# Scaling up parallel robot plans to improve plan execution time: an industrial use-case

Edoardo Giordani[1], Sofia Santilli[1], Luca Iocchi[1], Fabio Patrizi[1] and Fabio Zonfrilli[2]

[1]*DIAG, Sapienza University of Rome, Italy*
[2]*P&G, Belgium*

**Abstract**
We present a greedy approach to generate parallel plans for domains featuring a large number of objects which need to undergo the same operational procedure. The number of objects is such that not even a total-order (linear) plan can be obtained, thus calling for an alternative method. We propose a compositional approach based on obtaining template solutions for the same problem but over small batches of objects and then combining them into a parallel, sub-optimal solution for the original problem. The approach is showcased on a real-world industrial use-case provided by the Procter&Gamble company in the context of the EU project AIPlan4EU.

**Keywords**
Plan execution, Robot planning, Parallel plans

## 1. Introduction

In this work we deal with an industrial use-case provided by the Procter&Gamble company (P&G) in the context of the EU project AIPlan4EU (www.aiplan4eu-project.eu). This is a real-world industrial use-case from the field of R&D consumer product testing.

In order to guide the scientists towards the evolution and patent of new superior products to be launched on the market, thousands of samples of the new proposals have to undergo a large number of different tests, so as to generate data to be subsequently examined. Typically, tests are performed manually in labs, resulting in a very slow and time-consuming activity for human operators. Since the activities to be carried out are rather repetitive, the use of a robotic system equipped with online sensors represents an attractive opportunity to standardize the testing process while dramatically improving its speed, as well as quality and reliability of the generated data. In this work we address the problem of automatically defining the behavior of such a system.

In the considered use-case, a robotic arm equipped with a gripper is employed to support quality control tests for laundry detergent soluble capsules, or *pouches*. Every test consists in measuring and recording weight, size, elasticity, strength and tightness of a pouch and, in order to guarantee statistical significance, a large number of tests must be performed on as many pouches. For simplicity, we only deal with weight, strength and tightness but the approach can be easily applied to the entire pool of features. The test unit consists of a robotic arm

*10th Italian Workshop on Artificial Intelligence and Robotics (AIRO 2023)*
giordani.2020434@studenti.uniroma1.it (E. Giordani); santilli@diag.uniroma1.it (S. Santilli); iocchi@diag.uniroma1.it (L. Iocchi); patrizi@diag.uniroma1.it (F. Patrizi); zonfrilli.f@pg.com (F. Zonfrilli)

(Universal Robot UR5e with Robotiq 2-Finger 85 adaptive gripper), a scale, a press for measuring the strength, a device for measuring the tightness, some drawers containing the pouches (to simplify the description here we assume only one drawer), and a bin to dispose of the tested pouches. The arm interacts with the various devices and the queue of incoming pouches.

Our work aims at automatizing the testing procedure. We want to automatically generate a plan that can be executed by all the involved devices (robotic arm, scale, press, etc.). Observe that each device acts independently of each other and actions have a (non-negligible) duration. For instance, moving a pouch from the drawer to the scale is not instantaneous and takes longer than weighing the pouch. Thus, in addition to constructing a plan that allows for successfully testing all the pouches, we desire the plan to be efficient, meaning that it can reduce the execution time by executing actions in parallel.

The problem is thus a multi-agent planning problem with durative actions, with the optimization goal of minimizing the total plan execution time. To the best of our knowledge, there is no available tool that solves the problem directly in a multi-agent setting, thus some approach is needed, which possibly takes advantage of existing technology. One of these consists in first finding a linear plan by suitably encoding the multi-agent problem as a single-agent one (this is possible as in our setting there are no privacy requirements) and then finding a possibly parallel rearrangement of the obtained plan to minimize the total duration, taking into account the capability of each device (which affects action parallelization) and the actions' duration.

The idea of rearranging a plan to relax the causal relationships among actions and facilitate parallelization has been deeply investigated in related literature, with notable examples including [1] where the NoLimit approach is described, which generates a partial-order solution from a total-order one, and [2], which introduces the notions of plan *deordering* and *reordering*, provides algorithms for rearranging a linear plan, and analyzes the problem in depth from the computational viewpoint. In the present setting, only the latter approach represents a viable alternative, as [1] does not deal with action duration, which are instead used here.

While theoretically sound, the approach of [2] turns out not to be effective, as it assumes a linear plan to be available. Yet, in the case at hand, the high number of pouches involved, close to 500, makes it impossible to even obtain the initial solution to be subsequently de/re-ordered. Not only. Even if the plan were available, computing the optimal parallel de/re-ordering of the initial plan would be extremely time-consuming, indeed the decision version of the optimization problem is NP-hard [2], thus making the approach not practical in our case.

