A Study on the Effects of Model Simplifications for Generating Human-Like Robot Movements in Refill Tasks

Daniel Rodrigues[1(✉)], Eliana Costa e Silva[1,2], Gianpaolo Gulletta[1], Pedro Ribeiro[1], Inês Costa[1], Wolfram Erlhagen[3], Cátia Ferreira[4], Luís Louro[1], and Estela Bicho[1]

[1] Center Algoritmi/Department of Industrial Electronics, University of Minho, Braga, Portugal
{pg47125,a84489,pg50435}@alunos.uminho.pt, {d6468,estela.bicho}@dei.uminho.pt, luislouro@algoritmi.uminho.pt
[2] CIICESI, ESTG, Polytechnic of Porto, Porto, Portugal
eos@estg.ipp.pt
[3] Center of Mathematics/Department of Mathematics, University of Minho, Braga, Portugal
wolfram.erlhagen@math.uminho.pt
[4] MC SONAE, Matosinhos, Portugal
catsferreira@mc.pt

Abstract. In the present work, the optimization problems arising during the generation of human-like movements for an anthropomorphic robot performing shelf replenishment tasks are evaluated in detail. Specifically, a set of movements that involve grasping five objects from a support table and transporting each object to a shelf is analyzed. These movements are formulated as nonlinear optimization problems, with various constraints related to the posture of the arm and the objects in the workspace. The impact of different model simplifications on the computational time necessary for the Interior Point OPTimizer (IPOPT) solver to obtain an optimal solution is explored. The optimization problems are modeled using the A Mathematical Programming Language (AMPL) modeling language and different levels of presolve are used. Our results demonstrate that increasing the presolve level enhances computational efficiency. However, excessively high presolve may negatively affect performance for some problems when compared to lower levels. This research highlights the importance of selecting an appropriate presolve level to optimize both problem formulation and solution time.

Keywords: Nonlinear optimization · Motion planning · Statistical analysis · Retail · Robotic manipulation

1 Introduction

The Human-like Upper-limb motion planner (HUMP) is a motion planning algorithm developed to generate collision-free trajectories with human-like characteristics for anthropomorphic robotic arms [1,2]. In this, the underlying nonlinear

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2026
O. Gervasi et al. (Eds.): ICCSA 2025 Workshops, LNCS 15888, pp. 225–240, 2026.
https://doi.org/10.1007/978-3-031-97596-7_15

constrained optimization problems were modeled using AMPL [3] and solved using the IPOPT solver [4]. The performance of HUMP, in terms of computation time and quality of the solution found, is affected by IPOPT's ability to solve the underlying optimization problems. IPOPT is a widely used interior point optimization solver for large-scale nonlinear programming problems [5]. Its performance depends on several aspects that will be summarized next. The choice of the linear solver affects IPOPT's performance, with parallel solvers potentially improving scalability for certain problem classes [5]. The mathematical formulation affects IPOPT's performance, as demonstrated by a new spherical coordinates model for progressive lens design, which exhibited better convexity properties than previous Cartesian models [6]. To improve robustness for problems without constraint qualifications, an l1-exact penalty-barrier phase was implemented in IPOPT, showing favorable performance for degenerate nonlinear programs [7]. For problems with nonconvex constraints, a new interior point method was developed that finds approximate Fritz John points with improved iteration bounds, representing a significant advancement in complexity analysis for such methods [8]. Scaling the linear system before solving can also improve performance in some cases [9]. The type of optimization problem, such as compliance minimization or stress-constrained mass minimization, affects solver performance [10]. Problem size, mesh structure, and design domain complexity can also influence IPOPT's effectiveness [10]. Comparative studies using benchmark problems have shown that different solvers may perform better for specific types of problem, highlighting the importance of selecting appropriate algorithms based on the problem characteristics [5,10].

Using the presolve option in AMPL can enhance IPOPT's performance by simplifying the optimization problem, leading to faster and more stable solutions. It is generally recommended to enable presolve and test its impact on specific problems. In this work, an empirical test is conducted on a set of 46 nonlinear constrained optimization problems related to the generation of reaching and grasping movements for five objects, followed by their placement on shelves. The objective is to perceive how the problem formulation affects the problem resolution in terms of computing time.

Section 2 introduces the language used to formulate the problems and provides an overview of the solver employed. Section 3 outlines the task under analysis and details the experimental procedure. Section 4 presents the results, with Sect. 4.1 examining the dimensionality of the optimization problems, particularly focusing on the number of inequality constraints. Section 4.2 discusses the influence of different presolve levels on the solving time of the optimization problems. Section 5 presents the conclusions drawn from the work undertaken.

