Multi-Horizon Multi-Agent Planning Using Decentralised Monte Carlo Tree Search

Konstantin M. Seiler, Felix H. Kong, Robert Fitch

***Abstract*—We propose multi-horizon Monte Carlo tree search (MH-MCTS), the first framework for integrated hierarchical multi-horizon, multi-agent planning based on Monte Carlo tree search (MCTS). The method employs multiple simultaneous MCTS optimisations for each planning level within each agent, which are designed to optimise a joint objective function. Using concepts from decentralised Monte Carlo tree search (Dec-MCTS), the individual optimisations continuously exchange information about their current plans. This breaks the common top-down only information flow within the planning hierarchy and allows higher level optimisers to consider progress made by lower level planners. The method is implemented for survey missions using a fleet of ground robots. Simulation results with different mission profiles show substantial performance improvements of the new method of up to 59% compared to traditional MCTS and Dec-MCTS.**

***Index Terms*—Multi-Robot Systems, Path Planning for Multiple Mobile Robots or Agents**

I. Introduction

IN RECENT years, the Monte Carlo tree search (MCTS) algorithm has gained increasing popularity as a planning method for a wide range of applications such as gameplay [1], [2], scheduling [3], [4], robot path planning [5] and multi-agent coordination [6], [7] due to its ability to create any-time plans using generative black-box models. However, for practical reasons the time horizon considered for planning is often kept well below the actual mission time to limit the depth of the search tree, reduce the noise of the rollout heuristic and allow MCTS to converge to a good solution within its computational budget. Limiting the time horizon leads to myopic behaviour, which is unfortunate because a strength of MCTS in finding global (non-myopic) optima. The vision of this paper is to overcome computationally-induced myopic behaviour by allowing MCTS planners to consider the entire mission time span while remaining computationally tractable.

The need to limit the planning horizon due to computational cost is not unique to MCTS and is a common issue in optimisation. This is often solved by employing hierarchical planning where a higher level planner creates plans with a longer planning horizon and reduced resolution while a lower level planner creates shorter plans of higher fidelity that follow the set points given by the higher level plan [4], [8]. In most cases this leads to a top-down approach where plans are sent from higher planners to lower planners as targets or constraints but no information about possible solutions found by the lower planner flows back up.

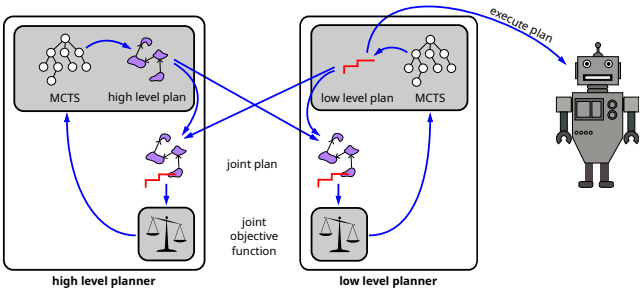


Fig. 1: Overview of the MH-MCTS algorithm. The high level planner and low level planner use the Dec-MCTS framework to jointly optimise an objective function. They continuously exchange intents to derive plans that are well aligned with each other.

The lack of information flow from the lower to the higher level planner is a limitation of such approaches. The low level planner is bound by the high level plan and subsequently may have to make decisions that are globally suboptimal or may even be faced with an infeasible problem. Instead it would be desirable for the planners to cooperatively perform a joint optimisation.

Decentralised Monte Carlo tree search (Dec-MCTS) [6] is a framework for collaborative planning across multiple agents which we can leverage here for cooperative planning across different planning levels. Dec-MCTS is an extension to MCTS where multiple agents perform their own MCTS optimisation while continuously exchanging their most likely solutions with their peers to allow for a joint optimisation.

This paper presents multi-horizon Monte Carlo tree search (MH-MCTS), the first hierarchical planning framework that allows multiple MCTS optimisations to simultaneously run at different planning horizons and resolutions while continuously exchanging information about their planning progress. MH-MCTS is built upon concepts from Dec-MCTS and thus not only allows vertical cooperation between high and low level planners but also maintains the ability for horizontal cooperation between multiple agents. Thus, MH-MCTS provides the first multi-horizon, multi-agent any-time planning solution based on MCTS. Furthermore the method is asynchronous which allows individual planners to operate at different planning intervals. This enables higher level planners to perform longer planning cycles to increase the overall solution quality. The concept is illustrated in Fig. 1. Our experiments with an implementation of MH-MCTS for survey missions with a fleet of ground vehicles in simulation show a reduced median

Manuscript received: February, 8, 2024 ; Revised May, 14, 2024; Accepted July, 2, 2024 .

This paper was recommended for publication by Editor Giuseppe Loianno upon evaluation of the Associate Editor and Reviewers' comments.

Authors are with the School of Mechanical and Mechatronic Engineering, University of Technology Sydney, Australia. Emails: konstantin.seiler@uts.edu.au; felix.kong@uts.edu.au; robert.fitch@uts.edu.au

Digital Object Identifier (DOI): 10.1109/LRA.2024.3426273

mission time of up to 59% compared to ordinary MCTS and Dec-MCTS.

The remainder of this paper is organised as follows: related work is discussed in Section II. The problem statement is formalised in Section III. Section IV recalls the workings of MCTS and Dec-MCTS before Section V describes MH-MCTS. Section VI details a concrete implementation of MH-MCTS for a survey mission and performance results are presented in Section VII. Section VIII concludes.

II. Related Work

MCTS and its many variations [9] optimises a plan by building a tree over possible action sequences. Each decision point forms a tree node and branches explore the outcomes of different actions. The tree is built incrementally in depth and width with most of the search being focused on branches where high reward values have been observed previously. The value of leaf nodes is estimated by performing a rollout where actions are chosen according to a randomised heuristic until a final state or the planning horizon is reached and the reward can be calculated. Because the tree is build incrementally from the root, MCTS is well suited to receding horizon control (RHC) where only the beginning of a plan is scheduled for execution before planning is restarted anew, potentially with an extended planning horizon.

