ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA241019

Planning for Human-Robot Collaboration Scenarios with Heterogeneous Costs and Durations

Silvia Izquierdo-Badiola[a,c,*], **Gerard Canal**[b], **Guillem Alenyà**[a], **Carlos Rizzo**[c] **and Andrew Coles**[b]

[a]Institut de Robotica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain
[b]Department of Informatics, King's College London, UK
[c]Eurecat, Centre Tecnològic de Catalunya, Robotics and Automation Unit, Cerdanyola, Spain

Abstract. This paper looks at human-robot collaboration (HRC) scenarios, in particular where the durations and costs of the actions are heterogeneous between agents, reflecting the agents' capabilities as well as environmental constraints. We explore the use of temporal PDDL planning as a means of finding over-arching task plans for such collaborative scenarios, and apply suitable heuristics and search algorithms to improve the extent to which plans can be found that are sensitive to combined duration and cost metrics. An evaluation in a kitchen scenario shows our approach is effective, finding cost-effective task plans compared to those from existing planners, and a hand-crafted baseline.

1 Introduction

Efficient task planning plays a crucial role in achieving effective performance in robotics applications. One common aspect in many robotic applications is the existence of actions that not only have a duration but also incur a certain cost. Considering both is especially important in multi-agent applications, and a widespread example of such applications consists of robots acting in collaborative environments, where human–robot teamwork is leveraged to achieve complex tasks. The actions involved in these domains vary in duration and cost depending on both environmental and agent factors such as capabilities or preferences. In such settings, the efficient production of task plans comprising actions for both humans and robots is crucial to maximize productivity, minimize costs, and ensure the completion of shared goals in a timely manner.

Motivated by the need to enhance the efficiency of human–robot collaboration (HRC), this paper presents novel methods to address the challenge of optimizing a plan quality metric that combines makespan (total time to complete the plan) with plan cost. While recent PDDL planners have proven successful at optimizing either cost or makespan, they often struggled to achieve a satisfactory balance between the two objectives. Taking either alone is not appropriate in HRC tasks: focusing solely on makespan may result in excessive costs; while focusing on costs could lead to unnecessarily prolonged makespan. Therefore, the development of a PDDL planner that can optimize a metric combining both of these is of great importance in allowing PDDL-based approaches to be used within compelling robotic applications. Our proposed method fills this gap, considering additional heuristic measures that capture both makespan and cost, and developing further a known-effective anytime search algorithm [25]. While general in nature, our implemented planner OPTIC-AEES embeds these within the planner OPTIC [2]. As a running example, we use a household robotics scenario where shared goals, corresponding to tasks such as cooking or cleaning, have to be assigned to a human and a robot, performing actions concurrently in a kitchen, such as the one seen in Figure 1.

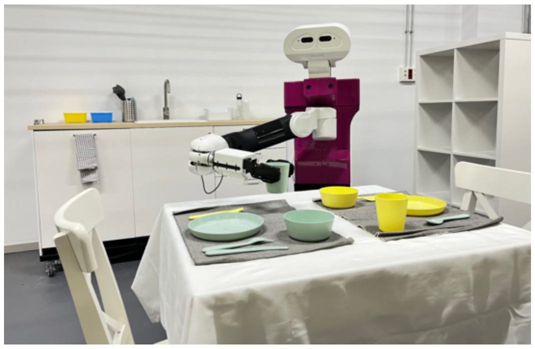
Figure 1. Example Human–Robot Collaboration Scenario where a Robot Assists in Household Tasks.

The remainder of this paper is structured as follows: Section 2 reviews related work, leading on to Section 3 which discusses the problem of modeling human-robot collaboration as a planning problem. Motivated by this, Section 4 details the planner optimizations necessary for the development of OPTIC-AEES. An evaluation in a kitchen scenario is presented in Section 5, showing our planner represents the new state-of-the-art for this scenario; and exploring its performance in the context of prior work, a handcrafted baseline, and ablations with respect to its own feature set. Finally, Section 6 concludes the paper and discusses future directions.

2 Related Work

Task Planning Background. A PDDL2.1 [16] planning problem is defined over a collection of propositions P, and a vector of numeric variables $\mathbf{v}$. These are manipulated and referred to by actions. The executability of actions is determined by their preconditions, conjunctions of *conditions*. A *condition* is either a single proposition $p \in P$, $\neg p$, or a numeric constraint over $\mathbf{v}$. We assume all such constraints are linear, and hence can be represented in the form $\mathbf{w}.\mathbf{v}\{>, \geq, <, \leq, =\}c$ where $\mathbf{w}$ is a vector of constants and c is a constant.

* Corresponding Author. Email: sizquierdo@iri.upc.edu.

Each durative action A has three sets of preconditions: $\text{pre}_{\vdash}(A)$, $\text{pre}_{\leftrightarrow}(A)$, $\text{pre}_{\dashv}(A)$. These represent the conditions that must hold at its start, throughout its execution ('over all' conditions), and at its end, respectively. Instantaneous effects can occur at the start or end of A: $\text{eff}^{+}_{\vdash}(A)$ ($\text{eff}^{-}_{\vdash}(A)$) denote propositions added (resp. deleted) at the start; $\text{eff}^{num}_{\vdash}(A)$ denotes numeric effects. Similarly, $\text{eff}^{+}_{\dashv}(A)$, $\text{eff}^{-}_{\dashv}(A)$ and $\text{eff}^{num}_{\dashv}(A)$ record effects at the end. We assume numeric effects are of the form:

$$v\{\text{+=}, \text{-=}, =\}\mathbf{w} \cdot \mathbf{v} + c \qquad (v \in \mathbf{v})$$

Finally, the action has a duration constraint: a conjunction of numeric constraints applied to a special variable ?duration, denoting its duration.

As a special case, *instantaneous* actions have duration zero, and only one set of preconditions $\text{pre}(A)$ and effects $\text{eff}^{+}(A)$, $\text{eff}^{-}(A)$, and $\text{eff}^{num}(A)$. Otherwise, following LPGP [22], a durative action A can be split into two instantaneous *snap*-actions, $A_{\vdash}$ and $A_{\dashv}$, representing the start and end of the action respectively, and a set of constraints ('over all' and duration constraints). Action $A_{\vdash}$ has precondition $\text{pre}_{\vdash}(A)$ and effects $\text{eff}^{+}_{\vdash}(A)$, $\text{eff}^{-}_{\vdash}(A)$, $\text{eff}^{num}_{\vdash}(A)$. $A_{\dashv}$ is the analogous action for the end of A.

