Auton Agent Multi-Agent Syst (2011) 22:415–438
DOI 10.1007/s10458-010-9134-5

Optimizing coalition formation for tasks with dynamically evolving rewards and nondeterministic action effects

Majid Ali Khan · Damla Turgut · Ladislau Bölöni

Published online: 14 May 2010
© The Author(s) 2010

Abstract We consider a problem domain where coalitions of agents are formed in order to execute tasks. Each task is assigned at most one coalition of agents, and the coalition can be reorganized during execution. Executing a task means bringing it to one of the desired terminal states, which might take several time steps. The state of the task evolves even if no coalition is assigned to its execution and depends nondeterministically on the cumulative actions of the agents in the coalition. Furthermore, we assume that the reward obtained for executing a task evolves in time: the more the execution of the task is delayed, the lesser the reward. A representative example of this class of problems is the allocation of firefighters to fires in a disaster rescue environment. We describe a practical methodology through which a problem of this class can be encoded as a Markov Decision Process. Due to the three levels of factoring in the resulting MDP (the states, actions and rewards are composites of the original features of the problem) the resulting MDP can be directly solved only for small problem instances. We describe two methods for parallel decomposition of the MDP: the MDP RSUA approach for random sampling and uniform allocation and the MDP REUSE method which reuses the lower level MDP to allocate resources to the parallel subproblems. Through an experimental study which models the problem domain using the fire simulation components of the Robocup Rescue simulator, we show that both methods significantly outperform heuristic approaches and MDP REUSE provides an overall higher performance than MDP RSUA.

M. A. Khan
Faculty of Computer Science and Engineering, Ghulam Ishaq Khan Institute of Engineering Science and Technology, Topi, Khyber Pakhtunkhwa, Pakistan
e-mail: majidkk@gmail.com

D. Turgut · L. Bölöni (✉)
School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816–2450, USA
e-mail: lboloni@eecs.ucf.edu

D. Turgut
e-mail: turgut@eecs.ucf.edu

Keywords Optimization · Coalition formation · Agents

1 Introduction

We consider a problem domain where a group of agents $\{Ag_1, \ldots Ag_k\}$ are forming coalitions to execute a set of tasks $\{T_1 \ldots T_n\}$. The tasks have a state which evolves in time even in the absence of the actions of the agents; in general, left on their own, tasks move to states which represent a higher difficulty. The effects of the actions of the agents are nondeterministic. The reward of executing the tasks evolves in time; in general, the later the execution is finished, the lower the reward. Our goal is to find the optimal allocation of the agents to the coalitions; this might involve the reorganization of the coalitions during task execution. We assume, however, that once a coalition is formed and assigned to a task, the agents in the coalition will choose the best possible collaborative action in order to further the execution of the task.

The running example used in the remainder of this paper is firefighting in disaster situations (such as, after an earthquake). Several fires are erupting in a city. There is an insufficient number of firefighters to cover all the current fires with an adequate number of resources simultaneously. The fires increase in intensity in time. The reward for putting out a fire, interpreted as a portion of the building which was saved, is decreasing in time. The result of the firefighting action is nondeterministic: the same amount of water might or might not put out a fire, due to some unknown parameters of the building.

How do we attack a problem like this? Naturally, simple heuristics can be developed through the analysis of the problem domain. Some ideas can be borrowed from related fields such as task scheduling or coalition games. However, these domains do not, by default, capture the dynamic nature of the problem, the evolving rewards and the uncertainty of the execution. They also do not exploit the ability to reorganize the coalitions during execution.

The objective of this paper is to provide practitioners a consistent methodology for encoding this class of problems as an optimization problem, which can then yield an appropriate coalition formation policy. The approach relies on the encoding of the problem as a Markov Decision Process (MDP). We assume that the practitioner has ample problem domain knowledge and has access to software libraries which allow him to solve general purpose MDPs. The main challenge, under these assumptions, is how to capture the problem domain in a MDP. Our solution builds a three-level factored MDP, where both the states, the actions and the rewards are composites of features of the original problem.

As any practical application of such a problem would involve hundreds or thousands of tasks and agents, and factored MDPs grow very quickly with the problem size, we also need to provide a practical solution for scaling. Our working assumption is that such a system would be used in real-time applications. In the time-accelerated world of the Robocup Rescue simulator our system was tuned such that the decision time was kept below 20 ms for 100 buildings and 200 firefighters. In many real world applications where actual physical coalitions are formed, somewhat higher running times would be acceptable. This, however, does not change the real time nature of the problem or the importance of the scaling techniques.

The scaling approach we propose is based on the parallel decomposition of the MDP. We describe two concrete algorithms: the MDP RSUA algorithm relies on the convergence of a random sample to the properties of the sampled distribution, and does not require domain specific information. The MDP REUSE approach reuses the MDPs calculated for the subproblems for an artificially created higher level problem which allocates resources to the partitions. This approach requires problem domain information: the user needs to provide a distance function over the state space.

The set of concerns on which the present paper is focused are shaped by the desire to be useful to a practitioner building a real-time system. While MDPs in general are a well understood tool, the encoding of a real world problem in an MDP raises non-trivial decisions. Thus, we allocate a relatively large part of the paper to a detailed methodology. In contrast, we choose not to explore some topics of theoretical interest, but which are not of practical applicability in a real time system. For instance, it would be of theoretical appeal to divide the uncertainty about the firefighters into a proper execution variance component and another component which models the uncertainty of our observations about the fuel available in the building. This would lead naturally to a POMDP model, introducing a whole new level of modeling and computational complexity. However, without a high quality observation model, which is rarely available in practical settings, the theoretical beauty of this model would not be translated into higher performance.

We also decided not to focus on the various accelerated and/or approximate techniques to solve factored MDPs. This field has a rich literature and certainly some of the techniques can be applied to the MDPs created following our methodology. However, the practitioner, whom we envision as the beneficiary of this work, can not be expected to perform a graph theoretic analysis on the created MDP. He might, however, swap out the classical MDP solution algorithms (variants of value or policy iteration) for a more advanced, possibly approximate solver. By allowing to solve larger MDPs, these approaches would allow to reduce or even eliminate the necessity of parallel decomposition, possibly increasing the quality of the solution.

The remainder of this paper is organized as follows. Section 2 discusses related problem domains. Section 3 describes the methodology to create an MDP to model the problem. The problem of scaling and the two proposed parallel partitioning approaches are described in Sect. 4. Section 5 describes the simulation study in which the methodology is applied to the firefighter allocation domain. We conclude in Sect. 6.

2 Related work

The subject of this paper can be identified as a multi-agent coalition structure formation problem, where the coalitions perform actions towards specific tasks, and the coalitions can change in time.

In *coalition games*, the value of the coalition is given by its structure [1,2,8,11,16], every coalition being assigned a value by the characteristic function (which can be stochastic [18]). Very often, the main challenge in game theory is the allocation of the value between the participants in a coalition.

In contrast to game theoretic approaches, in multi-agent settings, the value of the coalition also depends on the specific actions taken by the coalition. MDPs and the closely related reinforcement learning models are a good choice to represent multi-agent coalition problems because they can simultaneously model the coalition structure and the actions taken by the individual coalitions. The resulting MDPs have action and transition models which integrate both the coalition structure and the chosen action. This structure can lead to very large MDPs, but also opens opportunities for solution methods significantly faster than general purpose value or policy iteration.

One of the approaches most closely related to our work is described in [14]. The authors introduce a formal description of the finite horizon decision problems with both probabilities and rewards non-stationary. Then, they focus on a special case of this problem, targeted towards the application domain of air campaign planning and find an approximate solution using the decomposition of the problem into several weakly coupled MDPs.

The problem considered by us, although not strictly speaking finite horizon, can be made to conform to this general class by observing that any fire will burn for only a finite time. Thus, our work can also be considered a special case of the general problem, which, however, does not overlap with the one considered by Meuleau et al. The most important differences are:

- The state of the fire changes even if no firefighter is allocated to the fire. In an air campaign, damage to the target only occurs when airplanes are assigned to the target.
- A fire which was "contained" but not extinguished by the firefighters can "rekindle" itself if the firefighters are assigned elsewhere. Damaging a target is a cumulative process, targets do not self-heal.
- The reward in the air campaign domain is secured at the moment when the damage is done, while in case of the firefighting domain, the reward is allocated only when a terminal state is reached. There is no reward for slowing down the burning if eventually the building burns out.
- The reward of the firefighters is the saved portion of the building. There are no fixed deadlines, nor an explicit reward for putting out a fire faster. By allocating a small number of firefighters to the building, the increase of the fire can be delayed. In contrast, in the air campaign problem, there are strict availability windows when a target can be damaged.

