Online Multi-Robot Task Assignment with Stochastic Blockages

Wilde, N.; Alonso-Mora, J.

DOI
10.1109/CDC51059.2022.9993099

Publication date
2022

Document Version
Final published version

Published in
Proceedings of the IEEE 61st Conference on Decision and Control (CDC 2022)

2022 IEEE 61st Conference on Decision and Control (CDC)
December 6-9, 2022. Cancún, Mexico

Online Multi-Robot Task Assignment with Stochastic Blockages

Nils Wilde and Javier Alonso-Mora

***Abstract*— In this paper we study the multi-robot task assignment problem with tasks that appear online and need to be serviced within a fixed time window in an uncertain environment. For example, when deployed in dynamic, human-centered environments, the team of robots may not have perfect information about the environment. Parts of the environment may temporarily become blocked and blockages may only be observed on location. While numerous variants of the Canadian Traveler Problem describe the path planning aspect of this problem, few work has been done on multi-robot task allocation (MRTA) under this type of uncertainty. In this paper, we introduce and theoretically analyze the problem of MRTA with recoverable online blockages. Based on a stochastic blockage model, we compute offline tours using the expected travel costs for the online routing problem. The cost of the offline tours is used in a greedy task assignment algorithm. In simulation experiments we highlight the performance benefits of the proposed method under various settings.**

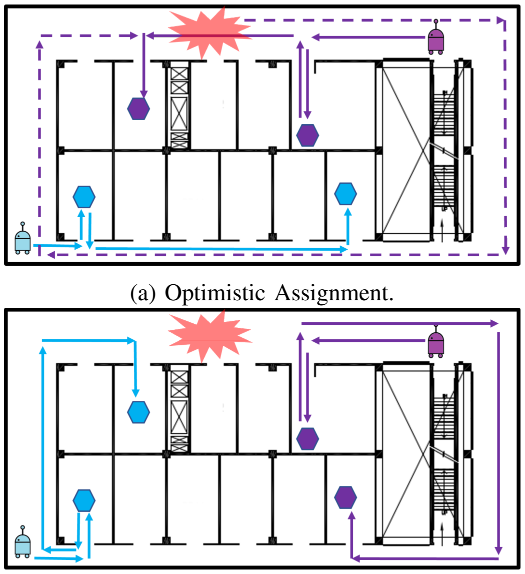
I. Introduction

Autonomous robots find increasingly widespread deployment in service applications. For instance, in hospital environments mobile robot platforms can help to reduce the workload for qualified hospital staff [1], [2]. Service tasks usually arrive periodically over time and then require a robot to travel to a specific location. This problem is often modelled as a (multi-) dynamic vehicle routing problem [3], [4]. The goal is to allocate tasks to robots such that they can provide the most efficient service, for example in terms of cumulative waiting times, in some cases subject to additional constraints such as time-windows.

In this paper we study the multi-robot task assignment (MRTA) problem when the environment – and thus the robot travel time to the service locations – is uncertain. In particular, parts of the environment can become temporarily untraversable. For instance, corridors in human-centered buildings can become cluttered with maintenance personnel and equipment, or large crowds of people make it very hard for a robot navigate in a socially acceptable way while avoiding the freezing robot problem [5], [6]. Often such events can only be observed when a robot is at the location. However, it might be possible to identify in what locations blockages *can* happen, e.g., narrow corridors with frequent maintenance activities, and with which frequency. Such online blockages pose two challenges in MRTA: 1)

This research is supported by the European Union's Horizon 2020 research and innovation program under Grant 101017008, and in part by Ahold Delhaize. All content represents the opinion of the author(s), which is not necessarily shared or endorsed by their respective employers and/or sponsors.

The authors are with the Department for Cognitive Robotics, 3ME, Delft University of Technology, Delft, Netherlands, {n.wilde, j.alonsomora}@tudelft.nl


(a) Optimistic Assignment.

(b) Informed Assignment.

Fig. 1: Illustrative example for multi-robot task assignment with blockages. Black indicates obstacles, and hexagon are task locations, where the colour indicates which robot is assigned to it, and the red area might become temporarily blocked. We compare an optimistic (a) and the proposed informed approach (b).

During the task assignment step, robots should be matched to tasks taking into account possible blockages. 2) When a robot is assigned to service some tasks, it needs to compute a route that maximizes the expected quality of service, given the chance of blockages appearing during navigation. In case a blockage is encountered, it replans its route.

We illustrate this problem with an example in Figure 1. In (a) we show an optimistic assignment that does not consider the risk of blockages. Robots are assigned such that the tour lengths are minimal. However, during execution the purple robot might find that the red area has become blocked, causing it to take a large detour, possibly resulting in missing the second task's service deadline. We propose an assignment algorithm that takes the possibility of blockages into account. In (b) the robots avoid the blockage area. This leads to slightly longer tours when the blockage is not present, but significantly shorter tours in case the blockage is present.

We pose the problem of MRTA with a homogenous fleet where tasks i) require one robot to visit a location, ii) appear online, and iii) have a fixed time window for service. Further, we model the environment as uncertain where blockages appear in a stochastic way, making parts of the environment temporarily untravesable. To the best of our knowledge, these two fundamental robot planning problems have not been combined before [3]. We model blockages with Poisson processes, and present a greedy task assignment algorithm using the expected state of this process, given the available observations. In a theoretical analysis, we show how this approach avoids failure cases of two naive alternatives, and demonstrate the performance benefits in numerous simulations with different environments, fleet sizes and task loads.

