Scheduling Spare Drones for Persistent Task Performance under Energy Constraints

Robotics Track

Erez Hartuv
Department of Computer Science
Bar-Ilan University, Isreal
erez.hartuv@gmail.com

Noa Agmon
Department of Computer Science
Bar-Ilan University, Isreal
agmon@cs.biu.ac.il

Sarit Kraus
Department of Computer Science
Bar-Ilan University, Isreal
sarit@cs.biu.ac.il

ABSTRACT

This paper considers the problem of enabling persistent execution of a multi-drone task under energy limitations. The drones are given a set of locations and their task is to ensure that at least one drone will be present, for example for monitoring, over each location at any given time. Because of energy limitations, drones must be replaced from time to time, and fly back home where their batteries can be replaced. Our goals are to identify the minimum number of spare drones needed to accomplish the task while no drone battery drains, and to provide a drone replacement strategy. We present an efficient procedure for calculating whether one spare drone is enough for a given task and provide an optimal replacement strategy. If more than one drone is needed, we aim at finding the minimum number of spare drones required, and extend the replacement strategy to multiple spare drones by introducing a new Bin-Packing variant, named Bin Maximum Item Double Packing (BMIDP). Since the problem is presumably computationally hard, we provide a first fit greedy approximation algorithm for efficiently solving the BMIDP problem. For the offline version, in which all locations are known in advance, we prove an approximation factor upper bound of 1.5, and for the online version, in which locations are given one by one, we show via extensive simulations, that the approximation yields an average factor of 1.7.

KEYWORDS

Multi-Robot Systems; Multiagent Scheduling; Task Allocation

ACM Reference Format:
Erez Hartuv, Noa Agmon, and Sarit Kraus. 2018. Scheduling Spare Drones for Persistent Task Performance under Energy Constraints. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 9 pages.

1 INTRODUCTION

Aerial drones are emerging as an effective and efficient tool for monitoring and surveillance. Applications include civil security operations, continuous surveillance of a disaster scene such as flooding and forest fires [21], traffic monitoring [10], and event photography [25]. The main limitation in deploying drones for such applications is their short flight times: commonly used drones, equipped with even minimal sensors, have a maximum flight time of approximately 20 to 30 minutes, usually less than that. To overcome this severe limitation in persistence monitoring tasks, spare drones should be available to replace drones that are running low on battery. The replaced drones could fly to a location where their batteries can be charged or replaced, enabling them to continue in their task [22].

This paper considers the problem of managing a fleet of drones that are performing a persistent monitoring task. In particular, the drones are given a set of locations, and their task is to ensure that at least one drone will be present over each location at any given time. To face the drones' energy limitations efficiently and enable drone replacement before their batteries drain, it is important to identify the minimum number of spare drones necessary to accomplish the persistent monitoring task. We therefore define the Minimal Spare Drone for Persistent Monitoring Problem, MSDPM, for determining this minimal number of spare drones, as well as finding a schedule of drone replacements that guarantees both that the persistent monitoring tasks are fulfilled indefinitely, and that no drone battery is drained. The drones are homogeneous with one home for battery exchange. See an illustration in our simulated environment in Figure 1. We consider two variations of the MSDPM problem: (i) An offline version, where the set of locations is given in advance, and (ii) An online version, where the set of locations is given one by one over time. When a location is given, it must be assigned immediately to one of the spare drones, which are added as needed.

While the paper describes the MSDPM problem and possible solutions for persistent monitoring for drones, it is valid for *any* robot type having energy constraints, performing *any* persistent task, as long as the travel cost of the robots in the environment satisfies the triangle inequality. We first consider the problem of deciding whether one spare drone is enough, and provide a formula that can be computed to make this decision. Furthermore, we show that a simple drone replacement in which the replaced drone always goes directly to the home is not worse than any other replacement procedure. When one spare drone is not enough for the drones to be present at the given locations at all times, deciding on the minimum of required drones is presumably hard. We first prove that an optimal replacement procedure could be based on the division of the locations to independent sets, and assign one spare drone to each set. The replacement could be done, back and forth, separately in each set, as in the case that one spare is needed. Based on this proof, we model the problem of deciding on the minimal number of necessary spare drones for a given set of locations as a new variant of the Bin-Packing problem, denoted as Bin Maximum Item Double Packing (BMIDP). We then provide a first fit greedy approximation algorithm to efficiently solve the BMIDP problem.

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden.

For the *offline* version we have proven an approximation ratio of 1.5. For the *online* version we ran extensive simulations showing approximation ratio of 1.7.

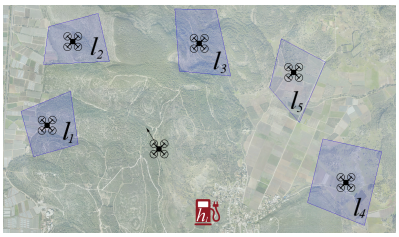


Figure 1: Five areas requiring persistent monitoring ($l_1, \ldots, l_5$), and one recharging station (h_1).

2 RELATED WORK

The problem of energy-aware persistent task performance is related to different problems in the literature, from theoretical analysis related to the vehicle-routing problem [12] to practical deployment of multi-robot teams in continuous missions, with energy-aware solutions, e.g., [14, 15, 18]. All these related problems lack the combination of two key features that are addressed in our research: (i) Persistent continuous (non-stop) service (ii) Minimizing the number of necessary robots for task completion.

The Vehicle Routing Problem (VRP) [2, 7, 17] seeks to generate routes, aka tours, for a team of agents leaving a starting location referred to as the depot, visiting a number of goal locations, and returning back to the depot. For VRP and its variants, one must find the optimal routing strategy that allows a fleet of vehicles to visit a set of targets while trying to minimize some objective, which usually takes the form of travel distance. The solutions will include, for each robot, a path that visits targets once and then revisits the first target, which is referred to as a tour. All tours together cover all of the targets. Then minimizing the maximum period of tours, yields minimizing the delay between customer visits. From the complexity point of view, the classical VRP is known to be NP-hard since it generalizes the Traveling Salesman Problem (TSP) and the Bin Packing Problem (BPP) which are both well-known NP-hard problems [9].

In [15] a heuristic for the Continuous Monitoring Problem with the goal to maximize the visiting frequency of targets is proposed. The travel distance of a UAV is limited by fuel constraints, however UAV's can refuel at any base station. This is similar to [16] where closed tours through targets and refueling depots for a number of robots are planned such that each target is visited. They address the Multi-robot Persistent Coverage Problem (MRPCP), which they express as a variant of the Vehicle Routing Problem (VRP). The Multi-Robot Routing problem (MRR) [23] seeks to plan paths for a team of robots to visit a large number of interchangeable goal locations as quickly as possible. The VRP variation that most closely resembles MRR is the Minimum Maximum Multiple Depot Vehicle Routing Problem (MMMDVRP) [4].

