Decentralized dynamic task allocation for UAVs with limited communication range

Marc Pujol-Gonzalez[a], Jesus Cerquides[a], Pedro Meseguer[a],
Juan A. Rodriguez-Aguilar[a,*], Milind Tambe[b]

[a]*Artificial Intelligence Research Institute of the Spanish National Research Council
(IIIA-CSIC), Campus de la UAB, 08193 Bellaterra, Spain*
[b]*University of Southern California, Los Angeles, CA 90089*

Abstract

We present the Limited-range Online Routing Problem (LORP), which involves a team of Unmanned Aerial Vehicles (UAVs) with limited communication range that must autonomously coordinate to service task requests. We first show a general approach to cast this dynamic problem as a sequence of decentralized task allocation problems. Then we present two solutions both based on modeling the allocation task as a Markov Random Field to subsequently assess decisions by means of the decentralized Max-Sum algorithm. Our first solution assumes independence between requests, whereas our second solution also considers the UAVs' workloads. A thorough empirical evaluation shows that our workload-based solution consistently outperforms current state-of-the-art methods in a wide range of scenarios, lowering the average service time up to 16%. In the best-case scenario there is no gap between our decentralized solution and centralized techniques. In the worst-case scenario we manage to reduce by 25% the gap between current decentralized and centralized techniques. Thus, our solution becomes the method of choice for our problem.

Keywords: task allocation, unmanned aerial vehicles, max-sum, decentralized

1. Introduction

Unmanned Aerial Vehicles (UAVs) are an attractive technology for large-area surveillance [1]. Today, there are readily available UAVs that are reasonably cheap, have many sensing abilities, exhibit a long endurance and can communicate using radios. Several applications can be efficiently tackled with a

---

*Corresponding author

Email addresses: mpujol@iiia.csic.es (Marc Pujol-Gonzalez), cerquide@iiia.csic.es (Jesus Cerquides), pedro@iiia.csic.es (Pedro Meseguer), jar@iiia.csic.es (Juan A. Rodriguez-Aguilar), tambe@usc.edu (Milind Tambe)

team of such UAVs: power line monitoring, fire detection, and disaster response among others [2].

UAVs have traditionally been controlled either remotely or by following externally-designed flight plans. Requiring human operators for each UAV implies a large, specialized and expensive human workforce. Likewise, letting UAVs follow externally prepared plans introduces a single point of failure (the planner) and requires UAVs with expensive (satellite) radios to maintain continuous communication with a central station. These constraints are acceptable in some application domains, other applications require more flexible techniques.

For instance, consider a force of park rangers tasked with the surveillance of a large natural park. Upon reception of an emergency notification, the rangers must assess the situation as quickly as possible. With this aim, they could deploy a team of UAVs continuously fly throughout the park. Thereafter, they could issue requests for their UAVs to check certain locations. To maintain the cost-effectiveness of the approach, such UAVs cannot employ expensive communication devices. Thus, the UAVs would have limited communication ranges, oftentimes significantly smaller than the park's extension. Notice that, in this setting, neither human remote control nor centralized planning is feasible due to such communication constraints.

In this scenario, a possibility would be to deploy the planes with a fixed mission, so that they just go to the desired location(s), perform any required checks and fly back to some base. However, this approach is grossly unequipped to handle the dynamism of the problem. That is, during a critical situation (e.g., when there is an actual fire in the park), the rangers will receive many reports, and will quickly run out of UAVs to handle them. A possibly better approach involves UAVs that can act autonomously and can coordinate between them. Such UAVs keep flying around the park and coordinate to attend requests as they are introduced into the system by rangers, thus hopefully improving their effectiveness.

Nevertheless, the autonomous operation and coordination of UAVs is an open research question receiving increasing attention. It involves challenges ranging from low-level operational details of flight control to high level coordination between UAVs [3]. In this paper we assume that low level control can be handled entirely by the UAVs' auto-pilot systems, and focus on high-level coordination challenges instead.

There are two main approaches to enable such coordination in the literature, each best suited for a different kind of UAV missions. On the one hand, the objective in exploratory missions is to collect accurate information and keep it updated. Hence, the first approach focuses on shared information collection, fusion and maintenance techniques to fulfill these missions [4]. On the other hand, some applications involve specific tasks that the UAVs should carry. For instance, check requests introduced by the rangers can be considered as tasks to be performed by the UAVs. Hence, coordination between UAVs in this context implies making decisions about which UAV should conduct each task.

Unfortunately, most state-of-the-art multi-agent allocation mechanisms cannot be employed in our settings, particularly due to the communication range

limitation. Hence, we first identify the characteristics and particular challenges of such applications and define the Limited-range Online Routing Problem (LORP) to capture them. After studying these characteristics, we advocate for a solution approach where UAVs make quick decisions based solely on local information, where the neighbors of a UAV are those UAVs with which it can directly communicate at a given point in time.

Next, we introduce a solution approach based on representing the problem as a factorized utility function (actually a Markov Random Field) and optimizing it using the Max-Sum distributed algorithm. This technique provides two significant advantages: (i) the problem is represented as a single global utility function. Hence, the collective behavior of the UAVs is easier to predict and reason about than when it emerges from the definition of individual behaviors; and (ii) we can draw results from the extensive Max-Sum literature, including its theoretical properties (e.g.: convergence and quality guarantees [5, 6]) and encouraging experimental results [7].

We present an initial development of this solution in Section 4, assuming that the cost of servicing each request is independent of other requests assigned to the UAV. Thereafter, we introduce a second solution where UAVs adjust their estimations of the cost of servicing a task depending on their workload, with a slight increment in complexity. As a result, UAVs using this solution are better equipped to dynamically capture and exploit the distribution of incoming requests.

The contributions of this paper are the following:

- We introduce the LORP and identify its need for quick loops of assessment, decision and action in highly dynamic, communication constrained UAV applications.

- We propose to cast the LORP as an optimization problem and distributedly solve it using Max-Sum, thus providing great flexibility to introduce new heuristics without modifying the solving algorithm.

- We capitalize on this advantage by introducing the workload heuristic, which exploits the dynamic characteristics of the problem to make better decisions.

- We empirically evaluate the proposed algorithms, showing that: (i) the proposed approach allows for effective request allocation in a highly dynamic, communication constrained domain; and (ii) our workload-based solution achieves between 6% and 16% lower service times than current state-of-the-art methods, and its actual performance comes very close to that of centralized solutions.

The contributions above build upon and extend the preliminary work presented in [8]. However, notice that here we improve that work by: (i) relating it to current state-of-the-art literature, (ii) presenting the LORP as a general problem and showing how to cast it as a distributed task allocation problem, (iii) providing proofs of the results that allow us to improve Max-Sum's efficiency,

3

and (iv) performing a thorough empirical comparison between the proposed algorithms and the related methods.

The rest of the paper is organized as follows. First, Section 2 reviews current coordination approaches. Next, Section 3 describes our problem in detail, along with an example scenario. Thereafter we present a first MRF-based solution in Section 4, explaining how the LORP can be encoded and efficiently solved using Max-Sum. Then, Section 5 shows how to improve this representation to better spread the workload between planes. Finally, Sections 7 and 8 empirically evaluate our approach, and Section 9 presents some conclusions and further lines of work.

2. Related Work

There exists a large body of research related to multi-agent (and/or multi-robot) task allocation.

Our problem is actually a variant of the Single-Task, Single-Robot (ST-SR) dynamic Multiple Travelling Salesman Problem (MTSP) as described in Gerkey's taxonomy of multi-robot task allocation systems [9]. In the MTSP, a number of robots (or agents) have to visit a number of waypoints in the minimum possible time, while new waypoints may be introduced at any time. In fact, our variant is even harder than the MTSP because of the communication range limitation, and hence it is also strongly NP-Hard.

Most works on the MSTP focus on market-based allocation mechanisms [10, 11, 12, 13], where tasks are allocated to agents using auctions. The pre-cursor in this area is the auction algorithm [14], an iterative distributed algorithm to optimally solve the classic *assignment problem* [15]. The assignment problem is a static task allocation problem where several tasks can be allocated to a number of agents. Each agent has a different cost for performing each task, and can only be assigned to a single task. Then, the objective is to find the allocation that minimizes the total cost incurred after assigning all tasks.

Nonetheless, on most actual-world applications there may be more tasks than agents. If the cost for a robot to perform two tasks is simply the addition of the costs of performing each task separately, then the above algorithm still yields an optimal allocation. However, in routing problems the cost of visiting multiple waypoints depends on the relative positions between those waypoints. Hence, the auction algorithm is not optimal in this case, and can lead to arbitrarily bad solutions [9].

A first step to mitigate this problem is introduced in [11]. Instead of performing an iterative auction, the idea here is to perform a parallel single-item auction for each task. These auctions can be resolved in a single round. Afterwards, tasks are continuously re-auctioned, but with agents' bids updated to account for the new state of the world. This approach, described as the Parallel Single-Item (PSI) auctions mechanism in [16], improves the robustness of the system and equips it to handle changing conditions and/or the introduction of new tasks. Although PSI auctions capture some of those dependencies by

reallocating tasks when conditions change, another possibility is to explicitly consider them. In the Sequential Single-Item auctions [17] method, agents perform an iterative auction where a single task is assigned at each iteration. After each round, agents update their bids according to the tasks they already got in previous iterations, hence taking into account the synergies between tasks. This approach even has quality guarantees in static domains [18], but is not well-suited for dynamic domains because it requires both an arbitrary number of communication cycles and global communication.