A. Related Work

Among the early works on path planning in uncertain environments is the Canadian Traveller Problem (CTP) [7]–[9]. An agent travels on a graph from a start to a goal vertex, however some edges can be blocked. Unfortunately, this can only be observed on-site, when the agent is at one of the edge's endpoints. In general, the CTP is PSPACE complete [7]. The two most prominent variants are a) the classic CTP where blockages are placed by an adversary, and b) stochastic CTP where blockages appear randomly. The classic CTP has been extended to the case where a robot needs to visit multiple locations, i.e., solve a Steiner Traveling Salesman Problem (S-TSP), and the number of blockages is known [10]. Similarly, [11] and [12] study minimum-latency variants of the problem. Moreover, the authors of [13] consider the multi-robot case of the S-TSP for a homogeneous fleet where any robot in the fleet can visit each location. This is closely related to our work. However, there are two major differences: Firstly, we consider stochastic blockages instead of adversarial ones, and secondly, tasks in our MRTA setting have deadlines.

Stochastic versions of CTP are considered in [14]–[18]. The stochastic nature of blockages can be unstructured [17], or structured [18]. In the latter case, the robot can infer about the status of edges that are not currently or previously observed. The authors of [14] tackle the stochastic CTP by sampling possible environments and then perform *rollouts* of an online path planning policy. In our approach, we also rely on sampling environments, but instead of rollouts we only compute offline Steiner-tours for computational efficiency.

With exception of the adversarial setting of [13], the discussed literature did not consider multi-robot coordination where a fleet needs to decide which tasks are served by which robot.

Multi-robot task assignment (MRTA) problems where tasks appear online and require robots to travel to locations are usually referred to as Dynamic Vehicle Routing Problems (DVRP) [3], [19]. Stochastic version of the DVRP include uncertainty on the load [20], travel time [21], [22], and demand [21]. For general MRTA, [23] considers uncertainty on the robots' capabilities to complete tasks (on time). Inter-agent congestion is another source of online blockages [24], [25], where the robots' paths have direct influence on the occurrence of blockages while in our problem blockages appear independently of the robots' behaviour.

The most distinctive features of our work are that travel time uncertainty takes a binary form as in the CTP – parts of the environment are either traversable or blocked – and blockages are *recoverable*, i.e., after some time they disappear allowing the robots to travel through again.

B. Contributions

We make the following contributions: i) Pose the problem of multi-robot task assignment with stochastic blockages in the environment, ii) propose a greedy assignment using the expected blockage status and discuss naive alternatives, and iii) demonstrate the practicality of the proposed framework in extensive simulations.

II. Problem Formulation

In this section we formalize the problem of MRTA where tasks require robots to visit some location in the workspace and perform a service operation. We consider an environment encapsulated in a directed weighted graph $G = (V, E, l)$ where V and E are vertices and edges, and weights l describe the duration of traversing an edge. A fleet of m robots $R = \{r_1, \dots, r_m\}$ serves a set of n tasks $\mathcal{T} = \{T_1, \dots, T_n\}$. Each task is a tuple $T = (v, t^r, t^d, d)$ where v is a vertex in V, t^r is the release time when the task is announced to the robot fleet, $t^d > t^r$ is a deadline, and d is the duration it takes for the robot to serve the task. Let $t^a(r)$ be the time a robot r servicing task T arrives at the vertex v. (*Note: To avoid unintentional servicing, we can design the graph G such that v is a copy of an existing vertex in V.*)

Let r_i be a robot currently located at position s_i, and let $\mathcal{T}_i$, be the tasks assigned to it. Further, let τ_i be a tour starting at s_i serving tasks $\mathcal{T}_i$. The tour then defines the robot arrival times $t^a(T, \tau)$ for all tasks $T \in \mathcal{T}_i$. The service quality for each task is measured by the wait time defined as

$$w(T, \tau) = \begin{cases} t^a(T, \tau) - t^r(T) \text{ if } t^a(T, \tau) \le t^d, \\ M \text{ otherwise.} \end{cases} \tag{1}$$

Here M is a large constant to penalize service after the deadline. The cost of a robot's tour τ is then defined by the sum of the associated service costs:

$$c(\tau_i) = \sum_{T \in \mathcal{T}_i} w(T, \tau_i). \tag{2}$$

An assignment is a set $\mathcal{A} \subseteq \{(r_i, T_j) | r_i \in R, \mathcal{T}_j \in \mathcal{T}\}$ such that for every $T_j \in \mathcal{T}$ there exists exactly one pair in $\mathcal{A}$ containing T_j, i.e., every task is assigned to exactly one robot. However, a robot can be assigned to multiple tasks; thus each r_i can appear in multiple pairs in $\mathcal{A}$. Further, let $\mathcal{T}_i(\mathcal{A})$ be the set of tasks assigned to robot r_i under $\mathcal{A}$. The MRTA problem then solves

$$\begin{aligned} &\min_{\mathcal{A}} \sum_{r_i \in R} c(\tau_i) \\ &s.t. \, \tau_i \text{ serves tasks } \mathcal{T}_i(\mathcal{A}), \\ &\qquad \mathcal{T}_1(\mathcal{A}) \cup \dots \cup \mathcal{T}_m(\mathcal{A}) = \mathcal{T}. \end{aligned} \tag{3}$$

Note: We do not explicitly model the possibility to not service a task. This case can be handled by assigning the *task to an arbitrary robot which will visit the task location at some time after the deadline with penalty M.*

In this paper, we consider dynamic changes in the environments. That is, edges in the graph can become blocked during execution. Whether an edge e is blocked is only revealed when one of the robots reaches an endpoint of e. We assume that this information is shared instantaneously among the fleet, but that previously assigned tasks cannot be reassigned. Thus, the routing problem for each robot over the destinations $\mathcal{T}_i$ becomes one of finding an online algorithm, i.e., a policy π, that recomputes the route whenever a robot discovers a blocked edge. Blockages appear following some stochastic process X. Thus, the set of edges in the graph becomes a random process over time $E(t)$ We denote the length of a tour starting at s, visiting $\mathcal{T}_i$ found by a policy π under edge presence $E(t)$ as $c(\pi, s, \mathcal{T}_i, E(t))$. This leads to our main problem statement.

