Applied Intelligence (2019) 49:1–15
https://doi.org/10.1007/s10489-018-1169-3

Addressing robustness in time-critical, distributed, task allocation algorithms

Amanda Whitbrook[1] · **Qinggang Meng**[2] · **Paul W. H. Chung**[2]

Published online: 18 April 2018

Abstract

The aim of this work is to produce and test a robustness module (ROB-M) that can be generally applied to distributed, multi-agent task allocation algorithms, as robust versions of these are scarce and not well-documented in the literature. ROB-M is developed using the Performance Impact (PI) algorithm, as this has previously shown good results in deterministic trials. Different candidate versions of the module are thus bolted on to the PI algorithm and tested using two different task allocation problems under simulated uncertain conditions, and results are compared with baseline PI. It is shown that the baseline does not handle uncertainty well; the task-allocation success rate tends to decrease linearly as degree of uncertainty increases. However, when PI is run with one of the candidate robustness modules, the failure rate becomes very low for both problems, even under high simulated uncertainty, and so its architecture is adopted for ROB-M and also applied to MIT's baseline Consensus Based Bundle Algorithm (CBBA) to demonstrate its flexibility. Strong evidence is provided to show that ROB-M can work effectively with CBBA to improve performance under simulated uncertain conditions, as long as the deterministic versions of the problems can be solved with baseline CBBA. Furthermore, the use of ROB-M does not appear to increase mean task completion time in either algorithm, and only 100 Monte Carlo samples are required compared to 10,000 in MIT's robust version of the CBBA algorithm. PI with ROB-M is also tested directly against MIT's robust algorithm and demonstrates clear superiority in terms of mean numbers of solved tasks.

Keywords Heuristic algorithms · Multi-agent systems · Distributed task allocation · Auction-based scheduling · Robust optimization

1 Introduction

The ability to assign tasks well in the light of intrinsic uncertainty is very valuable for multi-agent task allocation systems. However, despite the advantages of distributed systems [1] very few robust algorithms have been developed with this architecture. To date, centralized systems have dominated research focus. This is not surprising since distributed task allocation for multi-agent systems operating in uncertain environments is a challenging problem [2]. One of the main difficulties is that the scheduling system must run independently on each agent but must generate the same schedule in each case. This can be problematic if the agents record different values for the same variable because of sensor differences, limited or changed connectivity, or unreliable, imprecise or vague measurements from mixed sources. These situations typically arise in Search-and-Rescue (SAR) missions where each agent may record a different location for each survivor because of inaccuracies in sensor readings, and none of the locations may be exact. Furthermore, measurements of the agents' positions and velocities may also be uncertain and different for each agent. The uncertainties must be dealt with in some way, as SAR missions are time-critical and demand a low probability of failure; it is vital that the task assignment is reached quickly and that it represents a robust, conflict-free solution, where every survivor is rescued within the given time-frame. The main aim of this work is to address some of these challenges by designing, creating and testing a module that can be attached to general, distributed, multi-agent task allocation algorithms to make them more robust to uncertainty.

✉ Amanda Whitbrook
a.whitbrook@derby.ac.uk

[1] Department of Electronics, Computing and Mathematics, University of Derby, Derby DE22 3AW, England

[2] Department of Computer Science, Loughborough University, Loughborough LE11 3TU, England

To achieve the aim, three different candidate robustness modules (A, B, and C) are trialed with the Performance Impact algorithm (PI) [3] as this has demonstrated better results than CBBA [4] when solving deterministic model SAR problems, but lacks any mechanism for handling uncertainty. Candidate A uses the expected value metric [5] to determine task costs and candidate B uses the worst-case scenario metric. These metrics are selected based on the two extremes of approaches to handling calculated risk. Pursuing choices with the highest expected value is sensible when the payoff appears to be both high and probable (task deadlines are not so tight), but the worst-case option makes more sense when the payoff might be lower and less likely (there are some tight task deadlines). As both situations may be encountered in the type of task allocation problems considered, a third candidate (C) is created, which uses a hybrid combination of the expected value and worst-case scenario metrics, in a bid to match the time cost estimates with the degree of risk, and thus improve overall robustness.

For each candidate, key uncertain variables are modeled using a Gaussian distribution. This common continuous probability distribution is selected because of the central limit theorem, i.e. it is assumed that many different sources simultaneously contribute to the uncertainty in these key variables, see Section 4.2. As the number of sources approaches infinity the distribution function of the measurement of an uncertain variable approaches a Gaussian one, regardless of the underlying distribution functions of the contributing sources of uncertainty. In fact, the resulting distribution function begins to approach Gaussian when there are only between three and five contributing sources [6]. Thus, Gaussians are often used for variables with unknown distributions.

Working in conjunction with PI, the hybrid candidate C consistently demonstrates a very low failure rate and a low number of unallocated tasks in simulated uncertain model SAR problems, and so is adopted as the architecture for the general robustness module, which is named ROB-M. When used with PI, ROB-M does not appear to compromise the mean task time; in the experiments conducted it shows either a significantly faster mean task time when compared to the baseline PI algorithm or no significant difference. It also requires only 100 samples compared with the 10,000 needed in Ponda's robust CBBA model [7], and is tested using higher numbers of tasks and agents. ROB-M can also be bolted on to baseline CBBA, and it is shown that, for uncertain problems, the failure rate and number of unallocated tasks also reduces, as long as the CBBA-ROB-M algorithm is solving a problem that baseline CBBA can solve deterministically; CBBA generally has difficulty when time constraints are tight or the task to vehicle ratio is too high [8]. PI with ROB-M is also compared directly with Ponda's robust CBBA algorithm and demonstrates clear superioriority in terms of mean numbers of solved tasks.

2 Related work

There is an extensive body of work related to multi-agent task planning, task allocation, and scheduling with many solutions proposed. These include clustering graph methods [9], the Contract Net method [10], Markov Random Fields (MRFs) [11], auction-based methods [12], and Distributed Constraint Optimization methods (DCOPs) [13]. In addition, there are many variants on each theme, for example, the stochastic clustering auction with simulated annealing represents a particular type of auction-based solution [1]. Solution methods can also be sub-divided into optimization and heuristic types, online and offline types, and centralized and distributed communication architectures. A good review of the different approaches is presented in [14].

2.1 Summary of approaches

Time-critical, multi-agent, task allocation problems are NP-hard [15] and are thus difficult to solve using optimization approaches such as linear programming (LP), mixed integer linear programming (MILP), MRFs, and DCOPS. A MILP solution to the coordination of mutiple heterogeneous robots has been attempted, but the problem is not time-constrained; the only constraints are that the robots avoid obstacles and do not exceed the speed capabilities [16]. Moreover, only eight robots and six targets are tested and the authors state that scalablity is a major drawback to their method. Pujol-Gonzalez applies an MRF-based solution to UAV online routing using a novel problem encoding and the max-sum algorithm [11], but the problem is also not time-constrained and empirical tests restrict the number of UAVs to ten surveying a limited area of 100 km^2. In general, when the number of tasks and agents increases sufficiently, the optimization approach becomes intractable because of the exponential number of constraints in the model [17].

Heuristic-based methods provide an alternative as scalability is not such a problem. Popular heuristic methods include Tabu-search [18], genetic algorithms [19], and auction-based techniques [20]. In general, heuristic systems are less complex and demonstrate relatively fast execution times, although the trade-off is that they often provide sub-optimal solutions.

Distributed architectures are also a frequent choice for this type of problem as they have many advantages over centralized techniques; the communication overhead is generally lower, the agents do not need to be limited to the communication range of a central server, fewer resources

are required, and the reaction times to local information changes are generally faster [7]. Many existing centralized solution architectures can be adapted to an equivalent distributed paradigm, but distributed solution architectures are slightly more complex to implement and tend to run slower as a consensus processing stage is usually required to ensure that agents have identical information. However, conversion of robust centralized approaches to distibuted equivalents, especially those that require a consensus phase, has proved non-trivial, see Sections 2.3 and 2.4.

2.2 Auction-based approaches

Auction-based heuristic algorithms, a subset of market-based approaches [21] have been widely used as an alternative to optimization for solving time-critical, multi-agent, task allocation problems [22]. In the centralized version of this approach, one agent is selected as the auctioneer and has responsibility for coordinating bids on tasks from all agents so that the highest bidder wins the task assignment. The method is easily adapted to a decentralized architecture, where there is no centralized auctioneer, although this can increase complexity and communication overheads [23]. However, auction-based approaches have many benefits including high efficiency, good scalability [24], and robustness when implemented within a decentralized paradigm.

