Decentralized Markov Decision Processes for Handling Temporal and Resource constraints in a Multiple Robot System

Aurélie Beynier, Abdel-Illah Mouaddib

1

Decentralized Markov Decision Processes for Handling Temporal and Resource constraints in a Multiple Robot System

Aurélie Beynier[1], Abdel-Illah Mouaddib[1]

GREYC-CNRS, Bd Marechal Juin, Campus II, BP5186, 14032 Caen Cedex, France abeynier,mouaddib@info.unicaen.fr

Summary. We consider in this paper a multi-robot planning system where robots realize a common mission with the following characteristics : the mission is an acyclic graph of tasks with dependencies and temporal window validity. Tasks are distributed among robots which have uncertain durations and resource consumptions to achieve tasks. This class of problems can be solved by using decision-theoretic planning techniques that are able to handle local temporal constraints and dependencies between robots allowing them to synchronize their processing. A specific decision model and a value function allow robots to coordinate their actions at runtime to maximize the overall value of the mission realization. For that, we design in this paper a cooperative multi-robot planning system using distributed Markov Decision Processes (MDPs) without communicating. Robots take uncertainty on temporal intervals and dependencies into consideration and use a distributed value function to coordinate the actions of robots.

1.1 Introduction

Although a substantial progress with formal models for decision process of individual robots using Markov Decision Process (MDP), extensions of MDP to multiple robots is lacking. Recent attempts identify different classes of multi-robot decision process that include Multi-Agent Markov Decision Process (MMDP) proposed by Boutilier [Bou99], the Partial Observable Identical Payoff Stochastic Game (POIPSG) proposed by Peshkin et al. [PKMK00], the multi-agent decision process by Xuan and Lesser [XLZ00], the Communicative Multiagent Team Decision Problem (COM-MTDP) proposed by Pynadath and Tambe [NPY+03], the Decentralized Markov Decision Process (DEC-POMDP and DEC-MDP) by Bernstein et al. [BZI00], DEC-MDP with a central coordination proposed by Hanna and Mouaddib [HM02], the DEC-POMDP with communication proposed by Goldman and Zilberstein [GZ03] and Transition Independent DEC-MDP proposed by Becker et al. [BZLG03]. Bererton et al. [BGT03] present an extension of MDPs and apply auction mechanisms to coordinate multiple robots. Therefore, they reduce communication. Nevertheless, agents communicate during the executing.

Our approach is a specific structured DEC-MDPs to control multiple robots realizing a common mission with temporal and dependency constraints. Indeed, the formal model we use can be seen as an extension of the structured DEC-MDP, proposed by Becker et al. [BZLG03], to increase the expressiveness of the model to handle temporal constraints. The formal model we develop is based on an augmented MDP per robot using an augmented reward function that represents the reward a robot gains when it achieves a task and the opportunity cost of violating temporal constraints. As described in [NPY+03], our system builds the policy in a centralized way but the execution is decentralized. Nonetheless, in our approach, the robots don't have to communicate during their execution. This model is motivated by a more general scenario than the one introduced in [CMZW01] to control the operations of a rover. The extended scenario, considered, is a multiple robot scenario in which each robot has a mission to achieve (a set of tasks) similar to the planetary rovers [BDM+02] where each one has to visit sites and to take pictures, conduct experiments and collect data (see the example in [CMZW01]).

These scenarios are mainly characterized by

1. Single rover activities have an associated temporal window : the examples involve measurements of the environment – a "gravity wave" experiment that needs to be done "preferably in the morning", and atmospheric measurements at sunrise, sunset, and noon (look at the sun through the atmosphere).
2. There is uncertainty on time realization tasks and the temporal interval activity of rovers.
3. The temporal constraints are, in general, soft.
4. Precedence dependencies among tasks exist.
5. Interacting tasks could not be assigned to the same rover because of its limited resource : one popular scenario is preparing a site with a set of robotic bulldozers (pushing dirt and small stones). In this paper, we don't take into account tasks achieved by more than one robot.

We present in the next sections the multi-robot decision process system and how the decision process is distributed and coordinated among rovers.