An entirely different approach to task allocation is to employ consensus algorithms [19, 20] to let agents maintain consistent information about the system's state. After reaching consensus, each agent can independently compute its plan using a deterministic algorithm, and the resulting plans are guaranteed to be consistent with each other. The main advantage of these algorithms is that they have been shown to converge even on time-varying network topologies. However, the time required to reach such convergence is unbounded and they cannot converge during network partitions. Hence, consensus-based works typically strive to either guarantee network connectedness [21] or minimize the impact of such a potentially large decision making time. For instance, [22] presents a UAV coordination model for fire perimeter tracking, where UAVs reach a consensus on which section of the perimeter is to be monitored by each UAV. When the conditions change, the UAVs keep monitoring their section until a new consensus is eventually reached. Because the fire perimeter is unlikely to evolve very rapidly, the previous assignment works relatively well even if takes a long time to reach a new consensus.

Finally, some researchers tried to combine the advantages of market-based mechanisms (quick decisions) with those of the consensus-based ones (eventual consistency) [23]. Along these lines, Choi et. al. [24] introduced the Consensus Based Bundle Algorithm (CBBA), that directly reaches a consensus on (an auction-based) solution. Nonetheless, this approach still suffers from the low resiliency to network disconnections.

As detailed in the next section, LORP scenarios are highly dynamic and involve very rapid changes in the communication network topology, including frequent disconnections. Hence, consensus-based approaches do not seem particularly well suited for this domain. Moreover, we can only employ the simpler auction-based approaches (such as PSI) due to the lack of global communication.

3. The limited-range online routing problem

In this section we present the Limited-range Online Routing Problem. We first describe the problem intuitively and present an example scenario. Thereafter, we show how it can be cast as a distributed task allocation problem.

3.1. Motivating Example

Consider a force of park rangers that acquires a fleet of unmanned UAVs to help them monitor a large natural park. The force intends to request its
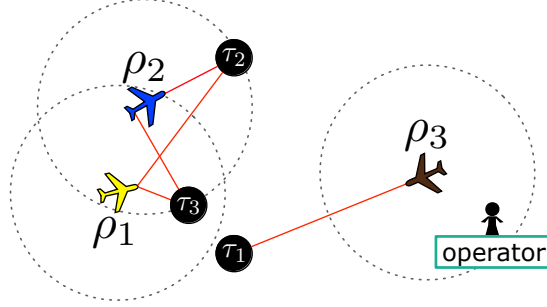
Figure 1: Example LORP scenario.

UAVs to check certain locations when alerts are received, or as part of their routine surveillance plans. In turn, UAVs are expected to fulfill these requests as quickly as possible. Specifically, UAVs should try to minimize the average request service time, where service time is the amount of time passed between the issue of a request by an operator and the arrival of an UAV to the request's location.[1] A common approach to this kind of problems is to adopt a centralized strategy: UAVs' routes are planned at a central station, which is in charge of commanding them. Therefore, the central station guarantees cooperation between UAVs. However, if the natural park is significantly larger than the UAVs communication ranges, performing centralized planning is unfeasible because the resulting plan can not be effectively transmitted to UAVs. Figure 1 represents a snapshot of what the rangers' scenario may look like at some point in time. In this example, the force owns three UAVs labeled ρ_1, ρ_2, and ρ_3. The communication range of each UAV is drawn as a dotted circle around it. By inspecting these circles, we observe that UAVs ρ_1 and ρ_2 can communicate between them. Moreover UAV ρ_3 can communicate with the rangers' operator. However, there is no way for UAVs ρ_1 and ρ_2 to communicate neither with UAV ρ_3 nor with the operator. The scenario contains three targets represented by the black circles labelled with τ_1, τ_2, and τ_3. A solid line linking a UAV with a target indicates that the UAV is aware of that target. For instance, UAV ρ_1 and UAV ρ_2 know that targets τ_2 and τ_3 exist, but they are not aware of target τ_1. Given this scenario, a centralized planner would probably send UAV ρ_1 to targets τ_3 and τ_1, UAV ρ_2 to target τ_2, and leave ρ_3 idle. However, this plan of action can never be computed nor implemented when assuming that UAVs have limited communication range, because there is simply no way for ρ_1 to discover τ_1 at this particular point in time.

Notice that the LORP itself does not specify how agents should represent and relay information about which requests are pending and/or completed, nor which messages can be exchanged. Hence, it is part of a LORP solution to define a specific model and an algorithm that allows agents to make specific decisions. Moreover, within this decision lies an essential trade-off that designers must

---

[1]We assume that a request is serviced as soon as a UAV reaches its location.

face. Intuitively, computing more accurate plans improves the quality of the solution *to the currently known part of the problem*, and hence it is a desirable goal. However, the better plans we want to compute, the more time our UAVs have to invest into it, thus hindering their ability to adapt to the new incoming requests. In contrast, we advocate for an approach specifically focused towards better handling *the unknown part of the problem*, as explained in the following section.

3.2. Approach

Notice that, unlike our example scenario above, the LORP is a highly dynamic problem where UAVs constantly move and new requests can be introduced at any time. As a result, we argue that LORP solutions should strive to make quick decisions. On the one hand, a long decision making process can lead to arbitrarily bad decisions being made. That is, a decision that was good when the decision making process started may end up being arbitrarily bad if the process itself took too much time (during which the scenario's conditions kept evolving). On the other hand, making complex decisions (such as long term plans for each UAV) is futile, because there's a high probability that these plans are invalidated. Likewise, operating in a decentralized manner introduces an additional challenge: how to spread and maintain a consistent view of the system between agents. For instance, a LORP solution must specify how UAVs notify others of newly introduced requests, as well as of already serviced ones.

In our approach, we deal with these issues by defining three main processes that are going on simultaneously:

1. At any time, an operator may introduce a new request by notifying a single UAV in its range, which becomes the request's *owner*. If no UAV is currently in its range, the operator must wait until one comes around. So long as a UAV is the owner of a request, it is responsible for the eventual servicing of that request. However, ownership of a request may be transferred to another UAV as explained next. Since there is exactly one owner of each request at a particular point in time, this guarantees a consistent (yet distributed) view of the system by the UAVs.

2. Concurrently, UAVs run cycles of a request reallocation process. The reallocation process is a sequential process with three phases. First, the UAVs construct a snapshot of the current situation by broadcasting their location and the requests they currently own. Next, the UAVs run some decentralized request allocation algorithm based on the collected information. This phase may take several communication rounds between neighboring UAVs, but such number of rounds must be strictly limited and known beforehand. At the end of these rounds, the algorithm must have decided on which plane should own each request. Finally, these decisions are executed by exchanging the requests' ownership between planes. When a full reallocation cycle finishes, the UAVs start a new one with updated information.

3. Meanwhile, each UAV flies towards the nearest it owns or tries to get in range of the closest operator if it does not own any request. When a UAV reaches the location of a request it owns, the request is considered serviced and removed from the system.
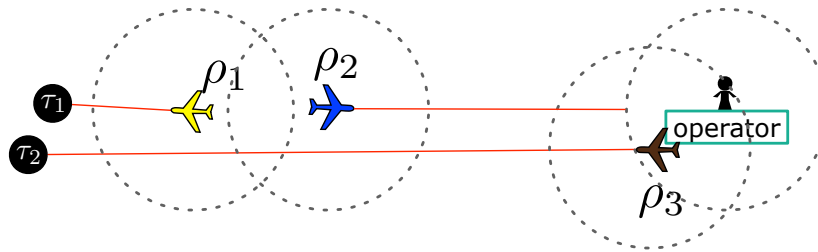
We argue that this approach is particularly well suited for the LORP problem. First, the ownership concept frees the allocation algorithms from having to maintain a consistent situational awareness because only the current owner of a request may decide to allocate it to another UAV. Additionally, by enforcing short allocation cycles the UAVs can make decisions quickly and re-consider them as the situation evolves. Obviously, enforcing quick allocations prevents us from employing sophisticated methods that best consider the currently known part of the problem. However, we expect the benefit of being able to constantly reconsider those decisions to make up for –and even surpass– the loss in plan accuracy from employing simpler allocation methods.