When the agents bid on bundles rather than individual tasks, the method is known as a combinatorial auction method. Such methods have been shown to exhibit superior performance to single-item auctions and have generated good results when compared to optimal centralized approaches [25]. Two particular combinatorial auction-based algorithms lend themselves to the solution of the problems of interest in this paper - CBBA [4] and the PI algorithm [3]. It has been shown empirically that the baseline PI algorithm performs better than the baseline CBBA algorithm [3, 8, 26] for deterministic task allocation problems with tight task deadlines, with PI demonstrating a much better success rate with different numbers of tasks and agents, and different network topologies. However, the papers mentioned do not examine PI's handling of uncertainty. The lack of a robust solution is a major shortcoming, as time-critical missions (such as SAR) demand a low probability of failure. Reassignment, as discussed in [8], can deal with some of the problems, but it can be hard to recover from a situation where rescue vehicles have already been sent in the wrong direction.

2.3 Incorporating uncertainty into task allocation algorithms

Optimization techniques for solving time-critical, multi-agent task allocation problems are not easily extended to the uncertainty domain, as they require the uncertainty to be represented within the system. Probability models of each uncertain variable could be embedded within the score functions, but this is difficult to achieve within an optimization framework, as many uncertainty distributions cannot be expressed analytically. Thus, Bayesian frameworks [27] cannot be used practically unless the distributions are of very specific types, e.g. Gaussian, Bernoulli etc.[7]. In addition, as mentioned earlier, combining probability distributions increases the dimensionality of the problem space exponentially. Yang uses a distributed robust optimization technique for solving control problems in communication networks [28]. However, the technique is limited to a particular set of optimization problems that have concave objective functions and linear constraints.

To account for uncertainty, many researchers use Monte Carlo sampling techniques to allow the approximation of complex distributions. A popular choice is the Markov Chain Monte Carlo (MCMC) method [29], as complex, high-dimensional systems can be simply and efficiently modeled [7]. However, MCMC methods generally require a large number of samples to represent the uncertainty, and most have been applied to trajectory optimization rather than task allocation optimization. Undurti and How formulate the problem as a Constrained Markov Decision Process (C-MDP) [30]. Their method allows risk, defined as the probability of violating constraints, to be kept below a threshold value while optimizing the reward. Simulation results show that the algorithm performs well, but the experiments are limited to only two agents to enable comparison with centralized MDP approaches. An online MDP method is used in [31], but results are inferior to basic reactive approaches and testing is based on a much simpler problem.

Maheswaran et al. enable users to encode their intuition as guidance for the system [32]. This approach simplifies a scheduling problem by decomposing it into simpler problems that can be solved in a centralized fashion with a single agent allocating the tasks. The work of Ramchurn et al. follows a similar approach, where human decisions are encoded as additional constraints for the optimization [33]. However, in the work presented here attention is restricted to solutions that do not involve human intervention.

Lui and Shell postulate an alternative method that assesses the robustness of any given solution to uncertainty given a measure of it, for example, a probability distribution [34]. They propose an extension to the Kuhn-Munkers Algorithm [35, 36], called the Interval Hungarian Algorithm. This provides a tolerance-to-uncertainty metric for a given allocation. In particular, the authors compute a set of inputs that yield the same output schedule, providing a reliable method for assessing the tolerance of the allocation to uncertainties.

2.4 Robust auction-based approaches

Distributed approaches to task allocation that include a consensus phase (for example auction-based methods) are not easily extended to include uncertainty models as the uncertainty needs to be represented locally in each agent's planning process and convergence must still be guaranteed. However, market-based task allocation systems have been adapted for uncertain environments by updating the cost estimates over time and reallocating the tasks online when necessary [37].

Creation of a solution with good robustness properties, that can hedge against uncertainty from the onset, is an alternative technique to online reallocation, and has the advantage that agents are less likely to become improperly positioned as a result of constant re-routing. Ponda implements this by adding probabilistic models of uncertain variables, Monte Carlo sampling, and stochastic metrics (such as the expected-value and worst-case scenario) to baseline CBBA to improve its robustness [7]. Simulation results showed improved performance over the baseline CBBA algorithm, achieving results similar to centralized approaches. However, the experiments involved a maximum of only six agents and 10,000 samples were required. In addition, beyond about twelve tasks the robust algorithms began to fail. This suggests that further research in this area is needed to address the problems.

3 Methodolody

3.1 Problem definition

The problem of interest is documented fully in [3, 8, 26] and [38]. It is the optimal, conflict-free assignment of a set of n heterogeneous agents $\mathbf{V} = [v_1, \ldots, v_n]^{\mathrm{T}}$ to an ordered sequence of heterogeneous tasks from an m-sized set $\mathbf{T} = [t_1, \ldots, t_m]^{\mathrm{T}}$. Each task k has a fixed location Ω_k, a duration D_k, and a maximum (latest) start time g_k, i.e., the problem is time-critical. Each agent i has a variable location Υ_i and a fixed velocity Θ_i. Each task requires only one agent to service it, and each agent can complete only one task at a time, although it can complete other tasks afterwards, if there is time. The objective function φ is the minimization of mean individual task cost c over all tasks rather than mean completion time for each agent, as the former takes into account the number of tasks that benefit from the time saving. This is important in applications such as SAR, where the tasks involve saving lives. The objective function is expressed mathematically in (1) as:

$$\varphi = \frac{1}{m} \sum_{i=1}^{n} \sum_{k=1}^{\rho_i} c_{i,k}(\mathbf{a}_i), \tag{1}$$

where $\mathbf{a}_i$ represents the set of tasks allocated to agent i and ρ_i represnts the number of tasks in the set $\mathbf{a}_i$ of agent i. In addition, a compatibility matrix $\mathbf{H}$ with entries $h_{i,j}$ defines whether agent i is able to perform task j (the value is 1 if it is able, 0 otherwise).

The constraints in the optimization are as follows:

$$\mathbf{a}_i \leq m, \tag{2}$$

$$\bigcup_{i=1}^{n} \mathbf{a}_i = \mathbf{T}, \tag{3}$$

$$\mathbf{a}_i \cap \mathbf{a}_j = \varnothing \quad \forall i \neq j, \tag{4}$$

$$c_{i,k}(\mathbf{a}_i) \leq g_k, \tag{5}$$

$$h_{i,k} \in [0, 1]. \tag{6}$$

In order, the equations (2) to (6) above denote that the number of tasks assigned to a particular agent must be less than or equal to the total number of tasks, that each ordered sequence of allocations is a subset of the whole set of tasks $\mathbf{T}$ and all tasks must be assigned to some agent, that tasks cannot be assigned to multiple agents, that an agent must complete a task before its latest start time, and that the elements of the compatibility matrix must be either 0 or 1 in value.

3.2 The PI algorithm

The candidate robustness modules are initially tested by integrating them with the PI algorithm [3]. This is a distributed, multi-agent task allocation system that runs simultaneously on each agent. As in CBBA, the tasks are grouped into bundles that are continuously updated as the auction proceeds. In CBBA, the agents bid on the bundles rather than individual tasks and the bundles are formed by logically grouping similar tasks. In contrast, the PI algorithm uses a novel concept called performance impact to score and organize the task bundles. These are incrementally built and updated by systematically swapping tasks between agents, and then measuring the benefit over all tasks using special metrics. The removal performance impact (RPI) measures the benefits of removing a task from a bundle and the inclusion performance impact (IPI) measures the benefits of adding a task. Full details of the metrics and the PI algorithm are presented in [3, 8, 26, 38]. The details are not reproduced here as, for the purposes of this paper, the reader only needs to know that there is a consensus phase, when the agents agree on the schedule, and that the RPI and the IPI are calculated using the time costs $c_{i,k}$ associated with each agent i and task k; the candidate robustness modules are created by using different robust versions of those costs (see Section 3.3).

Furthermore, the aim here is to create a robustness module that can work with any distributed task-allocation algorithm that employs task costs; the module should not be limited to working with the PI algorithm.

