This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.0000

A Unifying View of Estimation and Control Using Belief Propagation with Application to Path Planning

FRANCESCO A. N. PALMIERI[1] (Senior Member, IEEE), KRISHNA R. PATTIPATI[2] (Fellow, IEEE), GIOVANNI DI GENNARO[1], GIOVANNI FIORETTI[1], FRANCESCO VEROLLA[1], AND AMEDEO BUONANNO[3] (Senior Member, IEEE)

[1]Dipartimento di Ingegneria, Università degli Studi della Campania "Luigi Vanvitelli", Aversa (CE), Italy
(emails: {francesco.palmieri, giovanni.digennaro}@unicampania.it, {giovanni.fioretti, francesco.verolla}@studenti.unicampania.it)
[2]Department of Electrical and Computer Engineering, University of Connecticut, Storrs (CT), USA (email:krishna.pattipati@uconn.edu)
[3]ENEA, Department of Energy Technologies and Renewable Energy Sources, Portici (NA), Italy (email:amedeo.buonanno@enea.it)

Corresponding author: Francesco A. N. Palmieri (e-mail: francesco.palmieri@unicampania.it).

This work was supported in part by POR CAMPANIA FESR 2014/2020, ITS for Logistics, awarded to CNIT (Consorzio Nazionale Interuniversitario per le Telecomunicazioni). Research of Pattipati was supported in part by the U.S. Office of Naval Research and US Naval Research Laboratory under Grants #N00014-18-1-1238, #N00173-16-1-G905 and #HPCM034125HQU, and by a Space Technology Research Institutes grant (number 80NSSC19K1076) from NASAs Space Technology Research Grants Program.

ABSTRACT The use of estimation techniques on stochastic models to solve control problems is an emerging paradigm that falls under the rubric of Active Inference (AI) and Control as Inference (CAI). In this work, we use probability propagation on factor graphs to show that various algorithms proposed in the literature can be seen as specific composition rules in a factor graph. We show how this unified approach, presented both in probability space and in log of the probability space, provides a very general framework that includes the Sum-product, the Max-product, Dynamic programming and mixed Reward/Entropy criteria-based algorithms. The framework also expands algorithmic design options that lead to new smoother or sharper policy distributions. We propose original recursions such as: a generalized Sum/Max-product algorithm, a Smooth Dynamic programming algorithm and a modified versions of the Reward/Entropy algorithm. The discussion is carried over with reference to a path planning problem where the recursions that arise from various cost functions, although they may appear similar in scope, bear noticeable differences. We provide a comprehensive table of composition rules and a comparison through simulations, first on a synthetic small grid with a single goal with obstacles, and then on a grid extrapolated from a real-world scene with multiple goals and a semantic map.

INDEX TERMS Belief Propagation, Dynamic Programming, Markov Decision Process, Path Planning, Reinforcement Learning

I. INTRODUCTION

THERE is a growing interest in establishing connections between probabilistic estimation methods and more traditional stochastic control strategies [1]–[3]. Analogies between control and estimation can be traced back to the work of Kalman [4] and to more recent attempts to link probabilities and rewards under the same framework [3], [5]. The terms Active Inference (AI) and Control as Inference (CAI) have been recently coined [6] with some of these models based on the so-called free-energy principle [7], [8], on KL-learning [9]–[11], and on Maximum entropy [12]. Based on these proposals intriguing connections have also been drawn, to neuroscience and brain theory [13] and causal reasoning [14]; they all seem to share some elements with a goal-directed behavior.

The estimation/control framework on which we focus our attention here, may provide a unifying view for a wide range of scenarios whenever one has a stochastic model and the goal is to achieve a set of objectives by optimizing a cost function defined in probabilistic terms. In this paper, to better visualize the differences and the commonalities among the various methods, we have taken a classic path planning problem as a typical use case: an agent needs to navigate in a complex and uncertain scenario basing its actions on its best

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

inference of the environment to reach a goal subject to spatial constraints.