A solution to the problem is a *plan*: a timestamped sequence of actions with associated durations, that transforms the initial state I into some state s that satisfies the goal G. All pre/'over all' conditions must be satisfied at the time of/during execution and actions that have started must have finished. The quality *metric* of a plan is defined in terms of the values of state variables in s, and/or the *makespan* of the plan: the latest time at which an action in the plan ends. For our purposes, we assume the metric is to minimize a positive-weighted sum of plan makespan, and a single state variable total-cost; such that total-cost initially takes the value 0, and all effects on it by actions are of the form total-cost += c, $c \in \mathbb{R}_{\geq 0}$. Under this assumption we can say that the *cost* of a snap-action is the value c by which it increases total-cost (or 0 otherwise).

Our approach builds on the planner OPTIC [2], which performs heuristic forward-search for plans, starting at the initial state, using a Simple Temporal Problem [12] or Mixed Integer Program (MIP) to manage the temporal constraints, and reporting solution plans of increasing quality as they are found. OPTIC's search is guided by a TRPG heuristic: a Temporal variant of the Relaxed Planning Graph heuristic [19]. The TRPG heuristic, evaluated for a state s, returns either a *relaxed* plan $\pi(s)$ that satisfies the goals under the conditions of the relaxation (*inter alia*, that delete effects are ignored), or recognizes that s is a dead-end state from which the goals cannot be reached. A relaxed plan $\pi(s)$ can be represented as a time-stamped sequence of snap actions $[\langle a_{0x}, t_0\rangle..\langle a_{nx}, t_n\rangle]$. From $\pi(s)$ we can obtain two measures:

- $d(s) = |\pi(s)|$: the 'distance to go' measure for s, taken to be how many actions are needed for its relaxed plan;
- $hm(s) = \max((\max_{\langle a_{ix}, t_i\rangle \in \pi(s)} t_i), m(s))$: an admissible heuristic makespan for s, which is the maximum of either the makespan of the relaxed plan (the time of the latest snap-action) or $m(s)$ – the makespan already committed to in the actions taken to reach s.

OPTIC guides the search according to $d(s)$; and in problems where the metric is to minimize makespan, and a solution of cost c has already been found, it prunes any state s where $hm(s) \geq c$.

Note that while we build on OPTIC, our work is applicable to planners taking other approaches to managing temporal constraints in forward planning, such as the decision epoch approach [11] used in TFD [14] and SAPA [13]. We also note that SAPA has a range of available relaxed planning graphs (RPGs) that, for instance, can be built based on combined cost and makespan estimates; but it remains an open question how to apply such RPGs to non-decision-epoch planners such as OPTIC.

Task Planning for HRC. Work in the human-robot collaboration has traditionally focused on making plans for humans and robots, and improving the planning approach. For instance, Bezrucav and Corves [4] looked into improving the cooperation of teams of humans and robots using a three-level planning approach that decomposes the planning process into different goals and executors, while optimizing the plan makespan. Other approaches look into distributing the tasks between agents while monitoring the agent status to prevent failures [20]. In these cases, the agent is modeled via different action parameters affecting its cost, and the planner is intended to optimize the plan duration, which means that the cost is overlooked. Similar approaches have also included other human factors as the costs of the planning actions. Canal et al. [6] instead model the user in terms of preferences, which affect the cost of the actions. Alternative methods compute composite plans represented as a union of individual plans for each of the agents [8]. In this case, the cost is used as a metric and action durations are not considered. Where the problem to be solved is one of task allocation, rather than planning in general, other work has looked at encoding this as a mixed integer-linear program [21, 18]. Hierarchical Task Networks have also been used by Alami et al. [1], where a plan for the robot and the human is found while taking into account costs and utilities, and also considering verbal communication costs [5], but action durations are not considered. Task allocation between humans and robots is optimized by Cheng et al. [10], where parallel relationships are found to optimize actions that may be done in parallel with that of the human, but agent states are not considered. Similarly, Chen et al. [9] presented a Dual Generalized Stochastic Petri Net used to allocate assembly tasks to humans and robots while considering time and allocation costs as a Monte Carlo multi-objective optimization. Beyond Human-Robot scenarios, work has considered the multi-objective optimization problems inherent in combined task- and motion-planning [15] but this is somewhat orthogonal to our work which focuses on the task-planning level, with the execution of plans subsequently handled by a plan execution framework such as ROSPlan [7].

3 The Characteristics of HRC as a Planning Problem

We take a human-robot collaboration problem as a motivation example of temporal planning where optimizing a metric combining time and cost may be crucial to the success of the task. In human-robot collaboration problems, the objective at a task planning level is to find a plan assigning actions to both human and robot agents in order to achieve shared goals. We assume the metric to be minimized of a plan p is a weighted sum $g(p) = \alpha \cdot m(p) + (1 - \alpha) \cdot c(p)$ where $m(p)$ is the makespan of p (the time taken to execute the plan), $c(p)$ is the total cost of the actions in p, and $\alpha \in [0, 1]$ weights the metric towards makespan and/or cost respectively. We follow the approach of [20] of assuming that one can find a plan for a human that either they will follow, or for which it is assumed that human deviations from the plan will be handled by replanning from the state reached by the human's deviation. The problem is represented as a domain-independent PDDL model, incorporating any required physical agent interactions or coordination aspects at task planning level, such as ensuring that no two agents occupy the same space at once.

