ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 325-330

Parallel asynchronous distributed computations of optimal control in large state space Markov Decision processes

Bruno Scherrer

Cortex & Maia Teams
LORIA
Campus scientifique B.P. 239
F-54506 Vandœuvre-lès-Nancy Cedex
scherrer@loria.fr

Abstract. This paper emphasizes the link between parallel asynchronous distributed computations (PADC) and Markov Decision Processes (MDPs), which are a powerful generic model for computing optimal control. We review some results arguing that *reasonably small* state space MDPs can be solved with PADC. We then propose a solution for extending these results when the state space is *large*. This shows that difficult optimal control problems have natural neural network-like solutions and suggests a general methodology for constructing neural networks.

Introduction

When introducing their model of neuron, McCulloch & Pitts proved that a synchronous assembly of simple neuron-like units is capable of performing any computation that a digital computer can (though not necessarily as rapidly or conveniently) [3]. The aim of our research is to extend this fundamental result by:

- removing biologically unplausible constraints on this model of distributed computations by "neural networks" such as the need for synchronicity
- showing, on specific problems, that the "neural network" approach has computational advantages on the traditional digital approach, such as better temporal complexity.

In this paper, we consider the problem of finding an optimal control in a stochastic domain, a problem which is formulated by Markov Decision Processes (MDP). We show that it can be solved by parallel asynchronous distributed computations (PADC) and argue that this massively parallel approach is significantly faster than its sequential version. Section 1 reviews a result by [1] showing that the computation of a contraction mapping fixed point (CMFP) can efficiently be done with PADC. Section 2 formalizes the problem of finding an optimal control with MDPs and recalls that it involves the computation of a CMFP: when the problem state space is reasonably small, one can use PACD to find the optimal control. Section 3 extends the use of PACD for large state space MDPs: we propose and analyse an approximation scheme which only rely on computations of CMFPs.

1. A link between global and local computations

Consider a network of distributed and asynchronous processing units. In general, it is difficult to make the link between the local computations and the global activity of the network. Is is often more complicated when local computations

ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 325-330

are not globally synchronized. This section reviews a well-known result about the computation of a contraction mapping fixed point (CMFP), for which we can make the link between the local and the global levels. This result is fundamental as it stands for a general argument for the remaining of the paper: every computation that can be characterized in terms of a CMFP can be treated with parallel asynchronous distributed computations (PADC).

Consider a finite set X, a norm $\|.\|$ on $\mathbb{R}^X$, and a contraction mapping $M : \mathbb{R}^X \longmapsto \mathbb{R}^X$, i.e. verifying $\|M(f') - M(f)\| \leq \gamma.\|f' - f\|$ with $\gamma \in [0, 1)$. It is a well-known result that M has one and only one fixed point f^*, that is a function of $\mathbb{R}^X$ that verifies

$$f^* = M(f^*) \tag{1}$$

In traditional (sequential) computations, a technique for computing the fixed point f^* is to iterate the following process:

$$\forall x \in X, f^{t+1}(x) = M(f^t)(x) \tag{2}$$

Indeed we have $f^t \overset{t\to\infty}{\longrightarrow} f^*$ with an exponential rate of convergence:

$$\|f^{t+1} - f^*\| \leq \gamma.\|f^t - f^*\| \leq \gamma^{t+1}.\|f^0 - f^*\| \tag{3}$$

The temporal complexity of this iterative process can be seen as the product of two terms:

- t_1: the time required for doing the iteration given by equation 2
- $t_2(\varepsilon)$: the number of iterations needed to approximate f^* with a given precision $\varepsilon > 0$.

Is is shown in [1] that f^* can be computed with PACD. Indeed, the following process

$$f(x) \leftarrow M(f)(x) \tag{4}$$

can be distributed over all $x \in X$ and f will also converge to f^*. Furthermore, these computations need not be synchronized: updates of $f(x)$ for all $x \in X$ can be done in an arbitrary order, or even with different frequencies. If we neglict communication delays, parallelizing this way might roughly divide the time t_1 (thus the global temporal complexity for estimating the fixed point) by the size $|X|$ of X. For more discussion about the resulting parallel complexity see [1][1].

The computation of a CMFP can be done with PADC and can be significantly faster (roughly $|X|$ times faster) than the sequential iterative process described by equation 1. Therefore, any problem that can be formulated as the computation of a CMFP can be solved more rapidly with PADC than with the iterative process described by equation 2.

2. Markov Decision Processes

Markov Decision processes (MDPs) [5] provide the theoretical foundations of challenging problems to researchers in artificial intelligence and operation research. These problems include planning under uncertainty and reinforcement learning [7].

