# An empirical study of task bundling for sequential stochastic tasks in multi-robot task allocation

Changjoo Nam and Dylan A. Shell[*]

June 30, 2016

**Abstract**

This paper studies multi-robot task allocation in a setting where tasks are revealed sequentially and where it is possible to execute bundles of tasks. Particularly, we are interested in tasks that have synergies so that the greater the number of tasks executed together, the larger the potential performance gain. We consider tasks that are fed to robots for an infinite or indefinite time horizon. Robots may bundle multiple tasks to minimize some system cost (e.g., fuel), but doing so incurs an additional waiting time for bundling tasks. If the robots reduce their bundle size to minimize waiting time, task executions fail to make the most of possible synergies. Thus the system cost may increases, and the queue of waiting tasks may even overflow if task completions too slow. This paper is an analysis of bundling, giving an understanding of the important bundle size parameter. Based on qualitative properties of any multi-robot system that bundles sequential stochastic tasks, we propose multiple simple bundling policies. Experiments show how these polices perform in the multi-robot routing domain, showing that they are efficient compared to a baseline system where robots do not bundle tasks but iterate instantaneous assignments and executions of tasks.

## 1 Introduction

Multi-robot task allocation (MRTA) considers optimizing collective performance of a team of robots that execute a set of tasks. We consider a setting where

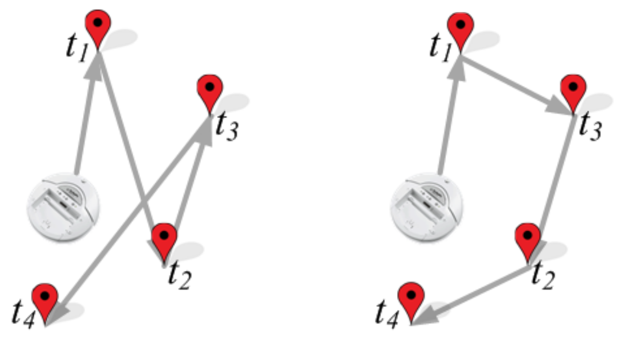
[*]Both authors are with the Department of Computer Science and Engineering at Texas A&M University, College Station, Texas, USA. cjnam@cse.tamu.edu

each robot performs one task at a time and each task requires only one robot to be performed, which falls under the single-task robot and the single-robot task category in the classic taxonomic description of such problems [1]. In the canonical formulation, the sets of robots and tasks are fixed, and a decision-maker (e.g., a central computation unit) has full access to all information about the tasks. Practically, it may be impossible to know the complete set of tasks beforehand, as tasks might be revealed sequentially through observations or upon arrivals. Such *online tasks* are seen in many applications (e.g., dial-a-ride, material handling for online orders, demining). Compared to the case where the task set is known *a priori*, relatively little MRTA work examines online instances in a way that involves reasoning about the arrival of yet to be revealed tasks [2].

In this paper, we study online tasks that are revealed sequentially and consider an infinite time horizon, with a focus on tasks with positive synergies. Serving tasks on an infinite time horizon needs a different performance metric other than the sum-of-cost which is conventionally used in MRTA. We consider two objectives which are the average system cost (e.g., fuel) per task and the average timespan of a task.[1] If multiple synergistic tasks are bundled together, then the total system cost of executing them is smaller than executing the tasks each independently. An obvious way to deal with these sorts of tasks is aggregating a set and then compute an allocation with the complete information for those tasks. Waiting for tasks to arrive in order to form a sufficiently large aggregation requires time which increases the timespan per task. This trade-off raises the question, *how many tasks should the robots bundle* over and above the standard question of *how should the tasks be allocated among robots*.

This paper explores the foundations of multi-robot task allocation for tasks which arrive online, sequentially, and are synergistic in nature. We begin with a qualitative study with the basic setting where a task is revealed deterministically (e.g, within some fixed interval), where cost of the task being serviced is a function of its location, and the location of the task is independently and identically drawn from a known probability distribution. Later, we extend our scope of study to consider probabilistic task arrivals (a Poisson process) and non-i.i.d. task distributions. The set of tasks need not necessarily be bounded, but the set of robots is assumed fixed. Tasks arrive concurrently with execution of tasks, so tasks accumulate while robots are execute previous bundles. Fig. 1 depicts one iteration of a routing example, showing the marked effect of bundles with different numbers of tasks. We explore the basic case where robots work independently without coor-

[1]In the domains where tasks are navigation, the cost and the timespan are equivalent to the distance traveled and the arrival time per task, respectively.



(a) The robot performs the tasks as soon as they are revealed.

(b) The robot waits for four tasks and performs them as a bundle.

Figure 1: A simple example: navigation tasks are revealed sequentially from $t_1$ to $t_4$. The robot performs tasks and loiters until a new task is introduced. In (a), the robot begins performing tasks right after they arrive. In (b), the robot waits until four tasks have been revealed, then finds a cheaper tour than (a).

dination and tasks arrive deterministically with an i.i.d. spatial distribution. Next, using information that describes stochastic tasks, we model the objective values analytically as functions of bundle size. These models yield an optimal bundle size for each objective. However, using the predicted (constant) bundle size becomes suboptimal once the modeling the assumptions are violated (e.g., robots are coordinated so they have a different model of the system cost, or the tasks do not have a regular arrival interval, the task distribution is not i.i.d.) To address this, we propose simple policies that work optimally in the base case and that also efficiently adapt to improve their performance for more complex settings.

This paper contributes a formulation of the problem of optimal task bundling for MRTA for sequentially revealed, synergistic tasks (Sec. 3.1). Within that formulation, we identify two objectives that describe two aspects performance, each depending on bundle size in a manner opposite the other. After analyzing the most basic scenario (Sec. 4.1), we introduce models that describe the objectives as a function of bundle size. Using these models, our study of iterated bundle executions leads us to propose simple and efficient bundling policies suitable for variations of the problem which generalize the basic instance. Evaluation of our policies (and comparison with a baseline sans-bundling) is carried out quantitatively with extensive experiments (Sec. 6).

## 2 Related work

Most previous work in MRTA with online tasks focus on the question of *how to allocate tasks*. Early work on auction mechanisms [3] and greedy allocation [4] studied the allocation of online tasks where the total number of tasks is known. Some recent work [5, 6] considers online tasks with unpredictable arrivals, but the option of bundling is not discussed. Online bipartite matching algorithms (e.g., [7]), which solve the underlying mathematical problem of the online MRTA, also do not consider bundling. Instead, they study how to match online vertices based on their stochastic information.

While bundling tasks has not received much attention, we are certainly not the first to propose the idea. Koenig et al. propose Sequential Single-Item (SSI) auctions with bundles in [8]. In the bidding phase, robots submit bids for bundles (i.e., subsets) of tasks from a known set of all tasks. The bidding phase and the winner determination phase iterate sequentially until all tasks are assigned. Compared to the standard parallel auctions, this method reduces the team's cost by exploiting synergies among tasks. The approach also reduces the time spent bidding compared to the standard combinatorial auctions since not all permutations of assignments are considered. Zheng et al. [9] propose SSI with roll-outs where single tasks are auctioned in each iteration, but the cost of each task is evaluated together with the previously allocated tasks to each robot in order to exploit synergies. Heap and Pagnucco [10] extended SSI to bundles with Sequential Single-Cluster (SSC) auctions where the robots bid on clusters of tasks, formed through a $k$-means clustering algorithm.

