An Optimality Principle for Concurrent Systems

Langford B White and Sarah L Hickmott

School of Electrical and Electronic Engineering
The University of Adelaide
Australia
{lwhite, shick}@eleceng.adelaide.edu.au

Abstract. This paper presents a formulation of an optimality principle for a new class of concurrent decision systems formed by products of deterministic Markov decision processes (MDPs). For a single MDP, the optimality principle reduces to the usual Bellman's equation. The formulation is significant because it provides a basis for the development of optimisation algorithms for decentralised decision systems including a recently proposed method based on Petri Net unfoldings.

1 Introduction

Markov decision processes have been widely studied and applied to a large number of optimisation problems, particularly in planning and scheduling applications (see eg [1], [2]). Many solution techniques have been developed but all basically rely on Bellman's *principle of optimality*. As we shall see, this principle is really quite straightforward and intuitive in nature.

We have a considerable interest in optimisation of concurrent systems. These systems don't have a global clock, so a class of models such as MDPs which are based on a global clock, are not appropriate for concurrent systems. In particular, we are interested in optimisation of automated planning systems which are inherently concurrent in nature. Much of the work which has been done in concurrent planning effectively translates the problem into a (huge) MDP by placing an artificial total ordering on the system. This can lead to excessively large problems which don't scale well.

Recently, a solution to optimisation of concurrent systems based on Petri Nets (PNs) has been proposed [3],[4]. The algorithm, known as ERV-Fly finds globally optimal solutions via a technique known as directed unfolding, described in [5]. A close analysis of the algorithm shown that it embodies (among other things) a principle of optimality for concurrent systems that is not unlike Bellman's principle for MDPs in nature. The purpose of this paper is to derive this principle and show that it is a natural extension of the totally ordered (MDP) case. The result, although appearing simple in hindsight, is significant because it provides a necessary condition that any optimisation algorithm must meet. Also, the principle may lead to other optimisation algorithms for concurrent systems,

much in the way that Bellman's principle has for MDPs. We believe that this is the first concise statement of such a principle for a class of concurrent systems.

The layout of the paper is as follows. We firstly review Bellman's principle for MDPs in a somewhat non-standard fashion, that motivates the generalisation to concurrent systems in a clear way. We then develop a PN model for MDPs. In section 3, we describe how to generalise these PN models to a class of concurrent systems described as *products of MDPs* (PMDPs). Products of transition systems (the Markov part of MDP) have been studied in detail in [6], although not from an decision theoretic point of view. Thus, PMDPs, in some sense, are a new class of models for concurrent decision systems. In section 4, we introduce PN unfoldings as a framework for deriving and proving an optimality principle for, firstly a single MDP, and secondly for a PMDP.

2 Petri Net Models for Markov Decision Processes

A (deterministic) Markov Decision Process (MDP) is a tuple $\mathcal{M} = (S, A, is, g, \pi, \sigma, \phi)$ where

- S is a finite set of *states*,
- A is a finite set of *actions*,
- $is \in S$ is the *initial state*,
- $g \in S$ is the *goal state*,
- $\pi : A \to S$ giving the unique predecessor state to an action,
- $\sigma : A \to S$ giving the unique successor state to an action, and
- $\phi : A \to \mathbb{R}^+$ is the *cost function*.

The semantics of the MDP are as follows. Suppose the system is in some state s, then each action in $\pi^{-1}(s)$ is enabled. Suppose we execute action $a \in \pi^{-1}(s)$, then the system moves to state $s' = \sigma(a)$. We say that $s = \pi(a)$ is the predecessor state for a, and $s' = \sigma(a)$ is the successor state for a. [1] Every time an action a is executed, an additive cost function is incremented by an amount $\phi(a) > 0$. Our objective is to find a sequence of actions $\{a_k\}$ starting at state is and terminating at state g [2] which minimises the total cost. Such a problem is called a *shortest path* problem, and $\{a_k\}$ is called a shortest path. Standard tools (eg value iteration, policy iteration, linear programming) can be applied to this problem. Most solution methods are based around solving *Bellman's equation*, an expression of the *principle of optimality* [1]. A *policy* is a mapping $\psi : S \to A$ which specifies which action a is executed when the system is in state s. The solution of Bellman's equation yields a policy which minimises total cost to

[1] In our model, we only address *deterministic* outcomes. Thus the successor state is uniquely defined by the action being executed. In the general case, there is a probability distribution specified on a set of successor states but we don't consider this case here.

