Kalpa Publications in Computing

Volume 8, 2018, Pages 1–10

TNC'18. Trusted Numerical Computations

kalpa Computing

Policy Iterations Without Selection Property

Assalé Adjé

Laboratoire de Mathématiques et Physique (LAMPS)
Université de Perpignan Via Domitia, France
first.last@univ-perp.fr

Abstract

In this paper, we propose a modified policy iterations algorithm which does not rely on the selection property. The selection property is the key argument to make improvements during policy iterations. Indeed, a new policy is computed as an optimal solution of a minimization problem. However, in some cases, it might be difficult to prove that an optimal solution exists. To overcome this issue, the new policy is computed as a guaranteed sub-optimal solution of the minimization problem. The good choice of the perturbation parameters preserves the advantages of the original policy iterations algorithm such as the computation of a post-fixed point at each step and the convergence to a fixed point.

1 Introduction

Policy iterations algorithms were introduced for stochastic control problems [11] in 1960 and for stochastic games [10] in 1966. In those both contexts, the *selection property* is a natural assumption. The selection property principle can be formulated as follows: whatever the value to pay there exist a policy (action) for the (min) player which can reach this value. Even though the selection property is natural in games, the selection property has a key role in the improvement of a policy iterations algorithm. The selection property ensures the existence of a new policy which decreases the value at each step. Numerically, the price to pay is huge since the new policy is actually an optimal solution of a minimization problem.

The very first use of a policy iterations algorithm in static analysis was in 2005 in [6]. In this context, the number of policies is finite and the selection property holds naturally. The finiteness also guarantees the selection property in the extension of policy iterations to relational domains [9]. The extension of policy iterations algorithms to non-linear templates domains [4, 2, 3] complicates the validity of the selection property. Those extensions lead to compute a new policy as the optimal solution of a minimization problem with an infinite feasible set. Hence, the number of available policies is infinite. The selection property is not obvious without supplementary hypothesis on the constraints set (compactness) and/or on the objective function (coercivity). In [4], the selection property can be ensured from the Slater's constraint qualification. Unfortunately, in the general case, an optimal solution is not proved to exist. Moreover, to be correct policy iterations algorithms need, in practice, an exact optimal solution. This exactness cannot be done with numerical optimization solvers based on interior-points methods [13] which can only provide ϵ-optimal solutions. Nevertheless, as they are fast,

M. Martel, N. Damouche and J. Alexandre Dit Sandretto (eds.), TNC'18 (Kalpa Publications in Computing, vol. 8), pp. 1–10

such solvers are mainly used to solve convex optimization problems. Second, the programs implementing the methods deal with floating points and the returned optimal solution may not be optimal in the real numbers.

Actually, policy iteration algorithms use optimization solvers twice. First, as mentioned earlier, the computation of a new policy is performed by a optimization solver. In a second time, when the new policy is selected, the fixed point (or the smallest fixed point) of the map associated with this policy is computed as the optimal solution of a linear program. Again, we would like to have some guarantees concerning the optimal solution of a perturbed problem.

In this paper, we propose to overcome the use of optimal solutions and develop a policy iteration scheme capable to use ϵ-optimal solutions which always exist. To ensure the convergence of the algorithm we propose to make decrease the sequence of ϵ to zero geometrically. Furthermore, we introduce perturbations into the linear programming solving the smallest fixed point equations.

The paper is organized as follows. Section 2 is devoted to the presentation of the context and the hypothesis. Section 2 also briefly recalls the classical policy iteration algorithm. Section 3 describes the main results and the modified policy iteration algorithm. Finally, section 4 concludes and presents potential future approaches.

2 Problem statement

A policy iterations algorithm solves non-linear fixed point problem for maps $F : \mathbb{R}^d \mapsto \mathbb{R}^d$ of the form :

$$x \mapsto F(x) = \inf_{\pi \in \Pi} f^\pi(x) \tag{1}$$

The set $\mathbb{R}^d$ is equipped with the partial order $\leq$ meaning that $x \leq y$ if and only if $x_i \leq y_i$ for all $i \in \{1, \ldots, d\}$. Recall that this partial order makes $\mathbb{R}^d$ a lattice. The infimum is thus defined from this partial order i.e. coordinate-wise.