As humans and robots are highly heterogeneous, the actions available for humans may be different to those for robots; and where actions for the humans and robots are similar (e.g. achieve the same goal or subgoal), they may differ in terms of their durations and/or the costs incurred, as they are not only based on environmental factors but also on the agents' preferences and unique capabilities. For instance, some actions may require less time for the human to perform but have a higher associated cost due to a human's preference not to carry them out. For the human agent, the cost of an action could indicate their preference to avoid performing the action, even if it means extending the overall makespan: if an action has a cost of x, it means that the human would rather the makespan was longer by $x \cdot ((1-\alpha)/\alpha)$, than the action was used. This cost would represent the inconvenience or effort required from the human's perspective, with $((1-\alpha)/\alpha)$ being the 'exchange rate' between cost and duration. On the other hand, for the robot agent, costs could have a number of meanings. If cost reflects battery usage, then a good plan will balance energy usage versus makespan. Or, the cost of an action could reflect the probability of failure, and the effort to overcome it: assigning an action a cost y would be principled if the expected time for which plan execution would need to be interrupted to overcome the failure is $y \cdot ((1-\alpha)/\alpha)$.

Consider an example HRC problem, where a human and a robot need to perform cooking and cleaning tasks in a kitchen. In such a scenario, the robot may be better suited to perform some of the tasks (e.g. it might be forbidden from tasks that deal with heat), while the human may have preferences on what to do (e.g. they may prefer not to clean the floor). These capabilities and preferences may be expressed as action costs [6, 20], such that all things considered equal, the planner will seek to avoid assigning those actions to the robot or the user. Then, timing is also important. For instance, the human and robot may take different amounts of time to complete actions that achieve (sub)goals; there may be different actions taking different durations, such as cooking something in the pan (quicker) or in the oven (slower); and moving around the kitchen from station to station also takes time. In such a case, the planner would have to balance costs with the temporal consequences of the actions taken.

Hence, from a task planning point of view, we have an interesting problem to solve (there are interestingly different ways to solve the problem), and an interesting quality metric: for the planner to find a good overall plan, with actions for both humans and robots, it is crucial to consider both makespan and cost simultaneously. Simply optimizing for the cheapest way to solve a problem may lead to plans with inefficient makespans; while only optimizing for makespan may lead to plans that are of unreasonably high cost.

4 Planner Optimizations for HRC

To be able to find usable solutions to our HRC problems, we need a planner that is able to find plans that are of good quality with respect to minimizing a weighted sum of makespan, and action costs. Planners such as OPTIC [2] and LPG-TD [17] are in principle able to do so, but empirically we noted the quality of the plans they found to fall behind what we knew to be theoretically possible.

Focusing on OPTIC, its performance was limited by a number of factors. First, its makespan estimate for reaching the goals from a given state s is based on a Temporal Relaxed Planning Graph (TRPG). This is admissible, but highly optimistic as the relaxation (to ignore delete effects, and relax numeric effects) supports a far greater degree of concurrency than is realistically possible. Second, its anytime search for plans of increasing quality uses WA* based on distance-to-go, $d(s)$; with the TRPG makespan estimate $hm(s)$ only used for pruning. Due to the optimism of the TRPG makespan estimate, this pruning is relatively weak. Moreover, while searching based on distance-to-go is a reasonable choice in the domains with preferences for which OPTIC was designed (its distance-to-go reflects the distance to satisfy any presently unsatisfied preferences, in addition to the hard goals – hence is strongly correlated with cost-to-go), this translates less well to problems without preferences, where distance-to-go is less strongly correlated with cost-to-go.

4.1 Inadmissible makespan and cost measures

Considering first how we can find a more informative makespan measure, at the price of forgoing admissibility, we inflate the makespan estimate from a temporal relaxed plan by considering mutexes between the actions chosen. We assume we have a function $mutex(a, a')$ which returns true if the execution of the actions a and a' cannot overlap; in our case, it is defined according to the analysis described in [3]. With this, we construct Algorithm 1[1], which works repeatedly iterating through the relaxed plan, to find an inadmissible makespan estimate $\widehat{hm}(s)$. It keeps track of which actions are currently executing (initially those in s – initialized at line 4), and the $facts$ that have been reached, along with the time they were achieved (use); and uses this to delay starting actions that are $mutex$ with one that is currently executing or has been executed. The practical upshot of this in our HRC problem, where all the actions for a given agent are pairwise mutex, is that $\widehat{hm}(s)$ would be inflated due to the actions for each agent being applied in series rather than in parallel.

In more detail, line 23 identifies start snap-actions that should be deferred to the next iteration through the relaxed plan (by being added to π'). This happens if either its propositional preconditions are not satisfied[2]; the subsequent propositional 'over all' conditions would not be satisfied; or if it would start an action that is mutex with one that has not yet ended. Otherwise, it can happen, but we compute the time (t'_i) this would be at, as the latest of: (i) when it originally occurred in the relaxed plan; (ii) the time it would need to go at to satisfy its preconditions; (iii) the time at which it would need to go to satisfy the subsequent 'over all' conditions; and (iv) ordering it after any prior actions it is mutex with (taken from $done$). t'_i is then used to update $\widehat{hm}(s)$, to set the time at which any new facts are achieved (line 28) and to note action a_i can end at time $t'_i + dur_{\min}(a_i)$ (line 32), where $dur_{\min}(a_i)$ is a global lower-bound on the duration of a_i.

End snap-actions are the simple case: if we are considering an end snap-action $a_{i\vdash}$, and it is not currently executing ($endat[a_i] = \emptyset$), then we defer it to the next iteration by adding it to π'. Otherwise, we apply it now: we update our makespan estimate $\widehat{hm}(s)$ to be at least the time at which the action would end; update $done$ to note that it has finished; and update $facts$ and use for any new facts it has added. An important difference to the case for start snap-actions is we do not check that the preconditions of $a_{i\dashv}$ hold. This is to avoid deadlock in the case where starting an action a (applying $a_{\vdash}$) blocks starting an

[1] For brevity, the algorithm does not explicitly consider instantaneous actions. WLOG, instantaneous actions can be replaced by equivalent zero-duration durative action, with all preconditions and effects at the start, and all other components empty.

[2] Note we do not check numeric preconditions; we degrade gracefully in the presence of these, as t'_i is at least never less than t_i – the time snap-action a_{ix} occurred in the relaxed plan.

