Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 753487, 10 pages
http://dx.doi.org/10.1155/2014/753487

Research Article

OL-DEC-MDP Model for Multiagent Online Scheduling with a Time-Dependent Probability of Success

Cheng Zhu, Jiangfeng Luo, Weiming Zhang, and Zhong Liu

Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, China

Correspondence should be addressed to Jiangfeng Luo; nudtluojiangfeng@gmail.com

Received 11 February 2014; Revised 5 June 2014; Accepted 1 July 2014; Published 22 July 2014

Academic Editor: Yun Li

Focusing on the on-line multiagent scheduling problem, this paper considers the time-dependent probability of success and processing duration and proposes an OL-DEC-MDP (opportunity loss-decentralized Markov Decision Processes) model to include opportunity loss into scheduling decision to improve overall performance. The success probability of job processing as well as the process duration is dependent on the time at which the processing is started. The probability of completing the assigned job by an agent would be higher when the process is started earlier, but the opportunity loss could also be high due to the longer engaging duration. As a result, OL-DEC-MDP model introduces a reward function considering the opportunity loss, which is estimated based on the prediction of the upcoming jobs by a sampling method on the job arrival. Heuristic strategies are introduced in computing the best starting time for an incoming job by each agent, and an incoming job will always be scheduled to the agent with the highest reward among all agents with their best starting policies. The simulation experiments show that the OL-DEC-MDP model will improve the overall scheduling performance compared with models not considering opportunity loss in heavy-loading environment.

1. Introduction

Problems involving time-dependent success probability extensively exist in manufacturing, industrial, and military domains. One example is the scheduling of a procrastinator [1], whose speed and success probability of job processing will increase as the due date is approaching. Practice shows that a procrastinator's performance varies under different time pressures when processing the same job. As higher time pressure is more likely to force a procrastinator to make mistakes when processing a sophisticated job, the success probability is consequentially dependent on the starting time. Another example is the antiship missile defense by SAM (surface-air-missile) systems shown in [2]. SAM systems are scheduled to intercept the incoming antiship missiles within feasible interception time window. Killing probability of the interception is associated with the range at which the interception missile and the antiship missile meets, which in turn depends on the launching time of the interception. Usually, an earlier firing time means more flight time before engagement.

Both of the above examples imply that though early starting strategy for job processing guarantees maximal time window for processing, longer processing duration will be spent as a price. Compared with the classic on-line scheduling [3, 4], extra trade-offs should be considered by the agent between the current job and the possible incoming jobs. For example, Figure 1 gives the killing probability associated with the time at which the engagement occurs for a SAM system of a Halifax ship against the incoming antiship missile [5]. It can be inferred that an antiship missile can be intercepted in the feasible time window $[t^l, t^u]$. Hence the SAM system can choose the best engaging time t_m to get the highest killing probability. If the interception fails, SAM system will have time window $[t_m, t^u]$ to take an immediate remedial interception. However, if the SAM system fires at earlier time and makes the engagement occur at t^l, though the killing probability is lowered down, longer time window is left in case of interception fail. Therefore, the SAM system needs to make trade-off between a high killing probability of current interception and more feasible time left to take remedial

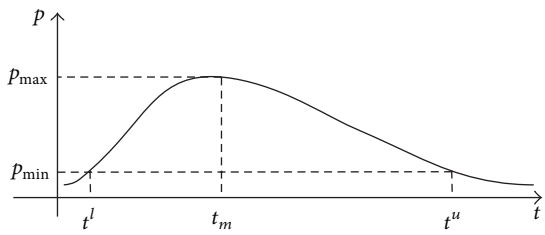

Figure 1: Engagement time-dependent killing probability of the interception missile.

action in case of interception fail. In addition, for an antiship missile and a SAM system, an earlier firing time always means more flight time before the engagement. Hence adopting the early firing strategy would cause SAM system to spend longer duration on the current interception, while losing more opportunities to intercept possible upcoming missiles. This is the second trade-off to be considered.

Similar trade-off also exists in the application of procrastinator on-line scheduling [1]. To the best of our knowledge, multiagent on-line scheduling with the trade-off discussed above is not studied by previous researches such as time-dependent scheduling [6–9], on-line stochastic optimization [3, 4, 10, 11], and stochastic resource allocation in a multiagent environment [12–15]. Therefore, in this paper, we consider the above trade-offs in a multiagent scheduling process. There are several independent agents that can be scheduled to process the stochastically arriving jobs. Each job has a specific feasible time window, during which an agent can process it with time-dependent success probability. In case of fail, an agent will immediately make another try as long as the remaining time window allows. The objective is to complete all jobs with high probability. A general problem definition is introduced in Section 2, and Section 3 surveys closely related studies. Section 4 builds a DEC-MDP (decentralized Markov Decision Processes) to model the on-line multiagent scheduling process without considering the opportunity loss. An OL-DEC-MDP (opportunity loss-decentralized Markov Decision Processes) model is proposed in Section 5 to include the opportunity loss in the scheduling decision with proofs on its properties. Section 6 is the simulation evaluation of the OL-DEC-MDP, and Section 7 contains the conclusions and the future work.

2. Problem Definitions

There is a group of agents, denoted as I, that should be scheduled to process a set of stochastically arriving jobs, which is denoted as J. For each job $j \in J$, there is a time interval $T_j = [t_j^l, t_j^u]$, during which the process of job j is feasible. For example, t_j^l and t_j^u are the low bound and upper bound of the feasible time window for job j to be processed. For each agent $i \in I$, there is a duration $d_{ij}(t)$, which should be spent to process job j for one time starting from time t. The outcome of the process by the end of $d_{ij}(t)$ is either success or fail, and probability of success is denoted as $p_{ij}(t)$.

Assumption 1. One agent can only be scheduled to process one job at a time.

Assumption 2. If the agent fails to complete the job by the end of the process, it will immediately start another try as long as the feasible processing time window of the job will not elapse before the next try can be finished.

Assumption 3. The agent will be released from the current job and be available for the next job, if either the current job is completed successfully or the current job is discarded because of insufficient time window left for another try.

According to the above assumptions, an agent i will have several opportunities to complete a job j before the feasible time window $[t_j^l, t_j^u]$ of job j elapses depending on the process duration $d_{ij}(t)$ of each try. For example, If a process starting from time t_0 fails by the end of time $t_0 + d_{ij}(t_0)$, agent i must try to reprocess the job immediately at time $t_0 + d_{ij}(t_0)$, as long as $t_0 + d_{ij}(t_0) + d_{ij}(t_0 + d_{ij}(t_0)) \le t_j^u$.

