Advances in Science and Technology Research Journal

Advances in Science and Technology Research Journal, 2027, 21(1), 225–240
https://doi.org/10.12913/22998624/231504
ISSN 2299-8624, License CC-BY 4.0

Received: 2026.07.22
Accepted: 2026.11.15
Published: 2026.12.01

Dynamic task scheduling for robotic pick-and-place systems using sequential planning with a decision horizon

Bartosz Poskart[1], Kamil Krot[1*], Jakub Mazur[1]

[1] Department of Production Engineering and Management, Faculty of Mechanical Engineering, Wroclaw University of Science and Technology, Wroclaw, Poland
* Corresponding author's e-mail: kamil.krot@pwr.edu.pl

ABSTRACT

This paper presented a comparative analysis of algorithms for planning the pick sequence in a robotic sorting process involving objects moving on a conveyor belt. The primary objective was to determine the effects of decision-horizon length on computation time and solution quality and to evaluate the effects of conveyor speed and the number of sorted classes on capture rate (CR). A custom simulation environment was developed in Python using a simplified kinematic model of a delta robot. The analysis included exhaustive search, Branch-and-Bound, a time-bounded Monte Carlo method, and the FIFO, SPT, and NNF heuristics. Under the investigated conditions, factorial growth limited the computationally feasible exhaustive-search horizon to seven objects, whereas B&B extended exact planning to 13 objects with a computation time of 1.53 s. Monte Carlo was evaluated over 100 independent runs per horizon using a 1 s computation-time budget. All 700 runs achieved the same maximum number of seven successful picks as B&B, although the probability of reproducing the complete B&B optimum generally decreased for longer horizons. For the 20-object horizon, the mean and maximum increases in sequence execution time were 4.12% and 6.47%, respectively, while the planning time decreased from 27.29 s for B&B to 1 s for Monte Carlo. In the experiment comprising 5.000 objects, B&B achieved the highest CR in 38 of the 56 scenarios, whereas FIFO and SPT achieved the same or higher CR under lower process loads. These results provide a basis for selecting the planning algorithm according to the decision horizon and current process conditions.

Keywords: dynamic pick-and-place, robotic task scheduling, pick sequence optimisation, decision horizon, conveyor tracking, combinatorial optimisation.

INTRODUCTION

Dynamic pick-and-place processes are widely used for sorting, packaging, and handling products moving on conveyor belts [1,3]. The performance of such systems depends not only on the robot's kinematic parameters but also on the order in which pick operations are executed. Selecting the nearest object or the object that can be handled in the shortest time may be locally advantageous, but it does not necessarily lead to the best sequence when several subsequent objects are considered during planning [1,8]. This issue becomes particularly important at high conveyor speeds and when sorting multiple object classes, especially when the system operates close to its throughput limit or when the robot is unable to handle all incoming objects, and the objective is to maximise the number of successfully completed picks [1,3,4]. Considering a larger number of future pick operations may improve planning quality; however, it also leads to a rapid increase in the space of feasible sequences and the associated computation time [1,3]. This creates a trade-off between the number of subsequent objects included in pick-sequence planning, the quality of the resulting solution, and the time available for decision-making. Numerous methods for sequencing robotic tasks have been reported in the literature; nevertheless, the practical applicability limits of individual methods with respect to decision horizon length and

changing process conditions remain insufficiently investigated [6,8,13,14].

The objective of this study was to investigate the effect of decision horizon length on computation time and pick-sequence quality in a dynamic sorting process. Exhaustive search, Branch-and-Bound, a time-bounded Monte Carlo method, and the FIFO, SPT, and NNF heuristics were compared. Their impact on capture rate (CR) was subsequently evaluated for different conveyor speeds and numbers of sorted classes, while picks per minute (PPM) was treated as a complementary throughput indicator. The main contribution of this work is the identification of a practical trade-off between planning scope and computational cost, as well as the development of an algorithm selection matrix adapted to the current process conditions.

RELATED WORK

Dynamic robotic pick-and-place on moving conveyors

In robotic pick-and-place processes performed on a moving conveyor, the position of each object changes both while the decision is being computed and during robot motion. Each object remains available for picking only for a limited period determined by the conveyor speed and the dimensions of the robot's workspace. The planning task, therefore, cannot be reduced to selecting the geometrically nearest object; instead, it requires predicting the object's future position and verifying whether the robot can complete the operation before the object leaves the reachable area. Accordingly, the problem may be interpreted as dynamic operation sequencing with time-dependent object positions, sequence-dependent processing times, and limited availability windows [1–3].

As conveyor speed or object-flow density increases, the time available for decision-making decreases, while the number of objects to be considered simultaneously may increase. This can lead to system overload, in which the robot is unable to handle all incoming objects before they leave the workspace [4, 5]. Under such conditions, simple dispatching rules, such as SPT or FIFO, offer low computational cost but do not guarantee the best sequence. Han et al. demonstrated that a decision that is optimal for a single pick does not necessarily minimise the execution time of the entire sequence, whereas planning over a finite horizon can substantially improve process throughput [1].

The problem becomes even more complex in multi-robot systems, where, in addition to determining the pick sequence, objects must be assigned to individual robots and the locations and timing of the operations must be specified [2, 3]. Nielsen et al. formulated this task as a time-dependent pick-and-place planning problem with time windows, emphasising the need for heuristics capable of generating decisions within the interval between object detection and handling [6]. Previous studies, therefore, indicated that a key challenge lies in balancing the extent to which future picks are anticipated, the quality of the resulting sequence, and the computation time available for online control.

Pick-sequence optimisation methods

Methods for determining the pick sequence can be divided into three main groups. The first group comprises greedy dispatching rules, such as NNF, SPT, FIFO, nearest-object selection, and selection of the operation with the shortest predicted execution time. These methods make decisions based on the current system state without evaluating their full consequences for subsequent operations. Their main advantages are simplicity, determinism, and low computational cost, which make them suitable for the applications requiring short response times. However, a locally advantageous decision does not necessarily lead to the globally best sequence. Han et al. showed that classical greedy rules do not guarantee the minimum sequence execution time [1], whereas Sheng et al. achieved a 20% increase in the throughput of a dual-arm system by replacing FIFO with the STFO (shortest time first out) rule based on the shortest operation time [7].

The second group comprises exact optimisation methods, including exhaustive enumeration over a predefined decision horizon, dynamic programming, mixed-integer linear programming (MILP), binary integer linear programming (BILP), and Branch-and-Bound algorithms. In this context, the decision horizon denotes the fixed number of subsequent objects considered when determining the pick sequence. These methods systematically evaluate multiple candidate sequences and can identify an optimal solution with respect to the adopted objective function. Dynamic programming has been applied

to pick planning on moving conveyors in both single- and dual-robot systems [1, 8], whereas BILP formulations have been used to optimise operation sequences for dual-arm robots while accounting for alternative trajectories and collision risk [9]. However, the applicability of exact methods is constrained by the rapid growth of the solution space as the number of objects, robots, and additional constraints increases. Consequently, guaranteeing optimality may require computation times that exceed the decision time available in a dynamic system.

The third group comprises heuristic and approximate methods, including heuristic-guided search, randomised approaches, and Monte Carlo algorithms. These methods reduce the number of candidate sequences evaluated or explore the solution space only within a predefined time limit, thereby allowing larger numbers of objects to be considered than is typically feasible with exact methods. However, the resulting solution is not guaranteed to be optimal, and in randomised approaches, its quality may vary between runs. Therefore, the selection of a planning method requires a trade-off between solution quality, computation time, and the number of objects included in the planning horizon. Although extending the horizon may improve decision quality, it also increases computational cost. Consequently, in online systems, the most useful method is not necessarily the one that yields the best solution, but rather the one that produces a sufficiently high-quality solution within the available decision time.

The main limitation of randomised methods compared with exact approaches is the lack of guaranteed optimality and result repeatability, as solution quality depends on the number of evaluated candidates and the adopted computation-time limit.

