Periodic Real-Time Resource Allocation for Teams of Progressive Processing Agents

Gilles Steeve Dibangoye
GREYC-CNRS & DAMAS
Laval University
G1K7P4, Quebec, Canada
gdibango@ift.ulaval.ca

Abdel-Illah Mouaddib
GREYC-CNRS
University of Caen, France
Bd. Marechal Juin, Campus II
mouaddib@info.unicaen.fr

Brahim Chaib-Draa
DAMAS Laboratory
Laval University
G1K7P4, Quebec, Canada
chaib@ift.ulaval.ca

ABSTRACT

In this paper, we focus on the problem of finding a periodic allocation strategy for teams of resource-bounded agents. We propose a real-time dynamic algorithm RTDA*, that exploits two key properties to avoid the exponential increase in the state and action spaces associated with multi-agent systems. First, resource allocation at each time period follows an earliest deadline first order (EDF) over agents. Second, the resources are *undivided*, i.e., the resources allocated to an agent restrict their availability to others over time. We can therefore view each incoming agent as a cyclic individual resource-bounded processing, namely *"cyclic progressive reasoning unit"* (C-PRU), and solve, off-line, the single agent resource allocation problem. In the on-line phase, our algorithm exploits pre-compiled policies, as heuristic metrics, to build near-optimal joint decisions at each time period.

Categories and Subject Descriptors

II.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms

Keywords

Coordination, cooperation and teamwork

1. MOTIVATING EXAMPLE

As an example to illustrate the class of problem we address, we discuss in this section a simplified model of a naval anti-air warfare problem, namely NEREUS[1]. NEREUS deals with a manned military platform evolving in a large and highly dynamic maritime environment. The objective consists in searching for schedulable control policy that avoids both the platform being hit by enemy missiles and the bounded resources being over-utilized. In order to accomplish those goals, the platform should manage its defense resources to deceive or destroy incoming targets.

Due to Command and Control (C2) system constraints, the defense resources must be used in a particular order to guaranty the task satisfiability. Those constraints define the task structure, which specifies a family of subtasks arranged in the specific order they have to be performed. As depicted in Figure 1, an enemy missile engagement task is designed as follows: In the first subtask, the missile has to be illuminated that calls for using radars (Stir or Ciw); In the second subtask, once *locked*, i.e., status of a missile for which illuminating methods have succeed, the missile can be intercepted using weapons (Sam, Gun and Ciw); Finally, the C2 system performs the *kill assessment* subtask to assess the missile destruction. Methods, referred in the following as actions, are ways of accomplishing subtasks and may be of many defense system types. For example (Figure 1), the subtask of illuminating a missile (subtask 1) may be solved by methods from either Stir or Ciw radars.

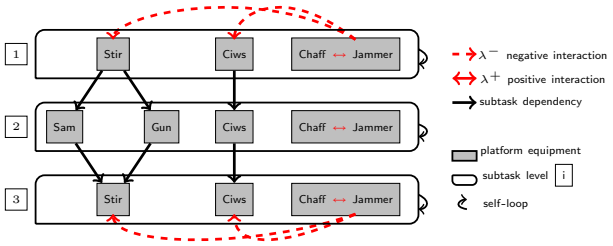

Figure 1: Cyclic progressive reasoning unit describing an enemy missile engagement task.

The naval warfare application we describe here, typically involves a large number of random variables that influence the current state. These random variables capture uncertainties from a variety of sources such as incoming targets at each time period, defense resource failures, failures of execution and defense resource interactions. An example of *negative interaction*, depicted in Figure 1, is the Chaff Cloud defense resource that is used as a delude to deceive enemy missiles, but also disturbs the operating quality of radars.

[1]Naval Environment for Resource Engagement in Unpredictable Situations.

AAMAS'07 May 14–18 2007, Honolulu, Hawai'i, USA.

978-81-904262-7-5 (RPS) ©2007 IFAAMAS

On the other hand, a *positive interaction* results from the concurrent use of Chaff Cloud and Jammer defense resources that increases their operating quality.

This problem consists of a periodic stochastic resource allocation to each of the agents present at each time period, that avoids bounded and shared resources being over-utilized. In the following, those resource-bounded agents, or *targets*, are formalized as the structured task described in Figure 1. Agent actions, in such domains, consist in assigning various resources at each subtask in the specified order previously discussed. This is a simple structured task, with the difference that decision or execution failures lead the agent to loop in the same subtask. Unfortunately, classical resource bounded processing model (PRU) does not address this domain. Therefore we introduce a new framework called *cyclic progressive reasoning unit* (C-PRU) that captures the earlier mentioned domain characteristics.