An *MDP* is a controlled stochastic process satisfying the Markov property with rewards (numerical values) assigned to state-control pairs[2]. Formally, an

[1] See paticularly part 6.3.5 which describes conditions under which asynchronism can make the global computation faster than synchronism.

[2] Though our definition of reward is a bit restrictive (rewards are sometimes assigned to state transitions), it is not a limitation: these two definitions are equivalent/

ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 325-330

MDP is a four-tuple $\langle S, A, T, R \rangle$ where S is the *state space*, A is the *action space*, T is the *transition function* and R is the *reward function*. T is the state-transition probability distribution conditioned by the control:

$$\forall (s, s') \in S^2, \forall a \in A, T(s, a, s') \stackrel{def}{=} \Pr(s_{t+1} = s' | s_t = s, a_t = a) \tag{5}$$

$R(s, a) \in \mathbb{R}$ is the instantaneous reward for taking action $a \in A$ in state S.

The usual *MDP problem* consists in finding a *policy*, that is a mapping $\pi : S \to A$ from states to actions, that maximises the following performance criterion, also called value function of policy π:

$$V^{\pi}(s) = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t . R(s_t, \pi(s_t)) | s_0 = s\right] \tag{6}$$

It is shown [5] that there exists a unique optimal value function V^* which is the fixed point of the following contraction mapping, also called Bellman operator:

$$[B^* . f](s) = \max_a \left(R(s, a) + \gamma . \sum_{s'} T(s, a, s') . f(s') \right) \tag{7}$$

Once an optimal value function V^* is computed, an optimal policy can immediately be derived as follows:

$$\pi^*(s) = \arg\max_a \left(R(s, a) + \gamma . \sum_{s'} T(s, a, s') . V^*(s') \right) \tag{8}$$

The temporal complexity for solving the MDP problem is the complexity for computing the optimal value function. The sequential iterative procedure similar to equation 2 is known as *Value Iteration*. Section 1 argued that using PADC would significantly accelerate the computation of an MDP solution. Further details (such as the temporal complexity on various parallel machine models) can be found in [1].

3. Addressing large state space MDPs

Even if PADC can accelerate the computation of the optimal value function V^* in an MDP, it might not be sufficient when the state space S is large (or even infinite). For instance, the theoretical number of required iterations $t_2(\varepsilon)$ in order to approximate V^* with precision ε can dramatically grow with the number of states [2]. PADC only diminish the part t_1 of the temporal complexity and this might not be sufficient for large state spaces.

In this section, we propose and analyse an approximation scheme, state aggregation, which allows to use PADC for large state space MDPs. We begin by reviewing some recently published foundations for approximation in MDPs. We then apply them to the state aggregation case and show that it only involves CMFP on reasonably small spaces.

3.1. Safely approximating an MDP

This subsection reviews some theoretical results by [4] for analysing and iteratively improving an MDP approximation.

Consider an MDP $\mathcal{M} = \langle S, A, T, R \rangle$ and an approximation of it $\widehat{\mathcal{M}} = \langle S, A, \widehat{T}, \widehat{R} \rangle$. Let B^* be the exact Bellman operator of $\mathcal{M}$ (see eq. 7) and let $\widehat{B}^*$ be the approximate Bellman operator:

$$\left[\widehat{B}^* . f\right](s) = \max_a \left(\widehat{R}(s, a) + \gamma . \sum_{s'} \widehat{T}(s, a, s') . f(s') \right) \tag{9}$$

ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 325-330

Let V^* and $\widehat{V}^*$ be their fixed points. In practice, we would like to compute the exact value function V^* but we can only compute its approximation $\widehat{V}^*$. It is then interesting to evaluate the *approximation error*:

$$E_{app}(s) \stackrel{def}{=} |V^*(s) - \widehat{V}^*(s)| \tag{10}$$

The authors of [4] show that the *approximation error* depends on a quantity they call *interpolation error*:

$$E_{int}(s) \stackrel{def}{=} |\widehat{B}^*.V^*(s) - B^*.V^*(s)| = |\widehat{B}^*.V^*(s) - V^*(s)| \tag{11}$$

The *interpolation error* is the error due to one approximate mapping $\widehat{B}^*$ of the real value function V^* ; it measures how the approximate parameters ($\widehat{R}(s,a)$, $\widehat{T}(s,a,.)$) locally differ from the real parameters ($R(s,a), T(s,a,.)$). Indeed, we can show (see appendix) that for some constant K

$$E_{int}(s) \leq \max_a \left| R(s,a) - \widehat{R}(s,a) \right| + K. \max_a \left(\sum_{s' \in S} \left| T(s,a,s') - \widehat{T}(s,a,s') \right| \right) \tag{12}$$