In several papers, Chalkiadakis and Boutilier considered the problem of sequential coalition formation with agents uncertain about each other's type. At each round, the agents need to decide about which coalition to join, and what coalition action to take. Rewards are acquired after each round, which can be used to infer the value of the coalition and the types of the other agents. The agent's goal is to optimize the long term reward. The agents use a reinforcement learning type strategy, with occasional exploration actions to learn the types of other agents. The optimal behavior for an agent can be obtained as a solution to a POMDP. Finding an exact solution of the POMDP is not feasible in practical applications, but approximation algorithms, either myopic [5] or more far-sighted [6] can provide a performance adequate for real-world deployment. The best performance has been found to be provided by approaches based on the exploration of the Value of Perfect Information (VPI).

The problem attacked by our paper is different in several important aspects. There is an externally given set of dynamically changing tasks, the agents can not choose their own actions, and the allocation is made by a manager trying to optimize the overall reward. Although coalitions can change, rewards are only given when the task is brought to a terminal state. Finally, the nature of the applications we consider does not allow for a reinforcement learning approach.

Our problem can also be considered as a case of planning. For instance, the coalitions formed to solve the tasks can be considered an example of conditional plans. An extensive survey of the use of MDP formalisms for planning is provided in [4].

The approach described in [9] relies on a system where a multi-agent planning system with locality properties is modeled through a factored MDP, with the transition model represented by a Dynamic Bayesian Network, and the value function of the MDP being a linear combination of pre-selected base functions. The one-step lookahead algorithm is used to calculate the local contributions of the agents to the total utility function. A series of algorithms for the efficient solving of factored MDPs are proposed in [10].

The MDP model is not the only approach for coalition formation in multi-agent scenarios. Other approaches are based on online, local interactions either through formal negotiation or argumentation, or predefined exchange of local information. Although such approaches cannot strive for optimality, they represent a simpler, more scalable alternative which does not require a central authority.

A summary of this type of approaches and a future research agenda is outlined in [13]. They also outline a customizable, simulation based dynamic coalition framework (DCF-S), which requires that the coalition leading agent determines valid coalitions before starting negotiations. The paper also calls for research on developing coalition algorithms for stochastic and fuzzy domains.

Shehory and Kraus [17] provide an extensive investigation of approaches where coalitions of agents must be formed for the allocation to tasks, but the value of coalitions is not super-additive. They consider both the case of disjoint and overlapping coalitions, as well as the case where tasks have priority values. The proposed coalition formation methods rely on a distributed, iterative calculation of the coalition values, and have the useful "anytime" property: algorithms interrupted before termination would still yield an approximate solution.

One recent example of this class of approaches is [7], where coalitions of sensors are dynamically created in order to more efficiently identify targets. Examples of these sensors might be remote controlled cameras deployed in a field where the actions of the sensors are the change of the orientation and of the zooming factor. In this approach, the sensors repeatedly exchange information with their neighbors until they converge to a specific coalition structure. The locality of the communication ensures the scalability of the approach.

3 A methodology for modeling the problem domain as an MDP

In the following we describe a general methodology to model the considered problem domain as an MDP. Agents $\{Ag_1, \ldots Ag_k\}$ are forming coalitions to execute tasks $\{T_1 \ldots T_n\}$. The tasks have a state which evolves nondeterministically in time, dependent on the actions of the coalition assigned to it (if any). The reward of executing the tasks evolves in time; in general, the later the execution is finished, the lower the reward. Our goal is to find the optimal allocation of the agents in the coalitions; coalitions can be reorganized at any moment in time.

The methodology is composed of the 7 steps described in Table 1. For the presentation of every step, we first discuss the general approach, then illustrate it using the firefighter domain. The same firefighter application will be used in the experimental evaluation of the approach.

Table 1 The steps of the MDP modeling

Step 1	Develop a stochastic model for the evolution of a single task in time
Step 2	Develop a model for the joint action of the agents
Step 3	Develop a stochastic model of the evolution of tasks in response to the actions
Step 4	Develop a cost and reward model
Step 5	Using the models developed above, specify a Markov decision process modeling the problem domain
Step 6	Solve the MDP using the appropriate algorithms
Step 7	Interpret the MDP. Assemble the policy for the agents from the solutions of the MDP.

3.1 Step 1: Develop a model for the evolution of the individual task in time

We assume that the state of the task T at time t can be characterized by a *measure* $M(T, t)$. We are looking for an expression which describes the state of the task at time $t + 1$, in the absence of any actions from the agents. The evolution of the state is assumed to be nondeterministic. The value of a (continuous) measure at time $t + 1$ would then be described as a probability distribution over possible values of M. As such distributions are very difficult to acquire, we choose to *discretize* the value of M, assuming that the value can be one of the discrete values $M(T, t) \in \{m_1, m_2 \ldots m_q\}$.

The number of discrete values is technically limited by the accuracy of the measure. In practice, however, we might choose to represent a smaller number of discrete states to reduce the size of the resulting MDP.

Once the $M(T, t)$ is expressed as a discrete value, $M(T, t + 1)$ can be expressed with a series of probabilities. If we assume $M(T, t) = m_x$, we have:

$$M(T, t+1) = \begin{cases} m_1 \text{ with probability } p^T(m_x \xrightarrow{\emptyset} m_1) = p^T_{x1} \\ \ldots \\ m_q \text{ with probability } p^T(m_x \xrightarrow{\emptyset} m_q) = p^T_{xq} \end{cases} \quad (1)$$

with $\sum_{i=1}^{q} p^T_{xi} = 1$. These probabilities form a matrix of q^2 values, which can be acquired either through theoretical analysis of the problem domain or from historical data. The probabilities p^T_{ij} are normally task dependent. However, in most scenarios, the tasks are not unique, but can be seen as coming of a finite number of classes. Thus we need to acquire only a finite number of probability matrices.

In the firefighter domain, the measure of the task is the state of the fire in the building. Our representation uses the eight discrete states shown in Table 2.

As every building is different and setting buildings on fire is not an acceptable method of obtaining transition probability data, we need to cluster the buildings in types and predict the evolution of the fire through historical data from buildings of the given type. We consider

Table 2 The discrete states of a task representing a building on fire

Fire level	Description
NO-FIRE	The building is not on fire
LOW-FIRE	The building has just caught fire
LOW-BURNT	The building was extinguished soon after catching fire
MEDIUM-FIRE	The building has been on fire for some time
MEDIUM-BURNT	The fire was extinguished after being on medium fire
HIGH-FIRE	The building has been on fire for a long time
HIGH-BURNT	The fire was extinguished after being on high fire
COMPLETELY-BURNT	The building was completely burnt out

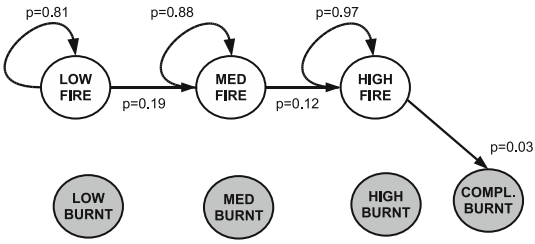


Fig. 1 The transition probabilities for a single small building. The terminal states are *gray*. Note that some of the states are not reachable

three types of buildings distinguished by their size: SMALL (defined as smaller than 1000 sqft), MEDIUM (between 1001 and 3000 sqft) and LARGE (larger than 3000 sqft).

For instance, the probabilities in the case of a small building create the Markov chain in Fig. 1. We assume LOW-FIRE to be the initial state. Note that some of the states are not reachable in this graph, reflecting the fact that buildings on fire do not extinguish themselves. If the fire is left unattended, the building will burn out completely.

3.2 Step 2: Develop a model for the coalition actions of the agents

At this step we need to determine the actions which can be executed by the individual agents, and the ways in which the actions are assembled into joint actions for the case when the agents are acting in a coalition.

In general, if agents $Ag_1 \ldots Ag_k$ are acting in a coalition towards achieving a task T, each agent A_i chooses an individual action α_i. The composition of the individual actions forms a *coalition action* $c\alpha = \langle \alpha_1 \ldots \alpha_k \rangle$. The effect of the tuple can be a complex function of the individual actions.

