Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Right Place, Right Time: Proactive Multi-Robot Task Allocation Under Spatiotemporal Uncertainty

Charlie Street, Bruno Lacerda, Manuel Mühlig, Nick Hawes

2024

Preprint:

This is an accepted article published in Journal of Artificial Intelligence Research. The final authenticated version is available online at: https://doi.org/10.1613/jair.1.15057

Journal of Artificial Intelligence Research 79 (2024) 137-171 Submitted 06/2023; published 01/2024

Right Place, Right Time: Proactive Multi-Robot Task Allocation Under Spatiotemporal Uncertainty

Charlie Street C.L.STREET@BHAM.AC.UK
School of Computer Science, University of Birmingham, UK

Bruno Lacerda BRUNO@ROBOTS.OX.AC.UK
Oxford Robotics Institute, University of Oxford, UK

Manuel Mühlig MANUEL.MUEHLIG@HONDA-RI.DE
Honda Research Institute Europe GmbH, Offenbach, Germany

Nick Hawes NICKH@ROBOTS.OX.AC.UK
Oxford Robotics Institute, University of Oxford, UK

Abstract

For many multi-robot problems, tasks are announced during execution, where task announcement times and locations are *uncertain*. To synthesise multi-robot behaviour that is robust to early announcements and unexpected delays, multi-robot task allocation methods must explicitly model the stochastic processes that govern task announcement. In this paper, we model task announcement using continuous-time Markov chains which predict when and where tasks will be announced. We then present a task allocation framework which uses the continuous-time Markov chains to allocate tasks *proactively*, such that robots are near or at the task location upon its announcement. Our method seeks to minimise the *expected total waiting duration* for each task, i.e. the duration between task announcement and a robot beginning to service the task. Our framework can be applied to any multi-robot task allocation problem where robots complete spatiotemporal tasks which are announced stochastically. We demonstrate the efficacy of our approach in simulation, where we outperform baselines which do not allocate tasks proactively, or do not fully exploit our task announcement models.

1. Introduction

Many applications of multi-robot systems (MRSs) require explicit coordination between robots to achieve a global goal. For example, autonomous taxis should coordinate to ensure a vehicle always reaches a customer within a fixed time (Mariani et al., 2021). Multi-robot task allocation (MRTA) techniques facilitate global coordination by assigning tasks to robots (Korsah et al., 2013), and have been used for teams of mobile robots to efficiently fulfil orders in warehouses (Xue et al., 2019), assist in healthcare facilities (Das et al., 2015), and search for survivors in disaster zones (Su et al., 2016). Further, task allocation techniques have been applied to teams of humans in applications such as fruit picking (Harman & Sklar, 2022), warehouse fulfilment (Ganbold et al., 2020), and taxi driver assignment (Glaschenko et al., 2009). In this paper, we consider MRTA problems where mobile robots are assigned *spatiotemporal tasks* across an environment. In these problems, robots *service* tasks by

©2024 The Authors. Published by AI Access Foundation under Creative Commons Attribution License CC BY 4.0.



Figure 1: Robots operating alongside fruit pickers in a fruit field.

travelling to a location and executing the actions required by the task. We assume task locations are unknown a priori, and become known once the task is *announced* during execution. For these problems the announcement time and location for a task are often stochastic, due to disturbances from the environment or agents not under our control, such as order processing delays in a fulfilment centre. A typical example of this class of problems are systems where a team of robots support humans performing logistics tasks.

Example 1. Consider Fig. 1, where robots operate alongside human pickers in a fruit field (Das et al., 2018; Khan et al., 2020). Pickers fill up baskets with fruit as they traverse the field, and request a robot once their basket is full. The robot collects the basket and provides the picker with an empty one. The picker is unable to continue work until they receive the new basket. This can be formulated as an MRTA problem, where each picker's request is a new task, and announcement occurs upon the picker making the request. Prior to announcement, the time and location of the picker's request is uncertain, due to factors such as human work rate and the yield of fruit in the field. The time and location of the request are also coupled, since a long announcement time suggests a picker has travelled further before requesting a robot.

Existing MRTA solutions either assume that all task announcements are known *a priori* (Nunes et al., 2017a), which is often unrealistic, or reactively allocate tasks after they are announced (Cordeau & Laporte, 2007; Choudhury et al., 2021). However, reactive allocation can yield long *waiting durations*, i.e. large gaps between a task's announcement and a robot beginning to service the task. To reduce waiting durations, tasks should be allocated *proactively*, such that robots can navigate towards *potential task locations* prior to announcement. To achieve this, we must explicitly model the spatiotemporal processes that govern task announcement, e.g. for Example 1 we must model the progress of the fruit pickers.

In this paper, we present an MRTA framework for tasks with stochastic announcements that seeks to minimise the *expected total waiting duration* across all tasks. We assume that all tasks are known, and that every task is eventually announced. This assumption holds in Example 1, where at any time there is at most one basket collection task per

picker, and each picker's basket eventually fills up, which triggers task announcement. We model task announcement using continuous-time Markov chains (CTMCs), and apply model checking techniques (Kwiatkowska et al., 2007) to reason over the time and location tasks are announced at. CTMCs are particularly well-suited to model task announcements for several reasons. First, they can be fitted from empirical data. Furthermore, given enough data, they can approximate any announcement distribution arbitrarily well (Thummler et al., 2006). Finally, they allow for seamless coupling of the processes governing announcement time and announcement location, by assuming a CTMC state represents a location and using the transition rates to represent the passage of time.

We construct a CTMC for each task, and present a framework which extends sequential single-item (SSI) auctioning (Koenig et al., 2006) for proactive task allocation. In our framework, each potential task location is considered as an item to be auctioned separately. To handle the uncertainty over where tasks will be announced, robots wait at *intermediate waiting points (IWPs)* which are near the potential task locations they have been assigned to. IWPs are computed such that the expected waiting duration for task servicing is minimised. This is done using the task announcement location distributions obtained from the CTMCs. In addition, we re-auction tasks online, synthesising solutions that can efficiently react to task updates and redistribute the robots to complete the tasks quickly. This online approach also enables us to effectively manage unexpected delays and early announcements.

Our approach assumes access to the announcement distribution for each task. We can accurately approximate the announcement models from empirical data, as robot tasks are often repetitive, and MRTA is repeated periodically. For example, fruit pickers in Example 1 complete multiple picking tasks a day, and work over multiple days. In warehouses, orders arrive frequently (De Koster et al., 2007), and order picking occurs every day. With this, we can collect data with or without our method, and then construct accurate announcement models which support proactive decision-making.

To the best of our knowledge, ours is the first proactive MRTA method which uses structured spatiotemporal models of task announcement. By exploiting these models during task allocation, we significantly reduce task waiting durations compared to reactive methods which do not model task announcement. Similar to existing work, we allocate tasks online, however we do this proactively using updates from our task models. Though we focus on SSI, our framework can be adapted to use other MRTA algorithms in a straightforward way. The primary contributions of this work are:

- A CTMC model for stochastic task announcement in continuous time.
- A framework that extends SSI auctioning for proactive task allocation.
- An empirical evaluation that demonstrates the efficacy of our approach in simulation.

Specifically, our experiments empirically show that our framework decreases waiting durations by 74.6% on average compared to a non-proactive method.

2. Related Work

In this section, we discuss existing techniques for auctioning, solving MRTA problems under uncertainty, and modelling stochastic processes with Markov chains.

2.1 Auctioning Solutions for MRTA

Auctions are a distributed MRTA method where robots bid the cost they can offer for each task to an auctioneer who determines the allocation (Gerkey & Mataric, 2002). Auctioning can be adapted to different objectives without adjusting the auction structure, as bid computation and winner determination are modular. Auctioning techniques trade off between optimality and scalability. For example, in combinatorial auctions, robots bid on task bundles representing possible schedules (Blumrosen & Nisan, 2007), and the auctioneer assigns the bundles with minimum combined cost. Combinatorial auctions are optimal but intractable, as the number of bundles increases exponentially with the number of tasks. In parallel auctions (Koenig et al., 2006), robots compute a bid for each task, which is assigned to the robot with the lowest bid. This is highly scalable but synthesises poor solutions, as spatial dependencies between tasks are ignored, i.e. parallel auctions ignore that two nearby tasks may be efficiently completed by the same robot. SSI auctions (Koenig et al., 2006) improve on this by running a multi-round auction, where one task is allocated per round, and the winning robot in each round recomputes its bids for the remaining tasks. Recomputing bids allows the bid to capture cost changes after a robot has been allocated a task, which may represent spatial dependencies. SSI is a scalable suboptimal MRTA solution method that has been widely used. For the sum of costs objective, the suboptimality has been shown to be at most a factor of two from the optimal (Koenig et al., 2006). We describe SSI further in Section 3.3.

SSI has been extended in many ways. To adapt SSI for online deployments, new tasks are inserted into a robot's existing schedule (Schoenig & Pagnucco, 2010). To improve allocations, Zheng et al. (2006) evaluate the future impact of assigning a task by increasing the SSI lookahead, at the cost of additional complexity. Further, Koenig et al. (2007) auction fixed size bundles, which brings SSI closer to combinatorial auctions. SSI with regret clearing assigns the task with maximal regret, i.e. the task with the largest gap between the two smallest bids, which improves performance (Zheng et al., 2008). These techniques could be applied to the SSI framework we present in this paper to improve performance. Further, Nunes and Gini (2015) model robot schedules as simple temporal networks to ensure tasks are completed within a time window. This has been extended for precedence constraints, where tasks with longer precedence chains have higher priority (Nunes et al., 2017b). SSI has also been applied to human-aware navigation problems, where robot bids are penalised for navigating through congested areas (Surma et al., 2021). In this paper, we extend SSI for proactive task allocation.

2.2 MRTA Under Uncertainty

To synthesise efficient allocations, MRTA methods must reason over uncertain robot dynamics and task announcements. Tasks can be reallocated to react to stochastic dynamics online. For example, Lippi and Marino (2021) reallocate tasks upon human behaviour changes, which allows for improved allocations using updated information. However, such methods ignore stochasticity, generating inefficient allocations as expectations of task execution diverge from what is observed during execution. Choudhury et al. (2021) improve on this by explicitly capturing the probability of robot failure during schedule evaluation in order to synthesise allocations which optimise the expected number of completed tasks.

However, robot failure is only one possible source of uncertainty. Markov decision processes (MDPs) model systems with non-deterministic action choice and uncertain action outcomes, and have been used for MRTA by Faruq et al. (2018) and Schillinger et al. (2018) to allocate temporal logic tasks. Faruq et al. (2018) build a team model, where robots select tasks and plan sequentially. Schillinger et al. (2018) decompose a global temporal logic specification into single-robot tasks which are allocated through an auction, where the synthesised allocation satisfies the global specification. Moreover, Schillinger et al. (2018) consider uncertain action durations during bidding. This is similar to our framework, where navigation action durations are continuous and stochastic. Capitan et al. (2013) represent each task as a partially observable MDP, which each robot solves to compute bids for a decentralised auction. To reduce the effects of duration uncertainty on task completion times, Prorok (2019) and Malencia et al. (2021) assign redundant robots to tasks, which increases the chance of a robot arriving quickly. However, in many MRTA problems, tasks outnumber robots, and so there is no redundancy to exploit.

Stochastic task announcement can be handled by allocating tasks as they are announced (Cordeau & Laporte, 2007; Choudhury et al., 2021). However, this poorly utilises the MRS, as robots are idle until tasks are announced. Pavone et al. (2009) and Bopardikar et al. (2014) assume tasks are announced uniformly across the environment according to a Poisson distribution. However, tasks are only allocated once they are announced. To reduce the waiting duration between announcement and a task being serviced, tasks should be allocated proactively, i.e. prior to announcement. Burns et al. (2012) and Claes et al. (2015) allow robots to anticipate new tasks during planning using task announcement distributions. Burns et al. (2012) sample a number of future task announcements from a known distribution and plan for each. The robot then selects the action with the highest expected performance over the possible futures. However, this approach only applies to a single robot. Claes et al. (2015) allow each robot to select tasks independently by predicting the tasks selected by other robots using approximations of their behaviour, where task locations are uniformly distributed. Burns et al. (2012) and Claes et al. (2015) both assume the announcement location is independent of the discrete time a task is announced. Tsao et al. (2018) address MRTA problems where the announcement probability at a given location changes over time. Announcement distributions are learned from historical data, and future announcements are sampled to support MRTA, similar to Burns et al. (2012). Though reallocation occurs as new tasks arrive, the discrete-time announcement distributions are never updated. This fails to capture problems such as Example 1, where the location distribution changes as the picker moves. In this paper, we construct structured models of stochastic, continuous-time task announcement which capture the joint spatiotemporal announcement process. We then use these models to proactively allocate tasks, reducing waiting durations. Further, we reallocate tasks online as we receive updates to our task models.

