*ECAI 2008*
*M. Ghallab et al. (Eds.)*
*IOS Press, 2008*
*© 2008 The authors and IOS Press. All rights reserved.*
*doi:10.3233/978-1-58603-891-5-583*



# A Simulation-based Approach for Solving Generalized Semi-Markov Decision Processes

**Emmanuel Rachelson**[1] and **Gauthier Quesnel** and **Frédérick Garcia** and **Patrick Fabiani**

**Abstract.** Time is a crucial variable in planning and often requires special attention since it introduces a specific structure along with additional complexity, especially in the case of decision under uncertainty. In this paper, after reviewing and comparing MDP frameworks designed to deal with temporal problems, we focus on Generalized Semi-Markov Decision Processes (GSMDP) with observable time. We highlight the inherent structure and complexity of these problems and present the differences with classical reinforcement learning problems. Finally, we introduce a new simulation-based reinforcement learning method for solving GSMDP, bringing together results from simulation-based policy iteration, regression techniques and simulation theory. We illustrate our approach on a subway network control example.

## 1 Introduction

Many problems in planning present both the features of decision under uncertainty and time-dependency. Imagine, for instance, having to plan the exploitation of a subway network, where available actions only consist in introducing or removing trains from service. In this problem, the goal is to maximize the number of passengers going through the network while minimizing the exploitation cost of the subway. Passenger arrival times, movements going in and out of the trains and possible delays in the system make the outcome of every action uncertain with regard to the next state and the date of the next decision epoch. On top of that, the flow of passengers and their destinations depend greatly on the time of day. All this defines the kind of problems we try to capture as Temporal Markov Problems. These problems cover a wide variety of other applications, as onboard UAV coordination or airport taxiway management, etc.

Problems of decision under uncertainty are commonly modelled as Markov Decision Processes (MDP). Recent work on solving large state-space MDP include, for example, factored MDP methods, approximate linear programming, hierarchical approaches, reinforcement learning, etc. Temporal Markov Problems, however, have received little attention from the planning and machine learning communities, even though simulation seems a promising approach to tackling these problems. This paper presents formalisation and algorithmic issues about Temporal Markov Problems and proposes a simulation-based algorithm designed to solve them. In section 2, we will review the models adapted from Markov Processes and designed to include time-dependency and decision making. Building on this first section's conclusions, we focus on controlling Generalized Semi-MDP (GSMDP). Section 4 presents our algorithm and discusses the issues and interests of simulation-based approaches for GSMDP. We illustrate our approach on the subway control example in section 4.3 and conclude in section 5.

[1] ONERA, France, email: emmanuel.rachelson@onera.fr

## 2 Temporal Markov Problems

MDP have become a popular model for describing problems of planning under uncertainty. Formally, an MDP is composed of a 4-tuple $\langle S, A, P, r\rangle$, where $S$ is a countable set of states for the system, $A$ is the countable set of possible actions, $P(s'|s,a)$ is a probability distribution function providing the transition model between states (as in a Markov Process, but conditioned with the action $a$) and $r(s,a)$ is a reward value associated with the $(s,a)$ transition, used to build criteria and to evaluate actions and policies. Solutions to MDP problems are often given as *Markovian policies* $\pi$, namely functions that map current states to actions. One can introduce criteria to evaluate these policies, as the discounted reward criterion given in equation 1. Criteria permit definition of the *value function* $V^\pi$ associated with a policy. An important result concerning MDP is that for any history-dependent policy, there exists a Markovian policy which is at least as good with regard to a given criterion. Consequently, one can safely seach for optimal control policies in the restricted space of Markovian policies without loss in optimality. Finally, algorithms as *value iteration* or *policy iteration* are based on the fact that the optimal policy's value function $V^*$ obeys Bellman's optimality equation 2 [1].

$$V_\gamma^\pi(s) = E\left(\sum_{\delta=0}^{\infty}\gamma^\delta r\left(s_\delta, \pi(s_\delta)\right)\right) \quad (1)$$

$$V^*(s) = \max_{a\in A}\left[r(s,a) + \gamma\sum_{s'\in S}P(s'|s,a)V^*(s')\right] \quad (2)$$