If each agent can choose from m possible actions, this means that m^k distinct coalition actions are theoretically possible. This number, however, can be usually drastically reduced with careful domain specific analysis. Most application domains have a limited set of feasible coalition actions. A coalition action, on its turn, determines the actions of the participating agents.

In the firefighting domain, the action model is very simple. The only action the agents can take is to use water to extinguish the fire. The agents are assumed to be homogeneous, and the resulting action is the sum of the actions. If each firefighter can apply 5 units of water in a unit of time, 10 firefighters will apply 50 units of water. Thus, the actions can be simply represented with the number of firefighters participating in the coalition.

Note, however, that the linear composition of the *actions* does not necessarily mean a linear composition of the *effects* of the actions. It is not, in general, true that 50 units of water will extinguish the fire 10 times faster than 5 units. Under a certain threshold, the application of water would not extinguish the fire at all, only extends the time it takes for the building to burn out.

3.3 Step 3: Develop a stochastic model of the evolution of the tasks in response to actions

In Step 1 we have considered the evolution of tasks without any actions from agents, in Step 2 we considered the model of the composition of the actions of the agents into coalition actions. In this step we consider the evolution of the tasks with the coalition actions being applied. We assume that task T is at time t in state m_x. A coalition of agents $CAg = \langle Ag_1, \ldots Ag_k \rangle$ is acting on the tasks performing a coalition action $c\alpha = \langle \alpha_1 \ldots \alpha_k \rangle$. We are interested in the state of the task at time $t + 1$.

$$M(T, t+1, c\alpha) = \begin{cases} m_1 \text{ with probability } p^T(m_x \overset{c\alpha}{\rightarrow} m_1) \\ \ldots \\ m_q \text{ with probability } p^T(m_x \overset{c\alpha}{\rightarrow} m_q) \end{cases} \quad (2)$$

Note that now the probabilities depend both on the current state, the task and the coalition action. If we assume that there is a discrete number n of possible coalition actions, we will have n independent probability matrices of size q^2. These probabilities can be acquired from historical information or domain specific analysis.

While the state of the possible states of the task is the same as in Step 1, the effects of the actions frequently make states which were not reachable in the absence of an action, reachable.

3.3.1 Problem subclass: commander assigning agents to tasks

Our running example involving firefighters in disaster situations is a specific case of a large class of problems in which a commander or manager is assigning coalitions of agents to tasks. This class of problems frequently come with two specific assumptions: *uniformity of agents* and *delegated action*.

A trained firefighter is assumed to be able to execute all the actions pertinent to his assignment at a certain expected level. Similar considerations apply to soldiers, clerks, taxi drivers, customer service representatives and many other jobs. Even highly skilled professionals such as surgeons are considered as having the same level of abilities from the point of view of hospital planning or insurance companies.

Tailored allocation based on specific skill sets, although an interesting and challenging problem on its own, is not practical at the level of allocating hundreds of agents. Part of the problem is the lack of a catalog of provable individual skills—as opposed to the uniform skill sets which are validated by degrees, certification programs and other similar means. Thus, from the point of view of a manager, all agents have uniform skill sets (or are grouped in a small number of classes with uniform skill sets).

The other assumption we make is delegated action. The manager assembles the coalitions of agents and assigns them to specific tasks. However, the manager does not directly assign the agent the concrete action to be executed. Rather, the assignment is "do the proper action for task t". At the local level, the team organizes itself and takes actions according to the situation—with the different agents being assigned specific actions. These specifics, however, are not visible from the point of view of the commander.

In the firefighting domain, the result of these assumptions is that we can model the coalition action simply with the number of agents participating in firefighting. Figure 2 illustrates the evolution of the state of a small building as the result of the coalition action of

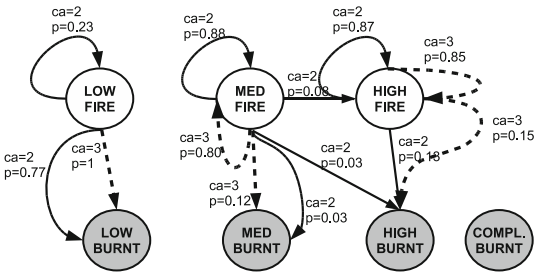


Fig. 2 The transition probabilities for a single small building as the effects of the coalition actions of 2 (*continuous lines*) or 3 (*dotted lines*) firefighters. The terminal states are gray

Table 3 Transition probabilities for small building with four firefighters

Fire level	Fire fighters	1	2	3	4	5	6	7
1	0	0.81	0.19	0.00	0.00	0.00	0.00	0.00
	1	0.81	0.19	0.00	0.00	0.00	0.00	0.00
	2	0.23	0.00	0.00	0.77	0.00	0.00	0.00
	3	0.00	0.00	0.00	1.00	0.00	0.00	0.00
	4	0.00	0.00	0.00	1.00	0.00	0.00	0.00
2	0	0.00	0.88	0.12	0.00	0.00	0.00	0.00
	1	0.00	0.88	0.12	0.00	0.00	0.00	0.00
	2	0.00	0.88	0.08	0.00	0.03	0.03	0.00
	3	0.00	0.80	0.00	0.00	0.20	0.00	0.00
	4	0.00	0.71	0.00	0.00	0.29	0.00	0.00
3	0	0.00	0.00	0.97	0.00	0.00	0.00	0.03
	1	0.00	0.00	0.93	0.00	0.00	0.07	0.00
	2	0.00	0.00	0.87	0.00	0.00	0.13	0.00
	3	0.00	0.00	0.85	0.00	0.00	0.15	0.00
	4	0.00	0.00	0.82	0.00	0.00	0.18	0.00

The fire levels are indicated with integer values and are associated as LOW-FIRE=1, MEDIUM-FIRE=2, HIGH-FIRE=3, LOW-BURNT=4, MEDIUM-BURNT=5, HIGH-BURNT=6 and COMPLETE-BURNT=7 respectively

2 (continuous lines) or 3 (dotted lines) firefighters. To keep the figure readable, we did not include actions with 1, 4 or more firefighters. Comparing with Fig. 1, we can make several observations. States such as LOW-BURNT, MEDIUM-BURNT and HIGH-BURNT, which were not reachable in the absence of actions, are now reachable. On the other hand the COMPLETE-BURNT state is not reachable in this subgraph, reflecting the fact that if there are at least two firefighters working on a small house, the house will never be completely burnt. Another observation is that the states form two disjoint graphs—there is no transition from the LOW-FIRE state to the MEDIUM-FIRE state for the action of the 2 or 3 firefighters. Still the building can reach the MEDIUM-FIRE state, for instance if no firefighter is working on it. As our problem domain assumes a *dynamic* formation of the coalitions, this is possible.

The source of these transition probabilities is normally *historical information*. In our case, we have extracted transition probabilities from a collection of simulator logs. Obviously, in our case, more accurate data could have been obtained by looking at the source code of the fire model and the parameters of the buildings, such as the type and quantity of available fuel. This, however, would be an information to which the firefighter commander would not have access. The full set of transmission probabilities for the case of a small building and coalitions of at most 4 agents is shown in Table 3.

3.4 Step 4: Develop a cost and reward model

Until now we did not consider which states of the tasks are more favorable, nor the cost associated with the actions. Let us discuss the problem of rewards by contrasting it with classical task scheduling. In task scheduling, a task can be in one of the four states: READY, RUNNING, DONE and FAILED, with the latter two being terminal states. Naturally, we prefer the DONE state to the FAILED state. We say that the DONE state carries a *reward* $r > 0$, while the FAILED state carries either no reward, or it carries a *penalty* $p < 0$.

In the class of problems we are considering, rewards are only associated with *entering* a terminal state (corresponding to the completion of a task). The reward expresses the relative preference over the terminal states. In addition to the rewards we also consider the costs associated with the actions. These costs allow us to express preferences over various actions with equivalent or similar effects.

In the firefighting domain we define the reward to be proportional with the saved (unburnt) area of the building:

$$B_i = \begin{cases} A_i \times 3/4, & \text{if } F_i = \text{LOW-BURNT} \\ A_i \times 1/2, & \text{if } F_i = \text{MEDIUM-BURNT} \\ A_i \times 1/4, & \text{if } F_i = \text{HIGH-BURNT} \\ 0, & \text{if } F_i = \text{COMPLETE-BURNT} \end{cases} \tag{3}$$

where, A_i represents the total area of the building i, B_i represents the unburnt area of the building i and F_i is the fire level of building i.