Prior work with synergistic tasks uses two objectives MiniSum and MiniMax that are analogous to our *system cost per task* and the *timespan of a task*, respectively. These works have the same objective, namely, seeking the maximal synergy and the minimal timespan. However, the prior work considers a fixed, finite set of tasks and, as will become clear in the next section, considering an unbounded (and unremitting) sequence of tasks constrains the set of solutions because one must ensure that the task buffer does not overrun. In general, the prior work does not identify or have to address the consequences of bundling which worsens time-related objectives. But when tasks arrive in an online manner, there is a true trade-off that must be made; this aspect is absent from previous investigations. The present work also treats synergistic tasks in rather more sophisticated manner, providing an explicit model that quantifies the improvement in objective value.

## 3 Problem description

This section formulates the problem mathematically, expresses constraints on the problem, and describes the objectives to be minimized. The multi-robot routing scenario is used as an example.

### 3.1 Problem formulation

Given a set $\mathbf{R} = \{r_1, \cdots, r_n\}$ of $n$ robots, every $\alpha > 0$ time interval, a task $t_j$ arrives and is enqueued along with other waiting tasks in a structure $\mathbf{T}$. The total number of tasks is unknown and it could be an unbounded sequence. Here $\alpha$ could be deterministic or a random variable—in the latter case, we use $\alpha$ to denote the mean of a distribution. We assume that robots share all available task information (i.e., $\mathbf{T}$) for example, through some communication network.

We model tasks by thinking about the costs associated with their performance—in order to make this concrete we will assume that the application entails mobile robots and the cost is a function of the locations of the robot and task. We assume that locations of tasks are drawn from a probability distribution, and this has the appropriate relationship on costs. Let $c(S)$ be the cost of performing the set of tasks $S$. We consider tasks with the property: for $S_1$ and $S_2$ where $S_1 \neq S_2$, $c(S_1) + c(S_2) > c(S_1 \cup S_2)$. In other words, performing multiple tasks together has the potential to lump some common work together and the cost of performing a bundle of tasks is sub-linear in bundle size, i.e., smaller than the sum of the costs when executed independently.

Robot $r_i$ forms its own bundle $\mathbf{X}_i$ by extracting tasks from $\mathbf{T}$, where $|\mathbf{X}_i|$ would change. Once $t_j$ is assigned to $\mathbf{X}_i$, it is no longer available to other robots unless $r_i$ releases the task. Tasks continue to arrive while robots perform the work assigned to them, that is, that which is within their respective bundles. The robots iterate bundling and executing tasks in turn. Depending on the number of tasks in $r_i$'s bundle, $r_i$ may be idle while waiting to fill $\mathbf{X}_i$, which we denote $r_i \in \mathbf{R}_{\text{idle}}$. Otherwise, $r_i \in \mathbf{R}_{\text{active}}$. Note that $\mathbf{R} = \mathbf{R}_{\text{idle}} \cup \mathbf{R}_{\text{active}}$ and $\mathbf{R}_{\text{idle}} \cap \mathbf{R}_{\text{active}} = \varnothing$.

Strategies for assigning tasks to robots make use of flexibility in (*i.*) making the choice of whom to assign to a certain task, and (*ii.*) when to assign the task. In general, waiting increases the available opportunities to optimize performance but, waiting itself, induces delays. Since (*ii.*) is a central consideration in the present work, it is important to delineate the requirements of the strategies for assigning tasks. We do this by noting two necessities for the performance of online tasks:

- *Unconditional Task Acceptance:* Any task that arrives, must be enqueued to $\mathbf{T}$.
- *Non-starvation:* No task may be abandoned to remain in $\mathbf{T}$ indefinitely.

Subject to fulfilling those two requirements, we consider two objectives to minimize. Since there is no fixed set of tasks in an infinite (or a very long) length horizon, the conventional sum-of-cost measure is no longer ideal. More meaningful are the average values of the following:

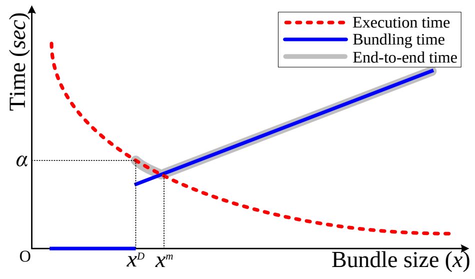
1. An important metric is the cost incurred by a robot to perform a task. Let $c_{ij} \in \mathbb{R}^{\geq 0}$ represent the cost of $r_i$ performing $t_j$, then the objective is the average cost spent by a robot to finish a task, $\bar{c}$. The average is taken across all robots and task pairs; we call this the *system cost*.
2. The *timespan*, or *end-to-end time*, of a task $\tau_j \in \mathbb{R}^{\geq 0}$ metric represents the elapsed time from the moment task is inserted to $\mathbf{T}$ until its completion. The objective is the average timespan of a task, $\bar{\tau}$, taken across all tasks.

## 3.2 An example: the multi-robot routing problem

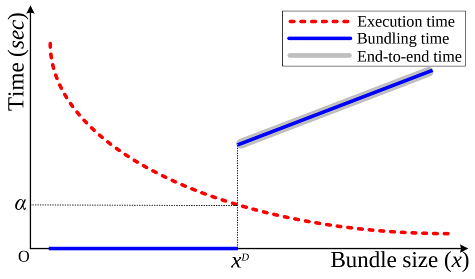
The multi-robot routing problem is a representative example of the setting we describe since it involves synergistic tasks that arrive online. It is also of natural interest since it includes features common to many application domains. Precisely, the tasks require that some robot visit a location. The locations are revealed sequentially and the goal is to visit all locations (as shown in Fig. 1) while minimizing the average time traveled $\bar{c}$, or the average task end-to-end time $\bar{\tau}$, or some combination of the objectives. A robot visits only one location at a time, and the task requires that one robot arrive. A robot bundling multiple tasks may then plan a Hamiltonian path. To solve the problem, the robots need a policy that determines how many locations they ought to bundle and how they should cooperate together to decide which robot should visit which locations. This is a running example throughout the paper because it is sufficiently rich to explore the fundamental properties underlying strategic bundling.

## 4 An analysis of bundle size

This section develops models for the objective values defined in the previous section. In the basic setting, robots work independently, and tasks are revealed



(a) The case where $x_g^* = x^m$.



(b) The case where $x_g^* = x^D$.

Figure 2: Illustrative functions that describe the average system cost (red) and timespan (gray) per task. The optimal bundle size for the system cost $x_f^*$ is unbounded for both (a) and (b) since $f(\cdot)$ is strictly decreasing owing to the synergies among tasks. $s(\cdot)$ is infinite for $x < x^D$ and the same with $f(\cdot)$ otherwise. There exists a finite bundle size $x$ that makes $g(\cdot)$ minimum. $g(\cdot)$ for $x < x^D$ is not shown since the value is infinite.

with a fixed interval. For these simple models, it is possible to find analytic expressions for the optima. Next, we introduce some complexity into the model exploring, empirically, how coordination methods affect outcomes.

## 4.1 The base case: independent robots

### 4.1.1 The general model

In the base case, robots do not coordinate amongst themselves to exploit task locality *between* bundles, rather they bundle and execute tasks independently. Ordering of tasks *within* their bundles is optimized locally and depends on the path they construct. We assume stochastic tasks with locations independently and identically distributed from a Uniform distribution over a sub-region of the plane with area $S$. Also, a new task is revealed every $\alpha$ seconds. Steady-state models of both the system cost $\bar{c}$, which is the average system cost (execution time) per task, and the average timespan (end-to-end time) $\bar{\tau}$ of a task, are constructed next. It may be helpful to refer to Fig. 2 during the exposition.

