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

Majid Ali Khan, Damla Turgut and Ladislau Bölöni
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816–2450
Email: khan@bond.cs.ucf.edu,turgut@eecs.ucf.edu, lboloni@eecs.ucf.edu

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 into 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: typically, the more delay in the execution, the lesser the reward. We exemplify this class of problems by the allocation of firefighters to fires in a disaster rescue environment. We describe a practical methodology through which the aspects of this problem can be encoded as a Markov Decision Process. An experimental study involving the Robocup Rescue simulator shows that a coalition formation policy developed following our methodology outperforms heuristic approaches.

1. INTRODUCTION

We consider a problem domain where a group of agents $\{A_1, \ldots A_k\}$ 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. We will call these tasks Dynamic Nondeterministic Tasks (DNT). Our goal is to find the optimal allocation of the agents in the coalitions; this might involve the reorganization of the coalitions during task execution as well. We assume, however, that once a coalition is formed and assigned to a task, the agents in the coalition will chose the best possible collaborative action in order to further the execution of the task.

The running example which we use 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 sufficient 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. There is an uncertainty in the result of the firefighting action: the same amount of water might or might not put out a fire, due to some unknown parameters of the building (nondeterminism).

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 a consistent methodology for encoding this class of problems as an optimization problem, which can then yield an appropriate coalition formation policy. We also provide a discussion of the approaches through which the method can be scaled for larger problems, in exchange for a loss of accuracy. The approach relies on the encoding of the problem as a Markov Decision Process (MDP) [7] in such a way that the agent actions normally considered in MDPs are replaced with coalition configurations.

One of our goals was to guarantee a practically usable methodology. For this goal, we had to avoid using some theoretically appealing but practically not useable techniques such a POMDPs. For instance, a convenient way to model the uncertainty of the firefighter actions might be as an uncertainty about the observation of the fuel available in the building. However, this would yield an unpractically high computational complexity.

The remainder of this paper is organized as follows. Section 2 discusses related problem domains. Section 3 describes the steps of the proposed methodology. Section 4 describes the simulation study in which the methodology is applied to the firefighter allocation domain using the Robocup Rescue simulator. We conclude in Section 5.

2. RELATED WORK

The problem considered in this paper is related to several problems extensively studied in computer science as well as game theory, operations research and management sciences.

In game theory, coalition games [1, 2, 6, 10, 13] are games in which agents are forming coalitions in order to increase their *payoff*. If we have a coalition of agents $\{A_{c1}, \ldots A_{cj}\}$ the value of the coalition is $v(\{A_{c1}, \ldots A_{cj}\})$.

Coalition games in stochastic environments (that is, when the payoffs are stochastic) were also considered by game theory researchers [12]. Most of the research in coalition games start with the assumption of a super-additive value function. Under this assumption, usually a grand coalition is formed, and the main challenge remains how to assign the payoff among the participating agents.

One of the fundamental advances for the coalition formation re-

Cite as: Title, Author(s), *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. XXX-XXX.

search in multi-agent environments is due to Shehory and Kraus [11]. They have relaxed the assumption of a super-additive domain, to account for the realities facing practical multi-agent systems. Under these assumptions, the challenges move to the choice of the appropriate coalition operations, as a great coalition will not be, in general, optimal.

One trend of multi-agent coalition research, exemplified in the work of Blankenburg, Klusch and others [3–5], centers around the game-theoretic concept of a *kernel*.

Klusch and Gerber [8] outline a research agenda for the development of coalition formation algorithms in a distributed environment (the Distributed Coalition Formation problem). They also outline a customizable algorithm framework DCF-S, which relies on the coalition leading agent pre-simulate to determine valid coalitions before starting negotiations. The paper also identifies the need for developing coalition algorithms for stochastic and fuzzy domains and calls for research on those directions.

3. A METHODOLOGY FOR OPTIMIZING COALITION ALLOCATION FOR DNT PROBLEMS

In the following we describe a general approach for solving DNT problems. The method is composed of 7 steps and it is described in Table 3. Note, that the most complex parts of the approach are concerned with the building of the domain models and the interpretation of the solutions. The solution of the MDP, although it is of a high computational complexity, is relying on well known, standard algorithms.

As with most problems involving real world settings with MDPs, there is always a danger of the combinatorial explosion of the number of states and actions. The developer needs to use domain specific knowledge to exploit simplification opportunities to maintain the MDP at a manageable size.