2. C-PRU FRAMEWORK

We consider a special form of MDP suitable for controlling the single resource-bounded agent environment. Here, we present an extended model named *cyclic* progressive reasoning unit [1] to infinite horizon domains handling different resource interactions. A C-PRU is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{K}, \varphi, \lambda \rangle$ where:

- $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ describes the MDP parameters corresponding to a single PRU (see [1] for further details).
- $\varphi\colon \mathfrak{P}(\mathcal{K}) \rightarrow \mathcal{A}$ is a bijective function, mapping any set of resource types to a set of actions, where $\mathfrak{P}(\mathcal{K})$ is the power set of $\mathcal{K}$;
- $\lambda\colon \mathcal{A} \times \mathcal{A} \rightarrow [0,1]$ is a function assessing the effect of dependent actions such that: $\lambda^{-}(aa')$ assesses the negative interactions those reduce the payoffs of using resource types $\varphi^{-1}(a')$ when resource types $\varphi^{-1}(a)$ are operating; $\lambda^{+}(aa')$ denotes positive interactions those increment the payoffs of using resource $\varphi^{-1}(a')$ when resource $\varphi^{-1}(a)$ are operating.

The optimal q-value functions for a C-PRU are well defined, and satisfy the following equations: for every state $s \in S_i$, all admissible actions $a \in \mathcal{A}_i$, its successors $s' \in \mathcal{S}_i^n$, and their admissible actions $a' \in \mathcal{A}$,

$$Q_{t+1}(sa) = \max \sum_{s'a'} \gamma P_i^a(ss') \left(R_i^a(ss') + Q_t(s'a') \right) \quad (1)$$

Let $P_i^a(ss'|\omega)$ be the probability of transiting to state s' when action a is taken in state s depending on the set of actions ω operating at the current time period. Since we assume the resources affect each other in an independent manner, e.g., $\lambda(a\omega) = \prod_{a' \in \omega} \lambda(aa')$, then $P_i^a(ss'|\omega)$ is defined as follows:

$$P_i^a(ss'|\omega) = (1 - \lambda^{-}(a\omega))(1 + \lambda^{+}(a\omega)) P^a(ss') \quad (2)$$

Finding an optimal policy depending on the current state and the unavailable resources is a very hard problem even for very small C-PRUs, because the equivalent MDP is very large. Indeed, the state space of that MDP consists of the cross product of the C-PRU state space and the unavailable resources. The major source of difficulties in team of shared resource agents, is that the allocation of resources to an agent influences the availability of those resources (either now or in the future) to others. Thus, a single agent policy must take into account the state of each of the other agents, rendering the problem intractable unless the number of agents, their individual state spaces and the available resources are very small. We now focus our attention to approximated strategies that limit the exponential increase of the state space.

We define *greedy* q-value functions which depend on the current state, the action taken and the set of unavailable resources. For each state $s \in \mathcal{S}_i$, all admissible action $a \in \mathcal{A}_i$ its successors $s' \in \mathcal{S}_i^n$, and their admissible actions $a' \in \mathcal{A}$, the following holds

$$Q_{t+1}(sa|\omega) = \max \sum_{s'a'} \gamma P_i^a(ss'|\omega) \left(R_i^a(ss') + Q_t(s'a') \right) \quad (3)$$

$Q_{t+1}(sa|\omega)$ is a greedy value of action a taken in state s under the set of unavailable resources ω, because it limits the influence of the unavailable resources only over the current time period. These q-value functions are particularly useful to merge individual agent solutions into a composite global solution of the team as described below.

We can therefore define the periodic multi-agent optimization problem under bounded and shared resources as a tuple $\langle \mathcal{C}\text{-PRU}, \Theta, O \rangle$ where:

$\mathcal{C}\text{-PRU} = [\{\mathcal{C}\text{-PRU}_t^j\}_{j=1}^{n_t}]_{t=1}^{T}$ is the vector of $\mathcal{C}\text{-PRU}_t^j$. Each agent j is formalized as a $\mathcal{C}\text{-PRU}^j = \{\mathcal{C}\text{-PRU}_t^j\}_{t=1}^{T}$ described by $\langle \mathcal{S}^j, \mathcal{A}^j, \mathcal{P}^j, \mathcal{R}^j, \mathcal{K}^j, \varphi, \lambda^j \rangle$. In general, for an MMDP, we have to define a new state space that is the cross-product of the state spaces of all agents, $S_t = \times_{j=1}^{n_t} \mathcal{S}^j$, and a new action space, e.g., $A_t = \times_{j=1}^{n_t} \mathcal{A}^j$. $S = [S_t]_{t=1}^{T}$ and $A = [A_t]_{t=1}^{T}$ are vectors of joint data over time periods;

$C = [C_t]_{t=1}^{T}$ is the vector of available resources, and $C_t = [c_{tk}]_{k=1}^{|\mathcal{K}|}$ defines the vector of total amounts of shared resources available to the team at time period t. There are c_{tk} units of resource k available at time period t to the agents.