3.3 The candidate robustness modules

The computed task costs in task allocation algorithms are frequently underestimated. A simplistic approach to addressing this issue is to introduce risk costs, i.e., a percentage addition to each task cost. However, the practical implementation of this is difficult as the accurate assignment of percentage additions to each task cost is very problem specific [7]. Moreover, the task costs are compound variables, calculated from a number of uncertain variables; nonlinear cost functions, complex coupling and interdependencies between these variables, and constraints can all affect the uncertainty in unpredictable ways.

A much better strategy is to use probabilistic sampling, provided the uncertainty distributions in the key variables are known. Here, we make the assumption that the key uncertain variables are the agent velocities $\mathbf{\Theta}$, agent locations in each of the three dimensions $\mathbf{\Upsilon}_x$, $\mathbf{\Upsilon}_y$, and $\mathbf{\Upsilon}_z$, fixed task locations in each of the three dimensions $\mathbf{\Omega}_x$, $\mathbf{\Omega}_y$, and $\mathbf{\Omega}_z$, and task durations $\mathbf{D}$. In any rescue situation it is obvious that task locations, i.e. the whereabouts of the survivors, have a degree of uncertainty. Task durations are also uncertain as they are only estimates of how long the rescue operation will take at each location. Agent velocities and agent positions can also be unreliable because of instrumentation, interference and communication problems. Section 4.2 provides a fuller discussion of the causes of uncertainty in these variables, and attempts to estimate the errors.

Each uncertain variable is modeled as a Gaussian distribution, given the central limit theorem (see Section 1) and the fact that there are a number of different sources that contribute to the uncertainty in each variable, see Section 4.2. Thus, each candidate robustness module creates robust time cost values $r_{i,k}$ for an agent i and task k by sampling these uncertain variables N times from a Gaussian distribution, defined by a mean equal to the measured value, and a standard deviation determined by the estimated level of uncertainty, see Section 4.2. Simple time-distance-velocity relations are used to calculate the time costs $c_{i,k}$ associated with a particular agent i and task k.

Three candidate modules, i.e. three methods of calculating robust task costs are selected. Candidates A and B are based on the two extremes of approaches to handling calculated risk, see Section 1. The third method, C, uses a hybrid combination of these metrics to try to align the time cost estimates with the tightness of the task deadlines, and thus improve overall robustness.

In Candidate A, the expected values of the actual time costs $c_{i,k}$ are taken as the robust time costs $r_{i,k}$. The expected value of a variable can be defned as the probability-weighted average of all possible values of that variable, i.e. each conceivable value the variable can take is multiplied by its probability of occurring, and these products are summed. The expected value $\mathbf{E}(\zeta)$ of an uncertain variable ζ defined by a discrete set of N samples can thus be expressed mathematically as follows:

$$\mathbf{E}(\zeta) = \sum_{s=1}^{N} \zeta_s ||P_s||. \tag{7}$$

In (7), P_s is the probability of the sample value ζ_s and $||P_s||$ is the normalized value of this given by:

$$||P_s|| = \frac{P_s}{\sum_{l=1}^{N} P_l}. \tag{8}$$

Thus, in Candidate A, the robust time costs are given by:

$$r_{i,k} = \mathbf{E}(c_{i,k}) = \sum_{s=1}^{N} c_{i,k_s} ||P_s||. \tag{9}$$

In (9), the scalar expected value of $c_{i,k}$ is calculated based on the vectors of samples from its component uncertain variables, i.e. the agent velocity $\mathbf{\Theta}$, agent location in each of the three dimensions $\mathbf{\Upsilon}_x$, $\mathbf{\Upsilon}_y$, and $\mathbf{\Upsilon}_z$, fixed task locations in each of the three dimensions $\mathbf{\Omega}_x$, $\mathbf{\Omega}_y$, and $\mathbf{\Omega}_z$, and task durations $\mathbf{D}$. In addition, the probability P_s of sample s is taken as the combined probability of these component uncertain variables. Calculating the expected value of $c_{i,k}$ using the set of sample vectors for each component variable is equivalent to minimizing the mean expected time cost over all tasks. Computing the time costs deterministically using the pre-calculated mean values of the component variables is not equivalent, as it does not take the coupling between the variables into account and this can lead to a poor solution [7]. The former method calculates a time cost c_{i,k_s} $s = 1 \ldots N$ for each of the N samples, multiplies it with its probability, and sums to generate the scalar expected value. This provides a much more robust estimate of the time costs as the coupling between variables is reflected in each sample. To illustrate the difference, the robust calculation of each c_{i,k_s}, the time cost for each sample, can be expressed mathematically as:

$$c_{i,k_s} = \mathbf{f}(\Upsilon_{x,i_s}, \Upsilon_{y,i_s}, \Upsilon_{z,i_s}, \Omega_{x,k_s}, \Omega_{y,k_s}, \Omega_{z,k_s}, D_{k_s}, \Theta_{i_s}). \tag{10}$$

In contrast, the deterministic case, which deals with the mean $c_{i,k}$ would be calculated using:

$$\bar{c}_{i,k} = \mathbf{f}(\bar{\Upsilon}_{x,i}, \bar{\Upsilon}_{y,i}, \bar{\Upsilon}_{z,i}, \bar{\Omega}_{x,k}, \bar{\Omega}_{y,k}, \bar{\Omega}_{z,k}, \bar{D}_k, \bar{\Theta}_i). \tag{11}$$

Candidate B makes use of the worst-case scenario metric to calculate the robust time costs so that:

$$r_{i,k} = \max_{s=1}^{N} c_{i,k}. \tag{12}$$

The maximum (worst case estimate) time cost occurring in the sample is taken, which is a very conservative strategy. For the problems considered here, the method never underestimates the time taken to complete a task. However, using an overestimate for every single task could be dubitable, because each task has its own latest start time; an agent may think, falsely, that a task is unreachable within this time frame.

As Candidate B uses a very pessimistic and risk averse strategy that could be potentially problematic when used with the task-allocation scenarios described here, an alternative method is proposed. Candidate C uses a hybrid technique that attempts to match the time cost estimates to the maximum start times, and thus improve overall robustness. The method places a buffer value ψ on the difference between the expected time cost and the maximum (latest) start time of the task g_k. If

$$g_k - \mathbf{E}(c_{i,k}) < \psi \tag{13}$$

is true then the worst case time cost (12) is used for $r_{i,k}$; the deadline for the task is tight so it pays to be pessimistic. In other words, candidate C is simply more cautious about accounting for uncertainty in its estimate. However, if (13) is false then the deadline is more flexible and the expected time cost can be used for $r_{i,k}$ as in (9).

The candidate modules are not task allocation algorithms in their own right. They are simply a means for estimating robust task costs in a given task allocation problem. Thus, they could be used in any algorithm that deals with calculating task costs. However, in the tests described in Section 4, that aim to establish the best candidate, they are used with the PI algorithm only, and each candidate is integrated into PI using the equations described above to calculate robust time costs $r_{i,k}$, instead of measured ones $c_{i,k}$, for working out the RPI and IPI metrics. Effectively, the same essential objective function is used in the resulting PI-candidate algorithm, but the robust time costs are used instead of the measured ones, i.e.

$$\varphi = \frac{1}{m} \sum_{i=1}^{n} \sum_{k=1}^{\rho_i} r_{i,k}(\mathbf{a}_i). \tag{14}$$

As stated above, this is equivalent to minimizing the mean expected time cost over all tasks in the case of Candidate A.

Apart from sampling and calculating the robust time costs in order to determine the RPI and IPI, the procedural details for the resulting PI-candidate algorithms are almost exactly the same as baseline PI, see [3, 8, 26] and [38]. However, there is another important difference between the baseline and the PI-candidate architecture; when using a candidate module, each agent i communicates its own estimate of the relevant, local uncertain parameters to the others (these include estimates of its own velocity Θ_i and location $\Upsilon_{x,i}$, $\Upsilon_{y,i}$, and $\Upsilon_{z,i}$, and its estimates of all the task locations $\mathbf{\Omega}_{x,i}$, $\mathbf{\Omega}_{y,i}$, and $\mathbf{\Omega}_{z,i}$, and durations $\mathbf{D}_i$). During the consensus phase of the PI algorithm, see [3, 8, 26] and [38], where the agents reach agreement upon the task allocations, each agent is thus aware of the other agents' estimates, and can use these values when calculating the other agents' robust time costs (and hence the resulting IPI and RPI values). These metrics, along with those of the agent itself, ultimately determine the agreed schedules. This method ensures that consensus takes account of each agent's information; it is not biased toward the data from any particular one.

