Discount and speed/execution tradeoffs in Markov Decision Process Games.

Reinaldo Uribe, Fernando Lozano, Katsunari Shibata and Charles Anderson

***Abstract*— We study Markov Decision Process (MDP) games with the usual ± 1 reinforcement signal. We consider the scenario in which the goal of the game, rather than just winning, is to maximize the number of wins in an allotted period of time (or maximize the expected reward in the same period). In the reinforcement learning literature, this type of tradeoff is often handled by tuning the discount parameter in order to encourage the learning algorithm to find policies that take fewer steps on average, at the cost of a lower probability of winning. We show that this approach is not guaranteed to solve the tradeoff problem optimally, and hence a different strategy is needed when tackling this type of problems.**

I. Introduction

Suppose there is a game for which, after years of practice, you know a nearly execution–optimal strategy, that allows you to win 99% of the episodes played. Notice that, for practical purposes, this means you have *solved* the game (or are close to a "weak solution" in the sense of [1]). Further, suppose the mean episode length for that strategy is 1000 "moves" (for example, die rolls or ply), each of which takes, for instance, 6 seconds. If the "natural" [2] reinforcement learning rewards of $+1$ on winning, -1 on losing, and 0 at all other times translate into receiving \$1, paying \$1, and no payout, respectively, every 10 hours played, you would win 0.99 episodes on average, with an expected profit of

$$\$1 \times 0.99 - \$1 \times 0.01 = \$0.98.$$

Now, if your brother and usual sparring partner had learned a further-from-optimal policy, in which two thirds of the games are won but with a mean episode length of only 50 moves, he would win an average 6.67 episodes and \$3.33 in the same ten-hour period. Observe that, while the second strategy leads to worse execution—it has a lower winning probability—, in the long run, across a large enough number of episodes, it outperforms the near-optimal policy in both number of victories and the received payout *per time unit*. Thus, tradeoffs between execution (the probability of winning an episode) and speed (the inverse of mean episode length) can be desirable—if you wanted to make a living from the game in the example above, for instance.

This paper explores why, in many cases, the policies found for games using a ± 1 reward scheme favour agent caution, resulting in overlong episodes of high execution, i.e., in poor tradeoff. It also shows that *discount*, the traditional approach to prompt the discovery of faster policies to the detriment of execution, is not guaranteed to optimally solve tradeoff problems of the type wins per unit step (hereon named *natural tradeoff*) or expected reward per time step (*greedy tradeoff*). This is done by introducing a transformation of the policy space to a subset of $\mathbb{R}^2$ in which undiscounted tradeoff level sets are linear whereas discounted value fails to have a functional dependency on policy features.

Reinaldo Uribe and Fernando Lozano are with the Dept. of Electrical and Electronic Engineering, University of the Andes. Katsunari Shibata is with the Dept. of Electrical and Electronic Engineering, Oita University. Charles Anderson is with the Dept. of Computer Science, Colorado State University. The corresponding author is Reinaldo Uribe, email: r-uribe@uniandes.edu.co

The focus of this paper are Markov Decision Process (MDP) games. An MDP game comprises the same elements of a common MDP, that is, a set of states observable by an agent who can take actions that, with certain transition probabilities assumed to hold the Markov Property, lead it from one state to another, observing a real-valued reinforcement. Additionally, in the MDP game there is a set of *terminal* states partitioned into "win" and "loss". For convenience, the custom is to assume transition from any terminal state to itself with probability 1 and reinforcement 0.

The reinforcement must provide the agent a representation of the *goal* of the game. The usual approach to solve this kind of games is to set a reinforcement scheme of 0 on transition to nonterminal states, $+1$ on winning and -1 (or 0) on losing. Reinforcement learning seeks the policy (mapping from the state set to the action space, i.e. what action to take at each point) that maximizes the expected reward from each state. Observe that in both cases (± 1 and $\{+1, 0\}$ rewards), solving the reinforcement learning problem is equivalent to finding the game strategy that maximizes execution.

In a game, the expected cumulative reward of a deterministic policy $\pi \in \Pi$ (where Π is the policy space), called the *value* of the policy, satisfies for all states the recursive relationship [2]

$$V^\pi(s) = \sum_{s'} P^\pi_{ss'} \left[R_{s'} + \gamma V^\pi(s') \right], \tag{1}$$

where s is an arbitrary nonterminal state, s' are the states reachable from s, $P^\pi_{ss'}$ is the probability of going from s to s' following π, $R_{s'}$ is the reward received on transition to s' and $0 \leq \gamma \leq 1$ is a *discount factor* (with $\gamma = 1$ corresponding to no discount). Equation (1) states that the value of a state under a policy is the average of the sum of 1-step reinforcements and the value of the next state, weighted by the probability of reaching each possible next state.

A discount $\gamma < 1$ is frequently used, as it ensures that the infinite sum of future rewards is bounded—assuming infinite episodes are possible—and, importantly, it makes instant rewards more desirable, since rewards k steps in

978-1-4577-0011-8/11/$26.00 ©2011 IEEE

the future are worth only γ^{k-1} what they would be if received immediately. Thus, discount is *in principle* a way to overcome the problem of having execution-optimal policies that yield very long episodes.