Song et al [11, 22] have suggested an architecture for persistent task performance by a team of multiple UAVs. They suggest a MILP-based solution, determining the optimal scheduling of UAVs to tasks, given that the UAVs may have different constraints (including energy). Their work is evaluated on the security-escort task: making sure that a ground vehicle is continuously monitored by a UAV. They *do not* examine the question of minimal necessary UAVs, nor do they consider multiple (static) targets for monitoring, as examined in this paper. Park and Morison [20] examine a similar problem of determining the minimal number of UAVs for persistent robotic service of immobile customers and one home base. They use Petri-nets modeling, and provide an efficient heuristic algorithm for determining the minimal number of UAVs and batteries for guaranteeing service to customers, similar to the vehicle routing problem. As opposed to their work, in our work all the targets (customers) must be continuously attended at all times, not just repeatedly visited for service and than left. Recent work by Burdakov et al. [3] considers the problem of replacing security UAVs performing the mission of surveillance along a perimeter. The goal is to determine which UAV to replace and when, while minimizing the effect this has on the chances of observing an intruder attempting to penetrate the perimeter. Also in this case, the authors assume that the number of UAVs is given, and focus on the optimal replacement schedule. Nigam and Kroo [19] offer a heuristic solution for determining paths for UAVs performing persistent monitoring by patrolling through an area. Mathew et al. [13] also considers UAV patrolling paths, with meeting points for UAV recharging. In both cases, the main focus is either on finding optimal paths and/or refueling points, and not on determining the minimal number of UAVs sufficient for the mission.

3 THE MSDPM PROBLEM DEFINITION

In this section we formally define the Minimal Spare Drone for Persistent Monitoring Problem, MSDPM. A set of drones is given a task of persistent monitoring, that is, given a set l of k locations, $l = \{l_1, l_2, \ldots, l_k\}$, at least one drone should be present at each location at all times. The drones have limited energy, thus they must be replaced *before* their battery drains. Moreover, when replaced, they must have enough energy to return them safely to their home (denoted by h_1) for battery exchange. Therefore, the required number of drones necessary to ensure persistent monitoring is greater than k. We refer to the p extra drones, that is, the drones used for replacing the drones in the monitoring task, as *spare drones*. The formal definition of the Minimal Spare Drone for Persistent Monitoring Problem, MSDPM, is as follows:

> *Given a set of k locations that require persistent monitoring, a set of $k+p$, $p > 0$ homogeneous drones with maximal battery capacity $L < \infty$, and one home location h_1 in which the drones replace batteries. Determine whether the p spare drones are sufficient for assuring that each location is monitored indefinitely by at least one drone, and no drone's battery will drain unless it is in h_1.*

The above description is the decision-version of the MSDPM problem. Our goal is to find the minimal number of spare drones satisfying the persistent monitoring task, that is, the minimal number p^* such that MSDPM is true.

As discussed in Section 5, the MSDPM problem is a special variant of the Bin-Packing problem. Since the Bin-Packing problem is $NP-Hard$ [5, 9], MSDPM is presumably hard as well. We leave the hardness proof for future work.

As mentioned above, at any given time, we divide the set of drones into two subsets: $D^k = \{d_1, \ldots, d_k\}$ are located in the k locations, and the other p spare drones, $D^s = \{d_{s_1}, \ldots, d_{s_p}\}$, are

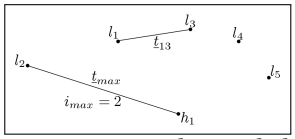


Figure 2: The paths are not necessarily straight lines, as long as the travel cost satisfies the triangle inequality.

either in h_1 or on their way to/from some location, and are used for replacing drones from D^k in the monitoring task. $D = D^k \uplus D^s$. We will use d_i or just i ($1 \le i \le k$) interchangeably while referring to a drone that is located in l_i, and we will use d_{s_i} ($1 \le i \le p$) for denoting a spare drone. We assume that all drones are homogeneous, have the same velocity v, and are all initially located in h_1 fully charged (L). We denote by $E_i(t)$ the battery level at time t of drone i located at l_i, and similarly $E_{s_i}(t)$ is the battery level at time t of a spare drone d_{s_i}. Let τ_i denote the most recent time before time t, in which drone d_i had full battery charge, Thus $E_i(\tau_i) = L$. We denote by c the rate of discharge per time unit. Following standard assumptions on battery discharge function (e.g., [3]), $E_i(t)$ decreases linearly with time: $\forall i, 1 \le i \le k \quad E_i(t) = L - c \cdot (t - \tau_i)$. Similarly we can obtain: $\forall i, 1 \le i \le p \quad E_{s_i}(t) = L - c \cdot (t - \tau_{s_i})$

The time it takes the drones to get from h_1 to their locations plays an important role in the minimal number of spare drones defined in the MSDPM problem. We denote by $dist(a, b)$ the distance between two points a and b. We use the following notations for these travel times (illustration in Figure 2):

$\underline{t}_{ij} \coloneqq \frac{dist(l_i, l_j)}{v}$ is the drone's travel time between l_i and l_j (same as l_j to l_i).

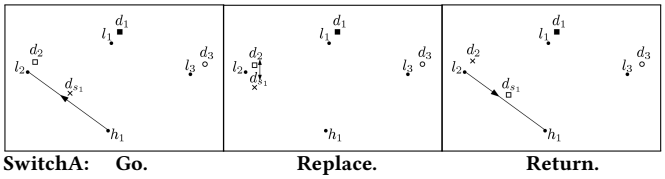
$\underline{t}_i \coloneqq \frac{dist(l_i, h_1)}{v}$ is the drone's travel time between h_1 and l_i. Therefore $c \cdot \underline{t}_i$ is the amount of charge units it takes a drone to get from h_1 to l_i.

$\underline{t}_{max} \coloneqq \max_{1 \le i \le k} (\underline{t}_i)$ is the travel time from h_1 to the farthest location $l_{i_{max}}$, $i_{max} = \operatorname{argmax}_{1 \le i \le k}(\underline{t}_i)$.

We define an important time step, t_0, as the time the last drone initially arrives at its location $l_{i_{max}}$, i.e., $t_0 \coloneqq \underline{t}_{max}$. We assume that at t_0 all the drones are at their locations, and the spares are at h_1. Therefore, $E_i(t_0) = L - c \cdot \underline{t}_{max}, \forall i \quad 1 \le i \le k$, and $E_{s_j}(t_0) = L, \forall j \quad 1 \le j \le p$.

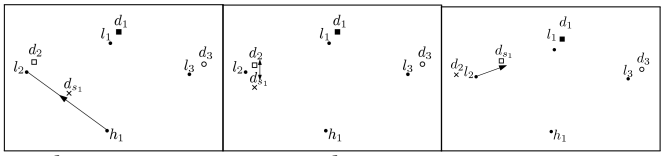
Each Drone i must stay over location l_i until it is replaced by one of the spare drones d_{s_j}. When the replaced drone leaves l_i, it exchanges names with the spare drone that just arrived: the spare drone is named d_i and the replaced drone becomes d_{s_j}. Thus we do not refer to a specific drone by its identity, but by the task it performs, that is, the location in which it performs its monitoring task. Exactly 4 switch types describe all possible drone actions regarding energetic optimal replacements at location l_i:

SwitchA: d_{s_j} travels from h_1 to l_i (Go). When it arrives at l_i, drones i and d_{s_j} exchange names (Replace). The former drone i (which is now d_{s_j}) travels from l_i directly to h_1 (Return).



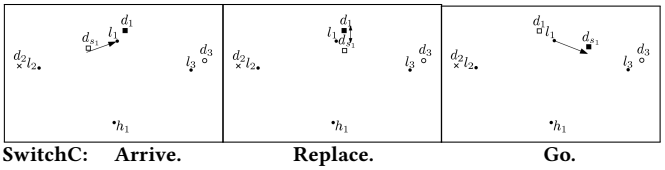
SwitchA: Go. Replace. Return.

SwitchB: d_{s_j} travels from h_1 to l_i (Go). When it arrives at l_i, drones i and d_{s_j} exchange names (Replace). The former drone i (which is now d_{s_j}) travels to some other location $l_x \in l$ (Go).