Best *et al.* [6] present Dec-MCTS, a framework which allows multiple agents to optimise a joint objective function in a decentralised manner with limited communication. The core idea is that each agent runs MCTS over its own action space and regularly broadcasts an 'intent', i.e. a probability distribution over its best solutions to its teammates. This in turn allows each agent to calculate a relative reward where the value of the own plan is calculated based on the likely actions the teammates are going to perform. Intent transmission happens asynchronously and thus the method is forgiving of communication lag and drop outs.

There exists a body of work about hierarchical MCTS [10]–[12] using meta actions that combine multiple primitive actions to intermediate goals. The approach builds a single MCTS tree in full depth and detail. However, nodes of the tree that are within meta actions use restricted action spaces and modified reward functions which guide the search towards intermediate goals. This contrasts the approach taken here where separate MCTS trees are used for different planning levels which may not share a common planning horizon.

Multi-level planning based on RHC is a common approach in control theory. The prevalent approach is to employ multiple hierarchical planners with different horizons and planning frequencies with a mostly top-down information flow. However, for higher level set points to remain feasible, limited information about the system's performance can also flow bottom-up. For further details see Scattolini [13] and references therein.

Shao *et al.* [14] use MCTS as part of method to solve hierarchical task networks (HTNs). The solver is based on the SHOP algorithm [15], the performance of which is sensitive to the chosen order of task decomposition which is domain specific. Instead of hand-crafting the decomposition order based on domain specific knowledge, MCTS is employed to provide estimates about the decomposition order online. Thus the two planners work together on the optimisation problem, effectively guiding each other instead of one constraining the other. This contrasts, for example, the approach taken earlier by Neufeld *et al.* [16] where the HTN solver 'adversarial planning with HTNs' (AHTN) [17] was used together with MCTS in classical constraining hierarchical planning.

III. Problem Statement

We consider a team of $N \geq 1$ robots or agents $\{R_1, ...R_N\}$ where each robot R_i plans its own action sequence $\mathbf{a}^i = (a^i_1, a^i_2, ...)$. The a^i_j are elements of a finite action space $\mathbf{A}$. A robot's state is described by an element of the state space $\mathbf{S}$ and the initial state of robot R_i is given as $s^i_0 \in \mathbf{S}$. The system dynamics are described by a transition function $T : \mathbf{S} \times \mathbf{A} \to \mathbf{S}$ such that $s^i_j = T(s^i_{j-1}, a^i_j)$ which gives rise to a state sequence $\mathbf{s^i} = (s^i_0, s^i_1, ...) = T(s^i_0, \mathbf{a}^i)$. The robots aren't necessarily homogeneous and as such the robots have potentially different state and action spaces $\mathbf{S}$ and $\mathbf{A}$ and transition functions T. Furthermore, the set of admissible actions may be state dependent, effectively varying $\mathbf{A}$. For ease of notation these variations are omitted in this presentation but it is understood that the sets and transition function may change depending on context. The team's performance is measured by a global objective function $g : \mathbf{S}^{\mathbb{N}\times N} \to \mathbb{R}$ which assigns a reward value to the robots' joint state sequences. Thus, the optimisation problem can be stated as

$$\underset{(\mathbf{a}^1,...,\mathbf{a}^N)\in\mathbf{A}^{\mathbb{N}\times N}}{\arg\max} g\left(T(s^1_0, \mathbf{a}^1), ..., T(s^N_0, \mathbf{a}^N)\right) . \qquad (1)$$

The team of robots are assumed to be within communication range with each other which allows them to regularly exchange updates about their planning progress to aid the joint maximisation of Eq. (1).

Note that the state and action sequences as written here are infinite sequences. However, in practise they are often finite due to reaching a terminal state. In cases of truly open ended problems, RHC is employed to turn each individual planning instance into a finite problem.

IV. Preliminaries

A. Monte Carlo tree search

MCTS allows to solve Eq. (1) for the case of a single robot ($N = 1$). See Browne *et al.* [9] for a detailed description of the algorithm and its history. The algorithm builds a tree where each vertex holds a state $s \in \mathbf{S}$ and each edge corresponds to an action $a \in \mathbf{A}$. At first, the root node is created with the initial state s_0. Then, during each iteration, a vertex s is selected which still has an open action a that does not correspond to any of its outgoing edges. A new outgoing edge with a is then added to s the with a new target vertex of $s' = T(a, s)$. Next the value of the new vertex s' is estimated by completing the state and action sequence until a terminal state is reached and evaluating the reward function g. Finally, statistics about s' and all its parent vertices are updated such that each vertex holds information about the

number of episodes that pass through it as well as the sum of their reward.

The algorithm forms an optimisation method due to the way the vertex s to extend is selected in the first step. The selection process starts at the root s_0. If the current vertex s has no open actions, all outgoing edges leading to a state s' are scored according to the upper confidence bound for trees (UCT) formula

$$\mathrm{UCT}(s, s') = \bar{r}_{s'} + c\sqrt{\frac{\ln n_s}{n_{s'}}} \tag{2}$$

where $\bar{r}_{s'}$ is the average reward of all episodes that passed through s' and n_s and $n_{s'}$ are the number of episodes passing through s and s' respectively. c is a tuning constant which balances exploitation against exploration. The process then continues with the child vertex that has the highest UCT score. Equation (2) ensures that the search focuses on branches of the tree where high reward values have been observed before while also exploring lower valued branches that haven't been visited often.

The tree growing algorithm is detailed in Algorithm 1. The final solution is extracted by traversing the tree one last time for pure exploitation by setting $c = 0$ in Eq. (2) and returning the state sequence $\mathbf{s}$ built by Algorithm 1.