### 2.1 Including continuous time in the MDP framework

Introducing time in Markov Processes (MP) models — and in their decisional counterparts, MDP — can be done by defining stochastic durations between decision epochs. In a standard MP or MDP, the sojourn time in a given state is one and decision epochs occur at integer time values (thus yielding the $\gamma^\delta$ in the discounted criterion). Allowing the sojourn time in a given state to be continuous and stochastic defines the Semi-MP, or Semi-MDP formalism. In an SMDP [11], state sojourn time is described through a distribution $F(\tau|s,a)$ indicating the time before transition, provided that we undertake action $a$ in state $s$. Therefore, an SMDP is a 5-tuple $< S, A, P, F, r >$ which corresponds to a Markov Process but with stochastic state sojourn time. Policies for the control of SMDP can be computed using standard MDP algorithms since solving a discounted reward SMDP turns

out to be equivalent to performing an integration over expected transition durations and to solving a total reward MDP. This is mainly due to the independence between state sojourn time $\tau$ and arrival state $s'$. This very strong assumption was lifted in the Time-dependent MDP (TMDP) model of [2] and generalized recently in the XMDP model of [13]. Formally, an XMDP is described by a 4-tuple $< S, A, p, r >$ where the state space $S$ can be composed of discrete and continuous variables and may include the process' time, $A$ is a continuous or discrete parametric action space and $p$ and $r$ correspond to transition and reward models for states of $S$ and actions of $A$. [13] proved that XMDP obeyed a similar optimality equation as equation 4, thus proving that standard algorithms as value iteration could be safely used to solve XMDP. Using the XMDP representation, one can model any stochastic decision process with continuous observable time and hybrid state and action spaces.

This seems to suit our Temporal Markov Problems well and some recent techniques for solving hybrid state space MDP ([6, 4]) could be applied here. However, writing transition and duration functions for Temporal Markov Problems is often a very complex task and requires a lot of engineering. For instance, the effect of a $RemoveTrain$ action on the global state of the subway problem is the result of several concurrent processes : the passenger arrivals, the trains movements, the removal of one train, etc.: all compete to change the system's state and it is a complex task to summarize all these process' concurrent stochastic influence into the transition and duration functions.

## 2.2 Concurrency and MDP

In the stochastic processes litterature, concurrent Markov processes are modelled as Generalized Semi-Markov Processes (GSMP) [5]. A GSMP is a natural representation of several concurrent SMP affecting the same state space. [16] introduced Generalized Semi-Markov Decision Processes (GSMDP) in order to model the problem of decision under uncertainty where actions compete with concurrent uncontrollable stochastic events. A GSMDP describes a problem by factoring the global transition function of the process by the different stochastic contributions of concurrent events. This makes GSMDP an elegant and efficient way of describing the complexity of Markov Temporal Problems. We will therefore focus on solving time-dependent GSMDP from now on and will give a more formal definition of GSMDP in section 3.

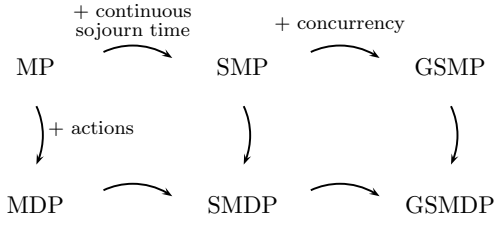


**Figure 1.** From MP to GSMDP

# 3 GSMP and GSMDP

The previous section illustrated how Temporal Markov Problems needed both continuous observable time models and an efficient representation of concurrency in order to represent the complexity of the phenomena at stake. In this section, we focus on the GSMDP formalism with observable time. We define control policies, the associated state variable issues and present resolution methods.

## 3.1 Concurrent processes

We start from the stochastic process point of view, with no decision making. Formally, a GSMP [5] is described by a set $S$ of states and a set $E$ of events. At any time, the process is in a state $s$ and there exists a subset $E_s$ of events that are called *active* or *enabled*. These events represent the different concurrent processes that compete for the next transition. To each active event $e$, we associate a clock $c_e$ representing the duration before this event triggers a transition. This duration would be the sojourn time in state $s$ if event $e$ was the only active event. The event $e^*$ with the smallest clock $c_{e^*}$ (the first to trigger) is the one that takes the process to a new state. The transition is then described by the transition model of the triggering event: the next state $s'$ is picked according to the probability distribution $P_{e^*}(s'|s)$. In the new state $s'$, events that are not in $E_{s'}$ are disabled (which actually implies setting their clocks to $+\infty$). For the events of $E_{s'}$, clocks are updated the following way:

- If $e \in E_s \setminus \{e^*\}$, then $c_e \leftarrow c_e - c_{e^*}$
- If $e \notin E_s$ or if $e = e^*$, pick $c_e$ according to $F_e(\tau|s')$

The first active event to trigger then takes the process to a new state where the above operations are repeated.

