Engineering the decentralized coordination of UAVs with limited communication range

Marc Pujol-Gonzalez[1], Jesus Cerquides[1], Pedro Meseguer[1], Juan A. Rodriguez-Aguilar[1], and Milind Tambe[2]

[1] IIIA-CSIC, Campus de la UAB, Bellaterra, Spain
{mpujol,cerquide,pedro,jar}@iiia.csic.es
[2] University of Southern California, Los Angeles, CA
tambe@usc.edu

Abstract. This paper tackles the problem of allowing a team of UAVs with limited communication range to autonomously coordinate to service requests. We present two MRF-based solutions: one assumes independence between requests; and the other considers also the UAVs' workloads. Empirical evaluation shows that the latter performs almost as well as state-of-the-art centralized techniques in realistic scenarios.

1 Introduction

Unmanned Aerial Vehicles (UAVs) are an attractive technology for large-area surveillance. UAVs are fairly cheap, have many sensing abilities, exhibit a long endurance and can communicate using radios. Several applications can be efficiently tackled with a team of UAVs: power line monitoring, fire detection, and disaster response among others. The autonomous coordination of a UAV team to service a sequence of requests is an open problem receiving increasing attention. In our scenario the requests to be serviced are submitted by a human operator, and the surveillance area is larger than the UAVs' communication range. Most related work comes from robotics, where multi-robot routing [1] is identified as a central problem. But the usual version of multi-robot routing assumes that all robots can directly communicate with each other, which is not our case. Although some works drop this assumption [2, 3], they are focused on exploration of locations, disallowing requests from operators. State-of-the-art research employs auctions to allocate requests to UAVs (robots bid on requests). Auctions are quite intuitive, and in some cases they provide quality guarantees [1]. However, the problem can also be modeled as a Markov Random Field (MRF) [4], or as a Distributed Constraint Optimization Problem (DCOP) [5], for which efficient and easy to distribute algorithms exist. Delle Fave *et.* al. [5] propose an encoding where each UAV directly selects which request it is going to service next. However their model disregards that UAVs can communicate with each other. In this paper we explore coordination solutions for a scenario in which each UAV can communicate with the neighboring UAVs in its range. First, we present an MRF-based solution where the cost of servicing each request is independent of

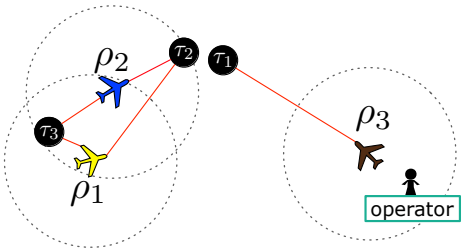

Fig. 1: Firefighting scenario: ρ_1, ρ_2, and ρ_3 are UAVs; dotted cicles around them are their communication ranges; τ_1, τ_2, and τ_3 are targets. A solid line between a target and a UAV means that the UAV is aware of the target.

the remaining 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. Empirical evaluation shows that this is a practical solution for realistic problems.

Motivating Example. In a firefighting context, consider a UAV team continuously monitoring a large natural park. A common approach is to adopt a centralized strategy: UAVs' routes are planned at a central commanding base that guarantees cooperation. However, if the natural park is significantly larger than the UAV communication range, performing centralized planning is unfeasible because the resulting plan can not be effectively transmitted to UAVs.

Figure 1 contains a snapshot of a possible firefighting scenario, with three UAVs, three targets, and a human operator. A good plan would send UAV ρ_1 to target τ_3 and UAV ρ_2 to targets τ_2 and τ_1. However, this plan of action can never be ascertained when assuming limited communication range. The only reasonable strategy to achieve cooperation with limited communication range is to make the UAVs directly coordinate between themselves, in a decentralised manner. Agents themselves must determine their best possible actions at each point in time so that the overall time to service requests is minimised.

Approach. This is a dynamic problem where UAVs move constantly and requests can be introduced at any time. In our approach, the operators send requests to some UAV in their range, which temporarily becomes its owner. Meanwhile, the UAVs use the algorithms detailed below to compute an allocation of all pending requests to some UAV. After each allocation cycle, each UAV becomes the owner of the requests that have been assigned to it, and a new allocation cycle begins. At every time, each UAV that owns some request flies towards the nearest of them, and each idle UAV tries to get in range of the closest operator.