Table 1: The steps of the general approach to the solution of the DNT problem.

Step 1	Develop a stochastic model for the evolution of the task in time.
Step 2	Develop a model for 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 one or more Markov decision processes modeling the problem domain.
Step 6	Solve the MDP(s) using the appropriate algorithms (value iteration, policy iteration, hybrid methods)
Step 7	Interpret the MDP(s). Assemble the policy for the agents from the solutions of the MDP(s).

In the following we describe the steps of our approach in greater detail. For the presentation of every step, we first discuss the general approach, and then we illustrate it using the firefighter domain. The same firefighter application will be used in the experimental evaluation of the approach.

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. In the DNT domain 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. We assume that the value can be one of the discrete values $M(T,t) \in \{m_1, m_2 \dots m_q\}$.

The choice of the number of discrete values is technically limited by the accuracy of the measure. In practice, however, we might choose to represent a lower 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:

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

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 our example scenario, 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.

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 building extinguished after being on medium fire.
HIGH-FIRE	The building has been on fire for a long time.
HIGH-BURNT	The building extinguished after being on high fire.
COMPLETELY-BURNT	The building has been completely burnt.

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 same type. We consider 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 Figure 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 left unattended, the fire will eventually cover and burn the building completely.

.

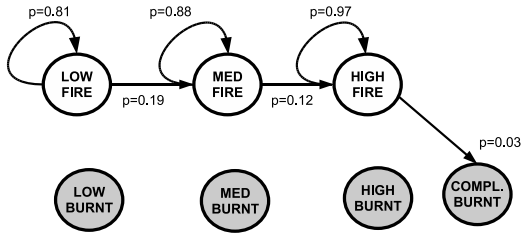

Figure 1: The transition probabilities for a single small building. The terminal states are gray. Note that some of the states are not reachable.

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 $A_1 \dots A_k$ are acting in a coalition towards achieving a task T, each agent A_i chooses an individual action a_i. The composition of the individual actions forms a *coalition action* $ca = \{a_1 \dots a_k\}$. 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 our example scenario, 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 means 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.

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 $CA = \{A_1, \dots A_k\}$ is acting on the tasks performing a coalition action $ca = \{a_1 \dots a_k\}$. We are interested in the state of the task at time $t + 1$.

$$M(T, t+1, ca) = \begin{cases} m_1 & \text{with probability } p^T(m_x \xrightarrow{ca} m_1) \\ \dots & \\ m_q & \text{with probability } p^T(m_x \xrightarrow{ca} m_q) \end{cases} \quad (2)$$

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.

In our example, 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 2 (continuous lines) or 3 (dotted lines) firefighters. We did not include actions with 1, 4 or more firefighters for the purpose of clarity. Comparing with Figure 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 becomes unreachable, 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 the DNT problem assumes a *dynamic* formation of the coalitions, this is possible.

The full collection of transmission probabilities for the case of a small building and coalitions of at most 4 agents is shown in Table 3.

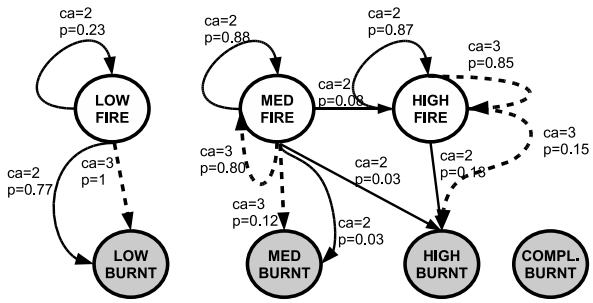

Figure 2: The transition probabilities for a single small building as the effects of the coalition actions of 2 (continuous lines) or 3 (dotted lines) firefighters. The terminal states are gray.

.

3.4 Step 4: Develop a cost and reward model

Until now we did not consider neither 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 DNT model, there can be several terminal states. We need to associate a certain reward with each of the states. The reward expresses the relative preference over the states and also the relative preference over the various states of the individual tasks.

In addition to the rewards associated with the states, we can also have costs associated with the actions. These costs allow us to express preferences over various actions with equivalent or similar effects.

Table 3: Transition probabilities for small building with four firefighters. 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

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

For our example we define a reward for the terminal states in the following way.

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

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.