2 Methods

The solver used is IPOPT, which is a powerful open-source solver for large-scale nonlinear optimization, developed by Andreas Wächter and Lorenz T. Biegler.

It uses a primal-dual interior point method, making it particularly effective for handling large, sparse problems with high precision and reliability. IPOPT supports various linear solvers, such as MUMPS, HSL, and Pardiso, and integrates seamlessly with AMPL, making it a robust and efficient choice for nonlinear optimization.

AMPL is a modeling language designed to solve large-scale optimization problems in mathematical computing. It supports linear, nonlinear, and integer optimization, using syntax that closely mirrors mathematical notation to clearly express constraints, variables, and objectives. AMPL integrates seamlessly with a wide range of solvers, including CPLEX, MINOS, IPOPT, SNOPT, and KNITRO, by transmitting problems through .nl files generated from its model and data files. Its primary strengths lie in its user-friendly design, exceptional flexibility, and smooth integration with solvers. Additionally, AMPL supports the use of trigonometric functions in problem descriptions, making it particularly effective for modeling the kinematics of robotic arms [11].

Presolve is a preprocessing step intended to simplify optimization problems before solving. It removes redundant constraints, fixes variables by deducing their values from constraints, and simplifies equations to eliminate unnecessary variables and constraints. These reductions decrease problem size, lower computational effort, and enable faster, more efficient problem resolution. For a detailed explanation of the presolve algorithm used in AMPL, it is recommended to see the work [14].

In addition to problem simplification, presolve enhances numerical stability and overall solver performance. By identifying and removing poorly scaled or redundant constraints, it improves convergence properties and reduces memory usage during optimization [12]. For instance, in [13], presolve reduced model complexity by an average of 52.8% and identified infeasibility in 20% of cases. These reductions help solvers like IPOPT achieve faster and more efficient convergence. The performance benefits of presolve are particularly significant for larger, more structured problems, which tend to see the greatest reductions. However, the extent of presolve's impact also depends on the quality of the initial problem formulation in AMPL. In order to ascertain how this preprocessing step affects the solving of nonlinear optimization problems by IPOPT, different levels of presolve are used on the same problem. The differences in the problem formulation and the solving process are then analyzed.

3 Experimental Setup

The scenario under examination comprises the anthropomorphic robot RAMBO (Robotic Anthropomorphic Manipulator for Bimanual Operations) [15], which is tasked with restocking a shelving unit with various products. In particular, the process of restocking five products on a rotative shelf using the right arm will be analyzed in detail (see Fig. 1). The task begins with the robot in its home position. The robotic arm initiates the process by grasping the first object from the table and placing it onto the bottom shelf. It then returns to the table to

retrieve the second object, which is placed on the second shelf. Next, it picks up the third object and places it on the top shelf, followed by the fourth object, which is positioned on the third shelf. Finally, the arm collects the fifth object from the table and places it on the third shelf as well. Once all objects have been restocked, the arm returns to its home position.



Fig. 1. RAMBO robotic platform restocking a shelving unit with products placed on a support table to the right of the robot.

For planning the robotic arm's movements, the HUMP [2] motion planner was used, which offers a global approach for generating obstacle-free, human-like trajectories of robotic arms. The refilling process is divided into two main stages: (i) picking the product from a support table, and (2) placing it on a shelf. The pick and place movements can be subdivided into three objected-directed primitives, designated as transport, approach, and retreat, whereas move movements, which consist of moving the arm without any object manipulation, only have the transport phase, as depicted in Fig. 2. To generate the pick-and-place movements, it is necessary to determine two key sets of postures: the target posture and the bounce posture. The determination of these postures involves addressing nonlinear optimization problems with different characteristics (for details on the nonlinear optimization problems see [2]). The target posture problem computes a posture that, given a desired position and orientation for the robot's hand, guarantees there are no collisions with obstacles in the workspace, provided the movement is feasible. Additionally, it ensures that the physical joint limits are respected while minimizing joint displacement from the initial to the target posture. Once the target posture is determined, a trajectory – referred to as the direct movement – is generated, guiding the robot from the starting posture to the calculated target posture. The bounce posture, on the other hand, is determined by checking for potential collisions at multiple discrete points along the trajectory, while again ensuring that joint limits are maintained and joint displacement is minimized. This posture facilitates the creation of a back-and-forth movement. To avoid collisions with obstacles, this back-and-forth movement is superimposed on the direct movement, producing a composite movement that ensures there are no collisions with obstacles over the duration of the movement.