As the firefighters have a single possible action (i.e. firefighting), we define a cost for a collaborative action to be a small number (in our case, 0.01) multiplied with the number of firefighters participating in the coalition. There is no cost for agents which are not assigned to any building.

This choice of rewards and costs ensures that there will never be a case where a building will be left burning to save on the action costs. On the other hand, in general we will prefer smaller coalitions, as long as the achieved terminal state is identical, and will prefer to reach the specific terminal states faster.

3.5 Step 5: Specify a Markov decision process

In the next step of the process, using the models developed in steps 1–4 we create a Markov decision process whose solution allows us to extract the optimum desired coalition structure.

To build the MDP we need to specify the (a) states, (b) the actions (c) the transition probabilities and (d) the rewards. Although the components rely on the models developed previously, assembling the MDP requires non-trivial technical decisions.

3.5.1 Determining the global states of the MDP

A *global state* defines the MDP state in terms of the set of tasks in the system. The global state of the system with n tasks is a n-tuple of the discrete states of the individual tasks: $gm = \langle m^{(1)}, \ldots m^{(n)} \rangle$ where $m^{(i)} \in \{m_1 \ldots m_q\}$. There are n^q number of possible states.

3.5.2 Determining the global actions

The actions of the system are defined by the coalitions formed and the actions taken by that coalition. For the sake of a uniform representation we assume that there is one coalition per task, which can be an empty coalition. As we had seen the coalition action is determined by the actions of the individual agents in the coalition. The global action $g\alpha$ is thus defined as:

$$g\alpha = \langle c\alpha_1 \ldots c\alpha_n \rangle \text{ where } c\alpha_i = \langle \alpha_{i1} \ldots \alpha_{ik_i} \rangle \tag{4}$$

where k_i is the number of agents in coalition i. The action of an empty coalition is the empty action.

The number of possible coalitions is, naturally, very large. If we have n tasks and k agents, the number of coalitions can be calculated by considering that each agent chooses

independently which task will it work on, with the number of possible coalitions being n^k. If we assume that the agent might also choose not to be part of any coalition, the number of alternatives becomes $(n+1)^k$.

Naturally, we need to find ways to reduce the number of potential coalitions and actions. This can be done by exploiting application specific features. In the firefighting domain, exploiting the previously introduced uniformity of agents and delegated actions properties we can represent a global action $g\alpha$ only with the number of agents participating in each coalition:

$$g\alpha = \langle k_1 \dots k_n \rangle \quad \text{where} \sum_{i=1}^{n} k_i = k \tag{5}$$

Algorithm 1 A recursive algorithm for integer partitioning

```
1: function IntegerPartition(k, n)
2:   C ← empty list of the list of integers
3:   if (n = 1) then
4:     L ← empty list of integers
5:     Add k to the list L
6:     Add the list L to the list C
7:     return C
8:   end if
9:   for i ← 0 to k
10:    for all list P ∈ IntegerPartition(k − i,n − 1)
11:      Append i to the front of the list P
12:      Add the list P to the list C
13:    end for
14:  end for
15:  return C
16: end function
```

The possible coalitions under these assumptions can be generated by the possible *weak integer partitions* of k into n places. For instance, for dividing 3 agents into 2 partitions we have the following choices: $\{\langle 0, 3\rangle, \langle 1, 2\rangle, \langle 2, 1\rangle, \langle 3, 0\rangle\}$. Weak integer partitions allow some of the partitions to have the value zero, in contrast to strong partitions where all partitions need to be non-zero. To our best knowledge, there is no simple analytical expression for the number of weak partitions,[1] but Table 4 gives a good idea of their increase.

A recursive algorithm for partitioning k agents to n tasks is presented in Algorithm 1.

3.5.3 Assigning probabilities

If we make the assumption that the tasks are independent of each other, then the probability of the transition from a global state to another as a response to a global action is the product of the transition probabilities on a per task basis, with respect to the coalition action applied to the given task.

[1] The number of strong partitions (where each partition is required to be non-zero) is given by the Stirling number of the second kind. The number of weak partitions is at least as large because every strong partition is also a weak partition, but not the other way around. The difference in some cases can be large. For instance in Table 4 the number of weak partitions of 10 agents into 10 coalitions is 92378, while the number of strict partitions is 1.

Table 4 The number of the weak partitions of m items into k partitions

m/k	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	1	3	6	10	15	21	28	36	45	55
3	1	4	10	20	35	56	84	120	165	220
4	1	5	15	35	70	126	210	330	495	715
5	1	6	21	56	126	252	462	792	1287	2002
6	1	7	28	84	210	462	924	1716	3003	5005
7	1	8	36	120	330	792	1716	3432	6435	11440
8	1	9	45	165	495	1287	3003	6435	12870	24310
9	1	10	55	220	715	2002	5005	11440	24310	48620
10	1	11	66	286	1001	3003	8008	19448	43758	92378

If the system is in a global state $gm_x = \langle m_x^{(1)}, \ldots m_x^{(n)} \rangle$, and the global action performed is $g\alpha = \langle c\alpha_1 \ldots c\alpha_n \rangle$, the probability to transition to a new global state $gm_y = \langle m_y^{(1)}, \ldots m_y^{(n)} \rangle$ will be:

$$p\left(gm_x \xrightarrow{g\alpha} gm_y\right) = \prod_{i=1}^{k} p^{(i)}\left(m_x^{(i)} \xrightarrow{c\alpha_i} m_y^{(i)}\right) \tag{6}$$

If the tasks are not independent, their interdependencies need to be taken into account. For instance, in the case of our running example, it is possible that two buildings on fire are located next to each other, and thus the evolution of the fires is co-dependent. This adds additional modeling difficulty, but it does not change the size of the resulting MDP.

Let us illustrate the building of the final MDP through a simple example. To keep the graph readable, we will simplify the firefighting model to a model where each building can be in only 3 possible states: Fire (F), Burned (B) and Extinguished (E). We assume that if there is at least one firefighter working on the building, it will be extinguished, while if no firefighter is working on the building, it will burn down. Figure 3 illustrates the building of the final MDP for two buildings and one single firefighter. The resulting MDP, as expected, has $3^2 = 9$ states. The possible team actions, in each state are two: $\langle 1, 0 \rangle$ representing that the firefighter is allocated to the first building, and $\langle 0, 1 \rangle$ representing that the firefighter is allocated to the second building.

3.5.4 Costs and rewards

The final problems regarding the building of the MDP are the assignment of the costs and rewards. The definition of the MDP can be formulated in three different ways: either by assigning rewards to being in a given state $R(s_i)$, with an action in a given state $R(s_i, \alpha)$ or with a certain transition taken as a result of an action $R(s_i, \alpha, s_j)$. While the formulations are ultimately equivalent, in the sense that they define the same design space, the MDP solvers usually support only one of them. Depending on the original problem formulation, we might need to transform the MDP to accommodate the different formulation. Most MDP solvers expect the rewards to be associated with state-action pairs.

In our model, we have costs associated with actions, and one-time rewards for reaching a terminal state. The cost for an action can be modeled in a $R(s_i, \alpha)$ model by replicating the same cost over all the possible states s_i where the action α can be taken. The representation

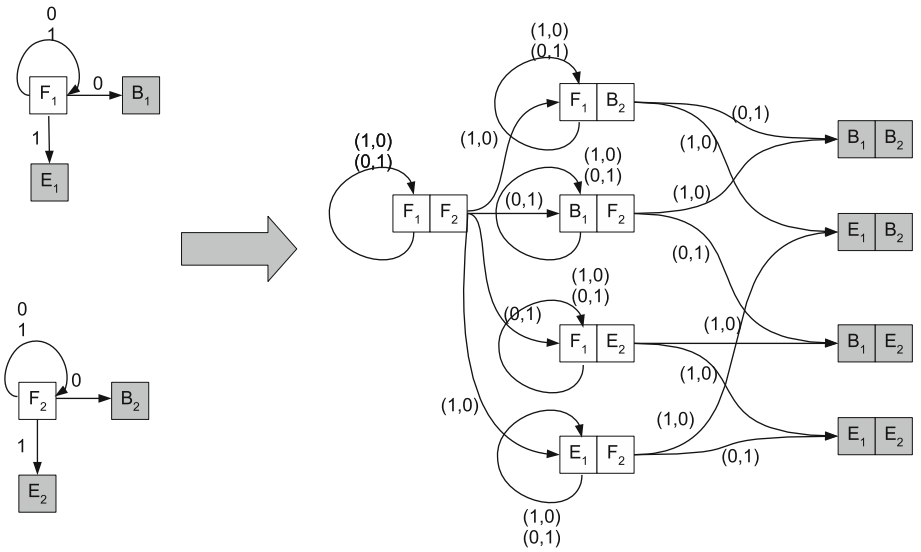


Fig. 3 Creating an MDP for a simplified version of the firefighter problem with only three states: Fire (F), Burned (B), and Extinguished (E). The final MDP is assembled for the case of one firefighter and two buildings