MCTS is effectively a single-agent algorithm and multi-agent planning is only possible using centralised planning, treating the entire team as a single system and sequentially choosing actions for the next robot that requires a decision. This approach was taken in [4] for example.

B. Decentralised Monte Carlo tree search

Dec-MCTS [6] adds a decentralised framework to MCTS which allows for multi-agent planning. Each agent in a team builds its own MCTS tree and regularly transmits a set of state sequences, called *intents*. The original presentation [6] transmits a probability distribution over multiple intents. However, the implementation can be simplified substantially by only transmitting the current MCTS solution as a single intent, an approach taken in this work.

Algorithm 1 GROW_TREE(T)

1: $s \leftarrow T$.root
2: $\mathbf{s} \leftarrow (s)$
3: **while** $\forall a \in \mathbf{A} : \exists$ out edge of s with action a **do**
4: $\quad s \leftarrow T(s, \arg\max_{a \in \mathbf{A}} \mathrm{UCT}(s, T(s, a)))$
5: $\quad$ Append s to $\mathbf{s}$.
6: **if** s is not terminal state **then**
7: $\quad$ Select open action $a \in \mathbf{A}$ and create a new edge a to new child vertex $T(s, a)$.
8: $\quad s \leftarrow T(s, a)$
9: $\quad$ Append s to $\mathbf{s}$.
10: **while** $\mathbf{s}$ does not end with a terminal state **do**
11: $\quad$ Append state to $\mathbf{s}$ according to rollout heuristic.
12: $r \leftarrow g(\mathbf{s})$
13: update statistics on s and all its parents.

The teammates' intents allow an agent to evaluate the common reward function g in the multi-agent setting. The reward evaluation in Line 12 of Algorithm 1 is replaced by a relative reward compared to a *null-policy*. The null-policy is an application dependent default state sequence $\hat{\mathbf{s}}$ that the robot could execute in the absence of a plan, e.g., it could stay stationary. Line 12 is thus replaced by

$$r \leftarrow g(\mathbf{s}^1, ..., \mathbf{s}^N) - g(\mathbf{s}^1, ..., \mathbf{s}^{i-1}, \hat{\mathbf{s}}^i, \mathbf{s}^{i+1}, ..., \mathbf{s}^N) \tag{3}$$

where i is the index of the robot evaluating the reward.

Updated team mates' intents effectively lead to a drifting reward function such that the same state sequence $\mathbf{s}$ contained in the tree yields different rewards at different times of the planning cycle. Thus, the UCT formula is replaced by discounted UCT (D-UCT) which discounts older rewards in favour of more recently observed rewards. To calculate the discounting let $\mathbf{1}_s(t)$ be the indicator function which is 1 if vertex s was selected in tree expansion round t and 0 otherwise. Then a discounted visitation count $C_t(s)$ for vertex s with discount factor $\gamma \in (0, 1]$ is defined as

$$C_t(s) = \sum_{u=1}^{t} \gamma^{t-u} \mathbf{1}_s(u) \; . \tag{4}$$

Let r_t denote the reward that was received in iteration t (Line 12 of Algorithm 1), then D-UCT is defined as

$$\text{D-UCT}(s, s') = \frac{\sum_{u=1}^{t} \gamma^{t-u} r_u \mathbf{1}_{s'}(u)}{C_t(s)} + c\sqrt{\frac{\ln C_t(s')}{C_t(s)}} \; . \tag{5}$$

V. Multi-horizon Monte Carlo tree search

We can now describe MH-MCTS in general before we present a concrete implementation in Section VI. Observe that the Dec-MCTS algorithm doesn't require the team of robots to be homogeneous. States, actions and transition functions can be entirely disjoint and the only requirement is the ability to evaluate a common objective function g. Thus, it is possible to deploy multiple MCTS planners per robot which operate on different planning horizons and combine them into a joint optimisation using the Dec-MCTS framework.

Little would be gained if a higher level planner merely extended the planning horizon but was otherwise identical to the low level planner. The high level planner would then suffer from the very same computational complexity that prevented using a single planner in the first place. Thus, the state and action space of higher level planners must be designed to have a coarser granularity than the desired output of the low level planner. For example, when performing path planning, the low level planners could reason over individual waypoints to visit whereas higher level planners reason over goals to select or general regions to visit.

The joint objective function is designed such that a higher reward is received if low and high level plans are aligned and support each other. For example if the mission is to visit a certain set of goals, the reward function could be an estimate of the time needed to complete the mission, taking into account the path planned by the low level planner and completion strategy devised by the high level planner. Thus, the low level planner is incentivised to produce a plan which aligns well with the high level plan and vice versa.

It is possible to design a reward function that can be decomposed into a sum of one part that is only dependent on the low level intent and a second part which depends on all intents. E.g., for N robots and two planning levels each we get:

$$g(\mathbf{s}^1, ..., \mathbf{s}^{2N}) = g_1(\mathbf{s}^1, \mathbf{s}^3, ..., \mathbf{s}^{2N-1}) + g_2(\mathbf{s}^1, \mathbf{s}^2, ..., \mathbf{s}^{2N}) \ . \tag{6}$$

Here we use the convention the planners with odd indices (1, 3, 5, ...) are low level planners whereas even indices (2, 4, 6, ...) refer to high level planners. Remember that different planners can have different action and state spaces $\mathbf{A}$ and $\mathbf{S}$ so the $\mathbf{s}^i$ are usually from different spaces.