In regard to the pick movement to determine the trajectories for each stage it is necessary to calculate four postures: (i) pre-approach posture, which involves moving the arm to a position near the selected product; (ii) bounce posture, which is responsible for ensuring that the trajectory generated does not have collisions with any obstacles present in the workspace of the robot; (iii) approach posture that corresponds to the posture of grasping the product; (iv) retreat posture that lifts the product from the table.

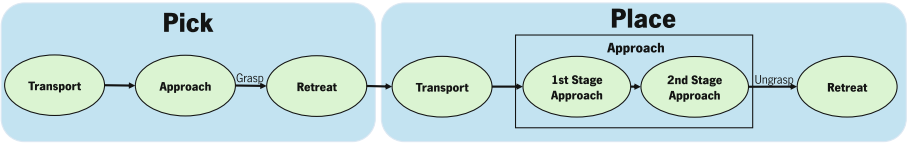


Fig. 2. Movement segmentation (for details it is recommended to see the work by [2]).

Conversely, the place movement comprises the determination of five distinct postures: (i) pre-approach posture, is characterized by the positioning of the product in close proximity to its intended location on the shelf; (ii) bounce posture, ensures that the robotic arm and the product being manipulated do not collide with any objects present in the scenario; (iii) first approach posture, which entails positioning the object just above its intended placement position; (iv) second approach posture, involves placing the product in its designated location; (v) retreat posture is the posture that takes the hand away from the product. Finally, there is a move movement that returns the robotic arm to its home posture. This movement only requires the calculation of the bounce posture, as the target posture is predefined as the arm's home position. The move movement may need a target posture selection problem if the intention is for the robotic arm to reach a pose other than its predefined home posture. The postures calculated in each stage of each movement are summarized in Table 1.

Table 1. Optimization problems solved on each movement.

Movement	Phase	Target Posture Selection	Bounce Posture Selection
Pick	Transport	✓	✓
	Approach	✓	×
	Retreat	✓	×
Place	Transport	✓	✓
	Approach	✓	×
	Retreat	✓	×
Move	Transport	×	✓

In all planned movements, the robot's body, the table, and the products to be replenished on the right side are modeled as obstacles using ellipsoids. However, during the pick movement, due to the geometry of the shelving unit and to simplify the optimization problems, ellipsoids are not used. Instead, a constraint is imposed that prevents any point of the robotic arm from exceeding the distance between the torso and the shelving unit, effectively avoiding collisions. For the place movement, this constraint is only considered for the transport phase. In the approach and retreat phases, individual shelves are modeled as ellipsoids, allowing the robot's hand to enter the shelf for a precise product placement.

The initial guess for the target posture optimization problems is derived from a small database containing previously recorded arm postures associated with specific hand positions and orientations. If the target posture falls within a defined range – specifically, a positional tolerance of 50 mm and an orientation tolerance of 5°C – of a stored posture, the corresponding posture is used as the initial guess. Otherwise, the arm's default initial configuration is used as the starting point. For the bounce posture optimization problems, the initial configuration of the arm is used as initial guess.

The refilling task consists of restocking five different products on the shelf. To accomplish this, the system must plan five pick movements, five place movements, and one move movement. As previously explained, each pick movement requires determining four distinct postures, while each place movement involves calculating five postures. The move movement, on the other hand, requires the computation of a single posture. Table 2 provides a breakdown of the 46 optimization problems organized by movement type.

Table 2. Number of optimization problems for the different type of movements.

Action	Pick	Place	Move
1	Vinegar	4th Shelf	-
2	Coffee	2nd Shelf	-
3	Conditioner	1st shelf	-
4	Can of beans	3rd shelf	-
5	Can of tomatoes	3rd shelf	-
6	-	-	Home
N. Target prob.	15	20	0
N. Bounce prob.	5	5	1

For evaluating the influence of the formulation on the CPU time on these problems, three presolve levels are considered, namely:

- 0, presolve is disabled, however AMPL may perform substitutions;
- 1, part one of the presolve is enabled, in this AMPL assigns the bounds to each variable and then tries to deduce tighter bounds. At the same time, it tries to detect variables that can be fixed and constraints that can be dropped;

– 10, both parts of the presolve are enabled, where several iterations of the problem are performed in order to deduce tighter bounds for the variables bounds and linear constraints for the current bounds.

4 Results

The evaluation of nonlinear optimization problems is based on three metrics: the number of variables, the number of inequality constraints, and the total CPU time required by the solver to reach a solution.