Arguably, another caveat of this approach is that it gives up the possibility of a more controlled network deployment that tries to maximize the UAV's coverage of space, as introduced in other works [21, 25]. Nonetheless, our proposal provides several significant qualities that those works cannot achieve. First and foremost, our proposal is inherently robust to failures. Even if a massive disruption renders several UAVs useless, our system will remain operative to make the best possible use of the remaining ones. Second, by avoiding any centralization whatsoever, our approach can scale to arbitrarily large numbers of tasks, UAVs and space to cover. Finally, it allows for the emerge of self-organized network structures adapted to the situation's characteristics. Because this characteristic is hard to understand, next we introduce a small example that illustrates how our approach works, and highlights how the UAVs may self-organize while following it.
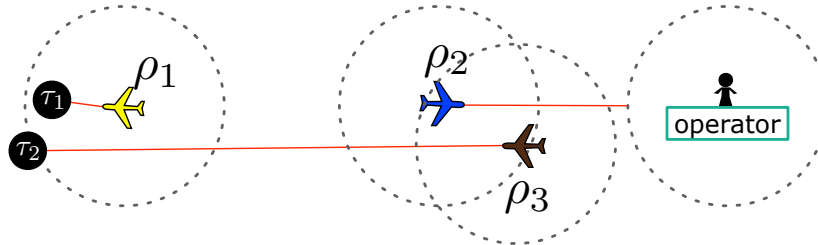
3.3. Example of operation

Consider the example in Figure 2a. Here UAV ρ_2 has no tasks assigned, and hence is flying back to get into operator range (as per process 3 explained above). Meanwhile, UAV ρ_1 is trying to complete task τ_1, whereas UAV ρ_3 just got task τ_2 from the operator, and starts moving towards its location (also because of process 3). After a while, as shown in Figure 2b, UAVs ρ_2 and ρ_3 get into communication range, and start a reallocation round (process 2). At this point, a sensible allocation method probably assigns task τ_2 to UAV ρ_2. Hence, τ_2's ownership is transferred from UAV ρ_3 to UAV ρ_2, which becomes the new owner of this task. Finally, (by process 3) ρ_2 now advances towards τ_2 whereas ρ_3 goes back to the operator to gather new tasks, as shown in Figure 2c. Notice that, if further tasks are introduced around τ_1 and τ_2, UAVs ρ_2 and ρ_3 will keep acting as "messengers" bringing those tasks from the operator to UAV ρ_1 as illustrated here. Thus, this example shows how UAVs following our approach can naturally form task relay chains, even if the actual tasks are further than what an connected communication chain could reach.
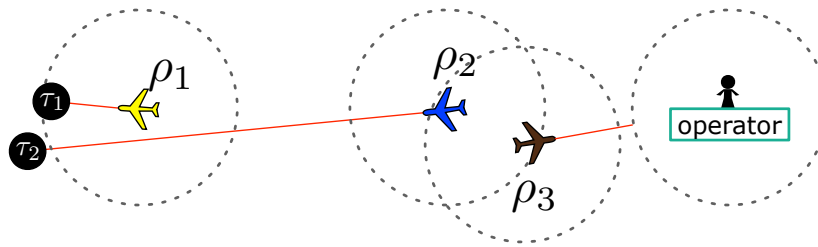
8

(a) Initial situation

(b) A reallocation negotiation begins

(c) Task relaying is now complete

Figure 2: Example of task relaying without an enforced communication chain.

4. Coordination using Independent Valuations

The work in [26] shows that it is possible to model multi-agent problems as MRFs, and then let agents find the Maximum-A-Posteriori (MAP) of that field to make their decisions. In this section we show how a snapshot of a LORP problem can be encoded as a MRF, assuming that requests are independent between them (i.e, that the cost for a UAV to service two requests is the same as the sum of costs of servicing them separately). Thereafter we use Max-Sum to let UAVs compute an allocation in a distributed manner.

4.1. Problem Encoding as Binary MRF

Prior to encoding our problem, some notation is in place. Henceforth, Let $R = \{\tau_1, \ldots, \tau_m\}$ be a set of requests, $P = \{\rho_1, \ldots, \rho_n\}$ be a set of UAVs, r and p be indexes for requests and UAVs respectively, $R_p \subseteq R$ be the set of requests that UAV ρ_p can service, and $P_r \subseteq P$ be the set of UAVs that can service request τ_r. A naive encoding of the requests-to-UAVs allocation as an MRF is:

- Create a variable x_r for each request τ_r. The domain of this variable is the set of UAVs that can service the request, namely P_r. If x_r takes value ρ_p, it means that request τ_r will be serviced by UAV ρ_p.

- Create an constraint c_p for each UAV ρ_p, that evaluates the cost of servicing the requests assigned to ρ_p. We assume independence, so the cost of servicing a set of requests is the sum of the costs of servicing each request.

X_p is the set of variables that have ρ_p in their domains. An assignment of values to each of the variables in X_p is noted as $\mathbf{X_p}$. Solving the MAP problem amounts to finding the combination of request-to-UAV assignments $\mathbf{X}^*$ that satisfies $\mathbf{X}^* = \arg\min_{\mathbf{X}} \sum_{p \in P} c_p(\mathbf{X_p})$.

Figure 3a shows an encoding of the motivating example. There is a variable for each request. The domain of x_1 is the set of UAVs that can service request τ_1. This is effectively the set of all UAVs that are in communication range of the owner of τ_1. Hence, the domain of x_1 is just $\{\rho_3\}$. Likewise, the domain of x_2 and x_3 is $\{\rho_1, \rho_2\}$ because both UAVs can fulfill it. Next, we create a function c_p for each UAV ρ_p. Because ρ_3 can only service τ_1, the scope of function c_3 is x_1. As a result, c_3 is a unary function that specifies the cost for UAV ρ_3 to service τ_1, namely the distance between ρ_3 and τ_1 (hereafter δ_{pr} will be employed as a shorthand for the distance between ρ_p and τ_r). The scope of c_2 is $\{x_2, x_3\}$, because UAV ρ_2 can service both τ_2 and τ_3. Hence, c_2 specifies four costs for ρ_2:

1. 0 if both requests are allocated to ρ_1.
2. 2 (δ_{23}) if τ_2 is allocated to ρ_1 but τ_3 is allocated to ρ_2.
3. 2 (δ_{22}) if τ_2 is allocated to ρ_2, but τ_3 is allocated to ρ_1.
4. 4 ($\delta_{22} + \delta_{23}$) if both requests are allocated to ρ_2.

x2 x3 c2

x_2	x_3	c_2
ρ_1	ρ_1	**0**
ρ_1	ρ_2	**2**
ρ_2	ρ_1	**2**
ρ_2	ρ_2	**4**

x2 x3 c1

x_2	x_3	c_1
ρ_1	ρ_1	**6**
ρ_1	ρ_2	**5**
ρ_2	ρ_1	**1**
ρ_2	ρ_2	**0**

x_1	c_3
ρ_3	**7**

(a) Naive MRF encoding.

x_3	c_{23}
ρ_1	**0**
ρ_2	**2**

x_2	c_{22}
ρ_1	**0**
ρ_2	**2**

x_3	c_{13}
ρ_1	**1**
ρ_2	**0**

x_2	c_{12}
ρ_1	**5**
ρ_2	**0**

x_1	c_{31}
ρ_3	**7**

(b) Independent valuation.

z_{23}	z_{13}	s_3
T	T	∞
T	F	**0**
F	T	**0**
F	F	∞

z_{23}	v_{23}
F	**0**
T	**2**

z_{22}	v_{22}
F	**0**
T	**2**

z_{12}	z_{22}	s_2
T	T	∞
T	F	**0**
F	T	**0**
F	F	∞

z_{13}	v_{13}
T	**1**
F	**0**

z_{12}	v_{12}
T	**5**
F	**0**

z_{31}	s_1
F	∞
T	**0**

z_{31}	v_{31}
F	**0**
T	**7**

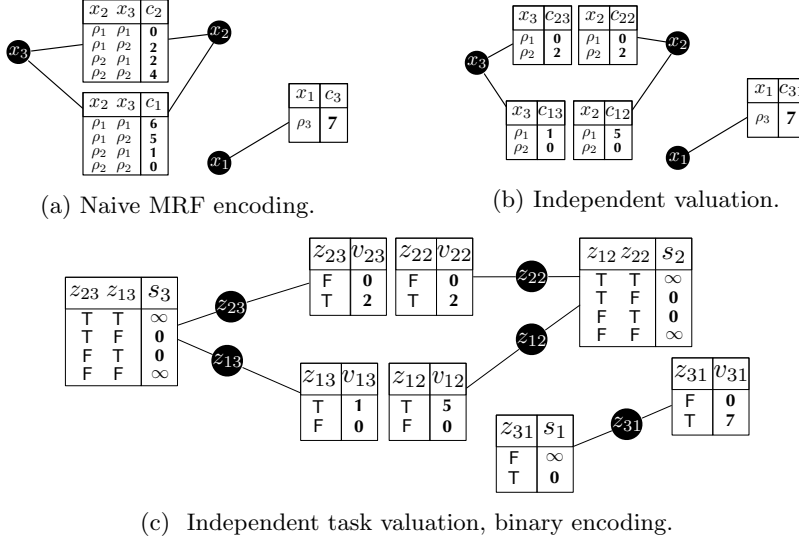(c) Independent task valuation, binary encoding.

Figure 3: Encoding the example LORP scenario.

c_1 is similarly computed. From Figure 3a costs, the optimal assignment is $\mathbf{X}^* = \langle x_1 = \rho_3, x_2 = \rho_2, x_3 = \rho_1 \rangle$.

This encoding scales poorly. First, it does not exploit the fact that we assume independence when computing the cost of servicing a combination of requests in the constraints c_p. The number of entries in c_p is the product of the domain sizes of each of the variables in its scope. Hence, the number of entries in c_p scales exponentially with respect to the number of requests that UAV ρ_p can service. However, we can exploit the independence between requests by decomposing each cost function c_p into smaller cost functions, each one evaluating the cost of servicing a single request. That is, thanks to that independence between requests, we can represent c_p as a combination of cost functions c_{pr}, one per variable in the scope of c_p, such that $c_p(\mathbf{X_p}) = \sum_{x_r \in X_p} c_{pr}(\mathbf{x_r})$. Now, the number of values to specify the cost of servicing a set of requests scales linearly with respect to the number of requests. Figure 3b represents the example in Figure 3a using this new encoding. Notice that for each UAV we specify the cost of servicing a given request when the request is assigned to it, or 0 when allocated to another UAV. However, the new encoding still suffers from redundancy. Say that another UAV ρ_4 is in the communication range of both ρ_1 and ρ_2. Since this UAV would be eligible to serve requests τ_2 and τ_3, the domain of x_2 and x_3 would become $\{\rho_1, \rho_2, \rho_4\}$. As a result, UAV ρ_1 must extend its cost function c_{12} to include a new entry where τ_2 is assigned to ρ_4, whose cost is obviously 0. Therefore, we must aim at an encoding such that a cost function c_{pr} contains only two values: δ_{pr} if τ_r is allocated to ρ_p, or 0 otherwise.