The set Π is called the set of *policies*, is possibly infinite and makes f^π a *policy map*. More formally, the set Π is a set of maps which act from $\{1, \ldots, d\}$ to the union over $i \in \{1, \ldots, d\}$ of sets Π_i. A map π associates to $i \in \{1, \ldots, d\}$ an element in Π_i. The maps f^π are actually maps of the form $(f_i^{\pi(i)})_{i \in \{1,\ldots,d\}}$. Taking the infimum over $\pi \in \Pi$ is equivalent to compute the vector of infima $(\inf_{a \in \Pi_i} f_i^a)_{i \in \{1,\ldots,d\}}$.

Example 1 (Running example)**.** *Let us consider the following linear dynamical systems in discrete-time. This corresponds to the one introduced in [4].*

$$x_0 \in [-1, 1]^2,\ x_{k+1} = Ax_k,\ \forall\, k \in \mathbb{N},\ \textit{where } A = \begin{pmatrix} 1 & 0.01 \\ -0.01 & 0.09 \end{pmatrix}\ .$$

To analyze such a system, we are looking for an over-approximation of the reachable values set i.e. the possible values taken by the state variable x_k. We propose here to construct an over-approximation of the reachable values set using the templates method [4]. More precisely, given a finite set $\mathbb{P}$ of functions from $\mathbb{R}^d$ to $\mathbb{R}$ called templates, the templates method consists in computing for each template q an associated extended real value $v(q)$ making the set $\{x \in \mathbb{R}^d \mid \forall\, q \in \mathbb{P},\ q(x) \leq v(q),\}$ an over-approximation of the reachable values set. As usual in abstract interpretation [7], the over-approximation is built from a fixed point of the abstract functional. The abstract functional is constructed from the abstract transformation of the collecting semantics in the abstract templates domain combined with Lagrange duality [1].

We specialize the templates method to quadratic templates [4]. The quadratic templates chosen here are $p_1(x,y) = x^2$ (the square of the first coordinate), $p_2(x,y) = y^2$ (the square of the second coordinate) and $p_3(x,y) = 2x^2+2xy+3y^2$ (a quadratic Lyapunov function associated with a matrix that solves the Lyapunov equation relative to the matrix A). In the rest of the paper, the notation $\mathbb{P}$ will stand for the set of quadratic templates.

Recall that we are looking for an over-approximation of the reachable values set and thus we are interested in the fixed points of the abstract functional. This abstract functional is, for all $v \in \mathbb{R}^{\mathbb{P}}$, for all $p \in \mathbb{P}$:

$$F(v)(p) = \inf_{\lambda \in \mathbb{R}_+^{\mathbb{P}}} \sup \left\{ \sum_{q \in \mathbb{P}} \lambda(q) v(q) + \sup_{x \in \mathbb{R}^2} p(Ax) - \sum_{q \in \mathbb{P}} \lambda(q) q(x), (X^0)^\dagger(p) \right\}$$

The map $(X^0)^\dagger$ is defined as follows:

$$(X^0)^\dagger(p) = \sup_{(x,y) \in X^0} p(x,y) = \sup_{(x,y) \in [-1,1]^2} p(x,y)$$

Then :

$$(X^0)^\dagger(p_1) = 1,\ (X^0)^\dagger(p_2) = 1,\ (X^0)^\dagger(p_3) = 7$$

The details on the construction of the map F are given in [1, 4].

