Bundling policies for sequential stochastic tasks in multi-robot systems

Changjoo Nam and Dylan A. Shell

Abstract This paper studies multi-robot task allocation in settings where tasks are revealed sequentially for an infinite or indefinite time horizon, and where robots may execute bundles of tasks. The tasks are assumed to be synergistic so efficiency gains accrue from performing more tasks together. Since there is a tension between the performance cost (e.g., fuel per task) and the task completion time, a robot needs to decide when to stop collecting tasks and to begin executing its whole bundle. This paper explores the problem of optimizing bundle size with respect to the two objectives and their trade-off. Based on qualitative properties of any multi-robot system that bundles sequential stochastic tasks, we introduce and explore an assortment of simple bundling policies. Our experiments examine how these policies perform in a warehouse automation scenario, showing that they are efficient compared to baseline policies where robots do not bundle tasks strategically.

1 Introduction

Multi-robot task allocation (MRTA) considers optimizing collective performance of a team of robots that execute a set of tasks [Gerkey and Matarić, 2004]. In the canonical formulation, the sets of robots and tasks are fixed, and a decision-maker has full access to all information about the tasks. In practice, knowledge of the complete set of tasks may unavailable beforehand. In many applications, tasks are only revealed sequentially in an online fashion, e.g., dial-a-ride, e-commerce orders, etc. Compared to the case where the tasks are known *a priori*, work examining online instances of MRTA is scant (*cf.* discussion at length in [Heap, 2013]).

In addition to considering sequential revelation of information, here we are concerned with *synergistic tasks*. By this we mean that work performed toward one task may be useful for others too, and planning with larger sets of tasks is beneficial. Also, in addition to conventional cumulative cost measures (e.g., fuel, time), we consider the timespan of tasks, *viz.* the elapsed time from creation until completion. There can be a tension between these two objectives. If robots wait and execute

Changjoo Nam
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA
Dylan A. Shell
Dept. of Computer Science and Engineering, Texas A&M University, College Station, TX, USA
e-mail: cjnam@cmu.edu

multiple tasks together (i.e., as a *bundle*), the costs per task may be less than for independent executions. But one incurs a delay in waiting for tasks to arrive in order to fill a bundle, so bundling drives up the timespan. Figure 1 shows an example of a setting where the robot fills its bundle by waiting for more items. Over and above the standard question of 'how should the tasks be allocated among robots?' one asks 'how many tasks should the robots bundle?'

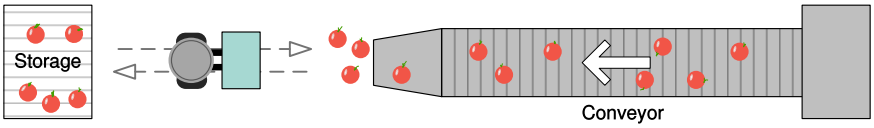


Fig. 1 A simple example of online synergistic tasks: cherries must be transported from the conveyor to the storage facility. The robot may reduce traveling costs (fruitility = Joules per fruit) by waiting for multiple items, aggregating and transporting them together at once (i.e., filling a bundle). But as the robot forms larger sets of items, the items wait longer on average.

This paper explores the structure of the bundling question at a high-level, abstracting away details of task performance itself, so that the findings can apply to a variety of settings. We begin with a qualitative study of the most basic instances in which tasks are revealed deterministically at a fixed frequency and where the task cost is a function of the task's location, and the location is independently and identically drawn from a known probability distribution. Also, robots are initially assumed to have no interactions. Based on the models of task arrival and execution, we compute the bundle sizes that minimize each objective. The simplifying assumptions that give this result are often violated in reality; to consider more realistic and complex settings, we explore a set of policies which efficiently adapt in order to improve performance.

This paper formulates of the problem of optimal task bundling for MRTA for sequentially revealed, synergistic tasks (Sec. 3). After analyzing the most basic scenario (Sec. 4), we introduce models that describe performance objectives as a function of bundle size. Using these models, our study of iterated bundle execution leads us to propose simple and efficient bundling policies suitable for variations of the problem which generalize the basic instance (Sec. 5). Evaluation of our policies is carried out quantitatively with extensive experiments (Sec. 6).

2 Related work

Most previous work in online MRTA focuses on the question of *how to allocate tasks*. Early work on auction and market mechanisms [Dias and Stentz, 2000; Gerkey and Matarić, 2002] studied the allocation of tasks when no model of their arrival was known. Some recent work [Amador et al., 2014; Meir et al., 2013] considers online tasks with unpredictable arrivals, but the option of bundling is not considered. Online bipartite matching solves the underlying optimization problem of the online MRTA, but does not consider the bundling as a first-class question.