With this aim, we now convert the request variables into binary variables, replacing each original variable $x_r \in X$ by a set of binary variables z_{pr}, one per UAV in P_r. Previous c_{pr} cost functions now generate v_{pr} cost functions

11

on these binary variables. In addition, for each r, z_{pr} are linked through a selection function s_r to ensure that a request can be only serviced by a single UAV. For instance, consider variable x_2 with domain $\{\rho_1, \rho_2\}$. We create two binary variables z_{12} and z_{22}. Intuitively, z_{12} being "on" means that request τ_2 is assigned to UAV ρ_1. A selection factor linked to both z_{12} and z_{22} would guarantee that only one of the two variables is set to "on". In our example, this selection function is a cost function s_2, which introduces an infinite cost whenever there is no single variable active. Figure 3c shows the binary encoding of the example in Figure 3b.

4.2. Solving the Problem with Max-Sum

Now we optimize the Max-Sum algorithm to run on the last encoding of Section 4 (Figure 3c). Max-Sum sends messages from factors to variables and from variables to factors. However, our factor graph allows for some simplifications. Notice that each z_{pr} is only linked to cost function v_{pr} and to selector function s_r. It is direct to observe that the message that z_{pr} must send to v_{pr} is exactly the one received from s_r, while the message that it must send to s_r is exactly the one received from v_{pr}. Then, since each variable simply relays messages between the cost function and selection function it is linked to, henceforth we will disregard variables' messages and instead we consider that functions directly exchange messages.

The Max-Sum general message expression from function f to function g is

$$\mu_{f \to g}(\mathbf{Z_{f \cap g}}) = \min_{\mathbf{Z_{f-g}}} \left[f(\mathbf{Z_{f-g}}, \mathbf{Z_{f \cap g}}) + \sum_{\substack{g' \in N(f) \\ g' \neq g}} \mu_{g' \to f}(\mathbf{Z_{g' \cap f}}) \right] \quad (1)$$

where $\mathbf{Z_{f \cap g}}$ stands for an assignment to the variables in the scope of f and g, $\mathbf{Z_{f-g}}$ stands for an assignment to the variables in the scope of f that are not in g, $N(f)$ stands for the set of functions liked to f (f's neighboring functions), and $\mu_{g' \to f}$ stands for the message from function g' to f.

Observe in Figure 3c that selection and cost functions are connected by a single binary variable. Thus, the messages exchanged between functions in our problem will refer to the assignments of a single binary variable. In other words, the assignment $\mathbf{Z_{f \cap g}}$ will correspond to some binary variable z_{pr}. Therefore, a message between functions must contain two values, one per assignment of a binary variable. At this point, we can make a further simplification and consider sending the difference between the two values. Intuitively, a function sending a message with a single value for a binary variable transmits the difference between the variable being active and inactive. In general, we will define the single-valued message exchanged between two functions as

$$\nu_{f \to g} = \mu_{f \to g}(1) - \mu_{f \to g}(0) \ . \quad (2)$$

Next, we compute the messages between cost and selection functions.

(1) From cost function to selection function. This message expresses the difference for a UAV ρ_p between serving request τ_r or not, therefore

$$\nu_{v_{pr} \to s_r} = v_{pr}(1) - v_{pr}(0) = \delta_{pr} - 0 = \delta_{pr} \ . \tag{3}$$

(2) From selection function to cost function. Consider selection function s_r and cost function v_{pr}. From equation 1, we obtain

$$\mu_{s_r \to v_{pr}}(1) = 0, \quad \text{and} \quad \mu_{s_r \to v_{pr}}(0) = \min_{\rho_{p'} \in P_r - \rho_p} \delta_{p'r} \ .$$

Then we can apply equation 2 to obtain the single-valued message $\nu_{s_r \to v_{pr}} = -\min_{\rho_{p'} \in P_r - \rho_p} \delta_{p'r}$. Moreover, this message can be computed efficiently. Consider the pair $\langle \nu^*, \nu^{**} \rangle$ as the two lowest values received by the selection function s_r. Then, the message that s_r must send to each v_{pr} is

$$\nu_{s_r \to v_{pr}} = \begin{cases} -\nu^* & \nu_{v_{pr} \to s_r} \neq \nu^* \\ -\nu^{**} & \nu_{v_{pr} \to s_r} = \nu^* \end{cases} . \tag{4}$$

To summarize, each cost function computes and sends messages using equation 3; each selection function computes and sends messages using equation 4.

Max-Sum operation. Max-Sum is an approximate algorithm in the general case, but it is provably optimal on trees. Due to how we encoded the problem, the resulting factor graph contains a disconnected, tree-shaped component for each request r (see Figure 3c). Thus, Max-Sum operates optimally in this case. Moreover, the algorithm is guaranteed to converge after traversing the tree from the leaves to the root and then back to the leaves again. In our case, the tree-shaped component for each request is actually a star-like tree, with the selection function s_r at the center, and all others connected to it. Hence, we are guaranteed to compute the optimal solution in two steps if we pick s_r as the root node of each component.

Typically, Max-Sum's decisions are made according to the costs seen by the variable nodes after running the algorithm. However, we have no variables in our graph anymore because we eliminated them. As a consequence, we have to make the decisions at either the selector nodes s_r or at the cost nodes v_{pr}. The best option is to let the selectors choose, because it guarantees that the same task is never simultaneously assigned to two different UAVs. Because the decisions are made by the s_r nodes, there is no need for the second Max-Sum iteration (messages from selector to cost functions) anymore.

Notice that, after our refinements, the only remaining Max-Sum nodes are the cost functions v_{pr} and a selection function s_r for each request. Hence, the UAVs can execute the algorithm distributedly by letting the cost functions v_{pr} run on UAV ρ_p, and the selection functions s_r run on the current owner of the request. Moreover, a UAV only needs to know which requests do its neighbors have to build the logical nodes it must run. Thus, the whole graph can also be built distributedly based only on the UAVs' local information.

Max-Sum runs on our example as follows. First, each leaf cost function v_{pr} must send its cost to the root of its tree, s_r. That is, UAV ρ_1 sends 1 to s_3 (within UAV ρ_2), and 5 to s_2 (within itself). Likewise, UAV ρ_2 sends 2 to s_2 and 2 to s_3, whereas UAV ρ_3 sends 7 to s_1. Thereafter, the s_r nodes decide by choosing the UAV whose message had a lower cost. Hence, s_3 (running within UAV ρ_2) decides to allocate τ_3 to ρ_1, s_2 allocates τ_2 to ρ_2, and s_1 allocates τ_1 to ρ_3. At this point the owner of each request knows whether a request should be reallocated or not and to whom, so we can use it to implement the reallocation phase described in Section 3.2.

5. Coordination using Workload-based Valuations

In realistic scenarios, requests do not appear uniformly across time and space but concentrated around one or several particular areas, hereafter referred to as *hot spots*. In that case, the assumption of independence in the valuation of the requests provides an allocation that assigns a large number of requests to the UAVs close to the hot spot, leaving the remaining UAVs idle. Thus, the independence assumption is too strong because it leads to utterly unbalanced allocations. Next, we show that it is possible to relax this assumption while keeping an acceptable time complexity for Max-Sum. We introduce a new factor for each UAV: a penalty that grows as the number of requests assigned to the UAV increases. Formally, let $Z_p = \{z_{pr}|\tau_r \in R_p\}$ be the set of variables encoding the assignment to UAV ρ_p. The number of requests assigned to UAV p is $\boldsymbol{\eta}_p = \sum_{r \in R_p} \mathbf{z}_{pr}$. The workload factor for UAV ρ_p is

$$w_p(\mathbf{Z}_p) = f(\boldsymbol{\eta}_p) = k \cdot (\boldsymbol{\eta}_p)^\alpha \quad , \tag{5}$$

where $k \geq 0$ and $\alpha \geq 1$ are parameters that can be used to control the fairness in the distribution of requests (in terms of how many requests are assigned to each UAV). Thus, the larger the α and the k, the fairer the request distribution.

The direct assessment of Max-Sum messages going out of the workload factor takes $O(N \cdot 2^{N-1})$ time, where $N = |Z_p|$. Interestingly, the workload factor is a particular case of a *cardinality potential* as defined by Tarlow *et.* al. [27]. A cardinality potential is a factor defined over a set of binary variables (Z_p in this case) that does only depend on the number of active variables. That is, it does not depend on which variables are active, but only on how many of them are active. As described in [27], the computation of the Max-Sum messages for these potentials can be done in $O(N \log N)$. Thus, using Tarlow's result we can reduce the time to assess the messages for the workload factors from exponential in the number of variables to linearithmic.