Problem 1 (MRTA under stochastic edge blockages)**.** Given is a graph $G = (V, E, l)$, a set of n tasks $\mathcal{T}$, a robot fleet $R = \{r_1, \dots, r_m\}$ located at a common depot s. Further, let X be a stochastic process describing the traversable edges $E(t)$ at any time $t \geq 0$. Find an assignment $\mathcal{A}$ and a routing policy π that solves

$$\min_{\mathcal{A},\pi} \mathbb{E}_{E(t)\sim X}\Big[\sum_{r_i\in R} c(\pi, s, \mathcal{T}_i(\mathcal{A}), E(t))\Big]. \tag{4}$$

III. Approach

We begin by characterizing a stochastic process for how blockages occur and how a robot fleet can keep track of observations. This prepares us for introducing algorithms for dynamic single robot path planning and a greedy MRTA assignment in the subsequent section.

A. Blockage Model

First, we specify how blockages occur on the graph. In this work, we assume that blockages appear independently at k different locations in the environment. That is there are k subsets of edges $\mathcal{E} = \{E_1, E_2, \dots, E_k\}$ where $E_i \subset E$ and E_i, E_j are disjoint for all $i, j = 1, \dots, k$ where $i \neq j$.

Definition 1 (Poisson Blockage Model)**.** Given subsets of edges $\mathcal{E} = \{E_1, \dots, E_k\}$, blockages appear on a subset E_i following a Poisson process A_i with parameter λ_i, and the edges E_i remain blocked following a second Poisson process B_i with parameter μ_i. The processes A_i, B_i, A_j, B_j are independent for all i, j. We denote the model as a triplet $X = (\mathcal{E}, \boldsymbol{\lambda}, \boldsymbol{\mu})$.

This model is known as a *Birth-Death Process*, often used in queuing theory [26], [27]. We focus on the case when the queue has a capacity of 1 such that blockages cannot accumulate, i.e., queue. This particular process is a $M/M/1/1$ queue, also known as a queuing system with *blocked calls cleared* [27]. This is equivalent to pausing process A by setting $\lambda = 0$ once an event happened (a blockage appeared), until the event of process B happened (the blockage cleared) and vice versa. The blockage model then describes a stochastic process $\psi_e(t)$ for each edge $e \in E$ taking values in $\{0, 1\}$ with the convention that 0 corresponds to no blockage. We summarize the process for all edges in the vector $\boldsymbol{\psi}(t)$.

Without loss of generality we can assume that the union of the partitioning $\mathcal{E}$ is equal to the set of all edges E. All edges that are never blocked can be summarized in a set E_{k+1} for which we set $\lambda_{k+1} = 0$. Moreover, we make the following assumption.

Assumption 1 (Connectivity)**.** Given a graph G and blockage process X, we assume that for any realization of X at an arbitrary time step t, the graph G remains connected.

Connectivity assures that each robot can always reach every task. Without this assumptions, robots could temporarily become trapped, requiring a richer formulation that allows them to wait.

B. Recording Observations

We introduce some notation for how the robot fleet keeps track of the observation about blocked edges. Recall that $\psi_e(t) \in \{0, 1\}$ denotes if the edge e is blocked at time t. The blockage model describes a random process $\boldsymbol{\psi}(t)$ for each edge being blocked at some time t. Now let $\Omega = \{\omega_1, \dots, \omega_{|E|}\}$ be the observations for all edges. That is, for each $e \in E$ there exists an $\omega_e = (t_e, \psi_e)$ where t_e indicates the last time some robot visited one of the end points of edge e, and ψ_e the observation about the edge being blocked at that time. If an edge was never observed we set $\omega_e = (-\infty, 0)$.

Property 1 (Most recent observations)**.** For any edge $e \in E$, let t be the time of the most recent observation. For any time $t' \geq t$, the probability $\mathbb{P}(\psi_e(t) = \psi_e(t'))$ is independent of any past observations made at time $t'' < t$. Thus, Ω only stores one observation per edge. This is a direct result of the memory-less property of the Poisson process [26].

C. Observation Model

An observation model takes as input observations Ω where the latest observation was made at time t and outputs a probability for blockages at some time $t' \geq t$, i.e., $\mathbb{P}(\boldsymbol{\psi}(t')|\Omega)$.

Definition 2 (Observation Model)**.** Given is a blockage model $(\mathcal{E}, \boldsymbol{\lambda}, \boldsymbol{\mu})$, current time t and some observation ψ about an edge e made at time $t' \leq t$. Further, let E_j denote the subset in $\mathcal{E}$ containing e. The probability of e having the same blockage status as at time t is then given by

$$\mathbb{P}(\psi_e(t) = \psi_e(t')) = \begin{cases} f_0(t, t') \text{ if } \psi_e(t') = 0, \\ f_1(t, t') \text{ otherwise,} \end{cases} \tag{5}$$

where

$$\begin{aligned} f_0(t, t') &= \frac{\mu}{\lambda + \mu} e^{-(\lambda+\mu)(t-t')}, \\ f_1(t, t') &= \frac{\lambda}{\lambda + \mu} e^{-(\lambda+\mu)(t-t')}. \end{aligned} \tag{6}$$

Algorithm 1: MRTA with Stochastic Edge Blockages

```
Input: Graph $G = (V, E, l)$, online task sequence $\mathcal{T}$,
       fleet $R$
1 $Q = \emptyset,\ \mathcal{A} = \emptyset$
2 for time $t = 1$ to $t^{\max}$ do
3     $\Omega \leftarrow$ Update_Observations($R$)
4     $Q \leftarrow$ receive_online_Tasks($\mathcal{T}$, t)
5     $\mathcal{A} \leftarrow$ Assign_Tasks($G, t, Q, R, \mathcal{A}$)
6     for $i = 1$ to $|R|$ do
7         if $\mathcal{T}_i$ changed or $\Omega$ changed then
8             $\tau_i \leftarrow$ Compute_Tour($G, \Omega, t, s_i, \mathcal{T}_i$)
9         $r_i \leftarrow$ Move_Robot($G, r_i, \tau_i$)
```

The probabilities f_0 and f_1 are given by the transient probabilities for an $M/M/1/1$ queue being either empty or filled at some given time [28].