1.2 A Formal Description of Decentralized MDP

One approach to this problem is to represent the system as a large Markov Decision Process (MDP) [SB98] where the "macro-action" is a joint action of all of the robots and the reward is the total reward of all of the robots. The problem is the large action space that is for n robots and m actions for each one, we have m^n macro-actions [GDP01]. Some approaches use Factored MDPs while other approaches are designed in such a way that most of them are based on distributed robots with partial information. Such approaches are known to be intractable [BZI00]. In our approach, we view the entire multi-robot system as distributed Markov Decision Process robots without communication but some information on the effect of the local robot's actions, on the plans of the other robots are assumed to be available. Differently speaking, the system is composed on robots each of which constructs a local MDP and derives a local policy, taking into account the temporal dependencies. Indeed, in the

local MDP we introduce an opportunity cost due to the current activity of the robot which delays the successor activities of the other robots. It measures the loss in value when starting the activity with a delay δt. This cost is the difference between the value when we start at time and the value when we start with a delay δt. Consequently, each robot develops a local MDP independently of the local policies of the other robots but, introducing in its expected value the opportunity cost due to the possible delay of his activity. The construction of the local MDP is based, first, on the construction of the state space, second, on the computation of the opportunity cost at each state and then, the computation of the value of each state to construct the local optimal policy. To define the state space, we need to define what is the state for the decision process. To do that, let us recall the characteristics of the problem.

The mission of the robot, as described previously, is a graph of tasks where each task is characterized by the execution time window, and the uncertainty on the execution time and resource consumption. In the rest of the paper, we assume that the mission graph is given and each task has a set of possible execution times and their probabilities, and a set of possible amounts of resource and their probabilities. It means that the representation of execution time and resource consumption are discrete. Given these information, the problem is to choose the best decision about which task to execute and when to execute it. This decision is based on the available resources and the temporal constraints. The respect of the temporal constraints requires to know the interval of time during which the current task has been executed. From that, the decision process constructs its decision given the current state of the last executed task, the remaining resources and the interval during which this task has been executed. The state of this decision process is then, [l_i, r, I] that corresponds to the last executed task l_i, the remaining resource r and the interval of time. Given the uncertainty on the execution time, there exist many possible intervals of time during which a task could be executed. In order to develop the state space of the decision process, we need to know for each task in the graph the set of its possible execution intervals of time. To do that, we develop an algorithm that computes for each task in the graph all the possible intervals of time by propagating different execution times.

The algorithm of developing a local MDP is divided into 5 major steps :

- Propagating the temporal constraints and computing the set of execution intervals of time for each task (node in the graph) among the graph (section 1.4).
- Computing the probability for each interval of time (section 1.5).
- Constructing the state space of the Markov Decision Process using the transition model (section 1.6).
- Computing the opportunity cost at each state (section 1.6).
- Using the value iteration algorithm to solve the MDP.

In the rest of the paper, we describe each step of the algorithm and we give the formal description of the local MDP and their interactions.

1.3 Preliminaries

In the previous section, we describe the overall basis of the model we develop and give its main lines. For that we consider a distribution probability on execution time and resource, and probabilities on start and end time of tasks.

1.3.1 Uncertain computation time

The uncertainty on execution time has been considered in several approaches developed in [CMZW01]. All those approaches ignore the uncertainty on the start time. We show in this paper how extensions can be considered in those approaches taking different temporal constraints into account.

Definition 1 *A probabilistic execution time distribution,* $Pr(\delta_c = t_c) = P_c(t_c)$ *is the probability that the activity takes* t_c *time units for its execution.*

The representation adopted of this distribution is discrete. We use a set of couples (t_c,p) where each couple means that there is a probability p that the execution will take t_c time units.

1.3.2 Uncertain resource consumption

The consumptions of resources (energy, memory, etc ...) are uncertain. We assume a probability distribution on the resource consumptions of a rover when performing an activity.

Definition 2 *A probabilistic resource consumption is a probability distribution,* $Pr(\Delta r)$ *of resource consumption measuring the probability that an activity consumes* Δr *units of resources.*

The representation adopted of this distribution is discrete. We use a set of couples (r,p) where each couple means that there is a probability p that the execution will consume r units.

1.3.3 Temporal window of Tasks

Each task is assigned a temporal window [EST,LET] during which is should be executed. EST is the earliest start time and LET is the latest end time. The temporal execution interval of the activity (start time and the end time) should be included in this interval.

1.3.4 Rewards

Each robot, i, receives a reward R presumably based on the quality of the solution and the remaining resources. For all states $[l_i, r, I]$, the reward function is assumed given $R([[l_i, r, I])$. However, we assume that all failure states have a zero reward.