To further speedup the algorithm and reduce message exchanges, we can add the workload factor and the cost factors that describe the cost for UAV ρ_p to service each of the requests. Lemma 2 in the appendix shows that if we have a procedure for determining the Max-Sum messages going out of a factor over binary variables, say f, we can reuse it to determine the messages going out of

Algorithm 1 process(v_p)   ▷ All undefined values are ∞

1: **for** $r \in R_p$ **do**   ▷ Incorporate incoming messages
2:     $I[r] = \nu_{s_r \to v_p} + v_{pr}(1)$
3: **end for**
4: $Pos, I' = \text{sorted}(I)$ ▷ *Pos* are the reverse indices, and I' the sorted list
5: $cs = 0$
6: **for** $i = 0$ **to** $|I'|$ **do**   ▷ Compute cumulative sums
7:     $CS^0[i] = cs + w_p(i)$
8:     $CS^-[i] = cs + w_p(i-1)$
9:     $CS^+[i] = cs + w_p(i+1)$
10:     $cs = I'[i] + cs$
11: **end for**
12: **for** $i = 0$ **to** $|I'|$ **do**   ▷ Compute cumulative mins
13:     $M^+[i] = min(CS^+[i], M^+[i-1])$
14:     $M^L[i] = min(CS^0[i], M^L[i-1])$
15:     $M^R[i] = min(CS^0[|I'|-i], M^R[|I'|-i+1])$
16:     $M^-[i] = min(CS^-[|I'|-i], M^-[|I'|-i+1])$
17: **end for**
18: **for** $r \in R_p$ **do**   ▷ Compute and send messages
19:     $i = Pos[r]$
20:     $\xi_0 = min(M^L[i-1], M^-[i+1] - I'[i])$
21:     $\xi_1 = min(M^+[i-1], M^R[i+1] - I'[i])$
22:     $\nu_{w_p \to s_r} = \xi_1 - \xi_0 + v_{pr}(1)$
23:     $\text{send}(\nu_{w_p \to s_r})$
24: **end for**

---

a factor h that is the sum of f with a set of independent costs, one for each variable.

Thus, we can define a single factor that expresses the complete cost of a UAV when assigned a set of requests, which amounts to the sum of the independent costs for each of the requests assigned plus the workload cost for accepting that number of requests. Formally the cost factor for UAV ρ_p is

$$w_p(\mathbf{Z}_p) + \sum_{\tau_r \in R_p} v_{pr}(z_{pr}) \quad . \tag{6}$$

Algorithm 1 defines how to assess the messages flowing out of the cost factor v_p. It is obtained by composing Lemma 2 with the algorithm described in [27] for cardinality potentials.[2] The algorithm is basically a dynamic programming procedure for computing minimizations over cumulative sums, which can be done in a few linear passes. The complexity is dominated by the initial sort operation.

Summarizing, by introducing workload valuations that do not only depend on each individual request, but also on the number of requests, we have shown that it is possible to relax the assumption of independence between valuations with a very minor impact on the computational effort required to assess the

---

[2]The algorithm description provided in [27] is inaccurate. However, the authors own implementation of the algorithm is correct. It does not match the description in [27] and is closer to the description provided by Algorithm 1.

messages (from linear to linearithmic). In the next section we show that this relaxation provides significantly better allocations in terms of median service time.

6. Evaluation methodology

In the following sections we want to empirically evaluate our MRF-based solutions for the LORP. The evaluation will be performed using the MAS-Planes [28] simulation environment. MASPlanes allows us to generate scenarios of varying characteristics, and then run any of the implemented algorithms against the exact same problem instances to see how they compare.

Comparing our algorithm's performance against the current state-of-the-art is tricky because, as explained in Section 2, most methods can not cope with the requirements of our problem domain. Therefore, we implemented a relaxed version of the problem to compare against them. In this relaxation, the allocation is performed by a central agent that has no communication restrictions. Furthermore, the centralized allocation procedure used by the centralized agent is considered to be instantaneous. That is, there is no delay at all introduced by the allocation procedure, whatever its computational cost may be. The only constraint that such central planner enforces is that a UAV can not be assigned to a request whose existence is unknown by that UAV. In the following experiments, the central agent employs one of two different request allocation algorithms to make decisions using this relaxed problem.

First, the *c-hungarian* algorithm runs the Hungarian method [15] to compute an allocation. This method optimally solves the assignment problem in $O(n^3)$ time. However, recall that our problem is not an assignment problem because: (i) unassigned requests will have to be performed later on; (ii) the cost of servicing a request changes depending on the order; and (iii) new requests may be introduced at any future time. Also notice that the solutions obtained using this method correspond to those that would be found by the auction algorithm [14]. Aside from the actual procedure, the only difference between these methods is that the auction algorithm may be distributed, at the expense of additional delays caused by communications. Hence, the results that we obtain for this method provide an upper bound on the best results that a distributed auction algorithm would obtain.

Second, *c-greedy* represents a straightforward sequential greedy algorithm as presented in Algorithm 2. Initially, the central agent considers that no request is allocated to any UAV. Then, it computes the best allocation (the allocation with minimum cost) of a single task to a single UAV. Thereafter, it keeps finding the best possible allocation of a single request to an UAV, but taking into account the previously allocated requests. The process ends when all requests have been assigned to some UAV. This procedure can be seen as a centralized version of the Sequential Single Item [16] auctions mechanism. Hence, this solution represents an upper bound of the results we would be able to achieve while using SSI auctions if we were somehow able to use them. Because we want to minimize the average service time, the *c-greedy* algorithm employs the

---
Algorithm 2 GreedyAllocation(P, R, V)
___

Input: P is a set of planes
  R is a set of requests
  V is a map where each $V[p]$ are all requests known by $p \in P$
1: $A[p] = \emptyset, \forall p \in P$
2: **while** $R \neq \emptyset$ **do**
3:   $\langle p^*, r^* \rangle = \arg\min_{p \in P, r \in R | r \in V[p]} evaluate(r, A[p])$
4:   $A[p^*] = \text{insert}(r^*, A[p^*])$
5:   $R = R \setminus \{r^*\}$
6: **end while**
Output: A, a map where each $A[p]$ are all requests allocated to p
___

recommended *BidMinPath* bidding rule from [18] as the *evaluate()* function in Algorithm 2. Finally, notice that this method is also equivalent to the *SGA* method presented in [24]. This implies that, as proven in that work, the decisions obtained by *c-greedy* are equivalent to the solutions that would be obtained by the Consensus-Based Bundle Algorithm (CBBA) in the ideal case of a static network structure. Thus, *c-greedy* also represents an upper bound on the quality of the solutions that CBBA would obtain if we were able to employ it.

For completeness, we also implemented centralized versions of our algorithms: *c-independent*, that represents our MRF-based solution using independent valuations; and *c-workload*, that uses workload valuations as detailed in Section 5. These work exactly like the distributed versions, but within a single central agent and thus without latency penalties when messages are exchanged.

All these centralized algorithms serve as baselines to assess the performance of the distributed solutions. Obviously, the distributed solutions include our proposed solutions *d-independent* (using independent valuations) and *d-workload* (using workload valuations). Additionally, we implemented the distributed state-of-the-art PSI method, and obtained very interesting results: the PSI method and our *d-independent* method provide exactly the same decisions. This can be seen by following the example execution in Section 4.2, and interpreting it as if request owners opened an auction for each of the tasks they have, and neighboring UAVs bid for them (lowest bid wins). Due to this exact functional match between PSI and our solution, we only report results about *c-independent* and *d-independent*, but these represent also the results obtained by the PSI method. At this point, a keen reader may ask whether we could employ workload evaluations (from eq. 6) in the PSI method. However, recall that in PSI all tasks are auctioned in parallel. Thus, when two UAVs meet and one has several more tasks than the other, the UAV with fewer tasks will actually win *all* tasks (at the same time) because it has a much lower workload. In the next reallocation cycle, the entirely opposite happens, and so on. In the end, the result is a thrashing behavior that blocks both planes from performing any useful work.

To Summarize, Table 1 presents the algorithms we test and their relationship with other state-of-the-art approaches. Under the assumptions presented in that table, the algorithm's decisions would match those that we obtain in our

Method	is the ideal implementation of	if disregarding
c-hungarian	Auction algorithm	CD, SA
c-greedy	CBBA	CD, ND, SA
	SSI auctions	CD, ND, SA
c-independent	PSI auctions	CD, SA
	Independent valuations	CD, SA
c-workload	Workload valuations	CD, SA
d-independent	PSI auctions	-
	Independent valuations	-
d-workload	Workload valuations	-

Table 1: Relationship of the evaluated algorithms with other state-of-the-art approaches. CD, DN and SA stand for Communication Delays, Network Dynamicity and Situational Awareness respectively.

evaluation. However, this does not necessary imply that the algorithms can not work without such assumptions. For instance, CBBA can operate on a dynamic network, but would obtain worse results than those of *c-greedy*.

7. Effects of the spatial distribution of requests

In this section we aim to validate the hypothesis that different spatial distributions of the incoming requests may favor some methods over others. Because this assumption is independent of whether the algorithms can or cannot work in a distributed manner, we start by comparing all algorithms using their idealistic (centralized) implementation. Thereafter, we compare the results of *c-independent* and *c-workload* with their actually applicable *d-independent* and *d-workload* distributed counterparts.