A model for $\bar{c}$ is given by $f(x|S, v)$ where $x$ is the bundle size and $v$ is the task performance rate (e.g., velocity) of a robot (the red curve in Fig. 2). Task synergies imply that $f(\cdot)$ is decreasing and, hence, the bundle size that minimizes the system cost is infinite ($x_f^* = \infty$). The functional $h(x|\alpha, n, f)$ describes the average time that a task stays awaiting sufficient tasks to been enqueued to form a complete bundle. Since tasks are added into $\mathbf{T}$ and are distributed to $n$ robots,

$h(\cdot)$ increases when more robots are bundling, for a fixed $\alpha$. But note also that $h(\cdot)$ is discontinuous since $h(x|\alpha, n, f) = 0$ if $x < x^D$ (the blue line in Fig. 2). Here the quantity $x^D$ denotes the bundle size when $f(x|S, v) = \alpha$, that is, the point of equilibrium between the rate of task arrivals and (average) executions. Below $x^D$, $|\mathbf{T}|$ diverges so there is no steady-state (tasks keep being accumulated) and robot take tasks out from $\mathbf{T}$ without waiting, so the bundling time is zero. For $x \geq x^D$, tasks do not accumulate in the queue $\mathbf{T}$, and a robot must wait for tasks to arrive in order to fill its bundle and so the bundling time is nonzero. Thus, $h(x|\alpha, n, f) = h'(x|\alpha, n)$ for $x \geq x^D$, where $h'(\cdot)$ represents the bundling time without considering the potential overflow of $\mathbf{T}$. There is another component $s(x|\alpha, f)$, the residing time of a task in $\mathbf{T}$, which is the time spent by a task before any robot has returned and begun to assemble its next bundle. We have $s(x|\alpha, f) = \infty$ for $x < x^D$. Otherwise, $s(x|\alpha, f) = f(\cdot)$ since tasks only stay in $\mathbf{T}$ while robots are execute their bundles.

A model of $\bar{\tau}$ is given by $g(x|S, v, \alpha, f)$:

$$g(x|S, v, \alpha, f) = \max(f(x|S, v), h(x|\alpha, n, f), s(x|\alpha, f)), \qquad (1)$$

which are the thick gray curves in Fig. 2. $g(\cdot)$ for $x < x^D$ is not shown since the value is infinite. The $x$ value that makes $g(\cdot)$ minimum is the optimal bundle size $x_g^*$. Determining $x_g^*$ consists of two possible cases, shown in Fig. 2(a) and Fig. 2(b). In Fig. 2(a), $x^m$ is the equilibrium between the task bundling time and the execution time. At $x^m$, the robot finishes executing a bundle when the next bundle has just filled. Thus, $x_g^* = x^m$ because $g(x^m|\cdot)$ takes the minimum at this point. It is important to note that there is no waiting time between iterations. In Fig. 2(b), $x^m$ does not exist because the tasks in $\mathbf{T}$ overflow. At $x^D$, the execution time dominates the zero bundling time, and $f(x^D|\cdot)$ has the minimum, so $x_g^* = x^D$. Practically, $x_g^*$ is computed by $\max(x^{m'}, x^D)$ where $x^{m'}$ is the intersection between $f(\cdot)$ and $h'(x|\alpha, n)$.

### 4.1.2 The multi-robot routing example

Since the pioneering work of Beardwood et al. [11], there has been extensive research on computing the optimal length of the tour in random instances of the traveling salesman problem (TSP). The early models in [11, 12] are simple but limited to the asymptotic behavior. Lee and Choi [13] proposed a more accurate model given a finite number of cities. The multi-robot routing problem aims to optimize the tour of each robot, which involves optimizing the Hamiltonian path

(HP)—a special case of the TSP where the return tour to the start location is unnecessary. We modify the model in [13] for the HP. The system cost (time traveled) per task is

$$\begin{aligned} & f(x|S,v) \\ &= \left(\frac{(0.7211\sqrt{x+1}+0.604)\sqrt{S}-E_d}{v(x+1)}\right)\cdot(\beta\log x+1) \end{aligned} \tag{2}$$

where $S$ is the area of a rectangular field and $v$ is the velocity of the robot. The scalar $E_d$ is the expected distance between two points drawn from a uniform distribution, representing the last visited location and the initial location (notice that the initial location is also random because it is the last visited location from the previous batch). The expression $0.7211\sqrt{x+1}+0.604$ from the model in [13] yields the optimal length of a TSP tour with $x$ locations,[2] and $E_d$ is subtracted because a HP does not include the return trip to the initial location. The entire expression is divided by the number of locations (per task) and scaled by $\frac{\sqrt{S}}{v}$ (length/velocity). However, $E_d$ is not scaled by $\sqrt{S}$ since it already includes the size of the area as a variable. A scaling factor, $\beta$, reflects how close the algorithm used is to optimal, where $\beta = 0$ for an optimal algorithm with larger values for practical suboptimal algorithms.

A task waits in a bundle (size of $x$) for $x\alpha - j\alpha$ seconds where $j$ is the time when the task is inserted. Then, the sum of the bundling time for all tasks is $\sum_{j=1}^{x} x\alpha - j\alpha = x^2\alpha - \frac{x(x+1)}{2}\alpha = \frac{\alpha}{2}x(x-1)$. And the function describing the bundling time per task is

$$h(x|\alpha,n,f) = \begin{cases} 0 & \text{if } x < x^D \\ h'(x|\alpha,n) = \frac{n\alpha}{2}(x-1) & \text{otherwise.} \end{cases} \tag{3}$$

Note that $n$ is multiplied since tasks are distributed to $n$ robots. Interestingly, $h'(\cdot)$ is a special case of the mean residual life of a customer in a renewal process presented in [14]. The residual life is the amount of time that the customer must wait until being served. The general form of (3) when task arrivals follow a Poisson process is

$$h'(x|\alpha,n,\lambda) = \frac{n\alpha}{2}\left(1+\frac{\lambda}{\alpha^2}\right)(x-1) \tag{4}$$

where $\lambda$ is the variance of the arrival interval. If $\lambda = 0$, then (4) reduces to (3).

[2] The salesman starts at one of the locations, but the robot starts from the last location in the previous batch in our case. Thus, the total number of locations is $x+1$.

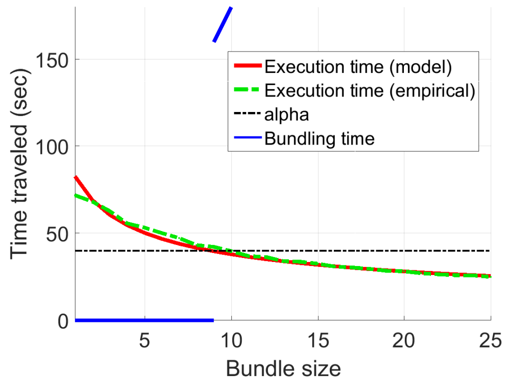


Figure 3: The models (2) (red dotted) and (3) (blue solid). The horizontal line represents $\alpha$. The greed curve shows the experimental result from a heuristic HP algorithm which is attached in the supplementary material.

Fig. 3 shows (2) (red) as a function of bundle size ($x$) along with values from experiments (green) with values $\alpha = 10$, $v = 1\,m/s$, and $S = 150\,m \times 150\,m$. The blue line represents (3). We implemented a simple heuristic Hamiltonian path algorithm to explore this model[3], and $\beta = 0.0542$ was empirically determined for our algorithm.