1.4 Temporal interval propagation

1.4.1 A simple temporal interval propagation algorithm

Given the possible transition times of different tasks, we determine the set of temporal intervals during which a task can be realized. Firstly, the possible *start times* is a one of instants $EST, EST + 1, \ldots, LET - \min \delta_i$ (Last start time that we node LST). However, a robot needs to know when its predecessor terminates its processing to validate some of those start times. For that, we

compute off-line all the possible end times of all its predecessors and compute its possible start times consequently. The possible intervals I of a robot are determined with a simple temporal propagation constraints in the graph. This propagation organizes the graph into levels such that : l_0 is the root of the graph, l_1 contains all the successors of the root (successors(root)), ..., l_i contains the successors of all nodes at level l_{i-1}. For each node in given level l, we compute all its possible intervals of time from its predecessors.

• $level_0$: the start time and the end times of the root node (the first task of the mission) are computed as follows : $st(root) = EST(root)$ and $ET(root) = \{st(root) + \delta_c^{root}, \forall \delta_c^{root}\}$ where δ_c^{root} is the execution time of the first activity (task) of the mission.

• $level_i$: for each node in level i, it starts its execution when all its predecessors end their own activities. The set of the possible end times of the node is then given thanks to the start times and the task's durations : $ET(node) = \bigcup_{\forall \delta_c^{node}, st} \{st + \delta_c^{node}\}$ where δ_c^{node} is the execution time of the activity (task) at node $node$ and $st \in ST(node)$. We recall also here that there is a probability that some end times can violate the deadline LET_{node}.

1.5 A Probability propagation algorithm

After computing off-line all the possible execution intervals of each activity (or node), we describe, in this section, how we can weight each of those intervals with a probability. This probabilistic weight allows us to know the probability that an activity can be executed during an interval of time. For that, a probability propagation algorithm among the graph of activities is described using the execution time probability and the temporal constraints EST, LST, and LET. This algorithm has to take the precedence-constraint that affects the start time of each node, and the uncertainty of execution time that affects the end time of the node. In the following, we describe how the conditional start time probability (DP) is computed and the probability of an execution interval P_w using DP and the probability of execution time P_c.

Conditional probability on start time

The computation of conditional start time has to consider the **precedence-constraint** that expresses the fact that an activity cannot start before its predecessors finish. The start time probability of an activity should express the uncertainty on the precedence-constraint dependency. This constraint expresses the fact that the activity starts its execution when all activities of its predecessors have been finished. Consequently, the probability $DP(t)$ that a robot starts an activity at t is the product of the probability that all predecessor robots terminate their activities before time t and there is, at least, one of them that finish at time t. More formally speaking :

•for the root : $DP_s(i) = 1, i \in [EST_{root}, LET_{root}]$

•for the other nodes : $DP_s(\delta_s = t) = \Pi_{a \in predecessors(c)} Pr^a(\delta_e \leq t) - \sum_{t' < t} DP(t')$

Where a is an activity of a predecessor robot of the robot performing the considered activity c and $Pr^a(\delta_e \leq t)$ is the probability that predecessor a finishes before time t. This probability is the sum of probabilities that the predecessor a executes its task in an interval I with an end time $et(I)$ less than t. More formally speaking, $Pr^a(\delta_e < t) = \sum_{t_i + t_j = t} DP_s(t_i).P_c(t_j)$ such

that $[t_i, t]$ is one of intervals $interval(a)$ computed for the activity of robot a. This probability can be rewritten as follows :

$Pr^a(\delta_e < t) = \sum_{I \in intervals(a), et(I) < t} Pr(I)$

In the following, we show how we compute the probability that an execution occurs in an interval I.

Probability on a temporal interval of an activity

Given the probability on start time and end time, we can compute the probability that the execution of an activity occurs during the interval I where *st(I)* is the start time and *et(I)* is the end time.

