/10.1016/j.asoc.2015.01.068
21. Shi XH, Liang YC, Lee HP, Lu C, Wang QX (2007) Particle swarm optimization based algorithms for TSP and generalized TSP. Inf Process Lett 103(5):169–176. https://doi.org/10.1016/j.ipl.2007.03.010
22. Aslan M, Baykan NA (2016) A performance comparison of graph coloring algorithms. Int J Intell Syst Appl Eng 4(Special Issue–1):1–7. https://doi.org/10.18201/ijisae.273053
23. Sayadi MK, Hafezalkotob A, Naini SGJ (2013) Firefly-inspired algorithm for discrete optimization problems: an application to manufacturing cell formation. J Manuf Syst 32(1):78–84. https://doi.org/10.1016/j.jmsy.2012.06.004
24. Geem ZW (2005) Harmony search in water pump switching problem. In: Wang L, Chen K, Ong YS (eds) Advances in natural computation. Springer, Berlin, Heidelberg, pp 751–760. https://doi.org/10.1007/11539902_92

25. Kiran MS (2015) TSA: tree-seed algorithm for continuous optimization. Expert Syst Appl 42(19):6686–6698. https://doi.org/10.1016/j.eswa.2015.04.055
26. Rao RV (2016) Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems. Int J Ind Eng Comput 7:19–34. https://doi.org/10.5267/j.ijiec.2015.8.004
27. Rao RV, Savsani VJ, Vakharia DP (2011) Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 43(3):303–315. https://doi.org/10.1016/j.cad.2010.12.015
28. Papadimitriou CH, Steiglitz K (1976) Some complexity results for the traveling salesman problem. In: Proceedings of the Eighth annual ACM symposium on theory of computing, pp 1–9. https://doi.org/10.1145/800113.803625
29. Garey MR, Johnson DS (1978) "Strong"NP completeness results: motivation, examples, and implications. J ACM (JACM) 25(3):499–508. https://doi.org/10.1145/322077.322090
30. Safra S, Schwartz O (2006) On the complexity of approximating TSP with neighborhoods and related problems. Comput Complex 14:281–307. https://doi.org/10.1007/s00037-005-0200-3

Authors and Affiliations

Francisco José Martínez-Peral[1] · Héctor Migallón[2] · Jorge Borrell-Méndez[1] · Miguel Martínez-Rach[2] · Carlos Pérez-Vidal[1]

Francisco José Martínez-Peral
francisco.martinezp@umh.es

Jorge Borrell-Méndez
jorge.borrell@goumh.umh.es

Miguel Martínez-Rach
mmrach@umh.es

Carlos Pérez-Vidal
carlos.perez@umh.es

[1] Instituto de Investigación en Ingeniería I3E, Miguel Hernández University, Av. de la Universidad s/n, Elche 03202, Alicante, Spain

[2] Department of Computer Engineering, Miguel Hernández University, Av. de la Universidad s/n, Elche 03202, Alicante, Spain