## 4.2 Coordinated robots

Next, we turn to coordinated robots. There are two major considerations in thinking about the objective values for teams of closely coupled robots. The first consideration is what task allocation method will be used to distribute tasks from $\mathbf{T}$ to the robots (e.g., assignment algorithms, integer programming methods, auction-based algorithms, or etc.)[4]. Our implementation uses integer linear programming (ILP) to optimally distribute tasks based on the distances between robots and tasks. The second consideration is the degree of synchronization in the team for the task distribution. Specifically, we need to decide how many robots are included when distributing tasks. If the robots that have completed their tasks wait until other robots become free, the distribution of tasks among them can make maximal use of their spatial dispersion. If the robots do not wait, the tasks they are assigned will suit them individually, being slightly myopic. Waiting for robots

[3]The algorithm is attached as a supplementary material.

[4]In some domains, combinatorial approaches have been studied to solve the task distribution and optimization together (e.g., the multiple TSP [15]).

to finish their tasks allows more robots to participate in the assignment, imposing greater synchronization on the robots. But, in a way analogous to the advantages of large task bundles over small ones, it gives more opportunity for the optimizer to find savings.

Modeling this coordinated case is possible when one has domain knowledge of the sort used to build the model in Sec. 4.1.2. However, the necessary domain knowledge is not always available so, as an alternative, one may fit a function to empirical data. We are not aware of any model describing the system cost of the coordinated robot team in the multi-robot routing problem. Fitting a function may require extensive experiments be performed, which may be tedious or expensive. Depending on the coordination method used, one can still draw inferences on how the system cost changes by examining the model of the base case, and making adjustments for coordinated case from a empirical data.

In Fig. 4, we show measured values of $f(\cdot)$ when robots are coordinated through an ILP (averaged over 10 repetitions). The x-axis and y-axis represent the bundle size ($x$) and the degree of synchronization ($n_{\text{ILP}}$). Given 5 robots, $n_{\text{ILP}}$ ranges from 1 (completely asynchronous) to 5 (completely synchronized). Fig. 4(b) shows the system cost across different bundle sizes. The uppermost line (lime green) shows the cost when $n_{\text{ILP}} = 1$. The lower-most line (brown) describes the case where $n_{\text{ILP}} = 5$. The result shows that a synchronized team outperforms asynchronous robots[5]. There are two reasons why synchronous robots perform better even though additional idle time is incurred in waiting for other robots. First, as already alluded to, including more robots in the ILP makes the resulting assignment globally optimal with the current tasks in $\mathbf{T}$. Secondly, while the robots wait for other robots, they are simultaneously getting more options for tasks because new ones keep arriving. Since the ILP includes all tasks in $\mathbf{T}$ and $\mathbf{R}_{\text{idle}}$, the chance of lowering the cost per task increases as $\mathbf{T}$ grows.

On the other hand, there appears to be an anomaly in Fig. 4 as, when $x$ is small, the cost does not decrease monotonically with increasing $x$. It indicates that bundling two tasks is worst than not bundling them. The transition between two lines in Fig. 4(c) comes from the synchronization and the optimal task distribution. The oddity is not observed in the independent and synchronized robot team. It is the optimal task distribution which explains the oddity. When the bundle size is small, tasks overflow $\mathbf{T}$. Their locations are uniformly distributed in space, so that as $\mathbf{T}$ overflows, robots are able to find one or two tasks with very small costs.

[5]The result from the independent robots case is not shown, but a synchronized team performs better than it too.

As the bundle size increases, tasks are removed more quickly and the density of tasks decreases, and the distance per task inevitably returns to normal. In the independent team, the tasks are randomly distributed to robots so the dense tasks make for no change.

By bringing these insights together one comes to the conclusion that, for the setting being examined, the base case is an upper bound of the system cost, across all combinations of coordination methods and degree synchronization. Fig. 4(c) shows the base case (upper black line) and the coordinated robots case where $n_{\text{ILP}} = 5$ (lower brown line). Thus, the base case provides a model that overestimates cost in other cases. If we compute $x_g^*$ using the basic model, $x_g^*$ is larger or equal to the actual optimal bundle size (e.g., the red curve in Fig. 2 moves below, so $x^m$ or $x^D$ decrease). Without deeper domain knowledge, one cannot know the exact $x_g^*$ but the range is determined from the basic model.

## 4.3 Elements of task stochasticity

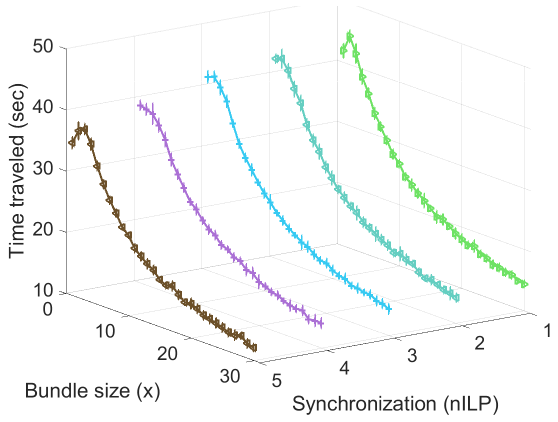
Tasks locations and arrivals include uncertain elements that induce a gap between the model and the performance observed in the system. Next, we address the issues arising from stochasticity in tasks; it motivates our introduction of bundling policies that adapt to circumstances, and can be useful when the ideal models fails to capture some aspects of the system.

### 4.3.1 Task locations

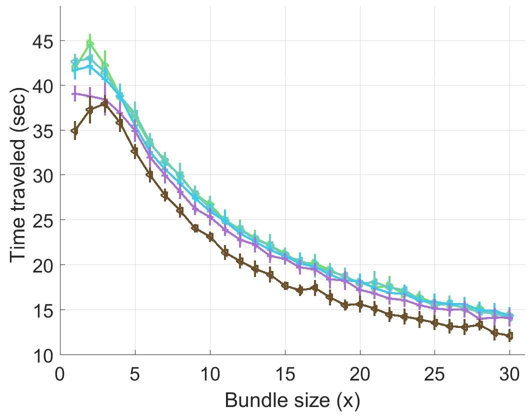
Owing to the stochasticity of task locations, the system's cost described thus far should be those of as describing the mean of a random variable. The particular realizations will differ from this average value. In practice, if a robot completes its bundle of tasks faster than the average execution time, the next bundle will not be completely filled yet. Thus the actual $x_g^*$ would differ from the value in the basic model.

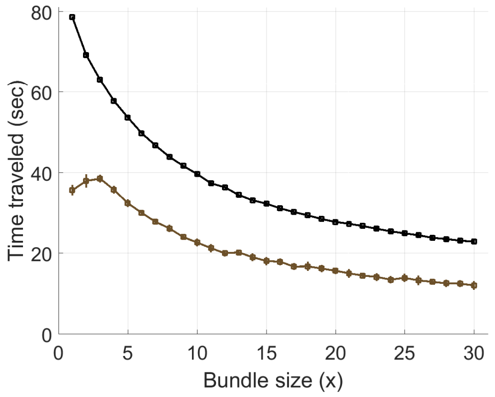
### 4.3.2 Task arrival process

Though some applications certain have tasks are revealed at fixed intervals, a probabilistic arrival process is more common. One generalization is to consider a Poisson arrival process, modeling completely random arrivals of events. Probabilistic arrivals will alter the true $x_g^*$ so that it differs from the one in the basic



(a) The system cost along $x$ and $n_{\text{ILP}}$.





(b) The system cost of different $n_{\text{ILP}}$ along $x$. (This view simply projects $n_{\text{ILP}}$ out of Fig. (a).)

(c) The comparison between the base case and the case of coordinated robots with $n_{\text{ILP}} = 5$.