Therefore, the selection of a planning method involves a trade-off between the decision-horizon length, solution quality, and computation time. A longer horizon may improve the resulting sequence and increase the number of captured objects, but it may also prevent the algorithm from producing a decision before the process state changes. Conversely, faster methods provide timely decisions but may select locally favourable sequences that reduce the overall process throughput.

Finite and receding decision horizons

Planning the complete sequence of all currently available and future pick operations may be computationally infeasible, particularly when object positions change over time and decisions must be made within the interval between object detection and departure from the robot workspace [10]. A finite decision horizon reduces the problem size by considering only a predefined number of upcoming operations. Finite-horizon planning can be implemented in two ways. In the classical receding-horizon approach, the plan is recomputed after each executed action [11]. Alternatively, in batch-based planning, the complete sequence is executed before the plan is updated for the next group of objects. The latter approach, adopted in this study, reduces the frequency of replanning, but reacts less frequently to newly detected objects [1].

Extending the decision horizon allows a larger number of future operations to be considered, but it also increases the size and computational complexity of the optimisation problem. Li et al. showed that receding-horizon optimisation can reduce this complexity by decomposing a long-horizon problem into shorter, partially overlapping subproblems, although repeated re-optimisation introduces additional computational overhead [12]. The practically feasible horizon length depends not only on the planning algorithm but also on the conveyor speed, object-flow density, number of sorted classes, and time available for decision-making [4, 5, 10]. Previous studies have demonstrated the benefits of multi-object planning and restricting the scope of the analysed sequence; however, a comparative assessment defining the practical horizon-length limits of exact and approximate methods is still lacking. It also remains unclear how algorithm selection should vary with conveyor speed and the number of sorted classes. This study bridged these gaps by analysing the trade-off between decision-horizon length, computation time, solution quality, and capture rate under variable process conditions.

METHODOLOGY

Problem definition and research framework

The research methodology was designed to evaluate the applicability of selected pick-sequence planning algorithms in a dynamic sorting process involving objects moving on a conveyor belt. The study was conducted in two stages. The first stage examined the effect of decision

horizon length on computation time and solution quality for exhaustive search, Branch-and-Bound, and Monte Carlo methods. Its objective was to determine the maximum horizon length for which a pick sequence could be generated within a time frame suitable for practical implementation. In the second stage, the SPT, FIFO, NNF, Branch-and-Bound, and Monte Carlo algorithms were compared under varying conveyor speeds and different numbers of sorted classes. Performance was assessed in terms of computation time, sorting success rate, picks per minute, and robot utilisation.

The analysed task involved a robot picking objects moving along a conveyor belt and placing them either at a predefined destination or in class-specific containers, depending on the experimental scenario. As the system had access only to information about objects currently detected within the observation area, pick sequences were planned using a batch-based finite decision horizon. The complete sequence generated for batch n was executed without replanning after individual picks. During its execution, the sequence for batch n+1 was computed concurrently using the available and newly detected objects. The overall workflow is illustrated in Figure 1.

The objective of the optimisation was to pick the maximum possible number of objects belonging to the current batch, considering the individual time during which each object remained available within the robot workspace. Among the sequences providing the same maximum number of picks, the sequence with the shortest execution time was selected. Reducing the execution time enabled the robot to begin processing the next batch earlier.

The objective function was defined lexicographically as:

$$lex_{\pi}\max\,[N(\pi), -T(\pi)] \tag{1}$$

where: π – candidate pick sequence containing objects from the current batch;
$N(\pi)$ – number of objects successfully picked when executing sequence π;
$T(\pi)$ – total execution time of sequence π;
lexmax – lexicographic maximisation, in which maximising $N(\pi)$ takes precedence over minimising $T(\pi)$.

Thus, the number of successfully picked objects was the primary optimisation criterion, while the sequence execution time was used as the secondary criterion.

Temporal and spatial constraints

The predicted pick time of each object accounted for the execution times of all preceding pick-and-place operations in the sequence. In addition, the sequence for the next batch had to be determined before completion of the currently executed batch:

$$t_{comp} \leq T_{batch} \tag{2}$$

An object could be picked only if its predicted position at the pick time remained within the robot workspace:

$$p_i\left(t_i^{pick}\right) \in W \tag{3}$$

Prediction of moving-object positions

The future position of object i was predicted from its position at the planning time and the conveyor velocity:

$$p_i(t) = p_i(t_0) + v_c(t - t_0) \tag{4}$$

Pick feasibility condition

A pick was considered feasible if the robot could reach the predicted object position before the object left the robot workspace. Objects that did not satisfy this condition were excluded from the planned sequence. Where t_{comp} – computation time required to determine the sequence for the next batch; T_{batch} – execution time of the currently processed batch; p_i (t) – predicted position of object *i* at time *t*; p_i (t_0) – detected position of object *i* at planning time t_0; v_c – conveyor-velocity vector; t_i^{pick} – predicted pick time of object i, including the execution times of all preceding operations; W – robot workspace.

Simulation environment and robot motion model

A custom simulation environment was developed in Python using the Pygame library. This approach provided full control over the model structure, enabled the implementation of dedicated control algorithms and decision-making logic, and allowed the process to be visualised during execution. The simulation parameters were selected to represent the kinematic characteristics of delta robots, using ABB IRB 360-1/1130 as a reference platform. The model included a 600 mm-wide

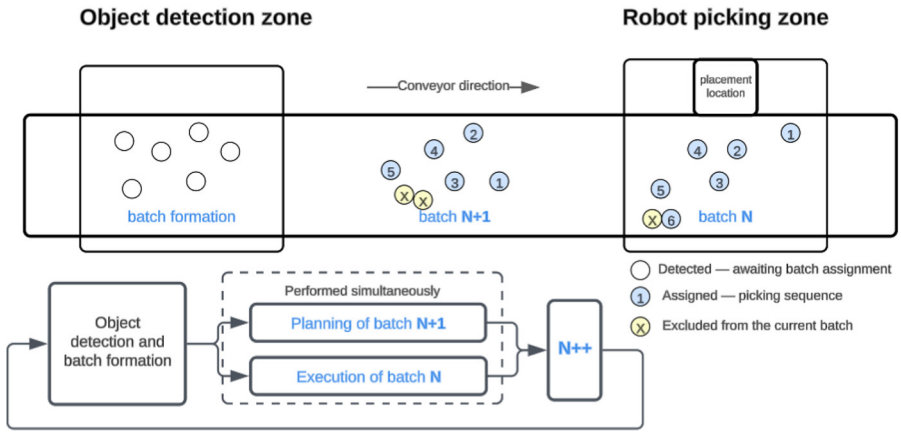


Figure 1. Schematic of batch-based finite-horizon planning, showing object detection, batch formation, and simultaneous execution of batch N and planning of batch N+1

conveyor belt moving at 400 mm/s. The robot workspace measured 800 × 800 mm and covered both a section of the conveyor and the area required to access the placement container. The robot dynamics were represented using a simplified model with a maximum velocity of V_{max} = 7 m/s and a maximum acceleration of a_{max} = 100 m/s^2.

Robot motion was modelled subject to maximum velocity and acceleration constraints. Depending on the length of the commanded trajectory, either a triangular or trapezoidal velocity profile was applied. The minimum travel distance required for the robot to accelerate from rest to v_{max} and subsequently decelerate to rest was determined as follows:

$$d_{min} = \frac{{V_{max}}^2}{a_{max}} \tag{5}$$

For $d_c \geq d_{min}$ a trapezoidal velocity profile was used, whereas for i $d_c < d_{min}$ a triangular velocity profile was applied. Where d_c – total tool travel distance, d_{min} – minimum distance required to accelerate to V_{max} and subsequently decelerate, V_{max} – specified maximum linear velocity of the robot, a_{max} – specified maximum acceleration of the robot.