4 Experimental design

Each of the candidate robustness modules is tested using the scenario described in Section 4.1 and its performance is measured using the metrics discussed in Section 4.3. Baseline PI is included to compare performance with and without the use of the candidate modules.

4.1 Scenario

Testing is based on a scenario that focuses upon on the rescue aspect of a SAR mission. It is fully described in [3, 8, 26], and [38] and it represents a particular application of the problem described in Section 3.1. There are a number of heterogeneuos agents (rescue vehicles) that must complete tasks, and a number of tasks (disaster survivors needing to be rescued, i.e. requiring either food or medicine to be despatched to them) that need servicing by the agents. The agents in the scenario are UAVs (carrying food) and helicopters (carrying medicine), and their mission is to rescue the disaster survivors by delivering the supplies; despatch of the food or medicine constitutes the completion of a task.

The start locations of the UAVs and helicopters $\mathbf{\Upsilon}_x$, $\mathbf{\Upsilon}_y$, and $\mathbf{\Upsilon}_z$ are known in advance, as are the 3-dimensional locations $\mathbf{\Omega}_x$, $\mathbf{\Omega}_y$, and $\mathbf{\Omega}_z$, and requirements of the survivors. The vehicles travel at a constant velocity $\mathbf{\Theta}$, which is 50 m/s for the UAVs, and 30 m/s for the helicopters. In addition, the task durations $\mathbf{D}$ are fixed at 350 s and 300 s for food and medicine respectively. As specified in Section 3.1, a latest start time g is specified for a particular task. This models the fact that the supplies must be delivered

before the condition of a survivor begins to deteriorate to such an extent that they are at risk of death.

Initially, to establish the best candidate module, two sets of experiments based on the above scenario are conducted (set 1 and set 2). In experiment set 1, for consistency, the particular parameters used in [3] and [26] are adopted, hence the world x and y coordinates range from -5000 to 5000 m, the z coordinates range from 0 to 1000 m, the mission time limit is set at 2000 s, the earliest start time for each task is 0 s and the latest start time g is generated using a random fraction of 2000 s, which means that the constraints upon the task times can be very tight in some cases. A 16-vehicle fleet is considered comprising 8 UAVs and 8 helicopters and there are 32 tasks, i.e. 16 survivors needing food and 16 needing medicine ($n = 16, m = 32$). Experiment set 1 thus has tight constraints but a low task to vehicle ratio of 2.

The above scenario has been used to test task allocation algorithms in earlier work but it is not representative of a realistic SAR mission as the task-to-vehicle ratio is too low. In reality, there would be fewer vehicles searching for more survivors. In fact, it is the problem structure that tends to limit the task-to-vehicle ratio, in particular, the stringent requirement that the mission is completed in 2000 s or under (about half an hour). The search area is 100 km^3, but a helicopter traveling at 30 m/s can cover only 60 km in the alloted time. Furthermore, each task k has its own critical time limit for commencement g_k, which varies between 0 s and 2000 s; this places further constraints on the solution and is unrealistic as it includes the possibility of a task that must be started as soon as the mission begins. For these reasons, a further set of experiments (set 2) is conducted in which the problem parameters are adapted to suit more realistic task-to-vehicle ratios and time constraints. In experiment set 2, the search area is reduced to 25 km^3 (the z coordinate range remains the same, but the x and y coordinate ranges are changed to ± 2500 m). The mission completion time is also extended to 5000 s, with individual task time limits g_k ranging from 1500 s to 5000 s. This allows for higher task-to-vehicle ratios, and a ratio of 8 is selected, with forty tasks, five vehicles ($n = 5, m = 40$). Experiment set 2 thus has more relaxed constraints but a higher task to vehicle ratio than experiment set 1.

It is important to note that both of the problems (experiment sets) represent artificial situations. However, they have been chosen to cover conditions that will test the candidates, i.e. they are not problems that are easily dealt with (they have a mixture of tight constraints and high task to vehicle ratios), but they have also demonstrated that they are solvable with baseline PI in their deterministic form, i.e they are not so difficult that no solutions will be found. In addition, both experiment sets consider higher numbers of tasks than the failure limit of 12 in [7], and set 1 uses more vehicles than in [7].

4.2 Uncertainty models

Three levels of uncertainty (low, medium and high) are considered, which vary according to prescribed errors in the key uncertain variables - task location $\mathbf{\Omega}_x$, $\mathbf{\Omega}_y$, and $\mathbf{\Omega}_z$, vehicle location $\mathbf{\Upsilon}_x$, $\mathbf{\Upsilon}_y$, and $\mathbf{\Upsilon}_z$, vehicle velocity $\mathbf{\Theta}$ and task duration $\mathbf{D}$. The levels are chosen so that the low uncertainty case models the lower error bounds that one might expect for these variables, i.e. a situation where there is reasonably high confidence in the instrumentation and the data provided. By the same token, the high uncertainty case models the upper error bounds in the uncertain variables - the potential extremes in their variance. In this situation there is low confidence in the readings and intelligence.

Thus, the uncertain variables are modeled as Gaussian distributions centered on a known mean (the measured values) with standard deviation equal to the estimated error ε appropriate for each uncertainty level. For each variable (apart from vehicle velocity, which is modeled with the same error for each level), the bell curves have the same mean but the low uncertainty curve is narrower than the medium uncertainty curve, and the medium uncertainty curve is narrower than the high uncertainty curve. Relatively large percentage errors (50% for the low level and 200% for the high level) are modeled for each dimension of the task location, as information relies upon intelligence from mixed external sources, for example word of mouth estimates of position, inaccurate GPS readings or grid references estimated using maps and compasses. Most UAVs carry satellite positioning (GNSS) receivers, which, for military purposes, should conform to sub-meter accuracy. However, a fixed, higher error of 15 m in each dimension (for all three uncertainty cases) is modeled for the coordinates of the vehicles as GNSS signals can be weak and are vulnerable to interference from many sources such as radio signals and power systems. UAVs generally use airspeed indicators to measure their velocity [39], but these can demonstrate instrument errors of up to about 7 m/s [40]. As the vehicle speeds are 30 m/s and 50 m/s, an upper limit of about 20% seems sensible. Task durations are the most uncertain parameters since many sources contribute to them, including earlier delays servicing other tasks, and innacurate estimates for other variables, such as vehicle velocity. For this reason, relatively large percentage errors (10% for the low level and 50% for the high case) are modeled, but the uncertain values are not allowed to fall below their real values by more than 50 s as it is assumed that there is a minimum duration for each task. The errors ε for each uncertain variable and uncertainty level are summarized in Table 1. Monte Carlo sampling is used to allocate a value from the distribution to each uncertain variable and, as discussed in Section 3.3, this is carried out separately for each vehicle.

Table 1 Uncertainty levels used in the experiments

Uncertainty level	Task Location Error (m) ε_Ω	Vehicle Location Error (m) ε_Υ	Vehicle Velocity Error (%) ε_Θ	Task Duration Error (%) ε_D
Low	50	15	5	10
Medium	100	15	10	25
High	200	15	20	50

4.3 Parameter settings and metrics

Preliminary trials with parameter ψ set between 5 s and 40 s confirmed that the best value to use for this in candidate C is about 20 s, although the algorithm does not appear to be very sensitive to its change. The size N of the samples is maintained at 100 throughout all the experiments as pretrials showed that increasing the sample size to 500 did not improve the results much, but increased computation time. For each scenario, the algorithm is run 100 times to obtain a percentage failure rate. Other numbers were trialed between 50 and 500, but not much variation in the results was enountered, and using more than 100 runs would have increased computation time. Indeed, there will always be variation in performance based on the actual scenario, but examining the results over 100 runs provides a good indication of the failure rate and allows t-tests to be carried out on the objective function values.