Bundling has been the focus of some prior attention. Koenig et al. [2007] proposed Sequential Single-Item (SSI) auctions with bundles, wherein robots submit bids for subsets of tasks from a known set of tasks. The bidding phase and the

winner determination phase iterate sequentially until all tasks have been assigned. Compared to standard parallel auctions, SSI reduces the team's cost by exploiting synergies among tasks. It also reduces the time spent bidding compared to the standard combinatorial auctions since not all permutations of assignments are considered. Zheng et al. [2006] propose SSI with roll-outs where the cost of a task is evaluated together with the previously allocated tasks in order to exploit synergies. The aspect we study, which is absent for these prior works, is the meta-reckoning that weighs the consequences of choosing to delay decision-making since doing so may improve per-task system costs but worsen completion times.

Bullo et al. [2011] show a variety of routing policies for multi-vehicle servicing demands in various scenarios, applying spatial-queuing theoretic models to vehicle routing where visiting locations arrive through a stochastic process. They provide theoretical analyses showing the stability (i.e., the number of waiting demands is bounded at all times) and the service quality of each routing policy. Their work provides thorough analyses and extensive comparisons of routing policies, demonstrating deep understanding in the routing domain. The techniques have yet to be shown to be directly applicable to more generic synergistic tasks. This paper adopts the hypothesis that the fundamentals of bundling and the principle trade-offs are not tied to the particular task-type or setting. It is possible that our study concerning the optimal bundle size would improve some of the routing policies of Bullo et al. [2011]. For example, the unbiased TSP policy has a design parameter n determining the size of sets to plan a TSP tour for all n demands. The set is analogous to our task bundle so n could be determined based on our result.

3 Problem description

This section formulates the problem, expresses constraints on the problem, and describes the objectives. Warehouse automation is used as an example.

3.1 Problem formulation

Given n robots, a task arrives and is inserted to a structure $\mathbf{T}$ every α seconds ($\alpha > 0$). The total number of tasks is unknown and the sequence may be unbounded. Here α could be deterministic or a random variable—in the latter case, we abuse notation slightly by using α to denote the mean of a distribution. The robots are assumed to share all available task information (i.e., $\mathbf{T}$) through a communication network or some other similar mechanism. The cost of performing a task is a function of both the robot and the task locations; we assume the locations of tasks are drawn from a probability distribution. Performing multiple tasks together enables some common work to be lumped together so that the cost, as a function of number of tasks performing in a bundle, is sub-linear.

Definition 3.1 (Synergistic task) Let $c(S)$ be the cost of performing a set of tasks S. Tasks are *synergistic* if $c(S_1) + c(S_2) > c(S_1 \cup S_2)$ where $S_1 \neq S_2$.

Definition 3.2 (Task bundle) Each robot owns a task bundle which is a group of $x \in \mathbb{Z}^+$ tasks extracted from $\mathbf{T}$. The *bundling time* is the average time that a task waits until the bundle of itself is completed.

We assume that the tasks in $\mathbf{T}$ are taken by order of arrival. Once a task is assigned to a bundle, it is no longer available to other robots. Once a robot has its bundle, it must finish the tasks without further modifications of that bundle. Tasks continue to arrive concurrently while the robots perform the work assigned to them. The robots iterate bundling and executing tasks in turn. A robot may be idle while waiting to fill its bundle and so idleness will depend on x. The simplifications in the preceding (e.g., no task reordering, no out-of-order execution, no pre-emption) ensure the operation will be starvation-free; judicious relaxation of these constraints may improve overall performance, but remains for future work.

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 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 inserted to $\mathbf{T}$.
- **Non-starvation:** no task may be abandoned to remain in $\mathbf{T}$ indefinitely.

We consider two objectives to minimize, subject to fulfillment of both these requirements. Since there is no fixed set of tasks, the conventional sum-of-cost measure is no longer directly applicable, though average values of the following are:

Definition 3.3 (System cost or execution cost) The system cost per task $\bar{c} \in \mathbb{R}^{\geq 0}$ is the average cost spent to finish a task by the system. The average is taken over all robots and task pairs.

Definition 3.4 (Timespan or end-to-end time) The timespan of a task $\bar{\tau} \in \mathbb{R}^{\geq 0}$ is the average time elapsed for a task from its insertion into $\mathbf{T}$ until its completion. The average is taken over all tasks.

3.2 An example: Warehouse automation

We use a warehouse automation problem (Fig. 2) as a running example to explore the properties of strategic bundling. Our study is not tied to this particular example as we have studied the optimization of bundle size for the vehicle routing problem too, see Nam and Shell [2016] for details. The reader interested in the generality of the model may find those additional examples beneficial.

In a warehouse, n robots are tasked with transporting items from racks to a packing station using baskets. Each rack has its own fixed location and contains identical items. Each rack has a space below permitting robots to navigate under it. The floor is discretized into grid cells where one cell contains one robot or one rack. Robots must navigate to their destinations while avoiding collisions. Once a robot is under-

neath its assigned rack, the rack dispenses its items in the quantity requested. The robot can visit multiple racks until its basket is full.