Total motion time for a trapezoidal velocity profile:

$$t_c = \frac{d_c}{v_{max}} + \frac{V_{max}}{a_{mac}} \text{ (when } \boldsymbol{d_c} \geq \frac{\boldsymbol{V_{max}}^2}{\boldsymbol{a_{max}}}) \tag{6}$$

Total motion time for a triangular velocity profile:

$$t_c = 2\sqrt{\frac{d_c}{a_{max}}} \text{ (when } \boldsymbol{d_c} < \frac{\boldsymbol{V_{max}}^2}{\boldsymbol{a_{max}}}) \tag{7}$$

In the simulation model, object positions on the conveyor were updated based on the conveyor speed and the duration of successive pick-and-place operations. The conveyor continued to move during both robot motion and sequence computation. During planning, the future positions of the objects were predicted based on the conveyor motion and the execution times of the preceding operations. The objects that could not be picked before leaving the robot workspace were considered infeasible and excluded from the planned sequence. Consequently, each recomputation of the pick sequence accounted for the current positions of all available objects. The model did not include the constraints arising from detailed trajectory planning within the robot workspace or the full manipulator dynamics. Grasping and release times were intentionally omitted, as the primary objective of the study was to compare pick-sequence planning algorithms, rather than to reproduce the complete pick-and-place cycle of a physical robot with high fidelity.

All algorithms evaluated in the experiments were implemented in Python 3.13 and executed on an industrial computer equipped with an Intel Core i7-8665U processor and 16 GB of RAM. Unless otherwise stated, the runtime measurements of the deterministic algorithms were repeated ten times, and the arithmetic mean was reported. Ten repetitions were considered sufficient because these algorithms returned the same solution for identical input data, and the repetitions were used only to reduce the influence of transient system-load variations while limiting computational cost. Because the Monte Carlo method is stochastic, its statistical evaluation was based on 100 independent runs for

each decision-horizon length. Each Monte Carlo run used a computation-time budget of 1 s and generated a new set of random candidate permutations. The resulting values were used primarily for relative performance comparison under the same hardware and software environment. Accordingly, the reported runtimes should not be interpreted as absolute execution times applicable to other computing platforms. The runtime analysis of full-horizon sequential planning algorithms was performed using a simplified kinematic model intended to assess their practical applicability to pick-sequence planning for different numbers of consecutive objects, rather than to provide an accurate estimate of the physical robot cycle time. All computations were based on a single fixed set of 20 points. For each analysed case, the required number of objects was selected from this set according to their identifiers, for example, 5 out of 20 objects, as illustrated in Figure 2. This procedure ensured consistency across comparisons involving different planning-set sizes.

Implementation of planning algorithms

Branch-and-Bound algorithm

In the Branch-and-Bound algorithm, the solution space was represented as a tree. The root node corresponded to an empty sequence, while each subsequent level represented the addition of one of the remaining objects. The nodes were explored using a depth-first strategy. For each partial sequence, the number of feasible picks and its accumulated execution time were calculated. A branch was pruned if the next object could not be picked before leaving the robot workspace or if, even after including all remaining objects, the branch could not provide more picks than the best solution found so far. For the same number of picks, a branch was also pruned if its execution time could not be shorter than that of the current best solution. A timing safeguard was implemented for the online application. In Experiment 1, it was disabled to enable measurement of the complete B&B runtime for all analysed horizons. In Experiment 2, the adopted 13-object horizon allowed the search to finish before currently executed batch was completed; therefore, the safeguard was not activated.

Time-bounded Monte Carlo method

The Monte Carlo method generated random permutations of the objects included in the current decision horizon. Each permutation was evaluated using the same object-position prediction, pick-feasibility conditions and objective function as the B&B algorithm. Candidate solutions were generated and evaluated until the computation-time limit of 1 s was reached. The best solution was selected by first maximising the number of feasible picks and then, among solutions with the same number of picks, minimising the sequence execution time. Because the number of evaluated solutions depended on the horizon length and the time required to evaluate a single permutation, it was not fixed in advance.

Experimental design

Experiment 1: computational performance

The first experiment investigated the effect of decision horizon length on computation time and the quality of sequences generated using exhaustive search, Branch-and-Bound, and a time-bounded Monte Carlo method. The evaluated metrics included the number of successful picks, trajectory execution time and length, and algorithm runtime. For the deterministic algorithms, each computation-time measurement was repeated ten times, and the arithmetic mean was reported. The Monte Carlo method was evaluated using 100 independent runs for each decision horizon. Solution quality was characterised using the mean, standard deviation, median, interquartile range, minimum, maximum, and a two-sided 95% confidence interval for the mean execution time calculated using Student's t-distribution.

Experiment 2: sorting performance

The second experiment evaluated the performance of the FIFO, SPT, NNF, Branch-and-Bound, and Monte Carlo algorithms under varying process conditions. The simulation comprised 5.000 objects distributed equally among ten classes. The number of classes selected for sorting ranged from one to eight, the conveyor speed varied from 0.1 to 1.0 m/s, and the density of each class ranged from 6 to 25 objects per metre of conveyor length. All algorithms were evaluated using the same incoming-object stream within each scenario.

For the sequential planning methods, a batch-based finite decision horizon of 13 objects was adopted based on the results of the first experiment. The horizon was selected so that the

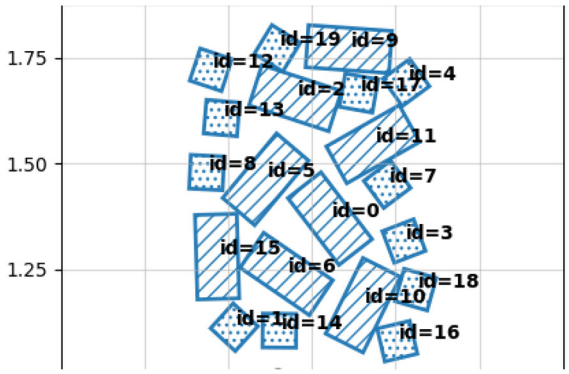


Figure 2. Spatial distribution of all objects considered in the experiments, showing objects of different sizes and their assigned IDs

sequence for the next batch could be determined before completion of the currently executed batch. Because planning was performed concurrently with robot motion, computation time was treated as a timing constraint and was not added to the batch execution time.

The primary process-performance measure was capture rate (CR), defined as the percentage of target objects that passed through the robot workspace and were successfully placed in the correct container. For each scenario, the algorithm achieving the highest CR was considered the best-performing method. PPM was defined as the number of successfully completed pick-and-place operations divided by the simulation time in minutes. It was treated only as a complementary throughput indicator and was not used for algorithm selection. When several algorithms achieved the same CR to the reported numerical precision, the result was classified as a tie.

RESULTS

The results are presented in two stages corresponding to the scope of the conducted experiments. The first stage evaluates the effect of decision horizon length on computation time and the quality of sequences generated using exhaustive search, Branch-and-Bound, and the Monte Carlo method. This analysis identifies the practical applicability range of each method and examines the trade-off between computational effort and solution quality. The second stage compares the performance of the analysed algorithms under varying conveyor speeds and different numbers of sorted object classes.

Computational performance for different decision-horizon lengths

Exhaustive search

In the first stage of the study, the effect of decision horizon length on the computation time of the exhaustive search algorithm was evaluated. The analysis covered horizons ranging from one to ten objects selected from the same input set. In accordance with the objective function, the algorithm first maximised the number of objects that could be successfully picked and then minimised the execution time of the resulting sequence. The results are presented in Table 1.

As shown in Table 1, increasing the decision horizon length resulted in a rapid growth in both the number of feasible sequences and the time required to evaluate them. For a horizon of five objects, the solution space comprised 120 sequences, and the mean computation time was 22.4 ms. For seven objects, the number of candidate sequences increased to 5,040, while the computation time reached 811.5 ms.