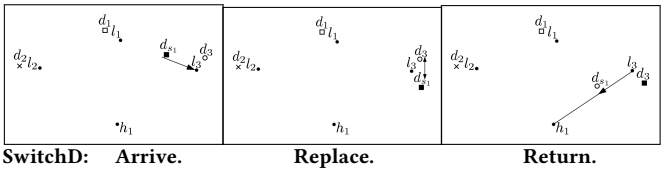
SwitchB: Go. Replace. Go.

SwitchC: d_{s_j} travels from some location $l_y \in l$ to l_i (arrive). When it arrives at l_i, drones i and d_{s_j} exchange names (Replace). The former drone i (which is now d_{s_j}) travels to some other location $l_x \in l$ (Go).



SwitchC: Arrive. Replace. Go.

SwitchD: d_{s_j} travels from some location $l_y \in l$ to l_i (Arrive). When it arrives at l_i, drones i and d_{s_j} exchange names (Replace). The former drone i (which is now d_{s_j}) travels from l_i to h_1 and arrives there (Return).



SwitchD: Arrive. Replace. Return.

We will later prove (in theorem 4.4 and lemma 5.1) that for minimizing the number of spare drones, it is enough to focus only on replacements of type SwitchA, where each drone that is replaced should return directly to h_1 to replace its battery.

4 SINGLE SPARE DRONE

In this section we consider the case in which $p = 1$, that is, $D = \{d_1, \ldots, d_k, d_{s_1}\}$, and we should determine whether it is possible to guarantee persistent monitoring using one single spare drone. To encompass all possibilities of drones replacements we first define a mathematical notation.

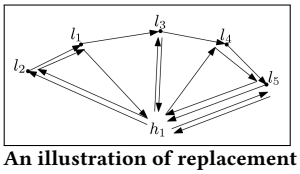


Figure 7: An illustration of replacement scheme.

Definition 4.1. A *replacement scheme* $R = (i_1, i_2, \ldots, i_j, \ldots, i_{k_1})$ is a series of time consecutive drone replacements until all the batteries of all k drones are replaced at h_1. A drone replacement over location l_{i_j} of switch type s_j at time t_j is denoted by $i_j \in \{1, 2, \ldots, k\}$ for $1 \le j \le k_1, \quad k_1 \ge k$. The series of replacements timings, denoted by $R^T = (t_1, t_2 \ldots, t_j, \ldots, t_{k_1})$, is non decreasing. The series of replacement switch types is: $R^S = (s_1, s_2 \ldots, s_j, \ldots, s_{k_1})$, where $s_j \in \{A, B, C, D\}$.

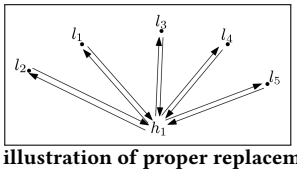


Figure 8: An illustration of proper replacement scheme.

R=(2, 1, 3, 4, 5, 2, 1, 3, 4, 5, 5) is the replacement scheme illustrated in figure 7, R^S=(B, C, C, C, D, B, D, A, B, D, A) is the corresponding series of replacement switch types. R^T=(10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110) can be the corresponding series of replacements timings, if all travel times are equal to 10 time units.

A replacement scheme which contains only drone replacements of type SwitchA, is most useful, and we refer to it as a *proper replacement scheme*. The length of a proper replacement scheme is k, and it is a permutations of $(1, 2, \ldots, k)$. $R = (2, 1, 3, 4, 5)$ is the proper replacement scheme illustrated in figure 8.

A sub-series of a replacement scheme that starts with a spare drone that leaves h_1 to replace a drone which in turn may replace another drone and so on until the last replaced drone in the sub-series returns to h_1, is referred to as a *cycle*, defined formally as follows.

Definition 4.2. A *Drone replacement cycle* (*cycle*, in short) is a sub-series of a replacement scheme $R = (i_1, i_2, \ldots, i_j, \ldots, i_{k_1})$, and has one of two types:

(1) $C_j = (i_{j_1})\ 1 \le j_1 \le k_1$ is a *type 1 cycle*, when i_{j_1} is a drone replacement of type SwitchA over location $l_{i_{j_1}}$.
(2) $C_j = (i_{j_1}, i_{j_2}, \ldots, i_{j_q})\ 1 \le j_1 < j_q \le k_1$ is a *type 2 cycle*, when:
   - i_{j_1} is a drone replacement of type SwitchB over location $l_{i_{j_1}}$
   - $i_{j_2}, \ldots, i_{j_{q-1}}$ are drone replacements of type SwitchC over locations $l_{i_{j_2}}, \ldots, l_{i_{j_{q-1}}}$, respectively
   - i_{j_q} is a drone replacement of type SwitchD over location $l_{i_{j_q}}$

We denote a *type 1* cycle $C = (x)$ over location l_x by C_{xx}, and a *type 2* cycle $C = (v, w, x, y, z)$ over a set of locations (for example) l_v, l_w, l_x, l_y, l_z by C_{vwxyz}.

Note that any replacement scheme, $R = (i_1, i_2, \ldots, i_j, \ldots, i_{k_1})$, has a corresponding series of drones replacements cycles: $R^C = (C_1, C_2, \ldots, C_j, \ldots, C_n)$, where $k \le n \le k_1$, When $n = k$ there is no redundancy, i.e., there are exactly k (number of locations) battery replacements at h_1 during R. Each cycle C_j is one of the two replacement cycle types, and if R is a proper replacement scheme, then all of them are *type 1 cycles*. In the general case with several spare drones, each cycle is a sub-series of R and has the form: $C_j = (i_{j_1}, i_{j_2}, \ldots, i_{j_q})\ 1 \le j_1 < j_q \le k_1$, the concatenation of all cycles gives R up to order of elements due to the simultaneous independent performance of cycles by several spare drones. With one spare drone, the elements of a cycle are consecutive in R: $C_j = (i_{j_1}, i_{j_1+1}, i_{j_1+2}, \ldots, i_{j_q})\ 1 \le j_1 < j_q \le k_1$, the concatenation of all cycles gives exactly R. In the illustration of figure 7: $R = (C_1, C_2, C_3, C_4, C_5)$, where $C_1 = (2, 1, 3, 4, 5) = C_{21345}$, $C_2 = (2, 1) = C_{21}$, $C_3 = (3) = C_{33}$, $C_4 = (4, 5) = C_{45}$, $C_5 = (5) = C_{55}$. In the illustration of figure 8: $R = (C_1, C_2, C_3, C_4, C_5)$, where $C_1 = (2) = C_{22}$, $C_2 = (1) = C_{11}$, $C_3 = (3) = C_{33}$, $C_4 = (4) = C_{44}$, $C_5 = (5) = C_{55}$.

Recall that L denotes the maximal level of energy per drone. Therefore, our main goal when considering situations with a single spare drone, is to decide whether L is enough for enabling the persistent monitoring tasks over l. We will first focus on proper replacement schemes specifying the needed constraint on L as a function of the times it takes the drones to reach the locations in l.