Thus, we define the reward of the firefighting as the area of the buildings saved. As the firefighters have a single possible action (i.e. firefighting), we define a cost for a collaborative action to be proportional with the number of firefighters participating in the coalition. Thus, in general we will prefer smaller coalitions, as long as the achieved terminal state is identical.

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. The components rely on the models developed previously, still, as we will see, assembling the MDP requires non-trivial technical decisions.

(a) 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 k tasks is a k-tuple of the discrete states of the individual task: $gm = (m^{(1)}, \dots m^{(k)})$ where $m^{(i)} \in \{m_1 \dots m_q\}$. There are k^q number of possible states.

(b) 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 ga is thus defined as:

$$ga = \{ca_1 \dots ca_k\} \text{ where } ca_i = \{a_{i1} \dots a_{in_i}\} \quad (4)$$

where n_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 k tasks and m agents, the number of coalitions can be calculated by considering that each agent chooses independently which task will it work on. In this case the number of possible coalitions is k^m. If we assume that the agent might also choose not to be part of any coalition, the number of alternatives becomes $(k+1)^m$.

Naturally, we need to find ways to reduce the number of potential coalitions and actions. This can be done by exploiting application specific features. For our example, we have two specific properties of the domain which allow us to drastically reduce the number of the actions: (i) we assume that all the firefighter agents are equivalent and interchangeable and (ii) the only action of the agent is firefighting. We can thus represent a global action ga only with the number of agents participating in each coalition:

$$ga = \{n_1 \dots n_k\} \text{ where } \sum_{i=1}^{k} n_i = m \quad (5)$$

The possible coalitions under these assumptions can be generated by the possible *integer partitions* of m into k places. One such recursive algorithm for partitioning m agents to k tasks is presented in Algorithm 1.

Algorithm 1 A recursive algorithm for integer partitioning

```
IntegerPartition(m, k)
    C ← An empty list of the list of integers
    if (k = 1)
        L ← An empty list of integers
        Add m to the list L
        Add the list L to the list C
        output C
    for i ← 0 to m
        for each list P ∈ IntegerPartition(m − i,k − 1)
            Append i to the front of the list P
            Add the list P to the list C
    output C
```

(c) 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.

If the system is in a global state $gm_x = (m_x^{(1)}, \dots m_x^{(k)})$, and the global action performed is $ga = \{ca_1 \dots ca_k\}$, the probability to transition to a new global state $gm_y = (m_y^{(1)}, \dots m_y^{(k)})$ will be:

$$p\left(gm_x \xrightarrow{ga} gm_y\right) = \prod_{i=1}^{k} p^{(i)}\left(m_x^{(i)} \xrightarrow{ca_i} m_y^{(i)}\right) \quad (6)$$

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 maintain the graphs readable, we will simplify the firefighting model to a model where each building can be in only

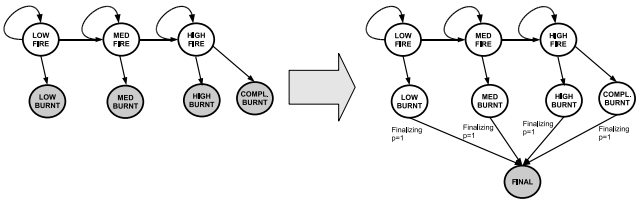

Figure 4: Transformation of the MDP to allow for the expression of a reward for reaching a terminal state. A new, single terminal state FINAL is introduced. The previous terminal states are becoming non-terminal, and a special action Finalizing moves from them to the FINAL state with the probability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

$k = 3$ 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 $k^2 = 9$ states. The possible team actions, in each state are two: $(1, 0)$ representing that the firefighter is allocated to the first building, and $(0, 1)$ representing that the firefighter is allocated to the second building.

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 $R(s_i, a)$ or with a certain transition taken as a result of an action $R(s_i, a, s_j)$. 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 of 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 the case of our model, we are concerned with the cost of the actions and the reward to reaching one of the preferred terminal states. The cost of the action can be represented easily, by replicating it for all the current states where the actions can be taken. However, there is a problem with the rewards for the terminal states: 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 previous 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 lead to 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.

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

3.6 Scalability issues

The methodology discussed here, will find the optimal coalition formation policy, given the available information. Unfortunately, the methodology creates very large and "dense" MDPs, with a very large number of transitions possible at every state. This limits the scalability of the approach. In the following we discuss some methods through which the scalability of the methodology can be improved by exploiting domain knowledge.