One first important remark concerning GSMP is that the overall process does not retain Markov's property anymore : knowing the current state $s$ is not sufficient to predict the distribution on the next state of the process. [9] showed that by augmenting the state space with the events' clocks, one could retain the Semi-Markov behaviour for a GSMP, we will discuss this issue in the next section.

Introducing action choice in a GSMP yields a GSMDP as defined by [16]. In a GSMDP, we identify a subset $A$ of controlable events or actions, the remaining ones are called uncontrolable or exogenous events. Actions can be enabled or disabled at will and the subset $A_s = A \cap E_s$ of activable actions is never empty since it always contains at least the "idle" action $a_\infty$ (whose clock is always set $\infty$) which, in fact, does nothing and lets the first exogenous event take the process to a new state. As in the MDP case, searching for control strategies on GSMDP imply defining rewards $r(s, e)$ or $r(s, e, s')$ associated to transitions and introducing policies and criteria.

## 3.2 Controling GSMDP

As mentionned before, the transition function for the global semi-Markov process does not retain the Markov property without augmenting the state space. In the classical MDP framework, one can make use of the Markov property of the transition function to prove that there exists a Markovian policy (which only depends on the current state) which is at least as good as any history-dependent policy [11]. In the GSMDP case however, this is no longer possible and in order to define criteria and to find optimal policies, we need - in the general case - to allow the policy to depend on the whole *execution path* of the process. An execution path [16] of length $n$ from state $s_0$ to state $s_n$ is a sequence $\sigma = (s_0, t_0, e_0, s_1, \ldots, s_{n-1}, t_{n-1}, e_{n-1}, s_n)$ where $t_i$ is the sojourn time in state $s_i$ before event $e_i$ triggers. As in [16], we define the discounted value of an execution path by:

$$V_\gamma^\pi(\sigma) = \sum_{i=0}^{n-1} \gamma^{T_i} \left( \gamma^{t_i} k(s_i, e_i, s_{i+1}) + \int_0^{t_i} \gamma^t c(s_i, e_i) dt \right) \quad (3)$$

where $k$ and $c$ are traditional SMDP lump sum reward and reward rate functions, and $T_i = \sum_{j=0}^{i-1} t_j$. One can then define the expected value of policy $\pi$ in state $s$ as the expectation over all execution paths starting in $s$: $V_\gamma^\pi(s) = E_s^\pi \left[V_\gamma^\pi(\sigma)\right]$.

This provides a criterion for evaluating policies. The goal is now to find policies that maximize this criterion. The main problem here is that it is hard to search the space of history-dependent policies. On the other hand, the supplementary variable technique is often used to transform non-Markovian processes into Markovian ones. It consists in augmenting the state space with just enough variables so that the distribution over future states only depends on the current value of these variables. In [9], Nielsen augments the natural state $s$ of the process with all the clock readings and shows that this operation brings Markov behavior back to the GSMP process. We will note this augmented state space $(s, c)$ for convenience.

Unfortunately, it is unrealistic to define policies over this augmented state space since clock readings contain information about the *future* of the system. From here, several options are possible:

- One could decide to sacrifice optimality and to search for "good" policies among a restricted set of policies, say the policies defined on the current natural state only.
- One could also search for representation hypotheses that simplify the GSMDP model and that make natural state Markovian again.
- One could compute optimal policies on the augmented state space $(s, c)$ and then derive a policy on observable variables only.
- Finally, one could search for a set of *observable* variables which retain the Markov property for the process, for example the set composed of the natural state of the process $s$, the duration for which each active event $e_i$ has been active $\tau_i$ and its activation state $s_i$. We will note this augmented state $(s, \tau, s_a)$

[16] is based on the second option listed above. In the next paragraph, we briefly present this approach and introduce our reinforcement learning method designed to deal with very large state spaces for GSMDP with continuous observable time and that can be adapted to the three other options.