Decomposed this way, g_1 expresses an immediate reward received for low level actions (e.g., goal points reached, information gained) whereas g_2 reflects the long term performance (e.g., time to mission completion). It also provides computational benefits. Because for a higher level planner (even index), g_1 cancels out in Eq. (3) and the reward calculation simplifies to

$$\begin{aligned} r &\leftarrow g(\mathbf{s}^1, ..., \mathbf{s}^N) - g(\mathbf{s}^1, ..., \mathbf{s}^{i-1}, \hat{\mathbf{s}}^i, \mathbf{s}^{i+1}, ..., \mathbf{s}^N) \\ &= g_2(\mathbf{s}^1, ..., \mathbf{s}^N) - g_2(\mathbf{s}^1, ..., \mathbf{s}^{i-1}, \hat{\mathbf{s}}^i, \mathbf{s}^{i+1}, ..., \mathbf{s}^N) \ . \end{aligned} \tag{7}$$

Thus, decomposing the reward function reduces the computational burden on higher level planners.

A further observation is that the Dec-MCTS framework doesn't require the agents to synchronise their planning cycles such that all planning cycles in an RHC setting start and end at the same time. As long as the objective function stays consistent across planning cycles, it is possible that one agent completes planning and starts a new cycle while others continue with their current MCTS search. Due to the discounting used in D-UCT, the algorithm can even tolerate moderate inconsistencies in the objective function as new planning cycles are commenced asynchronously.

This is beneficial in a multi-horizon setting because it allows to run higher level planners on longer planning cycles which leads to longer running MCTS trees and subsequently a higher chance to converge to a good solution. Keeping the high level planner running over multiple planning cycles also introduces a hysteresis effect where the low level planner is biased towards solutions that are aligned with the same high level plan as the solution of the previous cycle.

In MH-MCTS, the high level planners only start a new planning cycle if the problem definition ($\mathbf{A}$, $\mathbf{S}$, s_0, T, g) changes in an incompatible way and the MCTS tree must be invalidated. In the example of a moving robot this could be when goal or search areas have been completed or otherwise change.

A. Theoretical analysis

From an algorithmic point of view, MH-MCTS is a direct application of Dec-MCTS, albeit an unusual one. As such, the in-depth theoretical analysis from Best *et al.* [6] carries over to the work presented here. In particular, Eq. (5) allows individual trees to converge towards their locally optimal solution in probability as the number of iterations grow large, provided that γ is set to a value consistent with the expected changes (breaks) in the reward function as the tree grows.

The theoretical shortcoming that is shared between Dec-MCTS and MH-MCTS is that convergence to global joint optimality cannot be guaranteed. However, as addressed in [6], this is a limitation shared by solution algorithms for decentralised, long-horizon planning problems when general objective functions that lack simplifying properties such as submodularity are considered.

VI. A Concrete Implementation

Here we detail a concrete implementation of MH-MCTS for a survey mission using a fleet of ground vehicles. The scenario was deliberately kept simple to solely investigate the benefits MH-MCTS can provide compared to ordinary MCTS or Dec-MCTS. Two separate scenarios are tested. In the first scenario, a set of points of interest or targets must be observed by the robots whereas the second asks for comprehensive coverage of a search area. The mission objective is to observe all targets in as little time as possible.

To make the robots' continuous state space tractable for planning with MH-MCTS, a discrete road map in in the form of a directed graph is defined. The examples here use a probabilistic road map (PRM) [18] based on Dubins paths [19] and a regular grid pattern respectively. More details about this setup are given in Section VII.

MH-MCTS is implemented with two planning levels per robot. The low level planner reasons over paths on the road map and as such the state space $\mathbf{S}$ contains the vertices of the road map and the action space $\mathbf{A}$ equals the outgoing edges at each vertex. The transition function T advances the state along the edges in the obvious way and a state is considered terminal if the path length has reached a threshold given by the planning horizon. The low level reward function g_1 simply counts the number of previously unseen targets that become observed by the team of vehicles. The null-policy for calculating the relative reward assumes the vehicle stays stationary.

The high level planner is designed to reason over targets or clusters of targets that have not been observed yet. A vertex of the road map graph is considered *open* if from its position a target can be observed. The implementation used here groups targets that are close by to each other into clusters. This allows efficient handling of groups of targets while isolated targets will still be treated individually. Clusters are formed by incrementally adding vertices that are less than a threshold away from existing members of a cluster.

The action space $\mathbf{A}$ of the high level planner is the set of goal areas with the interpretation that taking action $a \in \mathbf{A}$ results in the vehicle visiting open vertices of a until none of them remain open. The state space $\mathbf{S}$ is the power set of $\mathbf{A}$ with the interpretation of goal areas that must still be completed. Thus, the initial state is $s_0 = \mathbf{A} \in \mathbf{S}$, the only terminal state is the empty set, and the transition function is defined as $T(s, a) = s \setminus \{a\}$.

The high level reward function g_2 estimates the time needed by the vehicles to complete the mission. This estimate is calculated as follows.

First each vehicle is assumed to complete the path given by the low level intent. The map is updated according to the observed targets and the completion time and position are noted for each robot. Then, as long as there are still open vertices, the robot with the earliest completion time is selected to move to an open vertex. To choose the vertex to move to, the first goal area of the robot's long horizon intent that still has open vertices is taken and a random open vertex from it is chosen. The targets that can be seen from the vertex are marked as observed and the estimated time to move to the vertex is added to the robot's completion time.

The time it takes to reach the vertex is estimated in one of three ways depending on the situation: 1) if the robot just completed its low level intent, the length of the shortest path from the robot's position to any vertex in the goal area the vertex is chosen from is used; 2) if the robot just visited a vertex in the same target cluster, then a nominal distance to move between vertices within the cluster is used; 3) if the robot last visited a vertex in another cluster, the shortest path connecting the two clusters is used. To calculate the nominal distance within a cluster for case 2, the average edge length of a minimal spanning tree between the cluster's targets is used. The spanning tree can be computed for instance using Kruskal's algorithm [20].