2.3 Modelling Stochastic Processes with Markov Chains

Markov chains are established models for stochastic processes. Discrete-time Markov chains have been used to model environmental hazards (Tihanyi et al., 2021), resource constraints (de Nijs et al., 2018), disease spread (Gómez et al., 2010), and human task performance (Costen et al., 2022). However, many real world processes operate in continuous

time. We can represent such processes as CTMCs, which can model any nonnegative continuous stochastic process to an arbitrary precision (Thummler et al., 2006). CTMCs have been used to model networks (Khayari et al., 2003; Meng et al., 2016), traffic (Jalel et al., 2020), chemical reaction networks (Anderson & Kurtz, 2011), hospital stays (Faddy & McClean, 1999), and robot policy execution under congestion (Street et al., 2020, 2022b). We use CTMCs to model task announcement, for example fruit pickers requesting robots, or orders arriving in a fulfilment centre.

3. Preliminaries

For a set X, let $\text{Dist}(X)$ denote the set of distributions over X, and X^* denote the set of sequences with elements of type X. For distribution P, $\mathbb{E}[P]$ denotes the expected value of P. A glossary of acronyms and symbols is provided at the end of the paper for easy lookup.

3.1 Topological Maps

In this paper, we represent the environment as a topological map. Topological maps simplify the environment by considering the relevant locations for tasks and navigation decisions, which robots navigate between.

Definition 3.1. A *topological map* is a tuple $\mathcal{T} = \langle V, E, \rho \rangle$, where V is a finite set of nodes representing locations in the environment; $E \subseteq V \times V$ is a set of directed edges which robots can travel on; and $\rho : E \to \text{Dist}(\mathbb{R}_{\geq 0})$ maps each edge to a duration distribution for navigating on that edge.

3.2 Continuous-Time Markov Chains (CTMCs)

A CTMC describes the continuous-time evolution of a system as a sequence of exponential distributions. In this paper, we use CTMCs to model continuous-time task announcement.

Definition 3.2. A *CTMC* (Kwiatkowska et al., 2007) is a tuple $\mathcal{Q} = \langle S, init, \Delta \rangle$, where S is a finite set of states, and $init : S \to [0, 1]$ gives the probability of a state being the initial state. $\Delta : S \times S \to \mathbb{R}_{>0}$ is an exponential transition function, i.e. $\Delta(s, s')$ is the rate between states s and s'.

The exponential transition between states s and s' is associated with a value $\Delta(s, s')$, which is the rate parameter of an exponential distribution associated with its duration. The probability that we transition to s' from s within time t is $1 - e^{-\Delta(s,s') \cdot t}$. The *exit rate* is the sum of the outgoing rates from s, i.e. $E(s) = \sum_{s'} \Delta(s, s')$. The probability of a transition firing from state s within time t is $1 - e^{-E(s) \cdot t}$, and the probability of branching to state s' from state s is $\Delta(s, s')/E(s)$. The exit rate $E(s)$, transition rates $\Delta(s, s')$, and branching probabilities at a state s are independent of the sojourn time spent in s. This is due to the memoryless property of the exponential distribution, which states that the expected time until a transition fires is independent of the time that has already passed.

3.3 Sequential Single-Item (SSI) Auctioning

MRTA solutions assign a set of tasks $T = \{\tau^1, ..., \tau^m\}$ to a set of robots $R = \{r_1, ..., r_n\}$ to minimise cost.

Definition 3.3. A *task* $\tau \in T$ is announced at a topological node $v \in V$. To complete a task, a robot must navigate to the task location and service it, i.e. execute the task's required actions. Tasks cannot be serviced before announcement.

SSI auctions (Koenig et al., 2006) allocate a task per round for efficient allocation while considering dependencies between tasks. SSI is traditionally run after the set of tasks is announced. The auctioneer begins by offering the tasks to all robots, and then conducts a multi-round auction, where each round allocates a single task. In each round, the auctioneer waits for robots to submit bids, which represent the increase in cost if that robot were allocated the task. Upon receiving the bids, the auctioneer determines the winner, i.e. the robot with the minimum bid across all tasks, and notifies the robots. For standard SSI, only the winner needs to recompute their bids in the next round, as the bid values remain the same for all other robots.

During SSI, each robot maintains a *schedule*, which is commonly an ordered sequence of allocated tasks. In each auction round, robots compute a bid for each unallocated task $\tau \in T$. To do this, robot r_i inserts τ into a cost-minimising position in its schedule, which can be found by solving a travelling salesperson problem, or using the cheapest insertion heuristic (Koenig et al., 2006). The cost of adding task τ into robot r_i's schedule is then evaluated by computing the cost of the schedule before and after τ is added. Each robot only submits its lowest bid to the auctioneer. This reduces the number of bids the auctioneer must minimise over without affecting solution quality. If a robot wins a task, it is added to its schedule for the next round of bidding.

Despite its handling of task dependencies, SSI is greedy and suboptimal. In each round, SSI allocates the task with the lowest bid, without considering how this affects the other unallocated tasks. However, SSI is sufficient for many applications, and is efficient enough to support online reauctioning for solution improvement.

4. Problem Formulation

In this section, we propose CTMCs as models for stochastic task announcement on a topological map, and define the problem of MRTA under spatiotemporal uncertainty.

4.1 Task Announcement Model

Task announcement may be uncertain in both space and time. We write $P_\tau^A \in \text{Dist}(\mathbb{R}_{\geq 0})$ to denote the continuous distribution over the duration until announcement for task τ, and $P_\tau^L \in \text{Dist}(V)$ to denote the discrete location distribution for task τ. For a task τ, we refer to the nodes $v \in V$ where $P_\tau^L(v) > 0$ as *potential task locations (PTLs)* for τ. In this subsection, we present *task CTMCs* as a model of stochastic task announcement in continuous time which allow us to evaluate the announcement time and location distributions. Task CTMCs model the joint spatiotemporal announcement process, allowing for dependencies between the announcement time and location. For example, in Example 1, a longer announcement time often corresponds to the picker travelling longer distances.

Definition 4.1. A *task CTMC* for a task τ is a tuple $\mathcal{Q}_\tau = \langle S_\tau, init_\tau, \Delta_\tau, S_\tau^f, Lab_\tau \rangle$, where S_τ, $init_\tau$, and Δ_τ form a CTMC; and $S_\tau^f \subseteq S_\tau$ is a non-empty set of absorbing states, i.e. $E(s) = 0, \forall s \in S_\tau^f$, where reaching $s \in S_\tau^f$ represents τ being announced. $Lab_\tau : S_\tau^f \to V$

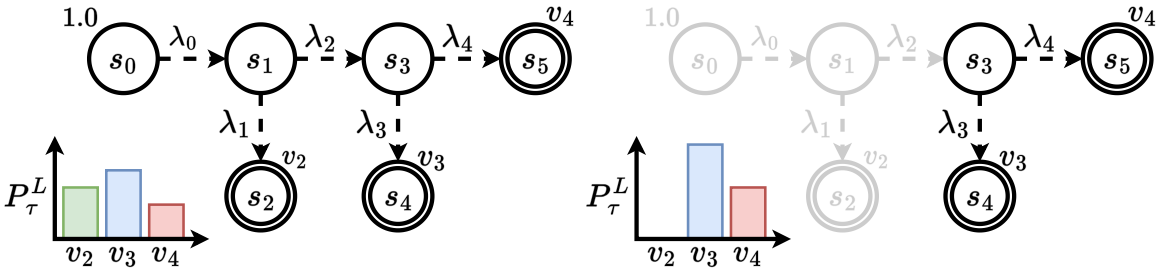
is a labelling function for absorbing states, where $Lab_\tau(s)$ is the physical announcement location for absorbing state $s \in S^f_\tau$.

Task CTMCs must have at least one absorbing state, and an absorbing state must eventually be reached with probability 1, i.e. the task must eventually be announced. Task CTMCs can model any nonnegative distribution to an arbitrary precision. Specifically, task CTMCs are phase-type distributions (PTDs) with a labelling function (Buchholz et al., 2014). PTDs model the time to reach an absorbing state in a CTMC, and we can fit task CTMCs from empirical data using techniques for PTDs (Thummler et al., 2006). For example, assume we have time and location data for multiple runs of a picker in Example 1. From the location data, we can build a discrete-time Markov chain (DTMC) which captures the picker's path, and the probability they request a robot at different locations. From the time data, we can build PTDs which represent the navigation duration for each topological edge the picker travels on. The PTDs can then be composed with the DTMC using techniques presented by Street et al. (2022b) to construct the final task CTMC. If limited data is available, the resulting task CTMCs may be inaccurate. However, if the MRTA problem is repeated periodically, such as in Example 1, the task CTMCs can be updated as new data is collected online.

For a task $\tau \in T$ represented as a task CTMC $\mathcal{Q}_\tau$, we can compute the announcement time and location distributions using standard model checking techniques (Kwiatkowska et al., 2007, 2011). The announcement time distribution P^A_τ can be computed through transient analysis (Baier et al., 2000). However, the worst-case time complexity of transient analysis is cubic in the number of CTMC states (Kwiatkowska et al., 2007; Pulungan & Hermanns, 2018), and transient analysis would be required at each point in continuous time. Therefore, we only consider the expected announcement time $\mathbb{E}[P^A_\tau]$, which can be computed as the expected time to reach an absorbing state. This can be achieved in linear time by propagating expected durations backwards from the absorbing states, similar to Bellman backups in acyclic MDPs (Mausam & Kolobov, 2012). To compute the discrete location distribution P^L_τ at a topological node $v \in V$, we use the branching probabilities in $\mathcal{Q}_\tau$ to compute the probability of reaching each absorbing state s, denoted $P_\tau(\mathit{reach}(s))$ (Kwiatkowska et al., 2007). We then sum the reachability probabilities based on their labelling, i.e. their topological location. Formally:

$$P^L_\tau(v) = \sum_{s \in \{s' \in S^f_\tau \mid Lab_\tau(s') = v\}} P_\tau(\mathit{reach}(s)). \tag{1}$$

The task CTMC branching probabilities are independent of the time spent in a CTMC state due to the memoryless property of the exponential distribution. With this, the location distribution P^L_τ is independent of the time since the task announcement process modelled by $\mathcal{Q}_\tau$ began. During execution, the current state of each task CTMC is updated when observations of the process modelled by the CTMC are received. These task CTMC state updates affect the task announcement time and location. We assume task CTMC states are fully observable. Observability is satisfied in many multi-robot problems, such as in Example 1, where human pickers carry global navigation satellite system trackers for robots to locate them (Khan et al., 2020).



(a) When the current state is s_0.

(b) When the current state is s_3.

Figure 2: A task CTMC $\mathcal{Q}_\tau$ and the corresponding location distribution P_τ^L when at different states in the model. Concentric circles represent absorbing states, λ values are exponential rates, and the value above each absorbing state is the corresponding PTL, defined by the labelling function.

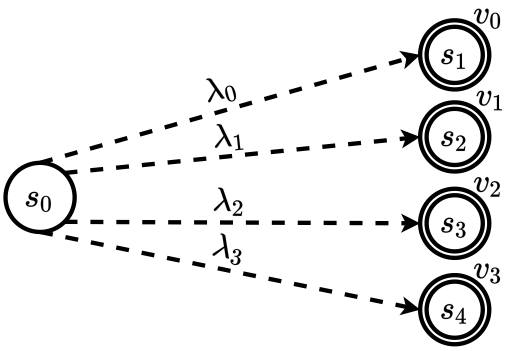


Figure 3: A task CTMC $\mathcal{Q}_\tau$ with a star topology, where concentric circles are absorbing states, λ values are exponential rates, and above each absorbing state is the corresponding PTL, as defined by the labelling function.

Example 2. Consider the task CTMC $\mathcal{Q}_\tau$ in Fig. 2, which models a fruit picker as per Example 1. The picker starts at s_0 (Fig. 2a) and will finish filling up their basket at location v_2, v_3, or v_4. If the picker reaches state s_3 (Fig. 2b), the announcement distributions P_τ^A and P_τ^L reflect that the duration before the picker requests a robot is reduced, and that v_2 can no longer be the task location.

Task CTMCs support proactive decision-making, as robots can navigate towards PTLs using predictions of when and where tasks will be announced. However, the structure of a task CTMC affects the extent to which robots can exploit these predictions. We now introduce three example task CTMC structures, which we use in Section 6 to assess the proactive MRTA framework we present in Section 5.

- *Contiguous sequence* CTMCs are structured as in Fig. 2. Here, the PTLs are contiguous on the topological map, i.e. there is a single edge connecting the first and second PTLs, as well as the second and third. This structure captures the behaviour of the fruit pickers in Example 1. The state of a contiguous sequence CTMC is updated multiple times prior to announcement. State updates refine the announcement

distributions as announcement gets nearer. This allows robots acting proactively to adjust their behaviour, which improves the chance of a robot being near the true task location upon announcement. Moreover, as the PTLs are contiguous, robots can navigate towards all PTLs simultaneously, which simplifies decision-making.