The structure of the paper is as follows: section II presents the necessary mathematical background and introduces a formalization of winning probability and mean episode length of a policy as variations of equation (1). It also introduces a linear method of computing the features of a policy. In section III a sample game, Decision Snakes and Ladders is introduced, and its discounted and undiscounted performance related to the natural tradeoff problem is discussed in depth. A non-value-optimal policy is presented and shown to have better natural tradeoff performance than any discount-optimal strategy. Section IV introduces a transformation of the policy space to a subset of $\mathbb{R}^2$ in which a whole family of tradeoff problems, including natural and greedy, are linear on policy features that depend only on speed and execution. Decision Snakes and Ladders and an additional example are used to show that even if *some* discount can *sometimes* improve the tradeoff performance, since there is no functional dependency between discounted value and tradeoff, it has no guarantees of optimally solving tradeoff problems.

II. Computation of a policy's features

This section presents extensions of (1) to the winning probability and mean episode length of a given policy. A linear method for solving the resulting recursive relationships is presented. Nevertheless, notice that such method is useful only as long as it is feasible to invert a matrix of size $|S|\times|S|$ (that is, with one column per state). Otherwise an iterative computation method may be necessary. All of the following computations correspond to an arbitrary, given policy π. For notational simplicity, the index π is omitted hereon. The linear solution of (1) is commonly known. The solutions of winning probability and mean episode length are similar to [3], although the derivation varies somewhat.

A. Preliminary definitions

Let, for an MDP game with terminal-state set S_t partitioned into "win" (S_{win}) and "loss" (S_{loss}) states, the reinforcements $\mathbf{r}(s')$, nonterminal-state indicator $\mathbf{i}_{nt}(s)$, and "win" indicator $\mathbf{i}_w(s)$ vectors be

$$\mathbf{r}(s') = \begin{cases} 1 & s' \in S_{win} \\ -1 & s' \in S_{loss} \\ 0 & \text{otherwise,} \end{cases}$$

$$\mathbf{i}_{nt}(s) = \begin{cases} 1 & s \notin S_t \\ 0 & \text{otherwise,} \end{cases}$$

$$\mathbf{i}_w(s) = \begin{cases} 1 & s \in S_{win} \\ 0 & \text{otherwise.} \end{cases}$$

We use the after-state reward notation (s') in the reinforcement vector $\mathbf{r}$ to emphasize that the nonzero rewards are received *on transition* to the terminal states. Additionally, define the *Modified transition matrix* as the $|S| \times |S|$ matrix $\mathbf{T}$ with entries

$$\mathbf{T}_{ss'} = \begin{cases} 0 & i = j;\ i \in S_t \\ P_{ss'} & \text{otherwise.} \end{cases}$$

Notice that if $\mathbf{e}$ is a vector of ones of appropriate size, then $\mathbf{T}\,\mathbf{e} = \mathbf{i}_{nt}$.

B. Value

Observe that, as a consequence of the convention of having terminal states transition to themselves with probability 1 and reinforcement 0, the recursive value formula (1) can be expressed in vector form as

$$\begin{aligned} \mathbf{v}(s) &= \sum_{s'} \mathbf{T}_{ss'}\,(\mathbf{r}(s') + \gamma\mathbf{v}(s')), \\ \mathbf{v} &= \mathbf{T}\,(\mathbf{r} + \gamma\mathbf{v}), \end{aligned}$$

which results in

$$\mathbf{v} = (\mathbf{I} - \gamma\mathbf{T})^{-1}\,\mathbf{T}\,\mathbf{r}, \tag{2}$$

and, in the undiscounted case

$$\mathbf{v} = (\mathbf{I} - \mathbf{T})^{-1}\,\mathbf{T}\,\mathbf{r}, \tag{3}$$

C. Winning probability and mean episode length

Analogously to (1), the duration of the episodes from an arbitrary state s following a given policy can be expressed as the average of *one step* plus the duration from all possible next states, weighted by the probability of reaching those states, with the duration of an episode after it has terminated being zero by definition ($\mathbf{d}(s \in S_t) = 0$). Thus,

$$\begin{aligned} \mathbf{d}(s) &= \sum_{s'} \mathbf{T}_{ss'}\,(1 + \mathbf{d}(s')), \\ \mathbf{d} &= \mathbf{T}\,(\mathbf{e} + \mathbf{d}), \\ \mathbf{d} &= (\mathbf{I} - \mathbf{T})^{-1}\,\mathbf{i}_{nt}. \end{aligned} \tag{4}$$

From a nonterminal state s, the possibility of winning is twofold. Its first constitutive element is the cumulative probability of winning in one step and the second element is the sum of the winning probabilities of all possible nonterminal next states weighted by the probability of reaching each of them,

$$\begin{aligned} \mathbf{p}_w(s) &= \sum_{s' \in S_{win}} \mathbf{T}_{ss'} + \sum_{s'' \notin S_t} \mathbf{T}_{ss''}\,\mathbf{p}_w(s''), \\ \mathbf{p}_w &= \mathbf{T}\,\mathbf{i}_w + \mathbf{T}\,\mathbf{p}_w, \\ \mathbf{p}_w &= (\mathbf{I} - \mathbf{T})^{-1}\,\mathbf{T}\,\mathbf{i}_w. \end{aligned} \tag{5}$$

In all examples below, discounted and undiscounted values, mean episode lengths, and winning probabilities are computed using (2), (3), (4), and (5), respectively. To extract the features from the initial state, the full vectors are computed and the corresponding element chosen.