7.1. Empirical settings

We prepared two scenarios that represent a time-span (T) of one month. During that time, 10 UAVs with a communication range of $2\,\mathrm{km}$ survey a square field of $100\,\mathrm{km}^2$. We assume that the UAVs always travel at a cruise speed of $50\,\mathrm{km/h}$. In these scenarios, a single operator submits requests at a mean rate of one request per minute. However, we introduce four crisis periods during which the rate of requests is much higher. To accomplish this, the requests submission times are sampled from a mixture of distributions. The mixture contains four normal distributions $\mathcal{N}_i(\mu_i, 7.2\,\mathrm{h})$ (one per crisis period) and a uniform distribution for the non-crisis period. The μ_i means themselves are sampled from a uniform distribution $\mathcal{U}(T)$.

The two scenarios differ on the spatial distribution of requests. In the *uniform* scenario, requests are uniformly distributed along space, whereas the *hot spot* scenario models a more realistic setting where crisis requests are localized around hot spots. These spatial hot spots are defined as bivariate Gaussian distributions with randomly generated parameters. Figure 4 depicts an example of such scenario, where colored dots stand for requests. The scattered dots

correspond to non-related requests, whereas related requests form dot clouds around their hot spot. Finally, the strong dot represents the operator, and the circle surrounding it represents its communication range.

To use the *c-workload* method we have to set values for the k and α parameters. Hence, we performed an exploration on the space of these parameters to determine which values are suitable to the *hot spot* scenarios. Figure 5 shows the results we obtained after this exploration. The colors correspond to the median of the average service time that we obtained after running the algorithm in 30 different scenarios for each pair (k, α). For instance, when $k = 10^2$ and $\alpha = 1.12$ the algorithm achieved a median average service time of 137 s. Observe that the algorithm exhibits a smooth gradient for any fixed value of α or k. Hence, good combinations of k and α can be found by fixing one parameter to a reasonable value and performing a descent search on the other one. We chose $k = 1000$, and found the best corresponding α to be 1.36 with 0.01 precision. We also conducted a similar exploration in the uniform scenario. However, the results were similar for k values between 10^2 and 10^5, and α values between 1.01 and 1.68. Hence, we use the same values than in the hot spot scenario.

7.2. Results

Next we ran all centralized algorithms on a new set of 30 new problems of each scenario to ensure that the parameters were not overfitted. Figure 6a shows the average service time (time passed between when the request is generated and when the planes service it) achieved by the different algorithms in the *uniform* scenario. The best performing algorithm is *c-independent*, with *c-workload* being just 1% worse. Despite the small difference, all results are found significant by the (paired, nonparametric) Wilcoxon signed rank test [29] with $p = 0.01$. The worst algorithm is clearly *c-hungarian*, taking nearly 16% more time on average.

These results contrast with those of the *hot spot* scenario. As shown in Figure 6b, *c-workload* rises as the best method in this case, taking 10% less time than *c-greedy* and 14% less time than *c-independent*. Interestingly, this means that both *c-workload* and *c-greedy* are able to capture and exploit the underlying spatial distribution of requests, whereas *c-independent* works very well when the requests are randomly distributed but is not well equipped to handle other, more realistic distributions. Finally, *c-hungarian*'s results show that completely disregarding the dynamism of the problem leads to overall bad decisions.

At this point, it is clear that the underlying spatial distribution of tasks has different effects on the varying methods. However, we must check whether such differences still hold in the actually applicable distributed versions. Hence, we ran the same scenarios, but this time using *d-independent* and *d-workload* to compare their results with the respective idealized cases. Figure 7 shows the results obtained on both scenarios. Notice that the differences persist, with *d-independent* achieving insignificantly ($p = 0.01$) better results than *d-workload* in the uniform scenario while being significantly worse in the *hot spots* one.

In fact, *d-workload* turns out to be as good as *c-independent* in that scenario according to the Wilcoxon test.

Aside from confirming our hypothesis about the spatial distribution of requests, this experiment also provides an idea about the cost of distributed solving. Specifically, it shows that there's a 20% time increase between the distributed methods and their idealized counterparts in the *uniform* scenario, and between 6% ({c,d}-independent) and 13% ({c,d}-workload) in the *hot spots* one. The lower overhead in the *hot spots* scenario is due to requests being closer between them, giving the planes more opportunities to coordinate.

8. Exploring d-workload's behavior

In the previous Section we presented results on two scenarios that differed only on the spatial distribution of requests. However, it is still unclear how *d-workload* will perform in other scenarios. Hence, in this section we aim at providing a better inside on the behavior of *d-workload* under varying scenario's characteristics.

8.1. Empirical settings

To this end, we chose a number of scenario's characteristics and prepared a full factorial experiment to evaluate their impact on the different algorithms. To capture a wide range of scenarios yet keep the number of experiments within a reasonable number, we selected the following four characteristics:

- **Load.** Defines the overall amount of requests per UAV. To keep things simple, we always introduce a total of $43,200$ requests (1 req/min on average), so the load is controlled solely by varying the number of available UAVs.

- **Spread.** Captures how spatially spread the requests are between them. In scenarios with low spread, the requests tend to be clustered together around hotspots. In high-spread scenarios, the tasks are scattered around a larger area. We control the spread of the scenario by defining a *hostspot-radius*. In the problems of that scenario, approximately 90% of the tasks of each hotspot will be within hotspot-radius distance from the hotspot center. Such distribution is achieved by sampling from an Inverse Wishart distribution with 2.5 degrees of freedom and a scale matrix experimentally adjusted to produce locations within the desired hotspot radius.

- **Communication Range.** Represents the UAVs' (and operator) communication ranges.

- **Time distribution Sharpness.** Specifies how constant is the rate at which requests are introduced. Because we keep a constant number of tasks for all scenarios, sharpness can be defined fully in terms of the number of crises. In high sharpness scenarios, there is a single short and highly active crisis period. In contrast, low sharpness scenarios contain nine different crisis periods of moderate activity, but no highly active period.

Feature	Parameter	(Low, Mid, High)
Load	number of planes	20, 10, 5
Spread	hotspot-radius	1 km, 3 km, 6 km
Communication Range	communication range	1 km, 2 km, 3 km
Time distribution Sharpness	number of crises	9, 3, 1

Table 2: Problem dimensions explored and their values

The chosen values for each feature are detailed in Table 2. For the characteristics not described here we employed the values reported in Section 7. Namely, the scenarios represent a full month of simulated time, with planes that move at a constant speed of 52 km/h on a square area of 100 km^2.

Once again, we must decide the k and α values to use in our *d-workload* method. From the previous results, we know that a good value for α can be found for any reasonable k value. Therefore, we fixed $k = 1000$ and explored on the α parameter. Thus, we generated an example problem from each of the above 81 scenarios, and ran *d-workload* with $k = 1000$ and $\alpha = [1.01, 1.02, .., 2]$ on each of them. Figure 8 shows the mean and median average request service times obtained for each α value. In those graphs we observe a (somewhat rough) gradient, as expected by the previous results. Overall, the best α value is $\alpha = 1.25$, with a median service time of 276.9 s and a mean service time of 183.1 s.

8.2. Results

After choosing the k and α values, we are ready to compare all algorithms in the different scenarios. With this aim, we generated 30 problems of each scenario (for a total of 2430 problems). On each of these problems we evaluate the performance of: (i) *d-independent*, which represents both the state-of-the-art PSI auctions method as well as our MRF-based solution with independent valuations; (ii) *c-greedy*, that came up as the best existing centralized method; (iii) *d-workload*, which is our novel proposed algorithm; and (iv) *c-workload*, which represents a bound on the best performance achievable by our novel *d-workload* mechanism.

Figure 9 shows the main effects plots of each characteristic, with the results of each algorithm represented as a percentage of the average service time with respect to that of *c-greedy*. Albeit sometimes small, the differences between all methods are deemed significant by the Wilcoxon signed-rank test with $p = 0.01$. Each plot shows the effect of the studied characteristic averaging across the levels of all other characteristics. For instance, the *Low* point for *d-workload* in Figure 9a represents the average service time of *d-workload* between all problems with a *Low* load, whatever their spread, communication range and sharpness levels are.

Figure 9a shows that the higher the load, the worse *d-workload* compares to *c-greedy*. However, notice that *c-workload* displays an opposite behavior, becoming better than *c-greedy* when there are more tasks per plane. This may seem counterintuitive at first, but has an easy explanation: recall that we control

the load by varying the number of available planes. Hence, the higher the load, the less planes involved in the experiment. As a consequence, *d-workload*'s performance degrades not because there are more tasks, but because the planes are more spread and have less opportunities to coordinate. Moreover, *d-workload* approaches *c-greedy*'s results when the number of planes increases (reading the graph from right to left), almost catching up to it despite operating in a distributed manner.

Figure 9b shows that, unsurprisingly, *d-workload* achieves comparatively worse results as the spatial distribution of requests becomes sparser. This is due to two reasons. First, because the more spread request are, the more UAVs will fly apart, and the lower chances to coordinate because of the communication range limit. Second, because workload methods exploit the spatial correlation between requests: if requests are more spread, this means that there is a lower spatial correlation between them, and hence the workload heuristic becomes worse. Finally, the previous section showed that *d-independent* works comparatively better when the requests are uniformly distributed. This works at its favor when the spread increases. However, being distributed mechanism it is also negatively affected by the first reason above. As a consequence, the relative performance of *d-independent* varies unpredictably when the spread changes.