A job managing server (JM) receives orders that arrive sequentially and indefinitely according to a stochastic process. The JM splits each order into atomic transportation tasks: robots must deliver items from racks to the packing station. The JM inserts these into $\mathbf{T}$. A robot bundles k tasks from $\mathbf{T}$ after it has completed its previous tasks. The extraction of tasks is done from the head of $\mathbf{T}$. Also, robots execute their bundles from the head.[1]

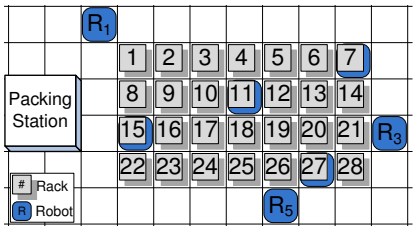


Fig. 2 A warehouse scenario where delivery tasks arrive sequentially. Delivering multiple items in one tour reduces the traveling cost but would delay the shipment of some items. Bundling strategically can manage this efficiency trade-off.

4 An analysis of bundle size

This section develops models for the optimization objectives. We begin with a simplified setting in which enables introduction of basic execution cost and task arrival models. Next, some complexity is added to help improve realism and applicability.

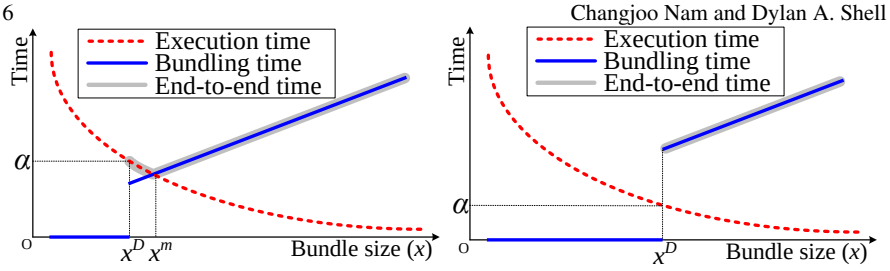
4.1 The basic case: independent robots and regular task arrivals

4.1.1 The general model

Consider a multi-robot system with negligible physical interference among robots and tasks in $\mathbf{T}$ that are sufficiently abundant so no contention occurs. The degree to which this is an over-simplification depends on practical circumstances, but this model results in objective values possessing invariant, steady-state properties. We assume stochastic tasks with locations independently and identically distributed from a uniform distribution over a rectangular grid with area S. A new task is revealed every α seconds. In what follows, refer to Fig. 3 as it shows models of both the system cost $\bar{c}$ per task and the average timespan $\bar{\tau}$ of a task.

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 (red in Fig. 3). Task synergies imply that $f(\cdot)$ is decreasing and, hence, the bundle size that minimizes the system cost is infinite ($x_f^* = \infty$). A model for bundling time is given by a functional $h(x|\alpha, f)$. Note that $h(\cdot)$ is discontinuous since $h(x|\alpha, f) = 0$ if $x < x^D$ (blue in Fig. 3). The quantity x^D denotes the bundle size when $f(x|S,v) = \alpha$, that is, the point of balance between the rate of task arrivals and (average) executions. Below equilibrium x^D, a task arrives before an existing task is completed. Thus, tasks accumulate ($|\mathbf{T}|$ is

[1] Again, robots may employ more advanced methods rather than this in-order task bundling and execution. The focus of this paper is not on elaborate methods for these aspects, so we leave them as modules that can be replaced by well-designed allocation and planning methods.



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

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

Fig. 3 Illustrative functions that describe the system cost (red) and timespan (gray) per task. The time in $\mathbf{T}$, $s(\cdot)$, 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, and $g(\cdot)$ for $x < x^D$ is not shown since the value is infinite.

unbounded) and robots remove tasks from $\mathbf{T}$ without waiting, i.e., the bundling time is zero. For $x \geq x^D$, tasks do not accumulate in $\mathbf{T}$, and a robot must wait for tasks in order to fill its bundle and so the bundling time is nonzero. So, $h(x|\alpha, f) = h'(x|\alpha)$ for $x \geq x^D$, where $h'(\cdot)$ represents the bundling time without considering the potential unbounded accumulation of tasks in $\mathbf{T}$. But there is also another component $s(x|\alpha, f)$, the average time a task resides in $\mathbf{T}$ before it is taken by a robot. We have $s(x|\alpha, f) = \infty$ for $x < x^D$ because $\mathbf{T}$ keeps accumulating tasks so it has a significant number of tasks that wait indefinitely to be bundled. Otherwise, $s(x|\alpha, f) = f(\cdot)$ since tasks only stay in $\mathbf{T}$ while robots are executing their bundles.