Lemma 4.3. *In order to guarantee persistent monitoring on a location set l using the set of drones D with a maximum battery level L, if starting the drone replacements at t_0, it is sufficient and necessary when using a proper replacement scheme that the following requirement is satisfied:*

$$L - 2c \cdot \underline{t}_i \ge \sum_{j=1}^{k} 2c \cdot \underline{t}_j \qquad \text{for } i = 1, \ldots, k \tag{1}$$

Proof. Let $R = (i_1, i_2, \ldots, i_j, \ldots, i_k)$ be a proper replacement scheme with timings $R^T = (t_1, t_2, \ldots, t_j, \ldots, t_k)$. t_1 is the time of first drone, i_1, replacement. The time it takes the spare drone to travel from h_1 to l_{i_1}, and for replaced drone (former i_1) to get back to h_1 is $t_1 - t_0 = 2\underline{t}_{i_1}$. We exchange names such that the spare drone just arrived at l_{i_1} is called i_1, and the drone that was replaced and returned to h_1 is called spare.

$$E_{i_1}(t_1) = L - 2c \cdot \underline{t}_{i_1}$$

$$E_{i_u}(t_1) = L - c \cdot \underline{t}_{max} - 2c \cdot \underline{t}_{i_1} \qquad \text{for } u = 2, \ldots, k$$

t_2 is the time of the second drone, i_2, replacement. The time for spare drone to get from h_1 to l_{i_2}, and for replaced drone (former i_2) to get back to h_1 is $t_2 - t_1 = 2\underline{t}_{i_2}$. We exchange names such that the spare drone just arrived at l_{i_2} is called i_2, and the drone that was replaced and returned to h_1 is called spare.

$$E_{i_w}(t_2) = L - \sum_{j=w}^{2} 2c \cdot \underline{t}_{i_j} \qquad \text{for } w = 1, 2$$

$$E_{i_u}(t_2) = L - c \cdot \underline{t}_{max} - 2c \cdot \underline{t}_{i_1} - 2c \cdot \underline{t}_{i_2} \qquad \text{for } u = 3, \ldots, k$$

t_{k-1} is the time of (k-1)'th drone, i_{k-1}, replacement. The time for spare drone to get from h_1 to $l_{i_{k-1}}$, and for replaced drone (former i_{k-1}) to get back to h_1 is $t_{k-1} - t_{k-2} = 2\underline{t}_{i_{k-1}}$. We exchange names such that the spare drone just arrived at $l_{i_{k-1}}$ is called i_{k-1}, and the drone that was replaced and returned to h_1 is called spare.

$$E_{i_w}(t_{k-1}) = L - \sum_{j=w}^{k-1} 2c \cdot \underline{t}_{i_j} \qquad \text{for } w = 1, \ldots, k\text{–}1$$

$$E_{i_k}(t_{k-1}) = L - c \cdot \underline{t}_{max} - \sum_{j=1}^{k-1} 2c \cdot \underline{t}_{i_j} \tag{2}$$

t_k is the time of k'th drone (the last in R), i_k, replacement. The time for spare drone to get from h_1 to l_{i_k}, and for replaced drone (former i_k) to get back to h_1 is $t_k - t_{k-1} = 2\underline{t}_{i_k}$. We exchange names such that the spare drone just arrived at l_{i_k} is called i_k, and the drone that was replaced and returned to h_1 is called spare.

$$E_{i_w}(t_k) = \quad L - \sum_{j=w}^{k} 2c \cdot \underline{t}_{i_j} \qquad \text{for } w = 1, \ldots, k$$

Lets take $w = 1$ to get:

$$E_{i_1}(t_k) = L - \sum_{j=1}^{k} 2c \cdot \underline{t}_{i_j} \underset{permutation}{=} L - \sum_{j=1}^{k} 2c \cdot \underline{t}_j \tag{3}$$

In order to have enough battery charge until this stage, where all k drones replaced battery at h_1, and to be able to start a new cycle of the proper replacement scheme R, and repeat it periodically, requirements (2) and (3) become the following sufficient and

necessary requirements (4) and (5):

$$L - c\cdot \underline{t}_{max} \geq \sum_{j=1}^{k} 2c\cdot \underline{t}_j \tag{4}$$

$$L - 2c\cdot \underline{t}_i \quad \geq \sum_{j=1}^{k} 2c\cdot \underline{t}_j \qquad \text{for i=1,...,k} \tag{5}$$

Requirement (5) holds in particular for $i = i_{max}$ in which case $\underline{t}_i = \underline{t}_{max}$. Therefore:

$$L - c\cdot \underline{t}_{max} > L - 2c\cdot \underline{t}_{max} \overset{req.(5)}{\geq} \sum_{j=1}^{k} 2c\cdot \underline{t}_j$$

i.e. requirement (5) implies also requirement (4). We conclude that requirement (5) which is requirement (1) of the lemma, is sufficient and necessary for L to be enough battery charge to repeatedly perform the proper replacement scheme R. □

We generalize Lemma 4.3 to a case in which any series of drones replacements (containing all 4 switch types) with a single spare drone, that starts at t_0, is allowed. Note that any such series, is a concatenation of replacement schemes that are not necessarily proper.

Lemma 4.4. *In order to keep persistent monitoring on a location set l, using the set of drones D with maximum battery charge L, performing any series of drone replacements that starts at t_0, it is necessary that the following requirement is satisfied:*

$$L - 2c\cdot \underline{t}_i \geq \sum_{j=1}^{k} 2c\cdot \underline{t}_j \quad \text{for } i = 1,\ldots,k \tag{1}$$

Proof. Given any series of drone replacements starting at t_0, we notice that it is a concatenation of replacement schemes, let $R_1 = (i_1, i_2, \ldots, i_{k_1}), k_1 \geq k$ and $R_2 = (i_{k_1+1}, i_{k_1+2}, \ldots, i_j, \ldots, i_{k_1+k_2}), k_2 \geq k$ be the first two replacement schemes in this concatenation. We write R_1 *and* R_2 as concatenations of drones replacements cycles (def 4.2, single spare drone, no redundancies): $R_1 = (C_1, \ldots, C_k)$, and $R_2 = (C'_1, \ldots, C'_k)$. There are no redundancies, therefore the total number of cycles in each replacement scheme is k.

R_1 ends with drone replacement $i_{k_1} \in C_k$, i.e. at time t_{k_1} the last drone that until now has not replaced battery, arrives at h_1 after it left location $l_{i_{k_1}}$. This last drone has been replaced at $l_{i_{k_1}}$ by a spare drone which came from h_1 if $C_k = C_{i_{k_1} i_{k_1}}$, or from some location l_z if $C_k = C_{xyzi_{k_1}}$ (for example). We now carefully examine the situation at the end of R_1. If we sort the locations according to the time of entry from h_1 of the replacement drones, we get: $l_{j_1}, l_{j_2}, \ldots, l_{j_k}$ (a permutation of the locations $l_1, l_2, \ldots, l_k$). Over l_{j_1} is the drone with earliest entry time, it has entered from h_1 by the first cycle C_1, but not necessarily to its final location l_{j_1}. Similarly C_2 entered the drone that eventually arrived during R_1 to l_{j_2}, C_3 relates to l_{j_3}, and so on. The drone over l_{j_1} had to make a path from h_1 to l_{j_1}: $\{(h_1, x), (x, y), (y, z), (z, j_1)\}$ denoted $P_{h_1 xyzj_1}$. These path edges are scattered among the cycles of R_1, after performing them alternately, this same drone has to wait for the sequential performance of the path $P_{j_1 uvwh_1}$ which is the suffix of the cycle containing the last path edge (z, j_1) of $P_{h_1 xyzj_1}$. Combining $P_{h_1 xyzj_1}$ and $P_{j_1 uvwh_1}$ we get a cycle $\widetilde{C}_{xyzj_1 uvw}$ therefore by triangle inequality and applying the same arguments on drones over locations $l_{j_2}, \ldots, l_{j_k}$ we get

$$0 \leq E_{j_w}(t_{k_1}) \leq L - \sum_{i=w}^{k} 2c\cdot \underline{t}_{j_i} \qquad \text{for } w = 1,\ldots,k$$

Each cycle of R_2: $C'_1, C'_2, \ldots, C'_k$ puts out one drone to h_1 from all k locations in some permutation of the locations $l_1, l_2, \ldots, l_k$. Thus we get requirement (1) of the lemma. □

Let t'_0 be the time in which all drones have initially arrived at their locations, and d_{s_1} is at h_1. We do not make any assumptions on the drones activities before t'_0. This leads to the following lemma.