4.1 Dimension of the Optimization Problems

Table 3 summarizes the statistics on the number of inequality constraints for each optimization problem solved, during the task of restocking five products, considering three presolve levels: 0, 1, and 10. The data is divided by movement type and problem type(target or bounce). For the target posture problems, the number of decision variables is seven, corresponding to the seven joints of the robotic arm. In contrast, for bounce posture problems, the number of variables increases to eight, as the hand aperture is also considered.

Table 3. Statistics of the number of inequality constraints.

Movement	Type	Presolve	Variables	Min	Median	Max	Count
Pick	Target	0	7	17	37	73	15
		1					
		10					
Pick	Bounce	0	8	656	1301	1616	5
		1		599	1196	1463	
		10		455	1028	1319	
Place	Target	0	7	5	68	135	20
		1					
		10					
Place	Bounce	0	8	501	1197	1751	5
		1		454	1098	1578	
		10		321	944	1452	
Move	Bounce	0	8	257			1
		1		221			
		10		149			

In optimization problems involving the selection of target postures for both pick and place movements, the number of variables and constraints remains small. Additionally, the presolve level has no impact on the number of constraints. Specifically, for pick movements, the number of constraints ranges from 17 to 73, with half of the problems involving 37 or fewer constraints. However, when bounce postures are selected, the number of constraints increases significantly. At presolve level 0, the number of constraints range from 656 to

1616, with half of the problems having 1301 or fewer constraints. Increasing the presolve level to 1 and 10 reduces the number of constraints. At presolve level 10, the number of inequality constraints ranges from 455 to 1319, with half of the problems having 1028 or less constraints. This demonstrates that higher presolve levels consistently reduce inequality constraints in bounce posture problems. For place movements, the target posture problems have the same number of constraints across all presolve levels, with a range from 5 to 135, indicating no effect from presolve. However, for bounce posture problems, the number of constraints is much higher. At presolve level 0, the constraints range from 501 to 1751, with half of the problems having 1197 or fewer constraints. As the presolve level increases to 1 and 10, the number of constraints decreases, with presolve level 10 yielding values ranging from 321 to 1452, and half of the problems having 944 constraints or less. For the move movement, the number of constraints consistently decreases as the presolve setting increases. At presolve level 0, the constraints are fixed at 257, while presolve level 1 reduces this to 221, and presolve level 10 further reduces it to 149.

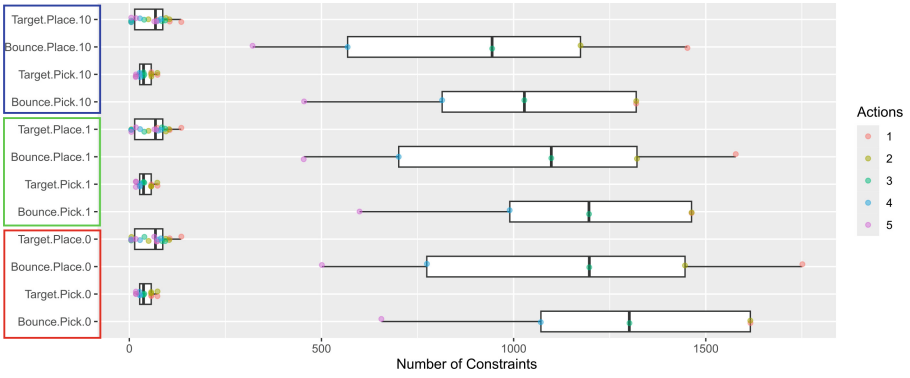


Fig. 3. Distribution of the number of inequality constraints.

Overall, the presolve process effectively reduces inequality constraints for the bounce type problems, with the most significant reductions seen at higher presolve values. In contrast, the target posture problems are less affected by presolve, with place movements showing no changes at all. The move movement shows uniform reductions in constraints with higher presolve levels but no variability. The number of inequality constraints for each solved problem is depicted in Fig. 3. Due to the significant differences in the problem formulation between target posture problems and bounce posture problems, they were categorized into these two distinct groups.

Changing the presolve level in target posture problems had no effect, as the number of inequality constraints remained unchanged. This is likely because the problems associated with the target postures are simpler in nature.