Algorithm 1: Inflating a relaxed plan makespan estimate using action mutexes

```
Data: A state $s$, a relaxed plan $\pi = [\langle a_{0x}, t_0\rangle .. \langle a_{nx}, t_n\rangle]$
Result: An inadmissible makespan estimate $\widehat{hm}(s)$
1  $\widehat{hm}(s) \leftarrow$ the makespan of the plan to $s$;
2  $endat \leftarrow \emptyset$;
3  foreach action $a$ executing in $s$, with minimum start time $t$ do
4      $endat[a] \leftarrow endat[a] \cup \{(t + dur_{min}(a))\}$;
5  $facts \leftarrow$ the facts in $s$;
6  $use[f] \leftarrow 0$ for each $f \in facts$;
7  $done \leftarrow \emptyset$;
8  while $\pi \neq \emptyset$ do
9      $\pi' \leftarrow []$;
10     foreach $\langle a_{ix}, t_i\rangle \in \pi$ do
11         if $a_{ix}$ is an end snap-action $a_{i\vdash}$ then
12             if $endat[a_i] = \emptyset$ then
13                 $\pi' \leftarrow \pi' + [\langle a_i, t_i\rangle]$;
14                 continue;
15             remove smallest $t$ from $endat[a_i]$;
16             $t'_i \leftarrow \max(t, t_i)$;
17             $\widehat{hm}(s) \leftarrow \max(\widehat{hm}(s), t'_i)$;
18             $done \leftarrow done \cup \{\langle a_i, t'_i\rangle\}$;
19             foreach $f \in (eff^+(a_{i\dashv}) \setminus facts)$ do
20                 $facts \leftarrow facts \cup \{f\}$;
21                 $use[f] \leftarrow t'_i$;
22         else
23             if $(pre(a_{i\vdash}) \not\subseteq facts)$
                   $\vee\ (pre(a_{i\leftrightarrow}) \not\subseteq (facts \cup eff^+(a_{i\vdash})))$
                   $\vee\ (\exists a' s.t. endat[a'] \neq \emptyset \wedge mutex(a_i, a'))$
               then
24                 $\pi' \leftarrow \pi' + [\langle a_i, t_i\rangle]$;
25                 continue;
26             $t'_i \leftarrow \max(t_i, \max_{f \in pre(a_{i\vdash})} use[f],$
                   $\max_{f \in (pre(a_{i\leftrightarrow}) \setminus eff^+(a_{i\vdash}))} use[f],$
                   $\max(\{t' \mid \langle a', t'\rangle \in done \wedge mutex(a_i, a')\}))$;
27             $\widehat{hm}(s) \leftarrow \max(\widehat{hm}(s), t'_i)$;
28             foreach $f \in (eff^+(a_{i\vdash}) \setminus facts)$ do
29                 $facts \leftarrow facts \cup \{f\}$;
30                 $use[f] \leftarrow t'_i$;
31             $t'_i \leftarrow t'_i + dur_{\min}(a)$;
32             $endat[a_i] \leftarrow endat[a_i] \cup \{t'_i\}$;
33     $\pi \leftarrow \pi'$;
34 return $\widehat{hm}(s)$;
```

action b where $mutex(a, b)$; but where b was (directly or transitively) necessary in the relaxed plan to achieve the end conditions of a. Not checking the preconditions of $a_{i\dashv}$ releases this deadlock, by ensuring once an action has been started, it can always be ended; so b would be able to be applied on a subsequent iteration through the relaxed plan.

Moving on from finding an inadmissible makespan estimate, we can find an inadmissible cost estimate to reach the goals from a state, by taking the sum of the costs of actions in the relaxed plan π [13]. For a state s we denote this $\widehat{hc}(s)$. With this, a naïve heuristic measure $nh(s)$ for states in our HRC tasks, whose quality metric is a weighted sum of cost and makespan, could be defined as:

$$\widehat{nh}(s) = w_m.(\widehat{hm}(s) - m(s)) + w_c.\widehat{hc}(s)$$

...where $m(s)$ is the makespan of the plan steps that reached s. This yields a naïve $\hat{f}$ value $\widehat{nf}(s) = g(s) + \widehat{nh}(s)$. However, this still has its shortcomings, not least as both $\widehat{hm}(s)$ and $\widehat{hc}(s)$ are subject to the vagaries of the relaxed plan, which is extracted via a greedy algorithm from the temporal RPG. To address this, we turn to online correction-learning approach of Thayer et al. [24], to learn global 'mean one-step errors' during search, and use these to refine this naïve heuristic. The one-step errors on distance-to-go and our naïve heuristic cost-to-go for the expansion of a state p are:

$$\epsilon_{dp} = (1 + d(bc(p))) - d(p)$$
$$\epsilon_{nhp} = \widehat{nf}(bc(p)) - \widehat{nf}(p)$$

...where $bc(p)$ is the best child of p. To estimate $bc(p)$, as per [24] we assume it is the node with minimum $\widehat{nf}(n)$ among all of p's children, breaking ties on $\widehat{nf}(n)$ in favor of low $d(n)$. The mean values of these across all expansions, $\bar{\epsilon}_d^{global}$ and $\bar{\epsilon}_{nh}^{global}$, yield what we will use as an inadmissible f value:

$$\hat{f}(s) = g(s) + \widehat{nh}(s) + \frac{d(s)}{1 - \bar{\epsilon}_d^{global}} \cdot \bar{\epsilon}_{nh}^{global}$$

..with the latter term here equating to a 'per-d' correction added to $\widehat{nf}(s)$.

4.2 Searching for better plans

Now we have an inadmissible $\hat{f}(s)$, the question is what to do with it. It would not be appropriate to use it as a hard pruning heuristic to complement OPTIC's 'WA* on distance-to-go', as it is not admissible; but it has potential for use for prioritizing the expansion of some states over others. To do this, we turn to Anytime Explicit Estimation Search (AEES) [25], which can use an inadmissible heuristic for exactly this purpose.