Lemma 4.5. *In order to keep persistent monitoring on a location set l, using the set of drones D with maximum battery charge L, starting drones replacements at t'_0, it is necessary when using a proper replacement scheme, that the following requirement is satisfied:*

$$L - 2c\cdot \underline{t}_i \geq \sum_{j=1}^{k} 2c\cdot \underline{t}_j \qquad \text{for } i = 1,\ldots,k \tag{1}$$

Proof. In any other start condition t'_0, requirement (1) is also necessary, since in the ideal case (which is not feasible, but is used to prove that we can use our start condition t_0 w.l.o.g) all k drones have full charge L at time t'_0, e.g. $E_i(t'_0) = L$, instead of, $E_i(t_0) = L - c\underline{t}_{max}$ for $1 \leq i \leq k$. Repeating the same computation as in the proof of lemma 4.3 using t'_0 instead of t_0, yields the same requirement (5). Similarly, instead of requirement (4) we get: $L \geq \sum_{j=1}^{k} 2c\cdot \underline{t}_j$, which is also implied by requirement (5) which is requirement (1). □

We conclude with the general requirement on the battery capacity of the drones, L, to enable k drones to perform persistent monitoring on a set of locations l with a single spare drone.

Theorem 4.6. *In order to keep persistent monitoring on a location set l, using the set of drones D with maximum battery charge L, it is sufficient and necessary that the following requirement is satisfied:*

$$L - 2c\cdot \underline{t}_i \geq \sum_{j=1}^{k} 2c\cdot \underline{t}_j \quad \text{for } i = 1,\ldots,k \tag{1}$$

Proof. Follows directly from lemma 4.3, lemma 4.4 and lemma 4.5. □

If the requirement is satisfied, then any proper replacement scheme (containing only drone replacements of type `SwitchA`) guarantees that the persistent monitoring task is performed indefinitely. If the requirement is not satisfied, then one spare drone is *not* enough for performing persistent monitoring over the given locations. Thus this requirement solves the `MSDPM` problem for $p = 1$.

5 FINDING THE MINIMUM NUMBER OF SPARE DRONES

In this section we consider the case in which one spare drone is *not* enough for performing a given persistent monitoring task. That is, given the set of k locations l and k drones with battery capacity of L, requirement (1) is not satisfied, i.e., there is at least one i s.t. $L - 2c\cdot \underline{t}_i < \sum_{j=1}^{k} 2c\cdot \underline{t}_j$, or equivalently, $L - 2c\cdot \underline{t}_{max} < \sum_{j=1}^{k} 2c\cdot \underline{t}_j$.

In this case, our goal is to identify the minimal number of spare drones that are needed to satisfy the task, i.e., solve the `MSDPM` problem for general p (the number of spare drones). Recall that $D = \{d_1, \ldots, d_k\} \cup \{d_{s_1}, \ldots, d_{s_p}\}$ is the set of $n_d = k + p$ homogeneous drones. In order to find the minimum p needed, we prove first that we can divide l into p disjoint subsets $l = S_1 \uplus S_2 \uplus \ldots \uplus S_p$, each will be persistently monitored by a proper replacement scheme with one spare drone. Note that some drone replacements may happen

at the same time, in which case they will appear consecutively in the replacement scheme, breaking ties arbitrarily. However, it is easy to see there will not be a case of two or more spare drones that arrive simultaneously to the same location, in order to replace a drone, because only one drone over a specific location should be replaced in a single moment, otherwise it will be an energy waste by the triangle inequality.

Lemma 5.1. *If p spare drones are needed in order to keep persistent monitoring on locations set l, using the set of drones D with maximum battery charge L, performing any series of drones replacements, then it can also be achieved by dividing l into p disjoint subsets $l = S_1 \uplus S_2 \uplus \ldots \uplus S_p$ such that for each subset S_i, persistent monitoring is achieved using one spare drone d_{s_i} to repeatedly perform a proper replacement scheme over the locations in S_i.*

Proof. Given any series of drones replacements, we notice that it is a concatenation of replacement schemes, let $R_1 = (i_1, i_2, \ldots, i_{k_1})$, $k_1 \geq k$ and $R_2 = (i_{k_1+1}, i_{k_1+2}, \ldots, i_j, \ldots, i_{k_1+k_2})$, $k_2 \geq k$ be the first two replacement schemes in this concatenation. R_1 has a corresponding series of drones replacements cycles (def 4.2, no redundancies): $R_1^C = (C_1, C_2, \ldots, C_j, \ldots, C_k)$. R_1^C can be rewritten using a different superscript for each spare drone. Here is an example with 3 spare drones: $R_1^C = (C_1^1, C_1^2, C_1^3, C_2^1, C_3^1, C_2^2, C_3^2, C_2^3, \ldots, C_m^b)$, the total number of cycles is k. R_1 ends, with drone replacement i_{k_1}, i.e. at time t_{k_1} the last drone that has not yet replaced its battery arrives at h_1 after leaving location $l_{i_{k_1}}$. This last drone has been replaced at $l_{i_{k_1}}$ by a spare drone which came from h_1 if $C_m^b = C_{i_{k_1} i_{k_1}}$, or from some location l_z if $C_m^b = C_{xyzi_{k_1}}$ (for example). We now carefully examine the situation at the end of R_1. If we sort the locations according to the time of entry from h_1 of the replacement drones, we get: $l_{j_1}, l_{j_2}, \ldots, l_{j_k}$ (a permutation of the locations $l_1, \ldots, l_k$). Over l_{j_1} is the drone with earliest entry time, it has entered from h_1 by the first cycle, but not necessarily to its final location l_{j_1}. Similarly the second cycle entered the drone that eventually arrived during R_1 to l_{j_2}, and so on. The drone over l_{j_1} had to travel along a path from h_1 to l_{j_1}: $P_{h_1 x y z j_1}$ with edges $\{(h_1, x), (x, y), (y, z), (z, j_1)\}$ that are scattered among the cycles of R_1. Although different cycles can be performed in parallel, those edges of $P_{h_1 x y z j_1}$ cannot be performed in parallel because it is the same drone performing them. After performing $P_{h_1 x y z j_1}$ alternately, this same drone has to wait the sequential performance of the suffix of the cycle containing the last edge path (z, j_1) of $P_{h_1 x y z j_1}$. This suffix starts with the replaced drone leaving l_{j_1} and ends with some drone arriving h_1, denote it $P_{j_1 u v w h_1}$. Combining these two paths, $P_{h_1 x y z j_1}$ and $P_{j_1 u v w h_1}$ we get a cycle $\widetilde{C}_{x y z j_1 u v w}$, therefore, by the triangle inequality and by applying the same arguments to drones over locations $l_{j_2}, \ldots, l_{j_k}$, we know that battery charge L, which is sufficient for R_1, is also sufficient for the proper replacement scheme we now define: $R_1^{'} = (j_1, j_2, \ldots, j_k)$. $R_1^{'}$ has a corresponding series of drones replacements cycles, here is an example with 3 spare drones: $(C_{j_1 j_1}^1, C_{j_2 j_2}^2, C_{j_3 j_3}^3, C_{j_4 j_4}^1, C_{j_5 j_5}^1, C_{j_6 j_6}^2, C_{j_7 j_7}^2, \ldots, C_{j_k j_k}^b)$. We now show that L is also enough to repeatedly perform $R_1^{'}$. By the triangle inequality the energetic state of the drones at the end of $R_1^{'}$ is better than at the end of R_1. The second replacement scheme R_2 has a corresponding series of drones replacements cycles R_2^C with k cycles (definition 4.2, no redundancies). Each cycle of R_2^C puts out one drone to h_1, it is done from all k locations in some permutation of the locations $l_1, \ldots, l_k$. Thus instead of R_2 we can use $R_1^{'}$ and continue repeatedly. □

