Decentralised Monte Carlo Tree Search for Active Perception

Graeme Best[1], Oliver M. Cliff[1], Timothy Patten[1],
Ramgopal R. Mettu[2], and Robert Fitch[1,3]

[1]Australian Centre for Field Robotics (ACFR), The University of Sydney, Australia,
{g.best,o.cliff,t.patten,rfitch}@acfr.usyd.edu.au
[2]Department of Computer Science, Tulane University, New Orleans, LA, USA
rmettu@tulane.edu
[3]Centre for Autonomous Systems, University of Technology Sydney, Australia

Abstract. We propose a decentralised variant of Monte Carlo tree search (MCTS) that is suitable for a variety of tasks in multi-robot active perception. Our algorithm allows each robot to optimise its own individual action space by maintaining a probability distribution over plans in the joint-action space. Robots periodically communicate a compressed form of these search trees, which are used to update the locally-stored joint distributions using an optimisation approach inspired by variational methods. Our method admits any objective function defined over robot actions, assumes intermittent communication, and is anytime. We extend the analysis of the standard MCTS for our algorithm and characterise asymptotic convergence under reasonable assumptions. We evaluate the practical performance of our method for generalised team orienteering and active object recognition using real data, and show that it compares favourably to centralised MCTS even with severely degraded communication. These examples support the relevance of our algorithm for real-world active perception with multi-robot systems.

1 Introduction

Information gathering is a fundamentally important family of problems in robotics that plays a primary role in a wide variety of tasks, ranging from scene understanding to manipulation. Although the idea of exploiting robot motion to improve the quality of information gathering has been studied for nearly three decades [3], most real robot systems today (both single- and multi-robot) still gather information passively. The motivation for an active approach is that sensor data quality (and hence, perception quality) relies critically on an appropriate choice of viewpoints [17]. One way to efficiently achieve an improved set of viewpoints is through teams of robots, where concurrency allows for scaling up the number of observations in time and space. The key challenge, however, is to coordinate the behaviour of robots as they actively gather information, ideally in a decentralised manner. This paper presents a new, decentralised approach

for active perception that allows a team of robots to perform complex information gathering tasks using physically feasible sensor and motion models, and reasonable communication assumptions.

Decentralised active information gathering can be viewed, in general, as a partially observable Markov decision process (POMDP) in decentralised form (Dec-POMDP) [1]. There are several known techniques for solving this class of problems, but policies are usually computed in advance and executed in a distributed fashion. There are powerful approximation algorithms for special cases where the objective function is monotone submodular [21]. Unfortunately this is not always the case, particularly in field robotics applications.

Our approach provides convergence guarantees but does not require submodularity assumptions, and is essentially a novel decentralised variant of the Monte Carlo tree search (MCTS) algorithm [5]. At a high level, our method alternates between exploring each robot's individual action space and optimising a probability distribution over the joint-action space. In any particular round, we first use MCTS to find locally favourable sequences of actions for each robot, given a probabilistic estimate of other robots' actions. Then, robots periodically attempt to communicate a highly compressed version of their local search trees which, together, correspond to a product distribution approximation. These communicated distributions are used to estimate the underlying joint distribution. The estimates are probabilistic, unlike the deterministic representation of joint actions typically used in multi-robot coordination algorithms. Optimising a product distribution is similar in spirit to the mean-field approximation from variational inference, and also has a natural game-theoretic interpretation [19].

Our algorithm is a powerful new method of decentralised coordination for any objective function defined over the robot action sequences. Notably, this implies that our method is suitable for complex perception tasks such as object classification, which is known to be highly viewpoint-dependent [17]. Further, communication is assumed to be intermittent, and the amount of data sent over the network is small in comparison to the raw data generated by typical range sensors and cameras. Our method also inherits important properties from MCTS, such as the ability to compute anytime solutions and to incorporate prior knowledge about the environment. Moreover, our method is suitable for online replanning to adapt to changes in the objective function or team behaviour.

We evaluate our algorithm in two scenarios: generalised team orienteering and active object recognition. These experiments are run in simulation, and the second scenario uses range sensor data collected a priori by real robots. We show that our decentralised approach performs as well as or better than centralised MCTS even with a significant rate of message loss. We also show the benefits of our algorithm in performing long-horizon and online planning.

2 Related work

Information gathering problems can be viewed as sequential decision processes in which actions are chosen to maximise an objective function. Decentralised

coordination in these problems is typically solved myopically by maximising the objective function over a limited time horizon [25, 10]. Unfortunately, the quality of solutions produced by myopic methods can be arbitrarily poor in the general case. Recently, however, analysis of submodularity [16] has shown that myopic methods can achieve near-optimal performance [14], which has led to considerable interest in their application to information gathering with multiple robots [21]. While these methods provide theoretical guarantees, they require a submodular objective function, which is not applicable in all cases.

Efficient non-myopic solutions can be designed by exploiting problem-specific characteristics [4, 6]. But in general, the problem is a POMDP, which is notoriously difficult to solve. The difficulty of solving Dec-POMDPs is compounded because the search space grows exponentially with the number of robots. For our problem, we focus our attention on reasoning over the unknown plans of the other robots, while assuming other aspects of the problem are fully observable. The problems we consider here are therefore not Dec-POMDPs, but our algorithm is general enough to be extended to problems with partial observability.

MCTS is a promising approach for online planning because it efficiently searches over long planning horizons and is anytime [5]. MCTS has also been extended to partially observable environments [20]. However, MCTS has not been extended for decentralised multi-agent planning, and that is our focus here.

MCTS is parallelisable [7], and various techniques have been proposed that split the search tree across multiple processors and combine their results. In the multi-robot case, the joint search tree interleaves actions of individual robots and it remains a challenge to effectively partition this tree. A related case is multi-player games, where a separate tree may be maintained for each player [2]; however, a single simulation traverses all of the trees and therefore this approach would be difficult to decentralise. We propose a similar approach, except that each robot performs independent simulations while sampling from a locally stored probability distribution that represents the other robots' action sequences.