[2] We can easily generalise to the case there the goal consists of a subset of states.

reach the goal. One interpretation of Bellman's principle is that on any globally optimal path (sequence of actions) from is to g, each local policy must also be optimal in the sense of choosing an action which minimises the total cost of it and all past actions. More precisely, if $J^*(s)$ denotes the optimal cost to arrive at a state s from is, then

$$J^*(s) = \min_{a \in \sigma^{-1}(s)} \left(J^*(\pi(a)) + \phi(a) \right) \ , \tag{1}$$

and a minimising action a in (1) specifies the optimal (local) policy to be executed in state $\pi(a)$ to arrive in state s. [3]

Let A^* denote the set of all sequences of actions (called *words*), then for any $v = \{a_i\} \in A^*$ we define the cost of v by extending $\phi : A^* \to \mathbb{R}^+ \cup \{\infty\}$ as

$$\phi(v) = \sum_i \phi(a_i) \ .$$

Observe that ϕ is additive on concatenation of finite words, ie if $v, w \in A^*$ are finite then we can define a concatenation $v \circ w$ (all the actions in v followed by the actions in w), and $\phi(v \circ w) = \phi(v) + \phi(w)$. A word u is called a *prefix* of a word w if there is a word v such that $w = u \circ v$. Such a prefix is *proper* if v contains at least one action. A word $a_1, a_2, \ldots, a_k$ is called a *computation* if $\sigma(a_i) = \pi(a_{i+1})$ for all $i = 1, \ldots, k-1$. Such a computation is called a *history* if $\pi(a_1) = is$. The *final state* of a finite history $a_1, a_2, \ldots, a_k$ is $\sigma(a_k)$. Such a history is called *feasible* if its final state is the goal g. A feasible history is *optimal* if its cost is the minimum over all feasible histories.

We can thus deduce from (1) that *every prefix of an optimal history is itself optimal* in the following sense. Let w^* denote an optimal history, and consider any finite proper prefix u^* of w^*. Then for any history u with the same final state as u^*, $\phi(u^*) \leq \phi(u)$.

Markov processes have the property that there is a total order on actions. Concurrent systems are systems where there is no global clock, and thus no total ordering on actions is generally possible. We can only assume a partial ordering. Because we are interested in considering the shortest path problem for concurrent systems, we wish to use a class of models which captures this behaviour. Petri Nets (PNs) have been shown to be a useful set of models for concurrent systems (see eg [7], [8]). Our first step in generalising MDPs to concurrent systems will be to derive a PN model for an MDP.

[3] The conventional formalism for Bellman's equation uses the notion of an optimal cost to go from a state, rather than the optimal cost to arrive in a state. The two expressions of optimality are equivalent.We adopt the second approach as it is more appropriate for developing the optimality principle we seek for concurrent systems.

A Petri Net is a tuple $\mathcal{P} = (P, T, F, M_0)$ where

- P is a set of *places*,
- T is a set of *transitions*,
- $F \subseteq \{P \times T\} \cup \{T \times P\}$ is a set of (directed) *arcs*, and
- $M_0 \subseteq P$ is the *initial marking*.

A place p is said to be *marked* is there is a token held in p. The system evolves by moving tokens between places. The initial marking M_0 is the set of places initially holding a token. The semantics of a PN are as follows. Consider a transition t. Let $\pi(t) = \{p \in P : \exists\ (p,t) \in F\}$ denote the set of *predecessors* to t. If every place in $\pi(t)$ is marked, we say that t is enabled. An enabled transition *may* fire, in which case, a token is removed from each place in $\pi(t)$, and a token is placed in each place in $\sigma(t) = \{p \in P : \exists\ (t,p) \in F\}$ (the set of *successors* to t). If a transition t fires when the system has marking M, the system marking becomes $M' = (M \setminus \pi(t)) \cup \sigma(t)$. We write $M \xrightarrow{t} M'$. [4] A marking M is *reachable* from M_0 if there is a sequence of firings $t_1, t_2, \ldots, t_k$ such that $M_0 \xrightarrow{t_1} \cdots \xrightarrow{t_k} M$. The sequence $t_1, \ldots, t_k$ is known as an *occurrence sequence*.