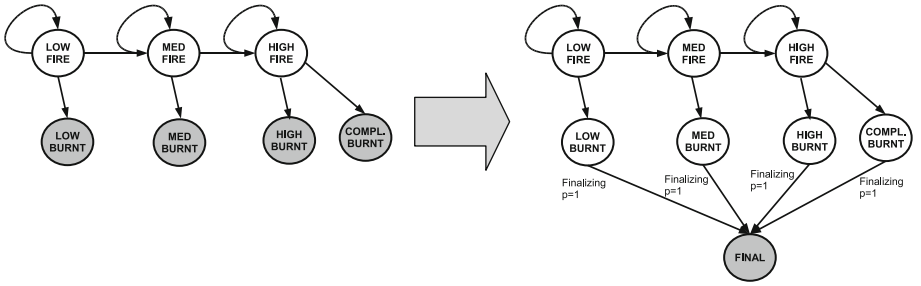


Fig. 4 Transformation of the MDP to allow for the expression of a reward for reaching a terminal state. A new, single terminal state FINAL is introduced. The previous terminal states are becoming non-terminal, and a special action Finalizing moves from them to the FINAL state with the probability $p = 1$. The rewards will be attached to the Finalizing transition

of the one time reward, however, is problematic: as the reward is defined on the starting state and the action taken, we can not assign the reward, as there is no guarantee that the next state will be the terminal state. To work around this problem, we modify the MDP in such a way that the reward can be attached to a transition which happens with a probability of 1. We introduce a new terminal state called FINAL and a special action called Finalizing. The original terminal states of the MDP will become non-terminal (we will mark them in our code as *semi-terminal*). The Finalizing action taken in these states will transition with a probability $p = 1$ to the FINAL state. Now, we can attach the reward to the combination of the semi-terminal state and the Finalizing action. The actual cost or reward will be the sum of rewards of the component states.

For the case of firefighting with a single building the transformation is shown in Fig. 4.

3.6 Step 6: Solve the MDP

This step involves solving the MDP using one of the available methods. The classical methods for MDP solving are value iteration, policy iteration, and various hybrid methods and variants. Solvers are available as libraries in various programming languages. There is a rich literature of various approximate solvers as well as solvers which can exploit specific features of the MDP. For approaches for solving factured MDPs, see [10].

3.7 Step 7: Interpreting the results

Let us now discuss how the methodology creates a solution for the problem we started with. Returning to our running example, let us assume that we have a problem with 5 buildings on fire: 2 LARGE, 2 MEDIUM and 1 SMALL and 10 firefighters. The individual states of the MDP will be quintuples, each member describing the state of a building. An example of this state would be ⟨LARGE_MEDIUM_FIRE, LARGE_LOW_FIRE, MEDIUM_MEDIUM_FIRE, MEDIUM_COMPLETELY_BURNT, SMALL_LOW_FIRE⟩. All the possible combinations of fire levels for all the buildings present will be in the MDP.

The solution of the MDP will be a *policy* which maps every state to the optimal action. The optimal action for the above state, for example, might be ⟨5, 2, 2, 0, 1⟩. This means that the first building will be assigned a coalition of 5 firefighters, and so on. When one of the tasks is in a terminal state, in our case MEDIUM_COMPLETELY_BURNT, we will assign it an empty coalition.

A firefighter commander, operating in real time, would operate as follows. When first learning of the set of tasks and the available resources, he or she will create the MDP according to the methodology and solve it. It is also possible to pre-solve a number of MDPs and store their associated policies. In any case the MDP needs to be solved only once for a certain set of tasks and agents.

The commander then proceeds to create the coalitions according to the current policy and based on the current state. As a note, there is no particular entry state in the MDP – it is possible that the buildings have been burning for some time before the controller takes the first decision (a phenomena called *preburn*, which will be modeled in our experimental study).

The allocated agent coalitions would then proceed on working on the tasks. Every time the state is changed, the commander looks up a new policy and, if necessary, reorganizes the coalitions. Note, again, that this reallocation of coalitions does not require the solving of an MDP, only a simple policy lookup.

Let us assume that our new state is ⟨LARGE_MEDIUM_FIRE, LARGE_MEDIUM_FIRE, MEDIUM_HIGH_BURNT, MEDIUM_COMPLETELY_BURNT, SMALL_LOW_BURNT⟩. The corresponding action would be ⟨7, 3, 0, 0, 0⟩ which requires the reallocation of the firefighters from the third and fifth building. Note that two identical buildings in identical states might not be assigned identical size coalitions. In fact, in our experimental study we shall see that spreading our forces thin by performing a uniform allocation is one of the worst possible strategies. Other problem domains, naturally, have their own requirements.

4 Scaling up

4.1 Scalability challenges

Let us denote the problem involving n tasks and k agents $CF(k, n)$. The MDP created according to the technique described previously has q^n states, where q is the number of discrete states of a task. For each state, the number of actions corresponding to the possible coalitions which the agents can form is the number of weak integer partitions of k agents into n partitions, which, as we have seen in Table 4, can be very large.

The resulting MDP not only has a large number of nodes (increasing exponentially with the number of tasks) but it is also *dense*, with a very large number of actions for every state. This number can be somewhat reduced by commonsense filtering of the actions—for instance, by eliminating the actions where we assign firefighters to extinguished fires. Unfortunately, for a problem with 10 buildings and 10 firefighters, we will have 10^{10} states, with each state having 92378 actions. Obviously, even this relatively small problem pushes the limits of what is possible to solve (even assuming an offline solution).

4.2 A parallel decomposition approach for scaling

To extend the proposed methodology to problems of realistic size we will use a *parallel decomposition* of the MDP, where the state space will be a cross-product of the individual MDPs. The MDP resulting from the described methodology is *three-way factored*—the states, actions and reward function are each assembled from a number of independent features. The states are composed of the states of n tasks. The actions are a composition of a number of coalitions formed to perform the task—the effects of the coalitions on the respective tasks are independent, but the sub-actions are linked by the resource constraints [14].

We are performing the parallel decomposition as follows. Consider the set of tasks to be executed $\{T_1, \ldots T_n\}$ and the set of k agents $\{Ag_1, \ldots Ag_k\}$. We partition the tasks into N disjoint partitions of sizes $n_1, \ldots, n_N$. Usually we will choose the sets to be of the same size (as much as it is possible). We partition the set of agents into corresponding sets of sizes $k_1 \ldots k_N$. This way, we will need to solve N problems $CF(k_i, n_i)$ instead of a single large problem $CF(k, n)$. The subproblems will be solved optimally, and they immediately define the coalitions which are going to be assigned to the tasks. However, the act of partitioning already involves allocation decisions and limits the choices in the subproblems.

In general, we want as few partitions as possible because the act of partitioning reduces the accuracy of the solution. The rule of thumb is: choose the smallest N for which the partitioned subproblems are still optimally solvable.

In the following we describe two choices for the partitioning of the coalition formation problem: random sampling with uniform allocation and partitioning by reusing the lower level MDP in the partitioning process.

4.3 Random sampling with uniform allocation (MDP RSUA)

The idea behind this approach is to partition the original problem into subproblems of the same difficulty level, and allocate a uniform number of agents to the partitions. This is a sensible approach as long as the problem size is not too small. The challenge is how to create the partitions of equal difficulty: in the firefighting domain, the difficulty is a complex multidimensional function depending on the various properties of the subtasks (such as the size of the building and the level of the fire), for which we do not have an explicit model.

On the other hand, bad partitioning can seriously impact the performance. For instance, in a natural disaster, the larger fires will be reported earlier than the smaller ones. Assigning the tasks in the order of arrival to two partitions will create one partition with all the difficult tasks (large buildings, high fire), and one with all the easy ones (small buildings, low fire).