Assumption 4. For an assignment of job j to agent i, later a try of process starts; later the process will end, but the process duration will be shorter.

Assumption 4 is in accordance with the observations in the time-sensitive applications such as air defense. With the threat approaching, the time needed for an interception is diminishing. For example, the earlier a process starts, the earlier the effect can be observed, but a longer duration needs to be spent. According to Assumption 4, For any t_1, $t_2 \in [t_j^l, t_j^u]$, s.t. $t_1 + d_{ij}(t_1) \in [t_j^l, t_j^u]$, $t_2 + d_{ij}(t_2) \in [t_j^l, t_j^u]$, if $t_1 < t_2$, then $d_{ij}(t_1) > d_{ij}(t_2)$ and $t_1 + d_{ij}(t_1) < t_2 + d_{ij}(t_2)$.

Assumption 5. Each agent operates independently, and there is no resource competition or mutual influence between agents.

The objective is to schedule the agents on-line to successfully complete all the arriving jobs with highest probability. In the off-line case, the objective of the problem can be modelled as (1)

$$\max \prod_{i \in I} \left(1 - \prod_{j \in J} \left(1 - p_{ij}(t)\right)^{u_{ij}(t)} \right) \tag{1}$$

s.t.

$$u_{ij}(t) = \begin{cases} 1 & \text{if agent } i \text{ starts to process} \\ & \text{or reprocess job } j \text{ at time } t \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

$$\sum u_{ij}(t) \ge 0,$$

$$t \in \left[t_j^l, t_j^u\right], \quad t + d_{ij} \in \left[t_j^l, t_j^u\right], \quad \text{for any } u_{ij}(t) = 1; \tag{3}$$

objective function (1) is to maximize the probability of successfully completing all the incoming jobs. Constraint (2) implies that an agent can process a job more than one time.

Constraint (3) ensures that a try of job processing should not be started if the feasible time window of the job will elapse before a try can be finished.

Compared with a similar model in Karasakal et al. [2], the starting time of job processing in the above model is continuously distributed in a job's feasible time window. Moreover, in the on-line version, the future arriving jobs could not be known in advance, which makes the model more difficult to solve.

As a result, trade-off should be made during the on-line scheduling of the problem to ensure good scheduling quality:

(1) the trade-off between the probability of successful process of the current job and the probability of the successful reprocess of the job in case of fail;

(2) the trade-off between the reward of successful process of the current job and the opportunity loss that the agent might have with other incoming jobs during the current job processing.

3. Related Works

3.1. Scheduling. Job scheduling [16] is a classic domain to solve the problem, in which jobs need to be handled by one ore more machines regarding the constraints of due date, processing time, priorities, and so forth. There are many different models such as single or parallel machine model depending on the number of machines. If a job needs to be handled by a series of machines in-order, the models are called flow shops, job shops, or open shops under different situations. The objective is to handle all the jobs with a minimum makespan [17] or lateness [18, 19]. Time-independent uncertainties such as machine breakdowns, unexpected releases of jobs with high priority [16], duration of a processing [20], and execution uncertainty [21] are introduced in the scheduling, which are called stochastic scheduling. Recently, models on time-dependent scheduling are proposed, in which parameters of the scheduling are time-dependent. For example, learning effect and processing time are defined as increasing funtion [22, 23] or deceasing function [1, 24, 25] of their start times. However, most of the above studies are discussed in the off-line case, where all of the jobs exist from the beginning. Moreover, time-dependent parameters mainly focus on the processing times or the cost of processing, while time-dependent success probability of job processing is not discussed.

3.2. On-Line Stochastic Optimization. On-line stochastic optimization, such as the on-line packet scheduling, stochastic reservations, vehicle dispatching, or routing, has been studied [3, 4], in which a job or requisition arrives stochastically in queue to wait for a certain machine or a sever to be served. A job or requisition will be successfully processed once a machine is scheduled. Scheduling can be centralized or decentralized depending on whether the scheduling decision is made globally or by each agent. To model real-world problems, time-independent uncertainties such as action duration [11], resource consumption [10], and operation outcomes [12, 14] are introduced into the scheduling process. Sampling approach is also introduced to estimate the future arriving jobs to achieve a global optimal solution [26–28]. Similar to our problem, each agent of the above problem is dynamically scheduled against the stochastically arriving jobs, and there is no resource competition or dependences among the agents. However, time-dependent probability of success is not considered, and each job will be successfully processed once an agent is scheduled to process it. Moreover, job discarding is allowed if a more important job is arriving. Instead, in our problem, each agent should retry in case of fail as long as the time allows.

3.3. Stochastic Resource Allocation in Multiagent Environment. The most related studies in the area of stochastic resource allocation in multiagent environment mainly focus on the following problems; each agent can execute a task independently while different agents may share the same resources. An agent consuming shared resources may decrease the reward of other agents. As the outcome of the job execution is uncertain, the resources are allocated to achieve the global optimal solution. [12] solves this type of problems by introducing dynamic constraint satisfaction problem (DCSP) model into MDP and constructing a Markovian CSP (MaCSP) model. The best action at each Markovian step depends on the resource availability. As the state space increases exponentially with the number of agent and the types of resource, some studies propose heuristic search [29] and decomposition approaches [14, 15] in solving Decentralized Markov Decision Processes (DEC-MDP). As the dependency between different agents is taken into account, starting an action too early or too late by an agent may jeopardize the operation of others. Hence, trade-off is introduced into DEC-MDP to estimate the cost that one agent may suffer due to the negative influence of others [13].

Comparing between our problems with closely related studies is listed in Table 1.

4. Decentralize MDP (DEC-MDP)

In theory, models (1)–(3) can obtain the optimal scheduling solution for an off-line problem. However, in the on-line case, the scheduling decision should be made according to the state of each agent and the incoming job in real time. As a result, MDP provides a suitable approach to model the on-line scheduling by mapping the current state of agents and incoming jobs to an optimal scheduling decision. In order to construct the MDP model of a problem, the state space of the problem should be defined.