Figure 4: Empirical results of a team of five robots. In (b), it is shown that synchronization improves the performance (the uppermost is the case where $n_{\text{ILP}} = 1$ and the lower-most has $n_{\text{ILP}} = 5$). (c) shows that the basic model from Sec. 4.1 is the upper bound of all combinations of coordination methods, synchronization.

model because $h(\cdot)$ is no longer deterministic (e.g., the blue line in Fig. 2 has some variance).

## 5 Bundling policies

Based on the previous discussion, this section proposes some bundling policies, while the following section provides through evaluations. Each of the policies are simple algorithms that perform optimally in the basic case, but provide broader support, being flexible enough to behave agreeably across a range of more complex cases.

### 5.1 Model-based policies

#### 5.1.1 Fixed-$x$ policy

If the models $g(\cdot)$ and $h(\cdot)$ are available, finding an $x$ that minimizes (1), the end-to-end time, gives $x_g^*$. Also, finding the $x$ that minimizes $h(\cdot)$, the system cost, gives $x_f^*$. Let $k$ be the bundle size that each robot takes (it may differ for different robots if their performance rates differ, such as having different velocities). Each robot keeps $k = x_g^*$ or $k = x_f^*$ depending on its objective. Since $x_f^*$ goes to infinity with synergistic tasks, this is not a practical bundle size. Thus, we only consider $k = x_g^*$. When a robot finishes its current bundle and tries to execute the next bundle, there might be insufficient tasks in $\mathbf{T}$ to form that bundle. The robot may wait, idly, for new tasks. Let $x_{\mathrm{P}}$ denote the size of $\mathbf{T}$ which triggers execution of the current bundle. In the fixed-$x$ policy, $x_{\mathrm{P}} = x_g^*$. If $|\mathbf{T}| \geq x_{\mathrm{P}}$, the robot takes $k$ tasks and executes them immediately. Because this policy cannot handle uncertainties in the task profile (discussed in Sec. 4.3) it is possible that end-to-end time could diverge if $\mathbf{T}$ overflows.

#### 5.1.2 Up-to-$x$ policy

This policy is the same with the fixed-$x$ policy except that the robot does not wait to fill its bundle. In every iteration, the robot takes $k$ tasks from $\mathbf{T}$, where $k \leq x_g^*$, if $|\mathbf{T}| \geq x_{\mathrm{P}} = 1$. This eliminates the bundling time.

## 5.2 Remarks

As we discussed in Sec. 4.2, when the robots are coordinated and/or synchronized, then the $x_g^*$ computed from the basic model differs from the actual optimal bundle size. Since $x_g^*$ overestimates the true value, the consequence is that it may increase the end-to-end time to some degree, but it never causes $\mathbf{T}$ diverge. The sub-optimality in end-to-end time is somewhat allayed by having a better system cost (from the larger task bundles).

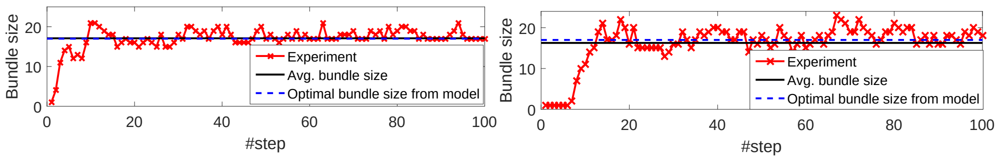
## 5.3 Model-free policies

### 5.3.1 Sweeping policy

The sweeping policy takes all tasks $k = |\mathbf{T}|$ if $|\mathbf{T}| \geq x_\mathrm{P} = 1$. Since the previous two polices cannot handle the case where the robots complete their current bundles earlier than the expectation, or tasks arrive faster than the mean interval. These two types of behavior may happen due to fluctuations away from the mean for some period of time. This policy saves bundling time when the number of tasks is insufficient. Also, the policy exploits the synergies maximally by taking all available tasks. Most importantly, the policy is useful even when the models are unavailable. The bundle size converges to a value reflecting the equilibrium in which the execution time and bundling times are equal (in average). Fig. 5(a) shows the changes of the bundle size versus time when task locations are uniformly distributed and tasks arrive regularly.

### 5.3.2 Averaging policy

The sweeping policy's equilibrium is constant unless the stochastic properties of the task location and arrival process change. The sweeping policy does not make explicit use of a representation of the equilibrium. It is worthwhile to modify to track the equilibrium via history. The averaging policy begins with $x_\mathrm{P} = 1$ and repeats the following: if $|\mathbf{T}| \geq x_\mathrm{P}$, then the robot records the current $|\mathbf{T}|$ in the history window $W$ (i.e., $W \leftarrow \textsc{Enqueue}(W, |\mathbf{T}|)$). It then takes $k = x_\mathrm{P}$ tasks and leaves the remaining tasks behind in $\mathbf{T}$. Next, a new $x_\mathrm{P}$ is computed by averaging the previous values saved in $W$ (i.e., $x_\mathrm{P} = \textsc{Mean}(W)$). The smaller the window size, the more sensitive the policy to variability.



(a) The sweeping policy (b) The averaging policy

Figure 5: Plots showing bundle size vs. iterations. The policies converge to the optimal bundle predicted from the model.

## 5.4 Algorithm

We develop a multi-robot routing algorithm with a centralized task distribution mechanism. Pseudocode appears in Alg. 1, using of the fixed bundle-size policy. In the initialization (lines 1–8), the bundle $\mathbf{X}_i$ of robot $i$ is filled with the current location of the robot, and the optimal bundle size $x_i^*$ is computed. After the initialization, the algorithm iterates lines 10–31 infinitely. Lines 10–17 run for those robots that execute their Hamiltonian paths. Line 11 executes the path and removes visited locations from the bundle. A robot transitioning into the idle state in lines 12–16. The algorithm allocates tasks to idle robots in each step (line 19). If there are available tasks and idle robots, the bundles are updated through ALLOCATE function.

ALLOCATE computes an optimal assignment of tasks in $\mathbf{T}$ to the idle robots in $\mathbf{R}_{\text{idle}}$, where $|\mathbf{T}| = m$ and $|\mathbf{R}_{\text{idle}}| = n'$. One or more tasks are assigned to each robot based on the cost $c_{ij}$ where $c_{ij} = ||l(r_i) - l(t_j)||$. Let $y_{ij}$ be a binary variable that equals to $0$ or $1$, where $y_{ij} = 1$ indicates that $r_i$ performs $t_j$, and $y_{ij} = 0$ elsewhere. Then a mathematical description of the assignment problem is

$$\min \sum_{i=1}^{n'} \sum_{j=1}^{m} c_{ij} y_{ij} \tag{5}$$

subject to

$$\sum_{i=1}^{n'} y_{ij} = 1 \qquad \forall j, \tag{6}$$

$$y_{ij} \in \{0, 1\} \qquad \forall \{i, j\}. \tag{7}$$

(6) prohibits a task to be assigned to multiple robots.

After computing the assignment, $\forall j$, with $y_{ij} = 1$, the associated $t_j$ is added to $\mathbf{X}_i$. Those assigned tasks are removed from $\mathbf{T}$ (line 20). Lines 22–31 are policy dependent. The lines run for all idle robots, which wait until the condition for the policy is satisfied. After that, robot $i$ becomes active, computing a Hamiltonian path from the locations in its batch (lines 27–29).

## 6 Quantitative study: comparisons of the policies

In this section, we describe experiments that examine the various polices on the multi-robot routing problem. The coordination method called 'IND' randomly distributes tasks from $\mathbf{T}$ to robots. As to the dimension of synchronization: asynchronous robots do not wait for other robots (but may be included with other robots by chance). In contrast, synchronized teams works as a block. We also include numbers from a baseline where robots do not bundle but instantaneously execute tasks one by one.