As the number of templates is finite (equal to three), the map F can be viewed as a map with three coordinates $(F(\cdot)(p_1), F(\cdot)(p_2), F(\cdot)(p_3))$. The policies are then map from $\pi : \mathbb{P} \mapsto \mathbb{R}_+^{\mathbb{P}}$ which associates to a template a vector of Lagrange multipliers λ i.e. $\pi(p) = \lambda$. The policy map f^π is defined, for all $v \in \mathbb{R}^{\mathbb{P}}$, for all $p \in \mathbb{P}$ by:

$$f^{\pi(p)}(v)(p) = \sup \left\{ \sum_{q \in \mathbb{P}} \lambda(q) v(q) + \sup_{x \in \mathbb{R}^2} p(Ax) - \sum_{q \in \mathbb{P}} \lambda(q) q(x), (X^0)^\dagger(p) \right\} .$$

The computation of the smallest fixed point of the maps f^π is supposed to be easy; for example they can be computed in polynomial time. Those maps are often the maxima of affine maps and in this case, the smallest fixed point can be computed using linear programming(see [9]). In practice, this assumption is crucial since the principle of a policy iterations algorithm is to replace a very difficult problem (in the sense of the theory of complexity) by a sequence of simple problems. In theory, this is not necessary. In the rest of the paper, we will precise the moment when the assumption is mandatory.

To avoid that a fixed point of F has a coordinate equal to $-\infty$, we need a supplementary assumption. The map F is supposed to be lower bounded in $\mathbb{R}^d$ meaning that there exists $\underline{F} \in \mathbb{R}^d$ such that for all $x \in \mathbb{R}^d$, $\underline{F} \leq F(x)$. This assumption is natural in static analysis or in dynamical systems. Program initializations (or initial conditions) play the role of $\underline{F}$. This hypothesis implies that none of the fixed point of policy maps f^π (or F) has a coordinate equal to $-\infty$.

Moreover, we add a complementary hypothesis: we suppose that the smallest fixed point of any policy map f^π belongs to $\mathbb{R}^d$. Actually from the previous assumption, it suffices to exclude coordinates equal to $+\infty$. Note that, since $\mathbb{R}^d$ is not complete, the smallest fixed point of a map may not exist in the lattice $\mathbb{R}^d$. This assumption permits to only deal with finite values at each step of the policy iterations algorithm. This eludes discussions about the propagation of infinities and the absorption rules ($0 \times \infty$). This assumption is not so restrictive since it suffices

to keep in the set Π the policies for which the associated policy map have a smallest fixed point with only finite coordinates.

Policy iterations algorithms only computes fixed point of monotonic maps. Then, we impose that the maps f^π are monotonic i.e. $\forall\, x, y \in \mathbb{R}^d,\ x \leq y \implies f^\pi(x) \leq f^\pi(y)$. This latter assumption implies that F is also monotonic. This assumption is completely natural in stochastic games (vector of probabilities are non-negative) and in static analysis (using collecting semantics). Furthermore, the monotonicity of the policy maps permits to characterize, from Tarski's theorem, their smallest fixed point as the infimum of their post-fixed points.

We sum up all the assumptions made on the maps F and f^π as follows.

Assumption 1. *The maps F and f^π satisfy:*

1. *For all $\pi \in \Pi$, f^π are monotonic (and so F);*
2. *For all $\pi \in \Pi$, the smallest fixed point of f^π belongs to $\mathbb{R}^d$;*
3. *There exists $\underline{F} \in \mathbb{R}^d$ such that, for all $x \in \mathbb{R}^d$, $\underline{F} \leq F(x)$.*

Example 2 (Assumption satisfaction)**.** *The map F of Example 1 satisfies the first and the third statement of Assumption 1. The second does not necessarily hold but can be overcome as said earlier. Indeed, since all λ are vectors with non-negative coordinates, then the policy maps are monotonic. For the third statement, it suffices to see that $F(v) \geq (X^0)^\dagger$. As we said earlier, we can relax the second statement by only keeping the policy maps for which the smallest fixed point has finite coordinates. However we cannot easily characterize such a subclass of policies.*

From the two first assertions of Assumption 1, we can characterize the smallest fixed point of policy maps.