In Figure 9c we can observe that, as the communication range increases, the distributed methods' results approach those of the centralized ones. This is because, as the communication range grows, the UAVs have more opportunities to coordinate. There is an exception in the case of *d-independent* and a high communication range. In this case, a larger communication range actually worsens the results, because it causes the planes to stay very spread out, thus dealing much worse especially in the low spread scenarios. In contrast, *d-workload* ends up achieving very close (less than 1% worse) results to those of *c-greedy* in the largest range we tested.

Finally, Figure 9d also shows some interesting behaviors. On the one hand, *d-independent* becomes better as the sharpness increases. This is because UAVs that coordinate with this method tend to evenly split the covered space between them. Hence, the more requests there are at the same time, the better its outcome. In contrast, *c-greedy* and *d-workload* try to be more clever than that, which pays off better when there are less requests than when the system is overloaded. This is even more noticeable with the workload-based methods. Recall that the main strength of such methods is to prepare for future requests. However, high sharpness scenarios reward better plans for the current requests than better predictions about the future.

Given that the only state-of-the-art method that satisfies all our problem's constrains is *d-independent*, these graphs show a very interesting figure: *d-workload* outperforms *d-independent* on all scenarios, lowering the average service time between 36 s (16%) and 18 s (6%). Hence *d-workload* becomes the method of choice for the limited range online routing problem. Moreover, in our experiments *d-workload* closes between 25% and 100% of the gap existing between *d-workload* and the state-of-the-art centralized *c-greedy* algorithm. Finally, these results imply that, in the face of highly dynamic situations, it be-

comes more important to properly capture the distribution of incoming requests (as *d-workload* does) than to compute better plans for the currently known ones (as *c-greedy* does).

9. Conclusions and future work

This paper introduced the limited-range online routing problem, which requires that UAVs coordinate to serve requests submitted by external operators. To tackle this problem, we employed an MRF-based solution instead of the more common market-based approaches. Using a novel encoding of the problem and the max-sum algorithm, we showed that this approach can functionally mimic the operation of a decentralized parallel single-auctions approach. While the re-discovery of a method on top of a different theoretical foundation may seem irrelevant, it is a significant result in our case. Namely, the MRF-based approach provides an easily extensible framework where the PSI method does not. Particularly, in this work we have shown that it is possible to introduce new factors to represent the workload of each UAV while maintaining low computational and communication requirements. Empirical evaluation shows that the improved version achieves up to 16% lower service times than the single-auctions approach. Moreover, the actual performance comes very close to that of employing state-of-the-art centralized SSI auctions. Because of the communication range limit, centralized SSI auctions can not be implemented in the real-world. Therefore, our workload-based mechanism stands as a good choice for robust decentralized coordination with a communication range limit.

In the future, we would like to improve our approach and allocation methods in two ways. Firstly, we would like to allow for planes to notify the operators when their tasks are completed. This could be achieved by introducing new tasks into the system that represent such notifications. Also, some further signaling would be required to avoid losing tasks when a UAV fails (and hence disappears along with those tasks it is carrying). Secondly, we envision better cost functions for the UAVs. A clear improvement in that front would be to better consider the actual paths UAVs will follow while completing tasks [30]. Likewise, it would be desirable to introduce additional factors in the cost function to better control the spread of UAVs throughout space, as well as the need for UAVs to visit operators regularly to allow them to introduce new tasks in the system.

References

[1] D. Kingston, R. W. Beard, R. S. Holt, Decentralized perimeter surveillance using a team of UAVs, IEEE Transactions on Robotics 24 (6) (2008) 1394–1404.

[2] T. H. Cox, C. J. Nagy, M. A. Skoog, I. A. Somers, R. Warner, Overview of the civil UAV assessment document, `http://www.nasa.gov/centers/dryden/pdf/111760main_UAV_Assessment_Report_Overview.pdf` (2005).

[3] N. R. C. Steering Committee for the Decadal Survey of Civil Aeronautics, Decadal Survey of Civil Aeronautics: Foundation for the Future, The National Academies Press, 2006.

[4] R. Zlot, A. Stentz, M. B. Dias, S. Thayer, Multi-robot exploration controlled by a market economy, in: IEEE International Conference on Robotics and Automation., Vol. 3, 2002, pp. 3016–3023.

[5] Y. Weiss, W. T. Freeman, On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs, IEEE Transactions on Information Theory 47 (2) (2001) 736–744.

[6] M. Vinyals, Exploiting the structure of Distributed Constraint Optimization Problems to assess and bound coordinated actions in Multi-Agent Systems, Ph.D. Thesis, 2011.

[7] A. Rogers, A. Farinelli, R. Stranders, N. R. Jennings, Bounded approximate decentralised coordination via the max-sum algorithm, Artificial Intelligence 175 (2) (2011) 730–759.

[8] M. Pujol-Gonzalez, J. Cerquides, P. Meseguer, J. A. Rodríguez-Aguilar, M. Tambe, Engineering the decentralized coordination of uavs with limited communication range, in: Advances in Artificial Intelligence: 15th Conference of the Spanish Association for Artificial Intelligence, CAEPIA 2013, Madrid, Spain, September 17-20, 2013. Proceedings, Springer, 2013, pp. 199–208. `doi:10.1007/978-3-642-40643-0_21`.
URL `http://dx.doi.org/10.1007/978-3-642-40643-0_21`

[9] B. P. Gerkey, M. J. Matarić, A formal analysis and taxonomy of task allocation in multi-robot systems, The International Journal of Robotics Research 23 (9) (2004) 939–954.

[10] M. B. Dias, A. Stentz, A free market architecture for distributed control of a multirobot system, in: 6th International Conference on Intelligent Autonomous Systems (IAS-6), 2000, pp. 115–122.

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

[12] M. B. Dias, R. Zlot, N. Kalra, A. Stentz, Market-based multirobot coordination: A survey and analysis, Proceedings of the IEEE 94 (7) (2006) 1257–1270.

[13] R. K. Dash, P. Vytelingum, A. Rogers, E. David, N. R. Jennings, Market-based task allocation mechanisms for limited-capacity suppliers, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 37 (3) (2007) 391–405. `doi:10.1109/TSMCA.2007.893474`.

[14] D. P. Bertsekas, The auction algorithm: A distributed relaxation method for the assignment problem, Annals of operations research 14 (1) (1988) 105–123.

[15] H. W. Kuhn, The hungarian method for the assignment problem, Naval research logistics quarterly 2 (1-2) (1955) 83–97.

[16] S. Koenig, P. Keskinocak, C. Tovey, Progress on agent coordination with cooperative auctions, in: Proc. AAAI, Vol. 10, 2010, pp. 1713–1717.

[17] S. Koenig, C. Tovey, M. Lagoudakis, V. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, A. Meyerson, S. Jain, The power of sequential single-item auctions for agent coordination, in: Proceedings of the National Conference on Artificial Intelligence, Vol. 21, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006, p. 1625.

[18] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. J. Kleywegt, S. Koenig, C. A. Tovey, A. Meyerson, S. Jain, Auction-based multi-robot routing., in: Robotics: Science and Systems, 2005.

[19] M. Alighanbari, J. P. How, Decentralized task assignment for unmanned aerial vehicles, in: IEEE Conference on Decision and Control, IEEE, 2005, pp. 5668–5673.

[20] Z. Li, Z. Duan, G. Chen, L. Huang, Consensus of multiagent systems and synchronization of complex networks: a unified viewpoint, IEEE Transactions on Circuits and Systems I: Regular Papers 57 (1) (2010) 213–224.

[21] A. R. Mosteo, L. Montano, M. G. Lagoudakis, Multi-robot routing under limited communication range, in: IEEE International Conference on Robotics and Automation, IEEE, 2008, pp. 1531–1536.

[22] W. Ren, R. Beard, Cooperative fire monitoring with multiple UAVs, in: Distributed Consensus in Multi-vehicle Cooperative Control, Communications and Control Engineering, Springer London, 2008, pp. 247–264. `doi:doi:10.1007/978-1-84800-015-5_13`.

[23] W. Zhao, Q. Meng, P. W. H. Chung, A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario, IEEE Transactions on Cybernetics 46 (4) (2016) 902–915. `doi:10.1109/TCYB.2015.2418052`.

[24] H.-L. Choi, L. Brunet, J. P. How, Consensus-based decentralized auctions for robust task allocation, IEEE Transactions on Robotics 25 (4) (2009) 912–926.

[25] J. J. Acevedo, B. C. Arrue, J. M. Diaz-Banez, I. Ventura, I. Maza, A. Ollero, Decentralized strategy to ensure information propagation in area monitoring missions with a team of uavs under limited communications, in: Unmanned Aircraft Systems (ICUAS), 2013 International Conference on, IEEE, 2013, pp. 565–574.

[26] J. Butterfield, O. Jenkins, B. Gerkey, Multi-robot markov random fields, in: Proceedings of the international joint conference on Autonomous Agents and Multiagent Systems, Vol. 3, 2008, pp. 1211–1214.

[27] D. Tarlow, I. E. Givoni, R. S. Zemel, HOP-MAP : Efficient Message Passing with High Order Potentials, in: International Conference on Artificial Intelligence and Statistics, Vol. 9, 2010, pp. 812–819.

[28] M. Pujol-Gonzalez, J. Cerquides, P. Meseguer, MASPlanes: A multi-agent simulation environment to investigate decentralised coordination for teams of UAVs (demonstration), in: The 13th International Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 1695–1696.

[29] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics bulletin 1 (6) (1945) 80–83.