Coordination between robots is achieved in our method by combining MCTS with a framework that optimises a product distribution over the joint action space in a decentralised manner. Our approach is analogous to the classic mean-field approximation and related variational approaches [26, 19]. Variational methods seek to approximate the underlying global likelihood with a collection of structurally simpler distributions that can be evaluated efficiently and independently. These methods characterise convergence based on the choice of product distribution, and work best when it is possible to strike a balance between the convergence properties of the product distribution and the KL-divergence between the product and joint distributions. As discussed in the body of work on *probability collectives* [22, 24, 23], such variational methods can also be viewed under a game theoretic interpretation, where the goal is to optimise each agent's choice of actions based on examples of the global reward/utility function. The latter method has been used for solving the multiple-TSP in a decentralised manner [15]; we propose a similar approach, but we leverage the power of the MCTS to select an effective and compact sample space of action sequences.

3 Problem statement

We consider a team of R robots $\{1, 2, ..., R\}$, where each robot i plans its own sequence of future actions $\boldsymbol{x}^i = (x_1^i, x_2^i, ...)$. Each action x_j^i has an associated cost c_j^i and each robot has a cost budget B^i such that the sum of the costs must be less than the budget, i.e., $\sum_{x_j^i \in \boldsymbol{x}^i} c_j^i \leq B^i$. This cost budget may be an energy or time constraint defined by the application, or it may be used to enforce a planning horizon. The feasible set of actions and associated costs at each step n are a function of the previous actions $(x_1^i, x_2^i, ..., x_{n-1}^i)$. Thus, there is a predefined set of feasible action sequences for each robot $\boldsymbol{x}^i \in \mathcal{X}^i$. Further, we denote $\boldsymbol{x}$ as the set of action sequences for all robots $\boldsymbol{x} := \{\boldsymbol{x}^1, \boldsymbol{x}^2, ..., \boldsymbol{x}^R\}$ and $\boldsymbol{x}^{(i)}$ as the set of action sequences for all robots except i, i.e., $\boldsymbol{x}^{(i)} := \boldsymbol{x} \setminus \boldsymbol{x}^i$.

The aim is to maximise a global objective function $g(\boldsymbol{x})$ which is a function of the action sequences of all robots. We assume each robot i knows the global objective function $g(\boldsymbol{x})$, but does not know the actions $\boldsymbol{x}^{(i)}$ selected by the others. Moreover, the problem must be solved in an online setting.

We assume that robots can communicate to improve coordination. The communication channel may be unpredictable and intermittent, and all communication is asynchronous. Therefore, each robot will plan based on the information it has available locally. Bandwidth may be constrained and therefore message sizes should remain small, even as the plans grow. Although we do not consider explicitly planning to maintain communication connectivity, this may be encoded in the objective function $g(\boldsymbol{x})$ if a reliable communication model is available.

4 Dec-MCTS

In this section we present the Dec-MCTS algorithm. Dec-MCTS runs simultaneously and asynchronously on all robots; here we present the algorithm from the perspective of robot i. The algorithm cycles between three phases (Alg. 1): 1) grow a search tree using MCTS, while taking into account information about the other robots, 2) update the probability distribution over possible action sequences, and 3) communicate probability distributions with the other robots. These three phases continue regardless of whether or not the communication was successful, until a computation budget is met.

4.1 Local utility function

The global objective function g is optimised by each robot i using a local utility function f^i. We define f^i as the difference in utility between robot i taking actions $\boldsymbol{x}^i$ and a default "no reward" sequence $\boldsymbol{x}_\emptyset^i$, assuming fixed actions $\boldsymbol{x}^{(i)}$ for the other robots, i.e., $f^i(\boldsymbol{x}) := g(\boldsymbol{x}^i \cup \boldsymbol{x}^{(i)}) - g(\boldsymbol{x}_\emptyset^i \cup \boldsymbol{x}^{(i)})$. In practice, this improves the performance compared to optimising g directly since f^i is more sensitive to robot i's plan and the variance of f^i is less affected by the uncertainty of the other robots' plans [23].

Algorithm 1 Overview of Dec-MCTS for robot i.

input: global objective function g, budget B^i, feasible action sequences and costs
output: sequence of actions $\boldsymbol{x}^i$ for robot i

$\mathcal{T} \leftarrow$ initialise MCTS tree
while not converged or computation budget not met **do**
  $\hat{\mathcal{X}}_n^i \leftarrow \textsc{SelectSetOfSequences}(\mathcal{T})$
  for fixed number of iterations **do**
    $\mathcal{T} \leftarrow \textsc{GrowTree}(\mathcal{T}, \hat{\mathcal{X}}_n^{(i)}, q^{(i)}, B^i)$
    $q^i \leftarrow \textsc{UpdateDistribution}(\hat{\mathcal{X}}_n^i, q^i, \hat{\mathcal{X}}_n^{(i)}, q^{(i)}, \beta)$
    $\textsc{CommunicationTransmit}(\hat{\mathcal{X}}_n^i, q^i)$
    $(\hat{\mathcal{X}}_n^{(i)}, q^{(i)}) \leftarrow \textsc{CommunicationReceive}$
    $\beta \leftarrow \textsc{cool}(\beta)$
return $\boldsymbol{x}^i \leftarrow \arg\max_{\boldsymbol{x}^i \in \hat{\mathcal{X}}_n^i} \left[q^i(\boldsymbol{x}^i)\right]$

4.2 Monte Carlo tree search

The first phase of the algorithm is the MCTS update shown in Alg. 2. A single tree is maintained by robot i which only contains the actions of robot i. Coordination occurs implicitly by considering the plans of the other robots when performing the rollout policy and evaluation of the global objective function. This information about the other robots' plans comes from the distributed optimisation of probability distributions detailed in the following subsection. We use MCTS with a novel bandit-based node selection policy.

Standard MCTS incrementally grows a tree by iterating through four phases: selection, expansion, simulation and backprogation [5]. In each iteration t, a new leaf node is added, where each node represents a sequence of actions and contains statistics about the expected reward. The selection phase begins at the root node of the tree and recursively selects child nodes s_j until an expandable node is reached. For selecting the next child, we propose an extension of the UCT policy [13], detailed later, to balance exploration and exploitation. In the expansion phase, a new child is added to the selected node, which extends the parent's action sequence with an additional action.