Definition 3 *A probabilistic execution interval I is the probability $P_w(I)$ of the interval during which an activity can be executed. This probability measures the probability that an activity starts at $st(I)$ and it ends at $et(I)$.*

$$P_w(I) = DP_s(st(I)).P_c(et(I) - st(I))$$

An activity l_{i+1} of an agent is executed during an interval I' when the agent finishes its activity l_i and that all predecessor agents finish their activities. To compute the probability of the interval I', we need to add the fact that we know the end time of activity l_i. For that, we compute the probability $P_w(I'|et_{l_i}(I))$ such that :

$$P_w(I'|et_{l_i}(I)) = DP_s(st(I')|et_{l_i}(I)).P_c(et(I')_{l_{i+1}} - st(I)_{l_{i+1}})$$

And the probability $DP_s(st(I')|et_{l_i}(I))$ is computed as follows :

$$DP_s(st(I')|et_{l_i}(I)) = \prod_{a \in predecessors(l_{i+1}) - l_i} P_r^a(\delta_e \leq st(I')|et_{l_i}(I)) - \sum_{t_1 < st(I')} DP_s(t_1|et_{l_i}(I))$$

$$P_r^a(\delta_e \leq t|et(I)_{l_i}) = \sum_{I_1 \in, et(I_1) = et(I)_{l_i}, et(I_1) \leq t} P_w(I_1)$$

This equation expresses the fact that we know activity l_i has finished at $et(I)$, and allows us to consider only the probability that the other predecessor activities finish.

1.6 A Decision Model for Multi-robot Planning system with temporal dependencies

As mentioned above, we formalize this problem with Markov Decision Processes. To do that, we need to define what is the state space, the transition model and the value function for each robot.

Each rover develops its local policy using the off-line temporal interval propagation. Robots don't need to communicate since all information needed for each to make a decision are available. The consequence of representing all the intervals and all the remaining resources, is that the state space becomes fully observable and the decision process can start its maximization action selection using the modified Bellman equation defined bellow. However, the maximization action selection uses an uncertain start time that is computed from an uncertain end time of the predecessors.

The start time selected by the policy can be earlier or later than the end time of the predecessors. When the start time is later than the end time of the predecessors, we need to handle the situation where the start time is later than $LET - \min \delta_i$. The other case is when the policy selects a start time earlier than the end time of the predecessors, in such case, the action selected won't succeed since the precedence constraint is not respected. In such case, we assume that the new state is a partial failure and a penalty should be paid. For example, rover A assumes that at time t_1, rover B (bulldozer) finishes its processing and it can moves toward its destination. When rover B finishes later than t_1, the moving action of rover A fails. In the rest of this section we formalize this decision process.

State Representation, Transition model and Values

Each robot, i, observes its resource levels and the progress made in achieving its tasks which represent the state of the robot. The state is then a triplet $[l, r, I]$ where l is the last task, r is the available resource and I is the interval during which the task has been executed.

We assume that the actions of one robot is enough to achieve a task. The robot should make a decision on which task to execute and when to start its execution. However, the actions to perform consist of *Execute the next task* l_i at time st ($\mathsf{E(st)}$) that is the action to achieve task i at time st when the task $i-1$ has been executed . This action is probabilistic since the processing time of the task is uncertain. This action allows the Decision process to move from state $[l_i, r, I]$ to state $[l_{i+1}, r', I']$. When a robot starts before its predecessors terminate, this transition leads to a failure state that we recover by a state $[l_i, r - \Delta r, [st, unknown]]$ where Δr is the consumed resources. This recovery allows us to represent the situation where the robot acts with no results, except the fact that further resource has been consumed. Finally, the execution can lead to a failure state that we represent with $[failure, 0, [st, +\infty]]$ when the remaining resources are not enough to realize a task. It can also lead to another failure state when the execution starts too late ($st > LET - \min \delta_i$). We use ∞ or *unknown* in order to indicate to the policy that those states needs a special consideration by considering a special value that we explain in the next section. Let us just give a meaning to $+\infty$: $+\infty$ means that the robot is never be able to achieve the task while *unknown* means that the robot tries and fails but there is a chance to succeed another time by starting later. The transitions are formalized as follows :

- *Successful transition :* The action allows the policy to transition to a state $[l_{i+1}, r', I']$ where task l_{i+1} has been achieved during the interval I' respecting the EST and LET time of this task and that r' is the remaining resource for the rest of the plan. The expected value to move to the state $[l_{i+1}, r', I']$ is : $V1 = \sum_{\Delta_r \leq r} \sum_{et(I') \leq LET} P_r(\Delta_r).P_w(I'|et(I)).V([l_{i+1}, r', I']$
- *Too late start time Transition :* The action starts too late and the execution meets the deadline LET. In such case, the action allows the policy to move to a $[failure, r, [st, +\infty]]$. The expected value to move to this state is : $V2 = Pr(st > LET - min\delta_i).V([failure, r, [st, +\infty]]) =$

$$\prod_{a \in pred(l_{i+1}) - \{l_i\}} \sum_{I_a} P_w(I_a) - \prod_{a \in pred(l_{i+1}) - l_i} \sum_{I_a : et(I_a) \leq LET - min\delta_i} P_w(I_a).V([failure, r, [st, +\infty]])$$