Success is measured by selecting random real values for the uncertain parameters from the known probability distributions and calculating the actual number of tasks allocated to vehicles using the robust solution. If any tasks are unassigned then the run is counted as a failure; a run is only successful when all tasks are allocated. The percentage of unassigned tasks across all runs is also of interest as a run might fail, for example, because all the tasks are unassigned or because only one is unassigned. In addition, a solution where the tasks are completed faster i.e., one in which the make-span is minimized, is preferable. For this reason, the mean objective function value $\tilde{\varphi}$ across all successful scenarios is also recorded for each algorithm to provide further comparison.

4.4 Network topology

Both sets of experiments are conducted using a randomly generated mesh network topology. This is similar to a circular topology where vehicles are connected in a ring and each communicates with the next and previous vehicle only, except that in a mesh other communication pairs may also be connected. In these experiments, only half of the other connection pairs are set as communicable, which means that some vehicles cannot communicate directly with each other. The resulting topology thus represents a realistic communication structure as it is not fully connected and is not as simplistic as a row or circular structure.

5 Results

Figures 1 and 2 show the results for the first experiment set across all 100 runs and for each algorithm and each uncertainty case. Figure 1 displays the percentages of unassigned tasks (the maximum number of unassigned tasks possible in this case was 3200) and Fig. 2 shows the percentages of failed runs, i.e. runs where the algorithm was unable to allocate all of the tasks. Figures 3 and 4 repeat this information for experiment set 2 (the maximum number of unassigned tasks possible in set 2 was 4000). Table 2 highlights the mean objective function values for each algorithm and experiment set.

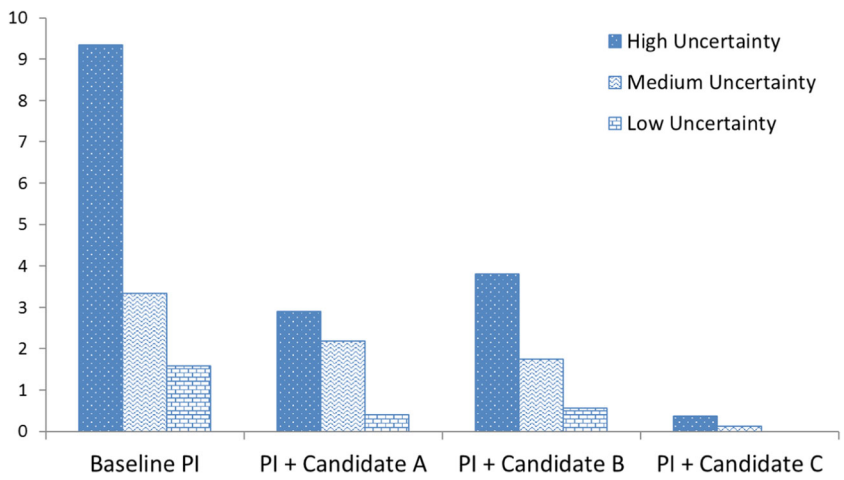


Fig. 1 Percentage of unallocated tasks for each algorithm and each uncertainty case in experiment set 1

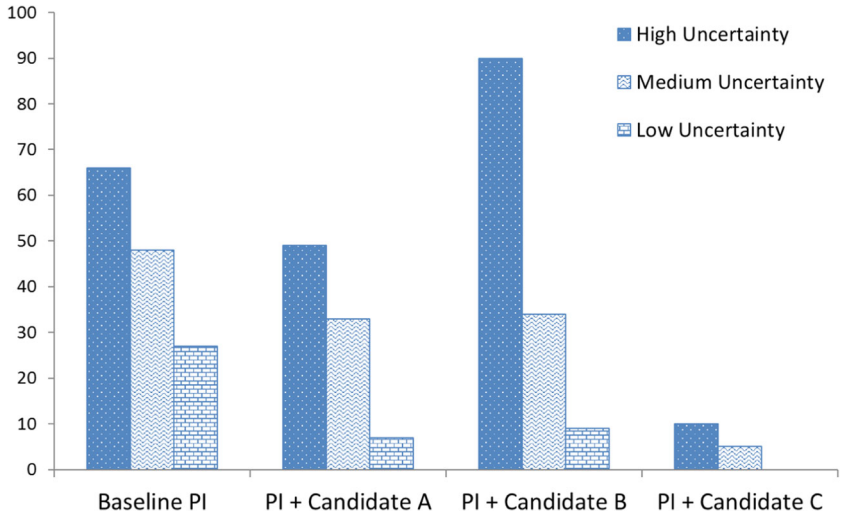


Fig. 2 Percentage of failed runs for each algorithm and each uncertainty case in experiment set 1

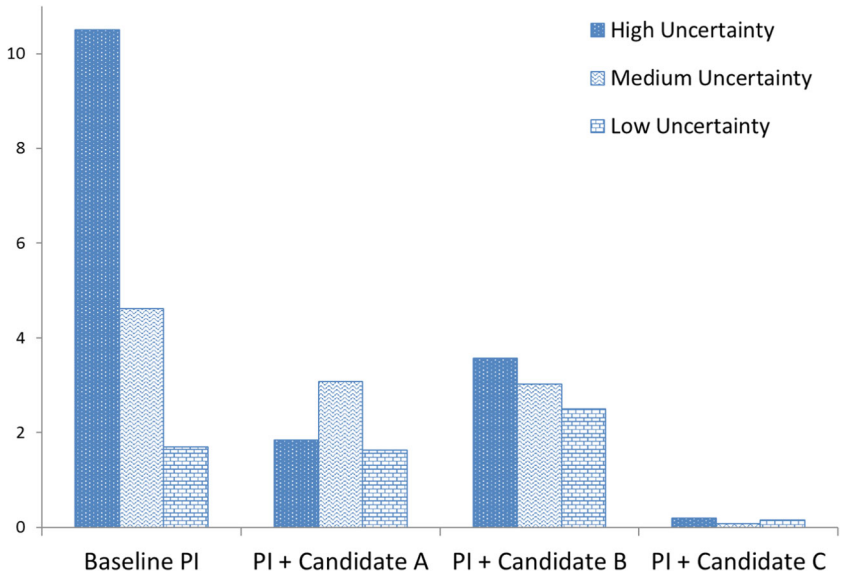


Fig. 3 Percentage of unallocated tasks for each algorithm and each uncertainty case in experiment set 2

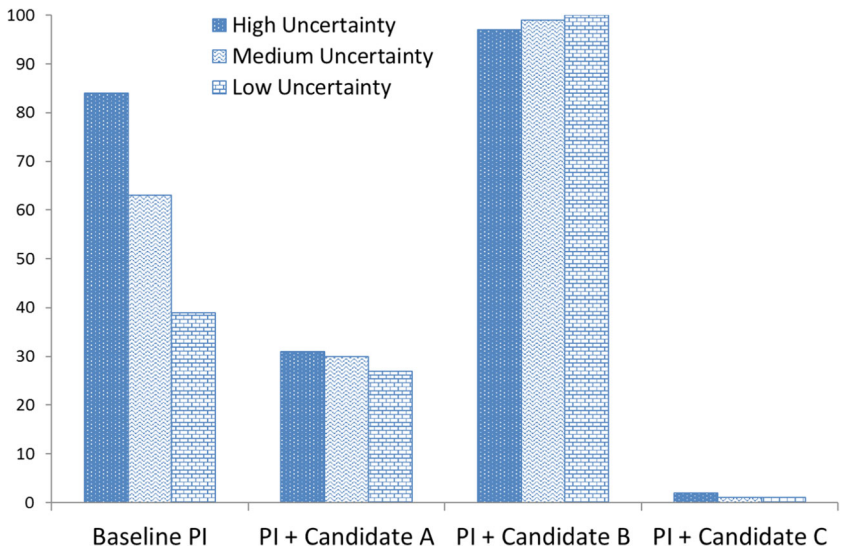


Fig. 4 Percentage of failed runs for each algorithm and each uncertainty case in experiment set 2

Table 2 Mean objective function values $\tilde{\varphi}$ across successful runs

Algorithm	Experiment Set 1			Experiment Set 2		
	High	Med	Low	High	Med	Low
Baseline PI	282	264	261	1378	1340	1321
PI + candidate A	276	262	257	1354	1338	1333
PI + candidate B	268	263	259	1342	1348	–
PI + candidate C	262	267	262	1340	1322	1316

The results for baseline PI show that failure to allocate tasks is a serious drawback when uncertainty is modeled, and the problem scales linearly with the level of uncertainty. For high uncertainty the failure rate is 66% for experiment set 1 and 84% for experiment set 2, with the figures reducing to 27% and 39% respectively for the low uncertainty case. The total number of unallocated tasks across all 100 runs was 299 (about 9%) for experiment set 1 and 420 (about 11%) for experiment set 2 when uncertainty was high. These figures reduced to 51 (about 2%) and 68 (about 2%) respectively for the low uncertainty case. These failure rates and unallocated task figures are unacceptable for SAR missions, as all survivors need to be rescued.