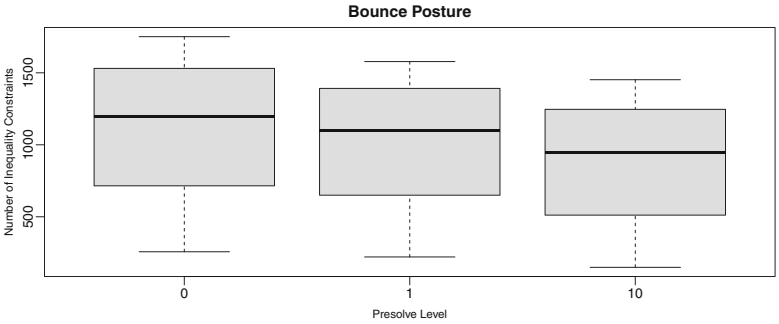


Fig. 4. Distribution of the number of constraints for different presolve levels of bounce posture problems.

On the other hand, when calculating the bounce posture, increasing the presolve level results in a reduction of the dimension of the associated optimization problems, more precisely, in the number of inequality constraints. This might be explained by the fact that the selection of the bounce posture is such that a collision-free trajectory of the arm and hand is obtained. There are constraints representing each object in the robot's workspace and for each point of the robot at every step of the trajectory. As a result, the problem is more complex compared to the target posture problems. Figure 4 illustrates the relationship between presolve levels and the number of inequality constraints in the optimization problems solved to determine the bounce posture.

To determine whether the observed decrease in the number of inequality constraints with an increase in the presolve level is statistically significant, a Kruskal-Wallis rank sum test was conducted to infer if there are significant differences in the median number of constraints for each presolve level. Since p-value=0.4211 $> 5\%$ ($\chi^2 \simeq 1.73$), for a significance level of 5%, there are no significant differences in the median number of inequality constraints in the bounce posture problems. Therefore, there is no evidence to support a significant effect of the presolve on the number of inequality constraints.

4.2 Computational Time

For each presolve level, each of the nonlinear optimization problems was executed 30 times to ensure the reliability of the results. Data regarding computation time and other pertinent metrics were gathered in a systematic manner across these trials with the objective of evaluating the influence of presolve on the overall efficiency of solving the optimization problems.

Nevertheless, it is important to note that, in some cases, the correlation between presolve and computation time may not be straightforward. This suggests that the complexity and structure of the problem may impact the effectiveness of presolve in optimizing performance. The data raises the question of whether increasing the presolve level could prove detrimental to the resolution

of the problem. When plotting the computation time for different presolve levels of target posture problems (Fig. 5), there is minimal variation between levels. However, in bounce posture problems, the variability is higher, and the median value for presolve equal to 10 is worse than that for lower presolve settings.

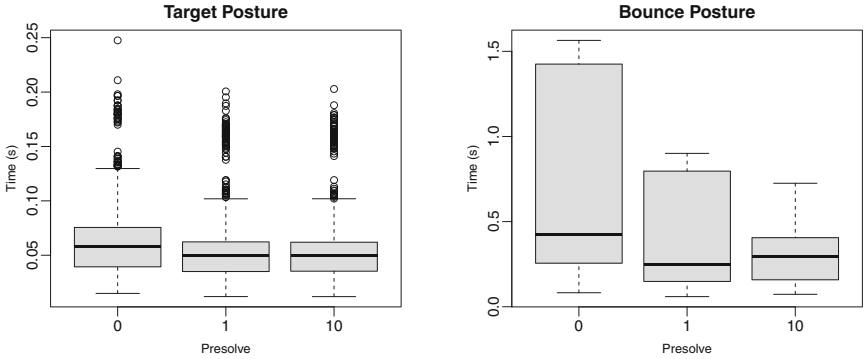


Fig. 5. Distribution of the time computation for optimization problems for different presolve levels.

Since the optimization problems solved for determining the bounce posture depend also on the spatial distribution of the objects on the robot's workspace and on the specific objects of the target-oriented movements, a closer examination was conducted on its computation. Figure 6 illustrates the computation times of bounce posture problems for different types of movements under various levels of presolve. The horizontal axis presents the computation time in seconds, while the vertical axis displays the types of movement that are planned. The initial five actions represent the reaching and placement of a product on the shelves, while action 6 represents a move movement, which corresponds to the arm returning to its home posture. As can be observed in Fig. 5, higher presolve levels generally result in shorter computation times.