In this work, we overcome the above problems by proposing a greedy sub-optimal approach to construct a parallel plan for the P&G task, which, while sub-optimal, allows for successfully producing a parallel plan for a number of pouches close to 500. The approach is based on the assumption that a repetitive task must be executed on a set of similar objects (pouches); it works by first constructing a set of parallel-plan templates for small batches of objects (as many as possible in a reasonable time) and then arranging the so-obtained templates into a larger plan that can process the entire set of objects. The reported experiments show that, besides being actually able to construct the plan, our approach yields a significant saving in execution time.

## 2. Problem description and solution algorithm

We address the problem of generating a parallel robot plan to process a large number of objects belonging to a (small) number of classes, while exhibiting an execution time as short as possible. Objects belonging to a same class must undergo the same operational procedure. Ideally, we would like to minimize (not just reduce) the execution time, yet the large number of objects that have to be processed makes finding optimal solutions infeasible in practice.

Specifically, in the P&G use-case described earlier, there are hundreds of pouches of two or three types that have to be manipulated by a robot arm, in order to perform several measurements. Pouches of the same type have to be processed in the same way to complete a set of measurements, which require interacting with specific instruments. During a measurement operation, the robot can manipulate other pouches (e.g., put another pouch on another device). The overall goal of the system is to minimize the plan execution time.

Addressing this problem requires facing the central issue of computing a plan, even non-parallel, over a domain featuring hundreds of objects, which proved impractical even for less than ten objects, due to the PSPACE-completeness of the problem, which essentially yields an exponential solution time. To overcome this, we propose the following alternative approach.

Firstly, we define a set of *batch planning problems*, each modelling a prototypical task over a *batch*, i.e., a suitably defined set containing only a small number of pouches (less than the original problem) that have to undergo the same process. Then, we solve all the batch problems. To this end, for each batch problem, we compute a linear plan and then, using the method described in [3], make it parallel. We call the so-obtained plans *batch parallel plans*. Observe that while we have used the (greedy and sub-optimal) approach of [3], any other method for computing a parallel solution to the batch problems can be used (as long as practical), e.g., [2]. Finally, we generate the *actual parallel plan* by instantiating and suitably composing the batch parallel plans into a larger parallel plan which fulfills the overall goal.

We stress that this solution does not guarantee optimality in terms of minimal execution time, but allows for obtaining a suitable trade-off between planning and execution time. Indeed, in the worst case, computing an optimal solution requires the planner to solve the problem on all the $n$ objects, yielding an extremely large planning time (if solvable at all), while with our approach the (exponential) planning time depends only on the batch size $k$, with $k << n$, while the generation of the actual parallel plan is linear in $n$, thus extremely faster than the general case.

In the experiments reported below, we show examples of plan execution with a total number of $n = 480$ pouches, using a batch size $k = 4$ to generate the batch parallel plans. That is, the proposed method generates a final plan for $n = 480$ objects, by solving several smaller problems with $k = 4$ prototypical objects.

Notice that with a batch size $k = 1$, the proposed method amounts to solving one problem for each (unique) object of every class and then replicating the plan for all the objects. Such a setting, though, produces no parallel actions, as parallelization is possible only when several objects can be processed at the same time. This is the baseline we used for our experiments.

| **Batch size** | **2** | **3** | **4** |
|---|---|---|---|
| Avg. time saved | 14m 38s | 20m 21s | 22m 55s |
| Avg. % time saved | 4.32 % | 5.86 % | 6.70 % |
| Max time saved | 43m 41s | 1h 1m 56s | 1h 11m 15s |
| Max % time saved | 5.83 % | 7.47 % | 7.99 % |

**Table 1**
Average and maximum reduction of execution time for different batch sizes.

## 3. Modelling and implementation

The problem is modelled using the Unified Planning (UP) framework (github.com/aiplan4eu/unified-planning), a Python framework devised within the AIPlan4EU Project, aimed at making the planning technology available and readily usable to the broad public. The specification of predicates and actions have been tuned with a bottom-up approach in order to guarantee a semantic alignment between symbols in the planning domain and the actual execution of the actions in the real setting. The modelling of the problem as a planning domain enforced the development of modular software components for the implementation of the basic actions.

The programming interface of the UP framework has been fundamental to properly integrate AI planning engines and other components in the overall solution, thus bringing the required modularity and flexibility that was a primary objective for the considered use-case.

As already mentioned, the main experimental objective has been to measure the improved performance (in terms of minimizing the execution time) when using the proposed approach to generate parallel plans for many objects, with respect to a sequential plan execution.