Lemma 5.1 means that if the minimum number of spare drones p is achieved by some replacement scheme, then there is a replacement scheme which uses the same minimum number of spare drones p in which each spare drone is associated with a subset of the locations and follows a *proper* replacement scheme. The subsets are disjoint and their union includes all the locations. Recall that the Bin-Packing problem [1, 5, 6, 8, 26] can be described as follows: Given a set of n items with sizes a_1,..., a_n and a supply of identical bins ("containers") with capacity V, find a packing of all items into a minimal number of bins. This task is one of the classical problems of combinatorial optimization and is NP-hard [9]. There exist a family of greedy-based heuristic algorithms for solving the problem with proven approximation ratio that is no worse than 1.7, for the online version. The offline version has a proven approximation ratio that is no worse than 1.22. Following Lemma 5.1, the MSDPM problem is similar to the Bin-Packing problem: we want to find the minimal number of spare drones (bins) to jointly attend all locations (items) for replacement.

Therefore, we can find the minimum number of spare drones p, and allocate locations to the spare drones, by solving a variation of the Bin-Packing problem with an additional constraint: in each bin, items are packed such that the maximum item at this bin is packed twice. The reason for this is that by lemma 5.1 we can divide l into p disjoint subsets $l = S_1 \uplus S_2 \uplus \ldots \uplus S_p$, each will be persistently monitored by a proper replacement scheme with one spare drone. Requirement 1 of theorem 4.6 implies that in each set, each drone, and in particular the one with maximum distance from h_1, has to wait for all other drones to be replaced by a spare drone which takes a back and forth travel to h_1, than to perform one more back and forth to h_1 to start the next replacement scheme. We name this new variant *Bin Maximum Item Doubled Packing* (BMIDP). The items to be packed are the battery charge amounts: $2c \cdot \underline{t}_1, 2c \cdot \underline{t}_2, \ldots, 2c \cdot \underline{t}_k$. The formal definition of the BMIDP is as follows.

*Definition 5.2. Bin Maximum Item Doubled Packing (*BMIDP*).* Given a list of n items $S = \{1, 2, \ldots, n\}$ with positive sizes: $a_1, \ldots, a_n$ find the minimum number B of disjoint subsets of S, called Bins, having the same capacity V, such that, $S = S_1 \uplus S_2 \uplus \ldots \uplus S_B$ and for each subset $S_j \in \{S_1, \ldots, S_B\}$ $\sum_{i \in S_j} a_i + \max_{i \in S_j}\{a_i\} \leq V$

It follows from the definition that $\max_{i \in S}\{a_i\} \leq \frac{V}{2}$.

When applying BMIDP to find the minimal number of bins, B, the resulting solution is equivalent to finding the minimal number of spare drones p, the capacity V equals L (the energy capacity of the drones), and the sizes a_i are the battery charge amounts $2c \cdot \underline{t}_i$ which a spare drone requires in order to go from h_1 to l_i and for the replaced drone to go back from l_i to h_1. We consider two versions of the BMIDP problem: (i) The *offline* version, in which all items are known in advance. It solves the offline version of the MSDPM problem where the set of locations is given in advance (ii) The *online* version, in which items are given one by one. It solves the online version of the MSDPM problem where the locations are given one by one over time. BMIDP is a Bin-Packing variant, since Bin-Packing is NP-Hard [5, 9], BMIDP is presumably hard as well. Therefore: (i) We adjust First Fit (FF) online Bin-Packing approximation and call it Max Item Doubled First Fit (MIDFF) for the BMIDP *online* version. (ii) We

Algorithm 1: MIDFF approximation for BMIDP (def 5.2)

```
Initialize: one empty bin
for All items i = 1,2,...,n do
    for All bins j = 1,...,last opened bin do
        MAX ← max(size of item i, max item size in bin j)
        SUM ← sum of item sizes in bin j+size of item i+MAX
        if SUM ≤ V then
            Pack item i in bin j.
            Break inner loop to proceed with next item.
        end if
    end for
    if Item i did not fit any bin then
        Create new bin and pack item i in it.
    end if
end for
```

adjust First Fit Decreasing (FFD) offline Bin-Packing approximation and call it Max Item Doubled First Fit Decreasing (MIDFFD) for the BMIDP *offline* version. MIDFF and MIDFFD works in a very similar way to FF and FFD respectively, see Algorithm 1. Given an item we iterate over bins already open by the order of bin openings. We put the item in the first bin in which it fits. If the item doesn't fit in any bin, we open a new bin and put the item in it. In MIDFF and MIDFFD the current weight of a bin is the sum of sizes of items in it plus the size of the maximum item in the bin. This is the variation from FF and FFD. Therefore, when checking an item's fit to a bin, we first check if its weight is greater than the current maximum item in the bin. As with FFD and FF, MIDFFD is the same as MIDFF, except that improved approximation factor is obtained by first sorting (we know all locations in advance) the items in decreasing order by size. We first prove an approximation factor ≤ 1.5 for MIDFFD, than for MIDFF we show via extensive simulations an average approximation factor of 1.7.

5.1 MIDFFD approximation for offline BMIDP

We are using in the following proofs an equivalent Bin-Packing definition, where all sizes: $a_i \in (0, 1]$ $(1 \leq i \leq n)$, and $V = 1$. The equivalence is straightforward by dividing the original sizes by the original capacity. Thus definition 5.2 becomes:

Definition 5.3. BMIDP *equivalent.* Given a list of n items $S = \{1, 2, \ldots, n\}$ with sizes: $a_i \in (0, \frac{1}{2}]$ $i = 1,\ldots,n$. Find the minimum number B, of disjoint subsets of S, called Bins, such that, $S = S_1 \uplus S_2 \uplus \ldots \uplus S_B$ and for each subset $S_j \in \{S_1, \ldots, S_B\}$ $\sum\limits_{i \in S_j} a_i + \max\limits_{i \in S_j}\{a_i\} \leq 1$

In order to prove approximation factor ≤ 1.5 to MIDFFD we start with two lemmas (we extend the proof of 1.33 approximation factor for FFD [24], to accommodate MIDFFD).