The available planning time was clearly exceeded when the horizon was extended to eight objects. In this case, the mean computation time was 3.60 s, whereas the execution time of the resulting trajectory was only 1.75 s. For horizons of nine and ten objects, the computation time increased further to 35.43 s and 473.87 s, respectively. This indicates that the algorithm would be unable to determine the next sequence before the current batch is completed, and some objects could leave the robot workspace before the computation was completed. The results presented in Table 1 also show that extending the decision horizon did not produce a proportional increase in the number of successful picks. For a horizon of five objects, all five objects could be picked. For horizons ranging from seven to nine objects, the number of successful picks remained constant at six, while for a horizon of ten objects it increased to seven. Thus, including additional objects in the planning horizon enlarged the solution space, but not all of them could be picked before leaving the robot workspace. Based on these results, a horizon of seven objects was identified as the largest for which the exhaustive search runtime remained shorter than the execution time of the resulting trajectory. Therefore, exhaustive search should not be considered impractical for all online applications. Under the investigated conditions, it

Table 1. Computational performance and solution quality of exhaustive search for different decision-horizon lengths

Exhaustive search algorithm							
Number of objects on the conveyor	1	3	5	7	8	9	10
Number of possible permutations	1	6	120	5 040	40 320	362 880	3 628 800
Algorithm computation time [ms]	0.10	1.10	22.40	811.50	3 601.00	35 432.70	473 871.40
Algorithm computation time [s]				0.81	3.60	35.43	473.87
Maximum robot trajectory length [m]	1.29	4.38	6.41	8.31	8.31	8.90	8.27
Minimum robot trajectory length [m]	1.29	4.04	5.77	7.09	6.42	6.42	8.04
Execution time of the minimum-length trajectory [s]	0.32	0.99	1.52	1.85	1.75	1.75	2.12
Set of objects available for picking	[0]	[0.1.2]	[0.1.2.3.4]	[0.1.2.3.4.5.6]	[0.1.2.3.4.5.6.7]	[0.1.2.3.4.5.6.7.8]	[0.1.2.3.4.5.6.7.8.9]
Number of objects picked	1	3	5	6	6	6	7
Object pick sequence (optimal solution)	[0]	[1, 0, 2]	[3, 1, 0, 4, 2]	[3, 6, 0, 5, 4, 2]	[3, 0, 7, 5, 4, 2]	[3, 0, 7, 5, 4, 2]	[3, 0, 7, 5, 4, 2, 9]

remained applicable for the decision horizons of up to seven objects and could also be suitable at lower conveyor speeds. However, as the horizon becomes shorter, the amount of information about future operations decreases, reducing the potential advantage of exhaustive planning over computationally simpler greedy methods. Its use is therefore justified when the improvement in sequence quality compensates for the additional computational and implementation cost. This conclusion is specific to the adopted motion model, the analysed object set, and the computational platform used in the experiments. Exhaustive search was therefore retained as an exact reference method for short decision horizons, whereas its online application became impractical for horizons exceeding seven objects under the investigated conditions.

Branch and bound

The next stage investigated whether the decision horizon could be extended using the Branch-and-Bound (B&B) algorithm. The analysis was performed using the same ordered input set as in the exhaustive search experiment, with larger horizons obtained by progressively adding subsequent objects to the planning set. The evaluated metrics included computation time, the length and execution time of the resulting trajectory, and the number of objects that could be successfully picked. The results are presented in Table 2.

As shown in Table 2, pruning the solution space enabled substantially longer decision horizons to be analysed than with exhaustive search. Although the theoretical number of possible sequences increased from 3.99×10^7 for 11 objects to 2.43×10^{18} for 20 objects the algorithm did not need to evaluate every permutation. The branches that could not improve upon the best solution found so far were discarded using the pruning mechanism described in Section 3. For decision horizons of 11 and 12 objects, the mean computation times were 0.19 s and 0.52 s, respectively. In both cases, they were substantially shorter than the execution time of the resulting trajectory, which was approximately 2.0 s. For a horizon of 13 objects, the computation time increased to 1.53 s but still remained below the trajectory execution time of 2.03 s.

Under the adopted conditions, the algorithm could therefore determine the next batch before completion of the current executed batch.

Further extension of the decision horizon caused the computation time to exceed the available planning time. For 14 objects, the algorithm runtime was 3.09 s, while for 15 and 16 objects it increased to 3.33 s and 4.23 s, respectively. The largest increase was observed for a horizon of 20 objects, for which the computation time reached 27.29 s. This was more than fourteen times longer than the execution time of the resulting trajectory, which was 1.94 s. Based on these results, a horizon of 13 objects was adopted as the largest for which B&B satisfied the real-time requirement imposed by the dynamics of the analysed process.

In all analysed cases, the algorithm generated a sequence containing seven successful picks. Increasing the number of objects in the horizon, therefore, did not increase the number of captured objects, but it enabled a more favourable combination of objects to be selected. The minimum trajectory length decreased from 7.89 m for 11 objects to 6.78 m for 20 objects, while the corresponding execution time decreased from 2.10 s to 1.94 s. This indicates that a longer horizon can improve the quality of the generated sequence, although the benefit is achieved at the cost of increased computation time.

Exemplary planning results for sets containing 10, 15, and 20 objects are shown in Figure 3. These visualisations illustrate the arrangement of objects in the analysed scene and the objects included in the resulting pick sequence.

As shown in Figure 3, increasing the number of objects included in the decision horizon expanded the set of candidate combinations from which the algorithm could construct a pick sequence. However, this did not increase the number of successful picks, which in the presented cases remained constrained by the object positions, their motion along the conveyor, and the time required for the robot to execute successive operations. For longer horizons, the algorithm could select objects that reduced the total travel distance and, consequently, the execution time of the complete sequence. Under the adopted objective function, objects located closer to the placement point were favoured because they required shorter trajectories. This effect is particularly evident for the 20-object set, for which the shortest trajectory among the variants reported in Table 2 was obtained. At the same time, some objects located farther from the placement point were excluded from the sequence even though they remained within the robot workspace. Picking them would either increase the total travel distance and sequence execution time or could not be completed before they left the reachable area.

The results indicate that B&B effectively reduced the number of solutions that had to be explicitly evaluated and enabled the use of a longer decision horizon than exhaustive search. In the adopted computational environment, a horizon of

Table 2. Computational performance and solution quality of the Branch and Bound algorithm for different decision-horizon lengths

Branch & Bound algorithm							
Number of objects on the conveyor	11	12	13	14	15	16	20
Number of possible permutations	3.99E+07	4.79E+08	6.23E+09	8.72E+10	1.31E+12	2.09E+13	2.43E+18
Algorithm computation time [ms]	193.00	517.90	1527.50	3085.80	3325.40	4228.00	27294.60
Algorithm computation time [s]	0.19	0.52	1.53	3.09	3.33	4.23	27.29
Maximum robot trajectory length [m]	8.28	8.29	8.30	8.30	8.30	8.30	8.44
Minimum robot trajectory length [m]	7.89	7.39	7.39	7.39	7.39	7.39	6.78
Execution time of the minimum-length trajectory [s]	2.10	2.03	2.03	2.03	2.03	2.03	1.94
Set of objects available for picking	[0-10]	[0-11]	[0-12]	[0-13]	[0-14]	[0-15]	[0-19]
Number of objects picked	7	7	7	7	7	7	7
Object pick sequence (optimal solution)	[10, 3, 0, 7, 4, 2, 9]	[3, 0, 7, 11, 4, 2, 9]	[3, 0, 7, 11, 4, 2, 9]	[3, 0, 7, 11, 4, 2, 9]	[3, 0, 7, 11, 4, 2, 9]	[3, 0, 7, 11, 4, 2, 9]	[18, 3, 7, 11, 4, 17, 9]

13 objects represented the practical limit. Further extending the horizon produced only a minor reduction in trajectory length while causing the computation time to exceed the time available for planning the next sequence. It should also be noted that minimising the total travel distance may lead to the systematic exclusion of more distant objects. This behaviour results from the adopted objective function, rather than from the intrinsic properties of the B&B algorithm, which only identifies the best solution according to the defined optimisation criterion.

Monte Carlo

Owing to its stochastic nature, the Monte Carlo method was evaluated using 100 independent runs for each decision horizon containing 11, 12, 13, 14, 15, 16, or 20 objects. Each run was limited to 1 s and used the same objective function and feasibility constraints as the B&B reference method. The statistical results are presented in Table 3.

A Monte Carlo run was classified as matching the complete B&B reference result when it achieved the same number of successful picks and the same minimum execution time as B&B.