### 3.3 Resolution methods

The resolution method for GSMDP proposed by [16] relies on the memoryless property of the exponential distribution. If one approximates all duration functions $F$ by phase-type distributions (which are combinations of exponential distributions), then augmenting the state space with the distribution phases brings the overall behaviour of the GSMDP back to a Continuous Time MDP, which can, in turn, be transformed to a standard discrete time MDP by the method of *uniformization* [11]. We refer the reader to [16] for more details.

We wish not make hypotheses on the distributions that describe the dynamics of our system. On top of that, many problems we want to consider present other characteristics such as very large, and sometimes continuous state spaces. Therefore, we need to consider methods for policy search that can cope with large hybrid state spaces (yielding large hybrid trajectory spaces) and observable time. Finally, for some aspects of the problems, the stochastic behaviour might still be very complex to model formally while simulators might be readily available (for instance, in the airport taxiway management problem, the weather model is not given as probability distribution functions but as a simulator). In order to deal with such problems we turn towards reinforcement learning methods. More specifically, in order to avoid complete state space exploration, we introduce a version of approximate policy iteration where policies are defined and evaluated on a subset of states and then generalized by regression to the whole state space. The choice of the subset of states used for evaluation is guided by the simulation of the current policy. We present our algorithm in section 4.1 and then illustrate why simulation-based policy iteration is particularly adapted to temporal problems in section 4.2.

## 4 Simulation-based approaches

### 4.1 Algorithm

Our algorithm belongs to the Approximate Policy Iteration (API) family of algorithms. Policy Iteration is an algorithm for solving MDP which searches the policy space in a two-step fashion as illustrated on figure 2. Given a policy $\pi_n$ at step $n$, the first step consists in computing the value of $\pi_n$. The second step then performs a Bellman backup in every state of the state space, thus improving the policy. An important property of policy iteration is its good anytime behaviour: at any step $n$, policy $\pi_n$ will be at least as good as any previous policy. Policy Iteration usually converges in less iterations than the standard Value Iteration algorithm but takes longer since the evaluation step is very time consuming. To deal with real problems, one needs to allow for approximate policy evaluation (as in [7]) since exact computation is often infeasible. There are few theoretical guarantees on convergence and optimality of API, as explained in [8].

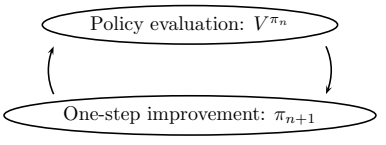


**Figure 2.** Policy Iteration

The version of simulation-based policy iteration we use performs simulations of the current policy $\pi_n$ starting from the current state of the process and stores the triplets of states, times and rewards $(s_\delta, t_\delta, r_\delta)$ obtained. Thus, one execution path yields a value function over the discrete set of states explored during simulation (equation 3). All the value functions issued from simulation form a training set $\{(s, v)\}$, $s \in S, v \in \mathbb{R}$, from which we wish to generalize a value function $\tilde{V}$ over all states. The average value of state $s$ in the training set tends to $V^{\pi_n}(s)$ as the number of simulations tends to $+\infty$. One major advantage of policy-driven simulation is that the policy guides the exploration of the state space to the states most likely to be visited, thus refining the training set over the states that have the largest probability of being reached by the policy. A second advantage is that this technique is adapted to large dimension state spaces.

Once simulation has provided the set of samples in the space of trajectories, we want to use it as a training set for a regression method that will generalize it to the entire state space. Several approaches to regression based reinforcement learning have been proposed in the machine learning community - methods based on trees [3], evolutionary functions [15], kernel methods [10], etc. - but few have been coupled with policy simulation. We chose to focus on support vector machines (SVM) because of their ability to handle the large dimension spaces over which our samples are defined. SVM belong to the family of kernel methods and can be used for both regression and classification. Training a standard SVM over a given training set corresponds to looking for a hyperplane interpolating the samples in a

higher dimensional space called *feature space*. Practically, SVM take advantage of the *kernel trick* to avoid expressing the feature space explicitely. For more details on SVM, we refer the reader to [14]. In our case, we call $\tilde{V}_n(s)$ the interpolated value function of policy $\pi_n$.