When candidate robustness module A is used with PI there is some improvment upon the baseline performance, especially in the case of experiment set 2, but the algorithm still fell short of producing an acceptable failure rate or a low enough number of unassigned tasks. For example, for high uncertainty its failure rates were 49% and 31% respectively, and the total number of unallocated tasks were 93 (about 3%) and 74 (about 2%) respectively.

The performance of PI with candidate robustness module B was worse than the baseline in terms of the failure rate for high uncertainty in both experiment sets, and for all levels of uncertainty in experiment set 2 where it demonstrated a near 100% failure rate. In terms of total number of unassigned tasks its performance bettered baseline PI in most cases, but it was still not as good as module A when uncertainty was high. It proved 100% capable of predicting its performance but, unfortunately, it was just predicting its own failure in most cases.

Candidate module C, which combines the expected-value metric and the worst-case metric, showed a low failure rate and low total number of unassigned tasks for both experiment sets. For the high uncertainty case the failure rate was reduced to 10% and 2% for sets 1 and 2 respectively, and decreased further to 0% and 1% when the uncertainty level was low. The number of unassigned tasks was also very low in all the uncertainty cases and across both experiment sets compared to baseline performance. The reduction in failures can be attributed to the more 'cautious' design of the algorithm. When the time margin for task completion is tight it acts pessimistically, selecting the worst-case metric to calculate the robust task costs. However, when there is plenty of slack in the time margin, the expected value is used. In addition, t-test results (at the 99% level) comparing the mean objective function for baseline PI and PI with module C revealed that the latter has a significantly faster mean task completion time in experiment set 2 for all of the uncertainty cases. This was also true in experiment set 1 for the high uncertainty case. For the medium and low uncertainty cases in this experiment set, there was no significant difference between the two results. This strongly suggests that module C enables PI to produce reliable solutions under uncertain conditions without compromising the solution quality, and may even improve the solution quality in some cases. For these reasons, module C is adopted as the architecture for the robustness module, which is now referred to as ROB-M.

There is a trade-off between a fast run-time and a robust solution; the modules that use expected time-costs (Candidate A and ROB-M) take about 30 times (experiment set 1) and 50 times (experiment set 2) longer to run than the baseline. In the baseline, the IPI-calculation dominates, taking up about 85% of execution time. Examination of the individual run times for each part of these composite algorithms shows that about 78% of

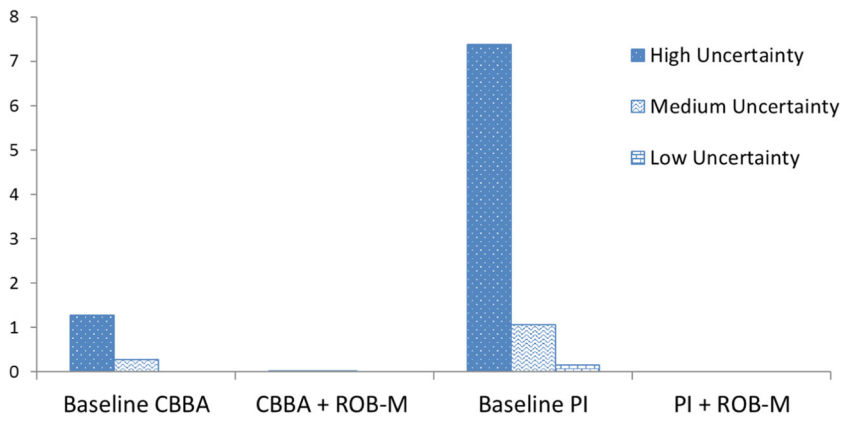


Fig. 5 Percentage of unallocated tasks for each algorithm and each uncertainty case in experiment set 3

run-time is devoted to the MATLAB statistical functions associated with determining the expected time costs and the probabilities of the uncertain variables. This is the key factor underlying the longer run-times. However, the IPI calculation still dominates the remaining routines, taking around 20% of total run-time, with actual time spent in the IPI routine longer than for equivalent problems solved with the baseline. Between 7 and 12 seconds are added to run-time, which may be attributed to the increase in complexity associated with computing the expected time costs and probabilities of the uncertain variables.

6 Intergrating ROB-M with CBBA

It has been shown that ROB-M works well with PI in both of the experiment sets. However, neither of the model problems is suited to CBBA as the problem in set 1 has very tight constraints and the task to vehicle ratio for the problem in set 2 is too high. Baseline CBBA fails to solve both of these problems, even in their deterministic form, and when 100 uncertain runs of the problems are attempted, baseline CBBA demostrates a 100% failure rate. Bolting ROB-M onto CBBA and solving the uncertain problems does not help as the underlying CBBA task allocation mechansim appears to be flawed for these problem types; the uncertainty is a secondary problem. However, one of the aims of the work here is to show that ROB-M can work with another distributed task allocation algorithm. Thus, a problem that baseline CBBA can solve deterministically is selected for experiment set 3. This problem has the same time constraint parameters as experiment set 2, so it is a more relaxed problem. In addition, 10 vehicles and 50 tasks are selected to produce a more reasonable task to vehicle ratio of 5. This experiment set also uses more vehicles than in [7], and many more tasks than the failure limit of 12 in [7]. The new experiment set is trialed with baseline CBBA and CBBA with ROB-M. It is also solved using baseline PI and PI with ROB-M for comparison.

Table 3 Mean objective function values $\tilde{\varphi}$ across successful runs

Algorithm	Experiment Set 3		
	High	Med	Low
Baseline CBBA	743	738	736
CBBA + ROB-M	747	738	736
Baseline PI	784	759	739
PI + ROB-M	752	740	735

6.1 Experimental design

As in the previous experiment sets, 100 uncertain runs are made, this time using baseline CBBA, baseline PI, CBBA with ROB-M, and PI with ROB-M. The percentage of failures, the percentage of unallocated tasks and the mean objective function values ($\tilde{\varphi}$) for successful runs are recorded, as before.

6.2 Results and discussion

Figures 5 and 6 show the percentage of unassigned tasks (the maximum number of unassigned tasks possible was 5000), and percentage failures respectively (across all 100 runs and for each algorithm and each uncertainty case) for experiment set 3. Table 3 highlights the mean objective function values for each algorithm.

The first observation is that baseline CBBA appears to be more robust to uncertainty than baseline PI in this problem. In the low uncertainty case baseline CBBA did not fail in any of the 100 runs. This compares to an 8% failure rate

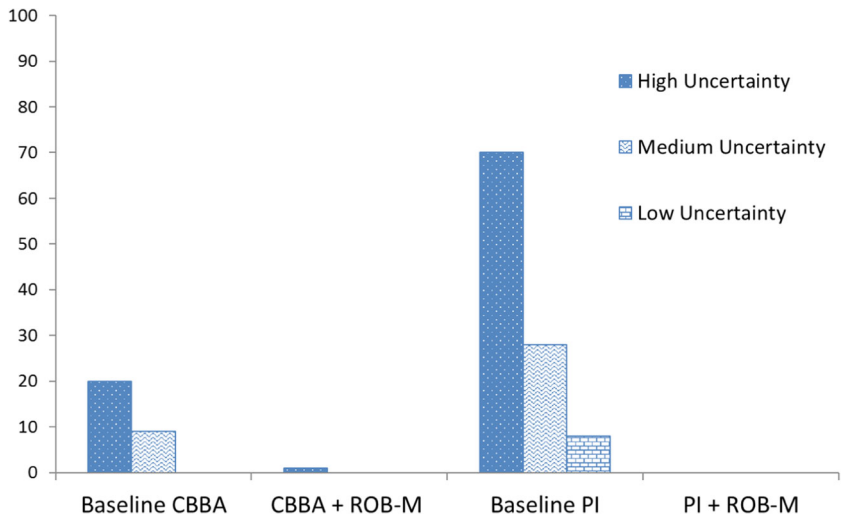


Fig. 6 Percentage of failed runs for each algorithm and each uncertainty case in experiment set 3