A model of timespan per task $\bar{\tau}$ is given by $g(x|S, v, \alpha, f)$, which describes the component that dominates the time:

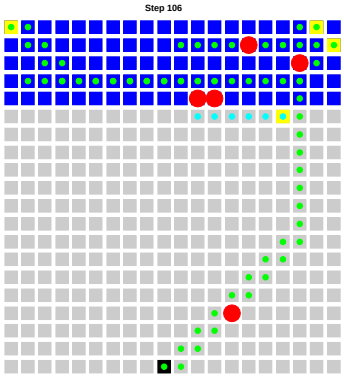
$$g(x|S, v, \alpha, f) := \max(f(x|S, v), h(x|\alpha, f), s(x|\alpha, f)), \tag{1}$$

illustrated with the thick gray curves in Fig. 3. For $x < x^D$, $g(\cdot)$ the value is infinite. The x value that minimizes $g(\cdot)$ is the optimal bundle size x_g^*. The value x_g^* can be determined via two cases, shown in Fig. 3a and Fig. 3b respectively. In Fig. 3a, 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 been filled, thus $g(\cdot)$ takes the minimum at this point, so $x_g^* = x^m$. It is worth noting that there is no waiting time between iterations. In Fig. 3b, x^m does not exist because the number of tasks in $\mathbf{T}$ are unbounded. The execution time dominates the (zero) bundling time, and $f(x^D|\cdot)$ takes the minimum at x^D, so $x_g^* = x^D$.

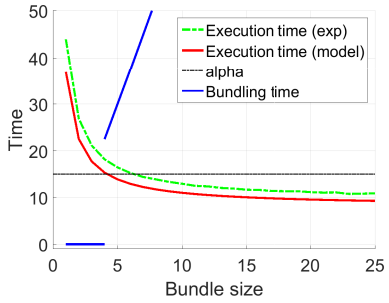
4.1.2 The warehouse example

We derive analytic models of $\bar{c}$ and $\bar{\tau}$ for the warehouse example. We assume that robots move one grid cell (with no diagonal) at each step. Robots bundle k tasks and execute them sequentially using a path planner (e.g., A*). Collisions are avoided by dynamic replanning. In an $a \times b$ rectangular space, racks are located within an $a' \times b'$ rectangle. The packing station is located at (x_0, y_0). Fig. 4a shows an example setting where $a = b = a' = 20$, $b' = 5$, and $(x_0, y_0) = (10, 1)$. Tasks arrive every α steps and their locations are uniformly distributed within the $a' \times b'$ rectangle.

At steady-state, executing a bundle consists of three trips: (*i.*) the trip from the station to a rack, (*ii.*) the trip among the k racks, and (*iii.*) the trip to the station. Let d be the expected distance from the station to a random rack and E_d be the expected



(a) A simulation of the warehouse problem. Red circles and yellow squares represent robots and tasks. The blue and black squares represent racks and the station, respectively. The small dots show paths where the cyan dots are replanned paths to avoid collisions.



(b) The models (2) in red dotted and (3) in blue solid. The horizontal line represents α. The green curve shows the experimental result from Fig. 4a.

Fig. 4 A snapshot of the warehouse simulator and the validation of the models from simulations.

distance between two random racks. We do not write out the expressions for d and E_d owing to limited space, but their computation requires nothing more than basic calculus. From (*i.*) and (*iii.*) we have $2d$. From (*ii.*), we have $(k-1)$ trips among k racks. Thus, it takes $2d+(k-1)E_d$ time steps for k tasks. The system cost (time traveled) per task is

$$f(k|S,v) = \left(\frac{2d+(k-1)E_d+(k+1)}{k}\right). \tag{2}$$

The term $(k+1)$ is added because we assume that planning a path between two points takes one time step. There are total $k+1$ runs of the A* planner. Notice that all environmental variables are represented by S.

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

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

Interestingly, $h'(\cdot)$ is a special case of the mean residual life of a customer in a renewal process presented in Kleinrock [1975]. 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

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

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

Fig. 4b shows (2) as a function of bundle size (red) along with the values from experiments (green) where $\alpha = 15$, and $a = b = 20$. The blue line represents (3). One may derive models analytically as above or make use of a body of research that provides such the models (e.g., the optimal length of a TSP tour, which is a function of the number of visit locations [Stein, 1978; Lee and Choi, 1994]).