As shown in Table 3, all 700 runs produced sequences containing seven successful picks, matching the B&B result for the primary optimisation criterion. However, the frequency of obtaining the complete B&B-optimal solution, including the minimum execution time, showed an overall decreasing tendency as the horizon increased. The optimum was obtained in 8%, 10%, 5%, 3%, and 1% of the runs for the horizons containing 11–15 objects, respectively, and was not observed for 16 or 20 objects. Thus, the method consistently achieved the primary objective, but its ability to reproduce the exact B&B optimum decreased for larger solution spaces.

Figure 4 shows that the trajectory length and execution time remained concentrated within relatively narrow ranges. The standard deviations of execution time were 0.01–0.02 s, the interquartile ranges were 0.01–0.03 s, and the widths of the 95% confidence intervals for the mean did not exceed 0.009 s. The mean Monte Carlo execution time differed from the corresponding B&B optimum by 0.01–0.08 s. Although individual results varied, as illustrated by execution-time ranges of 2.10–2.16 s for 11 objects and 1.98–2.07 s for 20 objects, this variability affected only the secondary criterion, whereas the number of successful picks remained equal to seven in every run.

Figure 5 shows the computational throughput achieved within the 1 s time limit. Because the stopping criterion was based on elapsed time rather than a fixed number of iterations, the number of evaluated sequences varied between runs. The nominal coverage of the search space decreased from approximately 0.00935% for 11 objects, equivalent to one in 10,700 candidate permutations, to approximately 1.16×10^{-13}% for 20 objects, equivalent to one in 865 trillion permutations. This decreasing coverage is consistent with the lower observed frequency of obtaining the B&B-optimal solution for longer horizons. Although Monte Carlo rarely reproduced the complete B&B-optimal solution, every run identified the maximum number of seven successful picks. Thus, the method always satisfied

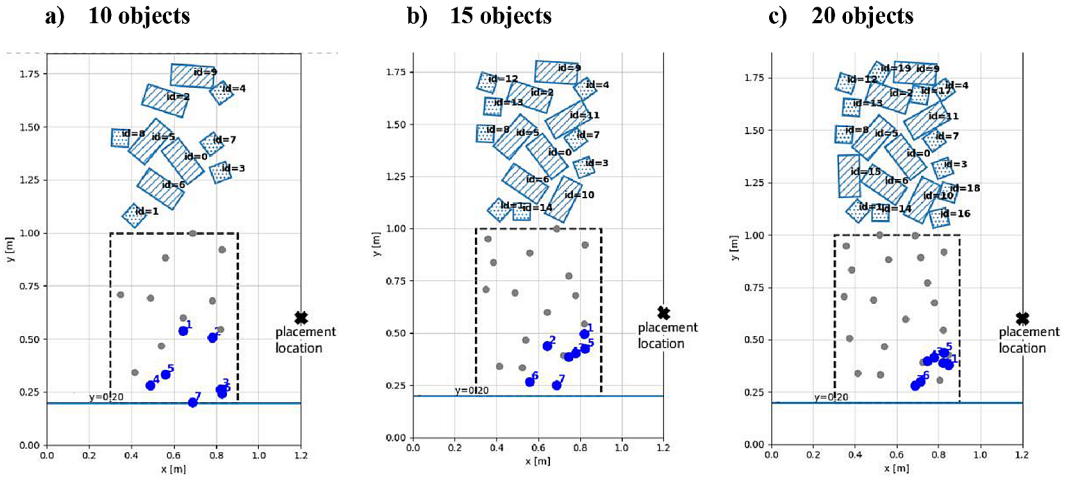


Figure 3. Examples of pick sequences determined using the Branch and Bound algorithm for decision horizons containing: (a) 10, (b) 15, and (c) 20 objects

Table 3. Statistical performance of the time-bounded Monte Carlo method over 100 independent runs for each decision horizon (1 s computation-time limit per run)

Monte Carlo algorithm							
Number of objects on the conveyor	11	12	13	14	15	16	20
Number of possible permutations	3.99E+07	4.79E+08	6.23E+09	8.72E+10	1.31E+12	2.09E+13	2.43E+18
Monte Carlo computation-time budget [s]	1.00						
Mean number of evaluated sequences per run	3732	5016	5075	4583	4683	3202	2814
Trajectory length — min–max [m]	7.90–8.27	7.39–7.56	7.39–7.71	7.39–7.84	7.39–7.96	7.43–8.15	7.06–7.62
Execution time of the best sequence found per run – mean ± SD [s]	2.13 ± 0.01	2.04 ± 0.01	2.05 ± 0.01	2.06 ± 0.02	2.06 ± 0.02	2.07 ± 0.02	2.02 ± 0.02
Execution time — median (IQR) [s]	2.13 (0.02)	2.04 (0.01)	2.05 (0.01)	2.06 (0.03)	2.06 (0.03)	2.07 (0.03)	2.02 (0.03)
Execution time — 95% CI of mean [s]	[2.124; 2.130]	[2.039; 2.041]	[2.045; 2.049]	[2.058; 2.064]	[2.059; 2.067]	[2.068; 2.077]	[2.016; 2.024]
Execution time — min–max [s]	2.10–2.16	2.03–2.06	2.03–2.08	2.03–2.10	2.03–2.11	2.04–2.14	1.98–2.07
B&B- Execution time of the minimum-length trajectory [s] (ref., Table 2)	2.10	2.03	2.03	2.03	2.03	2.03	1.94
Number of objects picked — min–max	7-7	7-7	7-7	7-7	7-7	7-7	7-7
Frequency of obtaining the B&B-optimal solution [%]	8.0	10.0	5.0	3.0	1.0	0.0	0.0

the primary objective, but did not always find the sequence with the shortest trajectory and execution time. This reduction in secondary solution quality may be acceptable because Monte Carlo always returned a solution within 1 s, whereas the resulting sequences required 1.98–2.16 s to execute. Whether the resulting deterioration in the secondary criterion is acceptable depends on its magnitude relative to the reduction in computation time. This comparison is presented in the following section.

Comparison of branch and bound and time-bounded Monte Carlo

The purpose of this comparison was to determine whether the predictable computation time of Monte Carlo compensated for the loss of exact optimality. Because both methods identified seven successful picks for every analysed horizon, they achieved the same value of the primary optimisation criterion. Therefore, the comparison focused on the planning-time constraint and the deterioration in the secondary criterion, i.e., sequence execution time.

Table 4 presents the mean relative increase with its 95% confidence interval and the maximum increase observed over 100 independent Monte Carlo runs. The planning-time constraint was considered satisfied when the sequence was determined before completion of the currently executed batch.

For horizons of up to 13 objects, B&B satisfied the planning-time constraint and was preferable because it guaranteed the optimal solution. For horizons of 14 objects or more, its computation time exceeded the sequence execution time, whereas Monte Carlo retained the fixed 1 s budget and produced seven successful picks in every run. Although Monte Carlo rarely reproduced the complete B&B optimum, the loss of solution quality remained limited. For 14 objects, the mean and maximum execution-time increases were 1.53% and 3.34%, respectively. Even for 20 objects, they remained at 4.12% and 6.47%, while the planning time decreased from 27.29 s to 1 s. These results support using B&B for horizons of up to 13 objects and time-bounded Monte Carlo for larger horizons.