III. Discounted Decision Snakes and Ladders

Recall the children's board game *Snakes and Ladders*. It is played on a board with—usually 10×10—numbered squares, and the goal is to reach the final square. Players start outside the board (or on square 1) and take turns at throwing one or two dice and advancing a token the resulting number of steps. Snakes and ladders are drawn between pairs of squares in the board. If a player lands on a snake's mouth, it must move back to its tail. Likewise, landing on a stair's bottom pushes a player forward to its top. Observe that, as described, the game requires no skill, only luck, and is a Markov chain with simple well-defined transition probabilities and a single absorbing state, the goal.

Nevertheless, it can easily be transformed into a Markov Decision Process by including a set of actions available at each state. Beaudry et al. [4], for example, proposed a variation in which the player can choose between advancing one step or throwing one die or two, in order to reach the goal square as quickly as possible. Observe that the resulting process, rather than a *game* as defined above, is a maze that the player strives to exit in the smallest number of steps.

An alternative action scheme is explored here: to decide, prior to the throw, whether to advance or retreat the number of squares indicated by a single die roll. Further, the process is made a game proper with the inclusion of at least one terminal "loss" state. Figure 1 shows an instance of a Decision Snakes and Ladders game, based on a 1901 board (from the National Archives of Australia: A1786, 9557B). The game is won reaching (blue) squares 100 or 80, and lost reaching (red) squares 23, 37, 45, 67, or 89. Following the standard rules of the original game, any move that would take the player off the board leaves the token in the same place (but still counts as one step).

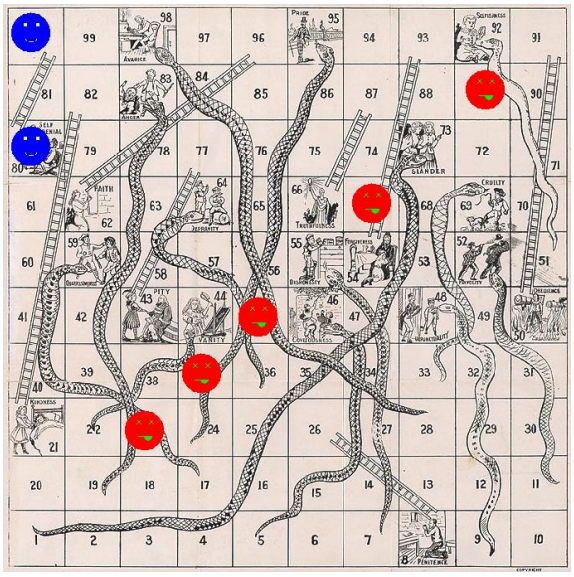

Fig. 1. An instance of Decision Snakes and Ladders.

A. The Undiscounted Optimal Policy

Assuming a ±1 reinforcement scheme and no discount,the solution to this game, that is, the policy that maximizes the probability of reaching the "win" states, can be readily found by any reinforcement learning method. A simple implementation of Value Iteration [2] finds the solution—shown in Figure 2—in seconds.

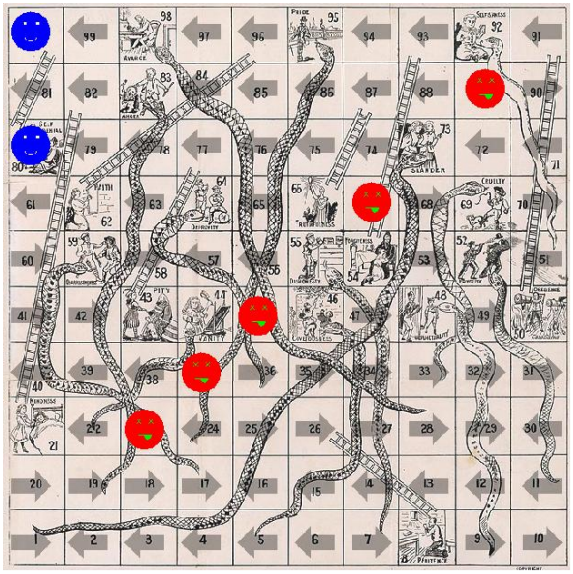

Fig. 2. The undiscounted optimal policy is to move in the direction of the arrows.

Observe that, after settling on any fixed policy, including of course the value-optimal, the game becomes a Markov chain again. Thus, it is possible to compute the mean episode length and winning probability, using the formulae from section II. The policy in Figure 2 has a winning probability of 0.9722 from the 0-state outside the board. Notice that execution is not perfect because a sequence of die rolls such as 1–6–1–4–6–1 results in a loss. Still, over 97% of the instances are won following this policy, although the mean episode duration is rather long: 84.58333 steps. This means that, if as above, one die throw/move takes 6 seconds, a reward scheme of $\$\pm1$ results in about 6.90 episodes won, out of 7.09 played every hour, for a net income of \$6.70.

We maintain this is a remarkably long policy: Recall that the goal of the game is defined—trivially—as *to win*. From the optimal policy shown in Figure 2, *winning* can be divided into three distinct types of behaviours:

- **Seek "win"**: Notice that the action for all valid states between 68 and 88 point towards the "win" state at square 80. Likewise, all valid states from 90 onwards point to square 100.
- **Avoid losses**: This is most noticeable around the red squares 37, 67 and 89. The optimal policy is such that none of them can ever be reached, as the actions of all valid states within six squares of them point outwards. In fact, square 45 is also out of reach under this policy and the disruption around 23, the only square where the

agent can lose, is caused by the strong attraction of the stair at 21, leading straight to the vicinity of the wins.