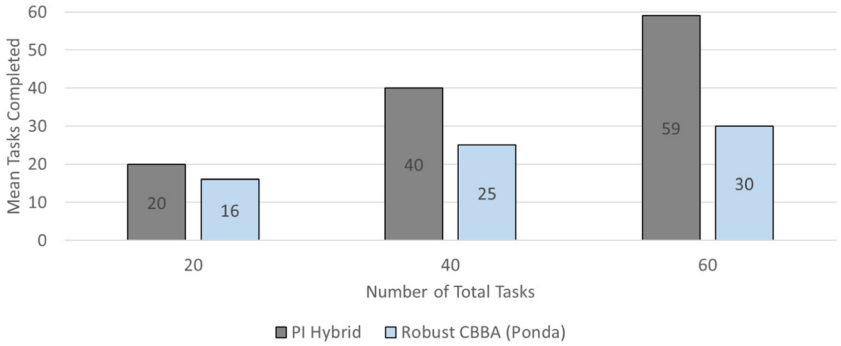


Fig. 7 Comparison with Ponda's results. Mean number of solved tasks versus total number of tasks

for PI, which demonstrated 8 unallocated tasks in total (less than 0.2%). When uncertainty was high, baseline CBBA failed in 20% of the runs (64 total unallocated tasks, about 1%) in contrast to PI, which failed in 70% of the runs (369 total unallocated tasks, about 7%). When ROB-M was bolted on, the failure rate for the high uncertainty case was reduced to 1% (1 total unallocated task) for CBBA and 0% (no unallocated tasks) for PI. In fact use of ROB-M with PI resulted in zero failures for all uncertainty cases. There were also zero failures for CBBA with ROB-M in the medium and low uncertainty cases. It appears that ROB-M works well with CBBA to reduce the effects of uncertainty, as long as baseline CBBA is capable of solving the deterministic version of the task allocation problem.

There were no significant differences (99% level) between the mean completion times for CBBA with and without ROB-M. However, PI with ROB-M showed a significantly faster mean completion time than baseline PI. Again, this strongly suggests that the use of ROB-M does not compromise the performance of the algorithm in terms of the objective function.

7 Comparing PI-ROB-M to Ponda's robust CBBA algorithm

In a final set of experiments, PI with ROB-M is compared directly to results published in [7] using Ponda's robust CBBA algorithm - the version that uses the expected value of the task costs. In these runs, consistent with the published work, six heterogeneous vehicles (three helicopters and three UAVs) must rescue an increasing number of survivors, first 20, then 40, then 60. The problem used in experiment set 2 is chosen, the high uncertainty model described in Section 4.2 is adopted, 100 runs are completed for each combination of tasks and vehicles, and the mean number of solved tasks over the 100 runs is recorded. The results are shown in Fig. 7.

For PI with ROB-M, the 20 task and 40 task scenarios demonstratred a 100% success rate. The success rate reduced to 47% when the number of tasks increased to 60, but the number of unallocated tasks in each run remained low, with the mean number of solved tasks equal to 59. These results compare to mean numbers of solved tasks of 16 (20 total tasks) 25 (40 total tasks) and 30 (60 total tasks) for Ponda's robust CBBA algorithm, and provide strong supporting evidence that PI with ROB-M copes better under uncertain conditions.

8 Conclusions

Baseline PI does not handle uncertainty well; in all three experiment sets with the baseline, a high percentage of the solutions fail to allocate all of the tasks when simulated uncertainty is high, and the number of unallocated tasks is also relatively large. These figures tend to decrease linearly as the degree of modeled uncertainty is reduced, but the problem is still apparent. Taking the expected value of the time costs in PI reduces the failure rate and the numbers of unallocated tasks, but the method is still not reliable enough for time-critical problems. When the worst-case scenario metric is used, the algorithm demonstrates poor performance, especially for high uncertainty, as it always overestimates the time costs; this provides evidence that this strategy is not suited to problems that have individual task time constraints. However, when a combination of the expected value and the worst-case scenario metric is used, the results are greatly improved, in terms of both a more robust solution (a 0 to 10% failure rate and low number of unallocated tasks) and a significantly smaller objective function value for most of the uncertainty cases tested. For this reason, the hybrid worst-case-expected value method is adopted for the ROB-M robustness module. In addition, only 100 samples are required for ROB-M to achieve this low failure rate, which compares very favorably to the 10,000 samples used by Ponda [7] for the robust version of CBBA; the experiments performed here have also demonstrated success using more tasks and agents than those reported in [7].

It has also been shown that ROB-M can easily be bolted on to CBBA, and can reduce the effects of uncertainty, as

long as baseline CBBA is capable of solving the deterministic version of the task allocation problem attempted. Furthermore, t-test analysis has shown that the solution quality is not compromised when ROB-M is used, either with CBBA or PI. The final set of experiments compared the PI-ROB-M combination to results published in [7] and provided strong evidence that the architecture is able to cope better under uncertainty than Ponda's robust CBBA alorithm, as much-improved mean numbers of solved tasks were obtained.

Despite the small sample size, scalability in the numbers of agents and tasks is still a problem, with robust versions of the PI algorithm displaying a higher run time compared to the baseline equivalents. For the model problems tested, one run still completes in a relatively short time compared to the mission length, but there is a limit to usability in terms of the numbers of tasks and agents because of the increased computation time.

Future work will aim to improve the efficiency of the core PI algorithm and the ROB-M bolt on by refactoring both algorithms. Experiments will then provide empirically-derived estimates of how the robust agorithm scales as the number of samples, tasks and agents increases. The effects of reducing and increasing network connectivity will also be examined in relation to both algorithm performance and run-time efficiency. Experiments will also be carried out to test whether performance can be enhanced by tailoring the size of the buffer ψ to the individual time limits for each task. In addition, there are other risk-averse strategies besides using a buffer, for example using different estimates of the sample standard deviation; these methods will be examined and compared to the existing one in order to determine whether ROB-M's performance can be improved. Finally, for these time-critical problems, the benefits of using ROB-M are tied to the time limit to complete the tasks; it would thus be worth exploring whether there is any time limit threshold over which the success of the bolt on begins to pay off compared to the baseline solution.

The study of distributed robust optimization remains wide open. Most methods designed for the types of problem discussed in this paper do not consider uncertainty in their solutions. Thus, the main contribution here is the successful design and implementation of a robustness module for use with general, distributed task allocation algorithms.

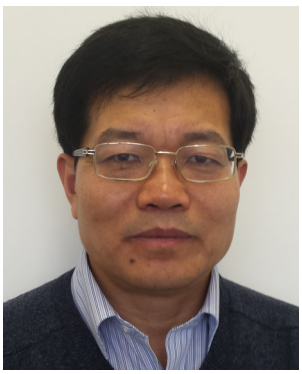
Acknowledgements This work was supported by EPSRC (grant number EP/J011525/1) with BAE Systems as the leading industrial partner.