Now, let's establish a PN representation for an MDP. We have the following associations specified by an isomorphism ℓ :

- Each state $s \in S$ corresponds to a place $\ell(s) \in P$,
- Each action in $a \in A$ corresponds to a transition $\ell(a) \in T$,
- For each action $a \in A$, there are arcs $(\ell(\pi(a)), \ell(a))$ and $(\ell(a), \ell(\sigma(a)))$ in F,
- $M_0 = \ell(is)$.

We also have a unique place $\ell(g)$ specified by the goal state g. The mappings π, σ and ϕ can be used consistently in the PN, observing that for a PN representation of an MDP, $\pi(t)$ and $\sigma(t)$ each have exactly one element for any transition $t \in T$. It can be easily verified that the goal $\ell(g)$ is reachable from M_0 if and only if the MDP has a shortest path.

The method we propose for determining the optimal policy using the PN representation of the MDP, is based on PN unfoldings [6]. An unfolding is another PN with specific properties that make it possible to determine all possible sequences of transition firings which take the initial marking to the goal marking, and the associated costs. We shall address unfoldings in section 4.

3 PN Representations for Products of MDPs

In this section, we define a concurrent system model based on the idea of products on MDPs. This approach is motivated by the ideas of [6]. Let $\mathcal{M}_i =$

[4] This is considered in the *multiset* framework where a particular place p appears once for each token present in p. We will focus on so-called *one-safe* PNs where the maximum number of tokens in a place is one, so standard set theory suffices.

$(S_i, A_i, is_i, g_i, \pi_i, \sigma_i, \phi_i)$, $i = 1, \ldots, n$ be MDPs, then we define the product $\mathbf{M} = \{\mathcal{M}_1, \ldots, \mathcal{M}_n, \mathbf{A}\}$, where $\mathbf{A}$ is a set of *global actions*. The set $\mathbf{A}$ is a subset of

$$(A_1 \cup \epsilon) \times \cdots \times (A_n \cup \epsilon) \setminus \{\epsilon, \ldots, \epsilon\} \ ,$$

where ϵ is a special action which means "do nothing". We call $\mathcal{M}_i$, the i-th *component* of the product $\mathbf{M}$. Global actions have the effect of synchronising MDPs through "shared" actions, ie those global actions with more that one non-ϵ component. We say that a component $\mathcal{M}_i$ *participates* in a global action $\mathbf{a} = (a_1, \ldots, a_n)$ if $a_i \neq \epsilon$. We remark that products are inherently partially ordered systems. For the product $\mathbf{M}$ we define the global state space $\mathbf{S} = S_1 \times \cdots \times S_n$, and the global initial state $is = (is_1, \ldots, is_n)$ and the global goal is $g = (g_1, \ldots, g_n)$.

Let $\mathbf{a} = (a_1, \ldots, a_n)$, then the cost of $\mathbf{a}$ is given by

$$\phi(\mathbf{a}) = \sum_{i=1}^{n} \phi_i\left(a_i\right) \ ,$$

where $\phi_i(\epsilon) = 0$ for all $i = 1, \ldots, n$. Observe that $\phi(\mathbf{a}) > 0$ for all $\mathbf{a} \in \mathbf{A}$. Let $\mathbf{A}^*$ denote the set of all sequences of global transitions, then we can extend ϕ to $\mathbf{A}^*$ in the same way as described in section 2. The shortest path problem for $\mathbf{M}$ involves finding a sequence of global actions taking the global state from is to g which results in the minimum cost.

PMDPs could be solved in the conventional way by using the so-called *interleaved representation* of the product [6]. This involves applying dynamic programming to the large MDP constructed by enumerating the global state space $\mathbf{S}$. However, such a method is intrinsically inefficient (exponential in the number of components in the product) because it places an unnatural total ordering on events which includes all interleavings of concurrent events. We shall return to this issue in section 4.

We can map products of MDPs to a PN in a way that preserves any concurrency in the system as follows. Let ℓ_i denote the isomorphism which takes component MDP $\mathcal{M}_i$ to its PN representation $\mathcal{P}_i$. Then define a PN $\mathcal{P} = (P, T, F, M_0)$ by

- $P = \cup_{i=1}^{n} \{\ell_i(s_i) : s_i \in S_i\}$,
- for each $\mathbf{a} \in \mathbf{A}$ there is a unique $t \in T$ which we label $\ell(\mathbf{a})$,
- Let $\mathbf{a} \in \mathbf{A}$, then $(\ell_i(s_i), \ell(\mathbf{a})) \in F \Leftrightarrow \exists\ i \ni s_i = \pi_i(a_i) \wedge a_i \neq \epsilon$, and $(\ell(\mathbf{a}), \ell_i(s_i)) \in F \Leftrightarrow \exists\ i \ni s_i = \sigma_i(a_i) \wedge a_i \neq \epsilon$,
- $M_0 = \cup_{i=1}^{n} \ell_i(is_i)$.