Lemma 5.4. *Let the n items of definition 5.3 be sorted in descending order: $\frac{1}{2} \geq a_1 \geq \ldots \geq a_n$. If an optimal packing for* BMIDP *(OPT) uses B bins, then all bins in* MIDFFD *packing for* BMIDP*, after bin number B, must have items of size $\leq \frac{1}{3}$*

Proof. Suppose by contradiction that a_i is the first item $> \frac{1}{3}$ to be packed by MIDFFD in bin $B+1$. All items that where packed by MIDFFD before a_i are greater or equal to it since all items are sorted in descending order of size, therefore $a_1, a_2, \ldots, a_{i-1}$ are also greater then $\frac{1}{3}$, and by the assumption on a_i they are packed in the first B bins. Thus each of the first $B+1$ bins has exactly one item each, and $i = B+1$. It follows that in any packing solution, including the optimal, there must be at least $B+1$ bins, a contradiction. □

Lemma 5.5. *Let the n items of definition 5.3 be sorted in descending order: $\frac{1}{2} \geq a_1 \geq \ldots \geq a_n$. If an optimal packing for* BMIDP *(OPT) uses B Bins denoted $S_1, \ldots, S_B$, then the number of items* MIDFFD *packs, in bins after bin number B, is at most $B - 1$.*

Proof. Suppose by contradiction that the number of items MIDFFD packs in bins after bin number B, is at least B. Denote the bins of MIDFFD packing by $T_1, \ldots, T_{B_1}$ $B_1 > B$. Denote the sizes of the first B items of those packed in bins after bin number B: $x_1, x_2, \ldots, x_B$. Let $W_j = \sum\limits_{i \in T_j} a_i + \max\limits_{i \in T_j}\{a_i\}$ denote the total size of items packed in bin number j, including maximum item doubled. The items packed by MIDFFD in the first B bins $T_1, \ldots, T_B$ + the B items $x_1, x_2, \ldots, x_B$ are a subset of the total n items, therefore: $\sum\limits_{i=1}^{n} a_i + \sum\limits_{j=1}^{B} \max\limits_{i \in T_j}\{a_i\} \geq \sum\limits_{j=1}^{B} W_j + \sum\limits_{j=1}^{B} x_j = \sum\limits_{j=1}^{B} (W_j + x_j)$ $W_j + x_j > 1$ otherwise MIDFFD would pack x_j in T_j, therefore: $\sum\limits_{j=1}^{B} (W_j + x_j) > \sum\limits_{j=1}^{B} 1 = B$. Recall that items are sorted in decreasing order, therefore MIDFFD packs items in bins in an optimal way regarding doubling the maximum item of a bin, thus: $\sum\limits_{j=1}^{B} \max\limits_{i \in T_j}\{a_i\} = \sum\limits_{j=1}^{B} \max\limits_{i \in S_j}\{a_i\}$, and we get a contradiction: $\sum\limits_{i=1}^{n} a_i + \sum\limits_{j=1}^{B} \max\limits_{i \in S_j}\{a_i\} > B$, because all the n items can be packed in B bins by OPT. □

Theorem 5.6. *Max Item Doubled First Fit Decreasing (*MIDFFD*) uses at most 1.5B bins if the optimal packing for* BMIDP *(OPT) uses B Bins.*

Proof. There are at most $B-1$ extra items by lemma 5.5, each of size $\leq \frac{1}{3}$ by lemma 5.4. Thus, there can be at most $\frac{B-1}{2}$ extra bins. Therefore, the total number of bins needed by MIDFFD is $B + \frac{B-1}{2} = 1.5B - \frac{1}{2}$ □

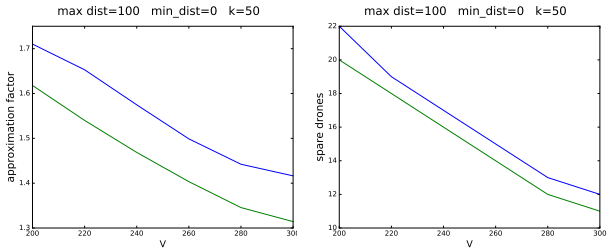
5.2 MIDFF approximation for online BMIDP

For the online version of BMIDP we hypothesize that the approximation factor of MIDFF is ≤ 2. In order to find an estimation of the approximation factor of MIDFF, we conducted extensive experiments with various parameters settings to check what values the approximation factor gets. Note that all distance units are presented as battery units required for traveling this distance. The parameters in the experiments are: (i) k - Number of locations for persistent monitoring, (ii) Maximum distance - The maximum distance from h_1 of all locations for persistent monitoring. (iii) Minimum distance - The minimum distance from h_1 to all locations for persistent monitoring. (iv) V - Capacity of all Bins, which equales the battery capacity L of all drones. This value must be at least twice as Maximum distance.

While searching for the value estimation for the MIDFF approximation factor we also checked the influence of the parameters on it and on the number of spare drones. We also checked the same for MIDFFD to confirm our theoretical result of theorem 5.6 and to make MIDFF and MIDFFD comparable. In order to avoid intractable computation of BMIDP optimal value, we used the minimal number of spare drones = $\frac{\sum\limits_{i \in S} a_i}{V - \min\limits_{i \in S}\{a_i\}} < OPT$, instead of OPT.

Therefore, the experimental approximation factor is a strict upper bound of the real approximation factor.

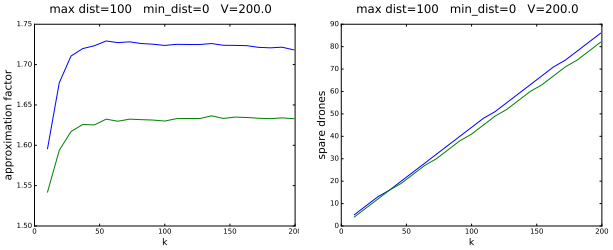
In all cases shown in Figures 9 - 11, we iterate at least 100 times, where in each iteration we generate uniformly random samples of k locations distances in the range (min dist, max dist). For both versions of the BMIDP problem, those are the item sizes to be packed. For the online version, the items ordering is kept random as in the sample, simulating locations that are given one by one along time, with no constraint on the order, as in the MSDPM online version problem. For the offline version, items are sorted by decreasing order, simulating a set of locations given in advance as in the MSDPM offline version problem. We compute the number of spare drones needed and the approximation factor and report the average over the 100 iterations. Each of the figures presents a unique setup of three parameters as constants, and present the influence of the remaining fourth parameter as it changes, i.e. the impact of the change in the fourth parameter on the approximation factor and on the number of spare drones.



(a) expr. approx. factor **(b) spare drones**

Figure 9: Influence of increasing battery capacity L.

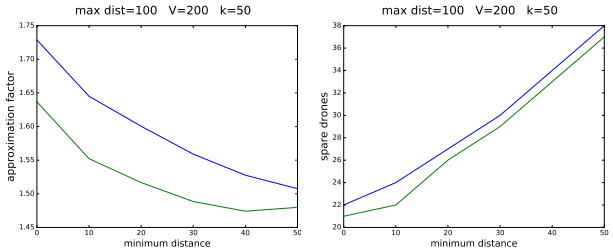
In all the graphs, upper line is the MIDFF results and lower (better) is the MIDFFD. Figure 9 describes the influence of L (the battery capacity) on the approximation factor and the number of spare drones. The parameters setup: (i) k - constant 50 locations to be monitored. (ii) Maximum distance - constant 100 (iii) Minimum distance - constant 0 (iv) L - Varies between 200 and 300. As L grows we need less spare drones, because more locations can be allocated to each spare drone (in the BMIDP problem: we can pack more items in a bigger bin). The approximation factor also gets better with increased capacity because packing gets easier as more efficient packing opportunities are available, making less penalty to greedy suboptimal decisions.



(a) expr. approx. factor **(b) spare drones**

Figure 10: Influence of increasing k, the number of locations.