4.1. States of the Agent and the Job. The state of an agent is either *busy* when it is processing a job or *unemployed* when it is released from the current job. Let $s_a^i(t)$ denote the state of agent i at time t:

$$s_a^i(t) = \begin{cases} \text{busy} & \text{if agent } i \text{ is processing a job} \\ & \text{at time } t \\ \text{unemployed} & \text{otherwise.} \end{cases} \quad (4)$$

Table 1: Comparing of the closely related works.

	Stochastic jobs or tasks	Multiagent (machine)	Dependences or resource competition among agents	Time-dependent processing time or action duration	Due date for jobs	On-line sampling	DEC-MDP	Time-dependent probability of success
Time-dependent scheduling		√		√	√			
On-line stochastic optimization	√	√			√	√		
Stochastic resource allocation of multiagent		√	√		√		√	
OL-DEC-MDP model based on-line scheduling	√	√		√	√	√	√	√

If a job is being processed by an agent, the state of the job is modelled as the ratio of the remaining time window feasible for the job to be completed. If it is waiting to be processed, its state is set to be 0; if it has been completed successfully, the state is set to be 1; otherwise the state is set to be −1. Let $s_m^j(t)$ denote the state of job j at time t:

$$s_m^j(t) = \begin{cases} 0 & \text{if } t < t_j^l \text{ or } j \text{ has not been scheduled to any agent;} \\ \dfrac{t - t_j^l}{t_j^u - t_j^l} & \text{if } t \in \left[t_j^l, t_j^u\right), \text{ and } j \text{ is being processed by an agent} \\ 1 & \text{if } t \geq t_j^l \text{ or } j \text{ has been completed;} \\ -1 & \text{if } j \text{ has been discarded.} \end{cases} \tag{5}$$

For example, at time 0s, there is an unemployed agent i without any job coming. The state of the agent i at 0s is

$$s_a^i(0) = \text{unemployed}. \tag{6}$$

At time 10s, a job j arrives with feasible time window [12, 30], and it is scheduled to agent i which is due to start at time 12s. Then

$$\begin{gathered} s_a^i(t) = \text{unemployed}, \quad s_m^j(t) = 0 \quad \text{for } t \in [10, 12) \\ s_a^i(t) = \text{busy}, \qquad s_m^j(t) = \frac{t - 12}{30 - 12} \\ \text{for } t \in \left[12, 12 + d_{ij}(12)\right). \end{gathered} \tag{7}$$

By Assumption 4, there is $12 + d_{ij}(12) \leq 30$. Suppose $d_{ij}(12) = 8$, and job processing ends successfully; then

$$s_a^i(20) = \text{unemployed}, \qquad s_m^j(20) = 1. \tag{8}$$

If job processing fails by the time 20s, agent i will start another try to process job j immediately as long as $20 + d_{ij}(20) \leq 30$ holds. Suppose $d_{ij}(20) = 9$; then

$$\begin{gathered} s_a^i(t) = \text{busy}, \qquad s_m^j(t) = \frac{t - 12}{30 - 12} \\ \text{for } t \in \left[20, 20 + d_{ij}(20)\right). \end{gathered} \tag{9}$$

If job processing fails again by the end of the second try (e.g., at time 29s), and the remaining time window of job j (only 1 second is left, since $t_j^u = 30$) is not enough for another try, then job j will be discarded from time 29s:

$$s_a^i(t) = \text{unemployed}, \quad s_m^j(t) = -1 \quad \text{for } t > 29. \tag{10}$$

In this case, agent i will be available for other incoming jobs from time 29s. As the different agents may be released or start to process a job at different times, it is hard to define a joint action, which is the set of actions for each agent in each decision step of the on-line scheduling process [13]. Moreover, because of the time-dependent state space, the reward of a joint action is difficult to evaluate by a recursive approach as introduced in [30]. Recently, in order to limit the set of state space in the multiagent environment, there is significant progress in extending the Markov Decision Processes (MDP) for optimizing decentralized control [13, 31]. In this paper, as there is no dependence or resource competition among agents, a decentralized MDP is adopted to model the decision process of each agent.

4.2. DEC-MDP. For an agent i and its allocated job j with time window $[t_j^l, t_j^u]$, the corresponding DEC-MDP is defined as a tuple $\langle s_{ij}, A_{ij}, p_{ij}, r_{ij} \rangle_{t_1}$, where

(i) $s_{ij} = (s_a^i, s_m^j)$ is the state set regarding agent i and job j during the whole process period (e.g., from the time

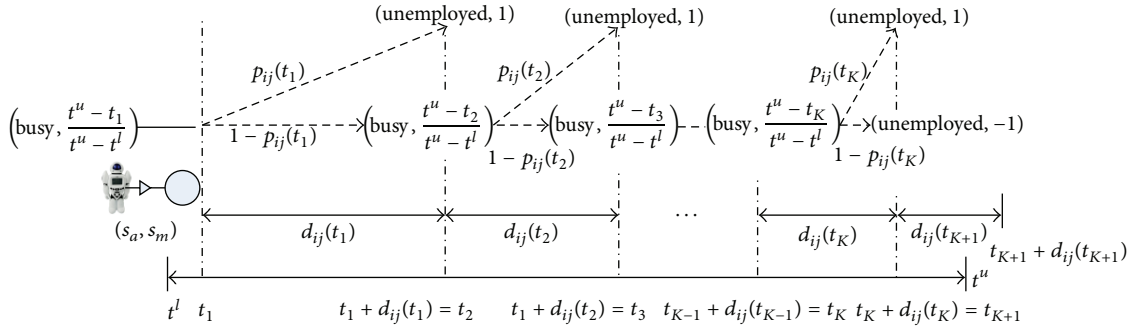

Figure 2: The state transition process of MDP $\langle s_{ij}, A_{ij}, p_{ij}, r_{ij}\rangle_{t_1}$.

when the process is started for the first time, denoted as t_1, to the time when agent i is released from job j);

(ii) A_{ij} is the strategy set of agent a_i, represented by the starting time that agent i may choose to process job j for the first time. $A_{ij} = \{t \mid t \in [t_j^l, t_j^u], t + d_{ij}(t) \in [t_j^l, t_j^u]\}$;

(iii) $p_{ij}(t)$ is a function of time t, which gives the success probability of completing job j by agent i by the time of $t + d_{ij}(t)$, when the process or the reprocess starts at time t. According to Assumptions 2 and 3, there is $p_{ij}(t) = 0$ if $t \notin [t_j^l, t_j^u]$ or $t + d_{ij}(t) \notin [t_j^l, t_j^u]$;

(iv) $r_{ij}(t)$ is the reward function as defined in (13).