Using this distance function is an optimistic underestimate which treats all vertices within a goal region equal and assumes that the robot can find an optimal path where it only ever moves from one open vertex to an open neighbour.

The mean of all robots' completion times is returned as the result of g_2. Pseudocode for g_2 is given in Algorithm 2.

Algorithm 2 $g_2(\mathbf{s}^1, \mathbf{s}^2, ..., \mathbf{s}^{2N})$

$V \leftarrow$ open vertices
for $i \leftarrow 1$ **to** N **do**
  $V \leftarrow V \setminus \{\text{vertices visited by } \mathbf{s}^{2i-1}\}$
  $t_i \leftarrow$ length of $\mathbf{s}^{2i-1}$
  $P_i \leftarrow \{\text{end position of } \mathbf{s}^{2i-1}\}$
$I \leftarrow \{1, \ldots, N\}$
while $I \neq \emptyset$ **do**
  $i \leftarrow \arg\min_{i \in I} t_i$
  if $\exists a \in \mathbf{s}^{2i} : a \cap V \neq \emptyset$ **then**
    $R \leftarrow$ first $a \in \mathbf{s}^{2i}$ with $a \cap V \neq \emptyset$
    if $R \cap P_i \neq \emptyset$ **then**
      $t_i \leftarrow t_i +$ nominal distance between vertices
    else
      $t_i \leftarrow t_i +$ shortest path distance from P_i to R
      $P_i \leftarrow R$
    $V \leftarrow V \setminus \{\text{random element from } R \cap V\}$
  else
    $I \leftarrow I \setminus \{i\}$
return $\frac{1}{N} \sum_{i=1}^{N} t_i$

During development we also attempted to use a more accurate distance function by selecting vertices in a greedy way and mapping the path exactly. However, this approach failed due to too much deterministic noise from the heuristic whereas the method detailed above yields good results.

For both planners a random rollout heuristic is used. The low level planner selects paths along the road map until the planning horizon is reached and the long horizon planner selects goal areas until the state is empty. To reduce rollout noise during reward calculation, the rollout section of the low level intent is ignored when calculating g_2, only the part of the intent that is actually in the MCTS tree is considered. g_1 uses the full intents including the rollout component.

VII. Results

For experiments, different scenarios with a survey mission were tested. Common to the scenarios is that the robots move at a constant speed and must observe (be within sensor range of) all targets that are present in the search area.

The purpose of the experiments presented here is to evaluate the advantage MH-MCTS can provide over plain MCTS and Dec-MCTS. Thus, other methods for solving the posed problems are not considered.

We observed that the performance of ordinary MCTS and Dec-MCTS can be improved enormously on the benchmark scenarios by incorporating the long horizon reward function g_2 as shown in Algorithm 2 and using completely random rollouts instead of actual high level plans. Because doing so merely constitutes a superior reward function as opposed to proper multi-level planning as performed by MH-MCTS, throughout the experiments Dec-MCTS is run with g_2 enabled and random rollouts being used during every evaluation. This as well as the weighting of g_1 and g_2 is discussed further in Section VII-C.

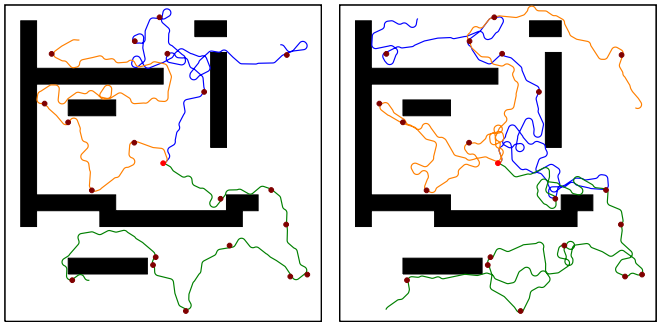
A. Waypoint survey

In the *waypoint survey* scenario, the robots operate on a work area with randomly placed obstacles and targets. The robots are assumed to be equipped with a sensor with a limited range and a target is considered 'observed' once a robot passes it within its sensor range. The robots' motions are constrained to forward movements obeying a minimum turning radius. Thus, the traversable space is covered by a PRM where nearby vertices are connected using Dubins paths.

The experiments shown here are performed on a simulated search area of 800 m $\times$ 800 m in size with randomly created walls as obstacles. The robots move with a speed of 2 m/s, the turning radius is 10 m and the sensor range for observations is 6 m. The PRM is created using 15,000 vertices. For the experiments 20 targets are created. An example of such a run is shown in Fig. 2.

MH-MCTS and Dec-MCTS are run in an RHC fashion with a planning horizon of 100 m where half of it is executed in each planning round. For benchmarking the simulation was sped up and each Dec-MCTS agent effectively had a computational budget of a little over 3 seconds per RHC round, single-threaded on an Intel Xeon Gold 6230R CPU with 2.10 GHz. Intents are exchanged between Dec-MCTS agents every 0.1 seconds. Each simulation was repeated 50 times and the median time to mission completion is reported here.

Note, that MH-MCTS employs two Dec-MCTS agents per robot and thus effectively uses twice as much CPU time than



(a) MH-MCTS (b) Dec-MCTS

Fig. 2: Example paths for MH-MCTS and Dec-MCTS on the waypoint survey with three vehicles. All three vehicles start in the middle of the map. The solution by MH-MCTS exhibits a better separation of the robots into different areas allowing for a more efficient mission.

ordinary Dec-MCTS. To rule out that the improved performance of MH-MCTS is merely due to additional compute, we performed additional simulations where Dec-MCTS was given twice the CPU time per agent. These runs are denoted with *double time*.