References

1. Zhang K, Collins EG Jr, Shi D (2012) Centralized and distributed task allocation in multi-robot team via a stochastic clustering auction. ACM Trans Autonom Adapt Syst 7(2):21
2. Horling B, Lesser V (2004) A survey of multi-agent organizational paradigms. Knowl Eng Rev 19:281–316
3. Zhao W, Meng Q, Chung PWH (2015) A heuristic distributed task allocation method for multivehicle multitask problems and its application to search and rescue scenario. IEEE Transactions on Cybernetics 46(4):902–915
4. Choi H-L, Brunet J, How JP (2009) Consensus-based decentralization auctions for robust task allocation. IEEE Trans Robot 25(4):912–926
5. Grinstead CM, Snell DA (1998) Expected value and variance. In: Introduction to probability. 2nd edn. American Mathematical Society, Rhode Island, pp 936–953
6. Leito I, Helm I, Jalukse L (2015) Estimation of measurement uncertainty in chemical analysis: the normal distribution [online]. Available at https://sisu.ut.ee/measurement/31-normal-distribution. Accessed 11 Feb 2018
7. Ponda SS (2012) Robust distributed planning strategies for autonomous multi-agent teams. Dissertation, Mass. Inst. Technol
8. Whitbrook A, Meng Q, Chung PWH (2017) Reliable, distributed scheduling and rescheduling for time-critical, multi-agent systems. IEEE Trans Autom Sci Eng. https://doi.org/10.1109/TASE.2017.2679278
9. Maimon O (1990) The robot task-sequencing planning problem. IEEE Trans Robot Autom 6(6):760–765
10. Smith RG (1998) The contract net protocol: high level communication and control in a distributed problem solver. IEEE Trans Comput C 19(12):1104–1113
11. Pujol-Gonzalez M, Cerquides J, Meseguer P, Rodríguez-Aguilar JA, Tambe M (2013) Engineering the decentralized coordination of UAVs with limited communication range. In: Bielza C et al (eds) Advances in artificial intelligence. CAEPIA 2013. Lecture notes in computer science, vol 8109. Springer, Berlin, pp 199–208
12. Lagoudakis M, Markakis E, Kempe D, Keskinocak P, KJleywegt A, Koenig S, Tovey C, Meyerson A, Jain S (2005) Auction-based multi-robot routing. In: Proceedings of robotics: science and systems. MIT Press, Cambridge
13. Fave FMD, Farinelli A, Rogers A, Jennings N (2012) A methodology for deploying the max-sum algorithm and a case study on unmanned aerial vehicles. In: Proceedings of IAAI-2012, pp 2275–2280
14. Khamis A, Hussein A, Elmogy A (2015) Multi-robot task allocation: a review of the state-of-the-art. In: Koubâa A, Martínez-de Dios J (eds) Cooperative robots and sensor networks 2015. Studies in computational intelligence, vol 604. Springer, Cham
15. Bruno JL, Coffman EG, Sethi R (1974) Scheduling independent tasks to reduce mean finishing time. Commun ACM 17(7):382–387
16. Atay N, Bayazit B (2006) Mixed-integer linear programming solution to multi-robot task allocation problem. Technical Report 2006-54, Washington University, St. Louis
17. Gerkey BP, Matarić MJM (2004) A formal analysis and taxonomy of task allocation in multi-robot systems. Intl J of Robotics Research 23(9):939–954
18. Glover F, Marti R (2006) Tabu search. In: Alba E, Marti R (eds) Metaheuristic procedures for training neural networks. Springer, USA, pp 53–69
19. Shima T, Rasmussen SJ, Sparks AG, Passino KM (2006) Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms. Comput Oper Res 33(11):3252–3269
20. Oliver G, Guerrero J (2011) Auction and swarm multi-robot task allocation algorithms in real time scenarios. In: Yasuda T (ed)

Multi-robot systems, trends and development. intech, pp 437–456
21. Dias MB, Stentz A (2002) Opportunistic optimization for market-based multi-robot control. In: Proceedings of IROS-2002, pp 2714–2720
22. Bertsekas DP (1989) The auction algorithm for assignment and other network flow problems. Technical Report, Mass. Inst. Technol., Cambridge, MA
23. Dias M, Zlot R, Kalra N, Stentz A (2006) Market-based multirobot coordination: a survey and analysis. Proc IEEE 94(7):1257–1270
24. Coltin B, Veloso M (2010) Mobile robot task allocation in hybrid wireless sensor networks. In: Proceedings of IROS-2010, pp 2932–2937
25. Cramton P, Shoham Y, Steinberg R (2007) An overview of combinatorial auction. ACM SIGecom Exchanges 7(1):3–14
26. Whitbrook A, Meng Q, Chung PWH (2015) A novel distributed scheduling algorithm for time-critical, multi-agent systems. In: Proceedings of IROS-2015, pp 6451–6458
27. Tipping ME (2006) Bayesian inference: an introduction to principles and practice in machine learning. In: Bousquet O, von Luxburg U, Ratsch G (eds) Advanced lectures on machine learning. Springer, pp 41–62
28. Yang K, Wu Y, Hunag J, Wang X, Verdu S (2008) Distributed robust optimization for communication networks. In: Proceedings of INFO-COM-2008, pp 1831–1839
29. Andrieu A, De Freitas N, Doucet A, Jordan MI (2003) An introduction to MCMC for machine learning. Mach Learn 50:5–43
30. Undurti A, How JP (2011) A decentralized approach to multi-agent planning in the presence of constraints and uncertainty. In: Proceedings of ICRA-2011, pp 2534–2539
31. Musliner DJ, Durfee E. H., Wu J, Dolgov DA, Goldman RP, Boddy MS (2006) Co-ordinated plan management using multiagent MDPs. In: Proceedings of SSDPSM. American association for artificial intelligence
32. Maheswaran RT, Rogers CM, Sanchez R, Szekely P (2010) Realtime multi-agent planning and scheduling in dynamic uncertain domains. In: Proceedings of ICAPS-2010, pp 16–28
33. Ramchurn SD, Fischer JE, Ikuno Y, Wu F, Flann J, Waldock A (2015) A study of human-agent collaboration for multi-UAV task allocation in dynamic environments. In: Proceedings of IJCAI-2015, pp 1184–1192
34. Liu L, Shell DA (2011) Assessing optimal assignment under uncertainty: an interval-based algorithm. Int J Robot Res 30(7):936–953
35. Kuhn HW (1955) The Hungarian method for the assignment problem. Naval Research Logistic Quarterly 2:83–97
36. Munkers J (1957) Algorithms for the assignment and transportation problem. J Soc Ind Appl Math 5(1):32–38
37. Zlot RM (2006) An auction-based approach to complex task allocation for multirobot teams. Dissertation, The Robotics Institute, Carnegie Mellon University
38. Turner J, Meng Q, Schaeffer G (2015) Increasing allocated tasks with a time minimization algorithm for a search and rescue scenario. In: Proceedings of ICRA-2015, pp 340–3407
39. Collinson RGP (2011) Introduction to avionic systems, 3rd edn. Springer, London
40. Huston WB (1948) Accuracy of airspeed measurements and flight calibration procedures. Technical Report No 919, NACA, Langley Memorial Aeronautical Laboratory

Amanda Whitbrook received her B.Sc. (Hons) degree in Mathematics and Physics in 1993 from Nottingham Trent University, UK, her M.Sc. degree in Management of Information Technology in 2005 from the University of Nottingham, UK, and her Ph.D. degree in Applied Mathematics (Numerical Analysis) in 1998 from Nottingham Trent University. She has previously worked as a Research Fellow and Teaching Associate at the University of Nottingham, a Senior Scientist in the Autonomous Systems Research Group at BAE Systems, UK, and as a Research Associate at Loughborough University, UK. She is currently a Senior Lecturer in Computer Science in the Department of Electronics, Computing and Mathematics at the University of Derby with research interests in biologically inspired artificial intelligence (artificial immune systems, genetic algorithms, swarm optimization), mobile robot navigation, artificial intelligence for computer games, and heuristic methods for multi-agent task allocation.



Qinggang Meng received his B.Sc. and M.Sc. degrees in electronic engineering from Tianjin University, Tianjin, China, and his Ph.D. degree in computer science from Aberystwyth University, Aberystwyth, UK. He is currently a Senior Lecturer with the Department of Computer Science, Loughborough University, Loughborough, UK. His current research interests include biologically and psychologically inspired learning algorithms and developmental robotics, service and assistive robotics, robot learning and adaptation, multi-UAV cooperation, situation awareness and decision making for driverless vehicles, driver's distraction detection, human motion analysis and activity recognition, activity pattern detection, pattern recognition, artificial intelligence and computer vision.

Paul W. H. Chun received his B.Sc. degree in computing science from Imperial College London, U.K., in 1981, and his Ph.D. degree in artificial intelligence from the University of Edinburgh, U.K., in 1986. From 1984 to 1991, he was with the Artificial Intelligence Applications Institute, University of Edinburgh. He joined Loughborough University, U.K., in 1991, where he has been a Professor of Computer Science since 1999, the Head of the Department of Computer Science from 2004 to 2008, and the Dean of the School of Science from 2011 to 2014. He was a Visiting Professor with the Beijing University of Posts and Telecommunications, Beijing, China, and the Institute Technology of Brunei, Gadong, Brunei. He was an Invitation Fellow with Okayama University, Okayama, Japan. His current research interests include applying advanced computing techniques to solve novel complex problems. He has successfully supervised 30 doctorate students and published over 200 papers.