- *Discontiguous sequence* CTMCs are structured as in Fig. 2, where the PTLs are spread across the environment. Similar to contiguous sequence CTMCs, robots can use discontiguous sequence CTMC state updates to modify their behaviour as task announcement draws nearer. However, as the PTLs are discontiguous, decision-making becomes more difficult, as proactively navigating towards one PTL may move a robot further away from another.
- *Discontiguous star* CTMCs are structured as in Fig. 3, where the PTLs are spread across the environment, similar to discontiguous sequence CTMCs. Discontiguous star CTMCs do not receive intermediate state updates, transitioning from the initial state directly to an absorbing state. With this, robots must make a single proactive decision using the initial announcement distributions, with no new information available until announcement.

4.2 MRTA Under Spatiotemporal Uncertainty

We now formulate the MRTA problem tackled in this paper. Traditional SSI auctions allocate tasks as a whole, where robot schedules are ordered sequences of tasks (Koenig et al., 2006). For MRTA under spatiotemporal uncertainty, the PTLs for task τ may be far apart, where different robots are closer to different PTLs. Therefore, we allocate each PTL individually, which allows different robots to be responsible for completing a task dependent on the announcement location, i.e. the robot who is allocated the PTL corresponding to the announcement location completes the task.

Definition 4.2. For a set of tasks T, the items we allocate correspond to the set of *PTL-task pairs*, written as:

$$I = \{(v, \tau) \in V \times T \mid P_\tau^L(v) > 0\}, \tag{2}$$

where (v, τ) is announced at location v with probability $P_\tau^L(v)$ at a time distributed by P_τ^A. We associate the full announcement time distribution P_τ^A with each PTL-task pair as robots must wait for a task to be announced before deciding to service it at a PTL they have been allocated.

In Section 5, we allow multiple PTLs for a task to be allocated to the same robot. Since at most one of these PTLs can be the announcement location, we group them together in the robot's schedule, where each element of the schedule is a set of PTLs alongside the corresponding task.

Definition 4.3. The *schedule* for robot r_i is a (possibly empty) sequence $\phi_i = \phi_i^1 \phi_i^2 ... \phi_i^{k_i} \in (2^V \times T)^*$, where $\phi_i^j = (V_i^j, \tau)$, and $V_i^j \subseteq V$ is the set of PTLs robot r_i has been allocated for its jth task τ.

We denote the length of schedule ϕ_i as $|\phi_i| = k_i$. Throughout this paper, we use subscript i to denote robot r_i, and superscript j to denote the jth task τ in robot r_i's schedule. Robot

r_i is responsible for completing the jth task in its schedule τ if it is announced at one of the PTLs in V_i^j. For each $\phi_i^j = (V_i^j, \tau)$ and $v \in V_i^j$, $P_\tau^L(v) > 0$, i.e. all nodes in V_i^j are PTLs for task τ. Further, since PTLs are grouped together each task may appear in at most one position j in schedule ϕ_i. Next, we define the set of valid allocations over the task set, which specify the PTLs allocated to each robot, and the order each robot services the tasks they have been allocated PTLs for.

Definition 4.4. Let $R = \{r_1, .., r_n\}$ be a set of robots, $T = \{\tau^1, ..., \tau^m\}$ be a set of tasks, and I be the set of PTL-task pairs. An *allocation* is a mapping from robots to schedules $\Phi : R \to (2^V \times T)^*$ such that every PTL is allocated, i.e. $\bigcup_{i,j}\{(v, \tau) \mid v \in V_i^j \text{ and } (V_i^j, \tau) = \phi_i^j\} = I$.

In this paper, we aim to minimise *task waiting durations*, i.e. the duration between a task being announced and a robot beginning to service it at the task location. We refer to the time a robot begins servicing a task as the *service time*. For simplicity of presentation, we assume that tasks are serviced instantaneously upon a robot arriving to the task location. However, extending our method to durative tasks is straightforward (see Section 5.2). Waiting durations are stochastic, due to uncertain task announcement and robot action durations.

Definition 4.5. Let W_τ^Φ be the waiting duration distribution for task $\tau \in T$ under allocation Φ. The *expected waiting duration*, $\mathbb{E}[W_\tau^\Phi]$, is given by:

$$\mathbb{E}[W_\tau^\Phi] = \mathbb{E}[B_\tau^\Phi - P_\tau^A] = \mathbb{E}[B_\tau^\Phi] - \mathbb{E}[P_\tau^A], \tag{3}$$

where B_τ^Φ is the service time distribution for task τ under allocation Φ, and P_τ^A is the announcement time distribution for task τ.

We describe how to evaluate the service time distributions B_τ^Φ required to compute Eq. 3 in Section 5. Finally, we define the problem of MRTA under spatiotemporal uncertainty.

Problem 1. Given a team of n robots $R = \{r_1, ..., r_n\}$, and task set $T = \{\tau^1, ..., \tau^m\}$, synthesise an allocation Φ^* that minimises the total expected waiting duration, i.e.:

$$\Phi^* = \operatorname*{argmin}_\Phi \sum_{\tau \in T} \mathbb{E}[W_\tau^\Phi]. \tag{4}$$

For simplicity, Problem 1 considers a fixed set of tasks whose announcement model is available prior to execution. However, the methods we present in Section 5 can be adapted to problems where new tasks appear online (see Section 5.2). This occurs in Example 1, where fruit pickers begin to fill a new basket after their last one has been collected.

5. Proactive SSI Auctioning Under Spatiotemporal Uncertainty

In this section, we present an auctioning framework that solves Problem 1 by extending SSI for proactive task allocation using task CTMCs.

5.1 Auctioning Framework

In Fig. 4, we present our proactive task allocation framework. The input to our framework is a set of task CTMCs $\mathcal{Q}_\tau$ for $\tau \in T = \{\tau^1, \ldots, \tau^m\}$, and the set of PTL-task pairs I as defined in Def. 4.2. We proceed by describing the centralised components of our framework (see Section 5.1.1) and the behaviour of each robot (see Section 5.1.2).

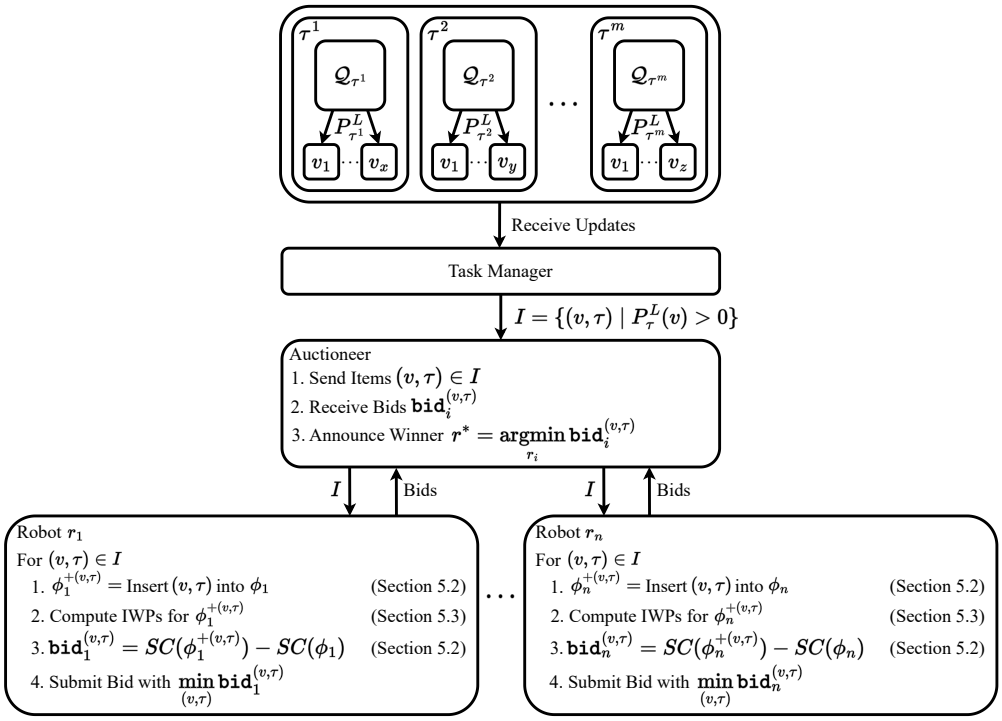


Figure 4: A framework that extends SSI for proactive task allocation. The steps for the auctioneer are as in SSI (Koenig et al., 2006). For the robots, each step which deviates from standard SSI is annotated with the subsection where that step is discussed.

5.1.1 Proactive Auctioning

For proactive auctioning, we use a *task manager*, which listens for task CTMC state updates and calls SSI auctions (Koenig et al., 2006) for sets of PTL-task pairs. Auctions are performed online and occur when:

- **Execution starts:** The task manager sends the full set of PTL-task pairs I to the auctioneer at the start of execution to synthesise an initial allocation.

- **A CTMC state updates:** When a state update is received from a task CTMC during execution, we re-auction the updated PTL-task pairs for the corresponding task. No other tasks are re-auctioned at this time. Task CTMC state updates may change a PTL's probability. If this probability drops to 0, the PTL is not re-auctioned, and is removed from the schedule of the robot who was previously allocated it. This prevents the robot from navigating to that location.

- **A task is announced:** When task $\tau \in T$ is announced, it becomes deterministic, i.e. the announcement distributions are updated such that $P^L_\tau(v) = 1$ for the announcement location v, and $P^A_\tau = 0$; (v,τ) is then re-auctioned for a final time, with the winning robot being allocated with the actual servicing of τ.

Online re-auctioning provides robustness to delays in robot navigation and task announcement. For example, if a task at the end of a robot's schedule is announced early, re-auctioning allows that robot, or another, to navigate to the task location immediately.

5.1.2 Proactive Robot Execution

We now discuss our framework from the perspective of a single robot r_i, who begins by computing bids for each PTL-task pair in the initial auction. The bid for PTL-task pair (v, τ) corresponds to the increase in r_i's schedule cost SC when inserting (v, τ) into schedule ϕ_i. To support proactive task allocation, SC is defined as the total expected waiting duration for the tasks in robot r_i's schedule. For this, we require the announcement time and location distributions, and knowledge of how r_i proactively executes tasks. We describe how to compute SC in Section 5.2. The announcement time affects where robot r_i inserts a task into its schedule. If task τ is likely to be announced far in the future, then τ can be placed late in r_i's schedule, as the waiting duration is zero until announcement. However, if τ is already announced it should appear near the start of r_i's schedule, as the waiting duration is already increasing. After the initial auction, robot r_i executes its schedule as follows:

1. **Get task:** Robot r_i removes the first task τ from its schedule ϕ_i alongside the PTLs it has been allocated for τ. Note that robot r_i executes only one task at a time.

2. **Navigate to waiting point (WP):** To execute task τ proactively, r_i first navigates to a *waiting point (WP)*. WPs may be chosen in different ways. For example, r_i could navigate to the most likely PTL, or wait at its initial location. In Section 5.3, we introduce *intermediate waiting points (IWPs)*, which are WPs that bring robots closer to the PTLs they have been allocated. IWPs minimise the expected waiting duration, and prevent costly backtracking manoeuvres which occur when a robot navigates to the wrong location. We compare different WPs in Section 6, and use WPs in Section 5.2 to compute the cost of robot schedules.

3. **Wait for announcement:** Robot r_i waits at the WP until τ is announced.

4. **Navigate to task location:** When task τ is announced, robot r_i observes the task location $v \in V$. If r_i was allocated v, it navigates there and services the task, before returning to step 1. Otherwise, the robot immediately returns to step 1 and begins execution for its next task.

Upon a task CTMC state update, robot r_i receives the updated PTL-task pairs for that task from the auctioneer and bids on them. If r_i loses the auction for a PTL-task pair it was assigned previously, the PTL is removed from r_i's schedule. If all PTLs for r_i's current task are removed, r_i moves onto its next task, whereas if at least one PTL is retained, r_i recomputes its current WP using the updated announcement distributions. Finally, if r_i is allocated a PTL-task pair for a task it has not previously been allocated any PTLs for, the PTL-task pair is inserted into r_i's schedule, as described in Section 5.2.

By auctioning each PTL-task pair separately, we allow multiple robots to be responsible for a single task, dependent on the announcement location. This allows for more flexible allocations, as if a task's PTLs are far apart, they can be allocated to different robots,

reducing the waiting duration. Note that, upon announcement, tasks are serviced by a single robot.