These results indicate a practical switching point between the two methods. The B&B method is preferred for time horizons of up to 13 objects because it provides the optimal solution within the available planning time. However, for longer time horizons, waiting for the B&B method result would delay the next sequence and could cause

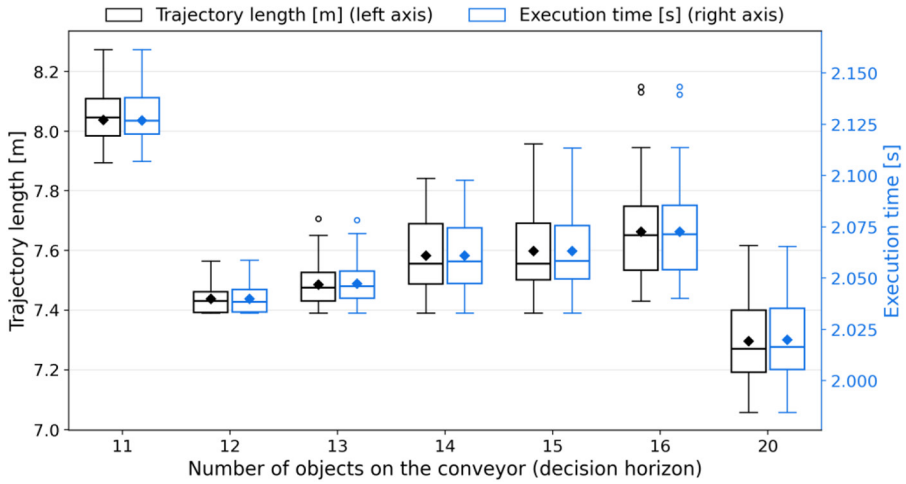


Figure 4. Distribution of trajectory length and execution time over 100 independent Monte Carlo runs for each decision horizon
Boxes represent the interquartile range, horizontal lines indicate the medians, diamonds indicate the means, whiskers extend to 1.5 times the interquartile range, and circles represent outliers

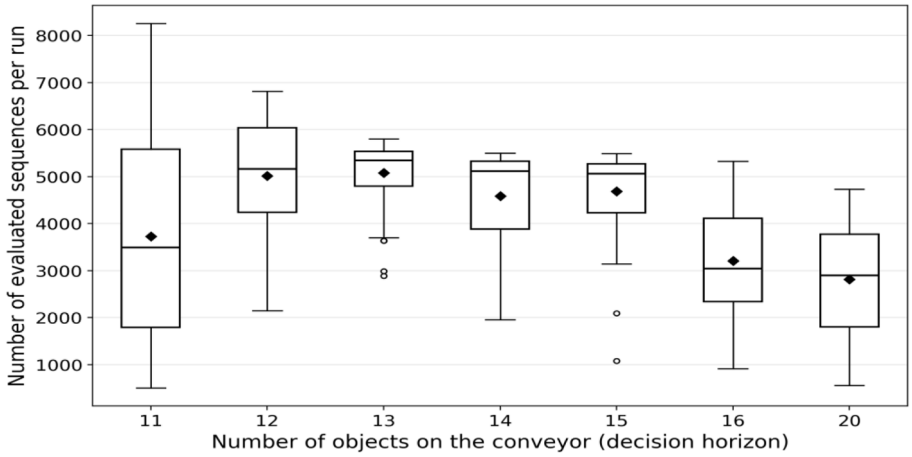


Figure 5. Distribution of the number of sequences evaluated within the 1 s Monte Carlo computation-time limit

the calculated plan to be inconsistent with the changing process state. Under such conditions, accepting a slightly longer sequence generated by the Monte Carlo method is preferable to obtaining an exact solution after the available decision time has elapsed. This trade-off is particularly evident for the 20-object case: the Monte Carlo method reduced the planning time from 27.29 s to 1 s while maintaining seven successful choices, at the cost of an average execution time increase of 4.12% and a maximum observed increase of 6.47%. Thus, the time-constrained Monte Carlo method allows for the online use of larger decision horizons with limited degradation of sequence quality. The identified switching threshold is specific to the adopted process model, computing platform, and Monte Carlo time budget of 1 s.

Scheduling performance under variable process conditions

After characterising the computational performance of the individual methods, the second stage of the study evaluated their effect on sorting effectiveness under varying process conditions. The primary process-performance measure was the capture rate (CR), defined as the percentage of target objects that were successfully collected. The analysis included first-in, first-out (FIFO), shortest processing time (SPT), nearest neighbour first (NNF) and Branch-and-Bound (B&B) methods under varying conveyor speeds and different numbers of sorted object classes. This made it possible to determine whether a single method maximised CR across the full range of process

Table 4. Comparison of the computation time and solution quality of B&B and time-bounded Monte Carlo

B&B vs Monte Carlo							
Number of objects on the conveyor	11	12	13	14	15	16	20
B&B algorithm computation time [s]	0.19	0.52	1.53	**3.09**	**3.33**	**4.23**	**27.29**
Monte Carlo computation-time budget [s]	1						
Execution time of the B&B-optimal sequence [s]	2.10	2.03	2.03	**2.03**	**2.03**	**2.03**	**1.94**
B&B satisfies the planning-time constraint	✓	✓	✓	**X**	**X**	**X**	**X**
Monte Carlo satisfies the planning-time constraint	✓	✓	✓	✓	✓	✓	✓
Mean relative increase in execution time vs B&B (95% CI) [%]	1.29 (1.14–1.43)	0.49 (0.44–0.54)	0.84 (0.74–0.94)	1.53 (1.38–1.67)	1.63 (1.43–1.82)	2.09 (1.87–2.32)	4.12 (3.92–4.33)
Maximum observed relative increase in execution time vs B&B [%]	2.93	1.42	2.38	3.34	4.11	5.57	6.47

conditions or whether algorithm selection should depend on the current process configuration. PPM was recorded only as a complementary throughput indicator and was not used as an algorithm-selection criterion.

Comparison of scheduling algorithms

The results demonstrated that the capture rate of each method depended on both conveyor speed and the number of object classes included in the sorting task. No algorithm achieved the highest CR across all analysed configurations. At low conveyor speeds and for a small number of classes, the highest CR was generally obtained using the FIFO and SPT dispatching rules. As conveyor speed and the number of sorted classes increased, multi-object sequence planning based on B&B became increasingly beneficial. The preferred algorithm for each process configuration was identified primarily according to CR. Because all algorithms were evaluated using the same incoming-object stream within each scenario, maximising CR was equivalent to maximising the number of successfully captured target objects. When two or more algorithms achieved the same CR to the reported numerical precision, the result was treated as a tie, and the computational complexity of the methods was considered. This trend is supported by the results in Table 5. B&B achieved the highest CR in 38 of the 56 configurations (67.9%) and was preferred for all numbers of classes at conveyor speeds of 0.8 and 1.0 m/s. Moreover, the transition to B&B occurred at 0.8 m/s for one class, but already at 0.2 m/s for five to eight classes.

FIFO has the lowest theoretical computational complexity because it directly selects the first available object without evaluating alternative candidates. Nevertheless, SPT is also a computationally lightweight dispatching rule and can be implemented without searching multi-object sequences. Unlike FIFO, it uses the estimated processing time of the currently available objects and therefore provides a process-aware decision while retaining a low computational cost. For this reason, SPT was retained in Table 5 as a representative low-complexity method in several CR-tie cases. FIFO remains an equally admissible choice in these configurations, and the indication of SPT should not be interpreted as evidence of a higher CR or a lower computational complexity than FIFO. This interpretation is supported by five CR-tie configurations in Table 5: for one class at 0.1 and 0.2 m/s and for two to four classes at 0.1 m/s, both SPT and FIFO achieved a CR of 100.00% (ΔCR=0.00 pp). Therefore, selecting SPT in these cases did not indicate better performance than FIFO.

The complexity criterion was primarily used to avoid selecting computationally demanding multi-object search methods when they did not provide an improvement in CR. Under more demanding conditions, involving a larger number of classes or higher conveyor speeds, B&B more frequently achieved the highest CR because it accounted for dependencies between successive operations. NNF were not selected as the preferred methods in any of the analysed configurations because they did not achieve a higher CR than the method indicated in Table 5 and did not provide a sufficient computational advantage in CR-tie cases.

Algorithm selection matrix

Because the algorithm achieving the highest CR varied across the 56 combinations of the number of sorted classes and conveyor speed, a

scenario-specific algorithm-selection matrix was constructed from the results of the second simulation stage. For each scenario, the algorithm achieving the highest CR was identified. When several algorithms achieved the same CR to the reported numerical precision, a low-complexity dispatching rule was retained as a representative selection. The other tied methods should therefore be regarded as equally effective with respect to CR.

Table 5 additionally reports the PPM achieved by the indicated algorithm as a complementary throughput measure. PPM was not used as an algorithm-selection criterion. The capture-rate margin over the second-ranked method is expressed in percentage points and calculated as ΔCR = CR_selected − CR_runner-up. A value of ΔCR = 0.00 pp indicates a CR tie rather than a measurable advantage of the algorithm listed in the corresponding table cell (Table 5).