It is largely agreed that intelligent planning, involves an agent taking a sequence of actions on the basis of its best estimate of the future. This is clearly the hallmark of Dynamic Programming (DP) algorithms for Markov Decision Processes (MDP) and Partially-Observable Markov Decision Processes (POMDP) [15], [16]. Indeed, in DP, an agent acts optimally using a value function back-propagated from the hypothetical future. Similarly, however, in an estimation context, a best-path search can be seen as a Maximum A Posteriori (MAP) solution in a stochastic dynamic model where the start (initial) state and goal (end) state are constrained [17]. This suggests that the two approaches can be viewed under a unified framework and that some of the powerful estimation techniques based on probability propagation on graphs, may provide an ideal way for combining the two disparate approaches and for leading to new generalizations. In this paper, we review the most popular algorithms as they translate into messages in probability and in log-probability spaces. We show how the unifying framework allows us also to derive original parametric extensions that provide the designer with a whole suite of new algorithmic options.

A. THE PROBABILITY GRAPHS

In this work, we use directed Factor Graphs (FG), that assign variables to edges and factors to interconnected blocks. Message propagation in FG is more easily handled in comparison to propagation in graphs in which the variables are in the nodes [18]. In an FG messages propagate through a block diagram, where each block's function is defined independently. Further reductions on the burden of defining message composition rules can be achieved using Factor Graphs in normal form (FGn), first proposed by Forney [19] [20]. In fact, a FGn conveniently includes *junction nodes* (equality constraint nodes) that split incoming and outgoing messages when variables are shared by multiple factors. We have proposed a small modification to the FGn in our *Factor Graph in Reduced normal form (FGrn)* [21] by including *shaded blocks* that map single variables to joint spaces. In an FGn, when a variable has more than one parent, proper forward and backward messages must go through the parents' joint space (married parents). In a FGrn instead, the shaded blocks describe this passage and allow a unique definition of message propagation rules through *Single-Input/Single-Output (SISO)* blocks. Since, in the standard sum-product algorithm, backward propagation through shaded blocks corresponds to marginalization, we show in this paper how this operation can be re-defined and how it may be mapped to different estimation/control algorithms. Computational complexity issues for some FGrn architectures are addressed in [22]. We confine ourselves here to discrete variables, even if factor graphs that propagate continuous distributions are possible and may be devised also for path planning. Gaussian messages are introduced in [23] and have been used for Kalman filter-based tracking in [24] using FGrn. This issue will not be addressed here and will be the subject of a future work.

B. THE PATH MODELING PROBLEM

We have proposed in some of our previous works various techniques for modeling the motion behaviors of pedestrians and ships [25]–[29]. More recently, while experimenting with probability propagation in path planning problms [30], we came to realize that the probabilistic algorithms may be the most promising approaches for agile modeling of intelligent agent motion in complex scenes. This led to the development of the unified belief propagation framework for estimation and control discussed in this paper.

We assume here that the system's stochastic transition function is known and that both the state and the action spaces are discrete finite sets that can be handled with tabular methods. Extensions to continuous spaces can be considered with approximations to the value function, but they will not be addressed here. Also, we do not address learning here, because we believe that a unified view on the various cost functions and recursions with known stochastic system dynamics should be the first step in trying to understand the more challenging Reinforcement Learning (RL) [31] adaptation rules. In this paper, standard probability message propagation, such as the Sum-product and the Max-product algorithms [32], are compared to DP using a unified view, together with other methods based on joint Reward/Entropy maximization [3], [6], [33]–[35]. To our knowledge, no comprehensive comparison exists in the literature, and our contribution aims at providing the reader with a ready-to-use suite of known algorithms and their original extensions, all derived withing the unifying framework presented here.

C. OUR CONTRIBUTIONS

The main contributions of this paper can be summarized as follows:

- The path planning problem is mapped to a Factor Graph in Reduced Normal Form. Various algorithms, such as the Sum-product, the Max-product, DP and Reward/Entropy maximization (the latter is related to structural variational inference), are included in our framework, both in probability and in log spaces. We show that all these algorithms are derived using different cost functions, but they all correspond to specific propagation rules through some of the FGrn blocks.
- Q-functions and V-functions are generalized in the log-probability space for all the algorithms. This formulation includes the well-known Q- and V-functions arising in DP and allows us to specify the policy distribution resulting from the algorithms with a unique expression.
- Some of the well-known algorithms are extended to a whole new suite of parametric updates that can control the smoothness in the policy distributions. These proposed parametric updates are original and can be used as hyper-paraameters to balance exploration and exploitation in reinforcement learning.