Quantitative results for the waypoint survey are shown in Fig. 3. MH-MCTS shows a clear performance benefit over plain Dec-MCTS, regardless of whether double compute time is used or not. It can be seen that the algorithms suffer from diminishing returns as more agents are added, a well-known limitation of Dec-MCTS. However, despite MH-MCTS effectively doubling the number of Dec-MCTS agents employed, returns stagnate only after adding the fourth vehicle whereas plain Dec-MCTS stagnates with three vehicles already. These significant performance improvements are due to the ability of MH-MCTS to perform long-term, non-myopic planning in a way that is consistent with the short-term, lower-level plans.

B. Grid world survey

In the *grid world survey*, the robots must survey the entirety of a squared area which is fitted with a road map graph that resembles a grid structure. To account for orientation, each grid position holds four vertices with different orientations (North, South, East, West) and each vertex is connected to three of its direct neighbours, allowing for movements forwards, left and right while disallowing doing U-turns. Such a contrived setup provides challenges to MCTS planning because many successive decisions have to be made to reach a position on the map.

For the experiments, the robots operate on a square shaped work area. Two different work area sizes are used here. The *small area* is 210 m $\times$ 210 m in size resulting in a 21 $\times$ 21 grid whereas the *large area* is 410 m $\times$ 410 m with a 41 $\times$ 41 grid. The goal is a complete survey of the area which means that all vertices of the road map must be visited. All vehicles start their mission at the center of the map looking East.

MH-MCTS and Dec-MCTS are again run in an RHC fashion with a planning horizon of 20 seconds, 10 seconds of which are executed. In each planning round, each Dec-MCTS agent has a computational budget of 10 seconds single-threaded on an Intel Xeon Gold 6230R CPU with 2.10 GHz. Intents are exchanged between Dec-MCTS agents every 0.1 seconds. Each simulation was repeated 50 times and the median time to mission completion is reported here.

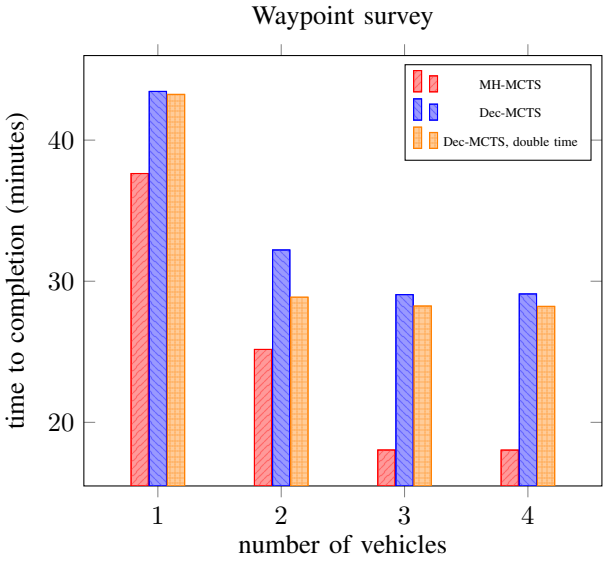


Fig. 3: Quantitative results for the waypoint survey scenario with different numbers of vehicles. MH-MCTS shows a clear advantage over plain Dec-MCTS, especially when multiple vehicles are deployed. All algorithms exhibit diminishing returns as more vehicles are added. However, MH-MCTS still shows an improvement when the third vehicle is added.

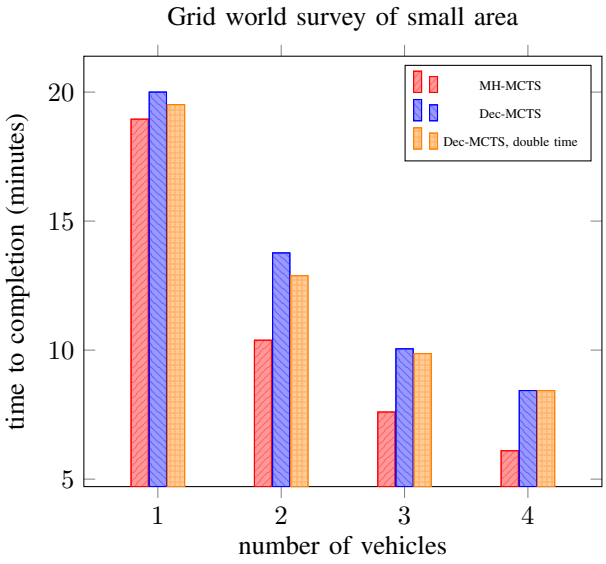


Fig. 4: Grid world survey of small area with different numbers of vehicles and planning algorithms. In this scenario MH-MCTS only provides a small advantage over Dec-MCTS.

Results for the small area are shown in Fig. 4. It can be seen that in this scenario MH-MCTS only provides a small advantage over Dec-MCTS. This suggests that in such small scenarios the advantage of long horizon planning is limited.

Results for the large area are shown in Fig. 5. Here MH-MCTS provides a substantial improvement over

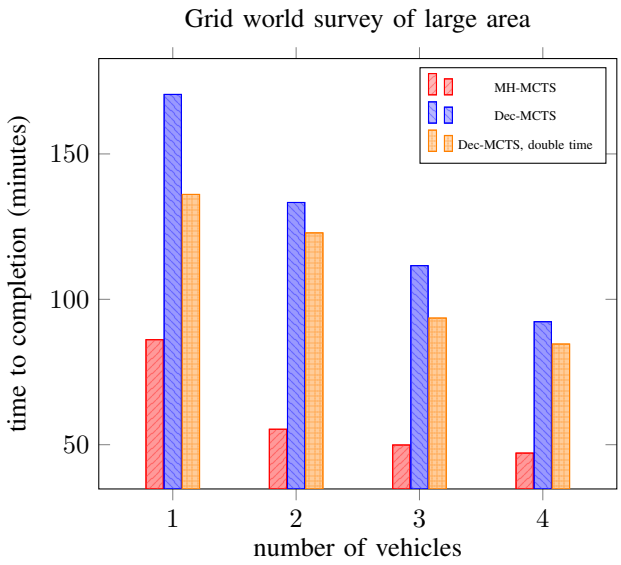


Fig. 5: Grid world survey of large area with different numbers of vehicles and planning algorithms. MH-MCTS provides substantial performance improvements compared to Dec-MCTS.