This observation model requires the blockage model $(\mathcal{E}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ to be known. In practise, the process parameters can be estimated from historic data for the robots' environment.

D. Task Model

As common in dynamic vehicle routing [19], we model the online arrival of tasks as a Poisson process with parameter ξ, i.e., the expected time between task arrivals is $1/\xi$. Task locations v are randomly distributed following some density $\alpha : V \to [0, 1]$.

IV. Assignment Algorithm

In this section we first present a general framework to solve Problem 1, and then derive a online path planning approach using an estimate of the blockage process.

Algorithm 1 provides a high level overview. Over the given time window, the robot fleet maintains up-to-date information on their observation set Ω as well as any newly arrived tasks Q (line 3,4). If Q is non-empty, the new tasks are added to the assignment $\mathcal{A}$ (line 5). Then, each robot recomputes their route if a) the set of tasks assigned to them $\mathcal{T}_i$ has changed, or b) new observations are available (line 6-8). This re-computation upon making new observations Ω makes the tour computation a *policy*. At the end of each iteration all robots move along their tour τ_i for the next time step. We will now first describe a greedy online task assignment algorithm before we then study different approaches for online path finding and discuss how they affect the assignment.

A. Computing Tours

We first present our approach for Compute_Tour using the expected blockage status for each edge, given current and past observations. The online property of the problem is taken care of in the recomputation condition in line 7 of Algorithm 1. Thus, the output of Compute_Tour is an offline tour, starting at s and visiting all task locations while minimizing cost c. We call this the *reference tour*.

a) Static Simplification: In general, blockages can appear while a robot executes its tour, even if the respective edges where observed as unblocked at start time. While this is captured in our observation model, such dynamic changes pose a major challenge for computing a tour, since it makes the edge costs time dependent.

The task assignment algorithm requires frequent computation of numerous tentative tours; thus, we make the simplifying assumption that the probability of edges being blocked is constant during the execution of the tour. That is, let $\Delta(\mathcal{T})$ be the time needed to service some tasks $\mathcal{T}$. The function Compute_Tour then assumes that

$$\mathbb{P}\big(\boldsymbol{\psi}(t)\big) \approx \mathbb{P}\big(\boldsymbol{\psi}(t + \Delta(\mathcal{T}))\big). \tag{7}$$

Notice, that this is solely a simplification of the tour planning algorithm, we do not assume that the actual blockage process satisfies this condition.

To plan a reference tour we define a new graph G', where the edge costs are defined by the current observations of blockages Ω (which we describe below). We then compute a tour starting at s and visiting all task locations while minimizing cost c. Due to the complexity of our cost function induced by the deadlines, we use a min-cost insertion heuristic [29] for computing tours.

b) Expected edge length and expected graph: To maintain a homogeneous structure of the graph under different blockage realizations, we make the modelling choice to not actually delete blocked edges, but instead update their costs accordingly. For any realisation of the edge blockages $\boldsymbol{\psi}$, let $G(\boldsymbol{\psi}) = (V, E \setminus E'(\boldsymbol{\psi}), l)$. This is equivalent to having a graph $G'(\boldsymbol{\psi}) = (V, E, l')$ where for l'_e is equal to the length of the shortest path from v to u on $G(\boldsymbol{\psi})$ with v and u being the start and end vertex of the edge e, respectively. Trivially, if e is not blocked $l'_e = l_e$. The advantage of using $G'(\boldsymbol{\psi})$ is that any path or tour that exists on the unblocked graph G also exists on $G'(\boldsymbol{\psi})$, but will have a different cost.

Using expected edge lengths, we propose a method for Compute_Tour, denoted by π^{Expect}. Given observations Ω and thus the blockage probabilities $\mathbb{P}(\boldsymbol{\psi}(t)|\Omega)$ defined in (5), we can generate a graph $G^{\text{Exp}}(\Omega, t) = (V, E, \hat{l})$ where the edge cost correspond to the expected edge costs:

$$\hat{l}_e = \mathbb{E}_{\boldsymbol{\psi} \propto \mathbb{P}(\boldsymbol{\psi}(t)|\Omega)}[l'_e(\boldsymbol{\psi})].$$

In practice, we rely on sampling N random configurations $\Psi = \{\boldsymbol{\psi}^1, \ldots, \boldsymbol{\psi}^N\}$ from the posterior. Thus,

$$\hat{l}_e \approx \frac{1}{N} \sum_{\boldsymbol{\psi} \in \Psi} l'_e(\boldsymbol{\psi}). \tag{8}$$

Our proposed approach π^{Expect} computes a reference tour of minimal cost on the graph G^{Exp}.