Here ℓ is defined on global states by $\ell(\mathbf{s}) = (\ell_1(s_1), \ell_2(s_2), \ldots, \ell_n(s_n))$. We also identify a global goal $\ell(g) = (\ell_1(g_1), \ell_2(g_2), \ldots, \ell_n(g_n))$. We can show that $\ell(g)$ is reachable from M_0 if and only if there is a (global) shortest path.

4 PN Unfoldings

In this section, we will introduce the concept of a branching process of the PN representation of a product of MDPs and thence the unfolding. We'll then show how unfoldings can be used to determine shortest paths. For notational clarity, in the sequel, we shall drop the explicit use of the isomorphism ℓ when the context is clear.

Let $\mathcal{P}$ be a PN representation of a product. A branching process is a special type of PN obtained from $\mathcal{P}$ which has a tree-like structure. In branching processes, places are called *conditions* and transitions are called *events*. The set of branching processes of $\mathcal{P}$ is constructed as follows. There is a homomorphism ψ defined on the set of PNs such that :

(a) The PN consisting of the initially marked places $\overline{M_0}$ with $\psi(\overline{M_0}) = M_0$, and no events is a branching process of $\mathcal{P}$,
(b) Let $\mathcal{N}$ be a branching process of $\mathcal{P}$ such that there is some reachable marking M in $\mathcal{N}$ with $\psi(M)$ enabling some transition t in $\mathcal{P}$. Let e be an event labelled by $t = \psi(e)$ and $M_1 \subseteq M$ be such that $\psi(M_1) = \pi(t)$. The set of all (e, M_1) is called the set of *possible extensions* to $\mathcal{N}$. Then the PN formed by adding the event e and all conditions c with $\psi(c) \in \sigma(t)$, along with the arcs (e, c) and $\{(m, e) : m \in M_1\}$ to $\mathcal{N}$ is also a branching process of $\mathcal{P}$.
(c) The union of any set of branching processes is also a branching process.

The union of all branching processes is known as the *unfolding* and any of its branching processes is a *prefix* of the unfolding. In general, we seek a *finite* prefix of the unfolding which has all information needed to solve the shortest path problem.

There are two specific technical issues associated with determining an unfolding. Firstly, we need to determine which events should be added to the unfolding as it is progressively computed according to the above process. Secondly, we need to determine when we can stop extending the unfolding after the addition of some new event. We do this by constructing a certain type of ordering on events which is based on the cost function ϕ associated with each event.

Firstly, we need to introduce some additional technicalities. We firstly consider the unfolding $\mathcal{U}$ of the PN representation of a single MDP. We shall define the cost associated with an event e by the cost of its label, ie $\phi(e) = \phi(\psi(e))$ (with obvious abuse of notation). If there is a directed path from an event e to another event e' in $\mathcal{U}$, we say $e < e'$. This is an irreflexive partial order on the events on $\mathcal{U}$ called the *dependency ordering*. Let e be an event in $\mathcal{U}$, and let $e_1, \ldots, e_m$ denote an occurrence sequence with $e_m = e$. The *history* of an event e is the history $\psi(e_1), \ldots, \psi(e_m)$ and is denoted by $H(e)$. The final state of $H(e)$ written $St(e)$ is the state reached after execution of e ie $\sigma(e)$. A history $H(e)$ is called feasible if $St(e) = g$, and then e is called a terminal event. A feasible history is optimal if

it has the minimum cost over all such feasible histories. A fundamental property for the single component case [6] is that $H(e) = H(e') \Leftrightarrow e = e'$. This is because every event has a unique predecessor condition.