Proposition 1. *Let $g : \mathbb{R}^d \mapsto \mathbb{R}^d$ be monotonic. Suppose that the smallest fixed point of g belongs to $\mathbb{R}^d$. Then the smallest fixed point of g is the unique solution of $\inf\{x \in \mathbb{R}^d \mid g(x) \leq x\}$.*

Example 3 (Computation of the smallest fixed points of policy maps)**.** *Recall that the policy maps f^π are of the form:*

$$\sup \left\{ \sum_{q \in \mathbb{P}} \lambda(q) v(q) + \sup_{x \in \mathbb{R}^2} p(Ax) - \sum_{q \in \mathbb{P}} \lambda(q) q(x), (X^0)^\dagger(p) \right\} .$$

Following Proposition 1, we have to solve:

$$\inf_v \left\{ v(p_1) + v(p_2) + v(p_3) \mid \forall\, p \in \mathbb{P},\ \sum_{q \in \mathbb{P}} \lambda(q) v(q) + \eta(p) \leq v(p),\ (X^0)^\dagger(p) \leq v \right\} . \quad (2)$$

Note that minimizing $v(p_1) + v(p_2) + v(p_3)$ amounts to minimizing each $v(p_i)$ individually. The number $\eta(p)$ is equal to $\sup_{x \in \mathbb{R}^2} p(Ax) - \sum_{q \in \mathbb{P}} \lambda(q) q(x)$ which will be computed just after the computation of the policy π. Moreover, since the templates q and $x \mapsto p(Ax)$ are homogeneous quadratic forms (of the form $z \mapsto z^\intercal Q z$ where Q is a symmetric matrix) then the computation of new policy will lead to $\eta(p) = 0$.

We warn the reader that the λ present in Eq. (2) depends on the template p. Indeed, we recall that λ is actually the image of p by the policy π. Hence different templates generate different vector of Lagrange multiplier λ.

In conclusion, we see that Problem (2) is a linear program where the only decision variable is v.

Finally, to make improvements and to ensure the convergence of a policy iterations algorithm, policy iterations algorithm needs the selection property.

Definition 1 (Selection property)**.** *A map $F : \mathbb{R}^d \to \mathbb{R}^d$ of the form (1) has the selection property if and only if for all $x \in \mathcal{L}$, there exists $\pi \in \Pi$ such that :*

$$F(x) = f^{\pi}(x)$$

This property is thus equivalent to the existence of an optimal solution for a minimization problem i.e. for all $x \in \mathbb{R}^d$ for all $i \in \{1, \ldots, d\}$, the minimization problem :

$$F_i(x) = \text{Min}\{f_i^a(x) \mid a \in \Pi_i\} \tag{3}$$

has an optimal solution a^*. This is always the case when the sets Π_i are finite. When the sets Π_i are infinite, to be true, restrictive assumptions are required.

Example 4 (Slater's condition and selection property)**.** *The map F of Example 1 has the selection property if we restrict its domain to the set :*

$$\mathcal{FS} := \{v \in \mathbb{R}^{\mathbb{P}} \mid \exists\, (x, y) \in \mathbb{R}^2,\ \forall\, p \in \mathbb{P},\ v(p) - p(x, y) > 0\}\ .$$

This condition seems to be restrictive. However, we can see that for all $v \in \mathbb{R}^{\mathbb{P}}$, $F(v) \in \mathcal{FS}$. Indeed, for all $v \in \mathbb{R}^{\mathbb{P}}$, $F(v) \geq (X^0)^{\dagger} = (1, 1, 7)$ and $(0, 0)$ satisfies $p_1(0, 0) = p_2(0, 0) = 0^2 < 1$ and $p_3(0, 0) = 2 \times 0^2 + 2 \times 0 \times 0 + 3 \times 0^2 < 7$. Hence for all $p \in \mathbb{P}$, $p(0, 0) < (X^0)^{\dagger}(p) \leq F(v)(p)$.

Actually, in constrained optimization theory, we said that the elements of $\mathcal{FS}$ satisfy the Slater's condition((see for example [15, 8]).

Algorithm 1 recalls briefly a classical policy iterations algorithm. The presented algorithm can be generalized to lattices (see [9]). Algorithm 1 does not require the finiteness of the set of policies. Finiteness of the set of policies implies the finite time convergence. To have the convergence to a fixed point in the infinite case, we need upper semi-continuity [1, 2, 3].