The initial state of a DEC-MDP is

$$s_{ij}(t_1) = \left\{ s_a^i(t_1) = \text{busy}, s_m^j(t_1) = \frac{\left(t_1 - t_j^l\right)}{\left(t_j^u - t_j^l\right)} \right\}. \quad (11)$$

The absorbing state is

$$s_{ij}(\cdot) = \left\{ s_a^i(\cdot) = \text{unemployed}, s_m^j(\cdot) = 1 \text{ or } -1 \right\}. \quad (12)$$

Figure 2 shows the state transition process when agent a_i starts to process job j at time t_1 for the first time, in which the maximal retry times are K, and K is the smallest number that satisfies $t_{K+1} + d_{ij}(t_{K+1}) > t_j^u$. The reward function is defined as in (13), which is represented by the probability that agent i will successfully complete job j when starting the first process at time t_1. Consider

$$r_{ij}(t_1) = p_{ij}(t_1) + \sum_{x=2}^{K} \left(\prod_{y=1}^{x-1} \left(1 - p_{ij}\left(t_y\right)\right) \right) \cdot p_{ij}(t_x) \quad (13)$$

s.t.

$$t_x \in \left[t_j^l, t_j^u\right], \quad t_{x+1} = t_x + d_{ij}(t_x), \quad t_{x+1} \in \left[t_j^l, t_j^u\right], \quad x = 1, 2, \ldots, K \quad (14)$$

$$t_{K+1} + d_{ij}(t_{K+1}) > t_j^u; \quad (15)$$

when starting from time t_1, agent i can process job j not more than K times. With (13)–(15), the best time for agent i to begin to process job j is t_1^*:

$$t_1^* = \arg\max_{t \in A_{ij}} r_{ij}(t). \quad (16)$$

Therefore, during the on-line scheduling, we prefer scheduling the incoming job j to an agent i^* with highest success probability:

$$i^* = \arg\max_{i \in I} r_{ij}\left(t_1^*\right). \quad (17)$$

However, as stated in Section 4, this decision does not take the opportunity loss into account. Agent may lose higher rewards with upcoming jobs during its engagement with the current job. As a result, we introduce a opportunity loss decentralized MDP (OL-DEC-MDP) model in the next section.

5. Opportunity Loss Decentralized MDP (OL-DEC-MDP)

An OL-DEC-MDP model has the same state space, strategy set, and transition probability with a DEC-MDP. However, the reward function of an OL-DEC-MDP should be redefined to take the opportunity loss into account.

5.1. Opportunity Loss. As shown in Figure 2, the agent may try at most K times before being released from the current job. It will not be available for other upcoming jobs during the period $\Delta T_{ij}^z(t_1) = [t_1, t_z + d_{ij}(t_z)]$ $(1 \le z \le K)$ with the probability

$$p_{ij}^z = \begin{cases} 1 & z = 1 \\ \prod_{x=1}^{z-1} \left(1 - p_{ij}\left(t_x\right)\right) & 2 \le z \le K, \end{cases} \quad (18)$$

where $t_{x+1} = t_x + d_{ij}(t_x)$ and $x = 1, 2, \ldots, z-1$. As a result, agent will lose all upcoming jobs during $\Delta T_{ij}^z(t_1)$ with probability of p_{ij}^z. Hence the opportunity loss for agent i to process job j starting from t_1 can be defined as

$$\text{OL}(i, j, t_1) = \max_{1 \le z \le K} \left(p_{ij}^z \max_{q \in M\left(\Delta T_{ij}^z(t_1)\right)} r_{iq}\left(t_1^q\right) \right). \quad (19)$$

In the above equation, t_1^q is the best starting time for agent i to process job q, which is decided by (16). $M(\Delta T_{ij}^z(t_1))$ is the set of all possible jobs that will arrive during period $\Delta T_{ij}^z(t_1)$. The opportunity loss of the agent is defined to be the highest possible reward that the agent may lose during the period of engagement with its current job. Considering both the reward and the potential loss in scheduling decision, we now refine the reward function in OL-DEC-MDP as following:

$$r'(i,j,t_1) = r_{ij}(t_1) - \mathrm{OL}(i,j,t_1). \tag{20}$$

As a result, the best starting time of agent i to process job j is t_1^*:

$$t_1^* = \arg\max_{t\in A_{ij}} r'(i,j,t). \tag{21}$$

Reward function (21) calculates the maximum reward when schedule agent i to process job j while taking the opportunity loss into account.

5.2. Computation of the Reward with Opportunity Loss. For an OL-DEC-MDP $\langle s_{ij}, A_{ij}, p_{ij}, r'_{ij}\rangle_{t_1}$, if job j is allocated to agent i, then agent i should choose a starting time t_1 from the time window $[t_j^l, t_e]$, while $t_e + d_{ij}(t_e) = t_j^u$. If derivation of the reward function (21) exists within interval $[t_j^l, t_e]$, the optimal starting time t_1^* can be decided as following:

$$\frac{\partial\left[r_{ij}(t_1) - \mathrm{OL}(i,j,t_1)\right]}{\partial t_1} = 0 \quad \left(t_1 \in \left[t_j^l, t_e\right]\right). \tag{22}$$

However, if derivation of reward function (21) does not exist within interval $[t_j^l, t_e]$, t_1 can be decided with the following heuristics.

(1) Start as early as possible. For example, set $t_1 = t_j^l$ if the agent is available earlier than t_j^l, or set t_1 to be the earliest time when agent i becomes available. We denote t_1 under this heuristic as t_a, $t_a \in [t_j^l, t_e]$.

(2) Start as late as possible, as long as agent will still have the same number of retrying opportunities (K) with starting as early as possible (e.g., at time t_a). With this heuristic, there are $t_{x+1} = t_x + d_{ij}(t_x)$ $(x = 1, 2, \ldots, K-1)$ and $t_K + d_{ij}(t_K) = t_j^u$. We denote t_1 under this heuristic as t'.

(3) Start at the time with highest success probability to complete job j within the first try, while still having the maximal retrying opportunities (K). For example, t_1 is the time point between $[t_a, t']$ with the highest success probability of the first try. With this heuristic, there is $t_1 = \arg\max_{t\in[t_a,t']} p_{ij}(t)$. We denote t_1 under this heuristic as t_m.