Pre-filtering undesireable coalitions

From the offset, there might be certain types of coalitions which are not desirable. One example are soft or hard limits on the number of certain types of agents in a coalition. For instance, we might have a policy that every firefighter team can include at most one trainee. Although these types of soft or hard constraints can be encoded in the reward function, it much more efficient to filter out these type of coalitions from the beginning.

Another example is when we are using our application domain knowledge to predict the results and remove coalition setups which are not going to be optimal. For instance, if we know that one of the resources is in short supply (for instance, a fire truck with chemical fire extinguishing agent), we can remove those coalition configurations where the agent in short supply is part of no coalition.

Coalescing equivalent coalitions

Another alternative is to coalesce coalitions which are equivalent, or differ very slightly. We might be able to identify coalition actions which are closely resembling each other and whose utility will be also closely related. We can replace these coalitions with a representative coalition, and obtain an approximate solution.

Iterative refinement

One of the causes of the "density" of the MDPs created by our methodology is the set of coalition configurations which differ very little from each other, but appear as transitions in the MDP. An iterative refinement method can be applied as follows. We perform a clustering over all the possible coalition configurations and choose a representative for each cluster. We first solve an MDP which contains only the cluster representatives. We will find that in the resulting policy, some of the cluster representatives will not be present. In the next step, we add back to the MDP the cluster members of those representatives which are part of the optimal policy for a certain task. We solve the resulting MDP.

Partitioning through uniform sampling

Let us assume that we have 40 tasks of type A and 60 tasks of type B, and 20 agents. The idea of partitioning through uniform sampling is that instead of solving a problem with 100 tasks and 20 agents, we solve two problems with 30 tasks of type B, 20 of type A and 10 agents. For this method to be applicable, there are a number of conditions which need to be met:

- There are multiple, equivalent classes of tasks.
- There are multiple, equivalent classes of agents.
- The remaining number of agents of each type after the partition significantly exceeds the saturation level for a single task. The saturation level for a task is the number of agents of a certain type which has the property that adding additional agents of a given type does not improve the execution type of the task.

The partitioning needs to be done in such a way that the ratio of different task types and classes are kept constant. In many cases these conditions are satisfied only in approximation. In our case, for instance, there is no "full" saturation level of the number of firefighters for a single task - adding extra firefighters always increases the probability that the fire will be put out in the next time interval. However, as the size of the coalition increases, the benefit of the extra firefighter is decreasing and a saturation level can be assigned empirically.

4. SIMULATION STUDY

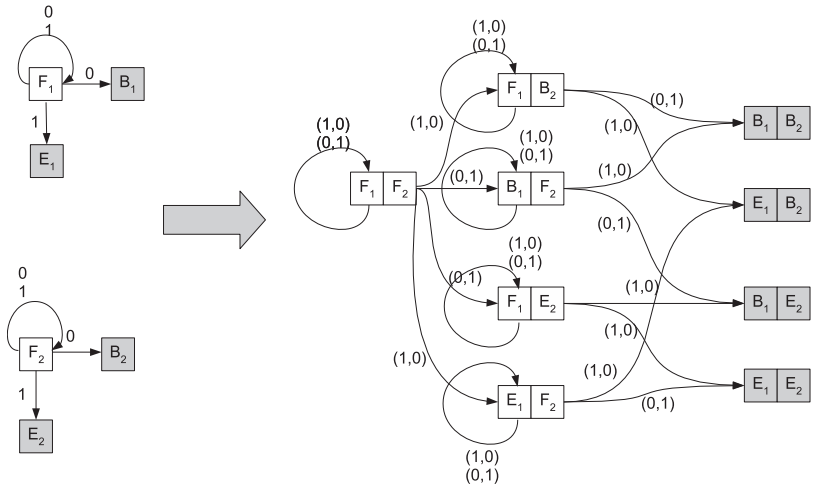

Figure 3: Creating an MDP for a simplified version of the firefighter problem with only three states: Fire (F), Burned (B), and Extinguished (E). The final MDP is assembled for the case of one firefighter and two buildings.

In the following, we describe the results of a simulation study. The fire simulation was performed using Robocup Rescue fire simulator which is designed to simulate a realistic physical model of heat development and heat transport in urban fires [9].