2 Coordination using Independent Valuations

Multi-agent coordination, and particularly task allocation, can be modeled as a MRF [4]. Despite the existance of very powerful algorithms for MRFs, this line of work has received much less attention than auction-based approaches.

2.1 Encoding the Problem as a Binary MRF

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 are indexes for requests and UAVs respectively, $R_p \subseteq R$ be the set of requests that UAV p can service, and $P_r \subseteq P$ be the set of UAVs that can service request 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 n-ary 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 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 2 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 it. This is 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 them. 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). c_2's scope is $\{x_2, x_3\}$, because UAV ρ_2 can service both τ_2 and τ_3. Hence, c_2 has to specify four costs for ρ_2:

1. Both requests are allocated to ρ_1, which is 0.
2. τ_2 is allocated to ρ_1 but τ_3 is allocated to ρ_2, which is $\delta_{23} = 2$.
3. τ_2 is allocated to ρ_2, but τ_3 is allocated to ρ_1, which is $\delta_{22} = 2$.
4. Both requests are allocated to ρ_2, which is $\delta_{22} + \delta_{23} = 4$.

c_1 is similarly computed. From Figure 2 costs, $\mathbf{X}^* = \langle x_1 = \rho_3, x_2 = \rho_2, x_3 = \rho_1 \rangle$.

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

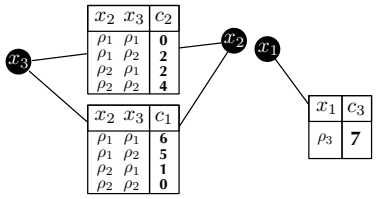

Fig. 2: Naive MRF encoding.

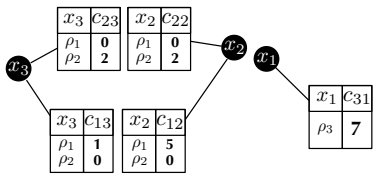

Fig. 3: Independent valuation.

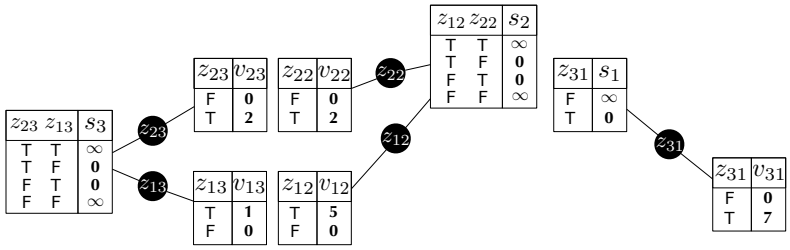

Fig. 4: Independent task valuation, binary encoding.

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 3 represents the example in Figure 2 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 $\{\rho_1, \rho_2, \rho_3\}$. 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 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 4 shows the binary encoding of the example in Figure 3.

2.2 Solving the Problem with Max-sum

Now we optimize the max-sum algorithm to run on the last encoding of Section 2. Max-sum sends messages from factors to variables and from variables to factors.

However, our factor graph allows us some simplifications. Notice that each z_{pr} is only linked to cost function v_{pr} and 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 will consider that functions directly exchange messages.

The max-sum general message expression from function f to function g is

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

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 f is linked to (its neighboring functions), and $\mu_{g'\to f}$ stands for the message from function g' to function f.

Observe in Figure 4 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

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

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:

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

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 4). Thus, Max-sum operates optimally in this case. 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. 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 by the variable nodes after running the algorithm. However, we have no variables in our graph because we eliminated them. Thus, we have to let either the selector nodes s_r or the cost nodes v_{pr} make the decision. Letting selectors choose is better 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. Instead, the selector nodes can directly communicate their decision to the UAVs.

The logical Max-sum nodes include: a cost function v_{pr} for each each UAV ρ_p; and a selection function s_r for each request, that runs in its current owner.

Max-sum runs on our motivating 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. Upon receiving the allocation messages, each UAV knows precisely which requests have been allocated to itself.

3 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, namely the *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. In these scenarios, the independence assumption is too strong. 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 i is $\boldsymbol{\eta}_p = \sum_{r \in R_p} \mathbf{z}_{pr}$. The workload factor for UAV ρ_p is

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

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. [6]. 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 [6], 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.