B. Greedy Assignment

With the function Compute_Tour now defined, we present our greedy task assignment approach. Algorithm 2 shows the assignment algorithm, that iteratively assigns all tasks in the

Algorithm 2: `Assign_Tasks`

```
Input: Graph $G$, current time $t$, task queue $Q$, latest
       observations $\Omega$, fleet $R$, current assignment $\mathcal{A}$
Output: New assignment $\mathcal{A}$
1  while $Q$ is not empty do
2      for $r_i \in R$ do
3          $\mathcal{T}_i \leftarrow$ tasks assigned to $r_i$ under assignment $\mathcal{A}$
4          $\tau = \texttt{Compute\_Tour}(G, \Omega, t, s_i, \mathcal{T}_i)$
5          for $T_j \in Q$ do
6              $\tau' = \texttt{Compute\_Tour}(G, \Omega, t, s_i, \mathcal{T}_i \cup \{T_j\})$
7              $c(r_i, T_j) = c(\tau') - c(\tau)$
8      $(r^*, T^*) \leftarrow \arg\min_{r_i, T_j} c(r_i, T_j)$
9      $\mathcal{A} \leftarrow \mathcal{A} \cup \{(r^*, T^*)\}$
10     $Q \leftarrow Q \setminus T^*$
11 return $\mathcal{A}$
```

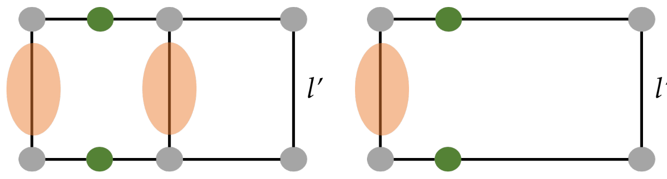
queue Q to the robot fleet. We compute the marginal increase in cost of the reference path found by `Compute_Tour` for all pairs of tasks in Q and robot in the fleet (lines 2-7). We then pick the pair with minimal value and add it to the assignment (line 8), and repeat until Q is empty.

C. *Naive Tour Computation Methods*

In addition to the proposed approach, we introduce two naive alternative methods for planning reference tours: Optimistic – $\pi^{\texttt{Optimistic}}$ and Static – $\pi^{\texttt{Static}}$. Both approaches lead to policies that do not use predictions about blockages, but only update the graph when making new observations. Such techniques are still commonly used due to their simplicity, as discussed in [18]. Further, we show how these two methods can fail in simple example cases.

a) Optimistic Planning: The first alternative $\pi^{\texttt{Optimistic}}$ plans in a purely optimistic fashion and only considers observations made at the current time step. At time t, let E^t be the set of edges for which we have an observation $\omega_e = (t, 1)$ in the current set of observations Ω, i.e., $E^t = \{e \in E | \omega_e = (t, 1)\}$. The optimistic approach then constructs a vector $\boldsymbol{\psi}^t$ where $\psi_e^t = 1$ if $e \in E^t$ and 0 otherwise. We then use generate the graph $G'(\boldsymbol{\psi}^t)$ and compute the corresponding optimal tour τ^0. This approach always assumes a best case scenario and thus underestimates the online cost.

We show a failure case for $\pi^{\texttt{Optimistic}}$ in (a). We assume that both edges going through orange regions are blocked at all times ($\lambda = \infty$, $\mu = 0$). Let a single robot start at the bottom left vertex, and a task with a deadline of 7 timesteps appears at the upper green vertex. After observing the first blockage, $\pi^{\texttt{Optimistic}}$ will reroute to go up through the central edge. Upon observing that it is also blocked, the optimistic approach will assume that the left blockage is no longer present. Then, for any $l' > 1$, the robot will reroute to the left vertical edge. This traps the online planner in a cycle, i.e., it will repeatedly try to use the left or central vertical edge and thus fail to ever reach the task location.



(a) Failure Case for $\pi^{\texttt{Optimistic}}$ (b) Failure Case for $\pi^{\texttt{Static}}$

Fig. 2: Problem instances creating failure cases. Green vertices indicate locations where tasks can arrive, all edges have length 1 with exception of the right most edge with length l'. The edge going through the orange area can become blocked.

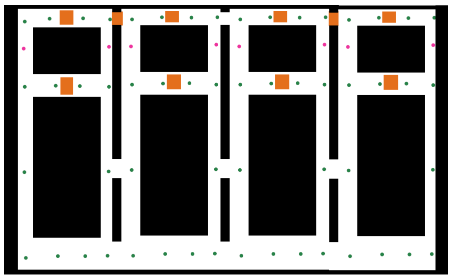
b) Static Planning: The second approach $\pi^{\texttt{Static}}$ takes into account past observations about blocked edges, but does not leverage the posterior probabilities. Instead, let $E^{\texttt{static}}$ be the set of edges where the most recent observation was blocked, i.e., $E^{\texttt{static}} = \{e \in E | \omega_e = (t', 1) \text{ for any } t' \leq t\}$ where t is the current time. Similar to $\pi^{\texttt{Optimistic}}$ we then construct the vector $\boldsymbol{\psi}^{\texttt{static}}$ and graph graph $G' = (\boldsymbol{\psi}^{\texttt{static}})$ and compute the corresponding optimal tour $\tau^{\texttt{static}}$. The static reference tour can yield an over- or an underestimate of the online cost.

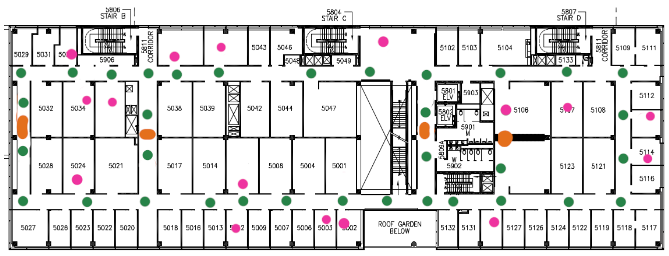
A failure case for $\pi^{\texttt{Static}}$ is shown in Figure 2 (b). We assume that the edge going through the orange region is blocked at time step 0, but then follow a process with finite parameters $\lambda, \mu > 0$. Tasks appear at both green vertices over time. The static method $\pi^{\texttt{Static}}$ observes the blockage at time 0 and will then use the right route. Due to the static assumption, it will never go to either of the vertices bordering the risk region, and thus the robots will not observe when the blockage disappears. Let $l' = 5$. Further, let tasks appear every 10 time steps, but have a deadline of 4 timesteps. Then $\pi^{\texttt{Static}}$ will not be able to deliver any task within the deadline.