$\Theta = [\Theta_t]_{t=1}^{T}$ is a vector of random variable state spaces. $\Theta_t = \{\theta_t\}$ is set of possible incoming random variables at time period t.

$O = [O_t]_{t=1}^{T}$ is a vector of functions, and $O_t\colon S \times \Theta \rightarrow S$ defines a mapping from sets of random variables and states to states.

The following theorem characterizes the complexity of the periodic multi-agent optimization decision problem tagged with DM-C-PRU (dynamic multi-agent C-PRU problem). Proof sketch is omitted because of lack of space.

Given an instance of multi-agent C-PRU$_{t=1,\cdots,T}$ defined as a tuple $\langle S, A, P, R, C, \mathcal{K}, \Theta, O \rangle$ and a rational number V, does there exists a sequential policy $\pi = [\pi_t]_{t=1,\cdots,T}$ whose expected total payoff equals or exceeds V?

THEOREM 1. *A DM-C-PRU problem is PSPACE-Complete.*

3. RTDA* ALGORITHM

In the following, we restrict our purpose to one period tree search with n_t agents, where $Tree_t = (V_t, E_t)$ corresponds to the current tree search space; $V_t = \{o_i^j\}_{i,j}$ denotes the set of nodes; $o_i^j = \langle s^j, \omega_i^j, \zeta_i^j \rangle$ describes the j-th agent characteristics (with respect to EDF order) where ω_i^j describes its unavailable resources, ζ_i^j its estimate value and s^j its state; $E_t = \{\langle o_i^j, a_{i'}^j, o_{i'}^{j+1} \rangle\}_{i,j,i'}$ is the set of edges, where each edge

$\langle o_i^j, a_{i'}^j, o_{i'}^{j+1}\rangle$ is tagged with one of the admissible action $a_{i'}^j$, providing a "*heuristic estimate*" of the edge $\langle o_i^j, a_{i'}^j, o_{i'}^{j+1}\rangle$.

RTDA* is a real-time tree search algorithm that finds a path from a given current node o_i^j (initially o_0^1) to a *goal* node. Intuitively, the algorithm proceed by expanding the most promising edge $\langle o_i^j, a_{i'}^j, o_{i'}^{j+1}\rangle$ chosen according to a "*heuristic estimate*" that ranks each edge leading from the current node o_i^j. It maintains a set of edges to be expanded next, stored in a priority queue l. The priority assigned to an edge $\langle o_i^j, a_{i'}^j, o_{i'}^{j+1}\rangle$ is determined by the q-value, precomputed in the off-line phase, related to the action $a_{i'}^j$. The algorithm explores as far as possible along each edge until a goal node is found, i.e., until it reaches a node that has no children, or until it hits a "*solved*" node. Recall that a node is said to be *solved* if all its immediate successor nodes have been completely explored. Before backtracking, RTDA* evaluates the current node $o_{i'}^{j+1} = \langle s^{j+1}, \omega_{i'}^{j+1}, \zeta_{i'}^{j+1}\rangle$ as follows: for all admissible actions $a_{i'}^j \in \mathcal{A}^j$ of agent associated with node o_i^j, $\omega_{i'}^{j+1} = \omega_i^j \cup \varphi^{-1}(a_{i'}^j)$

$$\zeta_{i'}^{j+1} = \sum_{m=j+1}^{n_t} \rho^m \prod_{l=j+1}^{m} \zeta^l \text{ and } \zeta^l = \max_{a^l \omega_{i'}^l} Q^l(s^l a^l | \omega_{i'}^l) \quad (4)$$

Evaluations of nodes visited are stored in a hash table. Then the search backtracks, returning to the most recent visited node not completely explored and continuing the evaluation. This evaluation uses previously stored evaluations, if they exist in the hash table as follows:

$$\zeta_i^j = \zeta^j \left[\rho^j + \max_{i'} \zeta_{i'}^{j+1}\right] \quad (5)$$

