

# Planning and Acting While the Clock Ticks: Supplementary Material

**Primary Keywords:** (6) *Temporal Planning*

This document contains pseudo-code and ablation studies that were not able to be included in the main paper due to space constraints.

## 1 Pseudo-code

- 5 This appendix includes the pseudo-code for solving the special cases of CoPEM, described in Section 4. Algorithm 1 solves the case of  $K = 0$  and  $L = 1$ , whereas Algorithm 2 solves the case of  $K = L = 1$ .

---

Algorithm 1: Solving  $\mathcal{I}_i$  ( $K = 0, L = 1$ )

---

```

1: function Solve01( $\mathcal{I}_i$ )
2:    $\text{Max}P \leftarrow 0$ 
3:   for  $j = 1$  to  $n$  do
4:      $P_j \leftarrow 0$ 
5:     for  $o \in \text{support}(m_j)$  do
6:        $(P, \pi_{ij}) \leftarrow \text{Solve00}(\mathcal{I}_i^0(c_j, o))$ 
7:        $P_j \leftarrow P_j + P \cdot m_j(o)$ 
8:     end for
9:     if  $\text{Max}P < P_j$  then
10:       $\text{Max}P \leftarrow P_j$ 
11:       $\text{best\_}c_j \leftarrow c_j$ 
12:     end if
13:   end for
14:   return  $\text{Max}P, \text{best\_}c_j$ 

```

---

## 2 Ablation Studies

- 10 We performed three ablation studies. In the first one, we set the minimum number of expanded nodes in the subtree and in *sim.open* to 1, thereby allowing dispatching of actions based on uncertain estimates. The results for these are shown in Figure 1. As the results show, *nodisp* performs better than *disp* with the different dispatch thresholds in RCLL for higher CPU speeds, showing that this VOI criterion is important.

- 15 In the second ablation, we set the subtree focus threshold to 1, thereby eliminating the option to focus search on promising candidates which have not been explored enough. The results for these are in Figure 2. While *disp* outperforms *nodisp* in most cases, in RCLL with 2 robots *nodisp* is better

---

Algorithm 2: Solving  $\mathcal{I}$  ( $K = 1, L = 1$ )

---

```

1:  $\text{Max}P \leftarrow 0$ 
2: for  $j = 1$  to  $n$  do // No dispatch at  $t = 0$ 
3:    $P_j \leftarrow 0$ 
4:   for  $o \in \text{support}(m_j)$  do
5:      $(P, \pi_j) \leftarrow \text{Solve00}(\mathcal{I}(c_j, o))$ 
6:      $P_j \leftarrow P_j + P \cdot m_j(o)$ 
7:   end for
8:   if  $\text{Max}P < P_j$  then
9:      $\text{Max}P \leftarrow P_j$ 
10:     $\text{policy} \leftarrow [c_j]$ 
11:   end if
12: end for
13: for  $i = 1$  to  $n$  do // Dispatch  $a_i \in H_i$  at  $t = 0$ 
14:    $(P'_i, \text{best\_}c_j) \leftarrow \text{Solve01}(\mathcal{I}_i)$ 
15:   if  $\text{Max}P < P'_i$  then
16:      $\text{Max}P \leftarrow P'_i$ 
17:      $\text{policy} \leftarrow [a_i, \text{best\_}c_j]$ 
18:   end if
19: end for
20: return  $\text{Max}P, \text{policy}$ 

```

---

with higher CPU speeds. This is likely because our dispatching planner commits to a wrong decision, as the promising nodes have not been explored enough.