### 6.1 Experimental settings

For a fixed number of robots ($n = 5$), we assume that all robots move at the same velocity $v$, and the bundle size is computed from the model is the same for all robots. As discussed above, using $x_f^*$ is unrealistic since it would make robots wait for an unbounded number of tasks. Thus, we minimize the end-to-end time only, and scrutinize how the system cost changes. We set $\alpha = 5$ for the regular task arrival process. The parameter for the Poisson arrival process is $\lambda = \frac{1}{\alpha}$, where the mean arrival interval is $\lambda^{-1} = \alpha$. Those two arrival processes have the same mean task arrival interval. We measure the two objective values and run 10 repetitions. The results appear in Table 1 and Fig. 6.

### 6.2 Analysis

The results show that all bundling polices in all combinations represent significant improvements over the non-bundling baseline. The improvement in the time traveled is a consequence of larger bundles although we aim to optimize the end-to-end metric. We focus on the end-to-end time in the following analysis. Table 1(a) and Table 1(b) show the results of the fixed task arrival interval case and the Poisson arrival case, respectively.

**Algorithm 1** M-HP

**Input:** the number of robots $n$, the velocity of robots $v_i$, the area of the field $S$, the task arrival interval $\alpha$

**Output:** Continuous executions of $\mathbf{X}_i$

```
1 $\mathbf{T} = \varnothing$//the set of tasks
2 $\mathbf{R}_{\text{active}} = \varnothing$//the set of working robots
3 $\mathbf{R}_{\text{idle}} = \{r_1, \cdots, r_n\}$//the set of idle robots
4 $\mathbf{TP}_i = \varnothing$//a temporal set
5 for each $r_i \in \mathbf{R}_{\text{idle}}$
6   $\mathbf{X}_i = \{l(r_i)\}$//$l(\cdot)$: the location of the input
7   Compute $x_i^* = \max(x^m, x^D)$
8 end for
9 while uninterrupted
10  for each $r_i \in \mathbf{R}_{\text{active}}$
11    Execute $\mathbf{X}_i$ and remove visited locations
12    if $\mathbf{X}_i = \varnothing$//no more location to visit
13      $\mathbf{R}_{\text{active}} = \mathbf{R}_{\text{active}} \setminus r_i$
14      $\mathbf{R}_{\text{idle}} = \mathbf{R}_{\text{idle}} \cup r_i$
15      $\mathbf{X}_i = \{l(r_i)\}$//the start location of next $\mathbf{X}_i$
16    end if
17  end for
18  if $|\mathbf{R}_{\text{idle}}| \geq n_{\text{ILP}}$
19    $\mathbf{X}_{1,\cdots,|\mathbf{R}_{\text{idle}}|}$ = ALLOCATE($\mathbf{T}$, $\mathbf{R}_{\text{idle}}$)//solve (5)-(7)
20    $\mathbf{T} = \mathbf{T} \setminus (\mathbf{X}_1 \cup \cdots \cup \mathbf{X}_{|\mathbf{R}_{\text{idle}}|})$//remove allocated tasks
21  end if
22  for each $r_i \in \mathbf{R}_{\text{idle}}$
23    $\mathbf{TP}_i = \mathbf{TP}_i \cup \mathbf{X}_i$//merge the bundle with
                          //the remaining tasks in $\mathbf{TP}_i$
24    if $\mathbf{TP}_i \geq x_{\text{P}}$//$x_{\text{P}}$: policy-dependent parameter
25     $\mathbf{X}_i = \mathbf{TP}_i(1 : k)$//$k$: policy-dependent bundle size
26     $\mathbf{TP}_i = \mathbf{TP}_i(k + 1 : \text{end})$
27     $\mathbf{X}_i$ = RANDHP($\mathbf{X}_i$)//compute an HP
28     $\mathbf{R}_{\text{idle}} = \mathbf{R}_{\text{idle}} \setminus r_i$
29     $\mathbf{R}_{\text{active}} = \mathbf{R}_{\text{active}} \cup r_i$
30    end if
31  end for
32 end while
```

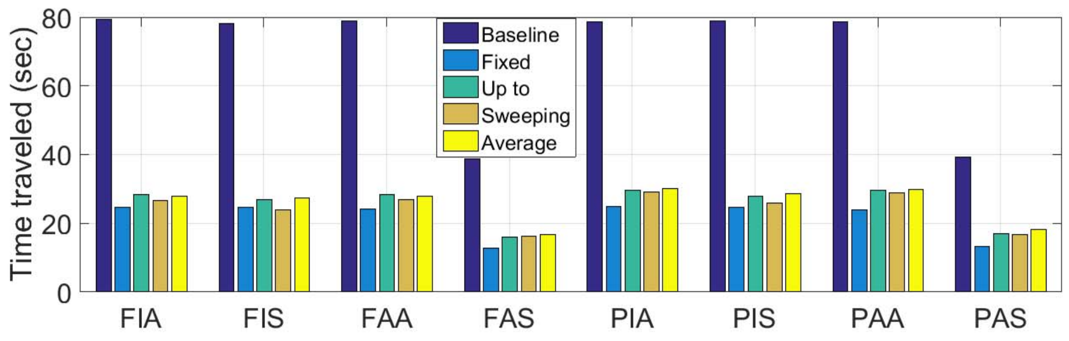
Table 1: Comparisons of policies. The values represent the mean and the standard deviation over 10 repetitions.

(a) Fixed task arrivals ($\alpha = 5$)

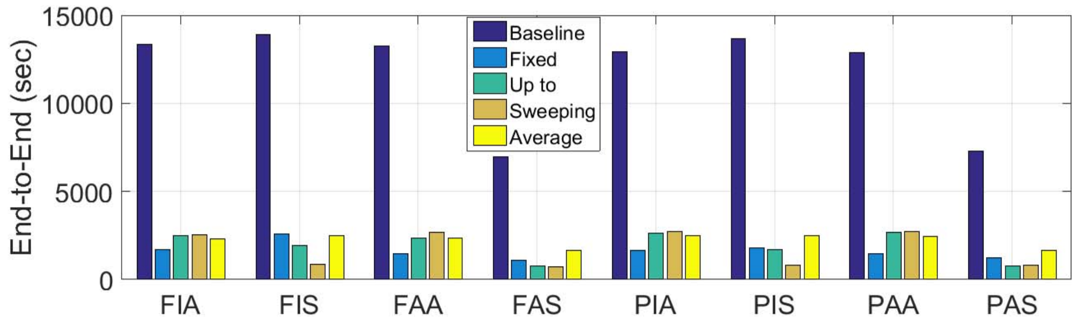
| | Deg. of Sync | Time traveled | | End-to-end | |
|---|---|---|---|---|---|
| | | IND | AS | IND | AS |
| Baseline | Async | 79.31 (1.073) | 78.77 (1.135) | 13300 (223.7) | 13210 (212.0) |
| | Sync | 78.09 (1.088) | 38.53 (0.9862) | 13890 (340.1) | 6939 (479.8) |
| Fixed $x$ | Async | 24.60 (0.3407) | 24.07 (0.1681) | 1696 (107.6) | 1434 (65.54) |
| | Sync | 24.45 (0.3596) | 12.70 (0.5858) | 2557 (177.7) | 1059 (166.2) |
| Up to $x$ | Async | 28.27 (0.5257) | 28.25 (0.4188) | 2476 (166.6) | 2344 (163.1) |
| | Sync | 26.84 (0.1947) | 15.74 (0.4869) | 1926 (165.0) | 726.3 (126.0) |
| Sweeping | Async | 26.58 (0.1870) | 26.72 (0.2104) | 2499 (160.5) | 2668 (201.7) |
| | Sync | 23.78 (0.6062) | 16.05 (0.9959) | 834.9 (51.30) | 714.96 (192.1) |
| Averaging | Async | 27.63 (0.4369) | 27.79 (0.2985) | 2266 (184.6) | 2321 (116.9) |
| | Sync | 27.18 (0.8579) | 16.60 (0.8263) | 2455 (151.3) | 1650 (127.6) |