In the simulation phase, the expected utility $\mathbb{E}[g_j]$ of the expanded node s_j is estimated by performing and evaluating a rollout policy that extends the action sequence represented by the node until a terminal state is reached. This rollout policy could be a random policy or a heuristic for the problem. The objective is evaluated for this sequence of actions and this result is saved as $\mathbb{E}[g_j]$.

For our problem, the objective is a function of the action sequence $\boldsymbol{x}^i$ as well as the unknown plans of the other robots $\boldsymbol{x}^{(i)}$. To compute the rollout score, we first sample $\boldsymbol{x}^{(i)}$ from a probability distribution $q_n^{(i)}$ over the plans of the other agents (defined later). A heuristic rollout policy extended from s_j defines $\boldsymbol{x}^i$, which should be a function of $\boldsymbol{x}^{(i)}$ to simulate coordination. Additionally, we optimise $\boldsymbol{x}^i$ using the local utility f^i (defined in Sec. 4.1) rather than g. The rollout score is then computed as the utility of this joint sample $f^i(\boldsymbol{x}^i \cup \boldsymbol{x}^{(i)})$,

Algorithm 2 Grow the tree using Monte Carlo Tree Search for robot i.

1: **function** GROWTREE($\mathcal{T}, \hat{\mathcal{X}}_n^{(i)}, q^{(i)}, B^i$)
   input: partial tree $\mathcal{T}$, distributions for other robots $(\hat{\mathcal{X}}_n^{(i)}, q^{(i)})$
   output: updated partial tree $\mathcal{T}$
2: **for** fixed number of samples τ **do**
3: $s_j \leftarrow$ NODESELECTIOND-UCT($\mathcal{T}$) ▷ Find the next node to expand
4: $s_j^+ \leftarrow$ EXPANDTREE(s_j) ▷ Add a new child to s_j
5: $\boldsymbol{x}^{(i)} \leftarrow$ SAMPLE($\hat{\mathcal{X}}_n^{(i)}, q^{(i)}$) ▷ Sample the policies of the other robots
6: $\boldsymbol{x}^i \leftarrow$ PERFORMROLLOUTPOLICY($s_j^+, \boldsymbol{x}^{(i)}, B^i$)
7: $score \leftarrow f^i(\boldsymbol{x}^i \cup \boldsymbol{x}^{(i)})$ ▷ Local utility function
8: $\mathcal{T} \leftarrow$ BACKPROPAGATION($s_j^+, score$) ▷ Update scores
9: **return** $\mathcal{T}$

which is an estimate for $\mathbb{E}_{q_n}[f^i \mid \boldsymbol{x}^i]$. Thus, we define the reward $F_t(s, s_j)$ as the rollout score when child s_j was expanded from node s at sample t.

In the backpropagation phase, the rollout evaluation is added to the statistics of all nodes along the path from the expanded node back to the root of the tree. Typically, each rollout t is treated as equally relevant and therefore $\mathbb{E}[g_j]$ is an unbiased average of the rollout evaluations. However, our algorithm alternates between growing the tree for a fixed number of rollouts τ and updating the probability distributions for other robots at each iteration n, where $n = \lfloor t/\tau \rfloor$. Therefore the most recent rollouts are more relevant since they are obtained by sampling the most recent distributions. Thus, we use a variation on the standard UCT algorithm, which we term D-UCT after discounted UCB [11]. This policy accounts for non-stationary reward distributions by biasing each sample by a weight γ which increases at each rollout.

Specifically, we propose a node selection policy (Alg. 2 line 3) that maximises a UCB $\bar{F}_t(\cdot) + c_t(\cdot)$ for the discounted expected reward, i.e., for parent node s and sample t, D-UCT selects the child node $s_t^+ = \arg\max_{s_j}[\bar{F}_t(\gamma, s, s_j) + c_t(\gamma, s, s_j)]$. This continues recursively until a node with unvisited children is reached. The discounted empirical average $\bar{F}_t$ is given by

$$\bar{F}_t(\gamma, s, s_j) = \frac{1}{N_t(\gamma, s, s_j)} \sum_{u=1}^{t} \gamma^{t-u} F_u(s, s_j) \mathbb{1}_{\{s_u^+ = s_j\}}, \tag{1}$$

$$N_t(\gamma, s, s_j) = \sum_{u=1}^{t} \gamma^{t-u} \mathbb{1}_{\{s_u^+ = s_j\}},$$

and the discounted exploration bonus c_t is given by

$$c_t(\gamma, s, s_j) = B\sqrt{\frac{\xi \log N_t(\gamma, s)}{N_t(\gamma, s, s_j)}}, \qquad N_t(\gamma, s) = \sum_{j=1}^{K} N_t(\gamma, s, s_j). \tag{2}$$

In this context, $N_t(\gamma, s)$ is the discounted number of times the current (parent) node s has been visited, and $N_t(\gamma, s, s_j)$ is the discounted number of times child

node s_j has been visited. The constants $\gamma \in (1/2, 1)$, $\xi > 0.5$, and B is the maximum reward. This D-UCT selection policy guarantees a rate of regret in the bandit case with abruptly changing distributions, which we discuss in Sec. 5.

4.3 Decentralised product distribution optimisation

The second phase of the algorithm updates a probability distribution q^i over the set of possible action sequences for robot i (Alg. 3). These distributions are communicated between robots and used for performing rollouts during MCTS. To optimise these distributions in a decentralised manner for improving global utility, we adapt a type of variational method originally proposed in [22]. This method can be viewed as a game between independent robots, where each robot selects their action sequence by sampling from a distribution.

One challenge is that the set of possible action sequences is typically of exponential size. We obtain a sparse representation by selecting the sample space $\hat{\mathcal{X}}^i_n \subset \mathcal{X}^i$ as the most promising action sequences $\{\boldsymbol{x}^i_1, \boldsymbol{x}^i_2, ...\}$ found by MCTS. We select a fixed number of nodes with the highest $\mathbb{E}[f^i]$ obtained so far. $\hat{\mathcal{X}}^i_n$ is the action sequences used during the initial rollouts when the selected nodes were first expanded.