If $\overline{E_{int}}(s)$ is an upper bound of $E_{int}(s)$ (e.g. take the bound given by equation 12), then, an upper bound $\overline{E_{app}}(s)$ of $E_{app}(s)$ is the fixed point of the following contraction mapping:

$$[E.f](s) = \overline{E_{int}}(s) + \max_a \left(\gamma. \sum_{s'} \widehat{T}(s,a,s').f(s') \right) \tag{13}$$

The authors of [4] also explain how to improve an approximation, i.e. how to distribute resources for describing the parameters R and T over the whole state space. They introduce the notion of *influence* $I_{S_0}(s)$ of the local interpolation error of state s on the *approximation error* over a subset $S_0 \subset S$:

$$I_{S_0}(s) \stackrel{def}{=} \frac{\partial \sum_{s' \in S_0} \overline{E_{app}}(s')}{\partial \overline{E_{int}}(s)} \tag{14}$$

Say we add or remove some resources near some state s. This might change the *interpolation error* by $\Delta\overline{E_{int}}(s)$. A gradient argument shows the effect this will have on the *approximation error*:

$$\Delta \left(\sum_{s' \in S_0} \overline{E_{app}}(s) \right) \simeq I_{S_0}(s).\Delta\overline{E_{int}}(s) \tag{15}$$

This suggests to remove resources where the above quantity is very little and to add resources where it is big. It is proven that the *influence* I_{S_0} is the fixed point of the following contraction mapping:

$$[D.f](s) = \left\{ \begin{array}{l} 1 \text{ iff } s \in S_0 \\ 0 \text{ iff } s \notin S_0 \end{array} \right. + \gamma. \sum_{s'} \widehat{T}(s', \pi_{err}(s'), s).f(s') \tag{16}$$

where $\pi_{err}(s) = \arg\max_a \sum_{s'} \widehat{T}(s,a,s').E_{app}(s')$ (see [4] for more details).

The *approximation error* E_{app} and the influence I_{S_0} are characterized as CMFPs on state space S. This means that they are as difficult to compute as the optimal value function in the real MDP $\mathcal{M}$. Next subsection describes a practical solution to this issue.

3.2. The state aggregation approximation

In this subsection, we introduce and discuss the MDP approximation with state aggregation. We apply the analysis of the previous subsection to this approximation scheme and show that the computations of the *approximation error* E_{app} and the influence I_{S_0} become tractable.

ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 325-330

Given an MDP $\mathcal{M} = \langle S, A, T, R \rangle$, the state aggregation approximation consists in introducing the MDP $\widehat{\mathcal{M}} = \langle \widehat{S}, A, \widehat{T}, \widehat{R} \rangle$ where the state space $\widehat{S}$ is a partition of the real state space S. Every element of $\widehat{S}$, which we call macro-state, is a subset of S and every element of S belongs to one and only one macro-state. In addition, every object defined on $\widehat{S}$ can be seen as an object of S which is constant on every macro-state. The number of elements of $\widehat{S}$ can be chosen little enough so that it is feasible to compute CMFPs on $\widehat{S}$. When doing a state aggregation approximation, natural choices[3] for $\widehat{R}$ and $\widehat{T}$ are the averages of the real parameters on each macro-state:

$$\left\{ \begin{array}{rcl} \widehat{R}(\widehat{s}, a) & = & \frac{1}{\left|\widehat{s}\right|} . \sum_{s \in \widehat{s}} R(s, a) \\ \widehat{T}(\widehat{s_1}, a, \widehat{s_2}) & = & \frac{1}{\left|\widehat{s_1}\right|} . \sum_{(s,s') \in \widehat{s_1} \times \widehat{s_2}} T(s, a, s') \end{array} \right. \tag{17}$$

We can deduce from equations 12 and 17 an upper bound of the *interpolation error* on the macro-state $\widehat{s_1} \in \widehat{S}$:

$$\forall s_1 \in \widehat{s_1}, E_{int}(s_1) \leq \overline{E_{int}}(\widehat{s_1}) = \overline{\Delta R}(\widehat{s_1}) + K . \sum_{\widehat{s_2} \in \widehat{S}} \overline{\Delta T}(\widehat{s_1}, \widehat{s_2}) \tag{18}$$

$$\text{with } \left\{ \begin{array}{l} \overline{\Delta R}(\widehat{s}) = \frac{1}{\left|\widehat{s}\right|} . \max_{(s,s') \in \widehat{s}} |R(s, a) - R(s', a)| \\ \overline{\Delta T}(\widehat{s_1}, \widehat{s_2}) = \frac{1}{\left|\widehat{s_1}\right|} . \max_{(s_1, s_1') \in \widehat{s_1}} |\sum_{s_2 \in \widehat{s_2}} T(s_1, a, s_2) - T(s_1', a, s_2)| \end{array} \right. \tag{19}$$