Finally, while simulation-based exploration and SVM generalization of the value function are techniques dedicated to improve the evaluation step of approximate policy iteration, the third specificity of our algorithm deals with improving the optimization step. For large and possibly continuous state spaces, it might be very long or impracticable to compute the one-step improvement of the policy. Indeed, most of the time, computing a complete policy is irrelevant since most of this policy will never be used for the simulation-based evaluation step. Instead, it might be easier to compute online the one-step lookahead best action in the current state with respect to the stored value function. More precisely, in a standard MDP, the optimization step consists in solving equation 4 in every state:

$$\pi_{n+1}(s) \leftarrow \arg\max_{a \in A} \tilde{Q}_{n+1}(s,a) \tag{4}$$
$$\text{with: } \tilde{Q}_{n+1}(s,a) = r(s,a) + \sum_{s' \in S} P(s'|s,a)\tilde{V}_n(s,a)$$

For continuous state spaces, computing $\pi_{n+1}$ implies being able to compute integrals over $P$ and $\tilde{V}_n$. We wish not make hypotheses on the model used and therefore will perform a discretization for evaluation of the integral. Finally, since the model of $P$ is not necessarily known to the decision maker and since we have a simulator of our system, we will make a second use of this simulator for the purpose of evaluating the expected reward $\tilde{Q}_{n+1}(s,a)$ associated with performing action $a$ in state $s$ with respect to value function $\tilde{V}_n$ (equation 5). At the end of the evaluation phase, the value function $\tilde{V}_n$ is stored and no policy is computed from it. Instead, we immediately enter a new simulation phase but whenever the policy $\pi_{n+1}$ is asked for the action to perform in the current state $s$ it performs *online* the estimation of all $Q$-values for state $s$ and then choses the best action to perform. The speed up in the execution of the policy iteration algorithm is easy to illustrate for discrete state spaces problems since we replace $|S|$ evaluations of the $Q$-values for policy update by the number of states visited during one simulation. This is especially interesting in the case of Temporal Markov Problems since (as we will explain in section 4.2) a state is never visited twice. Consequently, $\tilde{Q}_{n+1}(s,a)$ is calculated by simply simulating $N$ times the application of $a$ in $s$ and observing the set of $\{(r_i, s'_i)\}$ as in equation 5. Then the policy returns the action which corresponds to the largest $Q$-value. We call this online instanciation of the policy "online approximate policy iteration".

$$\tilde{Q}_{n+1}(s,a) = \frac{1}{N}\sum_{i=1}^{N}\left[r_i + \tilde{V}_n(s'_i)\right] \tag{5}$$

Our algorithm, called *online Approximate Temporal Policy Iteration* (online-ATPI), is summarized in algorithm 1.

Note that in algorithm 1, $s$ actually denotes the part of the state that is observable to the policy. This makes online-ATPI adaptable to any of the sets of policy variables presented in section 3.2. We tested a version of online-ATPI on the natural state of the process.

**Algorithm 1** Online-ATPI

**main:**
Input : $\pi_0$ or $\tilde{V}_0$, $s_0$
**loop**
  $TrainingSet \leftarrow \emptyset$
  **for** $i = 1$ to $N_{sim}$ **do**
    $\{(s,v)\} \leftarrow$ simulate$(\tilde{V}, s_0)$
    $TrainingSet \leftarrow TrainingSet \cup \{(s,v)\}$
  **end for**
  $\tilde{V} \leftarrow$ TrainApproximator$(TrainingSet)$
**end loop**

**simulate**$(\tilde{V}, s_0)$**:**
$ExecutionPath \leftarrow \emptyset$
$s \leftarrow s_0$
**while** horizon not reached **do**
  $action \leftarrow$ ComputePolicy$(s, \tilde{V})$
  $(s', r) \leftarrow$ GSMDPstep$(s, action)$
  $ExecutionPath \leftarrow ExecutionPath \cup (s', r)$
**end while**
convert execution path to value function $\{(s,v)\}$ (eqn 3)
**return** $\{(s,v)\}$

**ComputePolicy**$(s, \tilde{V})$**:**
**for** $a \in A$ **do**
  $\tilde{Q}(s,a) = 0$
  **for** $j = 1$ to $N_{samples}$ **do**
    $(s', r) \leftarrow$ GSMDPstep$(s,a)$
    $\tilde{Q}(s,a) \leftarrow \tilde{Q}(s,a) + r + \gamma^{t'-t}\tilde{V}(s')$
  **end for**
  $\tilde{Q}(s,a) \leftarrow \frac{1}{N_{samples}}\tilde{Q}(s,a)$
**end for**
$action \leftarrow \arg\max_{a \in A} \tilde{Q}(s,a)$
**return** $action$