We observe that both failure cases are avoided by the proposed method $\pi^{\texttt{Expect}}$. In the first case, $\pi^{\texttt{Expect}}$ will use the unblocked route and deliver the task after 6 timesteps. In the second case, the probability of a blockage is $\approx 1/2$ for some sufficiently large t for any finite parameters $\lambda, \mu > 0$. The expected length of the risk edge then is $\hat{l}_e = 1{+}6/2$, resulting a tour that attempts to traverse it. If the blockage is not present, the task will be serviced in time. Thus, at time t, the proposed approach $\pi^{\texttt{Expect}}$ has a .5 probability of successfully servicing a task, while $\pi^{\texttt{Static}}$ has a probability of 0. Finally, we notice that these failure cases can be extended to a multi-robot setting by creating connected copies of the shown example graphs. When sufficiently far apart, each subgraph will be served by one of the robots, such that we retain the single robot failure cases.

V. Evaluation

We show an extensive set of numerical experiments comparing the proposed approach to the naive baselines to highlight. We will observe that our approach performs well


(a) Artificial environment.



(b) Office environment.

Fig. 3: Simulation scenarios. Obstacles are shown in black, vertices of the graph in green. Purple indicate vertices where tasks can arrive, and edges going through the orange areas are at risk of becoming blocked temporarily.

under different blockage processes and thus outperforms the simpler techniques.

A. Experiment Setup

a) Environment: We consider two different environments: a) an artificial warehouse setting, reassembling the feature of the two failure cases and b) a real world office environment, shown in Figure 3. Both environments feature multiple dedicated sparsely distributed task locations (shown in purple). Moreover, orange indicates areas where there is a risk of blockages. For each area, all edges going through it form a *risk set* E_i, which are then the elements of the blockage process.

b) Performance Measure: Our key measure of performance is the *rejection rate*, i.e., percent of tasks that were not served on time before the deadline.

c) Other parameters: Each trial of the simulation is run for 4000 timesteps. We generate random arrival processes for tasks, where the arrival rate ξ is chosen such that the average number of tasks is 100, 200, 300, and 400. For each generated task sequence, we run a simulation with different blockages processes. For simplicity of analysis, we pick the same parameters λ_i, μ_i for all risk sets E_i. In the plots, we indicate the inverse values (λ^{-1}, μ^{-1}), which correspond to the expected unblocked / blocked times.

For the proposed method $\pi^{\texttt{Expect}}$ we compute the expectation with $N = 20$ samples in equation (8).

d) Baseline Algorithms: We compare our proposed assignment framework using $\pi^{\texttt{Expect}}$ to a number of baseline and competitive approaches.

We simulate the naive alternatives $\pi^{\texttt{Optimistic}}$ and $\pi^{\texttt{Static}}$ to show the benefits of computing the expected blockage status and how the failure cases can also happen in realistic scenarios. Moreover, we consider a heuristic lower bound on performance `Unblocked` where no blockages are present. Further, `all_blocked` considers all risk edges being blocked at all times, i.e., all risk edges are removed from the graph. This corresponds to the most *risk-adverse* strategy.

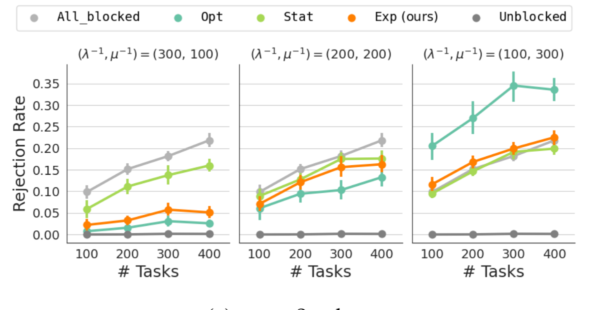
B. Results

a) Artificial Environment: In the artificial environment we first consider a fleet of 2 robots. Task appear only in the left half of the environment, allowing us to later compare the result with a fleet of 4 robots without changing other process parameters. In Figure 4a we show the result for three different blockage settings, varying the balance between expected unblocked and blocked time. Overall we observe that the proposed $\pi^{\texttt{Expect}}$ performs second best (neglecting the `Unblocked` baseline). However, $\pi^{\texttt{Optimistic}}$ and $\pi^{\texttt{Static}}$ each fail for one of the imbalanced blockage processes. When blockages are rare, i.e., $(\lambda^{-1}, \mu^{-1}) = (300, 100)$, the static method performs significantly weaker than the other two. This is due to the fact that it does not *forget* an observed blockage, but continues to assume that the blockage is still in place until observed otherwise. For some of the blocked edges in the artificial map, the robot has no incentive to go to an adjacent vertex, unless with the intent to traverse it. Thus, for these edges $\pi^{\texttt{Static}}$ does not make new observations once they were observed as blocked. For long simulations time the behaviour becomes similar to the `All_blocked` setting where any risk edges are avoided. In the opposite case $(\lambda^{-1}, \mu^{-1}) = (100, 300)$ where risk edges are blocked more often than not, the optimistic approach $\pi^{\texttt{Optimistic}}$ performs very poorly. This is closely related to the failure case in Section IV-C, the robots start to oscillate, i.e., go back and forth between two blockages until one of them disappears instead of using the blockage free detour. Finally, we observe that `All_blocked` performs poorly when blockages are rare, but is among the strongest when blockages appear more often.