The analysis is complemented with the information in Table 4 and Table 5, which present computation times for selecting target and bounce postures under different presolve settings (0, 1, 10) for pick, place, and move type movements. For the computation of the optimal bounce posture of each pick movement, presolve has a significant impact, half of the problems present computational times less than or equal to 1.046 s, while for presolve 1 this time reduces by 41% to 0.619 s, and presolve 10 reduces the time by 34% compared to presolve 1. For bounce posture problems, in place movements, half of the problems present a computational time of 0.422 s or less. When using presolve 1 the computation time reduces by 42%, however, when using a presolve 10, the computation time for half of the problems was not greater than 0.408 s, which represents an increase of 65% in relation to the previous level and a decrease of only 2% compared to presolve 0. For the move movement, computation times are generally lower, as

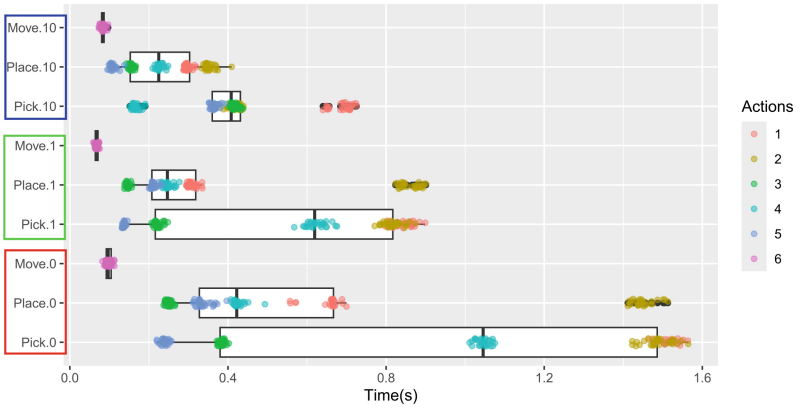


Fig. 6. Distribution of the computational time, in seconds, for selecting the bounce posture for restocking five products (Actions 1,2,3,4, and 5) in a shelving unit and return to home position (Action 6).

this movement only moves the arm from the third shelf to its home position, with presolve 1 offering the greatest reduction of 31%, while presolve 10 presents only a reduction of 14%. For target posture problems, computation times remain stable across different presolve levels, with a maximum decrease of about 2%. Increasing the presolve level reduces computation times for bounce posture problems, while having little impact on the computation time of target posture problems. This suggests the existence of differences in the computational time required by the solver to compute the bounce posture at different presolve levels for pick and place movements. To infer if these differences are significant, two Kruskal-Wallis tests were performed (see Table 6), namely to test whether there are differences in the median computational times using the three presolve levels for (i) pick

Table 4. Statistics of the computation time for different presolve levels of the bounce posture selection problems.

Movement	Type	Presolve	Min	Median	Max	N
Pick	Bounce	0	0.220	1.046	1.564	150
Pick		1	0.132	0.619	0.899	150
Pick		10	0.152	0.408	0.725	150
Place		0	0.237	0.422	1.513	150
Place		1	0.138	0.247	0.901	150
Place		10	0.152	0.408	0.725	150
Move		0	0.082	0.097	0.112	30
Move		1	0.059	0.067	0.075	30
Move		10	0.073	0.083	0.097	30

Table 5. Statistics of the computation time for different presolve levels of the target posture selection problems.

Movement	Type	Presolve	Min	Median	Max	N
Pick	Target	0	0.027	0.065	0.145	450
Pick		1	0.017	0.059	0.201	450
Pick		10	0.024	0.057	0.179	450
Place		0	0.015	0.049	0.248	600
Place		1	0.012	0.043	0.183	600
Place		10	0.012	0.045	0.203	600

and (ii) place movements. For (i) p-value=$4.2e-11 < 5\%$ ($\chi^2 \simeq 47.78$), while for (ii) p-value=$2.2e-16 < 5\%$ ($\chi^2 \simeq 142.43$). Therefore, for a significance level of 5%, the null hypothesis that there are no significant differences is rejected. These results imply that the choice of presolve level has a significant impact on computation performance for both groups, but the effect appears even more pronounced in the computation of the bounce posture, as indicated by the higher χ^2 statistic.

Table 6. Kruskal-Wallis test.

Movement	Type	χ^2	p-value
Target	Pick	28.81	5.547e-07
	Place	70.348	5.298e-16
Bounce	Pick	47.783	4.209e-11
	Place	142.43	2.2e-16

Subsequently, Pairwise Wilcox tests were conducted to assess whether the differences between the various presolve levels were statistically significant. From the p-values shown in Table 7, it can be inferred that to determine the bounce posture for pick movements, there are differences in the computation times between level 0 and levels 1 (p-value=6.7e-14 < 5%) and 10 (p-value=8.9e-11), and also level 1 and level 10 (p-value=0.0024 < 5%). The same applies to bounce posture problems in place movements, where it is possible to infer that there are differences in computation times between level 0 and levels 1 (p-value=2e-16 < 5%) and 10 (p-value=2e-16 < 5%), and also level 1 and level 10 (p-value=0.0037 < 5%). For target posture problems, in pick movements there are differences in the computation time between level 0 and levels 1 (p-value=4.7e-05 < 5%) and 10 (p-value=2e-06 < 5%), but there are no significant differences between presolve level 1 and 10 (p-value=0.46 > 5%). In place movements there are differences in the computation time between level 0 and levels 1 (p-value=1e-09 < 5%) and 10 (p-value=3.9e-8 < 5%), but there are no significant differences between presolve level 1 and 10 (p-value=0.41 > 5%).