## 4.2 Simulating GSMDP and learning

Simulation is a key aspect of ATPI. The Discrete EVents Simulation theory (DEVS) of [17] provides a general framework for specifying discrete event dynamic systems. We implemented GSMP and GSMDP extensions in the VLE multi-modeling platform [12] based on the DEVS specification; by doing so, we take advantage of the DEVS framework's properties which fit our simulation requirements, namely:

- Event driven simulation and time oriented output.
- The simulation engine deals with simultaneity issues and with simulation consistency and reproducibility.
- Simulation engines such as the VLE platform [12] are readily available and built on the same discrete events simulation theory.
- Multi-modelling possibilities, opens the algorithm to other formalisms than MP.

On top of that, the DEVS formalism allows for experimental frames definition, which would permit integration of the whole simulation and planning loop in a DEVS specification. We haven't used experimental frames yet but plan to do so in future versions.

Finally, we have claimed that Temporal Markov Problems present a specific structure that makes the problem both hard to deal with for classical reinforcement learning algorithms and particularly adapted for online approximate policy iteration. More specifically:

- Most reinforcement learning algorithms deal with discrete state spaces. Some approaches have been proposed ([10, 3, 6] for dealing with continuous or hybrid states but the topic is still very new. Often, continuous state resolution methods depend strongly on the

representation used and on the ability to calculate integrals over the probability functions. Simulation-based sampling approaches propose a different approach to this issue.

- When time is observable, the *causality principle* ensures that the process never goes back in time. This avoids loops and insures that online policy instanciation performs less operations than a complete offline policy improvement step.

### 4.3 Example

Table 1 presents optimization results for the first four iterations of online-ATPI for the subway problem initialized with a policy $\pi_0$ that sets trains to run all day long [2]. $N_{sim}$ was set to 20 and $N_{samples}$ to 15 with $\gamma = 1$ (finite horizon). This simple instance of the subway problem implied 4 trains and 6 stations. The problem's specification took time-dependency and stochastic behaviour into account; for example passenger arrival periods were represented using Gaussian distributions with means and standard deviations depending on the time of day. The state space for this problem included 22 discrete, boolean or continuous variables (including time), thus yielding a sample space of dimension 22 for the training set.

In table 1, $t_{sim}$ is the training set building time (which corresponds to performing the $N_{sim}$ simulations) while $t_{learn}$ is the SVM training time (in seconds). $\tilde{V}_{stat}(s_0)$ is the statistical evaluation of $\tilde{V}(s_0)$, while $\tilde{V}_{SVM}(s_0)$ is the value provided by the trained SVM. Lastly, #SV is the number of support vectors in the SVM.

The expected value of the initial state increases with iterations; this confirms the fact that policy quality improves with each iteration. This increase is not necessarily linear and depends on the problem's structure. If the policy takes the simulation to states that are "far" from explored states (states for which the interpolated value might be erroneous) and that provide very bad rewards, it can happen that the initial state's expected value drops for one iteration. This is the drawback from partial exploration of the state space and interpolation: very good or very bad regions of the state space might be discovered late in the iterations.

One can notice that simulation time increases with iterations. This is mainly due to the number of support vectors in the SVM. Depending on the iteration step, the SVM can be much simpler and simulation time can drop again. On the other hand, online-ATPI is still very sensitive to the initial policy and we are currently working on other possibilities to improve solution quality (such as roll-out techniques and estimator refinement during optimization by simulation-optimization interweaving).

**Table 1.** Subway control policy

| | $\pi_0$ | $\pi_1$ | $\pi_2$ | $\pi_3$ | $\pi_4$ |
|---|---|---|---|---|---|
| $t_{sim}$ | 47.1 | 203.43 | 206.45 | 446.15 | 1504.41 |
| $t_{learn}$ | 2.28 | 2.7 | 12.18 | 56.08 | 229.45 |
| $\tilde{V}_{stat}(s_0)$ | -3261.31 | -3188.11 | -2074.74 | -1850.12 | -887.076 |
| $\tilde{V}_{SVM}(s_0)$ | -2980.29 | -2962.46 | -2020.22 | -1837.41 | -875.417 |
| #SV | 55 | 61 | 439 | 3588 | 13596 |

Since $N_{sim} = 20$ simulations per iteration always provide a training set of around 45000 points for the SVM in the subway example, the number of support vectors for the SVM - and therefore, the iteration duration - is bounded. Longer runs on the subway problem show that the number of support vectors and learning time in column $\pi_4$ are a good estimate of the worst values.