These behaviours are not surprising; this is exactly what can be expected from a policy that maximizes the probability of winning (and minimizes that of losing). However, the third behaviour is the most noteworthy:

- **Stay safe**: Observe that the actions of the squares 9–22 form a "vortex" centered around squares 16 and 17. Once any state in that range has been reached, the agent will tend to circle the center until the appropriate combination of die outcomes puts the token on 16 and is followed by a 5, sending it straight to square 82. During a stay inside the vortex, the agent is not so much moving towards the wins or away from the losses, as waiting for the—relatively rare—chance to safely move forward.

Moreover, 9–22 is not the only vortex of the optimal policy; similar combinations occur at states 56–66 (centered at 57–58), 25–36 (centered at 29–30) and 1–6 (centered at 1–2). This initial vortex is the most remarkable, as the agent cannot elude it and is immersed in it from the start of the game. Thus, beginning every game, the agent must spend an average of 42 steps ostensibly *doing nothing*, waiting for the right die throws to put it on 1 followed by a 6 to get it to square 7, from where it will either get a 1 and move to the 25–36 vortex or any other number and into the 9–22 one. Furthermore, the only way out once state 26 has been reached is to get a 6 from state 30 unto 24, from where the player can lose with a 1 on the die, go to 82 with a 2, or enter

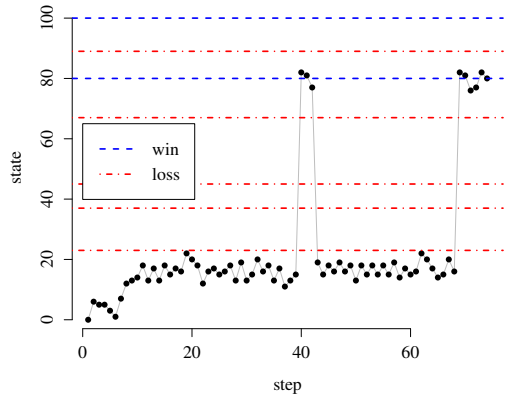

Fig. 3. Sample episode following the undiscounted optimal policy.

Hence, following the optimal policy, episodes—such as the sample one shown in Figure 3—follow with high probability the predictable path of long stays in the vortices, with a number of returns to the one between squares 9–22, before eventually escaping through the 21–82 stair and winning. Since the policy comes from the value iteration algorithm with a certificate of optimality, these long *safe* circles around the vortex centers are required to ensure the highest possible chance of winning, although they do seem rather *wasteful*.

This kind of behaviour can be termed "not losing, even if not (yet) winning" and conceptually fits the definition of an undiscounted optimal strategy: to circulate safe areas for as long as is necessary to ensure that the "loss" states are likely to be skipped, irrespective of whether that also extends the expected time to safely win. This is the reason why the pure (undiscounted) optimal game policies yield high execution in long episodes and discount is used.

B. Discount

Is it really necessary to spend a full half of the average episode waiting to get to the seventh square, and then merely jump into one of two other vortices? Obviously, it is for the policy to be execution-optimal, but assume that a mean of 84 steps per episode is prohibitively expensive. Suppose, for example that the problem at hand is natural tradeoff, that 6.9 wins per hour are too few. The common approach to find policies that transit "the shortest path through the state space" [5], "goading the player into trying to win sooner rather than later" [6], and "finding the target at a lower [time]" [7] is to discount the reward with a factor $\gamma < 1$ in equation (1).

A review of the reinforcement learning literature shows that a usual—if seldom justified—discount rate is $\gamma = 0.9$ [6] [8] [9] [10] [11] [12] [13] [14] [15] [16]. A little tweaking with the parameter shows that values around $\gamma = 0.95$ are better in Decision Snakes and Ladders. The resulting discounted optimal policy, shown in Figure 4, gets rid of the initial vortex, making the optimal action for squares 1 to 16 to move forward, and is riskier in squares 25 and 26, pointing towards the stair bottom at 21 at the danger of landing on 23 and losing (which could only happen from 24 before). The new optimal policy yields a mean episode length of *only* 35.6089 steps and a winning probability of 0.9392, so speed has more than doubled at the small price of a loss of under 5 points in execution. In our example of 6 seconds per die throw, after discount, the number of wins per hour increases from 6.90 to 16.85.

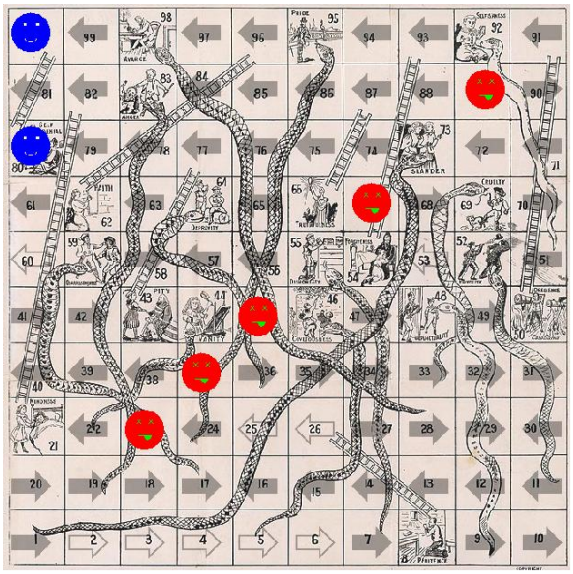

Fig. 4. Discounted optimal policy, $\gamma = 0.95$. Outlined arrows correspond to actions that differ from the undiscounted optimal.