4.2 Adding realism: robot interactions and stochastic task arrivals

Next, we consider settings where tasks involve uncertainty via a stochastic arrival process. In addition, non-negligible robot interactions are taken into account. These generalizations introduce a gap between the basic models and the performance observed in the system (different from Fig. 3). Factors responsible for the gap include:

1. *Task location:* Owing to the stochasticity of task locations, the system cost described above should really describe the mean of a random variable. Therefore $f(\cdot)$ is no longer deterministic, but has some variance.
2. *The task arrival process:* Stochastic arrivals mean that $h(\cdot)$ is no longer deterministic, making $h(\cdot)$ a random function with some variability.
3. *Physical interference:* Robots may interfere with one another, increasing the system cost so that $f(\cdot)$ in the basic model underestimates the actual cost.
4. *Task contention:* Robots may experience contention for tasks, increasing the bundling time while waiting to fill bundles. Thus, the $h(\cdot)$ in the basic model underestimates the actual bundling time.
5. *Robot coordination:* Robots that coordinate to optimize performance will reduce the system cost per task; thus, the basic model $f(\cdot)$ may overestimate actual costs.

In sum, the optimal bundle size x_g^* in the basic case is likely to differ from the optimal value for realistic settings. But the modifications needed depend on aspects of the domain and many details of the particular instance. This fact motivates our exploration of adaptive *bundling policies* to adjust to circumstances, improving performance of idealized treatments which ignore complicated aspects of the system. In Section 5, we propose model-free policies with a dynamic bundle sizes.

4.2.1 Remarks on synergies in the realistic case

The factors given above may have ramifications, not only for the parameters (e.g., slope, y-intercept) of $f(\cdot)$, but also its monotonicity. We assume that the tasks remain synergistic as negative robot interactions caused by resource contention can often be mitigated by (far-sighted or globally aware) planning algorithms—even if some optimality must be sacrificed, approximations should suffice in this regard. Thus, we believe that $f(\cdot)$ usually continues to decrease even if synergism diminishes.

Nevertheless, some extreme configurations (e.g., a very narrow warehouse causing heavy congestion on every passage) could make $f(\cdot)$ non-synergistic when no algorithm can reduce these negative effects. The analysis remains valid even though it becomes more involved when handling non-idealized cases: the question of what

bundle size, x, makes $g(\cdot)$ a minimum still persists. If $g(\cdot)$ is always infinite or constant, the fact that the optimal set is empty or all of $\mathbb{Z}^+$ is informative. And, if there are multiple bundle sizes which minimize $g(\cdot)$, any of them can be chosen. And, of course, when $f(\cdot)$ is no longer synergistic, the optimal bundle size is 1, as there is no benefit from executing tasks in concert.

5 Bundling policies

Before considering adaptive policies, we describe static policies where bundle sizes remain constant. Later, these static policies will serve as a performance baseline. Next, we propose simple policies that are flexible with agreeable behavior across a range of circumstances. We provide a condition for a stable policy (i.e., ensuring $|\mathbf{T}|$ is bounded) and a bound that describing the best possible performance, which no bundling policy can exceed. We consider only policies that minimize the timespan $\bar{\tau}$, since minimizing $\bar{c}$ has limited practical value, leading to infinite bundles.

5.1 Baseline static policies

5.1.1 The ideal policy

If the basic model $g(\cdot)$ is available, finding a k that minimizes (1), the timespan, gives x_g^*. In this ideal static policy, each robot keeps $k = x_g^*$. When a robot finishes its current bundle and tries to execute the next bundle, there may be insufficient tasks in $\mathbf{T}$ to form that next bundle, causing the robot to wait, idly, for new tasks. If $|\mathbf{T}| \geq k$, the robot takes k tasks and executes them immediately. Since this policy does not handle uncertainties in the task profile (discussed in Sec. 4.2) it is possible that the timespan can diverge if $\mathbf{T}$ is unbounded.

5.1.2 The min- and max-load policies

Another reasonable policy is to execute, instantaneously, tasks one by one (i.e., $k = 1$). This min-load policy does not exploit synergies but would yield a small timespan since robots never bundle multiple tasks. We also consider the max-load policy where bundles take up to some given capacity (e.g., the capacity of baskets of robots, the memory size of robots where task information is stored).

5.2 Model-free policies

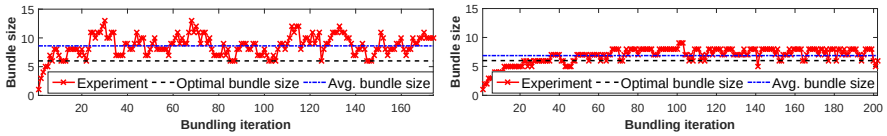
We propose two policies requiring neither domain knowledge nor models.

5.2.1 The sweeping policy

This policy takes all tasks $k = |\mathbf{T}|$ if $|\mathbf{T}| \geq 1$, never incurring any bundling time as long as at least one task is available. The policy exploits synergies among available tasks. Fig. 5a shows the bundle size *vs.* time (30,000 steps). The black dotted line shows the ideal bundle size reflecting the equilibrium in which the execution and bundling times are equal. The average bundle size (blue dotted) would reflect the equilibrium when the task uncertainties and robot interactions are accounted for.