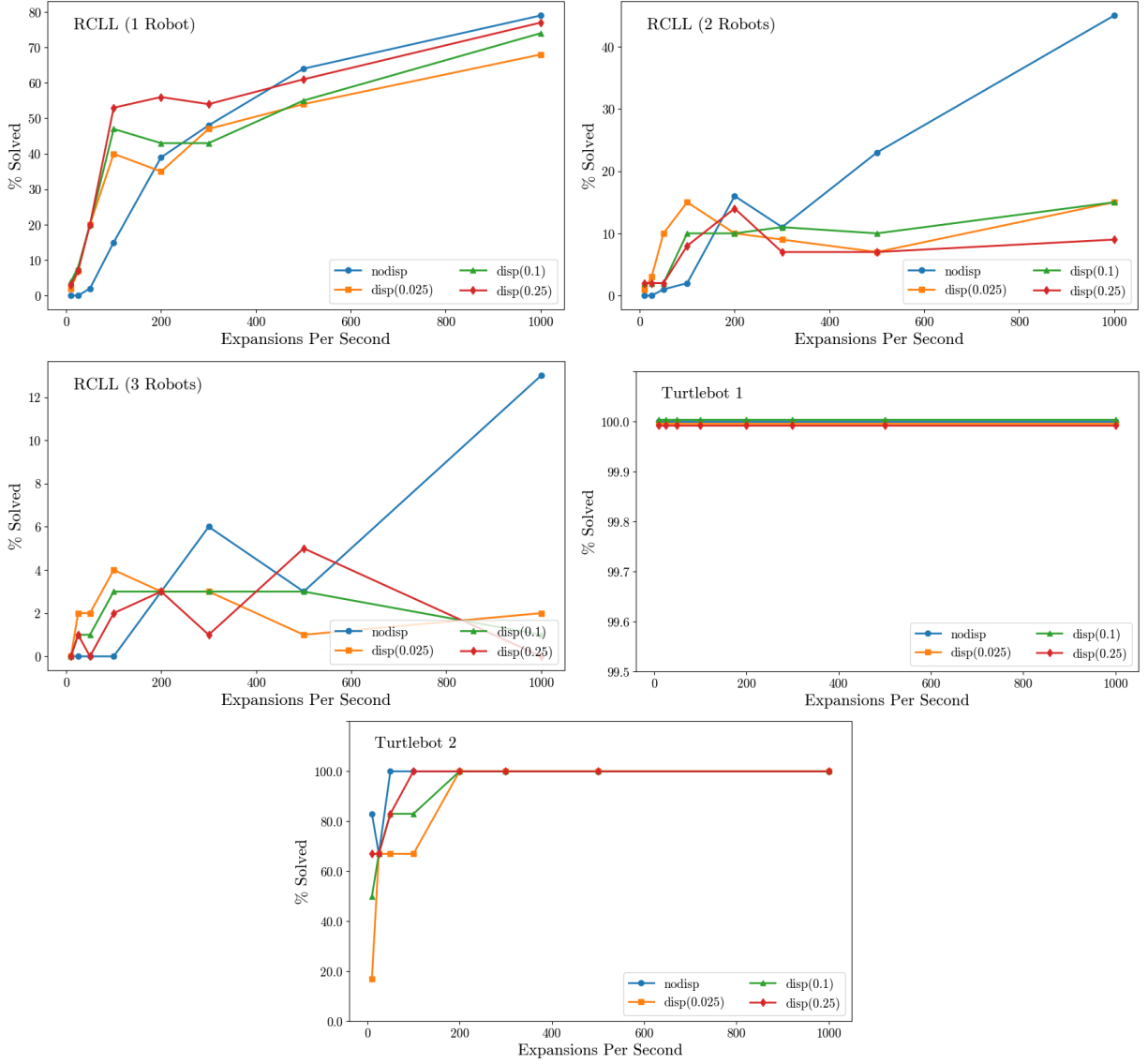


Figure 1: Ablation 1: Dispatch frontier size = 1

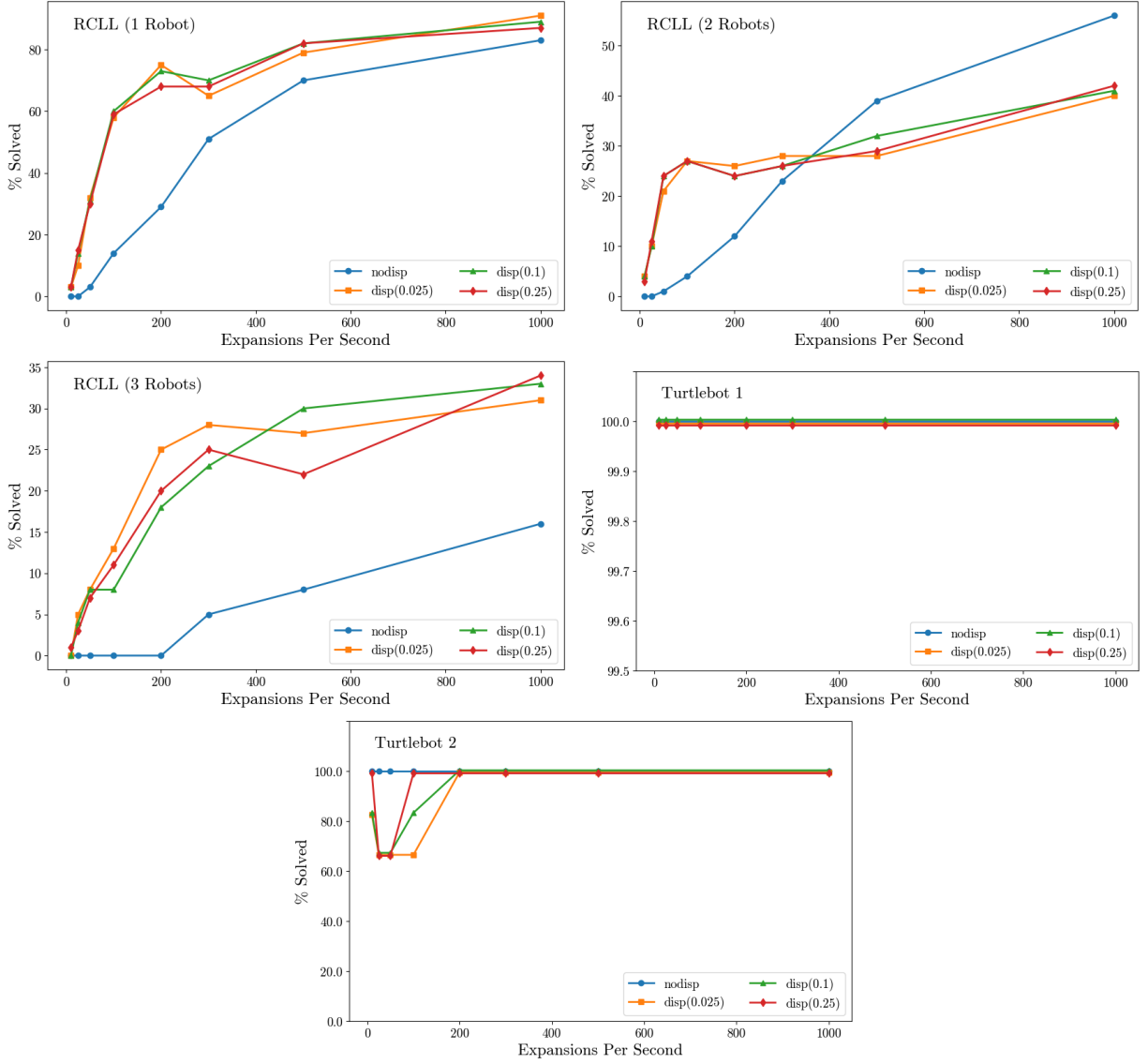


Figure 2: Ablation 2: Subtree focus threshold = 1