## 4. Results in the real setup

Results obtained in comparison of the different formalizations are summarised below. More specifically, we measured the reduction of plan execution time when using the parallelized plan with respect to the sequential one. We focused on the analysis of plan execution time, since plan generation time is negligible with respect to plan execution.

Details of the experimental results conducted on the real setup (real robot operations) are shown in Table 1, considering different batch sizes. The Table shows significant performance improvements increasing with the batch size. The percentage improvements correspond to an actual production improvement for the P&G use-case. For example, a typical daily operation to process 480 pouches enables a performance gain of 1h 11m (i.e., 16h 15m using a parallel plan vs. 17h 26m using a sequential plan).

As shown in Figure 1, the plan size (i.e., number of total actions executed by the robot) increases linearly with the number of pouches, reaching a size of over 5,000 actions to process 480 pouches. Plan execution and the difference between parallel and sequential execution also increases lineraly with the number of pouches.

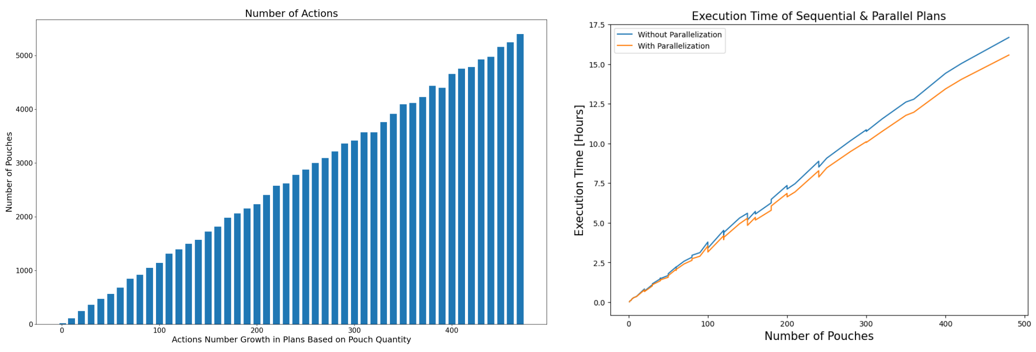

**Figure 1:** Execution time and plan size increase over the number of pouches.

## 5. Discussion and lessons learned

The developed solution satisfies all the requirements of the use-case and has great potential to actually increase the overall productivity of the system, in particular a more efficient use of the robotic platform. The use of the UP framework was fundamental to obtain this result, since no single planning engine would have been sufficient to solve the problem. In fact, the solution effectively integrates several UP components: Oneshot planners, compilers, and sequential simulators.

Lessons learned include an approach to apply AI planning techniques to industrial scenarios in which an operational solution is already existing, but it is not flexible and easily extensible. For example, it cannot be easily adapted to new situations or to achieve new goals. In this context, we applied a methodology based on modularizing the existing solution and building (through an iterative process) an AI planning domain to describe the modules (in terms of actions) and their properties (as predicates and fluents). Such a planning domain can then be used to solve different problems varying initial states and goals, thus overcoming the difficulties arising from a non-flexible and non-modular solution. Modularity is a fundamental prerequisite to apply AI planning technology and, in turn, applying AI planning forces the development of modular domain-specific components.

## Acknowledgments

The work of Fabio Patrizi was partially supported by projects EU ICT-49 2021 AIPlan4EU (No. 101016442), ERC Advanced Grant WhiteMech (No. 834228), and PNRR MUR PE0000013-FAIR. The work of Luca Iocchi was partially supported by projects EU ICT-49 2021 AIPlan4EU (No. 101016442) and the PNRR MUR PE0000013-FAIR. The work of Sofia Santilli was partially supported by projects EU ICT-49 2021 AIPlan4EU (No. 101016442) and the Sapienza Project MARLeN.

We thank Andrea Micheli and Erez Karpas for their useful comments about nonlinear plans and plan re/de-ordering.

## References

[1] M. M. Veloso, M. A. Pérez, J. G. Carbonell, Nonlinear planning with parallel resource allocation, in: Proceedings of the DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control, Morgan Kaufmann, San Diego, CA, 1990, pp. 207–212.

[2] C. Bäckström, Computational aspects of reordering plans, J. Artif. Intell. Res. 9 (1998) 99–137. URL: https://doi.org/10.1613/jair.477. doi:10.1613/jair.477.

[3] S. Santilli, A. Trapasso, L. Iocchi, F. Patrizi, A novel algorithm for parallelizing actions of a sequential plan, in: Proc. of PlanRob Workshop (ICAPS), 2023.