5.2.2 The averaging policy

The sweeping policy's equilibrium is constant unless the stochastic parameters describing the task location and arrival process change. The sweeping policy does not make explicit use of any representation of the equilibrium. Instead, it tracks the equilibrium via history: the averaging policy begins with $k = 1$ and averages the previous bundle sizes saved in a history window W. The smaller the window size, the more sensitive the policy to variability.



(a) The sweeping policy **(b)** The averaging policy ($|W| = 20$)

Fig. 5 Plots showing bundle size *vs.* time steps. Bundle sizes (red) stay around the blue value, which is the optimal bundle size for the given task setting.

5.3 An analysis of the bundling policies

We say a policy is *stable* if the number of waiting tasks in $\mathbf{T}$ is bounded.

Proposition 5.5 Let $x^{D'}$ be the value of x^D, incorporating the various non-idealizations for more realistic settings. A bundling policy is stable if $k \geq x^{D'}$.

Proof. Even in circumstances outside of the basic case, the shape of the objectives (Fig. 3) is invariant: non-idealized circumstances elevate or lower the lines, but do not affect their shape. Thus, the analysis of the models in Sec. 4.1.1 may still holds. Task queue $\mathbf{T}$ does not accumulate tasks if the bundle size is greater than the equilibrium between the task execution rate and the arrival rate (i.e., $x \geq x^{D'}$). In other words, a task is completed before or when a new task arrives and, thus, a bundling policy with $k \geq x^{D'}$ is must be stable. □

The ideal policy should be stable in the basic case but is cannot be guaranteed to be stable in more realistic settings, since $x^{D'}$ is not known exactly. Also, it is not obvious whether the sweeping and the average polices are stable. In practice, however, they appear to be stable as Fig. 5 shows. Specifically, $x_g^* \geq x^{D'}$ in any case as shown in Sec. 4.1.1. In Fig. 5, the average bundle size (blue) is larger than x_g^* (black) so larger than $x^{D'}$. The stability of the two is shown empirically in the following experiments.

Next, we show the lower bound in which any bundling policy cannot exceed.

Proposition 5.6 The lower bound of $\bar{\tau}$ is $\lim_{k\to\infty} f(k|\cdot)$ for any policy.

Proof. From (1), the lower bound of $\bar{\tau}$ is $g(\cdot) = \max(\min f(\cdot), \min h(\cdot), \min s(\cdot))$. The minimum of $f(\cdot)$ is at $k \to \infty$. The minimum of others are zero (i.e., no bundling at $k = 1$ and no residing time for $k < x^D$). Then, the maximum is $\min f(k|\cdot) = \lim_{k\to\infty} f(k|\cdot)$. No policy can exceed this bound. □

For example, in the warehouse problem, $\lim_{k\to\infty} f(k|\cdot) = E_d + 1$.

6 Quantitative study: comparisons of the policies

This section describes experiments in the warehouse setting. We provide experimental settings and analyze the results where task locations are i.i.d. with a fixed task interval. Then, we show results when tasks arrive according to a Poisson process whose arrival intervals and locations are non-i.i.d. Both cases involve physical interference and task contention among robots.

6.1 Experimental settings and results

The size of the warehouse is $a = b = 30$ where the number of racks is 300 so $a' = 30$, $b' = 10$, $E_d = 11.11$. The packing station is at $(15,1)$ so $d = 25.50$. For a fixed number of robots ($n = 5$), we assume that all robots move at the same velocity, one grid cell per one time step. As discussed above, using x_f^* is unrealistic so we minimize the timespan only and scrutinize how the system cost changes.

We set $\alpha = 2$ for the regular *intense* task arrival.[2] 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 interval. In addition, we ran experiments for $\alpha = 30$ which represent *intermittent* arrivals (i.e., $E_d + 1 < \alpha$). For the ideal policy, $k = 10$ for $\alpha = 2$ and $k = 2$ for $\alpha = 30$. The max-load and the min-load have $k = 30$ and $k = 1$, respectively. The size of window is $|W| = 20$ for the averaging policy. We measure the two objective values over 10,000 steps and run 10 repetitions. Table 1 summarizes the results. Fig. 6 shows the size of $\mathbf{T}$ over time.

Table 1 Comparisons of policies. The values are the mean and standard deviation (10 repetitions).