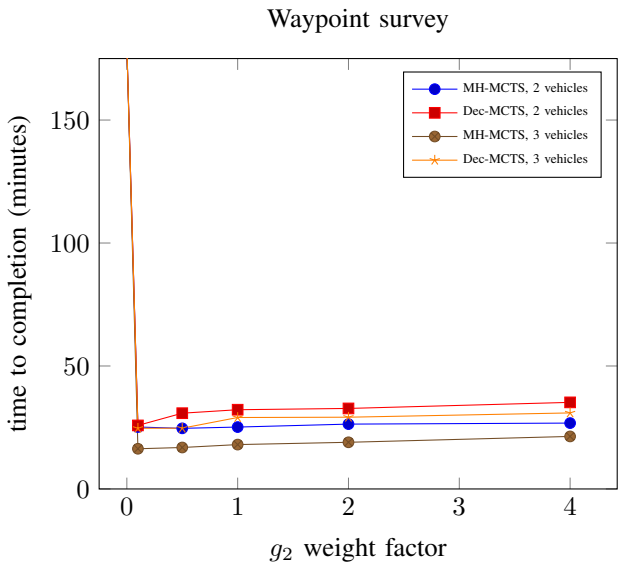


Fig. 6: The effect of varying weight factors for g_2. The system seems insensitive to a large range of weight values as long as they are larger than zero.

Dec-MCTS. It can be seen that the advantage diminishes as more vehicles are added. This is a somewhat expected effect as Dec-MCTS exhibits diminishing returns as more agents are added and MH-MCTS adds two agents per robot.

C. Effects of chosen parameters

1) Reward weight: In the implementation presented here, the reward function is decomposed into two parts according to Eq. (6). The two components measure different dimensions where g_1 returns a measure for the surface area that will be seen by the low level plan whereas g_2 returns an estimate for the total mission time. Thus, a question arises about how to weigh the two appropriately. The approach taken for the experiments above was to look at the average distance between adjacent targets. For the waypoint survey the average edge length of the minimum spanning tree between all targets is calculated whereas for the grid world survey the distance between neighbouring vertices is taken. The mission time returned by g_2 is then divided by the time it takes to nominally travel between targets:

$$g = g_1 + \frac{\text{robot speed}}{\text{average target distance}} g_2 \ . \tag{8}$$

We also conducted experiments to explore the system's sensitivity to the chosen value. An example of such experiments using the waypoint survey with two and three vehicles is shown in Fig. 6. Here, g_2 is multiplied with an additional weight factor such that a value of 1 corresponds to Eq. (8). It can be seen that the system is quite insensitive to the value of the weight as long as it is larger than zero. At a value of zero, the MCTS optimisation is performed using only g_1 which means no information beyond the current low level planning horizon is available to the planner. The high tolerance to different weighting values could indicate that the information from the high level plan is often used as a tie breaker in situations where multiple low level plans yield similar immediate rewards.

In case of ordinary Dec-MCTS, g_2 was evaluated using completely random choices as long horizon plans. That is, in Eq. (6), $\mathbf{s}^{2i}$ are random whereas $\mathbf{s}^{2i-1}$ are the intents for $i = 1, \ldots, N$.

Zooming into the tail end of Fig. 6 does not yield any additional insights and experiments with other scenarios produced very similar graphs. Thus, they are omitted here.

2) Gamma: Dec-MCTS introduced the parameter γ which discounts the importance of older episodes within the MCTS tree compared to recent ones in Eqs. (4) and (5). We performed a set of experiments to investigate the effect of γ on the performance of MH-MCTS in the benchmark scenarios. Results for the waypoint scenario are shown in Fig. 7. It can be seen that the system tends to be forgiving to specific values of γ as long as they are close enough to 1. The experiments before were performed with a value of $\gamma = 0.9999$. Other scenarios were also run with varied values for gamma and produced qualitatively similar results that suggest performance starts to deteriorate for values of γ that are too low.

VIII. Conclusion

We presented MH-MCTS, a novel framework which uses concepts from Dec-MCTS to enable multi-horizon multi-agent planning. The algorithm was evaluated using simple benchmark problems where it showed substantial improvements compared to ordinary Dec-MCTS. In experiments the algorithm seems tolerant to a variety of parameter choices including weighting of the reward components and discount factor γ. This indicates that MH-MCTS could be easily applicable over a wide range of applications. There appears to be a diminishing return as larger numbers of robots are added to the system. Thus, MH-MCTS is best used in situations where

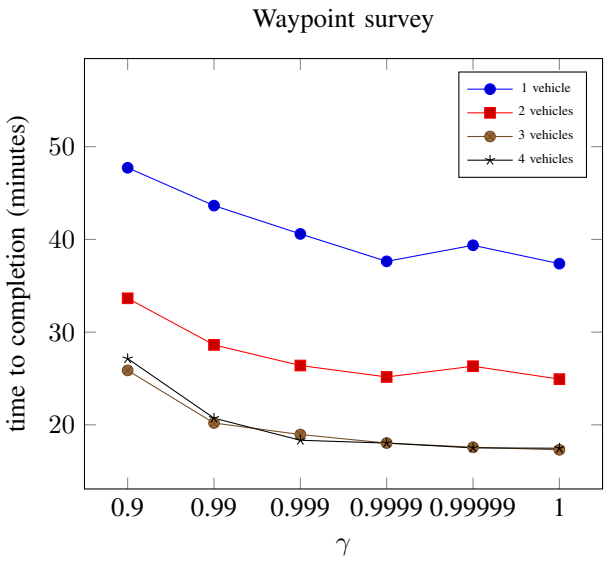


Fig. 7: Waypoint survey on the large map using MH-MCTS with different numbers of vehicles and values for γ. The system seems to be forgiving to specific values of γ as long as they are close enough to 1.

only a few robots must coordinate. Additionally, MH-MCTS can also provide benefits when only a single robot is used.