[30] M. Niendorf, P. T. Kabamba, A. R. Girard, Stability of solutions to classes of traveling salesman problems, IEEE Transactions on Cybernetics 46 (4) (2016) 973–985. `doi:10.1109/TCYB.2015.2418737`.

Appendix

Lemma 1. *Let $f(x_1, \dots, x_n)$ be a factor over binary variables $x_1, \dots, x_n$. Let $g(\mathbf{x}_i) = \gamma_i \mathbf{x}_i$ be another factor defined only over variable x_i. Let $h(x_1, \dots, x_n) = f(x_1, \dots, x_n) + g(x_i)$ be the factor obtained by adding factors f and g. Let*

$$\mu_{f \to x_j}(\mathbf{x}_j, \nu_1, \dots, \nu_n) = \min_{\mathbf{X}_{-\mathbf{j}}} \left(f(\mathbf{X}) + \sum_{k \neq j} \nu_k \mathbf{x}_k \right) \tag{.1}$$

and

$$\nu_{f \to x_j}(\nu_1, \dots, \nu_n) =$$
$$= \mu_{f \to x_j}(1, \nu_1, \dots, \nu_n) - \mu_{f \to x_j}(0, \nu_1, \dots, \nu_n) \tag{.2}$$

Then we have that:

$$\nu_{h \to x_j}(\nu_1, \dots, \nu_n) =$$
$$= \begin{cases} \nu_{f \to x_j}(\nu_1, \dots, \nu_i + \gamma_i, \dots, \nu_n) & j \neq i \\ \nu_{f \to x_j}(\nu_1, \dots, \nu_i + \gamma_i, \dots, \nu_n) + \gamma_i & j = i \end{cases} \tag{.3}$$

PROOF. see the suplementary material.

26

In terms of the Max-Sum algorithm, this means that if we can efficiently compute the messages flowing out of f, we can also efficiently compute the messages flowing out of h.

Lemma 2. *Let f be a factor over binary variables $Y = \{y_1, \dots, y_n\}$. Let $g(\mathbf{Y}) = \sum_{i=1}^{n} \gamma_i \mathbf{y}_i$ be another factor defined as the addition of a set of n independent factors, one over each variable y_i. Let $h(\mathbf{Y}) = f(\mathbf{Y}) + g(\mathbf{Y})$ be the factor obtained by adding f and g. Let*

$$\mu_{f \to y_j}(\mathbf{y}_j, \nu_1, \dots, \nu_n) = \min_{\mathbf{Y}_{-j}} \left[f(\mathbf{Y}) + \sum_{k \neq j} \nu_k \mathbf{y}_k \right],$$

and

$$\nu_{f \to y_j}(\nu_1, \dots, \nu_n) =$$
$$= \mu_{f \to y_j}(1, \nu_1, \dots, \nu_n) - \mu_{f \to y_j}(0, \nu_1, \dots, \nu_n).$$

We have that

$$\nu_{h \to y_j}(\nu_1, \dots, \nu_n) = \nu_{f \to y_j}(\nu_1 + \gamma_1, \dots, \nu_n + \gamma_n) + \gamma_j.$$
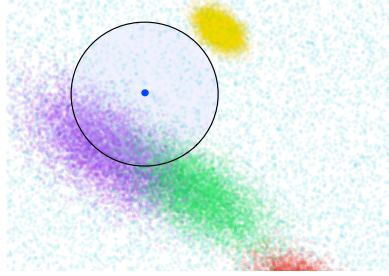
PROOF. see the supplementary material.

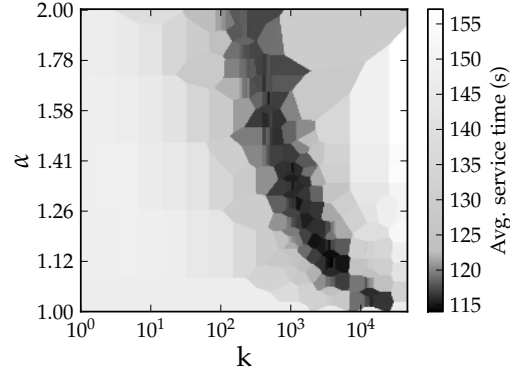Figure 4: Example *hot spots* requests distribution.

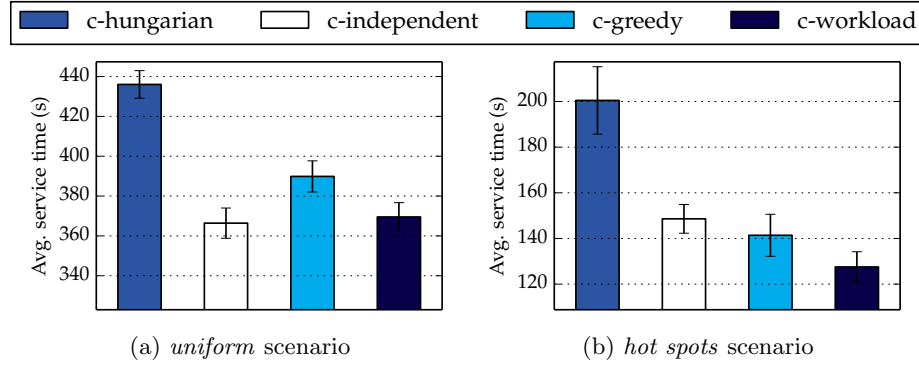Figure 5: Parameter exploration in the *hot spots* scenarios.

(a) *uniform* scenario

(b) *hot spots* scenario

Figure 6: Effects of the spatial distribution of requests. Error bars show the standard error of the mean.

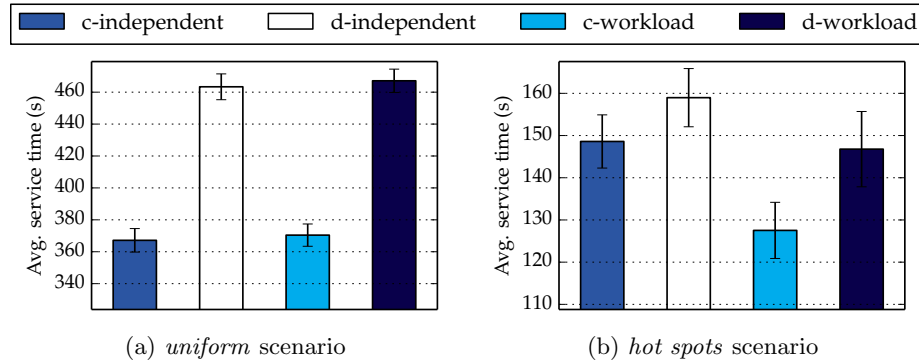(a) *uniform* scenario

(b) *hot spots* scenario

Figure 7: Impact of distributed solving.

28

Figure 8: Exploration of α values.

(a) Load

(b) Spread
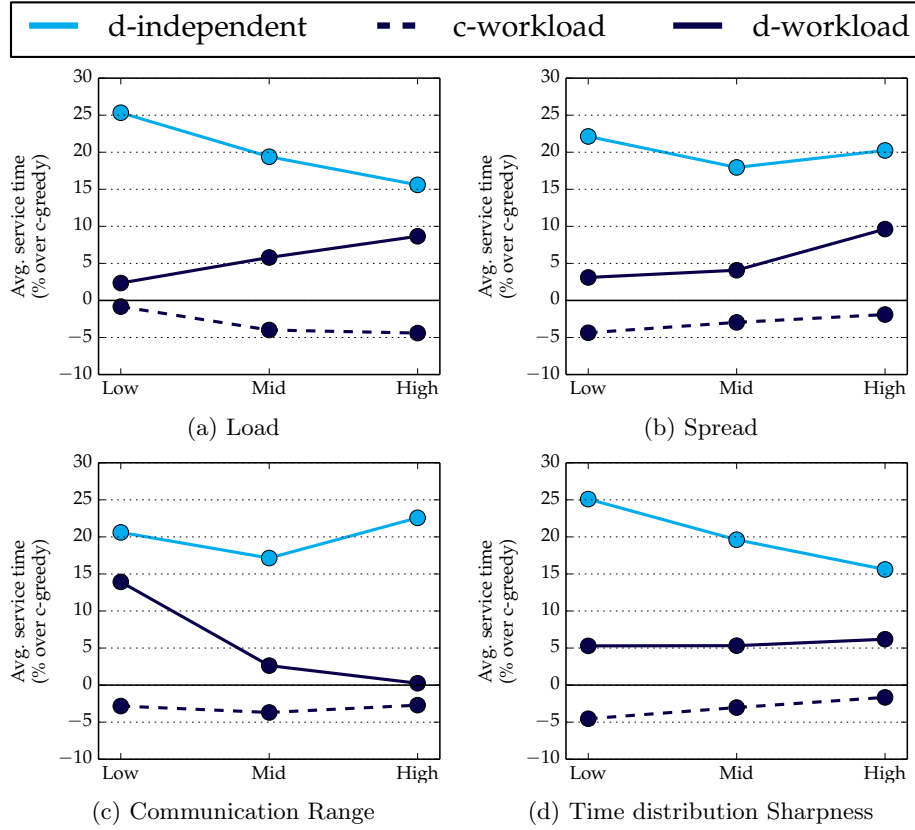
(c) Communication Range

(d) Time distribution Sharpness

Figure 9: Main effects plots (percentage *w.r.t. c-greedy*).