Random sampling with uniform allocation (MDP RSUA) relies on the fact that a random sample of a distribution will asymptotically converge to the original distribution with the increase in the size of the sample. We create a common pool of all the tasks, and allocate tasks to the partitions by random sampling. This approach guarantees that with the increase of the size of the partition, the partitions will become more and more representative of the original problem, and, in consequence, more and more similar to each other. As we have seen before, the size of the sample is limited by our ability to solve the resulting MDP.

The resulting approach is fast, fully automatic and it does not require domain specific knowledge. This approach has been taken in [12].

4.4 Partitioning with the reuse of the lower level MDP (MDP REUSE)

We can try to improve on MDP RSUA by:

- Intervening into the selection of the tasks which go into the partition (which will not be random)
- Intervening into the allocation of the resources on the subproblem (which will not be uniform)

MDP REUSE is based on the insight that the high level problem (assigning agents to the partitions) has analogies with the low level problem (assigning agents to tasks). The analogy can be made better if we create the partitions in such a way that they mirror the relative difficulty of specific tasks. In this case we can reuse the low level, optimally solved MDP for the high level partitioning task. Our approach will comprise the following steps:

(S1) Create N partitions of k/N tasks which each resemble, as closely as possible, the profile of an independent task.
(S2) Create an artificial problem for where the tasks are $T_1^* \ldots T_N^*$ where T_i^* is the state most representative for the partition i, while the number of agents is $k^* = k/N$.
(S3) Solve the resulting problem $CF(k^*, N)$. Assume that the resulting coalition sizes are $\langle c_1 \ldots c_N \rangle$, with $\sum c_i = k^*$.
(S4) To each partition i created in step S1 allocate $N \cdot c_i$ agents.
(S5) Solve the resulting subproblems and perform the allocations of coalitions to individual tasks accordingly.

For instance, if we have 100 tasks of the state LARGE_HIGH_FIRE, and 50 tasks of the state MEDIUM_MEDIUM_FIRE, with 300 agents available, we can create a representative problem of the type $CF(\langle$LARGE_HIGH_FIRE, LARGE_HIGH_FIRE, MEDIUM_MEDIUM_FIRE$\rangle, 6)$. If this leads to a solution $\langle 1, 1, 4 \rangle$, we will allocate 50 agents to the subproblems of 50 LARGE_HIGH_FIRE tasks, another 50 agents to another subproblem of 50 LARGE_HIGH_FIRE tasks, and 200 agents to a subproblem of 50 MEDIUM_MEDIUM_FIRE tasks. The resulting subproblems can be partitioned again if they are too large to be solved directly.

The problem, unfortunately, is that in the general case, the collections of tasks to solve will not be aligned so conveniently for partitioning into representative subproblems. We need domain specific information about the relative difficulty of the tasks. An intuitive and relatively simple way to represent this information is to require the practitioner to define a

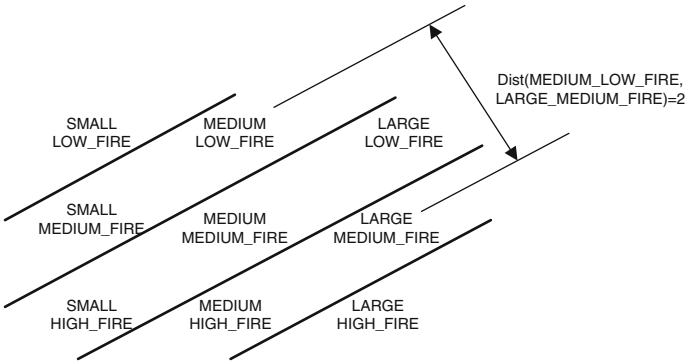


Fig. 5 Defining a distance function between the states of the individual tasks. This function is specific to the firefighter problem, other application domains might need different approaches

distance function with respect to the difficulty of states as part of the Step 1 of the methodology. We try to find groups of states which are of similar difficulty, then sort these groups in the order of difficulty. For instance, we can assume that the increase in the fire level represents (roughly) a similar resource requirement like the increase in the size of the building. With these assumptions, we divide the 9 states into 5 zones according to the Fig. 5. This will define a discrete distance function between states, for instance $dist($ MEDIUM_LOW_FIRE, LARGE_HIGH_FIRE $) = 3$.

Using this distance function, our algorithm first sorts the current tasks into groups based on their states. From the groups with the largest membership, it creates the initial set of N clusters (if necessary, splitting some of the clusters). Then, it assigns each of the remaining tasks to the closest incomplete cluster (based on the distance function defined by Fig. 5).

5 Simulation study

In the following, we describe the results of a simulation study performed using the Yet Another Extensible Simulator (YAES) [3]. The evolution of the burning buildings and the actions of firefighters were simulated using the fire simulator component of the Robocup Rescue framework which was designed to simulate a realistic physical model of the heat development and heat transport in urban fires [15].

The simulations were run with 100 buildings of random sizes. Each simulation was run for 100 cycles where a simulation cycle was treated as being equivalent to the time it takes a firefighter to move from the water source to the building, sprinkle the water on the building and go back to the water source to refill its water tank. We also assumed that each firefighter can be re-assigned to a new building at the beginning of each simulation cycle.

At the end of the simulation, a score was obtained based on the state of the buildings. The scoring mechanism was similar to the one used by the Robocup Rescue simulation environment. If A_i represents the total area of the building i, B_i represents the unburnt area of the building i and N_p is the number of buildings in state S_p then the score for state S_p is computed as follows:

$$score(S_p) = 100 \times \frac{\sum_{l=1}^{N_p} B_l}{\sum_{l=1}^{N_p} A_l} \tag{7}$$

where, for a given building i which is at fire level F_i, the unburnt area B_i is computed as shown in Eq. 3.

Under disaster response conditions, it is unrealistic to expect that every fire will be attended to as soon as it starts. To simulate more realistic response times, a certain number of buildings in the simulation were "preburnt" for a random amount of time. Preburning simply means letting the fire model evolve for a predetermined amount of time with no firefighter assigned to the building.

The simulation was repeated 100 times with different initial conditions. We plotted the average values and the 95% confidence interval. The number of firefighter units ranged from 25 to 200. To put these numbers in perspective, the Orlando Fire Department has 32 fire trucks, with some of the autonomous suburbs having fire departments with 4–6 trucks. The Orlando metropolitan area has about 2 million inhabitants.

5.1 Algorithms evaluated

We evaluated the approach described in this paper using the two parallel partitioning schemes:

MDP RSUA: Partitioning based on random sampling and uniform allocation—as described in Sect. 4.3. The MDP policy was learnt offline for all the combinations of three building sizes (i.e. SMALL, MEDIUM and LARGE) and with six available firefighters. Under these assumptions, we had to learn policies for 3^3=27 different MDPs where each MDP consisted of 7^3=343 states and 28 possible coalition actions from each state. It took about eight to ten hours to learn all the policies on a 2.66 GHz Pentium 4 machine with 1 GB RAM using the value iteration algorithm. During the real-time execution the decision time was approximately 10 ms/decision.

MDP REUSE: Partitioning with the reuse of the lower level MDP—as described in Sect. 4.4. The same precomputed MDP library was used as in the MDP RSUA case. The real-time decision time was, again, approximately 10 ms / decision.

In order to evaluate the quality of the solution provided by our methodology, we have implemented several approaches based on empirical considerations.

UNIFORM: Allocate a uniform number of firefighters to the fires. If the number of available firefighters is denoted with k, and the number of remaining fires to be allocated with n, then the allocation was made as follows:

- if $k <= n$: Exactly one firefighter was allocated to the first k fires
- if $k > n$: Exactly $\lfloor \frac{k}{n} \rfloor$ firefighter was allocated to the n fires. The remaining $k - \lfloor \frac{k}{n} \rfloor$ firefighters were allocated to first $k - \lfloor \frac{k}{n} \rfloor$ fires.

UNIFORM RANDOM: Allocate firefighters uniformly to randomly selected fires. At each simulation cycle, the algorithm selected one of the firefighters and allocated it to a randomly selected fire. This process was repeated until all available firefighters had been allocated.

CLUSTERED RANDOM: Allocate multiple firefighters to randomly selected fires. At each simulation cycle, the algorithm selected a random number of firefighters between 1 and 4 and allocated them to a randomly selected fire. This process was repeated until all available firefighters had been allocated.

HEURISTIC: Allocate firefighters based on the area of the building. At each simulation cycle, the algorithm allocated 2 firefighters to small, 3 firefighters to medium and 4 firefighters to large buildings. These numbers have been chosen in such a way that this is the smallest allocation which guarantees that the fire will be put out at its current level. The buildings have been considered in arrival order. This process was repeated until all available firefighters had been allocated.