We proceed in Section 5.2 by outlining how robots compute bids in proactive auctions, which brings two challenges. First, we require a method for inserting PTL-task pairs into a robot's schedule, where each element is a task alongside the PTLs the robot has been allocated for that task. Second, we must evaluate the total expected waiting duration for the tasks in a robot's schedule. This is challenging as task locations are uncertain prior to announcement, and the service time is different for each PTL. To capture this, we compute the expected waiting duration for each PTL and weight it by its probability. Schedule evaluation also requires the robot WPs, and in Section 5.3 we describe how to compute IWPs, which are WPs that minimise the expected waiting duration.

5.2 Computing Bids

In this subsection, we describe how robots compute bids that consider expected waiting durations. To compute a bid for a PTL-task pair (v, τ), robot r_i performs the following (see Fig. 4):

1. **Insert into schedule:** Insert (v, τ) into the best position in schedule ϕ_i (see Section 5.2.1).

2. **Compute WPs:** Compute the WPs for each task in its updated schedule. In Section 5.3, we describe how to compute WPs which minimise waiting duration.

3. **Compute waiting duration increase:** Compute the increase in total expected waiting duration after inserting PTL-task pair (v, τ) into schedule ϕ_i. This is captured by computing the schedule cost $SC(\phi_i)$ before and after (v, τ) is added. The schedule cost SC is defined as the total expected waiting duration for schedule ϕ_i (see Section 5.2.2).

5.2.1 Schedule Insertion

Robots group the PTLs they have been allocated for the jth task in their schedule τ into a set $V_i^j \subseteq V$, where robot r_i only services task τ if it is announced at a location in V_i^j, as discussed in Section 4.2. To insert PTL-task pair (v, τ) into its schedule ϕ_i, robot r_i must do the following:

1. **Find task in schedule:** Find the pair (V_i^j, τ) associated with task τ in its schedule ϕ_i, and remove it from ϕ_i. If there are no PTL-task pairs associated with τ in ϕ_i, then $V_i^j = \emptyset$.

2. **Add new PTL-task pair:** Add v to V_i^j, i.e. $V_i^j = V_i^j \cup \{v\}$.

3. **Find best position in schedule:** Find the best insertion position for the updated (V_i^j, τ) in schedule ϕ_i. In Section 6, we use the cheapest insertion heuristic, which tests (V_i^j, τ) in each schedule position, and selects the position with the lowest cost (Koenig et al., 2006). The cost at each position is the total expected waiting duration of the updated schedule (see Section 5.2.2). The cheapest insertion heuristic is suboptimal,

as the schedule is fixed aside for the task being inserted. However, it scales well with schedule length, in contrast with finding the optimal insertion position which requires solving a travelling salesperson problem.

5.2.2 Evaluating the Expected Waiting Duration

Robots use the cost of their schedule to compute bids (see Fig. 4). Here, the schedule cost SC for robot r_i is the total expected waiting duration for the tasks in its schedule ϕ_i. This is similar to the total expected waiting duration for a full allocation Φ, as defined in Eq. 5, but considering only the PTLs present in ϕ_i.

Definition 5.1. Let $W_\tau^{\phi_i}$ and $B_\tau^{\phi_i}$ be the waiting duration and service time distributions for the PTLs robot r_i has been allocated for task τ. The *schedule cost* $SC(\phi_i)$ for robot r_i is defined as:

$$SC(\phi_i) = \sum_{(V_i^j,\tau)\in\phi_i} \mathbb{E}[W_\tau^{\phi_i}] = \sum_{(V_i^j,\tau)\in\phi_i} \mathbb{E}[B_\tau^{\phi_i}] - \sum_{(V_i^j,\tau)\in\phi_i} \mathbb{E}[P_\tau^A]. \tag{5}$$

The expected sum of announcement times in Eq. 5 is computed by summing the expected times for each task CTMC to reach an absorbing state (see Section 4.1). Thus, for the remainder of this subsection we describe how to compute the expected sum of service times $\sum_{(V_i^j,\tau)\in\phi_i} \mathbb{E}[B_\tau^{\phi_i}]$. Recall that to execute tasks proactively, robots navigate to a WP and wait for task announcement. If the task is announced at a PTL the robot has been allocated, it navigates to the task location from the WP and services it, otherwise it navigates from the WP to its next task. To compute the expected sum of service times, we first define the duration distribution for following the shortest path between any two topological nodes.

Definition 5.2. Let $\rho(e)$ be the duration distribution for navigating on topological edge $e \in E$. For topological nodes $v, v' \in V$, $\rho^+(v, v')$ is the *duration distribution for navigating along the shortest path between v and v'*, i.e. the sum of distributions $\rho(e)$ for all edges e along the shortest path. Here, the shortest path is computed using the expected navigation durations $\mathbb{E}[\rho(e)]$.

Next, we define the expected service time for a task τ given the start time and location for the robot servicing τ, its WP, and the true task location.

Definition 5.3. Let task τ be announced at node v', where τ is serviced by a robot who starts navigating towards WP v_w from node v at time t. The *service time* for task τ is given by $B_\tau(v, t, v_w, v')$, where:

$$B_\tau(v, t, v_w, v') = \mathbb{E}[\max(t + \rho^+(v, v_w), P_\tau^A)] + \mathbb{E}[\rho^+(v_w, v')]. \tag{6}$$

We visualise Eq. 6 in Fig. 5. The first term in Eq. 6 represents the robot navigating to the WP and waiting for the task to be announced, and the second captures navigation from the WP to the task location. By setting $v' = v_w$, Eq. 6 captures the time a robot stops waiting at the WP, as $\mathbb{E}[\rho^+(v_w, v')]$ becomes zero. If task τ is announced at a PTL the robot has not been allocated, this corresponds to the time the robot leaves the WP for its next task. In Eq. 6, P_τ^A is the announcement time distribution for the current task

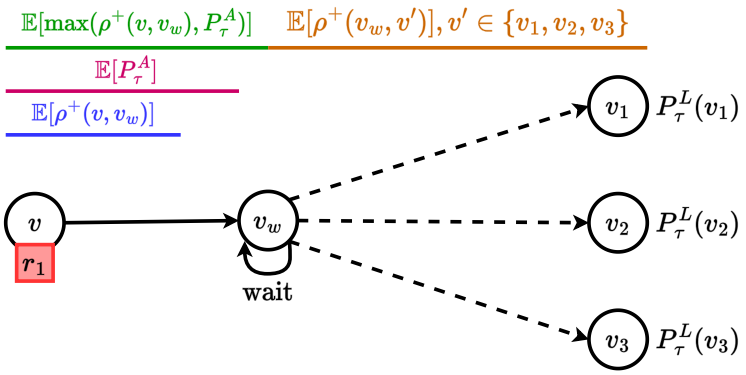


Figure 5: An example demonstrating how robots proactively execute tasks. Starting from node v at time 0, robot r_1 navigates to WP v_w and waits for task τ to be announced. PTLs for τ are v_1, v_2, and v_3. Upon announcement, r_1 navigates to the task location and services τ. Arrows between different nodes represent multiple topological edges. The coloured values show the terms in Eq. 6 and the lower bound we use in practice.

CTMC state. We do not consider the duration since the last task CTMC state update, as due to the memoryless property of exponential distributions, $\mathbb{E}[P_\tau^A]$ remains constant until the next state update. In practice, the expectation over the maximum in Eq. 6 is expensive to compute. For example, if the duration distribution $\rho^+(v, v_w)$ and announcement time distribution P_τ^A are represented as CTMCs, the maximum is defined over the joint CTMC, which is impractical to model check exactly due to its size. Therefore, we use the lower bound $\max(\mathbb{E}[t + \rho^+(v, v_w)], \mathbb{E}[P_\tau^A])$ (see Fig. 5).

By assuming a known task location, the service time B_τ for task τ ignores location uncertainty, as distributed by P_τ^L. To compute the expected sum of service times, we must account for this uncertainty. Recall that we assume tasks are serviced instantaneously at the task location. Therefore, from the perspective of robot r_i the task is finished at the announcement location if it is a PTL they were allocated, and the WP otherwise.

Definition 5.4. Let $v_{w,i}^j \in V$ be the WP for the jth task in r_i's schedule, and P_τ^L be the announcement location distribution for task τ. The location r_i finishes its jth task is given by the modified task location distribution $P_i^{L,j}$. In $P_i^{L,j}$, the probability mass for the PTLs r_i has not been allocated, i.e. the PTLs not in V_i^j, is assigned to the WP. Formally:

$$P_i^{L,j}(v) = \begin{cases} P_\tau^L(v) & \text{if } v \in V_i^j \text{ and } \phi_i^j = (V_i^j, \tau) \\ 1 - \sum_{v \in V_i^j} P_\tau^L(v) & \text{if } v = v_{w,i}^j \text{ and } \phi_i^j = (V_i^j, \tau) \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

Eq. 7 assumes that a task is never announced before r_i reaches the WP, which may occur during execution. We handle this by reallocating tasks upon announcement, which either removes the task from r_i's schedule, or sends r_i directly to the task location. We now use the location distribution in Eq. 7 to compute the expected sum of service times for r_i. The individual service times $B_\tau^{\phi_i}$ in Eq. 5 cannot be computed independently. This is because the service time for task τ' depends on the time and location r_i finishes its previous

task τ and begins proactive execution for τ' (see Eq. 6). Therefore, we introduce a recursive operator Ω_i^j for computing the sum of service times.

Definition 5.5. Assume robot r_i begins at node v at time t and executes the tasks in schedule ϕ_i, starting from the jth task τ, i.e. $\phi_i^j = (V_i^j, \tau)$. Further, let $v_{w,i}^j$ be the WP for the jth task τ. The *expected sum of service times* can be computed using $\Omega_i^j(v,t)$, which is defined as:

$$\Omega_i^j(v,t) = \begin{cases} \sum_{v'} P_i^{L,j}(v') \cdot (B_\tau(v,t,v_{w,i}^j,v') + \Omega_i^{j+1}(v', B_\tau(v,t,v_{w,i}^j,v'))) & \text{if } j \leq |\phi_i| \\ 0 & \text{if } j > |\phi_i|. \end{cases} \tag{8}$$

In Eq. 8, we compute the expected service time for the jth task, and add it to the sum of expected service times from the $j+1$th task. Robot r_i finishes its jth task τ at one of the PTLs it has been allocated or the WP, as defined by the modified task location distribution $P_i^{L,j}$, where each location has a different navigation duration from the WP $v_{w,i}^j$. Therefore, we compute the expected sum of service times for each node $v' \in V$ robot r_i may finish its jth task τ at, and weight it by $P_i^{L,j}(v')$. If r_i starts at node v at time 0, $\Omega_i^1(v,0)$ corresponds to the expected sum of service times for schedule ϕ_i. Thus, we rewrite the schedule cost $SC(\phi_i)$ in Eq. 5, replacing $\sum_{(V_i^j,\tau)\in\phi_i} \mathbb{E}[B_\tau^{\phi_i}]$ with $\Omega_i^1(v,0)$:

$$SC(\phi_i) = \Omega_i^1(v,0) - \sum_{(V_i^j,\tau)\in\phi_i} \mathbb{E}[P_\tau^A]. \tag{9}$$

If a tie occurs during bidding, we select the robot who reaches the WP for the task they are bidding on first. We can extend our framework to problems with durative tasks by modifying the recursive call in Eq. 8 to use the time and location the durative task is completed. Moreover, we can adapt our framework to problems where new tasks appear online. Assuming these tasks have a corresponding task CTMC, we can auction them proactively as soon as the task manager becomes aware of them.

5.3 Intermediate Waiting Points (IWPs)

In this subsection, we introduce IWPs, which are WPs that minimise the expected waiting duration for a task τ.

Definition 5.6. Let $\phi_i^j = (V_i^j, \tau)$ be the jth item in robot r_i's schedule. *IWPs* are defined according to $\texttt{IWP}_i^j : V \times \mathbb{R}_{\geq 0} \to V$, where $\texttt{IWP}_i^j(v,t)$ denotes the IWP for the PTLs V_i^j that have been allocated to r_i for task τ, given r_i starts at node v at time t:

$$\texttt{IWP}_i^j(v,t) = \operatorname*{argmin}_{v_w \in V} \sum_{v' \in V_i^j \cup \{v_w\}} P_i^{L,j}(v') \cdot B_\tau(v,t,v_w,v'). \tag{10}$$

The IWP is a WP that minimises the expected service time B_τ for task τ given the PTLs that have been allocated to r_i. Minimising the expected service time minimises the expected waiting duration, as the expected announcement time (see Eq. 3) is independent of the WP. IWPs also handle scenarios where r_i does not need to complete its jth task τ, as

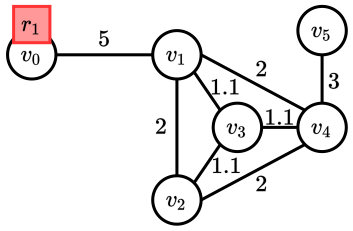


Figure 6: An example topological map, where edge weights are the expected duration of navigating that edge. Robot r_1 is initially located at node v_0.

when $v' = v_w$ in Eq. 10, $P_i^{L,j}(v')$ becomes the probability of τ being announced at a PTL that has not been allocated to r_i, and the service time $B_\tau(v, t, v_w, v')$ becomes the time r_i leaves v_w for its next task. If a tie occurs in Eq. 10, we select the WP v_w that minimises the expected duration from the WP to the task locations, i.e. $\sum_{v'} P_i^{L,j}(v') \cdot \mathbb{E}[\rho^+(v_w, v')]$. Eq. 10 is dependent on the time a robot starts a task, and so the IWP may change between bidding and execution, due to uncertain navigation durations and task announcement. Therefore, robots recompute IWPs just before starting a task to improve online performance. The IWP is also recomputed upon task CTMC state updates. Frequent updates which significantly affect the announcement distributions may cause a robot to travel long distances to reach the latest IWP. Though this reduces waiting durations, the robot will expend more energy, and require more frequent charging. This may limit multi-robot performance over longer horizons, which we will investigate in future work.