(4) Start at the time with the highest success probability to complete the job j within the first try. With this heuristic, there is $t_1 = \arg\max_{t\in[t_a,t_e]} p_{ij}(t)$, while $t_e + d_{ij}(t_e) = t_j^u$. We denote t_1 under this heuristic as t_M.

As a result, the best starting time t_1^* of agent i to process job j considering both rewards and opportunity loss can be computed as

$$t_1^* = \arg\max_{t_1\in\{t_a,t',t_m,t_M\}} \left[r_{ij}(t_1) - \mathrm{OL}(i,j,t_1)\right]. \tag{23}$$

According to (19), to compute $\mathrm{OL}(i,j,t_1)$, we should know $M(\Delta T_{ij}^K(t_1))$; for example, the set of all possible jobs that will arrive during time $\Delta T_{ij}^K(t_1)$. $M(\Delta T_{ij}^K(t_1))$ can be estimated on-line by the sampling approach, as described in [26, 27], which can forecast the possible events according to the job arriving distribution.

5.3. On-Line Scheduling Based on OL-DEC-MDP. The detailed scheduling algorithm is given as following.

(1) Queuing Up the Incoming Jobs. When a new job comes, it is queued up in a time-priority queue. A new arrived job with a smaller low bound of feasible time will have higher priority.

(2) Observing the System State Change. Each agent has a job list with length of 1, which indicates its next job to be processed. System state changes when

(a) agent is released from current job and starts to process the assigned job in its job list (the agent's job list will be empty);

(b) agent fails to complete the current job and begins to make another try (the job in the agent's job list will be still waiting, which will be rescheduled);

(c) a new job is coming, and there exists at least one agent with empty job list (the incoming job will be assigned to some agent by being pushed into its job list).

(3) Scheduling/Rescheduling When System State Changes. When system state changes, scheduling or rescheduling decision will be made to decide or adjust the best next job as well as the best starting time for each agent. J_L denote the current set of next job of all agents before rescheduling.

(a) If $\|J_L\| < \|I\|$, then dequeue $\|I\| - \|J_L\|$ jobs from queue. Let J_n be the set of these dequeued jobs. Then, $J_L \cup J_n$ is the job set to be scheduled/rescheduled, as denoted by J_s.

(b) Order jobs in J_s according to time priority, and clear the job list of all agents.

(c) Schedule each job by order in J_s to an agent as following.

(i) Given job j, it will be scheduled to the agent i^* with highest reward:

$$i^* = \arg\max_{i\in I_A} r'(i,j,t_1^*). \tag{24}$$

In the above computation, I_A is the set of all agents with empty job list; t_1^* is decided by (23) with given i and j. For each agent, If it

has not been released when job j comes, its earliest available time t_a is set to be the ending time of its current process cycle. For example, the scheduling decision is made based on the assumption that all agents will be released by the end of its current process cycle. If the assumption is violated according to the observation, it is thought to be a system state change, and rescheduling will be made as described in step 3.

(ii) Push job j into job list of i^*.

(4) Job Processing. when available (being released from the current job), agent will begin to process the job in its job list at time t_1^*, and its job list will be cleared. By Assumption 2, agents will try many times to complete assigned jobs before success or time window of the job expires.

5.4. Properties and Proofs

Property 1. The time complexity to compute the best starting time t_1^* for an agent i to process job j according to (21) is $O(t_j^u - t_j^l)$.

Proof. As shown in Figure 2, there is one state for an OL-DEC-MDP in t_1, and two possible states in t_k, where $k = 2, 3, \ldots, K, K+1$. Hence, the maximum number of possible states during the job process is $2K+1$. As a result, the time complexity to calculate the expectation value of completing current job is $O(2K+1)$. On the other hand, the time complexity to calculate the maximal possible opportunity loss is $O(|M(\Delta T_{ij}^K)| \cdot (2K+1))$. Therefore, the time complexity to calculate the reward for a given agent i, job j, and start time t_1 is $O((|M(\Delta T_{ij}^K(t_1))|+1) \cdot (2K+1))$. As $|M(\Delta T_{ij}^K(t_1))|$ and K are constant for a given instance, we can set a constant $C = (|M(\Delta T_{ij}^K(t_1))|+1) \cdot (2K+1)$. Therefore, the time complexity to compute t_1^* in (21) is

$$O\left(\int_{t_j^l}^{t_j^u} C\right) = O\left(C \cdot \left(t_j^u - t_j^l\right)\right) = O\left(t_j^u - t_j^l\right). \quad (25)$$

□

Property 2. The average time for agent i on processing job j is $\sum_{z=1}^{K} p_{ij}^z \cdot (t_z + d_{ij}(t_z) - t_1)$, while $t_{z+1} = t_z + d_{ij}(t_z)$ and $z = 1, 2, \ldots, K-1$.

Proof. As shown in (17), given the starting time of the first processing t_1, the probability is p_{ij}^z for the agent i to spend $\Delta T_{ij}^z(t_1) = [t_1, t_z + d_{ij}(t_z)]$ on processing job j, where $t_{z+1} = t_z + d_{ij}(t_z)$ and $z = 1, 2, \ldots, K-1$. Thus, the average time for agent i to spend on job j is

$$\sum_{z=1}^{K} p_{ij}^z \cdot \left(t_z + d_{ij}\left(t_z\right) - t_1\right). \quad (26)$$

□

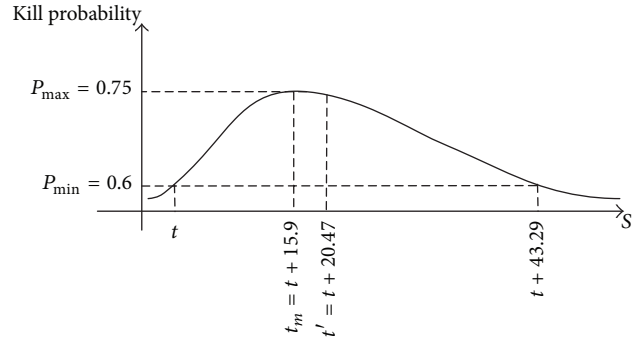

Figure 3: killing probability function.

6. Evaluations

6.1. Evaluation Setting. In the evaluation, a scenario of antiship missile defence by SAM systems is studied, which is introduced in [2].