Table 7. P-values of the Pairwise Wilcox tests.

	Pick				Place			
	Target		Bounce		Target		Bounce	
Presolve	0	1	0	1	0	1	0	1
1	4.7e-05	-	6.7e-14	-	1.0e-09	-	2e-16	-
10	2.0e-06	0.46	8.9e-11	0.0024	3.9e-08	0.41	2e-16	0.0037

In Fig. 6, it is also possible to observe, in different colors, the computational times to determine the bounce postures for picking and placing each of the five products on the shelves (actions 1–5) and moving the arm to the home position (action 6). It suggests differences for different actions. Actions 4 and 5 are comparable, as the pick movement occurs in a similar position and the place movement for these objects is to the same shelf, placed next to each other. In order to gain insight into the differences between similar movements, actions 4 and 5 are subjected to analysis. As previously observed, the target posture problems exhibit minimal variability between them. Table 8 displays data on the number of inequality constraints of these problems, the number of iterations required by IPOPT to reach an optimal solution, the discretization steps used for collision checking, and the minimum, median, and maximum computation times for the pick-and-place movements of two objects (actions 4 and 5) under different presolve settings.

Action 4 consistently has more inequality constraints than action 5 in both pick and place movements. In the pick movement, action 4 has 1071 inequalities with presolve 0, compared to action 5's 656, and this trend persists across all presolve levels. Similarly, in the place movement, action 4 has 774 inequalities with presolve 0, while action 5 has 501. Despite the reductions with an increase in the presolve, action 4 maintains more constraints in both movements. Iteration counts are similar between the actions, but the higher number of inequality constraints for action 4, especially in the pick movement, suggest greater complexity.

Following this, the time to reach an optimal solution for the bounce posture is studied. When comparing the computation time of the bounce posture problems in pick movements between actions 4 and 5, distinct differences emerge both in overall times and in the impact of varying presolve settings. In action 4, the bounce problem of the pick movement without presolve (presolve 0) takes significantly longer, taking a computation time 4.35 times greater than action 5. Specifically, action 4 has a median time of 1.046 s compared to 0.240 s for action 5. This indicates that the picking phase in action 4 is initially much more time-consuming. The difference can be explained by the starting positions: in action 4, the arm begins near the third shelf, while in action 5, the arm starts closer to the top shelf, resulting in a longer and more complex path for the robotic arm. When presolve is applied, action 4 shows considerable improvement. With presolve 1, the median computation time drops by 41% to 0.620 s, and further decreases to

Table 8. Statistics computation time with different presolve settings for the bounce posture in pick and place movements of object 4 and 5.

				Number of			Time (s)			
Action	Type	Mov	Presolve	Ineq	Iter	Nsteps	Min	Median	Max	N
4	Bounce	Pick	0	1071	40	23	1.011	1.046	1.075	30
		Pick	1	990	40	23	0.568	0.620	0.676	30
		Pick	10	814	11	23	0.152	0.165	0.191	30
		Place	0	774	25	20	0.407	0.422	0.493	30
		Place	1	701	25	20	0.231	0.247	0.277	30
		Place	10	568	25	20	0.209	0.225	0.249	30
5	Bounce	Pick	0	656	15	19	0.220	0.240	0.257	30
		Pick	1	599	15	19	0.132	0.137	0.146	30
		Pick	10	455	44	19	0.352	0.364	0.391	30
		Place	0	501	29	20	0.311	0.333	0.397	30
		Place	1	454	29	20	0.201	0.210	0.234	30
		Place	10	312	15	20	0.095	0.106	0.138	30

0.165 s with presolve 10. In contrast, action 5 follows a different pattern. While presolve 1 reduces the median time to 0.137 s, presolve 10 unexpectedly increases it to 0.364 s. This suggests that for action 4, increasing presolve consistently improves performance, while in action 5, higher presolve values do not offer the same benefit and may even hinder performance at the highest setting. For the place movements, the comparison between actions 4 and 5 is also noteworthy. Without presolve, action 4 has a median computation time of 0.422 s, which is slightly higher than action 5 where half of the problems have a solving time of 0.333 s or less. However, when presolve is applied, both actions see improvements, but to different extents. In action 4, the median time drops by 47% to 0.247 s with presolve 1, and further to 0.225 s with presolve 10. In action 5, the reduction is even more pronounced, with a 37% decrease to 0.210 s with presolve 1, and a reduction to 0.106 s with presolve 10.