C. Better than (discounted-)optimal

The preceding results show that using a discount rate smaller than 1 does indeed improve the temporal performance and, as expected, pushes the player to *winning sooner*—albeit slightly less frequently. Still, the discount factor γ *is* a parameter, and it does have to be set somehow. Furthermore, consider the policy pictured in Figure 5.

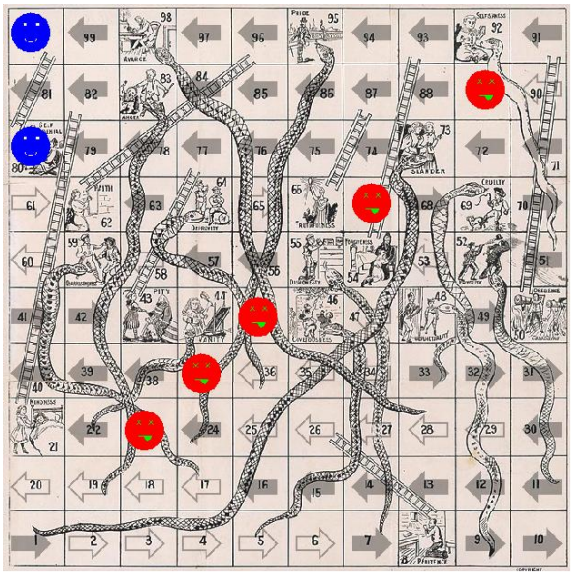

Fig. 5. A non-optimal policy with better natural tradeoff performance than the discounted and undiscounted optimal policies. Outlined arrows correspond to actions that differ from the undiscounted optimal.

Notice that it is a much less *cautious* policy than the undiscounted optimal; the squares 1–33 are now a giant vortex centered at the stair bottom in square 21. Typical episodes under this policy will head towards that square at a heightened risk of falling on 23—now reachable in one die throw from 17–20 and 24–26—and losing. It turns out to yield an ostensibly lackluster execution, with a winning probability of only 0.48673. However, with a mean episode length of 11.17627—equivalent to 26.13 wins per hour on average with 6-second plays—it far surpasses both the un- and discounted-optimal policies.

It can be proved, although that exceeds the scope of this paper, that the policy in Figure 5 is *natural-tradeoff-optimal*, but it suffices here to show it outperforms discount-optimal policies, and to call it "tradeoff-better". Table I summarizes the speed, execution and tradeoff ratio for both optimal and the tradeoff-better policies.

Figure 6 shows the winning probability/episode length ratio of the optimal policies found as the discount factor varies. For comparison, the ratios of the undiscounted-optimal and tradeoff-better policies are highlighted. Observe that the policy discussed above for $\gamma = 0.95$ remains optimal for discounts in the $\sim [0.928, 0.987]$ range and has the best tradeoff performance of the discounted-optimal policies. It is, nevertheless, quite far from the best known, corresponding to the *reckless* policy of Figure 5.

TABLE I
SUMMARY OF THE POLICIES DISCUSSED

Policy	Winning probability $\mathbf{p}_w$	Mean episode length $\mathbf{d}$	Natural tradeoff ratio $\frac{\mathbf{p}_w}{\mathbf{d}}$
Undiscounted-optimal	0.97222	84.58333	0.01149
Discounted-optimal	0.93918	35.60892	0.02637
'Tradeoff-better'	0.48673	11.17627	0.04355

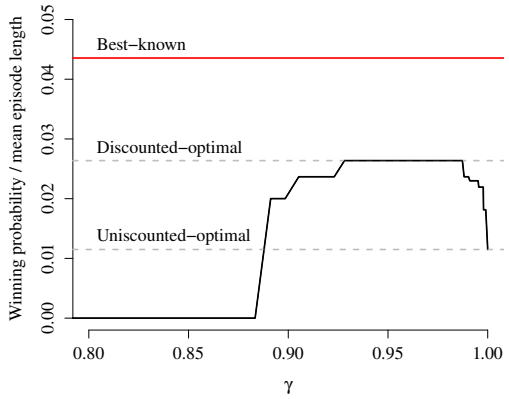

Fig. 6. Winning probability / mean episode length ratio of the value-optimal policy as γ varies.

Remarkably, the main perceived advantage of using discount, making infinite episodes tractable, acts here to the detriment of the secondary perceived advantage, urging the player to win sooner. Observe that *too much discount*, $\gamma < 0.89$ in Figure 6, results in an optimal policy that has infinite mean episode duration—and hence 0 tradeoff ratio. This happens because, compared with the undiscounted-optimal of Figure 2, the policy found keeps the vortex between squares 9–22, maintaining the "loss" at 23 out of reach while aiming for the stair at 22, and at the same time creates an inescapable vortex between 24 and 35, reachable from the stair at square 7 and leading to infinite episodes. Overall it is a policy that never loses, winning relatively quickly when it does win, or else entering an infinite loop.

For all values of $\gamma < 1$, particularly those yielding policies with zero tradeoff ratio, *winning sooner* is inevitably tied to *losing slower, or not at all*, which, opening the door to infinite episodes, is not good when the goal is to use the inverse mean episode length as an indicator of performance.

IV. Making tradeoff problems linear: the $w - l$ space

In this section we introduce a framework for transforming the policy space of a ± 1-reinforced MDP into $\mathbb{R}^2$ in such a way that a general class of tradeoff problems corresponds to discrete linear problems in the resulting space. The example from the preceding section is studied under this new outlook and a further example is used to prove that there is no

functional dependency between tradeoff values and discount, so the use of a discount factor is not guaranteed to optimally solve any tradeoff problems.