Based on the representative algorithm assignments reported in Table 5, B&B was indicated most often (38 cases; 67.9%), followed by FIFO (11 cases; 19.6%) and SPT (7 cases; 12.5%); NNF was not indicated in any scenario. These frequencies include six CR-tie cases and should therefore not be interpreted

Table 5. CR-based identification of the preferred algorithm for each simulated scenario, with the corresponding PPM and absolute capture-rate margin over the second-ranked method

Number of sorted classes	0.1	0.2	0.3	0.4	0.6	0.8	1
1	**SPT (tie)** CR: 100.00% PPM: 9.91 ΔCR: 0.00 pp vs FIFO	**SPT (tie)** CR: 100.00% PPM: 19.78 ΔCR: 0.00 pp vs FIFO	**SPT** CR: 99.81% PPM: 29.62 ΔCR: 0.06 pp vs NNF	**FIFO** CR: 99.42% PPM: 39.33 ΔCR: 0.59 pp vs NNF	**FIFO** CR: 91.99% PPM: 54.46 ΔCR: 0.05 pp vs B&B	**B&B** CR: 79.55% PPM: 62.77 ΔCR: 0.64 pp vs SPT	**B&B** CR: 68.39% PPM: 68.24 ΔCR: 0.05 pp vs SPT
2	**SPT (tie)** CR: 100.00% PPM: 19.01 ΔCR: 0.00 pp vs FIFO	**SPT** CR: 99.80% PPM: 37.94 ΔCR: 0.02 pp vs NNF	**FIFO** CR: 98.78% PPM: 56.26 ΔCR: 1.60 pp vs B&B	**FIFO** CR: 91.07% PPM: 69.07 ΔCR: 1.41 pp vs B&B	**B&B** CR: 70.11% PPM: 79.76 ΔCR: 1.41 pp vs FIFO	**B&B** CR: 53.56% PPM: 82.66 ΔCR: 0.39 pp vs SPT	**B&B** CR: 43.32% PPM: 85.90 ΔCR: 0.54 pp vs FIFO
3	**SPT (tie)** CR: 100.00% PPM: 28.35 ΔCR: 0.00 pp vs FIFO	**FIFO** CR: 99.73% PPM: 56.50 ΔCR: 1.49 pp vs NNF	**FIFO** CR: 89.46% PPM: 75.82 ΔCR: 2.72 pp vs SPT	**B&B** CR: 72.29% PPM: 81.69 ΔCR: 0.59 pp vs FIFO	**B&B** CR: 50.64% PPM: 87.54 ΔCR: 0.82 pp vs SPT	**B&B** CR: 38.36% PPM: 88.86 ΔCR: 0.33 pp vs SPT	**B&B** CR: 30.48% PPM: 88.92 ΔCR: 0.12 pp vs SPT
4	**SPT (tie)** CR: 100.00% PPM: 38.52 ΔCR: 0.00 pp vs FIFO	**FIFO** CR: 95.75% PPM: 73.55 ΔCR: 2.20 pp vs SPT	**B&B** CR: 74.42% PPM: 85.13 ΔCR: 1.92 pp vs SPT	**B&B** CR: 58.06% PPM: 89.11 ΔCR: 1.23 pp vs SPT	**B&B** CR: 39.34% PPM: 90.50 ΔCR: 0.23 pp vs SPT	**B&B** CR: 29.63% PPM: 90.95 ΔCR: 0.17 pp vs SPT	**B&B** CR: 23.68% PPM: 90.90 ΔCR: 0.13 pp vs SPT
5	**FIFO (tie)** CR: 100.00% PPM: 48.04 ΔCR: 0.00 pp vs NNF	**B&B** CR: 86.41% PPM: 83.96 ΔCR: 0.62 pp vs FIFO	**B&B** CR: 63.01% PPM: 90.80 ΔCR: 2.22 pp vs SPT	**B&B** CR: 48.18% PPM: 92.52 ΔCR: 1.53 pp vs SPT	**B&B** CR: 32.31% PPM: 93.08 ΔCR: 0.75 pp vs SPT	**B&B** CR: 24.37% PPM: 93.60 ΔCR: 0.59 pp vs SPT	**B&B** CR: 19.30% PPM: 92.66 ΔCR: 0.27 pp vs SPT
6	**FIFO** CR: 100.00% PPM: 57.66 ΔCR: 0.13 pp vs SPT	**B&B** CR: 76.95% PPM: 88.61 ΔCR: 1.80 pp vs SPT	**B&B** CR: 53.51% PPM: 92.40 ΔCR: 1.52 pp vs SPT	**B&B** CR: 40.62% PPM: 93.46 ΔCR: 1.34 pp vs SPT	**B&B** CR: 27.19% PPM: 93.82 ΔCR: 0.74 pp vs SPT	**B&B** CR: 20.40% PPM: 93.77 ΔCR: 0.45 pp vs SPT	**B&B** CR: 16.19% PPM: 93.06 ΔCR: 0.26 pp vs SPT
7	**FIFO** CR: 100.00% PPM: 66.93 ΔCR: 1.00 pp vs SPT	**B&B** CR: 67.60% PPM: 90.38 ΔCR: 1.70 pp vs SPT	**B&B** CR: 45.87% PPM: 91.97 ΔCR: 0.89 pp vs SPT	**B&B** CR: 34.80% PPM: 92.99 ΔCR: 0.75 pp vs SPT	**B&B** CR: 23.29% PPM: 93.29 ΔCR: 0.40 pp vs SPT	**B&B** CR: 17.44% PPM: 93.18 ΔCR: 0.26 pp vs SPT	**B&B** CR: 14.01% PPM: 93.55 ΔCR: 0.23 pp vs SPT
8	**FIFO** CR: 98.40% PPM: 75.17 ΔCR: 1.87 pp vs B&B	**B&B** CR: 59.28% PPM: 90.51 ΔCR: 0.90 pp vs SPT	**B&B** CR: 40.22% PPM: 92.08 ΔCR: 0.67 pp vs SPT	**B&B** CR: 30.34% PPM: 92.61 ΔCR: 0.50 pp vs SPT	**B&B** CR: 20.38% PPM: 93.25 ΔCR: 0.30 pp vs SPT	**B&B** CR: 15.28% PPM: 93.22 ΔCR: 0.16 pp vs SPT	**B&B** CR: 12.18% PPM: 92.83 ΔCR: 0.10 pp vs SPT

Notes: CR – capture rate; PPM – successful picks per minute; ΔCR – absolute difference between the CR of the indicated algorithm and the next-highest CR, expressed in percentage points (pp); "Tie" indicates that multiple algorithms achieved the same CR to the reported numerical precision (ΔCR – 0.00 pp). The algorithm listed in a tie case represents an admissible low-complexity choice and should not be interpreted as uniquely superior; FIFO has the lowest theoretical computational complexity. SPT was retained in selected tie cases as an alternative computationally lightweight rule that additionally uses operation-time information when selecting the next object.

as the number of scenarios in which the listed method was uniquely superior. In the tied configurations, FIFO would provide the theoretically simplest implementation, whereas SPT represents an alternative low-complexity and process-aware dispatching rule. SPT was indicated mainly at low conveyor speeds and for a small number of classes, while FIFO was indicated under several intermediate conditions. As conveyor speed and the number of classes increased, the preference shifted towards B&B, which achieved the highest CR for all class counts at 0.8 and 1.0 m/s. The CR advantage of the indicated algorithm over the second-ranked method was frequently small. Across all 56 scenarios, ΔCR was below 1 percentage point in 40 cases (71.4%), including six CR-tie cases with ΔCR = 0.00 pp. The maximum difference was 2.72 pp and occurred for three sorted classes at 0.3 m/s, where FIFO achieved a CR of 89.46%, compared with 86.74% for SPT. The reported ΔCR values represent descriptive performance margins and should not be interpreted as evidence of statistical significance.