[2] experiments were ran on a 1.7GHz single core processor with 1GB of RAM

## 5 Conclusion

This paper introduces a new reinforcement learning method for solving Generalized Semi-Markov Decision Processes. These processes are a natural and elegant way of representing the complexity of concurrent stochastic processes. In the framework of time-dependent GSMDP with explicit time, simulation seems to be an efficient way of exploring the state space and evaluating strategies. Drawing from this idea, we introduced a simulation-based version of Approximate Policy Iteration (API), which we called online-ATPI. This algorithm incrementally improves the quality of an initial policy by making use of simulation-based evaluation, SVM regression and online policy instanciation. Although there are few theorical results concerning the convergence and optimality of API, online-ATPI seems to perform well on an example of subway network control.

Future work will deal with making online-ATPI more robust to initialization; in fact, if the initial policy does not guide the simulation towards relevant areas of the state space, the error in policy evaluation can greatly penalize the algorithm. To avoid this drawback, we plan to use incremental refining methods for simulation initialization. This could result in building a more dense training set, therefore minimizing the risk of not exploring relevant parts of the state space.

## REFERENCES


[1] R. E. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, New Jersey, 1957.

[2] J. Boyan and M. Littman, 'Exact solutions to time dependent MDPs', *Advances in Neural Information Processing Systems*, **13**, 1026–1032, (2001).

[3] D. Ernst, P. Geurts, and L. Wehenkel, 'Tree-based batch mode reinforcement learning', *JMLR*, **6**, 503–556, (2005).

[4] Z. Feng, R. Dearden, N. Meuleau, and R. Washington, 'Dynamic programming for structured continuous markov decision problems', in *20th Conference on Uncertainty in AI*, pp. 154–161, (2004).

[5] P. Glynn, 'A GSMP formalism for discrete event systems', *Proc. of the IEEE*, **77**, (1989).

[6] M. Hauskrecht and B. Kveton, 'Approximate linear programming for solving hybrid factored MDPs', in *9th Int. Symp. on AI and Math.*, (2006).

[7] M. Lagoudakis and R. Parr, 'Least-squares policy iteration', *JMLR*, **4**, 1107–1149, (2003).

[8] R. Munos, 'Error bounds for approximate policy iteration', in *Int. Conf. on Machine Learning*, (2003).

[9] F. Nielsen, 'GMSim: a tool for compositionnal GSMP modeling', in *Winter Simulation Conference*, (1998).

[10] Dirk Ormoneit and Saunak Sen, 'Kernel-based reinforcement learning', *Machine Learning*, **49**, 161–178, (2002).

[11] M. Puterman, *Markov Decision Processes*, John Wiley & Sons, Inc, 1994.

[12] G. Quesnel, R. Duboz, É. Ramat, and M.K. Traore, 'VLE - A Multi-Modeling and Simulation Environment', in *Moving Towards the Unified Simulation Approach, Proc. of the 2007 Summer Simulation Conf.*, pp. 367–374, (2007).

[13] E. Rachelson, F. Garcia, and P. Fabiani, 'Extending the Bellman equation for MDP to continuous actions and continuous time in the discounted case', in *10th Int. Symp. on AI and Math.*, (2008).

[14] V. Vapnik, S. Golowich, and A. Smola, 'Support vector method for function approximation, regression estimation and signal processing', *Advances in Neural Information Processing Systems*, **9**, 281–287, (1996).

[15] Shimon Whiteson and Peter Stone, 'Evolutionary function approximation for reinforcement learning', *JMLR*, **7**, 877–917, (2006).

[16] H. Younes and R. Simmons, 'Solving Generalized semi-Markov Decision Processes using Continuous Phase-Type Distributions', in *AAAI*, (2004).

[17] B. P. Zeigler, D. Kim, and H. Praehofer, *Theory of modeling and simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*, Academic Press, 2000.