AEES maintains a suboptimality bound w of the incumbent best solution (initially, $w = \infty$), and uses this to bias search more or less towards expanding nodes that are close to goal states (low $d(n)$), or have attractive costs (low $\hat{f}(n)$ or $f(n)$). It considers at each iteration of search selecting one of the following candidate nodes for expansion[3]:

$$best_f = \text{argmin}_{n \in open}\, f(n)$$
$$best_{\hat{f}} = \text{argmin}_{n \in open}\, \hat{f}(n)$$
$$best_d = \text{argmin}_{n \in open \wedge \hat{f}(n) \leq w \cdot \hat{f}(best_{\hat{f}})}\, d(n)$$

Note $best_d$ is taken from a focal list of states: those whose $\hat{f}(n)$ values are within the bound $w \cdot best(f)$. Which of these three candidates is chosen and expanded is according to the following rules:

- If $\hat{f}(best_d) \leq w \cdot f(best_f)$ choose $best_d$;
- Else, if $\hat{f}(best_{\hat{f}}) \leq w \cdot f(best_f)$ choose $best_{\hat{f}}$;
- Otherwise, choose $best_f$.

The suboptimality bound w is updated every time a new goal state is found with $g(n)$ below that of the incumbent best solution. Recalling that $best_f$ is the node on the open list with best f value,

[3] In the general case, $best_d$ is correctly given as $best_{\hat{d}}$, but they are equivalent when using a global one-step-error model on d.

$f(best_f)$ gives us a lower bound on reachable solution cost, so w is set to $g(n)/f(best_f)$.

On initial experimentation, we noted two limitations of a straightforward usage of AEES. First, suppose we are yet to find a solution plan (so $w = \infty$), and expand a state s_0 generating two successor states s_1, s_2, with $d(s_1) = d(s_2) = 1$; and that this distance-to-go measure is the best seen so far, and is perfect: both are one step away from goal states, s_{g1} and s_{g2}, respectively. Also suppose $g(s_{g1}) > g(s_{g2})$: s_2 leads to the better goal state. If s_1 and s_2 are inserted in that order into the open list with the standard assumption of stable insertion, s_1 would be ordered before s_2. Then, on the next iteration, as $w = \infty$, we know $best_d = s_1$; this would be expanded, finding s_{g1}; and in turn, w would be recalculated as $g(s_{g1})/f(best_f)$. If this now-reduced w value means s_2 is not eligible for expansion as $best_d$, due to its $\hat{f}(s_2)$ value, then it may be some search iterations before s_2 is expanded, reaching s_{g2}. To address this, as a branch ordering heuristic, we sort the successors of each state prior to insertion into the open list into ascending $\hat{f}$ order.

Second, in prior implementations of AEES, the $\hat{f}$ value of state s is fixed at whatever $\hat{f}(s)$ was at the moment it was inserted into the open list; even if its value was computed based on a global heuristic error model whose per-d correction is subject to change. Our initial experiments showed this biased search towards expanding nodes that were added to the open list earlier in search (where mean one-step error was smaller), at the expense of expanding better states that were found later in search. To address this, we exploit the fact that the correction component in $\hat{f}(n)$ is a function only of $d(n)$, and not any other features of n, to be able to efficiently recompute $\hat{f}(n)$ at each expansion. Suppose we had an open list $open$, with $ins(n) < ins(n')$ if n was inserted into it before n' (capturing the stable insertion property), and derive from this per-d open lists each $open_i = \{n \in open \mid d(n) = i\}$. The $\hat{f}(n)$ value of each $n \in open_i$, according to the current global error model, can be written $\hat{f}(n) = \widehat{nf}(n) + ct(i)$, where $ct(i)$ is the correction term $\frac{i}{1-\bar{\epsilon}_d^{global}} \cdot \bar{\epsilon}_{nh}^{global}$ according to the global error model.

In Algorithm 2 we show how this observation can be used to efficiently compute $best_{\hat{f}}$ according to the current global error model. For each possible distance-to-go i, we identify the best node on $open_i$ according to the components of $\hat{f}(n)$ that do not vary under the global error model (the best $\widehat{nf}(n)$ – line 4), then add the correction term $ct(i)$ for nodes with $d(n) = i$ to this to compute $best^i_{\hat{f}}$: the candidate for $best_{\hat{f}}$ among nodes with $d(n) = i$. If this is a new best $best_{\hat{f}}$, or it is tied but was inserted first ($ins(best^i_{\hat{f}}) < ins(best_{\hat{f}})$), it becomes the new incumbent $best_{\hat{f}}$.

In Algorithm 3 we show how to compute $best_d$ based on the per-d open lists. This is somewhat more straightforward: going through the open lists in ascending order of $d(n)$, we build the focal list for each: the list of states where $\widehat{nf}(i) + ct(i)$ is within the dynamic quality threshold $w \cdot \hat{f}(best_{\hat{f}})$. As soon as one of the focal lists is non-empty, we have found $best_d$ and can return it: no later open list would find a candidate with lower $d(n)$.

Having found candidates for $best_{\hat{f}}$ and $best_d$ according to these algorithms, the core AEES candidate selection rule, in theory remains unchanged; the difference is that its practical application now makes decisions according to $\hat{f}$ values based on the current global error model, rather than fixing $\hat{f}(n)$ at the point of insertion.

Finally, to note, while we build on OPTIC, this development of AEES is not tied in any way to the choice of heuristic or the vagaries of any planner in particular. Similarly, while Algorithm 1 is planning-specific, it relies only on a temporal relaxed plan, which is commonly available in other temporal planners [13].

Algorithm 2: Identifying $best_{\hat{f}}$ from per-d open lists

```
Data: Per-d open lists $[open_0..open_m]$ for $d(n) \in [0..m]$
Result: $best_{\hat{f}}$
1 $best_{\hat{f}} \leftarrow \bot$;
2 $\hat{f}(best_{\hat{f}}) \leftarrow \infty$;
3 foreach $i \in [0..m]$ do
4     $best^i_{\hat{f}} \leftarrow \mathrm{argmin}_{n \in open_i} \widehat{nf}(n)$;
5     $\hat{f}(best^i_{\hat{f}}) \leftarrow \widehat{nf}(best^i_{\hat{f}}) + ct(i)$;
6     if $(\hat{f}(best^i_{\hat{f}}) < \hat{f}(best_{\hat{f}})) \vee (\hat{f}(best^i_{\hat{f}}) = \hat{f}(best_{\hat{f}}) \wedge ins(best^i_{\hat{f}}) < ins(best_{\hat{f}}))$ then
7         $\hat{f}(best_{\hat{f}}) \leftarrow \hat{f}(best^i_{\hat{f}})$;
8         $best_{\hat{f}} \leftarrow best^i_{\hat{f}}$;
9 return $best_{\hat{f}}$;
```

Algorithm 3: Identifying $best_d$ from per-d open lists

```
Data: Per-d open lists $[open_0..open_m]$ for $d(n) \in [0..m]$
Result: $best_d$
1 foreach $i \in [0..m]$ do
2     $focal = [n \in open_i \mid \widehat{nf}(n) + ct(i) \leq w \cdot \hat{f}(best_{\hat{f}})]$;
3     if focal is not empty then return the first $n \in focal$ ;
4 return $\bot$;
```

5 Evaluation

Having described Human–Robot Collaboration tasks at an abstract level, and presented planning techniques for use in the planning models of such tasks – with metrics that refer to both cost and makespan – we now describe an example HRC scenario, and use it as a basis of an evaluation.