Example 3. Consider Fig. 6, where robot r_1 is at v_0 at time 0, and has been allocated all PTLs for its jth task τ, where the location distribution $P_\tau^L = \{v_1 : 0.4, v_2 : 0.3, v_4 : 0.3\}$, and the expected announcement time $\mathbb{E}[P_\tau^A] = 7$. The IWP is $\texttt{IWP}_1^j(v_0, 0) = v_3$, where the weighted sum of B_τ for v_3 in Eq. 10 is $\max(6.1, 7) + 1.1 = 8.1$, compared to v_1, which evaluates to $0.4 \cdot (\max(5, 7) + 0) + 0.6 \cdot (\max(5, 7) + 2) = 8.2$.

In Eq. 10, we minimise over every possible WP in the set of topological nodes V, which is expensive for large environments. To reduce complexity, we consider minimising over a smaller set of candidate nodes. Candidates are computed using a greedy heuristic which removes nodes unlikely to be the IWP for robot r_i's jth task τ. Starting from r_i's initial location, we perform a breadth-first traversal of the topological map. During the traversal, we maintain the candidate set and a list of nodes to be expanded, where we iteratively expand each node in the list. To expand topological node v, we do the following:

1. If v has already been expanded, we move to the next node in the list.
2. Next, we find the topological neighbours of v, $N_v = \{v' \in V \mid (v, v') \in E\}$.
3. We then compute the expected duration from node v and each $v' \in N_v$ to each of the PTLs for task τ.
4. We add node $v' \in N_v$ to the list of nodes to be expanded if its expected duration to at least one PTL is less than the expected duration from v.

5. We add node v to the candidate set if there are no neighbours $v' \in N_v$ whose expected duration to all PTLs is less than from v. If such a neighbour v' exists, v is not added to the candidate set, as navigating to v' allows for more progress to the PTLs prior to announcement.

This heuristic is suboptimal, and in some cases may remove the minimal WP from the candidate set, as neighbours are greedily expanded towards the task locations.

Example 4. Consider Fig. 6, where robot r_1 starts at v_0, and the PTLs are $\{v_1, v_2, v_4\}$. For v_0, the expected navigation durations are $\mathbb{E}[\rho^+(v_0, v_1)] = 5$, $\mathbb{E}[\rho^+(v_0, v_2)] = 7$, and $\mathbb{E}[\rho^+(v_0, v_4)] = 7$. The only neighbour of v_0 is v_1, where $\mathbb{E}[\rho^+(v_1, v_1)] = 0$, $\mathbb{E}[\rho^+(v_1, v_2)] = 2$, and $\mathbb{E}[\rho^+(v_1, v_4)] = 2$. The expected duration to all PTLs is lower from v_1, and so v_0 is not included in the candidate set. From v_1, v_2 and v_4 are closer to themselves, and v_3 is closer to v_2 and v_4. With this, node v_1 is included in the candidate set. Nodes v_2, v_3, and v_4 are then expanded, and added to the candidate set. From v_4, v_5 moves away from all PTLs. Therefore, the final candidate set is $\{v_1, v_2, v_3, v_4\}$.

5.4 Complexity Analysis

In this subsection, we outline the time complexity of the algorithms run on the auctioneer and robots under our framework. We assume that the announcement time and announcement location distributions are precomputed, and ignore online updates, i.e. we just consider the initial auction.

5.4.1 Proactive Task Allocation (Auctioneer)

In each auction round, the auctioneer allocates a single PTL-task pair by determining the minimum bid. The auctioneer receives n bids per round, as each robot only submits its smallest bid to the auctioneer (Koenig et al., 2006). There are $|I|$ auction rounds, one for each PTL-task pair. Minimisation is linear in the number of bids, and so the time complexity for the auctioneer to run the initial auction corresponds to the total number of bids, given by $|I| \cdot n$, which is linear in the number of robots and PTL-task pairs.

5.4.2 IWP Computation

During auctions, robots compute IWPs using Eq. 10, which finds the WP that minimises the waiting duration from a set of at most $|V|$ candidate nodes. We compute the expected service time for each candidate as a weighted sum over the PTLs. The time complexity for Eq. 10 is given by $\mathcal{O}(|V| \cdot |P^L_{max}|)$, where $|P^L_{max}|$ is the maximum number of PTLs for a single task.

5.4.3 Computing the Schedule Cost

Next, we consider the time complexity of calculating the schedule cost SC in Eq. 5, which corresponds to the complexity of the sum of service times in Eq. 8. Eq. 8 computes the IWP for the current task, given its start time and location, and makes multiple recursive calls for the next task, one for each of the current task's PTLs. The current task's announcement location affects the IWP for the next task. Therefore, we compute one IWP for the first

task, $|P^L_{\max}|$ for the second, $|P^L_{\max}|^2$ for the third, and so on. Generally, for the jth task τ in schedule ϕ_i, i.e. $\phi_i^j = (V_i^j, \tau)$, we compute $|P^L_{max}|^{j-1}$ IWPs. In total, we compute $\sum_{j=0}^{|\phi_i|-1} |P^L_{max}|^j$ IWPs, and so the time complexity is $\mathcal{O}\big(\frac{|P^L_{max}|^{|\phi_i|}-1}{|P^L_{max}|-1} \cdot |V| \cdot |P^L_{max}|\big)$, i.e. the schedule cost function SC scales exponentially in the length of the schedule. In practice, we cache IWPs across multiple SC calls to improve scalability. We validate the scalability of our method for calculating SC empirically in Section 6.

5.4.4 Proactive Task Allocation (per Robot)

Finally, we consider the time complexity for a robot r_i to participate in a proactive auction, assuming r_i uses the cheapest insertion heuristic to find the best schedule insertion position (Koenig et al., 2006). Robot r_i initially computes bids for each of the $|I|$ PTL-task pairs, and recomputes bids upon being allocated a PTL-task pair. In the worst case, r_i always wins, and recomputes $|I| - (\sum_j |V_i^j|)$ bids for each of the $|I|$ rounds, where a new PTL-task pair is added to schedule ϕ_i each round. Each bid requires at most $|\phi_i| + 1$ calls to the schedule cost function SC, one for each schedule insertion position. Over the auction, robot r_i makes $\sum_{k=1}^{|I|}(|I|+1-k)\cdot k$ calls to SC. The time complexity of SC is dependent on $|\phi_i|$, which after k auction rounds is upper bounded by $\min(k, |T|)$, i.e. the number of PTL-task pairs that have been allocated, or the number of tasks, whichever is smaller. Therefore, the worst case time complexity is $\mathcal{O}\big(\sum_{k=1}^{|I|}(|I|+1-k)\cdot k\cdot(\frac{|P^L_{max}|^{\min(k,|T|)}-1}{|P^L_{max}|-1}\cdot|V|\cdot|P^L_{max}|)\big)$, where the exponential term $|P^L_{max}|^{\min(k,|T|)}$ from SC dominates, demonstrating that computing the schedule cost is a bottleneck in our framework.

6. Experiments

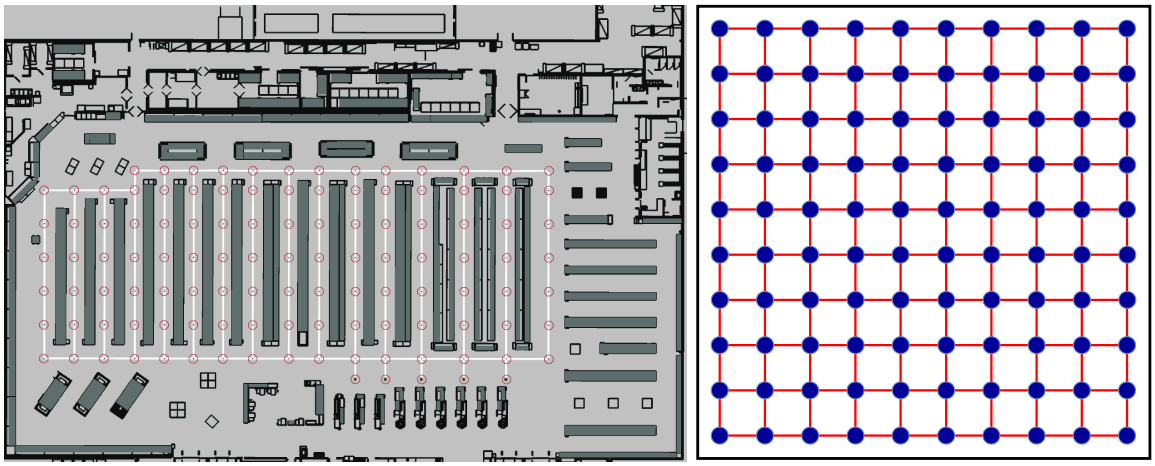
In this section, we demonstrate the efficacy of our framework across multiple problems in simulation by comparing against a number of baselines. We simulate robot behaviour using the context-aware multi-agent simulator (CAMAS), which captures the task-level behaviour of an MRS acting on a topological map (Street et al., 2022a). CAMAS samples through a multi-robot Markov automaton that uses continuous distributions over the duration of topological edges, and explicitly captures the effects of physical robot interactions such as congestion. All experiments are run on Ubuntu 18.04, with an Intel Core i9-10900K CPU@3.7GHz and 32GB of RAM. All software is written in Python, except for PRISM (Kwiatkowska et al., 2011), which we use for task CTMC analysis, which is written in Java/C++.

6.1 Experimental Setup

In this subsection, we describe our experimental environments, the methods we compare, and the experimental problems for each method.

6.1.1 Experimental Environments

We consider two environments: a supermarket (Fig. 7a) and a 4-connected 10×10 grid world (Fig. 7b). The supermarket topological map was created by hand. Nodes are evenly spaced within each aisle, and additional rows at the top and bottom connect the aisles. The topological nodes in the grid world are evenly spaced. Unlike the grid world, the aisles of the



(a) Supermarket. (b) 10×10 grid world.

Figure 7: The two experimental environments with topological maps overlaid.

supermarket constrain movement, making it harder to find effective IWPs. To construct a CAMAS simulation, we require navigation duration distributions for each topological edge (Street et al., 2022a). Duration distributions for the grid world are synthetic, and fitted using moment matching techniques (Marie, 1980). For the supermarket, we use a Gazebo simulation (Koenig & Howard, 2004) of the environment, and use edge partitioning to collect navigation duration data, as presented by Street et al. (2022a). To do so, we simulate five Clearpath Jackals navigating with Move Base Flex (Pütz et al., 2018) for 24 hours. Distributions are then fitted from this data using an algorithm presented by Thummler et al. (2006).

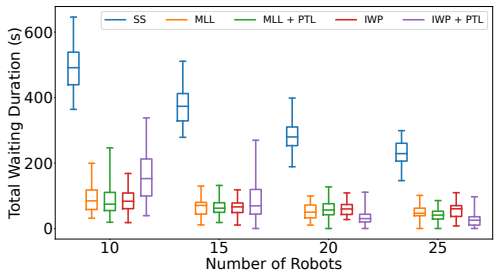
6.1.2 Experimental Methods/Baselines

The core components of our framework are: proactive allocation; IWPs; and auctioning each PTL-task pair separately. We compare five methods, which use some or all of these components:

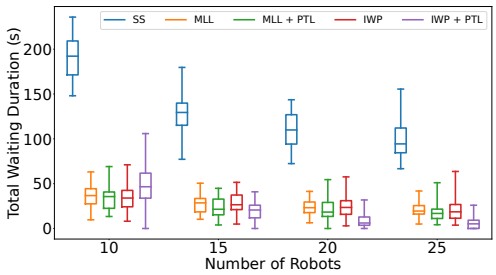
- *Stay Still (SS)*: A non-proactive MRTA technique, where each task is auctioned individually upon its announcement. As robots are not allocated tasks until they are announced, they wait at their initial location. This is representative of the state-of-the-art approaches which adapt SSI for online task announcement (Koenig et al., 2006; Schoenig & Pagnucco, 2010; Nunes & Gini, 2015; Schneider et al., 2015).