Suppose there are four ship-borne SAM systems that can be scheduled to intercept the incoming antiship missiles, and each SAM system is capable of working independently and intercepting antiship missile coming from any direction (the modern ship-borne vertical launching missile system matches these features and is becoming very popular). The feasible interception time window of each incoming antiship missile j is set to be $[t_j^l, t_j + 43.29s]$, in which t_j^l is the time when the antiship missile is detected. The length of interception time window is decided by the detection capability of SAM system as well as the speed of the antiship missile. As a result, based on Section 5, if a SAM system is available when the missile is detected, there are $t_a = t_j^l$, $t_m = t_M = t_a + 15.9s$, and $t' = t_a + 20.47s$ based on scenario in [2]. The killing probability associated with the starting time of each interception $p_{ij}(t)$ is shown in Figure 3, which is approximated by a cubic multinomial. The duration function is defined as $d_{ij}(t) = r_{ij}(t)/(\nu_{\text{sam}} + \nu_m)$, where $r_{ij}(t)$ is the range at time t between the SAM missile and the antiship missile; ν_{sam} and ν_m are the velocities of the SAM missile and the antiship missile.

For the incoming antiship missile, the total number is set to be m ($m \in \{10, 12, 14, 16, 18\}$), and its arrival follows uniform distribution during a time span $[0, T]$ ($T \in \{20, 30, 40, 50, 60, 70, 80, 90\}$). To compute the opportunity loss as described in Section 5, a sampling method [26, 27] is implemented. Simulations are run under each combination of m and T to compare the scheduling result under different circumstances.

6.2. Quality of Scheduling. In the air-defence scenario, fail to intercept even one time may result in severe damage. Hence the quality of scheduling is measured by the probability of successfully interception of all incoming antiship missiles, which is denoted as *P-interception*. As shown in Figures 4 and 5, both DEC-MDP and OL-DEC-MDP based scheduling approaches illustrate that less intensive the attack comes (fewer antiship missiles with fixed time span or longer time

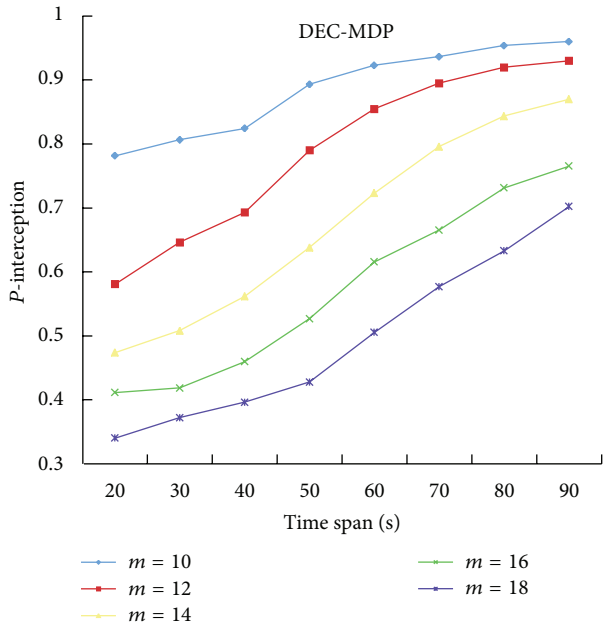

Figure 4: Probability of overall interception of DEC-MDP based approach.

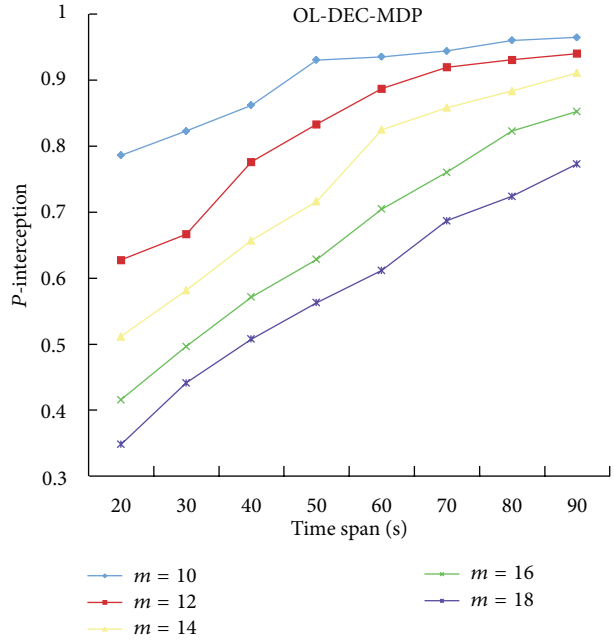

Figure 5: Probability of overall interception of OL-DEC-MDP based approach.

span with fixed incoming antiship missiles), higher the *P-interception* will be. The reason is that if there are fewer antiship missiles per time unit, there should be more available SAM systems that can be scheduled, hence the overall interception performance will be improved.

However, as shown in Figure 6, the OL-DEC-MDP based scheduling approach always has a higher probability of overall interception compared with DEC-MDP model. It can be observed that, for the same time span, the improvement of OL-DEC-MDP becomes more significant as the number of incoming missiles (m) increases. For example, there is performance improvement under more intensive attack environment. On the other hand, the overall shape of the improvement along the time span (for the fixed number of incoming antiship missiles, longer time span means less intensive attack) tends out to be a "cap." For example, the improvement rises sharply with the time span increasing at first and then comes down after reaching some peaks. The reason is that when the time span is small at first, which means that the antiship missiles are coming very intensively, it is very hard to improve the interception performance by OL-DEC-MDP since the SAM system reaches its saturation point under very intensive attack. The decision space left for each SAM system to decide the best starting time of the interception is quite small; hence OL-DEC-MDP has similar performance with DEC-MDP. However, when the intensity falls below the saturation point of the SAM system, the improvement brought by OL-DEC-MDP becomes gradually significant as opportunity loss is taken into account in on-line scheduling to achieve better overall performance. As the attack intensity continues to lower down with the increase of time span, the whole system has more than enough capability (available SAM systems) to intercept the incoming missiles; hence the improvement brought by OL-DEC-MDP becomes less significant.