Let $u, v \in A^*$ be words, then we define a partial order $\prec_\phi$ by $u \prec_\phi v \Leftrightarrow \phi(u) < \phi(v)$. Evidently, $\prec_\phi$ refines the prefix order ie if u is a proper prefix of v, then $u \prec_\phi v$ since costs are positive. We can define the reflexive order $\preceq_\phi$ by saying $u \preceq_\phi v$ if either $u \prec_\phi v$ or $u = v$. Since two words $u \neq v$ can have the same cost, $\preceq_\phi$ is not total. We can thus define an partial ordering on events according to the cost of their histories (as words). It is important for this partial order to refine the prefix order as we want to be able to formulate a principle of optimality that is essentially "causal", ie based on the idea of testing possible extensions for optimality. This is in the spirit of (1).

The unfolding algorithm ERV-Fly [3],[4] proceeds by retaining a queue of all possible extensions to the current branching process. Possible extensions in the queue are ordered according to the cost of their history. Those possible extensions leading to the same state but at a higher cost are discarded. The algorithm terminates when we add an event which corresponds to a transition having the goal state as its successor. Provided the queue ordering ensures that all events on an optimal history are removed from the queue before a terminating event of a non-optimal history, then the unfolding algorithm is guaranteed to find an optimal history. This is proven in more generality in [3],[4] (theorem 4.2.1). This approach embodies a concise characterisation of optimality in the single component case, as theorem 1 states.

Theorem 1. *Let $H(e^*)$ be an optimal history, and let $H(e_0^*)$ be any proper prefix of $H(e^*)$. Then for any history h with the same final state as $H(e_0^*)$, $\phi(H(e_0^*)) \leq \phi(h)$.*

Proof. If there was a history h with $\phi(h) < \phi(H(e_0^*))$, we could form the feasible history $h \circ v^*$, where $H(e^*) = H(e_0^*) \circ v^*$, and $\phi(h \circ v^*) < \phi(H(e^*))$ contradicting optimality of $H(e^*)$.

Comments

1. Theorem 1 is an identical statement to (1) but is in a form that we can naturally generalise to the multicomponent case.
2. This theorem motivates part of the ERV-Fly algorithm [3],[4]. If we have two possible extensions with different events e, e', with the same final state $\sigma(e) = \sigma(e')$, then we add that event with lowest cost history, and we need no longer extend the unfolding from the other event.

4.1 Multiple Components

Let's now address the case of a product of MDPs. In this case, there is no obvious direct analogy to Bellmans's principle, and we seek to develop such an analogy.

The main difficulty is that events are not uniquely defined by their histories, so we can't define an ordering on the events in the set of possible extensions based on their histories. However, the notion of the independence of (global) actions is useful for overcoming this difficulty. Two global actions in the PN representation of a product $\mathbf{M}$ are independent if no component of $\mathbf{M}$ participates in both of them. Two words $\mathbf{w}, \mathbf{w}'$ are said to be *1-equivalent* if (i) $\mathbf{w} = \mathbf{w}'$, or (ii) there are independent actions $\mathbf{a}, \mathbf{a}'$ and two words $\mathbf{u}, \mathbf{v}$ such that $\mathbf{w} = \mathbf{uaa}'\mathbf{v}$ and $\mathbf{w}' = \mathbf{ua}'\mathbf{av}$. We define an equivalence relation $\equiv$ on $\mathbf{A}^*$ as the transitive closure of the 1-equivalence relation. It can be easily shown that if $\mathbf{h}$ is a history of $\mathbf{A}$, then so is every word $\mathbf{w} \equiv \mathbf{h}$.

Importantly, from the viewpoint of optimisation, equivalent words have identical cost as theorem 2 shows.

Theorem 2. *Let* $\mathbf{u}, \mathbf{v} \in \mathbf{A}^*$*, then* $\mathbf{u} \equiv \mathbf{v} \Rightarrow \phi(\mathbf{u}) = \phi(\mathbf{v}))$.

Proof. If $\mathbf{u} \equiv \mathbf{v}$ there is a sequence of permutations P of the actions in $\mathbf{u}$, so that $P(\mathbf{u}) = \mathbf{v}$. Since permutations in the order of actions does not effect the cost of a word, $\phi(P(\mathbf{u})) = \phi(\mathbf{u}) = \phi(\mathbf{v})$. □

Definition 1. *A Mazurkiewicz trace (a trace) for a product* $\mathbf{M}$ *is an equivalence class of words under* $\equiv$*. We write* $[\mathbf{w}]$ *for a trace, and* $[\mathbf{A}^*]$ *for the set of all traces.*

From theorem 2, we can unambiguously define the cost of a trace by extending $\phi : [\mathbf{A}^*] \to \mathbb{R}^+ \cup \{\infty\}$. This allows us to extend the partial order $\prec_\phi$ to traces.