Data: A map F of the form (1) which satisfies the selection property
Result: A fixed point of F.
1 Choose $\pi^0 \in \Pi$;
2 $k := 0$;
3 Compute the smallest fixed point x_k of f^{π^k};
4 Evaluate $F(x_k)$;
5 **if** $F(x_k) = x_k$ **then**
6 | return x_k;
7 **else**
8 | Take $\pi^{k+1} \in \Pi$ such that $F(x_k) = f^{\pi^{k+1}}(x_k)$;
9 | Increment k;
10 | Go to line 3;
11 **end**

Algorithm 1: Classical Policy iterations algorithm using the selection property

The line 8 of Algorithm 1 relies on the selection property. Indeed, when x^k is not a fixed point, we have to compute a policy which is optimal at this vector x^k.

Example 5. *Let us consider the fixed point computation based on Algorithm 1 of Example 1. Following [1], we initialize the algorithm with π^0 identically equal to the vector of Lagrange multipliers $(0, 0, 1)$, where the zeros are associated with the templates p_1 and p_2 and 1 with the quadratic Lyapunov template p_3. Using linear programming [4], we get as first vector $x_0 = (7, 7, 7)$. This is not a fixed point of F and thus since $x_0 \in \mathcal{FS}$ we compute a new policy using Semi-Definite Programming [4, 14]:*

$$\pi^1(p_1) = (0, 0, 0.596),\ \pi^1(p_2) = (0, 0, 0.3961),\ \pi^1(p_3) = (0, 0, 0.9946)\ .$$

After five iterations, we get the fixed point:

$$x_5 = (3.5, 2.333, 7)\ .$$

However, since the map F is lower bounded, then the optimal value of Problem (3) is finite as soon as F has a finite coordinates post-fixed point. By definition of the infimum in $\mathbb{R}$, for all $x \in \mathbb{R}^d$, for all $i \in \{1, \ldots, d\}$, for all $\varepsilon > 0$, there exists an element $a \in \Pi_i$ such that $f_i^a(x) \leq F_i(x) + \varepsilon$. This key idea will be used to replace optimal improving policies by ε-optimal improving policies.

This paper concerns the construction of a policy iterations algorithm using ε-optimal policies (avoiding selection property). The main issue remains in a good choice of ε to keep the convergence of the algorithm and the computation of a fixed point. Moreover, we must keep the advantages of the policy iterations algorithms that is the computation of post-fixed point at each step, the strict decrease of the post-fixed point generated and the convergence to a fixed point.

3 A policy iterations algorithm without selection property

First, we remark that we can choose an ε per coordinate of F. Thus, we will work with positive parameters in $\mathbb{R}^d$. We denote by $\mathbb{R}^d_{+*}$ the set of vectors of $\mathbb{R}^d$ with strictly positive coordinates. In a second time, to get convergence of our policy iterations algorithm, we have to impose the geometric convergence of the parameters to zero. We introduce the sequence $(\varepsilon_n)_{n\in\mathbb{N}}$ which verifies the following hypothesis:

$$\forall\, n \in \mathbb{N},\ \varepsilon_n \in \mathbb{R}^d_{+*},\ \text{ and } \varepsilon_{n+1} \leq 1/2\varepsilon_n\ . \tag{4}$$

We also assume that there exists $\varepsilon_0 \in \mathbb{R}^d_{+*}$ such that the inequality:

$$F(y) + \varepsilon_0 \leq y \tag{5}$$

admits a solution. Let us choose an arbitrary solution that we call x_0.

From, this sequence $(\varepsilon_n)_{n\in\mathbb{N}}$, We construct a policy iterations algorithm as follows:

In Algorithm 2, two new parameters appear. First, we add a stopping criteria α. As we will see at Theorem 1 the produced sequence $(x_n)_{n\geq 0}$ will satisfy $F(x_n) + \varepsilon_n \leq x_n$ and then there does not exist n such that $F(x_n) = x_n$ since for all $n \in \mathbb{N}$, ε_n has strictly positive coordinates. This α ensures the finite convergence of the algorithm. This parameter can be chosen as is usually done for classical numerical algorithms, for example $\alpha = 10^{-6}$ or $\alpha = 10^{-8}$. Second, the sequence $(\varepsilon_n)_{n\in\mathbb{N}}$ satisfying Eq. (4). In practice, we will choose the geometric sequence $\varepsilon_n = \varepsilon_0/(2^n)$. The crucial point is to choose ε_0. For the moment, this choice is heuristic and