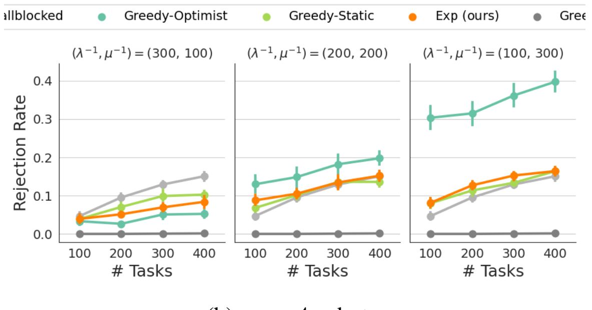
We repeat the experiment with a fleet of 4 robots and task appearing in all target locations as shown in Figure 3a; the results are plotted in Figure 4b. The overall trend is similar to the 2 robot case, yet the differences between the approaches are smaller. Moreover, when blockages remain for long, i.e., $(\lambda^{-1}, \mu^{-1}) = (100, 300)$, the `all_blocked` approach performs best.

In summary, the experiment showcases the advantage of the proposed approach. While $\pi^{\texttt{Optimistic}}$ and $\pi^{\texttt{Static}}$ are successful under some blockage parameters, their performance is poor under other settings. In contrast, $\pi^{\texttt{Expect}}$ avoids these pitfalls and performs strongly under all parameter settings.

b) Real-World Environment: Figure 5 shows the results of the second experiment using the map in Figure 3b.

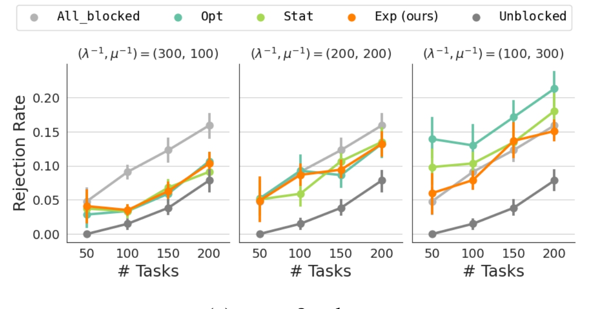


(a) $m = 2$ robots.

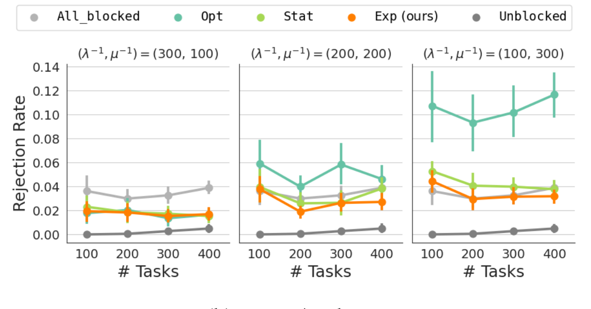


(b) $m = 4$ robots.

Fig. 4: Results in the artificial environment. For $m = 2$ tasks appear only in the left half of the environment, thus it can perform better for the same number of tasks despite a smaller fleet.



(a) $m = 2$ robots.



(b) $m = 4$ robots.

Fig. 5: Results in the real-world office environment.

For $m = 2$ robots, we first observe that the rejection rate of `Unblocked` increases with the number of tasks, indicating the the problem is harder even in the absence of blockages. Nonetheless, the proposed approach $\pi^{\texttt{Expect}}$ is still among the strongest of all other techniques. Interestingly, $\pi^{\texttt{Static}}$ does not fail when $(\lambda^{-1}, \mu^{-1}) = (300, 100)$, but performs worse in the opposite case. The optimistic approach $\pi^{\texttt{Optimistic}}$ shows the same trend as in the first experiment, showing a high rejection rate when blockages are lasting longer.

For $m = 4$ we doubled the number of tasks throughout the experiment. In case of `Unblocked` the rejection rate still grows with the number tasks. Again, we observe that $\pi^{\texttt{Optimistic}}$ fails for $(\lambda^{-1}, \mu^{-1}) = (100, 300)$. On the other hand, here $\pi^{\texttt{Static}}$ performs similar to $\pi^{\texttt{Expect}}$ in almost all settings.

In conclusion, the proposed method $\pi^{\texttt{Expect}}$ shows a strong performance across all settings in the real-world environment as well. While the static approach $\pi^{\texttt{Static}}$ performs nearly as good as $\pi^{\texttt{Expect}}$, the optimistic approach $\pi^{\texttt{Optimistic}}$ still exhibits its failure cases. Thus, the second experiments highlights the benefits of the proposed approach under realistic settings.

VI. Conclusion

In this paper we studied the problem of MRTA with dynamic temporary blockages in the environment. We used Poisson processes to model edge blockages, and derived an according observation model. We proposed an online routing policy based on reference tours using the expected travel distances, given observations. The cost of these reference tours is used in a greedy task assignment algorithm. We showed how using the expected travel distance avoids pitfalls of simpler naive routing policies, namely an optimistic and a static approach. Simulation experiments showed that the proposed approach is always among the strongest in various settings while the naive approaches perform much less reliably.

There are several directions for future work. Our method assumes that the parameters of the blockage process are known, which does not necessarily hold in practise. We would like to i) study the robustness of the proposed method towards inaccurate estimates of these parameters, and ii) include the estimation of the parameters in the problem setup. This poses an interesting exploration versus exploitation trade-off similar to [15], [18]. Moreover, the problem setup can be extended to pickup and delivery. We believe that blockages potentially have a larger impact on performance in such problems since robots cannot partition their workspace to avoid using risk edges. Finally, it would be interesting to consider large scale problems with tens to hundreds of robots, ideally using real world data sets.