, otherwise it uses the evaluation estimate function (Equation (4)). Therefore, RTDA* can be summarized in the following loop, executed once per time period: (*i*) Order the n_t agent states $\{s^j\}_{j=1}^{n_t}$ with respect to EDF order, the root node o_0^1 of the tree $Tree_t$ is the one composed with the first state following EDF order; (*ii*) Expand the current node (initially o_0^1) by generating its immediate successors and evaluate them as mentioned earlier, that is achieved by $search(o_i^j)$ method. $gen(o_i^j, a_{i'}^j)$ method in particular generates the successor node $o_{i'}^{j+1} = \langle s^{j+1}, \omega_{i'}^{j+1}, \zeta_{i'}^{j+1}\rangle$ leading from o_i^j (Equation (4)), if o_i^j is not a leaf, otherwise it returns the same node o_i^j; (*iii*) Compute $\{\zeta_i^j\}_{j,i}$ equations (4) and (5) and thus greedily determine the best action of each agent according to its predecessor choices; (*iv*) Execute the composite action $\{a^j\}_{j=1}^{n_t}$ at the end of the response time constraint, observe the resulting state s'_{t+1}, the random variables θ_{t+1}, and therefore create the new state $s_{t+1} = O_{t+1}(s'_{t+1}, \theta_{t+1})$ and determine the remaining resources $C_{t+1} = C_t - \sum_j a^j$. A more complex resource transition function may be used to cope different resource types such as *consumable* as well as *non-consumable*.

In some cases, only a few successors are generated and expanded rather than all. Indeed, since our algorithm proceeds by expanding the most promising node first, the resulted value can be used as a *lower bound* (LB) for the other successors. Thus, if the highest possible value, i.e., *upper bound*, of a given successor is higher than LB, then the node is expanding, otherwise it is not, i.e.,

$$\begin{aligned} \zeta_{i'}^j &= \textstyle\sum_{m=j}^{n_t} \rho^m \prod_{l=j}^{m} \zeta^m \\ &\leq \zeta^j \left[\rho^j + (n_t - j)\rho_{\max}\right] \end{aligned} \quad (6)$$

Algorithm 1 RTDA* algorithm: **Tree evaluation**.

Require: agent states $\{s^j\}_{j=1}^{n_t}$ ordered following EDF.
Ensure: $tree$.
1: $tree \leftarrow$ EmptyTree
2: $open \leftarrow$ EmptyStack
3: $search(o_0^1)$.
4: **while** $open \neq$ EmptyStack **do**
5: $\langle o_i^j, a_{i'}^j, o_{i'}^{j+1}\rangle \leftarrow open.pop()$.
6: **if** $o_{i'}^{j+1}$ is not yet *visited* **then**
7: add $\langle o_i^j, a_{i'}^j, o_{i'}^{j+1}\rangle$ to $tree$.
8: $search(o_{i'}^{j+1})$.
9: **else**
10: update ζ_i^j as mentioned Equation (4) and (5).
11: **end if**
12: **end while**

where $\rho_{\max} = \max_{j=1}^{n_t} \rho^j$ denotes the highest possible weight of any agent, thus, if $\zeta^j \left[\rho^j + (n_t - j)\rho_{\max}\right] \geq LB$, the node is expanded. As a result RTDA* is guaranteed to find a *joint decision* that maximizes the estimate function (Equation 4) for any finite tree in time bounded by $O(|Tree_t|)$ at each time period.

To validate our approach, we tried it on several randomly-generated instances of the naval anti-air warfare domain previously described, and compared them to the optimal policy computed using flat RTDP algorithm on the MMDP describing the multi-agent system with only one starting state, i.e., the joint states of all agents.

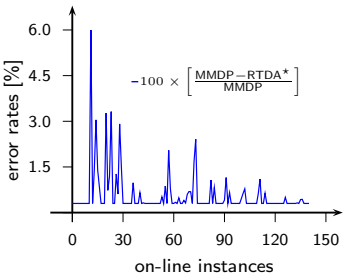

Figure 2: Average error rates using RTDA*.

4. CONCLUSIONS

We have presented an approach for solving large and periodic multi-agent optimization problems under resource constraints. We described how to use pre-compiled solutions as heuristic metrics to build on-line near-optimal solutions. The empirical results for the naval anti-air warfare domain we investigated, are extremely encouraging, demonstrating the ability of our technique to solve problems of magnitude higher than those arising in such applications.

5. REFERENCES

[1] A.-I. Mouaddib and S. Zilberstein. Optimal scheduling of dynamic progressive processing. In *ECAI*, pages 499–503, 1998.