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 took for a firefighter to move from the water source to the building, sprinkle the water on the building and come back to the water source to refill its water tank. We also assumed that each firefighter can be re-assigned to new a building at the beginning of each simulation cycle.

At the end of each 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:

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

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

As the problem size (100 tasks) was large, we used the partitioning through uniform sampling method discussed in the previous chapter. 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.

Under disaster response conditions, it is unrealistic to expect that every fire will be attended to as soon as it started. To simulate more realistic response times, a certain number of buildings in the simulation were "pre-burnt" for a random amount of time. Pre-burning simply meant 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 report the average score and the 95% confidence interval.

4.1 Baseline algorithms

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

UNIFORM: Allocate equal number of firefighters to the fires. In this approach, the available firefighters were allocated uniformly to the fires. If the number of available firefighters is denoted with m, and the number of remaining fires to be allocated with n, then the allocation was made as follows:

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

UNIFORM RANDOM: Allocate firefighters uniformly to randomly selected fires. In this approach, 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. In this approach, 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. In this approach, at each simulation cycle, the algorithm allocated certain number of firefighters to the selected fire based on the size of the building. The small building was allocated 2, medium building was allocated 3 and were the large building was allocated 4 firefighters. This process was repeated until all available firefighters had been allocated.

4.2 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

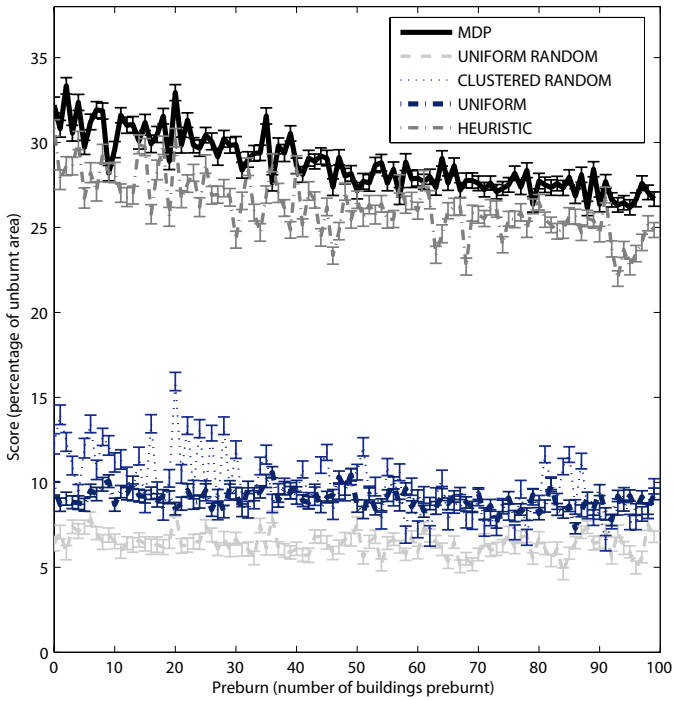

Figure 5: Evaluation of coalition formation strategies to extinguish 100 fires with 50 firefighters.

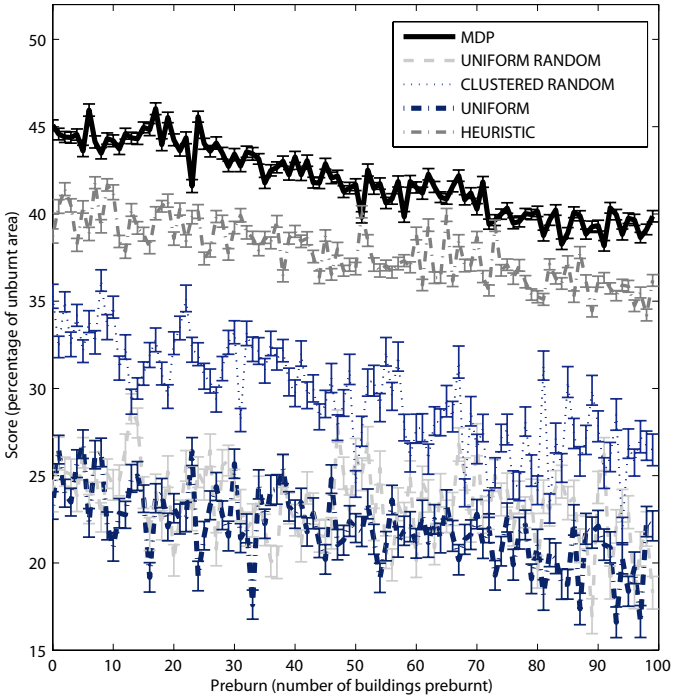

Figure 6: Evaluation of coalition formation strategies to extinguish 100 fires with 75 firefighters.

extinguish all of them at their initial fire level (as shown in the scoring mechanism in Equation 7).