A Kruskal-Wallis rank sum test was performed using the data of the bounce postures problems of actions 4 and 5, for different levels of presolve. It was obtained a p-value=2.2e-16<5% ($\chi^2 \simeq 108.98$), therefore there are significant differences in the computation time using different presolve settings to solve the bounce posture selection problems.

5 Conclusions

In general, the most efficient model simplifications obtained using the presolve algorithm implemented in AMPL were obtained for presolve level 1. For this, the computation time for both pick and place movements, presented significant reductions. In contrast, although presolve level 10 presented larger reductions on

the number of constraints of the optimization problems, does not consistently improve performance. Specifically, for pick movements, it increases both the number of iterations and computation time. Moderate presolve levels, such as level 1, may provide the best balance between constraint reduction and solver efficiency. Higher presolve levels, however, may degrade solver performance in certain cases.

Future work could focus on developing an algorithm to determine the most efficient presolve level for a given problem. Additionally, exploring how different parameters of the optimization problem, such as the initial guess, approach and retreat distances, or obstacle tolerances, affect the solving time, solution quality and complexity of the movement, could lead to a more effective parameter tuning for better results.

Acknowledgments. This work was supported by the project PT SMART RETAIL (ref. PRR 02/C05-i01.01/2022.PC645440011-00000062).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Costa e Silva, E., Costa, F., Bicho, E., Erlhagen, W.: Nonlinear optimization for human-like movements of a high degree of freedom robotics arm-hand system. In: Murgante, B., Gervasi, O., Iglesias, A., Taniar, D., Apduhan, B.O. (eds.) ICCSA 2011. LNCS, vol. 6784, pp. 327–342. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21931-3_26
2. Gulletta, G., Silva, E.C.e., Erlhagen, W., Meulenbroek, R., Costa, M.F.P., Bicho, E.: A human-like upper-limb motion planner: generating naturalistic movements for humanoid robots. Int. J. Adv. Robot. Syst. **18**(2) (2021)
3. AMPL Optimization: AMPL APIs. https://ampl.com/products/ampl/apis/. Accessed 27 Sept 2024
4. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. **106**, 25–57 (2006)
5. Tasseff, B., Coffrin, C., Wächter, A., Laird, C.: Exploring benefits of linear solver parallelism on modern nonlinear optimization applications. arXiv preprint arXiv:1909.08104 (2019)
6. Casanellas, G., Castro, J.: Using interior point solvers for optimizing progressive lens models with spherical coordinates. Optim. Eng. **21**(4), 1389–1421 (2020)
7. Thierry, D., Biegler, L.: The l1-exact penalty-barrier phase for degenerate nonlinear programming problems in ipopt. IFAC-PapersOnLine **53**(2), 6496–6501 (2020)
8. Hinder, O., Ye, Y.: A one-phase interior point method for nonconvex optimization. arXiv preprint arXiv:1801.03072 (2018)
9. Hogg, J.D., Scott, J.A.: On the effects of scaling on the performance of ipopt. arXiv preprint arXiv:1301.7283 (2013)
10. Kennedy, G., Fu, Y.: Topology optimization benchmark problems for assessing the performance of optimization algorithms. In: AIAA Scitech 2021 Forum. p. 1357 (2021)

11. Guamán, W., Pesántez, G., Falcones, S., Urquizo, J., et al.: Optimal dynamic reactive power compensation in power systems: Case study of ecuador-perú interconnection. Electric Power Syst. Res. **218**, 109191 (2023)
12. Puranik, Y., Sahinidis, N.V.: Domain reduction techniques for global NLP and minlp optimization. Constraints **22**(3), 338–376 (2017)
13. Jusevičius, V., Paulavičius, R.: Web-based tool for algebraic modeling and mathematical optimization. Mathematics **9**(21), 2751 (2021)
14. Fourer, R., Gay, D.M.: Experience with a primal presolve algorithm. Large Scale Optimization: State of the Art pp. 135–154 (1994)
15. Ribeiro, N., et al.: Rambo–robotic anthropomorphic manipulator for bimanual operations. In: 2024 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 157–164. IEEE (2024)