Our intuition tells us that HEURISTIC and UNIFORM are "sensible" approaches which can potentially be implemented by a fire station commander. While the UNIFORM approach tries to allocate at least some firefighters to every fire, the HEURISTIC approach attacks each fire with exactly the necessary firefighting power (at the cost of leaving some fires initially not attended). The UNIFORM RANDOM and CLUSTERED RANDOM might model a situation where individual firefighters or groups are wandering randomly and picking fire fighting tasks based on reports without the ability to coordinate with each other. These approaches are not intuitively "sensible" from the point of view of a commander, but their performance compared with the other approaches can yield some useful insights.

5.2 Results

In the following, we present and interpret the results of our experiments. The maximum attainable score is 75, because every building is initially put on fire and the best possible outcome is to extinguish all of them at their initial fire level (as shown in the scoring mechanism in Eq. 7).

Figure 6 shows the results with 25 firefighters. This is a very difficult problem, with only one firefighter for every 4 buildings on fire. Unsurprisingly, the scores obtained by all approaches are very low. There are two clearly separated groups of strategies: the MDP RSUA, MDP REUSE and HEURISTIC approaches obtained scores around 14–18, while the UNIFORM, UNIFORM RANDOM and CLUSTERED RANDOM obtained scores around 2–4.

There is no clear advantage of MDP REUSE over MDP RSUA, in fact the latter one obtained marginally better results for high preburn. Neither the advantage of the two MDP-based approaches over the heuristic approach is significant at this setting.

One interesting phenomena we see in this graph is that the performance of MDP REUSE, although starting up the highest, decreases at a higher rate than all the other approaches with

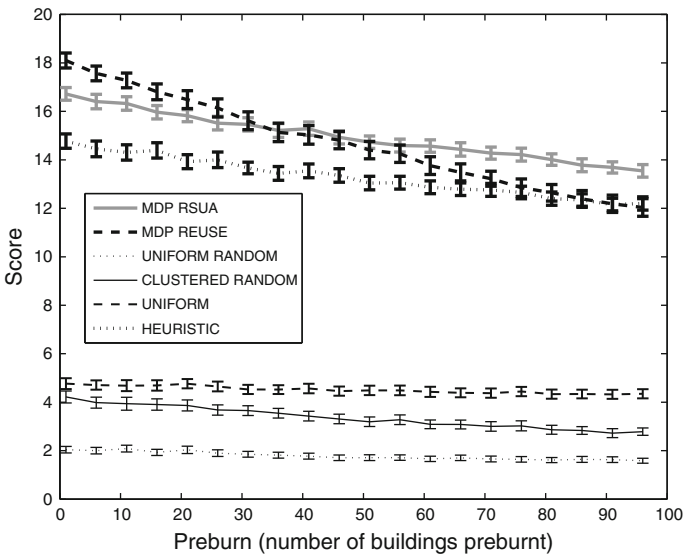


Fig. 6 Evaluation of coalition formation strategies to extinguish 100 fires with 25 firefighters

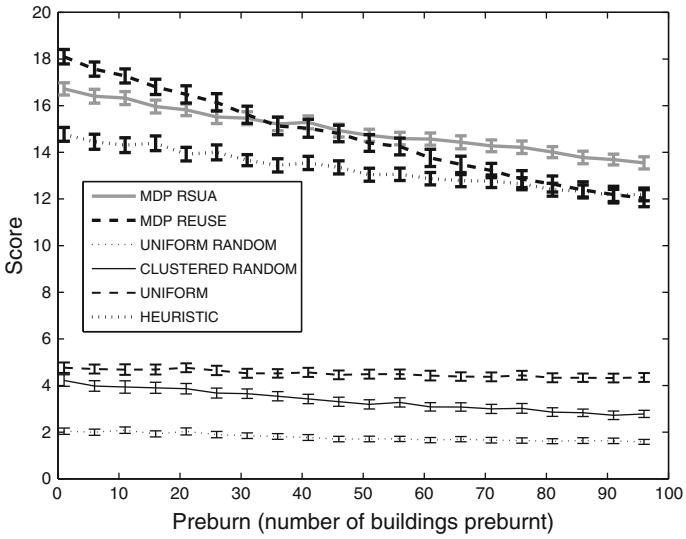


Fig. 7 Evaluation of coalition formation strategies to extinguish 100 fires with 50 firefighters

the increase of the preburn value, and, at around 35% preburn it actually dips below MDP RSUA. The reason for this behavior is that MDP REUSE relies on finding a representative task for its partitions. This is easy when all the buildings are on low fire (choose a representative which is also on low fire and is the same size). With preburn the possible states of the tasks become highly varied, thus finding good representative tasks becomes more difficult and the approximation errors are lowering the performance. For MDP RSUA this is not a factor, because its method of creating partitions by random sampling is not affected by the diversity of the task pool. Thus, MDP RSUA, just like HEURISTIC and the other approaches, declines only at the rate at which the preburn makes the problem inherently more difficult.

In the low score group, the random approaches fare the worst, because they reallocate the firefighters between turns. With only 1 firefighter to 4 buildings, if the firefighter is moved from building to building, eventually almost all the buildings will burn down. CLUSTERED RANDOM might allocate more than one firefighter to a building, which makes it more probable that the fire will be put out. The UNIFORM approach, by keeping one firefighter consistently at the same set of buildings, scores the best here, as there is some probability that a fire at a small building can be put out with one firefighter.

Figure 7 shows the results with 50 firefighters. With more firefighters to work with, the average score is higher, but the relative performance of the approaches changes. The MDP REUSE is now clearly the best with up to 10% relative increase over MDP RSUA. As before, MDP REUSE decreases faster with the preburn than the other approaches, but with 50 firefighters it shows the best performance throughout the range. Its advantage, however, diminishes with the increase in the preburn.

The HEURISTIC approach is still in the upper group. In the bottom pack there is an interesting shift in the relative performance: the CLUSTERED RANDOM approach clearly outperforms UNIFORM and UNIFORM RANDOM. With a larger number of firefighters, the CLUSTERED RANDOM approach which can put up to 4 firefighters to a single fire has a higher chance of extinguishing a fire than the more uniform distributions.

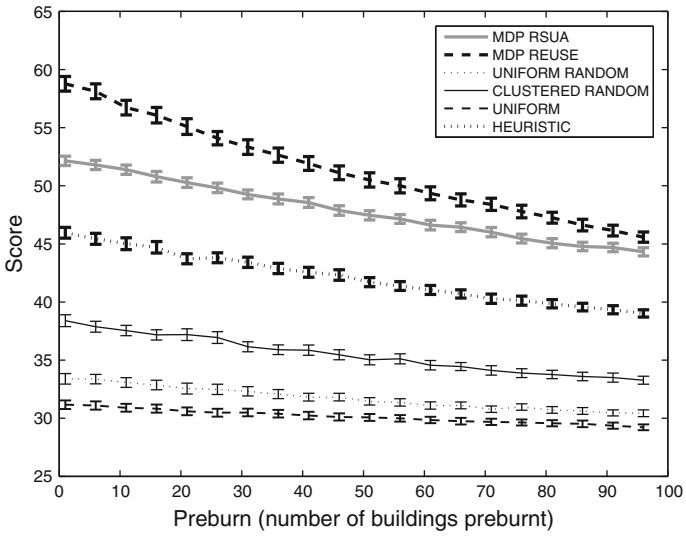


Fig. 8 Evaluation of coalition formation strategies to extinguish 100 fires with 100 firefighters

Figure 8 shows the results with 100 firefighters, and it continues the trends of the previous figures. The advantage of MDP REUSE over MDP RSUA becomes more pronounced and the HEURISTIC approach falls behind the MDP approaches. In the bottom half, the CLUSTERED RANDOM approach improves its lead over the other two approaches, and UNIFORM RANDOM becomes better than UNIFORM. This surprising reversal is due to the fact that the exactly one firefighter for a fire allocation, which is what the UNIFORM approach will do here, is quite bad—it can put out only low fires on small buildings, and with a non-zero probability that the fire will increase. On the other hand, allocating a firefighter to *every* building will slow down the burning of the building, preventing the firefighter from moving to another building. Due to this phenomena, the "sensible" uniform allocation is actually the worst one for this scenario.