5.1 Evaluation HRC Tasks

For our evaluation, we use an HRC task consisting of a kitchen environment where there are a set of shared goals to achieve, each corresponding to having completed a kitchen task; and actions must be executed by a human and a robot to achieve these goals. Two types of tasks have been defined, namely cooking and cleaning, and both can be achieved in a number of different ways, incurring different costs and durations for the two agents involved. With respect to the available equipment, cooking can be done on the stovetop or in the oven, while cleaning can be done with a mop or with a cloth. We focus on the problem of task assignment as a fundamental element of collaborative task planning, where a number of cooking and cleaning tasks need to be assigned and executed in parallel by collaborating agents. These high-level tasks involve lower-level intermediate actions that need to be planned for, defined in the planning domain and dealt with by the planner.

The snippet in Figure 2 shows part of the scenario definition in PDDL. The environment is specified in terms of agents (*agent*), objects (*obj*) and locations (*loc*). The tasks of cooking and cleaning are represented by predicates, which can be specified as part of the goal defined in the problem. The *cooked* predicate has a parameter of type *obj* (object), while *cleaned* has a parameter of type *loc* (location). The functions represent the costs and the durations of the actions, which are associated with an agent. Taking a look at the action *clean_cloth*, it can be seen how *(cleaned ?loc)* is part of its effects, achieving a cleaning task, and potentially completing part of the defined goal. The duration of the action is agent-dependent, defined by the *(clean_cloth_dur ?agent)* function, while the cost is increased by *(clean_cloth_cost ?agent)*. In addition to the actions achieving a goal, other actions correspond to the intermediate steps of the plan, such as moving from location to location. Our domain focuses on high-level task planning, and models high-level coordination constraints such as ensuring that no agents occupy the same location simultaneously. We separate high-level task planning from the lower-level constraints managed by motion planning systems. If the problem were to be extended to physical collaborative tasks, the necessary physical constraints would need to be integrated into our domain, which would still be compatible with our domain-independent method. Furthermore, note that the planning domain is set at a level of abstraction that leaves the mapping of domain actions into lower-level actions (e.g. grasping, placing) to the robot's executive.

Each problem file includes the instances for the agents (human and robot), objects (e.g. food), and locations, as well as the values for the costs and the durations. The goal, made of *cooked* and *cleaned* propositions for specific objects and locations respectively, is also specified in the problem. Finally, the metric to minimize is defined as the sum of the `total-cost` and `total-time` of the plan.

For our evaluation, we generated 270 test cases for this domain, with different action costs, action durations and numbers of goals.

```
(define (domain domain_kitchen_2agents)
(:requirements :durative-actions :action-costs)
(:types agent loc obj)
(:predicates
    ; other predicates
    ...
    ; goal predicates
    (cooked ?obj - obj)
    (cleaned ?loc - loc)
)

(:functions
    (clean_mop_cost ?agent - agent)
    (clean_cloth_cost ?agent - agent)

    (clean_mop_dur ?agent - agent)
    (clean_cloth_dur ?agent - agent)
    ...
    ; other functions
    ...
    (total-cost)
)

(:durative-action clean_cloth
 :parameters (?agent - agent ?loc - loc)
 :duration (= ?duration (clean_cloth_dur ?agent))
 :condition (and
    (at start (agent_not_busy ?agent))
    (at start (agent_at_loc ?agent ?loc))
    )
 :effect (and
    (at end (cleaned ?loc))
    (at start (not (agent_not_busy ?agent)))
    (at end (agent_not_busy ?agent))
    (at start (increase (total-cost)
                        (clean_cloth_cost ?agent)))
    )
)
; rest of actions
```

Figure 2. Domain snippet for our kitchen scenario.

5.2 Planners

The planners we use in this domain are:

- OPTIC-AEES: our planner as described in Section 4, with $\hat{f}$ used as a branch-ordering heuristic, and with per-d open lists to allow $\hat{f}(n)$ values to be re-computed at each expansion;
- OPTIC: the unmodified OPTIC planner [2]
- LPG-TD [17], the best performer among prior non-OPTIC planners.

Additionally, we define a baseline based on a hand-coded decomposition of the evaluation problems into two sub-problems (one for the human, one for the robot) where goals are greedily assigned to one or the other, based on which is estimated to be able to achieve the goal at lowest cost. Plans for these sub-problems are then found independently using a designated planner, and re-combined. We offer this not as a general-purpose technique for solving problems, but as a way of seeing what can be done if the planner is not able to directly reason about how to balance overall time and cost across both agents, but instead can reason about only one of them at once.

To support an ablation study, we additionally run:

- OPTIC-AEES-bf: OPTIC-AEES but with f rather than $\hat{f}$ as a branch-ordering heuristic
- OPTIC-AEES-bn: OPTIC-AEES but without a branch-ordering heuristic
- OPTIC-AEES-fixed: OPTIC-AEES but with $\hat{f}(n)$ fixed at the point n is added to the open list, rather than being re-computed at each expansion.

All planners were run for 60 seconds[4] on a i7-10850H CPU, as an analogue for an online planning setting; none exceeded the allocated 16GiB of memory. The plan found for each planner was taken to be the best found after 60 seconds. The baseline, in its favour, was run once for each of the available planners, allowing 60 seconds for each call to the planner (once per sub-problem), and keeping the best result on each problem across any of these planners.