Bundling policy	Intense ($\alpha = 2$)		Intermittent ($\alpha = 30$)	
	System cost	Timespan	System cost	Timespan
Min-load	64.64 (0.2614)	4231 (4.557)	67.54 (5.599)	65.74 (5.532)
Max-load	16.47 (0.0906)	2142 (30.41)	24.83 (14.49)	1131 (246.7)
Ideal	22.89 (0.1766)	2845 (23.22)	51.51 (7.291)	116.7 (14.43)
Sweeping	20.73 (1.784)	1488 (49.41)	71.98 (9.627)	69.87 (9.200)
Averaging	21.43 (0.3958)	1441 (25.59)	70.13 (9.627)	74.34 (0.7071)

(a) Regular task arrivals with an i.i.d. task distributions

Bundling policy	Non-i.i.d. interval	
	System cost	Timespan
Min-load	65.28 (0.5991)	4046 (38.98)
Max-load	11.38 (0.0675)	396.9 (12.79)
Sweeping	12.57 (0.4441)	301.3 (65.04)
Averaging	18.69 (2.234)	1921 (98.11)

(b) A Poisson task arrival process with a non-i.i.d. task distribution

6.2 Analysis of results

6.2.1 Regular task arrivals and an i.i.d. spatial distribution

The results of the intense arrivals (Table 1a) show that any bundling policy outperforms non-bundling (the min-load). The min-load policy does not make use of the synergies of tasks so its task execution time (system cost) is larger than other policies. Thus, tasks accumulate rapidly in $\mathbf{T}$ (Fig. 6a) which increases the timespan significantly. The max-load policy yields the smallest system cost since it exploits synergies maximally. Bundling that many tasks increases the bundling time which contributes to the total timespan. Thus, tasks are accumulated moderately in $\mathbf{T}$

[2] The lower bound of the execution time per task is greater than the arrival interval of a task, i.e., $E_d + 1 = 26.50 > \alpha = 2$.

(Fig. 6b) while robots spend time bundling. The ideal policy is outperformed by the max-load policy. It is possible that $k = 30$ is closer to the actual optimal bundle size than $k = 10$ when robot interactions are taken into account. Task accumulation (Fig. 6c) is faster than the max-load policy but slower than the min-load policy. The sweeping and averaging policies show the similar timespan, which outperform all the baseline polices because they can ensure **T** remains bounded as shown in (Fig. 6d and 6e). Their system costs are not the most efficient found, but are competitive.

The most important aspect of the performance is how a policy ensures **T** is bounded because it relates directly to the timespan at steady-state. If **T** is not bounded, the timespan, which includes the time a task resides in **T** will itself diverge. The max-load policy seems to have a moderate timespan, but must diverge if experiments are run longer. On the other hand, the proposed model-free policies are able to keep **T** short.

To determine the appropriate policy, one's overall purpose must be borne in mind. To help in choosing a policy, we show the results in the objective space in Fig. 7. For intense arrivals, we would use one of the model-free policies for the timespan. In considering system cost, one would choose the max-load policy.

The intermittent case shows a slightly different result. The min-load policy has the smallest timespan. For the system cost, the max-load policy still has the smallest value. Since tasks arrive slowly, only few robots (most times only one robot) executes tasks, resulting in little physical interference, and, thus, $f(\cdot)$ in Fig. 3 is the same as the basic case. On the other hand, severe task contention means that $h(\cdot)$ is steeper than the basic model. This causes the optimal bundle size to shrink. The min-load policy has the closest bundle size to this small value so its timespan is the smallest. All policies ensure that **T** is bounded since tasks arrive slowly. It would be beneficial for the robots to move to the centroid of the area of racks when they are idle since the expected distance to a random rack is the shortest at the point. This waiting location changes depending on the spatial distribution of tasks.

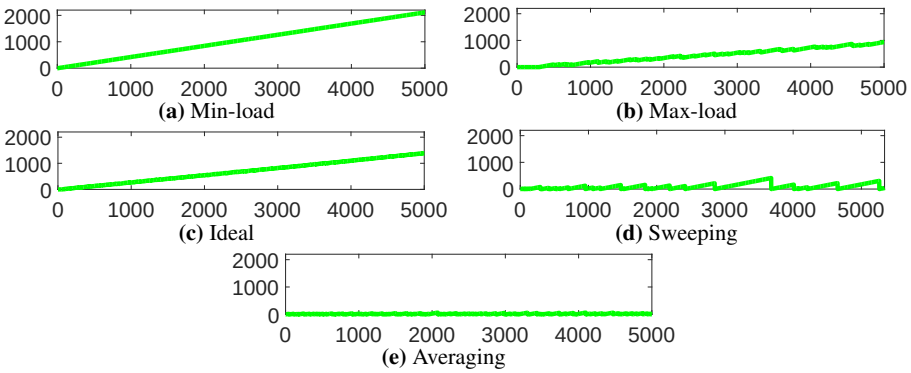


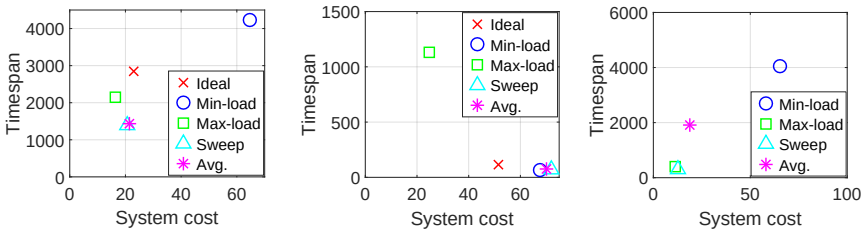
Fig. 6 Plots showing bundle size (y-axis) over time (x-axis). We show the case of regular and intense intervals. The sweeping and averaging policies are able to bound **T**.