Figure 5 shows the results with 50 firefighters, Figure 6 with 75 firefighters, while Figure 7 with 125 firefighters.

The MDP based allocation approach clearly performed better than all the other approaches, especially with the increased number of available firefighters. The HEURISTIC approach performed reasonably well with small number of firefighters. This was because: **a)** even with optimal firefighter allocations with MDP, many buildings could not be extinguished with fewer number of available firefighters and **b)** the heuristic was quite conservative in its allocations.

However, the difference between MDP and the heuristic based approach grew with the increased number of firefighters. This is because with large number of available firefighters, the MDP based approach could allocate the optimal number of firefighters to the key fires, while the heuristic allocated conservatively and thus resulted in not being able to extinguish some of the key fires. An example of such non-optimal allocation would be with a state (SM-MED, MD-LOW), which specifies a small building on medium fire level and a medium building on low fire level. If there were 5 firefighters available for allocation, the heuristic based approach would allocate (2,3), i.e. two firefighters to the small building and three firefighters to the medium building. This allocation might results in a state of (SM-MED-BURNT, MD-MED-BURNT) after the firefighters have worked on the fires for some amount of time. With MDP, the optimal allocation would be to first extinguish the medium building on low fire (i.e. (0,5) allocation); which can be achieved quite quickly. Later on, the MDP would allocate all the firefighters to the small building, which might have increased to the next fire level by that time. This would result in a state like (SM-HIGH-BURNT, MD-LOW-BURNT). Now, since the small

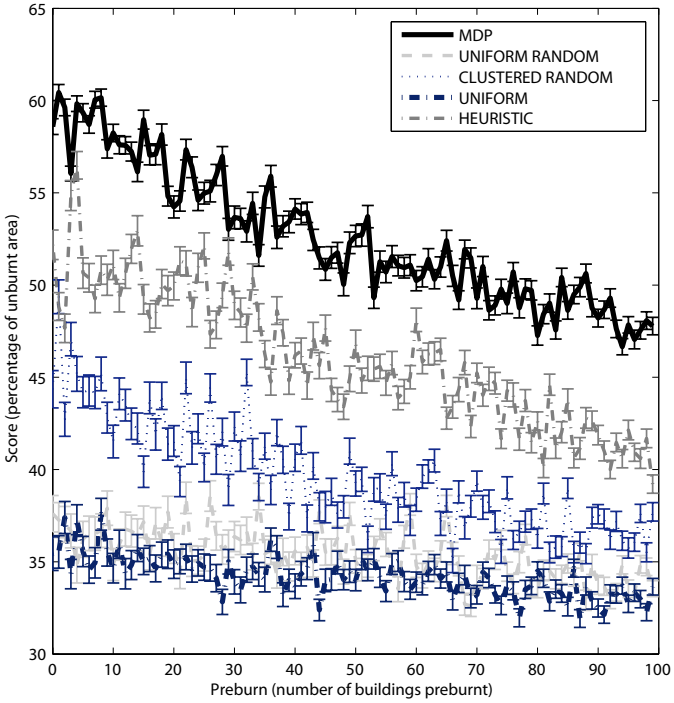

Figure 7: Evaluation of coalition formation strategies to extinguish 100 fires with 125 firefighters

building is 1,000 sq ft and medium building is 2,000 sq ft, the reward obtained by the heuristic would be $((1/2 \times 1000 + 1/2 \times 2000)/3000) \times 100 = 50$. The MDP allocation results in a higher reward of $((1/4 \times 1000 + 3/4 \times 2000)/3000) \times 100 = 58.3$.