Figure 10 reports the influence of k, the size of the location set l on the approximation factor and the number of spare drones. Parameters setup: (i) k - Varies from 10 to 200. (ii) Maximum distance - constant 100 (iii) Minimum distance - constant 0 (iv) L - constant 200. As k grows we need more spare drones, because we must visit more locations with the same energy capacity L (have to pack more items in bins with constant capacity in the BMIDP problem). The approximation factor gets worse as k increases, because the packing becomes more complicated and there is greater penalty for greedy suboptimal choices that are made more frequent. However, there is an asymptotic limit which evolves from the real approximation factor. The experimental value is compatible with our theoretic result in theorem 5.6 of 1.5 for MIDFFD, the experimental approximation factor is a strict upper bound for the real one, therefore > 1.5 . We also see here 1.7 asymptotic limit for MIDFF.



(a) expr. approx. factor **(b) spare drones**

Figure 11: Influence of increasing the minimum distance from h_1.

Figure 11 describes the influence of the minimum distance on the approximation factor and the number of spare drones. Parameters setup: (i) k - constant 50 (ii) Maximum distance - constant 100 (iii) Minimum distance - Varies from 0 to 50. (iv) L - constant 200. As the minimum distance grows, the random samples of k locations distances in range (min dist, max dist) have higher distance values, therefore the items to pack are bigger. We need more spare drones, since each drone can visit less locations that are more distant (we have to pack bigger items in bins with constant capacity, in the BMIDP problem). The approximation factor on the other hand gets better with increased minimum distance, because the items becomes more homogeneous as the samples are from a smaller range and we get more uniform item sizes. Packing gets easier when items are more uniform as there's less penalty to greedy suboptimal decisions, which are more likely to be corrected in next items (distances) packings, compared to the case of heterogeneous items (distances).

6 CONCLUSIONS

In this paper we introduced the problem of enabling persistent execution of a multi-drone task under energy limitations. The drones are given a set of locations and their task is to ensure that at least one drone will be present, over each location at any given time. Because of energy limitations, drones must be replaced from time to time, and fly back home where their batteries can be replaced. Our goals are to identify the minimum number of spare drones needed to accomplish the task while no drone battery drains and to provide a drone replacement strategy.

ACKNOWLEDGEMENTS

This research was funded in part by a grant from MOST, Israel and the JST Japan, and ISF grants no.1488/14 and 1337/15.

REFERENCES

[1] Brenda S Baker. 1985. A new proof for the first-fit decreasing bin-packing algorithm. *Journal of Algorithms* 6, 1 (1985), 49–70.

[2] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. 2016. The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering* 99 (2016), 300–313.

[3] Oleg Burdakov, Jonas Kvarnström, and Patrick Doherty. 2017. Optimal scheduling for replacing perimeter guarding unmanned aerial vehicles. *Annals of Operations Research* 249, 1-2 (2017), 163–174.

[4] John Carlsson, Dongdong Ge, Arjun Subramaniam, Amy Wu, and Yinyu Ye. 2009. Solving min-max multi-depot vehicle routing problem. *Lectures on global optimization* 55 (2009), 31–46.

[5] Edward G Coffman Jr, Michael R Garey, and David S Johnson. 1996. Approximation algorithms for bin packing: A survey. In *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 46–93.

[6] György Dósa. 2007. The tight bound of first fit decreasing bin-packing algorithm. *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies* (2007), 1–11.

[7] Burak Eksioglu, Arif Volkan Vural, and Arnold Reisman. 2009. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering* 57, 4 (2009), 1472–1483.

[8] Leah Epstein. 2006. Online bin packing with cardinality constraints. *SIAM Journal on Discrete Mathematics* 20, 4 (2006), 1015–1030.

[9] Michael R Garey and David S Johnson. 1979. Computers and intractibility. (1979).

[10] Konstantinos Kanistras, Goncalo Martins, Matthew J Rutherford, and Kimon P Valavanis. 2015. Survey of unmanned aerial vehicles (UAVs) for traffic monitoring. In *Handbook of unmanned aerial vehicles*. Springer, 2643–2666.

[11] Jonghoe Kim, Byung Duk Song, and James R Morrison. 2013. On the scheduling of systems of UAVs and fuel service stations for long-term mission fulfillment. *Journal of Intelligent & Robotic Systems* (2013), 1–13.

[12] Gilbert Laporte. 1992. The vehicle routing problem: An overview of exact and approximate algorithms. *European journal of operational research* 59, 3 (1992), 345–358.

[13] Neil Mathew, Stephen L Smith, and Steven L Waslander. 2013. A graph-based approach to multi-robot rendezvous for recharging in persistent tasks. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*. 3497–3502.

[14] Yongguo Mei, Yung-Hsiang Lu, Yu Charlie Hu, and CS George Lee. 2006. Deployment of mobile robots with energy and timing constraints. *IEEE Transactions on robotics* 22, 3 (2006), 507–522.

[15] Vera Mersheeva and Gerhard Friedrich. 2015. Multi-UAV Monitoring with Priorities and Limited Energy Resources.. In *ICAPS*. 347–356.

[16] Derek Mitchell, Micah Corah, Nilanjan Chakraborty, Katia Sycara, and Nathan Michael. 2015. Multi-robot long-term persistent coverage with fuel constrained robots. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 1093–1099.

[17] Jairo R Montoya-Torres, Julián López Franco, Santiago Nieto Isaza, Heriberto Felizzola Jiménez, and Nilson Herazo-Padilla. 2015. A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering* 79 (2015), 115–129.

[18] Thanh H Nguyen, Francesco M Delle Fave, Debarun Kar, Aravind S Lakshminarayanan, Amulya Yadav, Milind Tambe, Noa Agmon, Andrew J Plumptre, Margaret Driciru, Fred Wanyama, et al. 2015. Making the most of our regrets: Regret-based solutions to handle payoff uncertainty and elicitation in green security games. In *International Conference on Decision and Game Theory for Security*. Springer, 170–191.

[19] Nikhil Nigam and Ilan Kroo. 2008. Persistent surveillance using multiple unmanned air vehicles. In *IEEE Aerospace Conference*. 1–14.

[20] Hyorin Park and James R Morrison. 2014. On the resources required to provide persistent robotic service: Multiple immobile customers and a single service station. In *Proceedings of the Asia Pacific Industrial Engineering and Management Systems Conference (APIEMS'14)*. 1–8.

[21] Markus Quaritsch, Karin Kruggl, Daniel Wischounig-Strucl, Subhabrata Bhattacharya, Mubarak Shah, and Bernhard Rinner. 2010. Networked UAVs as aerial sensor network for disaster management applications. *e & i Elektrotechnik und Informationstechnik* 127, 3 (2010), 56–63.

[22] Byung Duk Song, Jonghoe Kim, Jeongwoon Kim, Hyorin Park, James R Morrison, and David Hyunchul Shim. 2014. Persistent UAV service: an improved scheduling formulation and prototypes of system components. *Journal of Intelligent & Robotic Systems* 74, 1-2 (2014), 221–232.

[23] Matthew Turpin, Nathan Michael, and Vijay Kumar. 2015. An approximation algorithm for time optimal multi-robot routing. In *Algorithmic Foundations of Robotics XI*. Springer, 627–640.

[24] Mark Allen Weiss. 2012. *Data Structures and Algorithm Analysis in Java. Mark Allen Weiss.* Pearson education.

[25] D Damon Willens, Chris Proudlove, Terry Miller, et al. 2017. DRONE AWAY: BOMBS, PHOTOGRAPHS, PIZZAS, HEARTS, AND HEADACHES. *The Brief* 46, 4 (2017).

[26] Binzhou Xia and Zhiyi Tan. 2010. Tighter bounds of the First Fit algorithm for the bin-packing problem. *Discrete Applied Mathematics* 158, 15 (2010), 1668–1675.