Data: The sequence $(\varepsilon_n)_{n\in\mathbb{N}}$ satisfying (4), x_0 the chosen solution of (5) and a stopping criteria α.
Result: An approximation of a fixed point of F.

```
n = 0;
while ||x_n − F(x_n)||_∞ > α do
    Define π_n such that: f^{π_n}(x_n) ≤ F(x_n) + ε_{n+1};
    if n > 0 and f^{π_{n−1}}(x_n) < f^{π_n}(x_n) then
        π_n := π_{n−1};
    end
    Compute the smallest fixed point x_{n+1} of f^{π_n} + ε_{n+1};
    Do n = n + 1;
end
```

Algorithm 2: Policy iterations algorithm without selection property

a deeper analysis should be performed. We remark that ε_0 can be determined from Eq. (5). Then we modify the initial vector of the classical policy iterations algorithm, x_0. If x_0 verifies $x_0 - F(x_0) > 0$ then we set $\varepsilon_0 = x_0 - F(x_0)$. Otherwise, for the coordinates of x_0 such that $x_{0,i} = F_i(x_0)$ we add a strictly positive real (1 for the running example). The choice of this positive real has a big influence on the performance of the algorithm. To choose a too big real increases the number of iterations whereas a too small real can generate numerical accuracies.

Example 6. *Now we use Algorithm 2 to compute a fixed point of the map F defined at the running example Example 1.*

The first modification is that we cannot initialize with the same $x_0 = (7, 7, 7)$ that we used in Example 5. Indeed, since for all $v \in \mathbb{R}^{\mathbb{P}}$, $F(v) \geq (1, 1, 7)$ and $x_0(p_3) = 7$, there does not exist ε_0 with strictly positive coordinates such that $F(x_0) + \varepsilon_0 \leq x_0$. Then we use a different x_0 and we choose $x_0 = (7, 7, 8)$ and we set $\varepsilon_0 = x_0 - F(x_0) > 0$ since, from Semi-Definite Programming, $F(x_0) = (4.7683, 3.1685, 7.9565)$. The sequence ε_n is thus defined by $\varepsilon_0/(2^n)$.

Now we compute a first policy π^0 such that $f^{\pi^0}(x_0) \leq F(x_0) + \varepsilon_0/2$. The policy returned by the Semi-Definite Programming solver (Mosek [5] interfaced with Yalmip [12] in Matlab):

$$\pi^0(p_1) = (0.1816, 0.0453, 0.5082),\ \pi^0(p_2) = (0.0889, 0.1735, 0.3689),$$
$$\pi^0(p_3) = (0.0015, 0.0009, 0.9941)$$

The smallest fixed point of $f^{\pi^0} + \varepsilon_0/2$ is given, using linear programming by:

$$x_1 = (6.0606, 6.1043, 7.0218)$$

After 25 iterations of Algorithm 2, we get the same (truncated) fixed point found with the classical policy iterations.

The number of iterations grows due to the small steps realized by the new policy iterations algorithm. Indeed, the computed policy are sub-optimal and the smallest fixed points of policy maps are perturbed by ε_{n+1}. So we cannot decrease as much as the original algorithm. However, with more complex templates (piecewise quadratic or polynomial) where Slater's condition cannot be ensured, the presented algorithm is the one to use.

Definition 2 (Upper semi-continuous function)**.** *A map G is upper-semicontinuous if and only if for all $y \in \mathbb{R}^d$ for all sequences $(y_n)_n$ that converge to y then:*

$$\limsup_{n\to+\infty} G(y_n) \leq G(y)$$

Compared to the original policy iterations algorithm, the modified policy iterations keeps the same mathematical properties up to the term of the sequence $(\varepsilon_n)_{n\in\mathbb{N}}$.