The set $\hat{\mathcal{X}}^i_n$ has an associated probability distribution q^i such that $q^i(\boldsymbol{x}^i)$ defines the probability that robot i will select $\boldsymbol{x}^i \in \hat{\mathcal{X}}^i_n$. The distributions for different robots are independent and therefore define a product distribution, such that the probability of a joint action sequence selection $\boldsymbol{x}$ is $q(\boldsymbol{x}) \coloneqq \prod_{i \in \{1:R\}} q^i(\boldsymbol{x}^i)$. The advantage of defining q as a product distribution is so that each robot selects its action sequence independently, and therefore allows decentralised execution.

Consider the general class of joint probability distributions p that are not restricted to product distributions. Define the expected global objective function for a joint distribution p as $\mathbb{E}_p[g]$, and let $\mathit{\Gamma}$ be a desired value for $\mathbb{E}_p[g]$. According to information theory, the most likely p that satisfies $\mathbb{E}[g] = \mathit{\Gamma}$ is the p that maximises entropy. The most likely p can be found by minimising the maxent Lagrangian: $L(p) \coloneqq \lambda\,(\mathit{\Gamma} - \mathbb{E}_p[g]) - \mathrm{H}(p)$, where $\mathrm{H}(p)$ is the Shannon entropy and λ is a Lagrange multiplier. The intuition is to iteratively increase $\mathit{\Gamma}$ and optimise p. A descent scheme for p can be formulated with Newton's method.

For decentralised planning and execution, we are interested in optimising the product distribution q rather than a more general joint distribution p. We can approximate q by finding the q with the minimum pq KL-divergence $D_{\mathrm{KL}}(p \parallel q)$. This formulates a descent scheme with the update policy for q^i shown in Alg. 3 line 5, and where we use f^i rather than g. Intuitively, this update rule increases the probability that robot i selects $\boldsymbol{x}^i$ if this results in an improved global utility, while also ensuring the entropy of q^i does not decrease too rapidly.

Pseudocode for this approach is in Alg. 3. We require computing two expectations (lines 3 and 4) to evaluate the update equation (line 5). It is often necessary to approximate these expectations by sampling $\boldsymbol{x}$, since it is intractable to sum over the enumeration of all $\boldsymbol{x} \in \hat{\mathcal{X}}_n$. Parameter β should slowly decrease and α remain fixed. For efficiency purposes, in our implementation q^i is set to a uniform distribution when $\hat{\mathcal{X}}^i_n$ changes (Alg. 1 line 3).

Algorithm 3 Probability distribution optimisation for robot i.

function UPDATEDISTRIBUTION($\hat{\mathcal{X}}_n^i, q^i, \hat{\mathcal{X}}_n^{(i)}, q^{(i)}, \beta$)
  input: action sequence set for each robot $\hat{\mathcal{X}}_n \coloneqq \{\hat{\mathcal{X}}_n^1, \hat{\mathcal{X}}_n^2, ..., \hat{\mathcal{X}}_n^R\}$ with associated probability distributions $\{q^1, q^2, ..., q^R\}$, update parameter β
  output: updated probability distribution q^i for robot i
  for each $\boldsymbol{x}^i \in \hat{\mathcal{X}}_n^i$ **do**
    $\mathbb{E}_q[f^i] \leftarrow \sum_{\boldsymbol{x} \in \hat{\mathcal{X}}_n} \left[f^i(\boldsymbol{x}) \prod_{i' \in \{1:R\}} q^{i'}(\boldsymbol{x}^{i'}) \right]$
    $\mathbb{E}_q[f^i \mid \boldsymbol{x}^i] \leftarrow \sum_{\boldsymbol{x}^{(i)} \in \hat{\mathcal{X}}_n^{(i)}} \left[f^i(\boldsymbol{x}^i \cup \boldsymbol{x}^{(i)}) \prod_{i' \in \{1:R\} \setminus i} q^{i'}(\boldsymbol{x}^{i'}) \right]$
    $q^i(\boldsymbol{x}^i) \leftarrow q^i(\boldsymbol{x}^i) - \alpha q^i(\boldsymbol{x}^i) \left[\frac{\mathbb{E}_q[f^i] - \mathbb{E}_q[f^i \mid \boldsymbol{x}^i]}{\beta} + \mathrm{H}(q^i) + \ln\left(q^i(\boldsymbol{x}^i)\right) \right]$
    $q^i \leftarrow$ NORMALISE(q^i)
  return q^i

4.4 Message passing

At each iteration n, robot i communicates its current probability distribution $(\hat{\mathcal{X}}_n^i, q_n^i)$ to the other robots. If robot i receives any updated distributions then this replaces its stored distribution. The updated distribution is used during the next iteration. If no new messages are received from a robot, then robot i continues to plan based on its most recently received distribution. If robot i is yet to receive any messages from a robot then it may assume a default policy.

4.5 Online replanning

The best action is selected as the first action in the highest probability action sequence in $\hat{\mathcal{X}}_n^i$. The search tree may then be pruned by removing all children of the root except the selected action. Planning may then continue while using the sub-tree's previous results. If the objective function changes, e.g. as a result of a new observation, then the tree should be restarted. In practice, if the change is minor then it may be appropriate to continue planning with the current tree.

5 Analysis

We characterise the convergence properties of the two key algorithmic components of our approach: tree search (Sec. 4.2) and sample space contraction (Sec. 4.3). Our first aim is to show that, with the D-UCT algorithm (Alg. 2), we maintain an exploration-exploitation trade-off for child selection while the distributions q_n^i are changing (and converging). Then we characterise the convergence of Alg. 3 given a contracted sample space of distributions $\hat{\mathcal{X}}_n^i \subset \mathcal{X}^i$.