This far, an implicit assumption is that games have a known, unique initial state, from which the winning probabilities and mean episode lengths are measured. In Decision Snakes and Ladders the initial state lies "outside the board" on an imaginary square 0 from where the first move starts. Generally a single initial state s_0 can be assumed to exist, either as an actual unique start (as in Chess), or as a *dummy* state with a single action and transition probabilities representing the distribution of all possible game starts (permutations of cards from a deck, for example).

Let $p_w^\pi(s_0)$—or equivalently, just p_w— represent the probability of reaching a "win" terminal state from s_0 following an arbitrary policy $\pi \in \Pi$, i.e., the probability of winning the game following that policy. Likewise, let $d^\pi(s_0)$, simplified as d, represent the mean duration (in steps) of the episodes played following π.

We define the values w and l of an arbitrary element of the policy space (index π omitted hereon) as the duple

$$(w,\, l) = \left(\frac{p_w}{d},\, \frac{1-p_w}{d}\right).$$

Observe that, since $p_w \in [0,\, 1]$ and, except for trivial games won or lost before starting, $d \geq 1$, the following properties must hold:

- $w,\, l \geq 0$,
- $w + l = \frac{1}{d} \leq 1$,
- $p_w = 1 \Rightarrow l = 0$,
- $p_w = 0 \Rightarrow w = 0$,
- $\lim_{d\to\infty}(w,\, l) = (0, 0)$.

Hence, any policy can be represented in the $w - l$ space as a point inside the triangle with vertices $(0,\, 0)$, $(0,\, 1)$, and $(1,\, 0)$. Further, all policies with infinite episodes—and thus infinite mean episode lengths—are collapsed into the origin, whereas policies that terminate in just one transition form the line $w + l = 1$, the diagonal of the triangle. Also the w and l axes—excluding the origin—correspond respectively to policies that always win and always lose. Figure 7 summarizes these properties in the $w-l$ coordinate system.

The inverse transformation, from $w - l$ to winning probabilities and mean episode lengths is

$$(p_w,\, d) = \left(\frac{w}{w+l},\, \frac{1}{w+l}\right).$$

Consider a game with a terminal reinforcement $r_w \geq 0$ on winning and $-r_l \leq 0$ on losing, not both simultaneously equal zero. For value to equal winning probabilities, for example, $r_w = 1$ and $r_l = 0$; in the usual scheme, $r_w = r_l = 1$. Observe that for an arbitrary policy the undiscounted value of the game (from its initial state) would then be the sum of r_w when it wins (with probability p_w) and $-r_l$ when it loses (with probability $1 - p_w$),

$$v = (r_w + r_l)p_w - r_l. \tag{6}$$

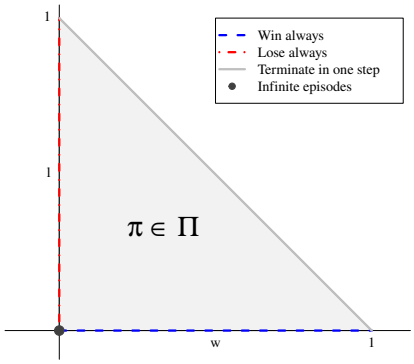

Fig. 7. Policy $w - l$ space.

Notice that even for asymmetric terminal rewards, undiscounted value is a linear function of the winning probability, so maximizing value equals maximizing performance in all cases. In the $w - l$ space, equation (6) becomes

$$v = \frac{r_w w - r_l l}{w + l}$$

Now consider the generic tradeoff problem of maximizing the received payout per unit of time. The model problem has the form:

$$\max_{\pi\in\Pi}\left(\frac{v}{d}\right) = \max_{\pi\in\Pi}\left(\frac{(r_w + r_l)p_w - r_l}{d}\right), \tag{7}$$

which in the $w - l$ space simply becomes

$$\max_{\pi\in\Pi}\left(r_w w - r_l l\right). \tag{8}$$

All tradeoff problems of the class discussed, then, are linear in $w - l$. For example, natural tradeoff, the maximum number of wins per step, is *simply* the policy with largest w component. Likewise, greedy tradeoff, maximum received payout per unit of time in a ± 1-reinforced game, is a linear problem with level sets parallel to the $w = l$ line.

Naturally, realizing that tradeoff problems are linear in the $w - l$ space does not (yet) bring us any closer to solving them. Recall that reinforcement learning just *knows* how to maximize expected cumulative (un)discounted rewards, which so far have only been shown to match winning probabilities when $\gamma = 1$. Technically, tradeoff problems remain, non-linear, non-reinforcement-learning problems. The $w - l$ transformation can, nevertheless, shed light upon the inadequacy of discount to optimally solve the kinds of *win sooner* problems it is supposedly suited for.

Figure 8 shows five million randomly generated terminating Decision Snakes and Ladders policies (out of 2^{72} possible) in the $w - l$ space, together with the convex hull they induce. Observe that many policies have 0 winning probability and, in general, the fastest policies have poor execution. Figure 9 shows the bottom of the convex hull of the random policies, the convex hull including the discounted optimal and the tradeoff better policy and the $w - l$ transformation of the discount-optimal policies from Figure 6.