The concatenation of two traces $[\mathbf{u}]$ and $[\mathbf{v}]$, written $[\mathbf{u}] \circ [\mathbf{v}]$ is defined to be the trace $[\mathbf{u} \circ \mathbf{v}]$. We say $[\mathbf{u}]$ is a prefix of $[\mathbf{w}]$ if there is a trace $[\mathbf{v}]$ such that $[\mathbf{w}] = [\mathbf{u}] \circ [\mathbf{v}]$. It is easy to show that the prefix relation is a partial order.

Lemma 1. *The partial order* $\prec_\phi$ *defined on* $[\mathbf{A}^*] \times [\mathbf{A}^*]$ *refines the prefix ordering on traces.*

Proof. Suppose $[\mathbf{w}] = [\mathbf{u}] \circ [\mathbf{v}] = [\mathbf{u} \circ \mathbf{v}]$, then $\phi([\mathbf{w}]) = \phi(\mathbf{w}) = \phi(\mathbf{u}) + \phi(\mathbf{v}) > \phi(\mathbf{u}) \Rightarrow \phi(\mathbf{u}) < \phi(\mathbf{w}) \Leftrightarrow \mathbf{u} \prec_\phi \mathbf{w} \Leftrightarrow [\mathbf{u}] \prec_\phi [\mathbf{w}]$. □

We now introduce the notion of a *configuration* of a branching process. Firstly, any two nodes x, y (condition or event) in a branching process are *causally related* if either $x \leq y$ or $y \leq x$ where $\leq$ is the reflexive dependency ordering. Two nodes are *in conflict* if there is a condition b so that x and y can be reached from b by exiting b on different paths. Two nodes are *concurrent* if they are neither causally related nor in conflict.

A set of events C of a branching process is a configuration if it is both causally closed (ie $e \in C \wedge e' < e \Rightarrow e' \in C$) and conflict-free.

A *realisation* of a set E of events is an occurrence sequence in which each event of E appears exactly once, and no other events appear. Every configuration has at least one realisation. It can be shown that every realisation of a finite configuration C leads to the same marking (see eg [6], proposition 3.18). We call this marking the *global state* $\mathbf{St}(C)$. Given a configuration C, suppose there is configuration $C_0 \subset C$, and a set of events $E \neq \emptyset$ disjoint from C_0 with $C = C_0 \cup E$. We say that C_0 is a *proper prefix* of C and that E is an *extension* of C_0, and write $C = C_0 \circ E$.

We can now define the history of a configuration C as a word $\mathbf{a_1} \ldots \mathbf{a_n}$ such that $e_1 \ldots e_n$, with $\psi(e_i) = \mathbf{a}_i, i = 1, \ldots, n$, is a realisation of C. We denote the set of histories of C by $\mathbf{H}(C)$. A history $\mathbf{H}(C)$ is called *feasible* if $\mathbf{St}(C) = \mathbf{g}$. The cost of a history is its cost defined as a trace. A feasible history is *optimal* if its cost is minimum over all feasible histories. The cost of a configuration is the cost of its history.

The following results are proven in [6] (proposition 4.28). (a) Let C_1 and C_2 be configurations then $C_1 = C_2 \Leftrightarrow \mathbf{H}(C_1) = \mathbf{H}(C_2)$. (b) Let C be a configuration, then $\mathbf{H}(C)$ is a trace. These results are important because they allow us to characterise configurations by their histories and that every history in $\mathbf{H}(C)$ has the same cost, and leads to the same reachable marking. This again leads to an optimality principle.

Theorem 3. *Let C^* denote an optimal configuration and suppose that C_0^* is a proper prefix of C^*. Then for all configurations C_0 with $St(C_0) = St(C_0^*)$, $\phi(C_0^*) \leq \phi(C_0)$.*

Proof. Write $C^* = C_0^* \circ E$ where E is an extension of C_0^*. Then $\mathbf{H}(C^*) = \mathbf{H}(C_0^*) \circ [e]$ for some $e \in \mathbf{A}^*$. Let $\mathbf{H}(C^*) = [c^*]$ and $\mathbf{H}(C_0^*) = [c_0^*]$, since histories of configurations are traces, then $\phi([c^*]) = \phi([c_0^*]) + \phi([e])$. Suppose there was a configuration C_0 with the same final state as C_0^* and with $\phi(C_0) < \phi(C_0^*)$, then with $\mathbf{H}(C_0) = [c_0]$, we have $\phi([c_0] \circ [e]) = \phi([c_0]) + \phi([e]) < \phi([c_0^*]) + \phi([e]) = \phi([c^*])$ which contradicts the optimality of C^*. □