We analyse the D-UCT algorithm by considering a particular type of non-stationary (adversarial) multi-armed bandit (MAB) problem [13, 11]. That is, we consider a unit-depth tree where the reward distributions can change abruptly at a given *epoch*, such that a previously optimal arm becomes suboptimal.

Theorem 1. *Suppose we restrict $\mathcal{T}$ to a unit-depth tree in Alg. 1. Then, Alg. 2 will sample suboptimal actions $\tilde{\boldsymbol{x}}^i$, where $\tilde{\boldsymbol{x}}^i$ is suboptimal w.r.t. $\mathbb{E}_{q_n}[f^i]$, at a rate of $\mathcal{O}(n^{(1+\varphi_{\tau,\beta})/2}\log n)$ for some $\varphi_{\tau,\beta} \in [0,1)$.*

Proof. Observe that the problem of action selection for a unit-depth tree is equivalent to an MAB problem [13]. Denote by Υ_T the number of switching points (breakpoints) of the reward distribution up until some sample horizon T. In the context of Sec. 4.2, these are sample rounds when a previously optimal child s_j becomes suboptimal (i.e., $F_t(s, s_j)$ changes abruptly). Under the D-UCB policy applied to the MAB problem, given a known number of breakpoints Υ_T, we can minimise the number of times a suboptimal action is taken by setting the discount factor to $\gamma = 1 - (4B)^{-1}\sqrt{\Upsilon_T/T}$. Selecting this discount factor upper bounds the expected number of times a suboptimal action is taken to $\mathcal{O}(\sqrt{T\Upsilon_T}\log T)$ [11]. Depending on the rate of increasing β, the expected utility will converge asymptotically. Without loss of generality, we can therefore assume the reward distributions $F_t(s, s_j)$ (and thus Υ_T) converges at a rate $\mathcal{O}(n^{\varphi_{\tau,\beta}})$, where $\varphi_{\tau,\beta} \in [0,1)$, depending on the (sample) epoch length τ and β. Given that an iteration can be calculated as $n = \lfloor T/\tau \rfloor$ for a given sample horizon T, the number of suboptimal action selections is upper bounded by $\mathcal{O}(n^{(1+\varphi_{\tau,\beta})/2}\log n)$ for some fixed (τ, β). □

Remark 1. Although Theorem 1 analyses Alg. 1 in the MAB setting, we believe it is possible to extend to arbitrary depth trees. This could be achieved in a similar way to [13] by showing drift and breakpoint conditions are satisfied at all levels of the tree and proving regret bounds by induction.

As a consequence of Theorem 1 and Remark 1, during abrupt changes to q_n, the child selection policy in the tree search balances exploration and exploitation.

Importantly, note that nearer to the root node of the tree the breakpoints will decay faster than the leaf nodes. Thus, toward the root node, as n becomes large, $\gamma \to 1$ and the number of breakpoints Υ_T remain constant with high probability. For these nodes, the D-UCT algorithm becomes equivalent to UCT (setting $\xi = 2C_p^2 = 2$ and $B = 1$ to satisfy the tail conditions in [13]). As per Theorem 2 of [13], the bias of the estimated utility then converges polynomially. This is reasonable since, given that we will typically want to adaptively replan (Sec. 4.5), it should be sufficient to guarantee optimal short-term action sequences. Moreover, note that γ can be optimally set given bounds on the convergence *rate* of these distributions (i.e., $\varphi_{\tau,\beta}$, which is dependent on τ and β).

We now consider the effect of contracting the sample space $\hat{\mathcal{X}}_n \subset \mathcal{X}$ on the convergence of Alg. 3. Recall that the *pq* KL-divergence is the divergence from a product distribution q to the optimal joint distribution p. We then have the following proposition:

Proposition 1. *Alg. 3 asymptotically converges to a distribution that locally minimises the pq KL-divergence, given an appropriate subset $\hat{\mathcal{X}}_n \subset \mathcal{X}$.*

Proposition 1 relies on growing β slowly. Consider the situation where, at each iteration n, we randomly choose a subset $\hat{\mathcal{X}}_n^i \subset \mathcal{X}^i$ for each robot. This approach

is equivalent to Monte Carlo sampling of the expected utility and thus the biased estimator is consistent (asymptotically converges to $\mathbb{E}[f^i]$). For tractable computation, in our algorithm we modify the random selection by choosing a sparse set of strategies $\hat{\mathcal{X}}_n$ with the highest expected utility (Sec. 4.3). Although this does not ensure we sample the entire domain $\mathcal{X}$ asymptotically, in practice $q_n(\hat{\mathcal{X}}_n)$ is a reasonably accurate representation of $q_n(\mathcal{X})$, and therefore this gives us an approximation to importance sampling [24].

The analyses above show separately that the tree search of Alg. 2 balances exploration and exploitation and that, under reasonable assumptions, Alg. 3 converges to the product distribution that best optimises the joint action sequence. These two components do not immediately yield a characterisation of optimality for Alg. 1. To prove global convergence rates, we would need to characterise the co-dependence between the evolution of the reward distributions $\mathbb{E}_{q_n}[f^i \mid \boldsymbol{x}^i]$ and the contraction of the sample space $\hat{\mathcal{X}}_n$. The following experiments show that the algorithm converges rapidly to a high-quality solution.

6 Experiments: Generalised team orienteering

In this section we evaluate the performance of our algorithm in an abstract multi-robot information gathering problem (Fig. 1). We show convergence, robustness to intermittent communication and a comparison to centralised MCTS.

The problem is motivated by tasks where a team of Dubins robots maximally observes a set of features of interest in an environment, given a travel budget [4]. Each feature can be viewed from multiple viewpoints and each viewpoint may be in observation range of multiple features. This formulation generalises the orienteering problem [12] by combining the set structure of the generalised travelling salesman problem with the budget constraints of the orienteering problem with neighbourhoods [9] extended for multi-agent scenarios [4].