Figure 7 shows the best starting time of interception used in OL-DEC-MDP obtained by heuristic strategy introduced in Section 5. Y-axis is the time indicating how long after an antiship missile is detected the first interception is launched. It can be observed that when the antiship missiles arrive intensively (which means smaller time span with fixed total incoming missiles), OL-DEC-MDP prefers to postpone the first interception launch. Study on the simulation data shows that the optimal starting time for interception under this case is near the time t_m, which means that the strategy to achieve the highest killing probability against the antiship missile by one shot is the superior strategy. This observation can be inferred from Property 2 that the superior strategy in this case is to release the SAM system as early as possible to treat the next incoming antiship missile. On the other hand, when the attack is less intensive (which means longer time span with fixed total incoming missiles), OL-DEC-MDP prefers to start the interception earlier as to leave more feasible time for retrying in case of interception fail.

7. Conclusions

This paper proposes an OL-DEC-MDP model for on-line multiagent stochastic scheduling, which considers the starting time-dependent probability of success and processing duration. The probability of completing the assigned job by an agent would be higher when the process is started earlier, but the opportunity loss could be also high due to the longer engaging duration. As a result, OL-DEC-MDP model introduces the reward function considering the opportunity

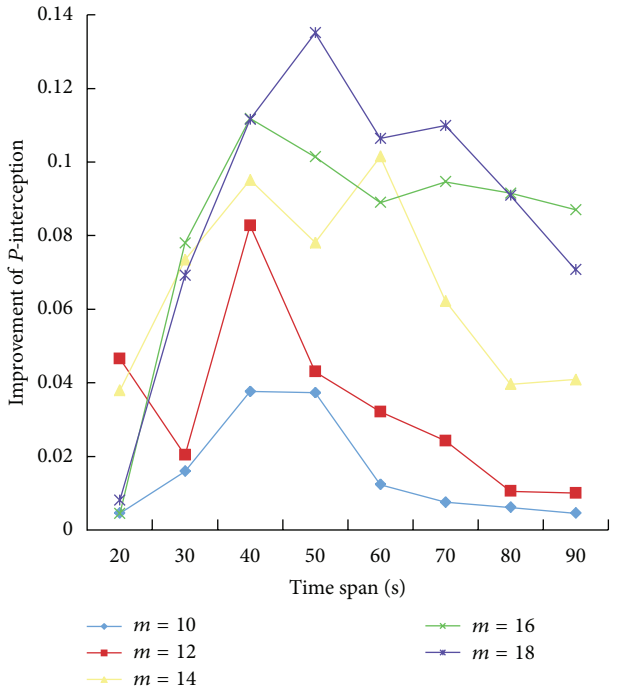

Figure 6: The increased probability of interception by OL-DEC-MDP compared with DEC-MDP.

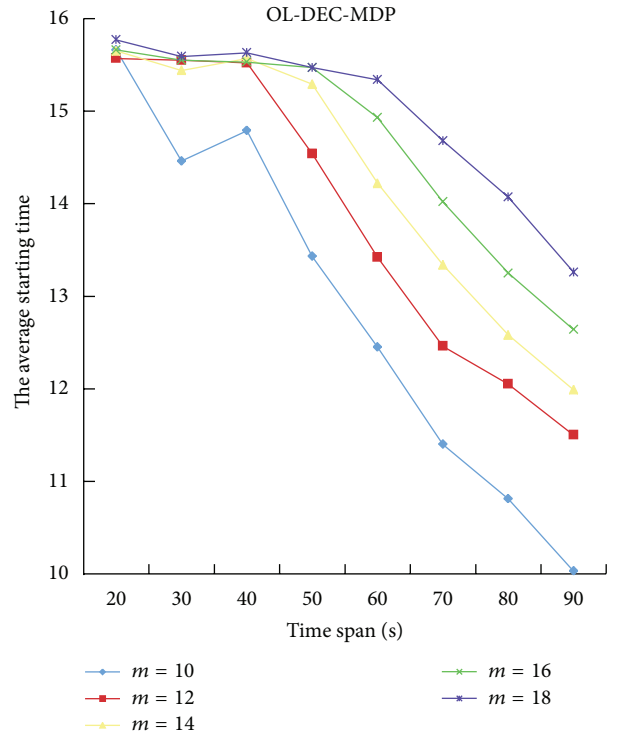

Figure 7: The average starting time of interception (t_1^*) computed by OL-DEC-MDP.

loss and schedules the incoming job to the agent with the highest reward. In order to measure the opportunity loss, OL-DEC-MDP model uses sampling method to predict the upcoming jobs and introduces heuristic strategies to compute the best starting time of an agent against an incoming job. The simulation experiments show that the OL-DEC-MDP model will improve the overall scheduling performance compared with models without considering opportunity loss, such as DEC-MDP. The overall trend of performance improvement is studied under different scenarios, which shows that the performance improvement is most significant if the jobs are coming intensively but within the saturation point of the multiagent system.

For the future research, we should extend the model to more general cases.

(1) *Dependency Between Agents*. In some cases, agents may interfere with other's operation. For example, if soft weapons such as chaff rocket are used during the interception, there may be mutual interference between different air defence weapons: firing a chaff rocket may prevent the missile guiding radar of the SAM system from working normally. In future work, the mutual influence between agents will be considered in constructing available strategy set and computing action reward.

(2) *Partial Observation*. For some real-world problem, the result of the action can only be partially observed. For example, the result of interception by a SAM system may not be totally observed by other agents due to the limitation of sensing capability. Hence the reward and the opportunity loss should be reevaluated, and POMDP (partial observation MDP) based approach could be a good candidate.

(3) *On-line Learning*. The sampling approach implemented in OL-DEC-MDP is based on the prior knowledge of the arrival distribution of the incoming jobs. If the prior knowledge of the arrival distribution does not exist, the on-line learning method could be used to learn and predict the future incoming jobs.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This paper is supported by the NSF of China under Grant nos. 61273322 and 71001105.

References

[1] M. A. Bender, R. Clifford, and K. Tsichlas, "Scheduling algorithms for procrastinators," *Journal of Scheduling*, vol. 11, no. 2, pp. 95–104, 2008.

[2] O. Karasakal, N. E. Özdemirel, and L. Kandiller, "Anti-ship missile defense for a naval task group," *Naval Research Logistics*, vol. 58, no. 3, pp. 304–321, 2011.

[3] P. Van Hentenryck and R. Bent, *Online Stochastic Combinatorial Optimization*, The MIT Press, Cambridge, Mass, USA, 2006.