5.3 Results

Table 1 presents the summary results for the different planning configurations over the 270 problems. To facilitate a headline comparison, we compute a score for each planner using the scoring measure of the satisficing track of the International Planning Competition [23]: the score for planner p on task i is:

$$S_i^p = \frac{C_i^\star}{C_i^p}$$

...where C_i^p is the metric value of the solution found by planner p to task i, and $C_i^\star$ is the best-seen metric value for a solution to task i across all planners. The sum of these scores across all problems is then taken as the overall score for the planner. We additionally note how many times each planner found the (joint) best solution to one of the problems, and also represent this value as a percentage.

The first observation to make in Table 1 is that the Baseline decomposition approach performs poorly: it rarely finds best solutions, and the overall quality of plans found as reflected in its IPC score is

[4] Longer planning times, while out of scope, do not change the overall relative performance of the tested configurations: the bottleneck becomes the memory limit, not the time limit.

	Baseline	OPTIC	LPG-TD	OPTIC-AEES	OPTIC-AEES-bf	OPTIC-AEES-bn	OPTIC-AEES-fixed
IPC Score Sum	193.84	235.66	203.14	**268.24**	267.03	258.68	246.48
Joint-Best Count	35	108	11	**249**	239	188	162
Joint-Best Count (%)	13%	40%	4%	**92%**	89%	70%	60%

Table 1. Result comparison for different planning configurations over the 270 test cases.

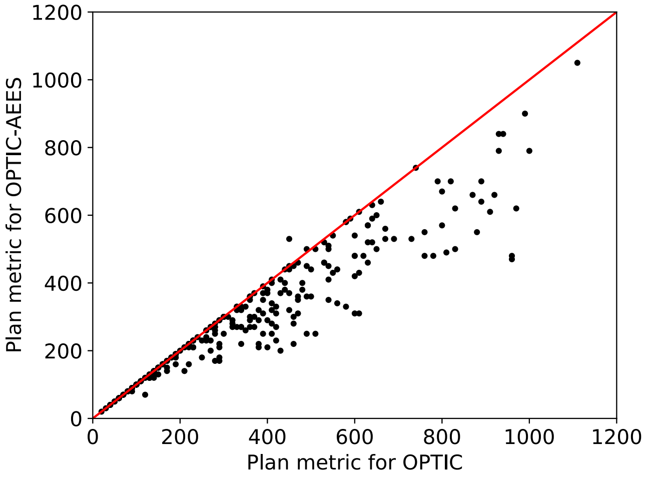

Figure 3. Comparison of plan metrics (sum of cost and makespan) for 270 test cases: OPTIC vs. OPTIC-AEES (inadmissible heuristic and inadmissible-heuristic $\hat{f}$ order insert.)

the lowest seen. This strongly motivates using *any* of the other options, i.e. a task planner that is able to manage the trade-off between action cost and makespan for both agents, to find plans with good overall metric values.

Moving on to these, we note OPTIC-AEES has a comfortable advantage in terms of plan quality as reflected in IPC score and the number of times it found the best solution. Of pre-existing planners, the strongest performer was OPTIC. Our expectation based on its general efficacy was that LPG-TD would perform relatively better than we have seen here. Scrutinizing its output, its local-search approach was sufficient to get it somewhat of the way towards a good solution; but over the course of a minute, the other systematic-search planners (OPTIC-AEES and OPTIC) were able to reduce the cost bound down to a level such that search-space pruning based on the admissible makespan estimate from the TRPG allowed them to ultimately do better.

Figure 3 compares the plan quality for OPTIC-AEES and OPTIC for all test cases. With the exception of a single outlier, the plans from OPTIC-AEES were always better or equal; and the potential for quality improvement is relatively consistent across all plan metrics.

Moving on to ablation studies, we first consider the impact of the branch-ordering heuristic – in Table 1 OPTIC-AEES orders nodes by $\hat{f}(n)$ prior to their insertion into the open list; OPTIC-AEES-bf orders nodes by $f(n)$ prior to insertion; and OPTIC-AEES-bn does not order them at all. The first two of these configurations perform relatively closely, with a slight edge for OPTIC-AEES, which is perhaps to be expected: for instance, it is $\hat{f}(best_d)$ that determines whether $best_d$ is chosen for expansion, so if there are equal-$d(n)$ children of a state s, inserting them in ascending order of $\hat{f}(n)$ will increase the likelihood that the lower-$\hat{f}$ of the two becomes $best_d$, and is chosen for expansion; rather than the higher-$\hat{f}$ of the two becoming $best_d$, and *not* being chosen for expansion. Regardless, OPTIC-AEES-bn performs much worse, especially in terms of the number of joint-best solutions found, so we can say ordering by a cost measure (be it $\hat{f}(n)$ or $f(n)$) is much better than not doing.

Second, we consider the impact of OPTIC-AEES allowing $\hat{f}(n)$ values to be recalculated at each expansion, as the global heuristic error model is updated; compared to OPTIC-AEES-fixed, where $\hat{f}(n)$ is instead fixed at the point n is inserted into the open list. As can be seen in Table 1, OPTIC-AEES-fixed performs dramatically worse. This confirms that whatever bias is being 'baked in' to the $\hat{f}(n)$ values if they are fixed at the point of insertion, is detrimental to the overall performance of an AEES-based approach. Looking at how the global heuristic error estimates change over search, the mean one-step-errors increased in value over the first few hundred expansions. Hence, children inserted into the open list relatively early in search (with higher $d(n)$ values, towards the top of the subtree) had relatively lower $\hat{f}(n)$ values. This led to a backlog of low $\hat{f}$ states accumulating in the open list, suppressing the value of $\hat{f}(best_{\hat{f}})$, hence dramatically reducing the size of the focal list used to find $best_d$ – as states with lower $d(n)$ values were more likely found later in search, so had higher $\hat{f}(n)$ values based on higher mean one-step errors. Clearing this backlog required a greater proportion of the expansions to be of $best_{\hat{f}}$, rather than $best_d$, reducing the frequency with which new best solutions were found.