Robots navigate within a graph representation of an environment with vertices $v_i \in \mathcal{V}$, edges $e_{ij} := \langle v_i, v_j \rangle \in \mathcal{E}$ and edge traversal costs c_{ij}. Each vertex v_i represents a location and orientation (x, y, θ) within a square workspace with randomly placed obstacles. The action sequences of each robot are defined as paths through the graph from a fixed start vertex unique to each robot to a free destination vertex. The edge costs are defined as the distance of the Dubins path between the two configurations. All edges are connected within a fixed distance.

For the objective function, we have a collection of sets $\mathcal{S} = (S_1, S_2, ...)$, where each $S_i \subseteq \mathcal{V}$. These sets may represent a set of features of interest, where a vertex is an element of a set only if the associated feature can be observed from the vertex location. We assume each set is a disc, however the formulation could extend to more complex models [4]. The vertices $v_j \in \mathcal{V}$ are randomly placed within the sets. A set S_i is visited if $\exists v_j \in \boldsymbol{x}, v_j \in S_i$ and each visited set yields an associated reward w_i. There is no additional reward for revisiting a set. The objective is defined as the sum of the rewards of all visited sets.

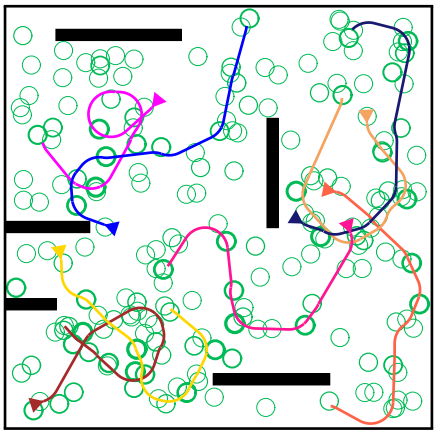

Fig. 1. The generalised team orienteering problem. The 8 robots (coloured paths) aim to collectively visit a maximal number of goal regions (green circles, weighted by importance). The robots follow Dubins paths, are constrained by distance budgets and must avoid obstacles (black).

6.1 Experiment setup

We compare our algorithm (Dec-MCTS) to centralised MCTS (Cen-MCTS), which consists of a single tree where robot i's actions appear at tree depths $(i, i+R, i+2R, ...)$. Intermittent communication is modelled by randomly dropping messages. Messages are broadcast by each robot at 4 Hz and a message has a probability of being received by each individual robot.

Experiments were performed with 8 simulated robots running in separate ROS nodes on an 8-core computer. Each random problem instance (Fig. 1) consisted of 200 discs with rewards between 1 and 10, 4000 graph vertices and 5 obstacles. Each iteration of Alg. 1 performs 10 MCTS rollouts, and $\hat{\mathcal{X}}_n^i$ consists of 10 paths that are resampled every 10 iterations. The MCTS rollout policy recursively selects the next edge that does not exceed the travel budget and maximises the ratio of the increase of the weighted set cover to the edge cost.

6.2 Results

The first experiments (Fig. 2(a)) show that Dec-MCTS achieved a median 7 % reward improvement over Cen-MCTS after 120 s, and a higher reward in 91 % of the environments. Dec-MCTS typically converged after ∼60 s. A paired single-tailed t-test supports the hypothesis ($p < 0.01$) that Dec-MCTS achieves a higher reward than Cen-MCTS for time > 7 s. Cen-MCTS performs well initially since it performs a centralised greedy rollout that finds reasonable solutions quickly. Dec-MCTS soon reaches deeper levels of the search trees, though, which allows it to outperform Cen-MCTS. Dec-MCTS uses a collection of search trees with smaller branching factors than Cen-MCTS, but still successfully optimises over the joint-action space.

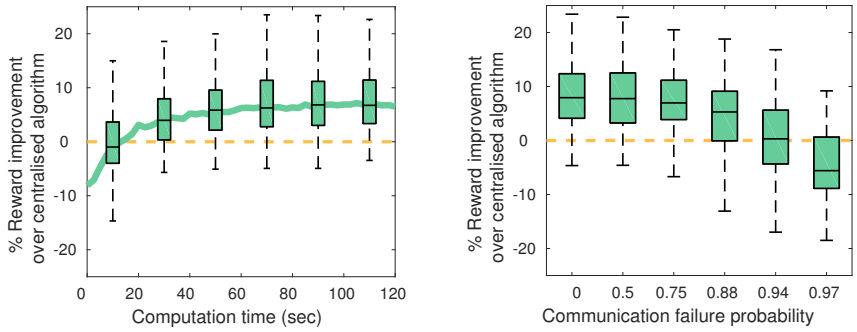

Fig. 2. (a) Comparison of Dec-MCTS with varying computation time to Cen-MCTS (120 s). (b) Performance of Dec-MCTS with intermittent communication (60 s computation time). (a,b) Vertical axes show percentage additional reward achieved by Dec-MCTS compared to Cen-MCTS. Error bars show 0, 25, 50, 75 and 100 percentiles (excluding outliers) of 100 random problem instances.

The second experiments analysed the effect of communication degradation. When the robots did not communicate, the algorithm achieved a median 31 % worse than Cen-MCTS, but with full communication achieves 7 % better than centralised, which shows the robots can successfully cooperate by using our proposed communication algorithm. Fig. 2(b) shows the results for partial communication degradation. When half of the packets are lost, there is no significant degradation of performance. When 97 % of packets are lost the performance is degraded but still performs significantly better than with no communication.

7 Experiments: Active object recognition

This section describes experiments in the context of online active object recognition, using point cloud data collected from an outdoor mobile robot in an urban scene (Fig. 3). We first outline the problem and experiment setup, and then present results that analyse the value of online replanning and compare Dec-MCTS to a greedy planner.

A team of robots aim to determine the identity of a set of static objects in an unknown environment. Each robot asynchronously executes the following cycle: 1) plan a path that is expected to improve the perception quality, 2) execute the first planned action, 3) make a point cloud observation using onboard sensors, and then 4) update the belief of the object identities. The robots have the same motion model and navigation graph as Sec. 6. Each graph edge has an additional constant cost for the observation processing time.