References

[1] K. Niechwiadowicz and Z. Khan, "Robot based logistics system for hospitals-survey," in *IDT Workshop on interesting results in computer science and engineering*, 2008.

[2] S. Abubakar, S. K. Das, C. Robinson, M. N. Saadatzi, M. C. Logsdon, H. Mitchell, D. Chlebowy, and D. O. Popa, "Arna, a service robot for nursing assistance: System overview and user acceptability," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2020, pp. 1408–1414.

[3] B. H. O. Rios, E. C. Xavier, F. K. Miyazawa, P. Amorim, E. Curcio, and M. J. Santos, "Recent dynamic vehicle routing problems: A survey," *Computers & Industrial Engineering*, vol. 160, p. 107604, 2021.

[4] H. N. Psaraftis, M. Wen, and C. A. Kontovas, “Dynamic vehicle routing problems: Three decades and counting,” *Networks*, vol. 67, no. 1, pp. 3–31, 2016.

[5] A. J. Sathyamoorthy, U. Patel, T. Guan, and D. Manocha, “Frozone: Freezing-free, pedestrian-friendly navigation in human crowds,” *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4352–4359, 2020.

[6] P. Trautman and A. Krause, “Unfreezing the robot: Navigation in dense, interacting crowds,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 797–803.

[7] A. Bar-Noy and B. Schieber, “The canadian traveller problem,” in *Proceedings of the second annual ACM-SIAM symposium on Discrete algorithms*. Citeseer, 1991, pp. 261–270.

[8] C. H. Papadimitriou and M. Yannakakis, “Shortest paths without a map,” *Theoretical Computer Science*, vol. 84, no. 1, pp. 127–150, 1991.

[9] H. Guo and T. D. Barfoot, “The robust canadian traveler problem applied to robot routing,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5523–5529.

[10] H. Zhang, W. Tong, Y. Xu, and G. Lin, “The steiner traveling salesman problem with online edge blockages,” *European Journal of Operational Research*, vol. 243, no. 1, pp. 30–40, 2015.

[11] H. Zhang, W. Tong, G. Lin, and Y. Xu, “Online minimum latency problem with edge uncertainty,” *European Journal of Operational Research*, vol. 273, no. 2, pp. 418–429, 2019.

[12] V. Akbari and D. Shiri, “Weighted online minimum latency problem with edge uncertainty,” *European Journal of Operational Research*, 2021.

[13] H. Liu, H. Zhang, and Y. Xu, “The m-steiner traveling salesman problem with online edge blockages,” *Journal of Combinatorial Optimization*, vol. 41, no. 4, pp. 844–860, 2021.

[14] P. Eyerich, T. Keller, and M. Helmert, “High-quality policies for the canadian traveler’s problem,” in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[15] Z. W. Lim, D. Hsu, W. S. Lee, and W. Sun, “Shortest path under uncertainty: Exploration versus exploitation.” in *UAI*, 2017.

[16] R. A. MacDonald and S. L. Smith, “Active sensing for motion planning in uncertain environments via mutual information policies,” *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 146–161, 2019.

[28] G. Rubino, “Transient analysis of markovian queueing systems: a

[17] J. J. Chung, A. J. Smith, R. Skeele, and G. A. Hollinger, “Risk-aware graph search with dynamic edge cost discovery,” *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 182–195, 2019.

[18] F. Tsang, T. Walker, R. A. MacDonald, A. Sadeghi, and S. L. Smith, “Lamp: Learning a motion policy to repeatedly navigate in an uncertain environment,” *IEEE Transactions on Robotics*, pp. 1–15, 2021.

[19] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith, “Dynamic Vehicle Routing for Robotic Systems,” *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1482–1504, 2011.

[20] M. Pavone, E. Frazzoli, and F. Bullo, “Adaptive and distributed algorithms for vehicle routing in a stochastic and dynamic environment,” *IEEE Transactions on automatic control*, vol. 56, no. 6, pp. 1259–1274, 2010.

[21] C. Lee, K. Lee, and S. Park, “Robust vehicle routing problem with deadlines and travel time/demand uncertainty,” *Journal of the Operational Research Society*, vol. 63, no. 9, pp. 1294–1306, 2012.

[22] A. Prorok, “Robust assignment using redundant robots on transport networks with uncertain travel time,” *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 2025–2037, 2020.

[23] S. Choudhury, J. K. Gupta, M. J. Kochenderfer, D. Sadigh, and J. Bohg, “Dynamic multi-robot task allocation under uncertainty and temporal constraints,” *arXiv preprint arXiv:2005.13109*, 2020.

[24] V. Nguyen, P. Obermeier, T. C. Son, T. Schaub, and W. Yeoh, “Generalized target assignment and path finding using answer set programming,” in *Twelfth Annual Symposium on Combinatorial Search*, 2019.

[25] Z. Chen, J. Alonso-Mora, X. Bai, D. D. Harabor, and P. J. Stuckey, “Integrated task assignment and path planning for capacitated multi-agent pickup and delivery,” *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5816–5823, 2021.

[26] G. Grimmett and D. Stirzaker, *Probability and random processes*. Oxford university press, 2020.

[27] J. Medhi, *Stochastic models in queueing theory*. Elsevier, 2002.
survey with focus on closed forms and uniformization,” *Queueing Theory 2: Advanced Trends*, pp. 269–307, 2021.

[29] A. Sadeghi and S. L. Smith, “Heterogeneous task allocation and sequencing via decentralized large neighborhood search,” *Unmanned Systems*, vol. 5, no. 02, pp. 79–95, 2017.