In addition, we can add the workload factor the cost factors that describe the cost for UAV ρ_p to service each of the requests. The following result[3] 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 a factor h that is the sum of f with a set of independent costs, one for each variable.

Lemma 1. *Let f be a factor over binary variables $Y = \{y_1, \ldots, y_n\}$. Let $g(\mathbf{Y}) = \sum_{i=1}^{n} \gamma_i \cdot \mathbf{y}_i$ 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*

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

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

We have that $$\nu_{h \to y_j}(\nu_1, \ldots, \nu_n) = \nu_{f \to y_j}(\nu_1 + \gamma_1, \ldots, \nu_n + \gamma_n) + \gamma_j.$$

Thus, we can define a single factor that expresses the complete costs of a UAV when assigned a set of requests, that is 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:

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

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 messages (from linear to linearithmic).

[3] Due to lack of space the proof is provided in a technical report [7]

4 Empirical evaluation

Next, we empirically evaluate our decentralized algorithms: (i) *d-independent*, that uses independent valuations on tasks; and (ii) *d-workload*, that employs workload-based request valuations. Comparing their performance against the current state-of-the-art is difficult because most methods can not cope with the communication range limitation of our problem. Thus, we implemented a relaxed version of the problem to compare against them. In this relaxation, UAVs delegate the allocation to a centralized planner agent, disregarding any communication limits. However, no request can be assigned to a UAV that is not aware of its existence. The central agent employs one of two different request allocation algorithms. The *c-independent* algorithm runs a single-item auction per request to allocate it to some plane. Hence, this technique assumes independent valuations for requests. In contrast, the *c-ssi* algorithm employs state-of-the-art Sequential Single Item [8] auctions to compute the allocation of requests to planes. Because we want to minimize the average service time, our SSI auctions employ the *BidMinPath* bidding rule as specified in [1]. Notice that these centralized methods are solving a simplified (less constrained) version of the problem.

We tested the performance of *c-independent*, *d-independent*, *c-workload* and *d-ssi* on multiple problems. Each problem represents a time-span (T) of a month. During that time, 10 UAVs with a communication range of 2 km survey a square field of 100 km^2. We assume that the UAVs always travel at a cruise speed of 50 km/h. In these scenarios, a single operator submits requests at a mean rate of one request per minute. We introduce four crisis periods during which the rate of requests is much higher. The requests submission times are sampled from a mixture of distributions. The mixture contains four normal distributions $\mathcal{N}_i(\mu_i, 7.2\,\text{h})$ (one per crisis period) and a uniform distribution for the non-crisis period. The u_i means themselves are sampled from a uniform distribution $\mathcal{U}(T)$.

Next, we introduce two scenarios that differ on the spatial distribution of requests. In the *uniform* scenario, the requests are uniformly distributed, 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 5 depicts an example of such scenario, where we painted one dot for each request. 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 light circle surrounding it represents its communication range.

To use our *d-workload* method we have to set the values of k and α. Hence, we performed an exploration on the space of these parameters to determine which values are suitable to the *hot spot* scenarios. Figure 6 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

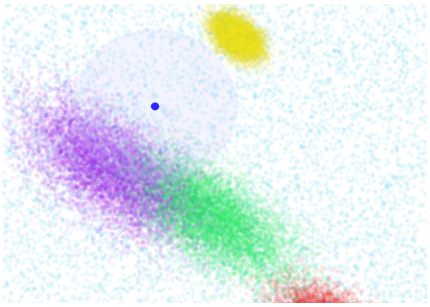

Fig. 5: Example task distribution in a Gaussian scenario

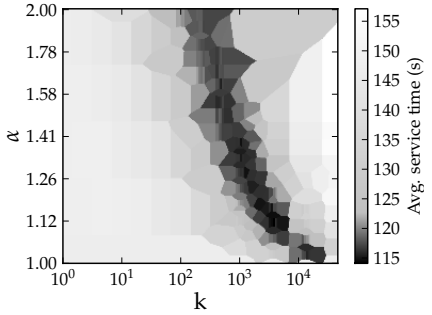

Fig. 6: Parameter exploration in the Gaussian scenario

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.