Finally, Fig. 9 with 200 firefighters marks a milestone in the sense that the MDP REUSE approach, at 0 preburn closely approximates the theoretical maximum score of 75. Note that this does not mean that it performs an optimal allocation: a better algorithm, for instance, based on the exact solution of the original, unpartitioned MDP might be able to reach the same performance with a lower number of firefighters, or maintain a higher performance for higher preburn values. As expected, MDP RSUA is lower, but MDP REUSE's advantage decreases for high preburn.

An interesting phenomena is that while the HEURISTIC approach is the third best for low preburns, for higher preburn values CLUSTERED RANDOM takes over, becoming better than the "sensible" HEURISTIC approach! HEURISTIC looses performance with higher preburn because it takes into account only the size of the building, not its level of fire. While its allocation always successfully extinguishes the fire, for higher fire levels it can take a long time, during which some fires will be unattended. Note that even in this comparatively easy scenario, there are not enough firefighters for HEURISTIC to allocate firefighters to every fire in the first round. In contrast, CLUSTERED RANDOM, which allocates a higher number of firefighters, although to randomly chosen fires, is able to extinguish fires more rapidly and move over to the next.

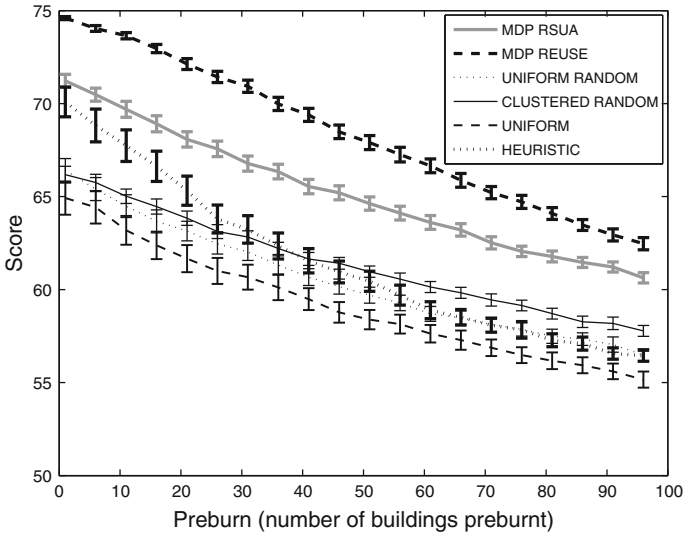


Fig. 9 Evaluation of coalition formation strategies to extinguish 100 fires with 200 firefighters

Let us now analyse some of the observed phenomena. Clearly, the MDP-based approaches outperform the other ones in every case—the differences are significant and consistent. There are significant differences between the two partitioning methods as well, but overall, the MDP REUSE has a greater advantage when it has more "freedom of operation", that is, when we have more firefighters and lower preburn.

The low performance of the UNIFORM and UNIFORM RANDOM approaches, and the relatively high performance of the CLUSTERED RANDOM can be explained by the properties of the problem domain. There is a race between the extension of the fire in the building and the extinguishing efforts of the firefighters. If the number of firefighters is lower than a minimum, the building will completely burn out (albeit slower). The result is that spreading the forces too thin is one of the worst choices one can make—and this is exactly what the UNIFORM and UNIFORM RANDOM approaches are doing. Forming larger teams is a better choice even if they are deployed at random buildings (the case of the CLUSTERED RANDOM approach).

Finally, the advantage of the two proposed approaches is evident not only from their higher performance, but also by the fact that they consistently were the first and the second, while all the other approaches went through a number of reversals for different scenarios. Even hand crafted approaches which perform well on a range of scenarios might show an unexpected dip on a relatively easy scenario—as shown by the case of HEURISTIC for 200 firefighters and high preburn.

6 Conclusions

In this paper we developed a methodology for optimizing coalition formation for execution of tasks which evolve in time, respond nondeterministically to the actions of the agents, and the execution reward changes in time. The methodology relies on the creation of a factored MDP, which then can be solved using the well-known MDP solving methods. As the resulting MDP

will likely be too large for any realistic problem size, we described two approaches for speeding up the computation using parallel partitioning of the resulting MDP. The MDP RSUA approach does not require any domain specific information. The MDP REUSE approach relies on the reuse of the low level MDP at the partitioning step, and requires the user to define a distance function with respect to difficulty in the state space of the tasks.

An experimental study confirms that both presented approaches scale well for hundreds of tasks and agents, and significantly outperform heuristic approaches. As expected, the MDP REUSE approach performs better than MDP RSUA, the difference being especially significant for scenarios with larger number of agents.

Acknowledgements This research was sponsored in part by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-06-2-0041. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation heron.

References

1. Ågotnes, T., van der Hoek, W., & Wooldridge, M. (2006a). On the logic of coalitional games. In: *5th International joint conference on autonomous agents and multiagent systems (AAMAS 2006)*, Hakodate, Japan, May 8–12, 2006, pp. 153–160.
2. Ågotnes, T., van der Hoek, W., & Wooldridge, M. (2006b). Temporal qualitative coalitional games. In: *5th International joint conference on autonomous agents and multiagent systems (AAMAS 2006)*, Hakodate, Japan, May 8–12, 2006, pp. 177–184.
3. Bölöni, L., & Turgut, D. (2005). YAES—a modular simulator for mobile networks. In: *Proceedings of the 8-th ACM/IEEE international symposium on modeling, analysis and simulation of wireless and mobile systems MSWIM 2005*.
4. Boutilier, C., Dean, T., & Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research, 11*(1), 94.
5. Chalkiadakis, G., & Boutilier, C. (2004). Bayesian reinforcement learning for coalition formation under uncertainty. In: *Proceedings of the Third international joint conference on autonomous agents and multiagent systems*, Vol. 3, pp. 1090–1097.
6. Chalkiadakis, G., & Boutilier, C. (2008). Sequential decision making in repeated coalition formation under uncertainty. In: *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems. International Foundation for Autonomous Agents and Multiagent Systems*, Vol. 1, pp. 347–354.
7. Ebden, M., Briers, M., & Roberts, S. (2008). Decentralized predictive sensor allocation. In: *47th IEEE Conference on decision and control, 2008*. CDC 2008, pp. 1702–1707.
8. Goranko, V. (2001). Coalition games and alternating temporal logics. In: *TARK '01: Proceedings of the 8th conference on theoretical aspects of rationality and knowledge*, pp. 259–272.
9. Guestrin, C., Koller, D., & Parr, R. (2002). Multiagent planning with factored MDPs. *Advances in Neural Information Processing Systems, 2*, 1523–1530.
10. Guestrin, C., Koller, D., Parr, R., & Venkataraman, S. (2003). Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research, 19*(10), 399–468.
11. Hoek, W., & van der Wooldridge, M. (2005). On the logic of cooperation and propositional control. *Artif Intell, 164*(1–2), 81–119.
12. Khan, M., Turgut, D., & Bölöni, L. (2008). Optimizing coalition formation for tasks with dynamically evolving rewards and nondeterministic action effects. In: *Proceedings of international workshop on optimisation in multi-agent systems (OptMas08), in conjunction with the seventh joint conference on autonomous and multi-agent systems (AAMAS 2008)*, pp. 69–76.
13. Klusch, M., & Gerber, A. (2002). Dynamic coalition formation among rational agents. *Intelligent Systems, 17*(3), 42–47.
14. Meuleau, N., Hauskrecht, M., Kim, K., Peshkin, L., Kaelbling, L. P., Dean, T., et al. (1998). Solving very large weakly coupled markov decision processes. In: *Proceedings of the fifteenth national conference on artificial intelligence*, pp. 165–172.

15. Nüssle, T. A., Kleiner, A., & Brenner, M. (2004). Approaching urban disaster reality: The resQ firesimulator. In: Nardi D, Riedmiller M, Sammut C, Santos-Victor J (Eds.), RoboCup 2004: Robot Soccer World Cup VIII, Springer, Lecture Notes in Computer Science, Vol. 3276, pp. 474–482.
16. Osborne, M., & Rubinstein, A. (1994). *A Course in Game Theory*. MIT Press.
17. Shehory, O., & Kraus, S. (1998). Methods for task allocation via agent coalition formation. *Artificial Intelligence, 101*(1–2), 165–200.
18. Suijs, J., Borm, P., Waegenaere, AD., & Tijs, S. (1999). Cooperative games with stochastic payoffs. *European Journal of Operational Research, 113*(1), 193–205.