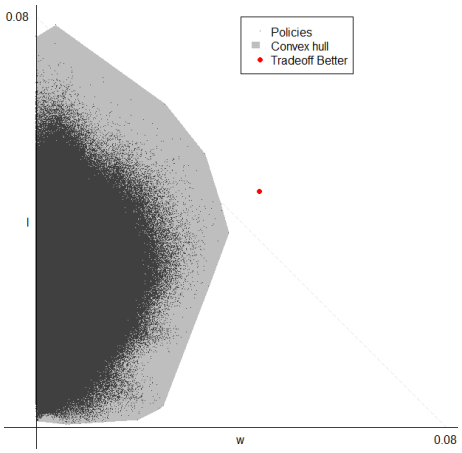

Fig. 8. Five million random Decision Snakes and Ladders policies in the $w - l$ space.

As the rewards are increasingly discounted, reducing γ, the resulting optimal policies initially *move forward* (have larger w component) up to a point, and then begin to recede before stabilizing in an optimal policy that has infinite mean episode length, located at the origin in the $w - l$ space. This is the same behaviour already shown in Figure 6, but with an added noteworthy perspective: observe that the red point on the discount-optimal policies trajectory, optimal for discount values $0.89 < \gamma < 0.90$, *must* lie inside the convex hull of the (complete) policy cloud. This shows that even if, as in this case, some discount might solve some tradeoff problems, as long as the optimal policies remain vertices of the convex hull, there are absolutely no guarantees that all discount-optimal policies will solve *some* tradeoff problems. Lying inside the cloud, the discounted optimal policy corresponding to the red point fails to optimize any tradeoff. It is, thus, little

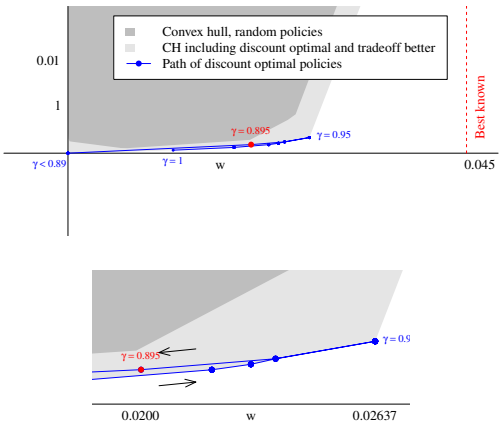

Fig. 9. Top: Discounted optimal and convex hull of random policies in the $w - l$ space. Bottom: detail.

less than a leap of faith to expect discount, despite some improvement for γ close to 1, to solve tradeoff problems of the form (7)–(8) to optimality. Furthermore, the discount factor remains a parameter that, even to obtain suboptimal tradeoff performance, needs to be set somehow.

The inability of discounting rewards to optimally solve tradeoff problems can be best explained formally with a simple example. Consider a MDP game with only two intermediate states, s_0 and s_1, and single "win" and "loss" terminal states, s_w and s_l, respectively. Assume there are only two available policies, shown in Figure 10:

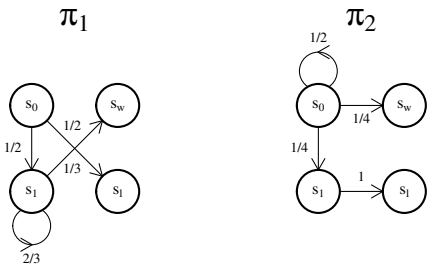

Fig. 10. A game with two policies.

It is easy to compute the features of π_1 and π_2. Both policies clearly have a winning probability of 0.5 and both can be shown to have a mean episode length of 2.5 steps. Therefore, both policies have the same undiscounted initial-state value ($v_1 = v_2 = 0$), correspond to the same point in the $w - l$ space, $(\frac{1}{5}, \frac{1}{5})$, and have equal fitness in any tradeoff problem. Nevertheless, observe that if rewards are discounted by a factor γ, initial-state values become:

$$v_1 = -\frac{3(1-\gamma)}{2(3-2\gamma)},$$
$$v_2 = \frac{(1-\gamma)}{2(2-\gamma)}.$$

Figure 11 plots policy values as functions of γ. Observe that, since all wins from s_0 under π_1 are "delayed" at least one step to s_1 plus some number of self-transitions there, whereas when it loses, it does so in just one step, the negative reinforcement associated with losing is less discounted than the positive reinforcement on winning, so discounted v_1 is *always negative*. Likewise, for π_2, since losses take one step more than wins after leaving s_0, the positive reward is always less discounted than the negative, and discounted v_2 is *always positive*.

Thus, two policies that in the $w-l$ space are indistinguishable, and therefore equally fit to solve any tradeoff problem of the family (7), for any $\gamma < 1$ have not only different initial-state values, but of opposite signs. This shows that there is a fundamental disconnection between discounted value and balancing policy execution and speed. Formally, the example proves that there is no functional dependency between tradeoff and discounted value so, in fact, discount comes with absolutely no guarantee of improving tradeoff, and, if at all, it does improve at the cost of making the analysis more complex and requiring fine tuning to only achieve suboptimal behaviour.

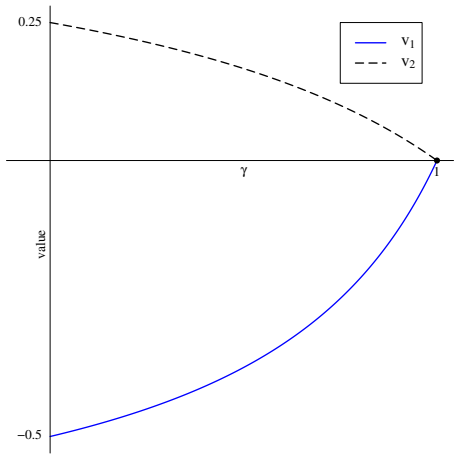

Fig. 11. Initial state value as a function of the discount factor.