(b) Poisson task arrivals ($\lambda = 1/5$)

| | Deg. of Sync | Time traveled | | End-to-end | |
|---|---|---|---|---|---|
| | | IND | AS | IND | AS |
| Baseline | Async | 78.49 (1.111) | 78.59 (0.7917) | 12910 (220.7) | 12880 (152.4) |
| | Sync | 78.65 (0.7509) | 39.24 (1.301) | 13670 (304.0) | 7264 (652.2) |
| Fixed $x$ | Async | 24.65 (0.2831) | 23.83 (0.3252) | 1613 (109.7) | 1434 (70.31) |
| | Sync | 24.49 (0.5503) | 13.03 (0.5994) | 1787 (322.0) | 1216 (286.6) |
| Up to $x$ | Async | 29.48 (0.6724) | 29.56 (0.4917) | 2610 (195.0) | 2639 (257.5) |
| | Sync | 27.75 (0.6123) | 16.91 (0.6896) | 1688 (129.4) | 760.3 (199.5) |
| Sweeping | Async | 29.02 (0.6002) | 28.76 (0.5024) | 2702 (236.6) | 2686 (231.7) |
| | Sync | 25.75 (0.4472) | 16.66 (0.9694) | 801.4 (51.73) | 812.8 (208.1) |
| Averaging | Async | 29.97 (0.5120) | 29.73 (0.5907) | 2475 (190.7) | 2402 (216.3) |
| | Sync | 28.59 (1.404) | 17.98 (0.8220) | 2473 (158.5) | 1615 (227.2) |



(a) The time traveled (the system cost).



(b) The end-to-end time (the timespan of a task).

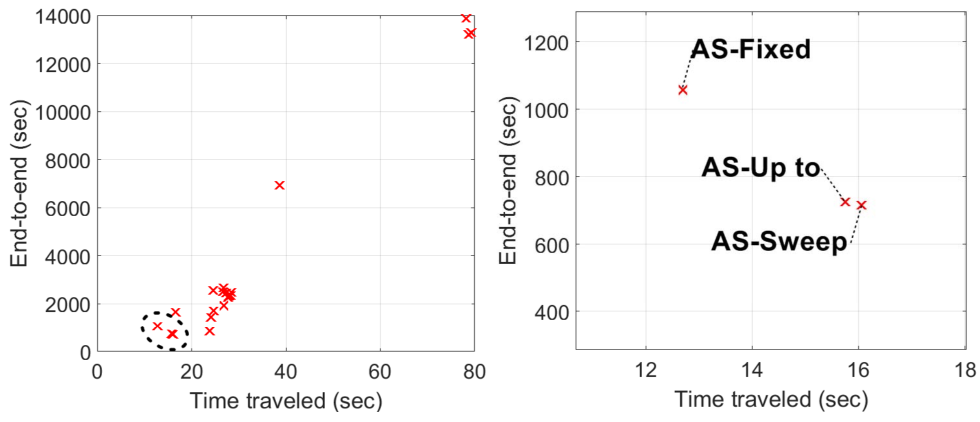
Figure 6: Comparisons of policies with all combinations of the arrival process (F̲ixed or P̲oisson), the coordination method (I̲ndependent or A̲ssignment), and the degree of synchronization (A̲sync or S̲ync). Three letters represent the combination.

In both cases, synchronizing the robots (Sync) with an optimal task distribution mechanism (AS) significantly increases performance. As discussed, this is because including all robots in AS results in a globally optimal assignment solution. If the robots bundle tasks, Sync gains more tasks while the robots wait for each other. This increases the chance of having lower-cost tasks in AS. Bundling improves the performance as well. Bundling reduces the execution time which is one of the components of the end-to-end measure (the residing time in $\mathbf{T}$, the bundling time, and the execution time). If robots finish tasks faster, the residing time in $\mathbf{T}$ decreases, which is also a component of the end-to-end time. A larger bundle increases the bundling time only, and this increase is dominated by the decrease of other two components.

Bundling outperforms instantaneous executions. Also, with regard to coordination method: Sync outperforms Async, and AS outperforms IND. Using only Sync does not always guarantee an improvement. If we use Sync, robots sometimes wait too long to fill the bundle. This bundling time becomes overshadows other times in Sync and IND because the bundles of all robots should be filled, and IND does not take maximum advantage of synchronized robots (i.e., there is no optimization in task distribution). Using only AS also does not always guarantee a significant improvement. Using AS with Async has robots that are quiet close to working independently. It is unlikely that robots will finish their bundles at the same time, so most time they individually take lower-cost tasks. In a bounded region, visiting a bundle of random locations versus only those locations close to the robot may have negligible difference, especially when we optimize the tour.

The fixed-$x$ policy yields the smallest system cost. The policy never takes fewer tasks than its overestimate of the optimal bundle size, and it exploits synergies among tasks. The up-to-$x$ policy reduces the bundling time by never waiting to fill bundles, but this may increase the system cost. The sweeping policy is similar to the up-to-$x$ but without limiting the number of tasks that the robots bundle. Therefore, the system cost is better than up-to-$x$, in general. The averaging policy does not show any remarkable performance; its advantage is in dealing with noisy patterns in the task profile.

In choosing a policy, one's purpose must be borne in mind. To help determine the appropriate policy, we show the results in the objective space in Fig. 7. Among all combinations (in Fig. 7a), only non-dominated combinations are shown in Fig. 7(b). The Pareto frontier is consists only of combinations of AS and Sync, so coordinating and synchronizing robots is the most beneficial for both objectives. Beyond this combination, one must select a policy. For the system cost, we would use the fixed-$x$ policy if models are available. Without any model, the sweeping



(a) All solutions. (b) The non-dominated solutions.

Figure 7: The objective space of the case where tasks arrive with $\alpha$. The polices form a Pareto frontier as marked in (a). The three polices correspond to the fixed-$x$, up-to-$x$, and the sweeping policy with the synchronized robots using the ILP. The Poisson arrival case is omitted since it has the same result.

policy is the best. For the end-to-end time, the up-to-$x$ and the sweeping policy show similar performances. We would choose between them according to the availability of the models.

## 6.3 Non-i.i.d. task locations and task arrival interval

We also run experiments with the task locations that are *not* independently and identically distributed. With a probability of 0.5, a task is drawn from a Uniform distribution within the arena that robots work. With a probability of 0.5, a task is drawn from a Normal distribution that has the location of the last task as the mean. The task arrival process also has the interval between tasks that is non-i.i.d. With a probability of 0.5, the arrival process follows the Poisson process with $\lambda$ which is a sinusoidal function. With a probability of 0.5, the interval is drawn from a Uniform distribution where the upper bound is related to the previous value of $\lambda$.

We only report results for Sync and AS. We tested the sweeping and the average policies since the models are not available in this non-i.i.d. case. The results (Table 2) show that the model-free policies outperforms the baseline method. Among the two model-free policies, one would choose the sweeping policy since this policy is shown to adapt itself well to changes in the task profile.