- *Most Likely Location (MLL)*: During proactive execution, robots choose the most likely PTL as their WP for a task, instead of the IWP. During bidding, robots assume the MLL occurs with probability 1. Tasks are auctioned as a whole, i.e. all PTLs for task $\tau \in T$ are assigned to a single robot who is responsible for servicing the task irrespective of its announcement location.

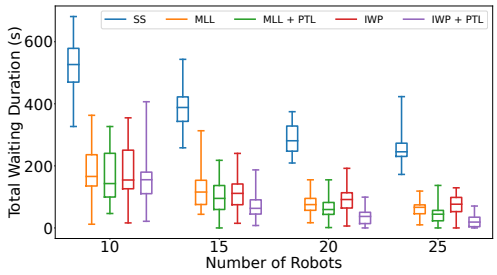
- *MLL+PTL*: The MLL method, but each PTL is auctioned separately. For each task, a robot's WP is the most likely PTL they have been allocated.



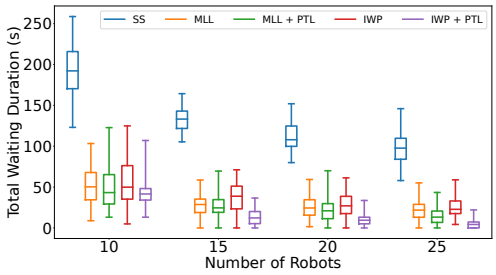
(a) Supermarket, contiguous sequence.



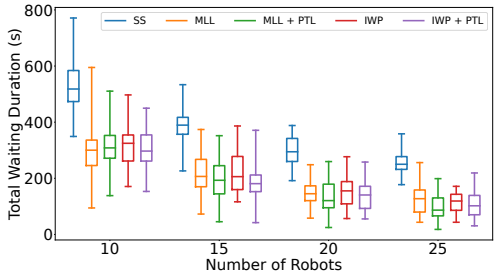
(b) 10×10 Grid, contiguous sequence.



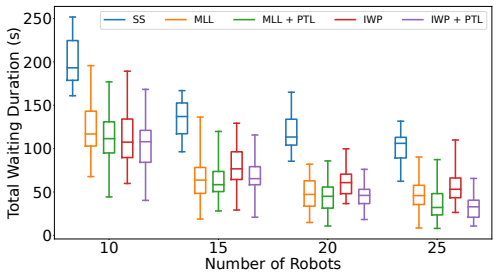
(c) Supermarket, discontiguous sequence.



(d) 10×10 Grid, discontiguous sequence.



(e) Supermarket, discontiguous star.



(f) 10×10 Grid, discontiguous star.

Figure 8: The performance of each method as we increase the team size.

- *IWP*: Robots wait for announcement at the IWP for each task (see Section 5.3), but tasks are auctioned as a whole, as with the MLL method.
- *IWP+PTL*: Our complete framework, where robots wait at IWPs for task announcement, and each PTL is auctioned separately.

6.1.3 Experimental Problem Setup

In all problems, a team of robots must service a set of tasks whose announcement is modelled with task CTMCs. The initial robot locations are spread as evenly as possible across the environment, and tasks are serviced instantaneously upon a robot reaching the task location. Robots navigate using a shortest path planner. During bidding, robots use the cheapest insertion heuristic for schedule insertion (Koenig et al., 2006). We consider prob-

Table 1: The mean waiting durations for the results in Fig. 8. The problem column is written <environment>-<CTMC structure>-<number of robots>, where Sup denotes the supermarket, and Grid denotes the 10×10 grid world. Bold values have the lowest mean for that problem. Highlighted cells are not statistically significantly lower than the method with lowest mean, according to a one-sided Mann-Whitney U test with $p = 0.05$.

Problem	SS	MLL	MLL+PTL	IWP	IWP+PTL
Sup-contiguous sequence-10	492.10	91.61	87.31	**83.40**	157.37
Sup-contiguous sequence-15	376.12	65.20	65.37	**64.32**	84.02
Sup-contiguous sequence-20	282.13	52.53	60.91	60.77	**37.55**
Sup-contiguous sequence-25	230.00	50.88	42.47	54.68	**26.43**
Sup-discontiguous sequence-10	522.58	179.21	163.17	172.97	**152.59**
Sup-discontiguous sequence-15	379.45	123.29	100.56	113.85	**71.06**
Sup-discontiguous sequence-20	286.28	76.58	62.27	92.09	**37.39**
Sup-discontiguous sequence-25	252.59	61.64	43.03	72.50	**22.56**
Sup-discontiguous star-10	532.91	**296.74**	311.00	310.03	305.23
Sup-discontiguous star-15	388.00	211.94	193.77	221.02	**190.41**
Sup-discontiguous star-20	295.21	148.08	133.36	157.77	**132.85**
Sup-discontiguous star-25	254.63	124.24	**97.42**	114.22	104.80
Grid-contiguous sequence-10	190.59	36.43	34.73	**34.29**	48.60
Grid-contiguous sequence-15	126.99	27.35	24.11	28.48	**19.54**
Grid-contiguous sequence-20	109.99	23.18	21.37	24.70	**8.35**
Grid-contiguous sequence-25	98.48	20.96	17.62	20.36	**6.83**
Grid-discontiguous sequence-10	191.16	52.20	49.06	55.48	**43.17**
Grid-discontiguous sequence-15	132.86	28.08	27.42	37.29	**13.93**
Grid-discontiguous sequence-20	111.57	26.51	22.68	28.38	**10.51**
Grid-discontiguous sequence-25	99.68	22.33	14.08	25.04	**5.15**
Grid-discontiguous star-10	199.34	119.40	110.30	113.95	**105.60**
Grid-discontiguous star-15	134.35	64.15	**63.21**	79.20	68.47
Grid-discontiguous star-20	118.84	49.14	**44.62**	62.75	45.37
Grid-discontiguous star-25	102.51	47.98	35.59	55.09	**32.34**

lems which use the task CTMC structures in Section 4.1, where CTMC transitions are sampled during execution, and all task CTMCs begin in their initial states at the start of execution. For each task CTMC type, we construct problems with 5-20 tasks in increments of 5, where the 10 task problem is the 5 task problem with 5 additional tasks etc. Contiguous and discontiguous sequence CTMCs have three PTLs, as in Fig. 2, and discontiguous star CTMCs have four PTLs, as in Fig. 3. For a given number of tasks, we construct a different problem for each repeat. The expected announcement times, PTLs, and announcement location distributions for each problem are randomised, but kept consistent between methods. PTLs for contiguous sequence CTMCs are constrained to be contiguous across the topological map.

6.2 Measuring Performance as Robots Increase

First, we analyse performance as we increase the team size. In each environment, we consider the 20 task problem for each task CTMC type, and run 40 repeats to measure the total waiting duration for 10-25 robot teams, in increments of 5. In Fig. 8 we present the observed waiting durations, and in Table 1 we show the mean waiting durations alongside the results of a one-sided Mann-Whitney U statistical significance test, where $p = 0.05$.

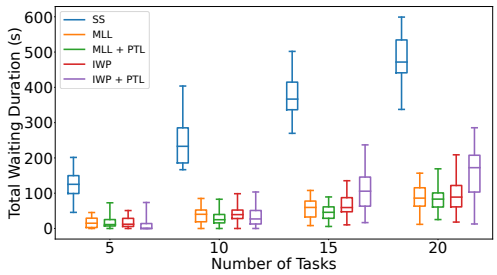
Overview of results. *In* 10/24 *of the tested problem configurations, IWP+PTL produces significantly lower waiting durations than all other methods.* In 12 problems, IWP+PTL ties with other methods, i.e. no method is significantly better than IWP+PTL, but some are statistically similar. In the smallest contiguous sequence problems in both environments, the MLL, MLL+PTL, and IWP waiting durations are significantly less than IWP+PTL, which we discuss below.

SS performance. The SS approach is representative of state-of-the-art techniques for online SSI. *However, in all problems, the SS waiting durations are significantly higher than every other method.* As tasks are not allocated until announcement in SS, robots remain idle at their initial location, which poorly utilises the MRS.

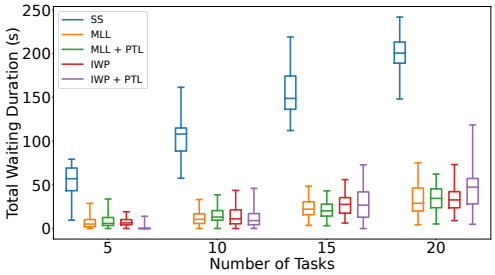
Trends as the number of robots increase. For larger teams, there is more likely to be a robot near the PTLs, decreasing the waiting duration. *Further, the gap between methods which auction PTLs separately and those that don't increases with the team size, as PTLs are split between robots to reduce the waiting duration.* Auctioning PTLs separately utilises robot redundancy, such as when we have more robots than tasks. Moreover, it extends the notion of spatial dependencies in SSI (Koenig et al., 2006), as if PTLs for different tasks are close to each other, they are likely to be allocated to the same robot.

Comparing IWP against MLL. *IWP+PTL decreases waiting durations by reasoning over the full task location distribution, and splitting PTLs between robots.* However, in many problems the related IWP method has a higher mean waiting duration than the simpler MLL-based methods. Under IWP, robots wait between the PTLs, as unlike the MLL method it reasons over the full location distribution at the current task CTMC state. In the contiguous and discontiguous sequence problems, task CTMC updates allow robots to adjust their WP during execution. For example, assume the first PTL is the most likely. We discover if this PTL is the announcement location upon the second state update. When this update occurs, robots using MLL will either incur a waiting duration of zero, or have time to move to one of the remaining PTLs. For IWP, the robots will likely have to move regardless of the announcement location, increasing the waiting duration. This problem occurs as we only consider the announcement distributions at the current task CTMC state when computing IWPs. Task CTMC state updates modify the announcement distributions, where some PTLs are announced earlier in the CTMC. By ignoring these updates, robots assume that the announcement time for each PTL is the same, which can cause robots to wait at suboptimal WPs. To mitigate this, we could compute IWPs by model checking a joint CTMC that captures the task CTMC and the robot's route, however this would inhibit the scalability of our framework.

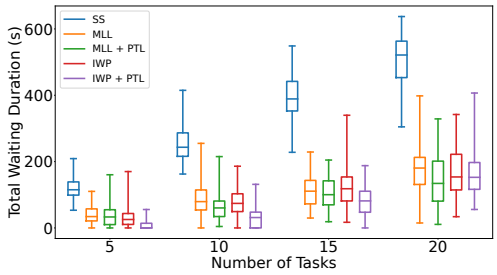
Comparing IWP against IWP+PTL. IWP+PTL statistically significantly outperforms IWP on 18/24 problems. *However, as the team size decreases, IWP+PTL assigns more*



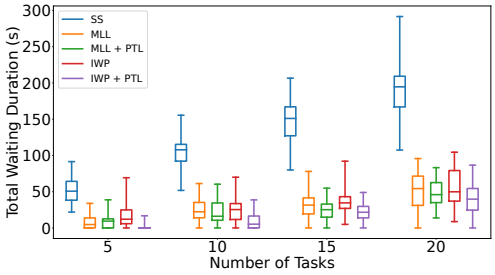
(a) Supermarket, contiguous sequence.



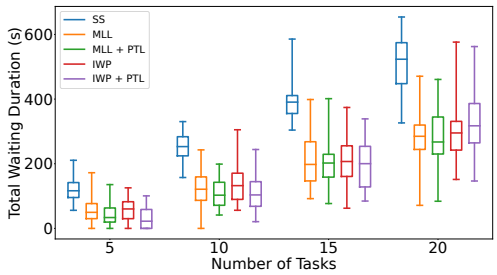
(b) 10×10 Grid, contiguous sequence.



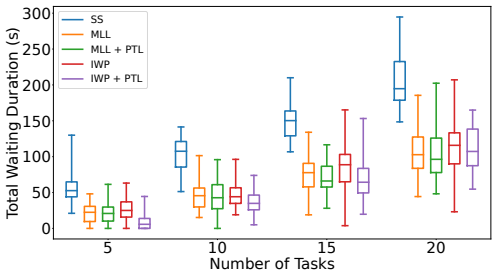
(c) Supermarket, discontiguous sequence.



(d) 10×10 Grid, discontiguous sequence.



(e) Supermarket, discontiguous star.



(f) 10×10 Grid, discontiguous star.

Figure 9: The performance of each method as we increase the number of tasks.