V. Conclusions

It has been often recognized that there is a difference between seeking to maximize the probability of winning—the usual goal in undiscounted ± 1-reinforced games—and searching policies that balance winning as much as possible while doing it as quickly as possible. In the first case, we maintain that, unless long loops are explicitly punished (as "threefold repetition" rules do in Chess, for example), undiscounted-optimal policies will show the same three behaviours described for the optimal policy in Decision Snakes and Ladders: seeking "win" states, avoiding "loss" states, and, specially, to avoid risk, circling emergent loops of nonterminal states until a—likely rare—opportunity arises to *move forward* to the goal.

Regarding the execution/speed balance, the example shows that properly tuned discount does indeed improve performance, reducing the mean episode length from ~ 84 to ~ 35 steps while only losing about 5% more games. However, an even better policy was presented, remarkable because it is never visited in the path of optimal policies as the discount factor *gamma* varies.

We argue that discounted-optimal policies fail to solve tradeoff problems because of the inherent bias they show on giving preference to not only *winning fast*, but to the combination of *quick-winning* (little discount of positive rewards) and *slow-losing* (making negative reinforcements insignificant through discount). This eventually results in optimal policies of infinite episode length, non-losing by definition, but with tradeoff 0.

The $w - l$ space was introduced as the natural arena in which to analyse tradeoff problems, given that a whole class of execution (or payout) / duration balancing tasks are linear in it. Additionally, under that new framework, the inability of discount to solve that kind of problems was further exposed by showing the fundamental schism between discounted value and tradeoff, as policies of equal fitness, corresponding to the same point in the $w-l$ space, can have widely different values, and also showing that discounted optimal policies can lie inside the convex hull of the policy space, and thus fail to solve (therein linear) tradeoff problems optimally.

Finally, it was suggested that the "tradeoff-better" policy of Figure 5 optimally solves the natural tradeoff problem for the instance of Decision Snakes and Ladders used as example. A future paper will present the required mathematical framework required to propose a rewards model with linear level sets in the $w - l$ space, together with an algorithm that shows when a policy solves a given tradeoff problem and that tunes the rewards to find such optimal policies.

References

[1] H. J. van den Herik, J. W. H. M. Uiterwijk, and J. van Rijswijck, "Games solved: Now and in the future," *Artificial Intelligence*, vol. 134, no. 1-2, pp. 277 – 311, 2002.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA: MIT press, 1998.

[3] J. G. Kemeny and J. L. Snell, *Finite Markov chains*, 2nd ed. New York: Springer-Verlag, 1976.

[4] E. Beaudry, F. Bisson, S. Chamberland, and F. Kabanza, "Using markov decision theory to provide a fair challenge in a roll-and-move board game," in *IEEE Computational Intelligence and Games (CIG)*, 2010, pp. 1–8.

[5] K. Främling, "Reinforcement learning in a noisy environment: light-seeking robot," *WSEAS Trans. Syst.*, vol. 3, no. 2, pp. 714–719, 2004.

[6] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *ICML '94: Proceedings of the Eleventh International Conference on Machine Learning*. San Francisco: Morgan Kaufmann, 1994, pp. 157–163.

[7] G. Hollinger, S. Singh, J. Djugash, and A. Kehagias, "Efficient multi-robot search for a moving target," *INT J ROBOT RES*, vol. 28, no. 2, pp. 201–219, 2009.

[8] R. McCallum, "Hidden state and reinforcement learning with instance-based state identification," *IEEE T SYST MAN CYB*, vol. 26, no. 3, pp. 464–473, 2002.

[9] H. Kimura and S. Kobayashi, "An analysis of actor-critic algorithms using eligibility traces: Reinforcement learning with imperfect value functions," *Journal of JSAI*, vol. 15, no. 2, pp. 267–275, 2000.

[10] C. Chew and G. Pratt, "Dynamic bipedal walking assisted by learning," *ROBOTICA*, vol. 20, no. 05, pp. 477–491, 2002.

[11] H. Cuayáhuitl, S. Renals, O. Lemon, and H. Shimodaira, "Reinforcement learning of dialogue strategies with hierarchical abstract machines," in *IEEE Spoken Language Technology Workshop*, 2006, pp. 182 –185.

[12] L. Matignon, G. Laurent, and N. Le Fort-Piat, "Hysteretic q-learning: An algorithm for decentralized reinforcement learning in cooperative multi-agent teams," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 64 –69.

[13] M. Ghavamzadeh and S. Mahadevan, "Hierarchically optimal average reward reinforcement learning," in *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann, 2002.

[14] V. Vassiliades, A. Cleanthous, and C. Christodoulou, "Multiagent reinforcement learning with spiking and non-spiking agents in the iterated prisoner's dilemma," *LECT NOTES COMPUT SC*, vol. 5768, pp. 737–746, 2009.

[15] Ö. Şimşek and A. G. Barto, "Using relative novelty to identify useful temporal abstractions in reinforcement learning," in *ICML '04: Proceedings of the Twenty-First International Conference on Machine Learning*. ACM Press, 2004, pp. 751–758.

[16] Y.-H. Chang, T. Ho, and L. P. Kaelbling, "All learning is local: Multi-agent learning in global reward games," *ADV NEUR IN*, vol. 16, 2004.