Theorem 1 (Convergence of Modified Policy Iteration)**.** *The following statements hold:*

1. *For all $n \in \mathbb{N}$, $F(x_n) + \epsilon_n \leq x_n$*
2. *The sequence $(x_n)_{n\in\mathbb{N}}$ is strictly decreasing and converges.*
3. *The limit x_∞ of $(x_n)_{n\in\mathbb{N}}$ satisfies $F(x_\infty) \leq x_\infty$.*
4. *If F is upper-semicontinuous, then x_∞ is a fixed point of F.*

Proof. 1. Let $n \in \mathbb{N}$. If $n = 0$, then the result holds by hypothesis. Now, let $n \in \mathbb{N}^*$. We have, by definition of F,

$$F(x_n) + \varepsilon_n \leq f^{\pi_{n-1}}(x_n) + \varepsilon_n = x_n$$

The last equality follows from the fact that x_n is the smallest fixed point of $f^{\pi_{n-1}}$.

2. We have to prove that for all $n \in \mathbb{N}$, $x_{n+1} \leq x_n$ and $x_{n+1} \neq x_n$. Let $n \in \mathbb{N}$. From Prop. 1, it suffices to show that $f^{\pi_n}(x_n) + \varepsilon_{n+1} \leq x_n$. Indeed, x_{n+1} being the smallest point of the set $\{x \mid f^{\pi_n}(x) + \varepsilon_{n+1} \leq x\}$, the conclusion follows.

   Let us prove that $f^{\pi_n}(x_n) + \varepsilon_{n+1} \leq x_n$. By definition of π_n, we have $f^{\pi_n}(x_n) + \varepsilon_{n+1} \leq F(x_n) + 2\varepsilon_{n+1}$. Note that, even if $f^{\pi_{n-1}}(x_n) < f^{\pi_n}(x_n)$, we get the same inequality.

   By definition of the sequence (ε_n), we get $f^{\pi_n}(x_n) + \varepsilon_{n+1} \leq F(x_n) + \varepsilon_n$. We conclude from the first statement, that $f^{\pi_n}(x_n) + \varepsilon_{n+1} \leq x_n$.

   Now suppose that $x_{n+1} = x_n$. Then $f^{\pi_n}(x_{n+1}) + \varepsilon_{n+1} = f^{\pi_{n-1}}(x_n) + \varepsilon_n$. This leads to $f^{\pi_n}(x_n) = f^{\pi_n}(x_{n+1}) = f^{\pi_{n-1}}(x_n) + \varepsilon_n - \varepsilon_{n+1} > f^{\pi_{n-1}}(x_n)$. According to line 5 of Algorithm 2, we have $\pi_n = \pi_{n-1}$ and thus $x_{n+1} = x_n$ implies that $f^{\pi_n}(x_n) + \varepsilon_{n+1} = f^{\pi_n}(x_n) + \varepsilon_n$ and $\varepsilon_{n+1} = \varepsilon_n$ which is not possible.

   The maps F is supposed to be lower bounded, then, from the first statement, $(x_n)_n$ is also lower-bounded. Then since the sequence $(x_n)_n$ strictly decreases then it converges to x_∞.

3. The map F is monotonic and $(x_n)_n$ is strictly decreasing then $F(x_\infty) \leq F(x_n)$ for all $n \in \mathbb{N}$. From the first statement, it follows that $F(x_\infty) \leq x_n$ for all $n \in \mathbb{N}$. Taking the limit as n tends to $+\infty$ leads to $F(x_\infty) \leq x_\infty$.

4. For all $n \in \mathbb{N}$, $x_{n+1} \leq x_n$ and by monotonicity of f^{π_n} we have $f^{\pi_n}(x_{n+1}) \leq f_n^{\pi}(x_n) \leq F(x_n) + \varepsilon_{n+1}$ by definition of π_n (the case where $f^{\pi_{n-1}}(x_n) < f^{\pi_n}(x_n)$ leads to the same conclusion). Hence, $x_{n+1} = f^{\pi_n}(x_{n+1}) + \varepsilon_{n+1} \leq F(x_n) + 2\varepsilon_{n+1}$. Taking the limsup as n tends to $+\infty$ of both sides implies that $x_\infty \leq \limsup_{n\to+\infty} F(x_n)$. Thus $x_\infty \leq F(x_\infty)$ from the upper semi-continuity of F. We conclude from the third assertion that $x_\infty = F(x_\infty)$.