The approximation by state aggregation is particularly interesting because the *approximation error* can be computed with a complexity that is similar to the one required for computing the optimal control in the approximate model $\widehat{\mathcal{M}}$. Indeed, as the upper bound of the *interpolation error* is constant over each macro-state, equation 13 becomes a contraction mapping on $\widehat{S}$ (and not anymore on S):

$$\left[\widehat{E'}.f\right](\widehat{s_1}) = \overline{E_{int}}(\widehat{s_1}) + \max_a \left(\gamma . \sum_{\widehat{s_2}} \widehat{T}(\widehat{s_1}, a, \widehat{s_2}) . f(\widehat{s_1}) \right) \tag{20}$$

Using the same arguments, the influence of the interpolation error on the approximation for a subset S_0 is the fixed point of the following contraction mapping defined on $\hat{S}$:

$$\left[D'.f\right](\widehat{s}) = \left\{ \begin{array}{l} 1 \text{ iff } \widehat{s} \subset S_0 \\ 0 \text{ iff } \widehat{s} \not\subset S_0 \end{array} \right. + \gamma . \sum_{\widehat{s'}} \widehat{T}(\widehat{s'}, \pi_{err}(\widehat{s'}), \widehat{s}) . f(\widehat{s'}) \tag{21}$$

When an MDP has a large state space, the approximation by state aggregation is particularly interesting because it involves computations of CMFPs on the approximate aggregate state space $\widehat{S}$. Therefore, all these computations can be done with PADC, and the acceleration that such an approach provides is as significant as explained in section 1.

4. Conclusion

In this article, we recall a fundamental theoretical result: any mathematical function that can be characterized as a contraction mapping fixed point can be

[3] We want the interpolation error to be as little as possible and we need to verify constraints such as $\sum_{\widehat{s_2}} T(\widehat{s_1}, a, \widehat{s_2}) = 1$

ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 325-330

efficiently estimated with parallel asynchronous distributed computations. By making the link between global and local computations, such a result constitutes an interesting basis for constructing fully-understandable neural networks.

We exploit this result several times throughout the paper in order to propose efficient neural-like algorithms for finding optimal control in small and large state space Markov Decision processes. Due to lack of space, we couldn't describe any experimental evaluation. We practically used this framework to solve challenging control problems such as planning a route for a car-like vehicule (controlled by acceleration and steering-wheel) in a continuous environment (details about this problem and many other control problems can be found in [6]).

References

[1] D.P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Prentice hall, 1989.

[2] M. L. Littman, T. L. Dean, and L. P. Kaelbling. On the complexity of solving Markov decision problems. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI–95)*, pages 394–402, Montreal, Québec, Canada, 1995.

[3] W.S McCulloch and W.P Pitts. A logical calculus in the ideas immanent in nerveous activity. *Bulletin of Mathematical Biophysics*, 1943.

[4] R. Munos and A. Moore. Rates of convergence for variable resolution schemes in optimal control. In *International Conference on Machine Learning*, 2000.

[5] M. Puterman. Markov Decision Processes, 1994.

[6] B. Scherrer. *Apprentissage de reprsentation et auto-organisation modulaire pour un agent autonome.* PhD thesis, Université Henri Poincaré - Nancy 1, To appear (January 2003).

[7] R.S. Sutton and A.G. Barto. *Reinforcement Learning, An introduction.* Brad-Ford Book. The MIT Press, 1998.

Appendix: derivation of an upper bound of the *interpolation error*

$$
\begin{aligned}
E_{int}(s_1) &= \left| \max_a \left(R(s_1,a) + \gamma . \sum_{s_2 \in S} T(s_1,a,s_2).V^*(s_2) \right) \right. \\
&\quad \left. - \max_a \left(\widehat{R}(s_1,a) + \gamma . \sum_{s_2 \in S} \widehat{T}(s_1,a,s_2).V^*(s_2) \right) \right| \\
&\leq \max_a \left| R(s_1,a) - \widehat{R}(s_1,a) + \gamma . \sum_{s_2 \in S} \left(T(s_1,a,s_2) - \widehat{T}(s_1,a,s_2) \right) .V^*(s_2) \right| \\
&\leq \max_a \left| R(s_1,a) - \widehat{R}(s_1,a) \right| + \gamma . \max_a \left(\sum_{s_2 \in S} \left| T(s_1,a,s_2) - \widehat{T}(s_1,a,s_2) \right| . |V^*(s_2)| \right) \\
&\leq \max_a \left| R(s_1,a) - \widehat{R}(s_1,a) \right| + \frac{\max_{s,a} |R(s,a)|}{1-\gamma} . \max_a \left(\sum_{s_2 \in S} \left| T(s_1,a,s_2) - \widehat{T}(s_1,a,s_2) \right| \right)
\end{aligned}
$$