The robots maintain a belief of the identity of each observed object, represented as the probability that each object is an instance of a particular object from a given database. The aim is to improve this belief, which is achieved by

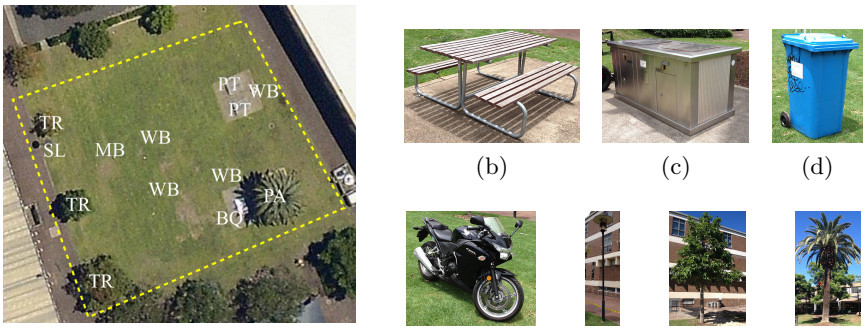

Fig. 3. Experiment setup for the point cloud dataset. (a) Environment with labelled locations, (b) picnic table (PT), (c) barbecue (BQ), (d) wheelie bin (WB), (e) motorbike (MB), (f) street light (ST), (g) tree (TR), (h) palm tree (PT).

maximising the mutual information objective proposed in [18]. The posterior probability distribution for each object after a set of observations is computed using recursive Bayes' rule. The observation likelihood is computed as the symmetric residual error of an iterative closest point (ICP) alignment [8] with each model in the database. Objects may merge or split after each observation if the segmentation changes. Observations are fused using decentralised data fusion or a central processor and shared between all robots. While planning, the value of future observations are estimated by simulating observations of objects in the database for all possible object identities, weighted by the belief.

7.1 Experiment setup

The experiments use a point cloud dataset [18] of Velodyne scans of outdoor objects in a $30\times30\,\mathrm{m}^2$ park (Fig. 3(a)). The environment consisted of 13 objects from 7 different model types as shown in Figs. 3(b)-(h). The dataset consists of single scans from 50 locations and each scan was split into 8 overlapping observations with different orientations. Each observation had a 180° field of view and 8 m range. These locations and orientations form the roadmap vertices with associated observations. Each object was analysed from separate data to generate the model database. The robots are given a single long-range observation from the start location to create an initial belief of most object locations.

The experiments simulate an online mission where each robot asynchronously alternates between planning and perceiving. Three planners were trialled: our Dec-MCTS algorithm with 120 s replanning after each action, Dec-MCTS without replanning, and a decentralised greedy planner that selects the next action that maximises the mutual information divided by the edge cost. The *recognition score* of an executed path was calculated as the belief probability that each object matched the ground-truth object type, averaged over all objects.

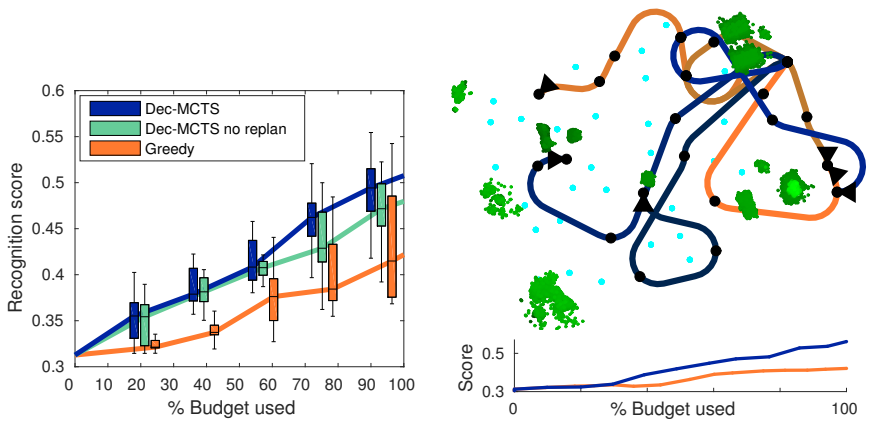

Fig. 4. (a) Task performance over mission duration for 10 trials (maximum possible score is 0.62). (b) Overlay of 2 example missions with 3 robots. Blue paths denote online Dec-MCTS (score 0.53). Orange paths denote greedy policy (score 0.42). Objects are green point clouds where shading indicates height. Robots observe at black dots in direction of travel. Start location top right.

7.2 Results

Overall, results validate the coordination performance of Dec-MCTS. Fig. 4(a) shows the recognition score (task performance) over the duration of the mission for 10 trials with 3 robots. The maximum possible recognition score subject to the perception algorithm and dataset was 0.62. Dec-MCTS outperformed greedy halfway through the missions since some early greedy decisions and poor coordination reduced the possibility of making subsequent valuable observations. By the end of the missions some greedy plans successfully made valuable observations, but less often than Dec-MCTS. The no-replanning scenario achieved a similar score as the online planner in the first half, showing that the initial plans are robust to changes in the belief. For the second half, replanning improved the recognition score since the belief had changed considerably since the start. This shows that while the generated plans are reasonable for many steps into the future, there is also value in replanning as new information becomes available.

Fig. 4(b) shows two example missions using online Dec-MCTS (blue) and greedy (orange) planners, and their score over the mission duration. Greedy stayed closer to the start location to improve the recognition of nearby objects, and consequently observed objects on the left less often; reaching this part of the environment would require making high cost/low immediate value actions. On the other hand, Dec-MCTS achieved a higher score since the longer planning horizon enabled finding the high value observations on the left, and was better able to coordinate to jointly observe most of the environment.

8 Discussion and future work

We have presented a novel approach to decentralised coordination with convergence properties. The performance (i.e., solution quality) of our approach is as good or better than its centralised counterpart in real-world applications with sensor data, and shows that our approach can effectively optimise a joint-action space even with intermittent communication. A key conceptual feature of our approach is its generality in representing joint actions probabilistically rather than deterministically. This facilitates the decentralisation of a variety of tasks while maintaining convergence properties drawn from statistics and game theory.