PTLs to a single-robot, bringing performance closer to the IWP method. In fact, for the 10 robot contiguous sequence problems, IWP outperforms IWP+PTL. Under IWP+PTL, multiple robots may be assigned different PTLs for the same task. With this, at least one robot will wait at a WP to then not service the task. This unnecessary wait may place the robot further from its next task, increasing the waiting duration. Under IWP, a robot waiting for a task will service it unless it is reallocated online. This problem is reduced as we add more robots, as on average robots will have to travel less between tasks, reducing the impact of unnecessary waiting.

Performance of the MLL-based methods. The MLL-based methods perform competitively in many problems, and are less computationally intensive, as we assume the MLL is the announcement location. *However, robots risk backtracking if the MLL is not the announcement location, which increases the waiting duration.* In the contiguous sequence

Table 2: The mean waiting durations for the results in Fig. 9. The table is structured as in Table 1, except for the number in the problem column, which refers to the number of tasks.

Problem	SS	MLL	MLL+PTL	IWP	IWP+PTL
Sup-contiguous sequence-5	124.78	16.56	16.64	16.23	**8.33**
Sup-contiguous sequence-10	242.45	37.18	**29.35**	40.96	34.26
Sup-contiguous sequence-15	373.84	57.73	**45.53**	64.51	107.27
Sup-contiguous sequence-20	484.10	87.30	**84.70**	90.89	159.08
Sup-discontiguous sequence-5	121.30	40.14	36.96	38.39	**8.59**
Sup-discontiguous sequence-10	251.18	87.79	61.18	77.77	**33.29**
Sup-discontiguous sequence-15	393.72	113.51	105.25	127.55	**84.52**
Sup-discontiguous sequence-20	509.11	178.26	**143.80**	167.04	167.91
Sup-discontiguous star-5	118.80	55.50	44.10	57.47	**32.05**
Sup-discontiguous star-10	250.17	127.77	**108.28**	138.49	114.16
Sup-discontiguous star-15	388.79	211.16	**199.19**	212.87	199.56
Sup-discontiguous star-20	513.09	286.47	**276.51**	306.26	325.05
Grid-contiguous sequence-5	55.14	6.55	8.60	7.09	**1.79**
Grid-contiguous sequence-10	104.72	12.02	15.05	14.06	**11.79**
Grid-contiguous sequence-15	153.31	23.75	**21.03**	27.90	28.39
Grid-contiguous sequence-20	198.89	**33.42**	33.97	34.28	46.34
Grid-discontiguous sequence-5	52.91	7.75	8.74	16.58	**1.96**
Grid-discontiguous sequence-10	103.80	24.18	21.65	26.63	**9.41**
Grid-discontiguous sequence-15	146.13	32.92	24.85	35.81	**22.11**
Grid-discontiguous sequence-20	189.62	52.31	47.19	55.13	**38.68**
Grid-discontiguous star-5	55.78	21.80	22.02	26.97	**8.44**
Grid-discontiguous star-10	105.44	45.91	46.22	46.54	**36.29**
Grid-discontiguous star-15	149.20	75.45	71.32	83.42	**69.19**
Grid-discontiguous star-20	205.88	**104.25**	106.74	113.76	110.91

problems, the effects of backtracking are significantly reduced as the PTLs are contiguous, and so MLL and MLL+PTL perform well. The MLL+PTL method often performs worse than IWP+PTL. If a robot using MLL+PTL has been assigned the MLL, all other PTLs for that task can be added to the schedule with zero cost, as robots assume the MLL is the announcement location. Therefore, the MLL+PTL robot will be assigned all PTLs for a task, causing behaviour similar to MLL. IWP+PTL avoids this by reasoning over the full location distribution.

Performance on discontiguous star problems. In the discontiguous star problems, the proactive methods perform similarly, i.e. there is always a tie for the best method. This is likely due to a lack of structure in the task CTMCs. In the contiguous sequence problems, task locations are contiguous, and the location distribution is updated over time, which allows for allocations to be improved online. Though the discontiguous sequence problems lose the spatial structure, the location distribution still receives updates, which we can exploit.

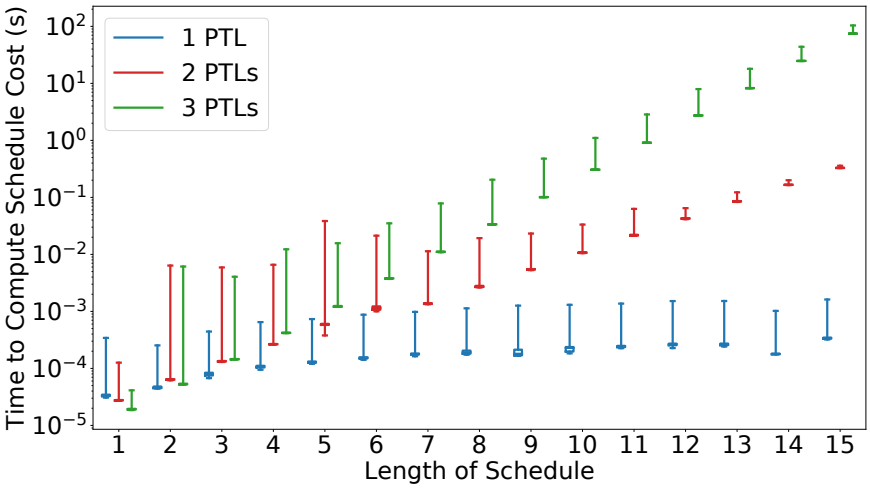


Figure 10: The scalability of the schedule cost function SC.

Conversely, the discontiguous star problems are one-shot problems, i.e. there are no updates, and no relationship between the task locations, which makes proactive reasoning less suitable.

6.3 Measuring Performance as Tasks Increase

Next, we consider how the number of tasks affects performance. For a 10 robot team, we consider the 5-20 task problems for each task CTMC type, and run 40 repeats to measure the total waiting duration. The observed waiting durations are shown in Fig. 9, and the mean waiting durations with statistical significance results are presented in Table 2, as in Table 1.

IWP+PTL produces significantly lower waiting durations in 10/24 *problems, ties in* 10/24*, and is beaten in* 4/24*.* Generally, the trends observed when increasing the tasks corroborate what we saw in the previous experiment. IWP+PTL performance decreases as the tasks increase, as there is less redundancy in the MRS, and less opportunity to distribute PTLs across the robots. Moreover, the MLL-based methods outperform the IWP-based methods on some problems, as the online state updates allow MLL-based methods time to react if the MLL is not the task location, and to serve tasks with zero waiting duration if it is.

6.4 Schedule Cost Scalability

In Fig. 10, we demonstrate the exponential scalability of computing the schedule cost SC, as discussed in Section 5.4. Computing the cost of robot schedules is a bottleneck in our framework. We run 40 repeats of SC for schedule lengths ranging from 1-15, and for tasks with 1-3 PTLs, where schedules contain all PTLs for a task. These results also act as a proxy for comparing the scalability of IWP and MLL-based methods, where the core difference is in the schedule cost function. IWP-based methods vary in scalability dependent on the PTLs per task, whereas MLL-based methods consider only one PTL, the MLL. This

significantly improves scalability, but at the cost of higher waiting durations, as discussed earlier in this section.

7. Discussion

Proactive vs non-proactive MRTA. In the previous section, we demonstrated that proactive task allocation decreases waiting durations. We compared against a non-proactive baseline which adapts SSI auctioning to online settings. SSI is suboptimal, and is at most a factor of two from the optimal for sum of cost objectives (Koenig et al., 2006). The performance of this baseline could be improved by using an optimal MRTA method such as combinatorial auctions (Blumrosen & Nisan, 2007). However, this is still unlikely to outperform proactive methods. For example, the SS waiting durations in Section 6 are often more than two times higher than the best proactive method. Moreover, our proactive framework can be adapted to use less suboptimal MRTA methods in a straightforward way. To effectively allocate tasks proactively we require accurate announcement models, which can be constructed from empirical data. If announcement data is not available, non-proactive methods may be a suitable alternative.

Improving scalability. Across both experiments in Section 6, IWP+PTL is significantly better in 41.7% of problems, and is unbeaten in 87.5%. However, the IWP-based methods are less scalable, as they reason over the full task location distribution during bidding (see Section 6.4). The MLL-based methods approximate the IWP-based methods by treating the MLL as the only PTL during bidding. As a result, the MLL-based methods are much faster, as schedule evaluation scales linearly in the number of tasks rather than exponentially. In 58.3% of the problems we evaluated, at least one of the MLL baselines is statistically similar to the method with lowest waiting durations, and in 12.5% of problems, at least one of the MLL baselines statistically significantly outperforms the IWP+PTL method. Therefore, if there are many tasks, or if computation time is limited, MLL-based methods can be used to effectively allocate tasks proactively. MLL-based methods also perform well when the PTLs are close together, as navigating towards the MLL brings the robot nearer to all PTLs, reducing waiting durations. The MLL- and IWP-based methods can be viewed as two ends of a spectrum of proactive MRTA methods which consider one and all of the PTLs for a task respectively. With this, the trade-off between performance and scalability can be controlled more precisely by selecting the number of PTLs to consider during bidding, e.g. the two most likely PTLs. An alternative approach to improve scalability would be to impose a maximum schedule length, or limit how many tasks are allocated in a given time frame. For example, if a task is due to be announced in an hour, it is unlikely that proactive allocation needs to occur straight away. This would not avoid the exponential scalability of schedule evaluation, but would provide a practical mitigation.

The impact of task CTMC structure on performance. The performance of our framework is sensitive to the structure embedded in the task CTMCs, which we exploit during allocation. In Section 6, we considered contiguous sequence CTMCs, which have spatial and temporal structure; discontiguous sequence CTMCs, which have temporal structure; and discontiguous star CTMCs, which are unstructured. If there is spatial structure, e.g. the PTLs are close together, then it is easier to find effective IWPs. If there is temporal

structure, i.e. if there are multiple task CTMC state updates before announcement, then we can improve our allocations online. This structure exists in many real-world problems. For example, in the agricultural problem in Example 1, there is a task CTMC state update each time the picker reaches a new node, which rules out the previous node as a PTL. Alternatively, for order picking tasks in a warehouse, commonly ordered items are often placed near packing stations, i.e. the most likely PTLs are often clustered within a small area (De Koster et al., 2007). To maximise performance, this structure should be captured while constructing the task CTMCs. Note that for more realistic problems, the task CTMCs may combine the CTMC types discussed above. For example, a small group of PTLs may be close together, while the rest are spread out. The performance gap between the proactive methods in Section 6 decreases with the structure in the task announcement process. If there is little or no structure, alternative criteria should be used to select the most appropriate method, such as scalability.

The suboptimality of our framework. Our proactive MRTA framework is suboptimal due to the use of multiple heuristics, such as the cheapest insertion heuristic for schedule insertion (Koenig et al., 2006) and the greedy heuristic used during IWP computation (see Section 5.3). These heuristics make it difficult to provide suboptimality bounds for our approach. Despite this, the results in Section 6 demonstrate the efficacy of our framework. Proactive MRTA could be solved optimally using a centralised approach which reasons over the product of the task CTMCs with CTMCs describing robot navigation. However, the size of the resulting CTMCs would render this approach intractable, limiting its practicality.

The effects of robot interactions. Our framework ignores physical robot interactions during bidding. This may limit performance, as the robot behaviour expected during bidding diverges from what is observed during execution. For example, if multiple robots appear in the same area simultaneously they may experience congestion, which slows them down and increases waiting durations. In the worst case, robot interactions may cause a robot to become blocked and unable to complete its task. To mitigate this, we could explicitly reason over the effects of robot interactions during bidding (Street et al., 2022a, 2022b). However, these interactions often occur stochastically, which would further increase the complexity of schedule evaluation. An alternative approach would be to use existing multi-robot navigation planners to handle interactions during execution. This would reduce robot navigation durations, and decrease the modelling inaccuracies during bidding. For example, hierarchical cooperative A* (HCA*) search could be used to solve discrete multi-agent path finding online for collision-free navigation (Silver, 2005). Under HCA* robots plan sequentially, avoiding all robots that planned previously. These techniques have been extended to continuous-time settings, where robots reason over the effects of congestion on navigation (Street et al., 2020, 2022b). This is more suitable for the problems in this paper, where robot navigation durations are continuous and stochastic. In either case, a robot could plan each time it begins a new task given the paths of all currently navigating robots.

8. Conclusion

In this paper, we presented a framework that extends SSI for proactive task allocation, where we use task CTMCs to model stochastic task announcement in continuous time.

Each PTL is auctioned separately, and robots wait at IWPs to handle uncertain locations. Our framework can solve any MRTA problem where task announcement is stochastic, provided that all tasks are known and eventually announced. Further, we outperform methods which do not allocate tasks proactively, as is commonly seen in the literature, as well as methods which do not use IWPs or do not auction PTLs separately. To the best of our knowledge, this is the first proactive MRTA method for continuous-time task announcement under spatiotemporal uncertainty. In future work, we will investigate how to allocate tasks proactively while satisfying global constraints on multi-robot behaviour, and consider proactively allocating tasks specified in temporal logic.