- *Deadline met Transition :* The action starts an execution at time st but the duration δ is so long that the deadline is met. This transition moves to the state $[failure, r, [st, +\infty]]$. The expected value to move to this state is : $V3 = \sum_{\Delta r \leq r} \sum_{st+t_c > LET} P_r(\Delta r).DP(st|et(I)_{l_i}).P_c(t_c).V([failure, r, [st, +\infty]])$
- *Insufficient resource Transition :* The execution action requires more resources than available. This transition moves to the state $[failure, 0, [st, +\infty]]$. The expected value to move to this state is : $V4 = \sum_{\Delta_r > r} P_r(\Delta_r).V([failure, 0, [st, +\infty]])$
- *Too early start time Transition :* The action starts too early before one of the predecessor robots has finished its tasks. In such case, the action allows the policy to move to $[failure, r, [st, st+1]]$. Conceptually, we proposed that the end time should be unknown, but in our model we formalize it by the fact that when a robot starts before its predecessor robots finish, it realizes it immediately. This means that the robot, at $st+1$, realizes that it fails. This state is a non-permanent failure because the robot can retry later. The robot should pay a penalty k visiting such states. The expected value to move to this state is :

$$V5 = \sum_{\Delta r \leq r} \Big(\Big(\prod_{a \in predecessors(l_{i+1}) - \{l_i\}} \sum_{I_a : et(I_a) > LET - min\delta_i} P_w(I_a) \Big)$$

$$- \sum_{s \leq st} DP(s|et(I)_{l_i}) \Big).P_r(\Delta r).V([failure, r, [st, st+1]])$$

Given these different transitions, we adapt our former to Bellman equation as follows :

$$V[l_i, r, I] = \overbrace{R([l_i, r, I])}^{\text{immediat gain}} \overbrace{- \sum_{k \in \text{sucessors}} OC_k(et(I) - et(I_{first}))}^{\text{Opportunity Cost}}$$

$$+ \overbrace{max_{E(l_{i+1}, st), st > current\_time}(V1 + V2 + V3 + V4 + V5)}^{\text{Expected value}}$$

This equation means that robot rewards $R([l_i, r, I])$ are reduced by an opportunity cost due to the delay caused in the successor robots.

Opportunity cost is the lost in value when a robot starts with a delay. This means it is a difference between V^0 when we start with no delay and $V^{\Delta t}$ when we start with a delay Δt : $OC_k(\Delta t) = V^0 - V^{\Delta t}$ with : i) $\forall t \leq 0, OC(t) = 0$, ii) $OC(unknown) = 0$, iii) $\forall t > LET - \min_i \delta_i$

$$OC([failure(l_i), r, [+\infty, +\infty]]) = R(l_i) + \sum_{a \in AllSucc(l_i)} R(a)$$

The opportunity cost has to be computed off-line for all the delays from 0 to latest start time (LET - $\min \delta_i$) and for each task. We store all these costs in library that allows to each robot to know the opportunity cost of its delay in the values of its successors. Each robot has, for each successor i, the corresponding opportunity cost OC_i function for each task.

1.7 Implementation

This multi-robot decision system has been implemented and run on a scenario involving two robots : R1 and R2. Each one must exit a warehouse, move to an

object, and catch it. Then, it must bring back the object to another warehouse. This problem implies constraints :

- *Temporal constraints* : warehouses have opening and closing hours. They open from 1 to 4 and from 25 to 32.
- *Precedence constraints* : the object O1 must be moved in order R2 to bring O2 back to warehouse 1.

The robots' mission can be represented by the graph shown in the figure 1.1. A fictitious task is added at the begin of the mission in order to have only one root. The intervals stand for the temporal constraints. For each task, the graph specifies the robot that must complete it. We have tested our decision system on this scenario. The results are promising. The robots perform the whole mission and act in an optimal way : for each step, each robot chooses the action that leads to the highest expected value.