6.2.2 A Poisson arrival of tasks and a non-i.i.d. spatial distribution

We also ran experiments to explore how the policies work in more complex situations with task locations chosen so as *not* to be independently and identically distributed. Specifically, a task is drawn from a uniform distribution within the area that robots work with a probability of 0.5. With a probability of 0.5, a task is drawn from a normal distribution that has the location of the previous task as the mean. The task arrival process also has a mean interval between tasks that is non-i.i.d. With a probability of 0.5, the arrival process follows the Poisson process with λ 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 λ.

In Table 1b and Fig. 7c, we show the results (no ideal policy is reported as we have no appropriate model). Except for the averaging policy, the policies exhibit a tendency similar to that of intense and regular arrivals before. As noted, the max-load policy accumulates tasks over time, so the timespan will increase with longer experiments. In the regular arrival case, the averaging policy shows performance that is competitive with the sweeping policy, but is now far worse. This is because of the frequent switching of the task profile (both arrivals and locations) disrupts the averaging policy's local estimates of the rate and various system costs.

This problem is resolved by having a short history: setting $|W| = 1$, the result is $\bar{c} = 11.98$ and $\bar{\tau} = 427.9$, which is now comparable to sweeping. The size of $\mathbf{T}$ in this experiment is essentially the same as with Fig. 6 and is therefore omitted.



(a) Regular intense **(b)** Regular intermittent **(c)** Poisson non-i.i.d.

Fig. 7 The objective space showing a Pareto frontier.

7 Conclusion and future work

This paper treats a variant of the MRTA 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 idealized models to understand the foundations of the bundling question. Then we explored how the models change for more realistic cases where task uncertainties and robot interactions are involved. We proposed adaptive bundling policies in order to deal with uncertainties, comparing the results to baseline policies which do not have a particular strategy for bundling. Evidence is provided to show that the proposed policies outperform the baseline. More importantly, the proposed policies can bound the number of waiting tasks at all time for intense task arrivals whereas the baseline policies cannot. Also, the results show that the policies are able to deal with other sources of complexity

and uncertainty, such as probabilistic task arrivals or non-i.i.d. task locations and arrival intervals—which can express aspects of spatio and temporal locality.

Further study of related strategies can improve the performance of bundling. For example, preemption of bundle executions may be useful so that some robots can stop working to perform other tasks. Or robots may swap the tasks in their bundles to reduce costs during execution. Out-of-order task insertion to **T** or bundles is also interesting. Several applications naturally impose temporal constraints between tasks which are worth considering. Lastly, it would be desirable to give the analytic bound of the team performance using the policies to compute the minimum number of robots to bound **T** for a given frequency of task arrivals.

Acknowledgments

This work was supported in part by NSF awards IIS-1302393 and IIS-1453652.

References

[Amador et al., 2014] S. Amador, S. Okamoto, and R. Zivan. Dynamic multi-agent task allocation with spatial and temporal constraints. In *Int. Conf. on Autonomous Agents and Multi-agent Systems*, pages 1495–1496, 2014.

[Bullo et al., 2011] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. Smith. Dynamic vehicle routing for robotic systems. *Proc. of the IEEE*, 99:1482–1504, 2011.

[Dias and Stentz, 2000] M. Dias and A. Stentz. A market approach to multirobot coordination. Technical report, Carnegie Mellon University, 2000.

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

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

[Heap, 2013] B. Heap. *Sequential Single-Cluster Auctions for Multi-Robot Task Allocation*. PhD thesis, The University of New South Wales, 2013.

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

[Koenig et al., 2007] S. Koenig, C. 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*, pages 1359–1365, 2007.

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

[Meir et al., 2013] R. Meir, Y. Chen, and M. Feldman. Efficient parking allocation as online bipartite matching with posted prices. In *Int. Conf. on Autonomous Agents and Multi-Agent Systems*, pages 303–310, 2013.

[Nam and Shell, 2016] C. Nam and D. Shell. An empirical study of task bundling for sequential stochastic tasks in multi-robot task allocation. Technical Report TAMU-CSE-16-7-1, CSE Dept., Texas A&M University, 2016.

[Stein, 1978] D. Stein. An asymptotic, probabilistic analysis of a routing problem. *Mathematics of Operations Research*, 3:89–101, 1978.

[Zheng et al., 2006] X. Zheng, S. Koenig, and C. Tovey. Improving sequential single-item auctions. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Syst.*, pages 2238–2244, 2006.