The UNIFORM, UNIFORM RANDOM and CLUSTERED RANDOM approaches, in general, performed significantly worse. For a small number of firefighters, the worst approach appears to be UNIFORM RANDOM, which allocated a uniform number of firefighters to randomly selected fires at each simulation cycle. Since buildings usually require more than one firefighter to extinguish them, it performed worse than the CLUSTERED RANDOM approach which allocates a group of firefighters to the fires. Also, since it does not focus on certain buildings (i.e. it keeps changing the allocation at every simulation cycle), it performs worse than the UNIFORM approach which keeps allocating the same number of firefighters to the same set of non-extinguished buildings at every simulation cycle.

However, with the increased number of firefighters, the UNIFORM RANDOM approach tends to perform slightly better than the UNIFORM approach. This is because, with increased number of available firefighters, the uniform random allocation would end up allocating multiple firefighters to at least some of the buildings. The UNIFORM approach would still allocate equal but small number of firefighters to each fire and thus result in worse performance.

Another interesting observation is the improved performance of CLUSTERED RANDOM approach with the increased number of firefighters. This again highlights the fact that it is better to allocate a group of firefighters to some fires and extinguish them early rather than keep allocating equal but small number of firefighters to all the fires. Since fires become more difficult with the passage of time, equal allocation results in multiple fires that keep getting worse (because of less than required number of firefighters allocated to them). The cluster approach performs better since by allocating multiple firefighters to some fires, those particular fires can be quickly extinguished and the firefighters can then be allocated to other fires.

As expected, the preburning of the buildings causes the attained score to get lower. The impact of preburning on the attained score also gets comparably higher with the increased number of available firefighters. This indicates that there is higher penalty for delayed action when the number of available firefighters is relatively large. We also observe that the effect of preburning is relatively small for uniform allocation approaches even with large number of available firefighters. This can be attributed to the fact that these approaches already result in quite suboptimal allocations and the worsening state of the buildings does not effect their eventual outcome.

5. CONCLUSIONS AND FUTURE WORK

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. We have shown that the algorithm developed following the methodology outperforms a series of heuristics in a simulation study involving firefighter allocation in a disaster environment.

Clearly, the main future challenge is the scaling of the approach for large problems without significant loss in the quality of the generated policy. While we have informally proposed some approaches which increase the scalability of the method, a more rigorous approach for the specification and validation of these approximate methods is part of our future work.

Acknowledgments

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.

6. REFERENCES

[1] T. Ågotnes, W. van der Hoek, and M. Wooldridge. 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*, pages 153–160, 2006.

[2] T. Ågotnes, W. van der Hoek, and M. Wooldridge. Temporal qualitative coalitional games. In *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*, pages 177–184, 2006.

[3] B. Blankenburg, R. K. Dash, S. R. , M. Klusch, and N. Jennings. Trusted kernel-based coalition formation. In *Proc. 4th Intl. Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005)*, pages 989–996, 2005.

[4] B. Blankenburg and M. Klusch. On safe kernel stable coalition forming among agents. In *Proc. 3rd Int. Conference on Autonomous Agents and Multiagent Systems (AAMAS-2004)*, pages 580–587, 2004.

[5] B. Blankenburg, M. Klusch, and O. Shehory. Fuzzy kernel-stable coalitions between rational agents. In *Proc. of the Second Int. Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2003)*, pages 9–16, 2003.

[6] V. Goranko. Coalition games and alternating temporal logics. In *TARK '01: Proceedings of the 8th conference on Theoretical aspects of rationality and knowledge*, pages 259–272, 2001.

[7] R. A. Howard. *Dynamic Programming and Markov Processes*. MIT. Press, 1960.

[8] M. Klusch and A. Gerber. Dynamic coalition formation among rational agents. *Intelligent Systems*, 17(3):42–47, May/Jun 2002.

[9] T. A. Nüssle, A. Kleiner, and M. Brenner. Approaching urban disaster reality: The resQ firesimulator. In D. Nardi, M. Riedmiller, C. Sammut, and J. Santos-Victor, editors, *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *Lecture Notes in Computer Science*, pages 474–482. Springer, 2004.

[10] M. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[11] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1–2):165–200, 1998.

[12] J. Suijs, P. Borm, A. D. Waegenaere, and S. Tijs. Cooperative games with stochastic payoffs. *European Journal of Operational Research*, 113(1):193–205, February 1999.

[13] W. van der Hoek and M. Wooldridge. On the logic of cooperation and propositional control. *Artif. Intell.*, 164(1-2):81–119, 2005.