6 Conclusions

This paper motivates modeling Human-Robot Collaboration scenarios as temporal PDDL problems, with heterogeneous costs and durations for the actions of humans and robots, which compete in terms of plan metric optimization. Such problems are challenging for off-the-shelf temporal PDDL planners, given plan optimization unavoidably needs to manage a trade-off between costs and makespan. To this end, we further developed planning techniques for such domains, with a novel inadmissible makespan measure (derived from relaxed plans) used in conjunction with a novel formulation of the AEES algorithm that facilitates recomputing nodes' $\hat{f}$ values at each expansion, in line with a global heuristic error correction model.

The results obtained show that our planner represents the new state-of-the-art on a benchmark HRC problem, finding plans of much better quality than both prior temporal planners. This represents a significant step towards broadening the use of PDDL planning in Human–Robot Collaboration and other tasks where plan quality reflects a genuine and necessary trade-off between costs and durations, which we note are otherwise un-represented in the International Planning Competition series. In future work, we will explore other ways of constructing heuristic error correction models that capture the quirks of quality metrics that refer to makespan (where there are large g-value plateaux if actions that are applied are able to be scheduled alongside those already in the plan); and will model and evaluate additional multi-agent scenarios, beyond purely HRC, to broaden the impact of our techniques, leveraging the domain-independent nature of PDDL planning with our developed variant of OPTIC.

Acknowledgments

This work was supported by the CHIST-ERA project COHERENT (EPSRC EP/V062506/1). S. Izquierdo is a fellow of Eurecat's *Vicente López* PhD grant programme. G. Canal has been supported by the Royal Academy of Engineering and the Office of the Chief Science Adviser for National Security under the UK Intelligence Community Postdoctoral Research Fellowship programme.

References

[1] R. Alami, A. Clodic, V. Montreuil, E. A. Sisbot, and R. Chatila. Toward human-aware robot task planning. In *AAAI spring symposium: to boldly go where no human-robot team has gone before*, pages 39–46, 2006.

[2] J. Benton, A. J. Coles, and A. Coles. Temporal planning with preferences and time-dependent continuous costs. In *Proceedings of the International Conference on Automated Planning and Scheduling ICAPS*, 2012.

[3] S. Bernardini, F. Fagnani, and D. E. Smith. Extracting mutual exclusion invariants from lifted temporal planning domains. *Artificial Intelligence*, 2018.

[4] S.-O. Bezrucav and B. Corves. Improved ai planning for cooperating teams of humans and robots. In *Proceedings of the Planning and Robotics (PlanRob) Workshop—ICAPS*, 2020.

[5] G. Buisan, G. Sarthou, and R. Alami. Human aware task planning using verbal communication feasibility and costs. In *International Conference on Social Robotics*, pages 554–565. Springer, 2020.

[6] G. Canal, G. Alenyà, and C. Torras. Adapting robot task planning to user preferences: an assistive shoe dressing example. *Autonomous Robots*, 43(6):1343–1356, 8 2019. ISSN 1573-7527. doi: 10.1007/s10514-018-9737-2.

[7] M. Cashmore, M. Fox, D. Long, D. Magazzeni, A. Carrera, N. Palomeras, N. Hurtos, and M. Carreras. ROSPlan: Planning in the Robot Operating System. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2015.

[8] T. Chakraborti, G. Briggs, K. Talamadupula, Y. Zhang, M. Scheutz, D. Smith, and S. Kambhampati. Planning for serendipity. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5300–5306. IEEE, 2015.

[9] F. Chen, K. Sekiyama, H. Sasaki, J. Huang, B. Sun, and T. Fukuda. Assembly strategy modeling and selection for human and robot coordinated cell assembly. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4670–4675. IEEE, 2011.

[10] Y. Cheng, L. Sun, and M. Tomizuka. Human-aware robot task planning based on a hierarchical task model. *IEEE Robotics and Automation Letters*, 6(2):1136–1143, 2021.

[11] W. Cushing, S. Kambhampati, Mausam, and D. Weld. When is temporal planning *really* temporal planning? In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[12] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49, 1991.

[13] M. B. Do and S. Kambhampati. Sapa: Multi-objective Heuristic Metric Temporal Planner. *Journal of Artificial Intelligence Research*, 20:155–194, 2003.

[14] P. Eyerich, R. Mattmüller, and G. Röger. Using the context-enhanced additive heuristic for temporal and numeric planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2009.

[15] M. Faroni, A. Umbrico, M. Beschi, A. Orlandini, A. Cesta, and N. Pedrocchi. Optimal Task and Motion Planning and Execution for Multiagent Systems in Dynamic Environments. *IEEE Transactions on Cybernetics*, 2023.

[16] M. Fox and D. Long. PDDL2.1: An Extension of PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

[17] A. Gerevini, A. Saetti, and I. Serina. An approach to temporal planning and scheduling in domains with predicatable exogenous events. *Journal of Artificial Intelligence Research (JAIR)*, 2006.

[18] A. Ham and M.-J. Park. Human–robot task allocation and scheduling: Boeing 777 case study. *IEEE Robotics and Automation Letters*, 2021.

[19] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[20] S. Izquierdo-Badiola, G. Canal, C. Rizzo, and G. Alenyà. Improved Task Planning through Failure Anticipation in Human-Robot Collaboration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 7875–7880, 5 2022. doi: 10.1109/ICRA46639.2022.9812236.

[21] M. Lippi and A. Marino. A Mixed-Integer Linear Programming Formulation for Human Multi-Robot Task Allocation. In *Proceedings of the IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, 2021.

[22] D. Long and M. Fox. Exploiting a Graphplan Framework in Temporal Planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2003.

[23] C. L. López, S. J. Celorrio, and Á. G. Olaya. The deterministic part of the seventh international planning competition. *Artificial Intelligence*, 223:82–119, 2015.

[24] J. T. Thayer, A. Dionne, and W. Ruml. Learning inadmissible heuristics during search. In *Proceedings of the International Conference on Automated Planning and Scheduling ICAPS*, 2011.

[25] J. T. Thayer, J. Benton, and M. Helmert. Better parameter-free anytime search by minimizing time between solutions. In *Proceedings of the Symposium on Combinatorial Search (SOCS)*, 2012.