Then we ran all the algorithms on a set of 30 new problems, to ensure that the parameters were not overfitted. In the *uniform* scenario, *c-independent* clearly obtains the best results. Figure 7 shows the results obtained by the other algorithms relative to *c-independent*'s performance (better algorithms appear lower in the graph). Surprisingly, dropping the independence assumption in these scenarios actually worsens performance instead of improving it. Nonetheless, the performance loss is much lower between *d-independent* and *d-workload* (5%) than between *c-independent* and *c-ssi* (17%).

In contrast, *c-ssi* obtains the best overall results in the *hot spot* scenarios. Figure 8 shows how the other algorithms fared in comparison. Our *d-workload* mechanism obtains very similar results than *c-ssi* (only 2% worse in median). Recall that *c-ssi* requires global communication between the agents, and can not be distributed without introducing major changes to the algorithm. Therefore, *d-workload* stands as the best algorithm when UAVs have limited communication ranges. These results show that, in the more realistic setting where there are request hot spots, relaxing the independence assumption provides significant gains in service time, both in the centralized and distributed algorithms.

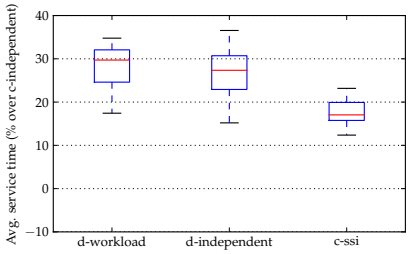

Fig. 7: *uniform* scenario results

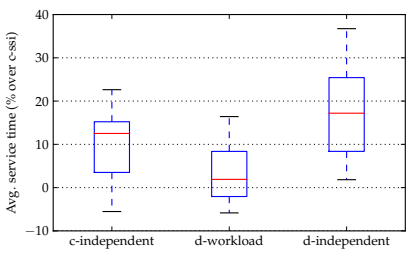

Fig. 8: *hot spot* scenario results

5 Conclusions

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. The MRF-based approach provides an easily extensible framework. In this case, we show 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 11% 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 is the method of choice for decentralized coordination with communication range limit.

Acknowledgments. Work funded by projects RECEDIT (TIN2009-13591-C02-02), AT (CSD2007-0022), COR (TIN2012-38876-C02-01), EVE (TIN2009-14702-C02-01), MECER (201250E053), the Generalitat of Catalunya grant 2009-SGR-1434, and the Spanish Ministry of Economy grant BES-2010-030466.

References

1. Lagoudakis, M., Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, A., Koenig, S., Tovey, C., Meyerson, A., Jain, S.: Auction-based multi-robot routing. In: Robotics: Science and Systems. Volume 5., MIT Press (2005)
2. Sujit, P., Beard, R.: Distributed sequential auctions for multiple uav task allocation. In: American Control Conference, 2007, IEEE (2007) 3955–3960
3. Zlot, R., Stentz, A., Dias, M.B., Thayer, S.: Multi-robot exploration controlled by a market economy. In: Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on. Volume 3., IEEE (2002) 3016–3023
4. Butterfield, J., Jenkins, O., Gerkey, B.: Multi-robot markov random fields. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3, International Foundation for Autonomous Agents and Multiagent Systems (2008) 1211–1214
5. Fave, F.M.D., Farinelli, A., Rogers, A., Jennings, N.: A methodology for deploying the max-sum algorithm and a case study on unmanned aerial vehicles. In: IAAI 2012. (2012) 2275–2280
6. Tarlow, D., Givoni, I.E., Zemel, R.S.: HOP-MAP : Efficient Message Passing with High Order Potentials. In: 13th International Conference on Artificial Intelligence and Statistics (AISTATS). Volume 9. (2010) 812–819
7. Pujol-Gonzalez, M., Cerquides, J., Meseguer, P., Rodriguez-Aguilar, J.A., Tambe, M.: Engineering the decentralized coordination of uavs with limited communication range. Technical report, http://bit.ly/Xuo5yA (2013)
8. Koenig, S., Keskinocak, P., Tovey, C.: Progress on agent coordination with cooperative auctions. In: Proc. AAAI. Volume 10. (2010) 1713–1717