One interesting aspect of our work is that it is straightforward to extend it to problems that have partial observability. That is, we can replace MCTS with POMCP [20] and apply the same general framework. For example, this provides a convenient decentralised approach to problems in active perception, such as active classification. An interesting question is whether the same or similar convergence properties to the fully observable case can be maintained.

Another interesting line of inquiry is to incorporate coalition forming into our approach. As formulated, static coalitions of agents can be formed by generalising the product distributions in our framework to be partial joint distributions. The product distribution described in Sec. 4.3 would be defined over *groups* of robots rather than individuals. Each group acts jointly, with a single distribution modelling the joint actions of its members, and coordination between groups is conducted as in our algorithm. A natural field robotics application would be mobile robots, each with multiple manipulators, for weeding and harvesting in agriculture. Just as our current approach corresponds to mean-field methods, this approach maps nicely to region-based variational methods [26] and guarantees from these approaches may be applicable. It would also be interesting to study *dynamic* coalition forming, where the mapping between agents and robots is allowed to change, and to develop convergence guarantees for this case.

Acknowledgements. This work was supported by the Australian Centre for Field Robotics; the NSW Government; the Australian Research Council's Discovery Project funding scheme (No. DP140104203); the Faculty of Engineering & Information Technologies, The University of Sydney, under the Faculty Research Cluster Program; and the University's International Research Collaboration Award.

References

1. Amato, C.: Decision Making Under Uncertainty: Theory and Application, chap. Cooperative Decision Making. MIT Press, Cambridge and London (2015)
2. Auger, D.: Multiple tree for partially observable Monte-Carlo tree search. In: Proc. of EvoApplications. pp. 53–62 (2011)
3. Bajcsy, R.: Active perception. Proceedings of the IEEE 76(8), 966–1005 (1988)
4. Best, G., Faigl, J., Fitch, R.: Multi-robot path planning for budgeted active perception with self-organising maps. In: Proc. of IEEE/RSJ IROS (2016)
5. Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A survey of Monte Carlo tree search methods. IEEE T. Comput. Int. AI Games 4(1), 1–43 (2012)

6. Charrow, B.: Information-theoretic active perception for multi-robot teams. Ph.D. thesis, University of Pennsylvania (2015)
7. Chaslot, G.M.J.B., Winands, M.H.M., van den Herik, H.J.: Parallel Monte-Carlo tree search. In: Proc. of CG. pp. 60–71 (2008)
8. Douillard, B., Quadros, A., Morton, P., Underwood, J.P., Deuge, M.D.: A 3D classifier trained without field samples. In: Proc. of ICARCV. pp. 805 – 810 (2012)
9. Faigl, J., Pěnička, R., Best, G.: Self-organizing map-based solution for the orienteering problem with neighborhoods. In: Proc. of IEEE SMC (2016)
10. Gan, S.K., Fitch, R., Sukkarieh, S.: Online decentralized information gathering with spatial–temporal constraints. Auton. Robots 37(1), 1 – 25 (2014)
11. Garivier, A., Moulines, E.: On upper-confidence bound policies for switching bandit problems. In: Proc. of AMA ALT. pp. 174–188 (2011)
12. Gunawan, A., Lau, H.C., Vansteenwegen, P.: Orienteering problem: A survey of recent variants, solution approaches and applications. Eur. J. Oper. Res. 255(2), 315 – 332 (2016)
13. Kocsis, L., Szepesvári, C.: Bandit based Monte-Carlo planning. In: Proc. of ECML. pp. 282–293 (2006)
14. Krause, A., Singh, A., Guestrin, C.: Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. J. Mach. Learn. Res. 9, 235 – 284 (2008)
15. Kulkarni, A.J., Tai, K.: Probability collectives: A multi-agent approach for solving combinatorial optimization problems. Appl. Soft Comput. 10(3), 759 – 771 (2010)
16. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions–I. Math. Program. 14(1), 265 – 294 (1978)
17. Patten, T., Zillich, M., Fitch, R., Vincze, M., Sukkarieh, S.: Viewpoint evaluation for online 3-D active object classification. IEEE Robot. Autom. Lett. 1(1), 73–81 (2016)
18. Patten, T., Kassir, A., Martens, W., Douillard, B., Fitch, R., Sukkarieh, S.: A Bayesian approach for time-constrained 3D outdoor object recognition. In: ICRA 2015 Workshop on Scaling Up Active Perception (2015)
19. Rezek, I., Leslie, D.S., Reece, S., Roberts, S.J., Rogers, A., Dash, R.K., Jennings, N.R.: On similarities between inference in game theory and machine learning. J. Artif. Intell. Res. 33, 259–283 (2008)
20. Silver, D., Veness, J.: Monte-Carlo planning in large POMDPs. In: Lafferty, J.D., Williams, C.K.I., Shawe-Taylor, J., Zemel, R.S., Culotta, A. (eds.) Advances in Neural Information Processing Systems 23, pp. 2164–2172. Curran Inc. (2010)
21. Singh, A., Krause, A., Guestrin, C., Kaiser, W.J.: Efficient informative sensing using multiple robots. J. Artif. Int. Res. 34(1), 707 – 755 (2009)
22. Wolpert, D.H., Bieniawski, S.: Distributed control by Lagrangian steepest descent. In: Proc. of IEEE CDC. pp. 1562–1567 (2004)
23. Wolpert, D.H., Bieniawski, S.R., Rajnarayan, D.G.: Handbook of Statistics 31: Machine Learning: Theory and Applications, chap. Probability collectives in optimization, pp. 61 – 99. Elsevier (2013)
24. Wolpert, D.H., Strauss, C.E.M., Rajnarayan, D.: Advances in distributed optimization using probability collectives. Adv. Complex Syst. 09(04), 383–436 (2006)
25. Xu, Z., Fitch, R., Underwood, J., Sukkarieh, S.: Decentralized coordinated tracking with mixed discrete-continuous decisions. J. Field Robot. 30(5), 717 – 740 (2013)
26. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Constructing free-energy approximations and generalized belief propagation algorithms. IEEE Trans. Inf. Theor. 51(7), 2282–2312 (2005)