Nomenclature

Acronyms

CTMC Continuous-time Markov chain

IWP Intermediate waiting point

MLL Most likely location

MRS Multi-robot system

MRTA Multi-robot task allocation

PTL Potential task location

SS Stay still

SSI Sequential single-item

WP Waiting point

Symbols

Φ An allocation which maps robots to schedules

$\mathcal{Q}$ A CTMC

ρ A function which maps each topological edge to a duration distribution for traversing that edge

ρ^+ A function which returns the duration distribution for navigating along the shortest path between two topological nodes

E The set of topological edges

$\texttt{IWP}_i^j$ A function which computes the IWP for the jth task in robot r_i's schedule

V The set of topological nodes

V_i^j The PTLs robot r_i has been allocated for the jth task in its schedule

I	The set of PTL-task pairs we allocate
ϕ_i	The schedule for robot r_i
SC	The schedule cost function
Ω_i^j	A function which computes the expected sum of task service times starting from the jth task in robot r_i's schedule
τ	A task
P_τ^A	A distribution over the time until announcement for task $\tau \in T$
P_τ^L	A distribution over the announcement location for task $\tau \in T$
$P_i^{L,j}$	A distribution over the location robot r_i finishes the jth task in its schedule
B_τ^Φ	The service time distribution for task $\tau \in T$ under allocation Φ
T	A finite set of tasks
$\mathcal{T}$	A topological map
$v_{w,i}^j$	The WP for the jth task in robot r_i's schedule
W_τ^Φ	The waiting duration distribution for task $\tau \in T$ under allocation Φ
R	The set of robots
r_i	The ith robot

Acknowledgments

This work is supported by the Honda Research Institute Europe GmbH, UK Research and Innovation and EPSRC through the Robotics and Artificial Intelligence for Nuclear (RAIN) hub [EP/R026084/1], the EPSRC Programme Grant 'From Sensing to Collaboration' [EP/V000748/1], and a gift from Amazon Web Services.

References

Anderson, D. F., & Kurtz, T. G. (2011). Continuous time markov chain models for chemical reaction networks. In *Design and Analysis of Biomolecular Circuits*, pp. 3–42. Springer.

Baier, C., Haverkort, B., Hermanns, H., & Katoen, J.-P. (2000). Model checking continuous-time markov chains by transient analysis. In *Proceedings of the International Conference on Computer Aided Verification (CAV)*, pp. 358–372. Springer.

Blumrosen, L., & Nisan, N. (2007). Combinatorial auctions. *Algorithmic Game Theory*, *267*, 300.

Bopardikar, S. D., Smith, S. L., & Bullo, F. (2014). On dynamic vehicle routing with time constraints. *IEEE Transactions on Robotics*, *30*(6), 1524–1532.

Buchholz, P., Kriege, J., & Felko, I. (2014). *Input Modeling with Phase-Type Distributions and Markov Models: Theory and Applications*. Springer.

Burns, E., Benton, J., Ruml, W., Yoon, S., & Do, M. (2012). Anticipatory on-line planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 22, pp. 333–337.

Capitan, J., Spaan, M. T., Merino, L., & Ollero, A. (2013). Decentralized multi-robot cooperation with auctioned pomdps. *The International Journal of Robotics Research*, *32*(6), 650–671.

Choudhury, S., Gupta, J. K., Kochenderfer, M. J., Sadigh, D., & Bohg, J. (2021). Dynamic multi-robot task allocation under uncertainty and temporal constraints. *Autonomous Robots*, *46*, 1–17.

Claes, D., Robbel, P., Oliehoek, F., Tuyls, K., Hennes, D., & Van der Hoek, W. (2015). Effective approximations for multi-robot coordination in spatially distributed tasks. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 881–890. International Foundation for Autonomous Agents and Multiagent Systems.

Cordeau, J.-F., & Laporte, G. (2007). The dial-a-ride problem: Models and algorithms. *Annals of Operations Research*, *153*(1), 29–46.

Costen, C., Rigter, M., Lacerda, B., & Hawes, N. (2022). Shared autonomy systems with stochastic operator models. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Das, G., Cielniak, G., From, P., & Hanheide, M. (2018). Discrete event simulations for scalability analysis of robotic in-field logistics in agriculture – a case study. In *Proceedings of the IEEE International Conference on Robotics and Automation, Workshop on Robotic Vision and Action in Agriculture (WRVAA)*.

Das, G. P., McGinnity, T. M., Coleman, S. A., & Behera, L. (2015). A distributed task allocation algorithm for a multi-robot system in healthcare facilities. *Journal of Intelligent & Robotic Systems*, *80*(1), 33–58.

De Koster, R., Le-Duc, T., & Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, *182*(2), 481–501.

de Nijs, F., Spaan, M., & de Weerdt, M. (2018). Preallocation and planning under stochastic resource constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Faddy, M., & McClean, S. (1999). Analysing data on lengths of stay of hospital patients using phase-type distributions. *Applied Stochastic Models in Business and Industry*, *15*(4), 311–317.

Faruq, F., Parker, D., Lacerda, B., & Hawes, N. (2018). Simultaneous task allocation and planning under uncertainty. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3559–3564. IEEE.

Ganbold, O., Kundu, K., Li, H., & Zhang, W. (2020). A simulation-based optimization method for warehouse worker assignment. *Algorithms*, *13*(12), 326.

Gerkey, B. P., & Mataric, M. J. (2002). Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, *18*(5), 758–768.

Glaschenko, A., Ivaschenko, A., Rzevski, G., & Skobelev, P. (2009). Multi-agent real time scheduling system for taxi companies. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 29–36.

Gómez, S., Arenas, A., Borge-Holthoefer, J., Meloni, S., & Moreno, Y. (2010). Discrete-time markov chain approach to contact-based disease spreading in complex networks. *Europhysics Letters (EPL)*, *89*(3), 38009.

Harman, H., & Sklar, E. I. (2022). Multi-agent task allocation for fruit picker team formation. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1618–1620.

Jalel, B. O., Véronique, V., et al. (2020). Continuous time markov chain traffic model for urban environments. In *Proceedings of the GLOBECOM IEEE Global Communications Conference*, pp. 1–6. IEEE.

Khan, M. W., Das, G. P., Hanheide, M., & Cielniak, G. (2020). Incorporating spatial constraints into a bayesian tracking framework for improved localisation in agricultural environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2440–2445. IEEE.

Khayari, R. E. A., Sadre, R., & Haverkort, B. R. (2003). Fitting world-wide web request traces with the em-algorithm. *Performance Evaluation*, *52*(2-3), 175–191.

Koenig, N., & Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2149–2154, Sendai, Japan.

Koenig, S., Tovey, C., Lagoudakis, M., Markakis, V., Kempe, D., Keskinocak, P., Kleywegt, A., Meyerson, A., & Jain, S. (2006). The power of sequential single-item auctions for agent coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1625–1629.

Koenig, S., Tovey, C. A., Zheng, X., & Sungur, I. (2007). Sequential bundle-bid single-sale auction algorithms for decentralized control. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1359–1365.

Korsah, G. A., Stentz, A., & Dias, M. B. (2013). A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research*, *32*(12), 1495–1512.

Kwiatkowska, M., Norman, G., & Parker, D. (2011). PRISM 4.0: Verification of probabilistic real-time systems. In *Proceedings of the International Conference on Computer Aided Verification (CAV)*, pp. 585–591. Springer.

Kwiatkowska, M., Norman, G., & Parker, D. (2007). Stochastic model checking. In *Proccedings of the International School on Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM)*, pp. 220–270. Springer.

Lippi, M., & Marino, A. (2021). A mixed-integer linear programming formulation for human multi-robot task allocation. In *Proceedings of the IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, pp. 1017–1023. IEEE.

Malencia, M., Kumar, V., Pappas, G., & Prorok, A. (2021). Fair robust assignment using redundancy. *IEEE Robotics and Automation Letters*, *6*(2), 4217–4224.

Mariani, S., Cabri, G., & Zambonelli, F. (2021). Coordination of autonomous vehicles: Taxonomy and survey. *ACM Computing Surveys (CSUR)*, *54*(1), 1–33.

Marie, R. (1980). Calculating equilibrium probabilities for λ (n)/ck/1/n queues. In *Proceedings of the International Symposium on Computer Performance Modelling, Measurement and Evaluation*, pp. 117–125.

Mausam, & Kolobov, A. (2012). *Planning with Markov Decision Processes: An AI Perspective*. Morgan & Claypool Publishers.

Meng, T., Li, X., Zhang, S., & Zhao, Y. (2016). A hybrid secure scheme for wireless sensor networks against timing attacks using continuous-time markov chain and queueing model. *Sensors*, *16*(10), 1606.

Nunes, E., & Gini, M. (2015). Multi-robot auctions for allocation of tasks with temporal constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Nunes, E., Manner, M., Mitiche, H., & Gini, M. (2017a). A taxonomy for task allocation problems with temporal and ordering constraints. *Robotics and Autonomous Systems*, *90*, 55–70.

Nunes, E., McIntire, M., & Gini, M. (2017b). Decentralized multi-robot allocation of tasks with temporal and precedence constraints. *Advanced Robotics*, *31*(22), 1193–1207.

Pavone, M., Bisnik, N., Frazzoli, E., & Isler, V. (2009). A stochastic and dynamic vehicle routing problem with time windows and customer impatience. *Mobile Networks and Applications*, *14*(3), 350–364.

Prorok, A. (2019). Redundant robot assignment on graphs with uncertain edge costs. In *Distributed Autonomous Robotic Systems*, pp. 313–327. Springer.

Pulungan, R., & Hermanns, H. (2018). Transient analysis of ctmcs: Uniformization or matrix exponential?. *IAENG International Journal of Computer Science*, *45*(2), 267–274.

Pütz, S., Simón, J. S., & Hertzberg, J. (2018). Move base flex: A highly flexible navigation framework for mobile robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3416–3421.

Schillinger, P., Bürger, M., & Dimarogonas, D. V. (2018). Improving multi-robot behavior using learning-based receding horizon task allocation. In *Proceedings of Robotics: Science and Systems (RSS)*.

Schneider, E., Sklar, E. I., Parsons, S., & Özgelen, A. (2015). Auction-based task allocation for multi-robot teams in dynamic environments. In *Proceedings of the Conference Towards Autonomous Robotic Systems*, pp. 246–257. Springer.

Schoenig, A., & Pagnucco, M. (2010). Evaluating sequential single-item auctions for dynamic task allocation. In *Proceedings of the Australasian Joint Conference on Artificial Intelligence*, pp. 506–515. Springer.

Silver, D. (2005). Cooperative pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pp. 117–122. AAAI Press.

Street, C., Lacerda, B., Mühlig, M., & Hawes, N. (2020). Multi-robot planning under uncertainty with congestion-aware models. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Street, C., Lacerda, B., Staniaszek, M., Mühlig, M., & Hawes, N. (2022a). Context-aware modelling for multi-robot systems under uncertainty. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Street, C., Pütz, S., Mühlig, M., Hawes, N., & Lacerda, B. (2022b). Congestion-aware policy synthesis for multirobot systems. *IEEE Transactions on Robotics*, *38*(1).

Su, X., Zhang, M., & Bai, Q. (2016). Coordination for dynamic weighted task allocation in disaster environments with time, space and communication constraints. *Journal of Parallel and Distributed Computing*, *97*, 47–56.

Surma, F., Kucner, T. P., & Mansouri, M. (2021). Multiple robots avoid humans to get the jobs done: An approach to human-aware task allocation. In *Proceedings of the European Conference on Mobile Robots (ECMR)*, pp. 1–6. IEEE.

Thummler, A., Buchholz, P., & Telek, M. (2006). A novel approach for phase-type fitting with the em algorithm. *IEEE Transactions on Dependable and Secure Computing*, *3*(3), 245–258.

Tihanyi, D., Lu, Y., Karaca, O., & Kamgarpour, M. (2021). Multi-robot task allocation for safe planning under dynamic uncertainties. *arXiv Preprint*, *arXiv:2103.01840*.

Tsao, M., Iglesias, R., & Pavone, M. (2018). Stochastic model predictive control for autonomous mobility on demand. In *Proceedings of the International Conference on Intelligent Transportation Systems (ITSC)*, pp. 3941–3948. IEEE.

Xue, F., Tang, H., Su, Q., & Li, T. (2019). Task allocation of intelligent warehouse picking system based on multi-robot coalition. *KSII Transactions on Internet and Information Systems (TIIS)*, *13*(7), 3566–3582.

Zheng, S. K. X., Tovey, C., Borie, R., Kilby, P., Markakis, V., & Keskinocak, P. (2008). Agent coordination with regret clearing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, p. 101.

Zheng, X., Koenig, S., & Tovey, C. (2006). Improving sequential single-item auctions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2238–2244. IEEE.