Future work will include the application of MH-MCTS to more complex problems as well as an investigation into the ability to scale to larger fleets.

References

[1] M. Świechowski, K. Godlewski, B. Sawicki, and J. Mańdziuk, "Monte Carlo Tree Search: A review of recent modifications and applications," *Artificial Intelligence Review*, vol. 56, no. 3, pp. 2497–2562, Mar. 1, 2023, ISSN: 1573-7462. DOI: 10.1007/s10462-022-10228-y.

[2] D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016. DOI: 10.1038/nature16961.

[3] Z. Hu, J. Tu, and B. Li, "Spear: Optimized Dependency-Aware Task Scheduling with Deep Reinforcement Learning," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, Jul. 2019, pp. 2037–2046. DOI: 10.1109/ICDCS.2019.00201.

[4] K. M. Seiler, A. W. Palmer, and A. J. Hill, "Flow-Achieving Online Planning and Dispatching for Continuous Transportation With Autonomous Vehicles," *IEEE Transactions on Automation Science and Engineering*, 2020, ISSN: 1558-3783. DOI: 10.1109/TASE.2020.3039908.

[5] T. Dam, G. Chalvatzaki, J. Peters, and J. Pajarinen, "Monte-Carlo Robot Path Planning," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 213–11 220, Oct. 2022, ISSN: 2377-3766. DOI: 10.1109/LRA.2022.3199674.

[6] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-MCTS: Decentralized planning for multi-robot active perception," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 316–337, Mar. 1, 2019, ISSN: 0278-3649. DOI: 10.1177/0278364918755924.

[7] S. Choudhury, J. K. Gupta, P. Morales, and M. J. Kochenderfer, "Scalable Online Planning for Multi-Agent MDPs," *Journal of Artificial Intelligence Research*, vol. 73, pp. 821–846, Mar. 10, 2022, ISSN: 1076-9757. DOI: 10.1613/jair.1.13261.

[8] G. D'Urso, J. J. Heon Lee, O. Pizarro, C. Yoo, and R. Fitch, "Hierarchical MCTS for Scalable Multi-Vessel Multi-Float Systems," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 8664–8670. DOI: 10.1109/ICRA48506.2021.9561064.

[9] C. B. Browne, E. Powley, D. Whitehouse, *et al.*, "A Survey of Monte Carlo Tree Search Methods," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 4, no. 1, pp. 1–43, Mar. 2012, ISSN: 1943-068X. DOI: 10.1109/TCIAIG.2012.2186810.

[10] N. A. Vien and M. Toussaint, "Hierarchical Monte-Carlo Planning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, Mar. 4, 2015, ISSN: 2374-3468. DOI: 10.1609/aaai.v29i1.9687.

[11] T. Gabor, J. Peter, T. Phan, C. Meyer, and C. Linnhoff-Popien, "Subgoal-Based Temporal Abstraction in Monte-Carlo Tree Search," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, Macao, China: International Joint Conferences on Artificial Intelligence Organization, Aug. 2019, pp. 5562–5568, ISBN: 978-0-9992411-4-1. DOI: 10.24963/ijcai.2019/772.

[12] L. Lu, W. Zhang, X. Gu, X. Ji, and J. Chen, "HMCTS-OP: Hierarchical MCTS Based Online Planning in the Asymmetric Adversarial Environment," *Symmetry*, vol. 12, no. 5, p. 719, 5 May 2020, ISSN: 2073-8994. DOI: 10.3390/sym12050719.

[13] R. Scattolini, "Architectures for distributed and hierarchical Model Predictive Control – A review," *Journal of Process Control*, vol. 19, no. 5, pp. 723–731, May 1, 2009, ISSN: 0959-1524. DOI: 10.1016/j.jprocont.2009.02.003.

[14] T. Shao, H. Zhang, K. Cheng, K. Zhang, and L. Bie, "The hierarchical task network planning method based on Monte Carlo Tree Search," *Knowledge-Based Systems*, vol. 225, p. 107 067, Aug. 5, 2021, ISSN: 0950-7051. DOI: 10.1016/j.knosys.2021.107067.

[15] D. Nau, Y. Cao, A. Lotem, and H. Munoz-Avila, "SHOP: Simple hierarchical ordered planner," in *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'99, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jul. 31, 1999, pp. 968–973.

[16] X. Neufeld, S. Mostaghim, and D. Perez-Liebana, "A hybrid planning and execution approach through HTN and MCTS," in *The 3rd Workshop on Integrated Planning, Acting, and Execution-ICAPS*, vol. 19, Berkeley, CA, USA, 2019, pp. 37–45. [Online]. Available: https://icaps19.icaps-conference.org/workshops/IntEx/IntEx-2019-proceedings.pdf.

[17] S. Ontañón and M. Buro, "Adversarial hierarchical-task network planning for complex real-time games," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15, Buenos Aires, Argentina: AAAI Press, Jul. 25, 2015, pp. 1652–1658, ISBN: 978-1-57735-738-4.

[18] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, 1996. DOI: 10.1109/70.508439.

[19] L. E. Dubins, "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957, ISSN: 0002-9327. DOI: 10.2307/2372560. JSTOR: 2372560.

[20] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50, 1956, ISSN: 0002-9939, 1088-6826. DOI: 10.1090/S0002-9939-1956-0078686-7.