The results also demonstrate that selecting the algorithm with the highest CR among the evaluated alternatives does not necessarily guarantee an operationally acceptable sorting result. The lowest CR among the indicated configurations was 12.18%, obtained for eight classes at 1.0 m/s. In this scenario, B&B provided the highest CR among the evaluated methods, but most target objects were still not captured. Therefore, Table 5 identifies the method that collected the largest proportion of target objects relative to the evaluated alternatives, but it does not establish whether the resulting CR is sufficient for a particular industrial application.

The developed matrix may serve as the basis for an adaptive algorithm-selection mechanism driven by the current process configuration. Its applicability is limited to the parameter ranges, workstation model, robot parameters, and object-flow assumptions adopted in this study. It should therefore not be interpreted as a universal algorithm-selection rule without independent validation under different operating conditions.

CONCLUSIONS

The results demonstrate that increasing the decision horizon enlarged the set of candidate sequences, but did not produce a proportional improvement in solution quality. For exhaustive search, increasing the horizon from seven to eight objects raised the computation time from 0.81 to 3.60 s, while the number of successful picks remained equal to six and the sequence execution time decreased only from 1.85 to 1.75 s (Table 1). At a horizon of ten objects, the number of successful picks increased to seven, but the computation time reached 473.87 s, compared with a sequence execution time of 2.12 s. Thus, seven objects represented the largest computationally feasible horizon for exhaustive search under the investigated conditions.

B&B extended the feasible exact-planning horizon to 13 objects. Increasing the horizon from 11 to 13 objects raised its computation time from 0.19 to 1.53 s, whereas the sequence execution time decreased only from 2.10 to 2.03 s, and the number of successful picks remained equal to seven (Table 2). Extending the horizon from 13 to 20 objects further reduced the sequence execution time from 2.03 to 1.94 s, corresponding to an improvement of 4.4%, but increased the computation time from 1.53 to 27.29 s. Therefore, the additional information provided by a longer horizon improved only the secondary optimisation criterion and was insufficient to compensate for the loss of computational feasibility.

The time-bounded Monte Carlo method retained a fixed computation-time budget of 1 s for all analysed decision horizons. Across 100 independent runs per horizon, it identified seven successful picks in every run and therefore matched B&B with respect to the primary optimisation criterion. However, the frequency of reproducing the complete B&B-optimal solution generally decreased from 8% and 10% for horizons of 11 and 12 objects, respectively, to 0% for horizons of 16 and 20 objects (Table 3). Thus, increasing the horizon did not affect the primary criterion, but reduced the probability of optimising the secondary criterion within the available computation time. For the 20-object horizon, the mean relative increase in sequence execution time was 4.12% (95% CI: 3.92–4.33%), and the maximum observed increase was 6.47%, while the planning time decreased from 27.29 s for B&B to 1 s for Monte Carlo (Table 4). Under the investigated conditions, this trade-off enabled longer decision horizons to be used when the exact B&B solution could not be obtained within the available planning time.

The second experimental stage showed that algorithm selection depended jointly on conveyor speed and the number of sorted classes. B&B achieved the highest CR in 38 of the 56 analysed scenarios and became the highest-CR method at progressively lower conveyor speeds as the number of classes increased (Table 5). It was selected from 0.8 m/s for one class, but already from 0.2 m/s for five to eight classes. The effect of the number of classes is also evident at 0.4 m/s, where the highest CR decreased from 99.42% for one class to 30.34% for eight classes. Under lower process loads, FIFO and SPT achieved the same or higher CR with simpler decision logic. These results indicate that multi-object planning becomes more beneficial as the target-object flow increases and the time available for handling individual objects decreases.

The results also confirm that CR, rather than PPM, should be the primary measure of sorting effectiveness. For eight classes, increasing the conveyor speed from 0.2 to 1.0 m/s changed PPM only from 90.51 to 92.83, whereas CR decreased from 59.28% to 12.18% (Table 5). Therefore, the robot maintained a similar number of picks per minute while capturing a substantially smaller proportion of the target objects. Moreover, the CR advantage over the second-ranked algorithm was less than percentage point in 40 of the 56 scenarios. Consequently, the selection matrix identifies the best-performing method among the evaluated alternatives, but does not guarantee an operationally acceptable sorting result. The identified thresholds are specific to the adopted simulation model, computational platform, object stream, and process parameters. They should therefore be interpreted as experimentally determined switching points, rather than universal limits.

REFERENCES

1. Han, S.D.; Feng, S.W.; Yu, J. Toward fast and optimal robotic pick-and-place on a moving conveyor. IEEE Robot. Autom. Lett. 2020, 5, 446–453. https://doi.org/10.1109/LRA.2019.2961605
2. Choudhury, S.; Gupta, J.K.; Kochenderfer, M.J.; Sadigh, D.; Bohg, J. Dynamic multi-robot task allocation under uncertainty and temporal constraints. Auton. Robot. 2022, 46, 231–247. https://doi.org/10.1007/s10514-021-10022-9
3. Teck, S.; Engelen, B.; Peeters, J.; Kellens, K.; Vansteenwegen, P. The development of an efficient scheduling heuristic for multi-robot pick-and-place operations. Int. J. Prod. Res. 2025, 63, 8923–8942. https://doi.org/10.1080/00207543.2025.2517835
4. Krot, K.; Poskart, B.; Mazur, J. Algorithm selection for dynamic robotic pick-and-place tasks on conveyor belts under time constraints. J. Mach. Eng. 2026, 26, 1–13. https://doi.org/10.36897/jme/220925
5. Schettini, T.; Malucelli, F.; Ramalhinho, H. Scheduling a pick and place packaging line. J. Sched. 2023, 26, 113–136. https://doi.org/10.1007/s10951-022-00755-5
6. Nielsen, K.G.; Sung, I.; El Yafrani, M.; Kılıç, D.K.; Nielsen, P. A scheduling solution for robotic arm-based batching systems with multiple conveyor belts. Algorithms 2023, 16, 172. https://doi.org/10.3390/a16030172
7. Sheng, J.; He, D.; Gu, Q.; Gong, D. Dual-arm robot fast pick-and-place on a moving conveyor. J. Phys. Conf. Ser. 2022, 2216, 012025. https://doi.org/10.1088/1742-6596/2216/1/012025.
8. Zhou, M.; Jiang, R. Optimal strategy for pick-and-place system with two robots. J. Phys. Conf. Ser. 2022, 2216, 012022. https://doi.org/10.1088/1742-6596/2216/1/012022.
9. Borrell, J.; Perez-Vidal, C.; Segura, J.V. Optimization of the pick-and-place sequence of a bimanual collaborative robot in an industrial production line. Int. J. Adv. Manuf. Technol. 2024, 130, 4221–4234. https://doi.org/10.1007/s00170-023-12922-9
10. Engelen, B.; Teck, S.; Peeters, J.R. System layout and grasp efficiency optimization for a multirobot waste sorting system. Robotics 2025, 14, 22. https://doi.org/10.3390/robotics14030022
11. Li, J.; Zanardi, A.; Zardini, G. Multi-agent path finding via finite-horizon hierarchical factorization. arXiv 2025, arXiv:2505.07779. https://doi.org/10.48550/arXiv.2505.07779
12. Li, S.; Ouyang, W.; Ma, Y.; Wu, C. Learning-guided rolling horizon optimization for long-horizon flexible job-shop scheduling. In Proceedings of the International Conference on Learning Representations (ICLR 2025), Singapore, 24–28 April 2025. https://doi.org/10.48550/arXiv.2502.15791
13. Touzani, H.; Seguy, N.; Hadj-Abdelkader, H.; Suárez, R.; Rosell, J.; Palomo-Avellaneda, L.; Bouchafa, S. Efficient industrial solution for robotic task sequencing problem with mutual collision avoidance and cycle time optimization. IEEE Robotics and Automation Letters 2022, 7(2), 2597–2604. https://doi.org/10.1109/LRA.2022.3142919
14. Hartmann, V.N.; Orthey, A.; Driess, D.; Oguz, O.S.; Toussaint, M. Long-horizon multi-robot rearrangement planning for construction assembly. IEEE Transactions on Robotics 2023, 39(1), 239–252. https://doi.org/10.1109/TRO.2022.3198020