Table 2: The results from non-i.i.d. task locations and arrival intervals. Sync and AS are used.

| | Time traveled | End-to-end |
|---|---|---|
| Baseline | 37.64 (0.7788) | 5856 (293.1) |
| Sweeping | 14.80 (1.0594) | 2660 (1096) |
| Averaging | 21.03 (0.5740) | 4566 (1461) |

## 7 Conclusion and future work

This paper treats a variant of the multi-robot task allocation problem where stochastic tasks arrive continuously, and the system must determine how to bundle tasks in order to make best use of synergies between tasks. First, we proposed a basic model to understand the foundations of bundling in this setting. Then, from empirical studies, we explored how the model changes as a function of bundle size, team size, task distribution method, and degree of synchronization. Based on this qualitative study, we proposed a set of simple bundling policies to optimize the system cost or the timespan. It is shown that bundling outperforms no bundling. Also, the policies are able to deal with uncertainties in the task profile, such as probabilistic task arrivals or non-i.i.d. task locations and arrival intervals.

We plan to further study the strategies to improve the performance of bundled task execution. There are several directions for improvement. For example, robots may swap the tasks in their bundles for additional refinements. Preemptions of bundle executions may be useful so that some robots can stop working if they exceed the expected execution time for their bundles. We also wish to extend our study to tasks with negative synergies, no synergy, and (non-monotonic) complex synergies. Moreover, we are interested in other variants of tasks, such as tasks with deadline constraints or tasks that could be abandoned or rejected.

## Appendix: a randomized heuristic HP algorithm

We implement a randomized heuristic algorithm for the HP problem. The algorithm receives the input $\mathbf{X}_i$ where $|\mathbf{X}_i| = x_i$. It generates a list $\mathbf{X}'_i$ of two elements: the initial robot location and a randomly chosen visit location from $\mathbf{X}_i$ (line 1). The chosen location is removed from $\mathbf{X}_i$ (line 2). We consider the insertion positions in $\mathbf{X}'_i$: between the two elements and after the last element (we do not consider the foremost position before the elements). In general, there are $|\mathbf{X}'_i|$ insertion positions in $\mathbf{X}'_i$. Let $p_l$ be the insertion position for $l = 1, \cdots, |\mathbf{X}'_i|$.

The algorithm involves the following procedure for all visit locations in $\mathbf{X}_i$. From the first location, i) inserting the chosen visit location at each insertion position and generating multiple paths (line 6), and ii) choosing the minimum length path among the paths generated from i) (line 8). $\mathbf{X}'_i$ increases by one in each iteration by inserting one visit location. Once all visit locations are inserted to $\mathbf{X}'_i$ (at line 10), the path of $x^i$ visit locations has a reasonably small distance but still not optimal (or near-optimal). Now the algorithm randomly chooses one visit location $t_b$ and removes it from $\mathbf{X}'_i$ (line 12). Then the algorithm runs i) and ii) with $t_b$ and $\mathbf{X}'_i$ (lines 13–16). If the resulting path is better than the previous one, the algorithm updates the path (line 18). This improvement repeats for $\gamma x_i$ iterations, where $\gamma \in \mathbb{Z}^+$. The resulting Hamiltonian path would not be optimal but near-optimal. For better results (but probably a longer running time), the stopping point could be the time when the path distance converges.

## References

[1] B. Gerkey and M. Matarić, “A formal analysis and taxonomy of task allocation in multi-robot systems,” *Int. J. of Robotics Research*, vol. 23, pp. 939–954, Sept. 2004.

[2] B. Heap, “Sequential single-cluster auctions for multi-robot task allocation,” Ph.D. dissertation, The University of New South Wales, 2013.

[3] M. B. Dias and A. Stentz, “A market approach to multirobot coordination,” Carnegie Mellon University, Tech. Rep., 2000.

[4] B. Gerkey and M. Matarić, “Sold!: Auction methods for multi-robot coordination,” *IEEE Trans. on Robotics*, vol. 18, pp. 758–768, 2002.

[5] S. Amador, S. Okamoto, and R. Zivan, “Dynamic multi-agent task allocation with spatial and temporal constraints,” in *International Conference on Autonomous Agents and Multi-agent Systems*, 2014, pp. 1495–1496.

[6] R. Meir, Y. Chen, and M. Feldman, “Efficient parking allocation as online bipartite matching with posted prices,” in *International Conference on Autonomous Agents and Multi-Agent Systems*, 2013, pp. 303–310.

[7] E. Vee, S. Vassilvitskii, and J. Shanmugasundaram, “Optimal online assignment with forecasts,” in *Proceedings of the ACM conference on Electronic commerce*, 2010, pp. 109–118.

**Algorithm 2** RANDHP

**Input:** $\mathbf{X}_i$, a bundle of $x_i$ tasks

**Output:** $\mathbf{X}'_i$, a reordered $\mathbf{X}_i$ forming a Hamiltonian path

```
1  $\mathbf{P} = \varnothing, s = 0$
2  $\mathbf{X}'_i = \{l(r_i), l(t_a)\}$//$t_a$ is randomly chosen from $\mathbf{X}_i$
3  $\mathbf{X}_i = \mathbf{X}_i \setminus \mathbf{X}'_i$
4  for each $t_j \in \mathbf{X}_i$
5      for each $p_k \in \mathbf{X}'_i$
6          $\mathbf{P} \leftarrow$ INSERT$(t_j, p_k)$//insert $t_j$ in $p_k$ and add to $\mathbf{P}$
7      end for
8      $\mathbf{X}'_i =$ MINPATH($\mathbf{P}$)//find the minimum length path
9      $\mathbf{P} = \varnothing$
10 end for
11 while $s < \gamma x_i$
12     $\mathbf{X}'_i = \mathbf{X}'_i \setminus t_b$//$t_b$ is randomly chosen from $\mathbf{X}'_i$
13     for each $p_k \in \mathbf{X}'_i$
14         $\mathbf{P} \leftarrow$ INSERT$(t_b, p_k)$//insert $t_b$ in $p_k$ and add to $\mathbf{P}$
15     end for
16     $\mathbf{X}''_i =$ MINPATH($\mathbf{P}$)//find the minimum length path
17     if DIST($\mathbf{X}'_i$)>DIST($\mathbf{X}''_i$)//compare distances of paths
18         $\mathbf{X}'_i = \mathbf{X}''_i$//if the randomly modified path $\mathbf{X}''_i$ is
                  //better than the previous one, update $\mathbf{X}'_i$
19     end if
20     $\mathbf{P} = \varnothing$
21 end while
22 return $\mathbf{X}'_i$
```

[8] S. Koenig, C. A. Tovey, X. Zheng, and I. Sungur, “Sequential bundle-bid single-sale auction algorithms for decentralized control,” in *Proc. of Int. Joint Conf. on Artificial intelligence*, 2007, pp. 1359–1365.

[9] X. Zheng, S. Koenig, and C. Tovey, “Improving sequential single-item auctions,” in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Syst.*, 2006, pp. 2238–2244.

[10] B. Heap and M. Pagnucco, “Sequential single-cluster auctions for robot task allocation,” in *Advances in Artificial Intelligence*, 2011, pp. 412–421.

[11] J. Beardwood, J. H. Halton, and J. M. Hammersley, “The shortest path through many points,” in *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 55, no. 04. Cambridge Univ Press, 1959, pp. 299–327.

[12] D. Stein, “An asymptotic, probabilistic analysis of a routing problem,” *Mathematics of Operations Research*, vol. 3, pp. 89–101, 1978.

[13] J. Lee and M. Choi, “Optimization by multicanonical annealing and the traveling salesman problem,” *Physical Review E*, vol. 50, p. R651, 1994.

[14] L. Kleinrock, *Queuing systems*. Wiley, 1975.

[15] T. Bektas, “The multiple traveling salesman problem: an overview of formulations and solution procedures,” *Omega*, vol. 34, pp. 209–219, 2006.