□

The modified policy iterations algorithm keeps the same advantages of the original policy iterations algorithm.

Corollary 1. *1. The modified policy iterations algorithms still computes a sequence of valid invariants which are more and more precise.*

2. *The modified policy iterations algorithms can still be stopped at any iteration step with a valid invariant.*

The advantage to avoid selection property in policy iterations algorithm is double. First, this allows to replace optimization problem by feasible problems to compute a new policy. In a second time, this allows to introduce guarantees in the sense of numerical accuracy.

4 Conclusion and Future Works

In this paper, we succeed to construct a policy iterations algorithm avoiding the selection property. The constructed algorithm keeps the same advantages of the original policy iterations algorithm. We replace optimal policies by ε-optimal policies which always exist. To avoid the selection property permits to construct a policy iterations algorithm in situations where we are not able to prove the existence of optimal solutions. Those situations happen when the number of policies is infinite.

To make our algorithm converge, we impose a geometric decrease for the ε parameters. The price to pay is the increase of the number of iterations (for the running example, from 5 to 25). Indeed, the presented algorithm makes small steps since the policy is sub-optimal.

This approach might also be used as a guaranteed method if we constraint the computed policies and abstract elements to have floating number coordinates. Indeed, to allow sub-optimal policies permits to only regard policies with floating numbers in the theoretical development. However, in the presented algorithm, the smallest fixed points of policy maps are still the optimal solutions of some linear programs. In order to allow sub-optimal solutions with floating number coordinates in the theory, we should differently tune the perturbation parameter which appears in the computation of smallest fixed points of policy maps.

References

[1] A. Adjé. Policy iteration in finite templates domain. *Electronic Notes in Theoretical Computer Science*, 317:3 – 18, 2015. The Seventh and Eighth International Workshops on Numerical Software Verification (NSV).

[2] A. Adjé. Coupling policy iterations with piecewise quadratic lyapunov functions. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control, HSCC 2017, Pittsburgh, PA, USA, April 18-20, 2017*, pages 143–152, 2017.

[3] A. Adjé, P.-L. Garoche, and V. Magron. A sums-of-squares extension of policy iterations. *Nonlinear Analysis: Hybrid Systems*, 25:60 – 78, 2017.

[4] A. Adjé, S. Gaubert, and E. Goubault. Coupling policy iteration with semi-definite relaxation to compute accurate numerical invariants in static analysis. *Logical Methods in Computer Science*, 8(1), 2012.

[5] E. D. Andersen and K. D. Andersen. *The Mosek Interior Point Optimizer for Linear Programming: An Implementation of the Homogeneous Algorithm*, pages 197–232. Springer US, Boston, MA, 2000.

[6] A. Costan, S. Gaubert, E. Goubault, M. Martel, and S. Putot. A policy iteration algorithm for computing fixed points in static analysis of programs. In *International Conference on Computer Aided Verification*, pages 462–475. Springer, 2005.

[7] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, California, 1977. ACM Press, New York, NY.

[8] R. G Eustaquio, E. W Karas, and A. A Ribeiro. Constraint qualifications for nonlinear programming. *Federal University of Parana*, 2008.

[9] S. Gaubert, E. Goubault, A. Taly, and S. Zennou. Static analysis by policy iteration on relational domains. In Rocco De Nicola, editor, *Programming Languages and Systems*, pages 237–252, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[10] A. J. Hoffman and R. M. Karp. On nonterminating stochastic games. *Management Science*, 12(5):359–370, 1966.

[11] R. A. Howard. Dynamic programming and markov processes. 1960.

[12] J. Löfberg. Yalmip : A toolbox for modeling and optimization in matlab. In *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[13] Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13. Siam, 1994.

[14] M. V. Ramana and P. M. Pardalos. *Semidefinite Programming*, pages 369–398. Springer US, Boston, MA, 1996.

[15] M. Slater. Lagrange multipliers revisited. *Cowles Commission Discussion*, 403, 1950.