Comments

1. It may seem that the expression of optimality in terms of configurations might be restrictive in the sense that the result doesn't apply in generality to all PMDPs. However the specific property that configurations represent the so-called "true" concurrency semantics of the PMDPs means that theorem 3 is indeed general.
2. Theorem 3 is a necessary condition to be satisfied for any algorithm which yields an optimal solution and is a part of the ERV-Fly algorithm [3],[4]. It is conjectured that the principle might also lead to other algorithms for determining optimal configurations in a similar way that Bellman's principle does. This is a subject of ongoing work. Of course, not all optimisation algorithms that exploit concurrency will necessarily be *derived* from theorem

3, since, in particular, it is based on Petri Nets and unfoldings, however, in some equivalent sense, any truly concurrent optimisation algorithm for PMDPs must satisfy something equivalent to theorem 3.

5 Conclusion

This paper has derived a principle for optimality for a new class of concurrent decision processes called products of Markov Decision Processes (PMDPs). This class of processes is motivated by the concurrent system models studied in [6]. The principle reduces to the standard Bellman's principle of optimality applying to the single MDP case. The principle is embodied in an existing algorithm for concurrent system optimisation called ERV-Fly [3],[4], although our results show it is a necessary part of any optimisation algorithm for PMDPs. Due to the fact that ERV-Fly can be applied to more general Petri Nets (see [3],[4]), we would also conjecture that a similar principle applies to general Petri Net models. This is, in part, supported by the properties of so-called adequate orderings on configurations and generalisations thereof. We also believe that the principle might lead to the development of new optimisation algorithms for concurrent systems in a similar way that Bellman's principle motivated a number of different optimisation algorithms for MDPs such as value iteration, policy iteration, and hybrids of these two methods.

In other ongoing work, we are considering the suitability of the above approach for systems with probabilistic outcomes. These systems are already included in MDP models but it can be difficult to extend this approach to concurrent systems (see eg [8], [9]). However, the restricted nature of PMDPs which still retain a notion of local state in each component may permit a solution.

Acknowledgements

The authors acknowledge the support of National ICT Australia and the Defence Science and Technology Organisation Australia through the Dynamic Planning, Optimisation and Learning Project. We also thank Sylvie Thiébaux, Patrik Haslum and Jussi Rintanen for useful discussions.

References

1. D. P. Bertsekas, *Dynamic Programming and Optimal Control*, Belmont, MA : Athena Scientific, second edition, 2001, volumes 1 and 2.
2. M. Ghallab, D. Nau and P. Traverso, *Automated Planning : Theory and Practice*, Morgan-Kaufmann, 2004.
3. Sarah L. Hickmott, *Directed Unfolding : Reachability Analysis of Concurrent Systems and Applications to Automated Planning*, Ph.D. Thesis, School of Electrical and Electronic Engineering, The University of Adelaide, 2008.

4. S. Hickmott, J. Rintanen, S. Thiébaux and L. B. White, "Planning via Petri Net Unfolding", *Proc. 20th Int. Joint Conf. on Artificial Intelligence*, India, 2007, pp. 1904-1911.
5. B. Bonet, P. Haslum, S. Hickmott and S. Thiébaux, "Directed Unfolding of Petri Nets", *Workshop on Unfolding and Partial Order Techniques*, in *28th Int. Conf. of Application and Theory of Petri Nets and other Models of Concurrency*, Poland, 2007.
6. J. Esparza and K. Heljanko, *Unfoldings : A Partial Order Approach to Model Checking*, Berlin : Springer, 2008.
7. T. Murata, "Petri Nets : Properties, Analysis and Applications", *Proceedings of the IEEE*, v. 77, no. 4, 1989, pp. 541-580.
8. A. Benveniste, E. Fabre and S. Haar, "Markov Nets : Probabilistic Models for Distributed and Concurrent Systems", *Rapport de Recherche INRIA RR-4253*, Sep. 2001.
9. S. Haar, "Probabilistic Cluster Unfoldings", *Fundamenta Informaticae*, v.53, vols 3,4, 2002, pp. 281-314.