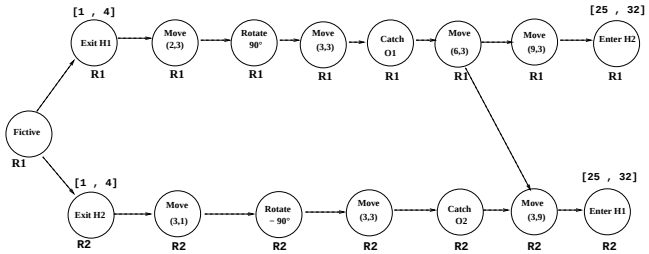

Fig. 1.1. Graph of the mission

More complex scenarios, involving 3 or 4 robots, are under development. They deal with planetary rovers or crisis scenario. For instance, a set of rovers must operate on the surface of Mars, they must complete different observations or experiments : take pictures, complete atmospheric measurements,... Another scenario deal with crisis control : an earth-quake arises and the firemen, the policemen and the ambulance men have several tasks to complete in order to rescue the inhabitants.

Currently, we are developing different experiments by modifying three parameters : *number of tasks*, *number of constraints* and *number of robots*. The first results show the robustness of our approach and its ability to support a large problem (200 tasks, 3 robots and 50 constraints) where the state space is almost 300000 states. Furthermore, given a mission, it can be shown that when we increase the number of agents, the state space size of each local MDP decreases. Indeed, each agent has less tasks to complete and the number of triplet $[l_i, r, I]$ decreases. If we increase the number of tasks, the state space size increases. When we increase the number of precedence constraints or we tighten the temporal constraints (smaller temporal windows $[EST, LET]$), the number of possible execution intervals goes down. Thus, the state space size diminishes. Also, the initial resource rate has effects on the space state size. Formalization of these fluctuations is under development.

1.8 Conclusion

In this paper we have deal with a multi-robot system under temporal and complex constraints. In this multi-robot system, robots are with limited and uncertain resources. This problem can be seen as a multi-robot planning under uncertainty with temporal and complex dependencies and limited resources. To address this problem, we proposed a decentralized MDPs framework. In this framework, MDPs are with no communication and they don't have a complete observation about the states of the other robots. This framework is based on the notion of opportunity cost to derive an optimal joint policy. Each robot constructs its optimal policy to achieve all its tasks taking for each one the dependency with the other tasks of the other robots. This policy respects the local temporal constraints (EST, LST and LET) and the temporal dependency between robots (precedence constraint).

Future work will concerns many techniques for computing exact or approximate opportunity cost.

References

[BDM+02] J. Bresina, R. Dearden, N. Meuleau, S. Ramakrishnan, D. Smith, and R. Washington. Planning under continuous time and resource uncertainty : A challenge for ai. In *UAI*, 2002.

[BGT03] C. Bererton, G. Gordon, and S. Thrun. Auction mechanism design for multi-robot coordination. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Proceedings of Conference on Neural Information Processing Systems (NIPS)*. MIT Press, 2003.

[Bou99] Graig Boutilier. Sequential optimality and coordination in multiagents systems. In *IJCAI*, 1999.

[BZI00] D. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of mdps. In *UAI*, 2000.

[BZLG03] R. Becker, S. Zilberstein, V. Lesser, and C. Goldman. Transition-independent decentralized markov decision processes. In *AAMAS*, 2003.

[CMZW01] S. Cardon, AI. Mouaddib, S. Zilberstein, and R. Washington. Adaptive control of acyclic progressive processing task structures. In *IJCAI*, pages 701–706, 2001.

[GDP01] C. Guestrin, D.Koller, and R. Parr. Multiagent planning with factored mdps. In *NIPS*, 2001.

[GZ03] C. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multiagent systems. In *AAMAS*, 2003.

[HM02] H. Hanna and AI Mouaddib. Task selection as decision making in multiagent system. In *AAMAS*, pages 616–623, 2002.

[NPY+03] R. Nair, D. Pynadath, M. Yokoo, M. Tambe, and S. Marsella. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 2003.

[PKMK00] L. Peshkin, K.E. Kim, N. Meuleu, and L.P. Kaelbling. Learning to cooperate via policy search. In *UAI*, pages 489–496, 2000.

[SB98] R.S. Sutton and A.G. Barto. Reinforcement learning : An introduction. *MIT press, Cambrige, MA*, 1998.

[XLZ00] P. Xuan, V. Lesser, and S. Zilberstein. Communication decisions in multiagent cooperation. In *Autonomous Agents*, pages 616–623, 2000.