[4] P. van Hentenryck, R. Bent, and E. Upfal, "Online stochastic optimization under time constraints," *Annals of Operations Research*, vol. 177, pp. 151–183, 2010.

[5] A. R. Benaskeur, F. Kabanza, and E. Beaudry, "CORALS: a real-time planner for anti-air defense operations," *ACM Transactions on Intelligent Systems and Technology*, vol. 1, no. 2, Article ID 1869402, p. 13, 2010.

[6] M. Debczynski and S. Gawiejnowicz, "An exact algorithm and a heuristic for scheduling linearly deteriorating jobs with arbitrary precedence constraints and the maximum cost criterion," in *Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS '12)*, pp. 401–405, September 2012.

[7] S. Gawiejnowicz, *Time-Dependent Scheduling*, Monographs in Theoretical Computer Science. An EATCS Series, Springer, Berlin, Germany, 2008.

[8] S. Gawiejnowicz and B. M. T. Lin, "Scheduling time-dependent jobs under mixed deterioration," *Applied Mathematics and Computation*, vol. 216, no. 2, pp. 438–447, 2010.

[9] S. Gawiejnowicz, W. Kurc, and L. Pankowska, "Equivalent time-dependent scheduling problems," *European Journal of Operational Research*, vol. 196, no. 3, pp. 919–929, 2009.

[10] C. Dance and A. A. Gaivoronski, "Stochastic optimization for real time service capacity allocation under random service demand," *Annals of Operations Research*, vol. 193, pp. 221–253, 2012.

[11] M. J. Sobel, J. G. Szmerekovsky, and V. Tilson, "Scheduling projects with stochastic activity duration to maximize expected net present value," *European Journal of Operational Research*, vol. 198, no. 3, pp. 697–705, 2009.

[12] C. Besse and B. Chaib-draa, "An efficient model for dynamic and constrained resource allocation problems," in *Proceedings of the 2nd International Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS '07)*, 2007.

[13] A. Beynier and A. Mouaddib, "Solving efficiently decentralized MDPs with temporal and resource constraints," *Autonomous Agents and Multi-Agent Systems*, vol. 23, no. 3, pp. 486–539, 2011.

[14] P. Plamondon and B. Chaib-Draa, "Stochastic resource allocation in multiagent environments: an approach based on distributed q-values and bounded real-time dynamic programming," *International Journal on Artificial Intelligence Tools*, vol. 21, no. 1, Article ID 1250003, 25 pages, 2012.

[15] P. Plamondon, B. Chaib-Draa, and A. R. Benaskeur, "A real-time dynamic programming decomposition approach to resource allocation," in *Proceedings of the IEEE Information, Decision and Control (IDC '07)*, pp. 308–313, February 2007.

[16] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Springer, New York, NY, USA, 2012.

[17] K. Fleszar, C. Charalambous, and K. S. Hindi, "A variable neighborhood descent heuristic for the problem of makespan minimisation on unrelated parallel machines with setup times," *Journal of Intelligent Manufacturing*, vol. 23, no. 5, pp. 1949–1958, 2012.

[18] M. A. González, C. R. Vela, I. González-Rodríguez, and R. Varela, "Lateness minimization with Tabu search for job shop scheduling problem with sequence dependent setup times," *Journal of Intelligent Manufacturing*, vol. 24, no. 4, pp. 741–754, 2013.

[19] M. D. ToksarI and E. Güner, "Parallel machine scheduling problem to minimize the earliness/tardiness costs with learning effect and deteriorating jobs," *Journal of Intelligent Manufacturing*, vol. 21, no. 6, pp. 843–851, 2010.

[20] J. Huang, G. A. Süer, and S. B. R. Urs, "Genetic algorithm for rotary machine scheduling with dependent processing times," *Journal of Intelligent Manufacturing*, vol. 23, no. 5, pp. 1931–1948, 2012.

[21] R. Rasconi, A. Cesta, and N. Policella, "Validating scheduling approaches against executional uncertainty," *Journal of Intelligent Manufacturing*, vol. 21, no. 1, pp. 49–64, 2010.

[22] M. Hosseinabadi Farahani and L. Hosseini, "Minimizing cycle time in single machine scheduling with start time-dependent processing times," *International Journal of Advanced Manufacturing Technology*, vol. 64, no. 9–12, pp. 1479–1486, 2013.

[23] W.-C. Lee, Y. S. Lin, and C.-C. Wu, "A branch-and-bound and heuristic algorithm for the single-machine time-dependent scheduling problem," *International Journal of Advanced Manufacturing Technology*, vol. 47, no. 9–12, pp. 1217–1223, 2010.

[24] D. Okołowski and S. Gawiejnowicz, "Exact and heuristic algorithms for parallel-machine scheduling with DeJong's learning effect," *Computers and Industrial Engineering*, vol. 59, no. 2, pp. 272–279, 2010.

[25] X.-Y. Wang, M.-Z. Wang, and J.-B. Wang, "Flow shop scheduling to minimize makespan with decreasing time-dependent job processing times," *Computers & Industrial Engineering*, vol. 60, no. 4, pp. 840–844, 2011.

[26] E. Burns, J. Benton, W. Ruml, S. Yoon, and M. B. Do, "Anticipatory on-line planning," in *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS '12)*, pp. 333–337, June 2012.

[27] H. S. Chang, R. Givan, and E. K. P. Chong, "On-line scheduling via sampling," in *Proceedings of the Conference on Artificial Intelligence Planning and Scheduling (AIPS '00)*, pp. 62–71, 2000.

[28] L. Mercier and P. van Hentenryck, "An anytime multistep anticipatory algorithm for online stochastic combinatorial optimization," *Annals of Operations Research*, vol. 184, pp. 233–271, 2011.

[29] N. Meuleau, E. Benazera, R. I. Brafman, E. A. Hansen, and M. Mausam, "A heuristic search approach to planning with continuous resources in stochastic domains," *Journal of Artificial Intelligence Research*, vol. 34, no. 1, pp. 27–59, 2009.

[30] R. Bellman, "A markovian decision process," DTIC Technical Document, DTIC, 1957.

[31] A. Beynier and A. Mouaddib, "A polynomial algorithm for decentralized Markov decision processes with temporal constraints," in *Proceedings of the 4th International Conference on Autonomous Agents and Multi agent Systems (AAMAS '05)*, pp. 963–969, July 2005.