- Simulations are provided, first on a small grid with one goal and a set of obstacles, and then on a larger grid extracted from a real scene with multiple goals (exits) and a semantic map. The results show marked differences in : (a) the speed of converge to the steady-state value function, where probabilistic methods are clearly favored; (b) how the Max-product algorithm may be preferred for its faster convergence and for the shape and smoothness of its value functions; and (c) how various algorithms can be controlled with parametric updates to exhibit varying levels of smoothness in their policy distributions.

D. OUTLINE OF THE PAPER

In Section II, we present the Bayesian graph model and the concomitant factor graph. In Section III, the Sum-product algorithm is discussed in the framework of FGrn. In Section IV, the maximum a posteriori solution of the Max-product algorithm is analyzed with our proposed Sum/Max-product algorithm described in Section V. Dynamic programming is translated into this framework in Section VI and our proposal for a generalized SoftDP is in Section VII. The approaches to combined maximum reward and entropy are discussed in Section VIII. Simplifications of some of the recursions for deterministic systems, are discussed in Section IX. The extension to infinite horizon models and considerations on the steady-state solutions are included in Section X. Simulations on small and realistic grids are in Section XI, with conclusions and suggestions for further research in Section XII.

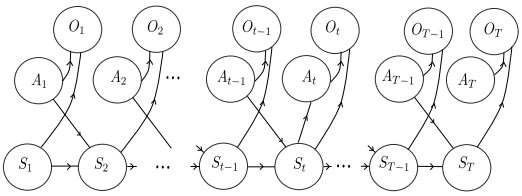


FIGURE 1. State-Action Model as a Bayesian graph

II. THE BAYESIAN MODEL

Figure 1 shows the state-action model as a Bayesian graph where $\{S_t\}$ is the *state* sequence, $\{A_t\}$ is the *action* sequence. We assume, without loss of generality, that both sequences belong to discrete finite sets: $A_t \in \mathcal{A}$ and $S_t \in \mathcal{S}$. The reward/outcome sequence $\{O_t\}$ is binary with $O_t \in \{0, 1\}$. The model evolves over a finite horizon T and the joint probability distribution of the state-action-outcome sequence corresponds to the factorization[1]

$$p(s_1 a_1 o_1 \dots s_T a_T o_T) = p(o_T|s_T a_T)p(s_1)p(a_T) \prod_{t=1}^{T-1} p(s_{t+1}|s_t a_t)p(a_t)p(o_t|s_t a_t),$$

where the function $p(s_{t+1}|s_t a_t)$ describes the *system dynamics*, $p(a_t)$ are the action priors and $p(o_t|s_t a_t)$ are the *reward/priors* of outcomes on the state-action pairs. More specifically, we assume that

$$\begin{cases} P(O_t = 1|s_t a_t) \propto \pi(s_t a_t) \geq 0; \\ P(O_t = 0|s_t a_t) \propto U(s_t a_t), \end{cases}$$

where the function $\pi(s_t a_t)$ serves as a prior distribution on the pair $(s_t a_t)$, only if $O_t = 1$. When $O_t = 0$, no prior information is available on that state-action pair, and the factor becomes the uniform distribution $U(s_t a_t)$.[2] Therefore, to simplify notations, we define a factor $c(s_t a_t) = \pi(s_t a_t)$ if prior information is available; otherwise we set $c(s_t a_t) = U(s_t a_t)$. This formulation allows the introduction of a *reward* function as

$$R(s_t a_t) = \log c(s_t a_t) + K, \tag{1}$$

where K is an arbitrary positive constant. The value K is really irrelevant because going back to probabilities we have $c(s_t a_t) \propto e^{R(s_t a_t) - K}$, with the constant disappearing after normalization. We can set K to a large value if we do not like to handle negative rewards we obtain from the log function for $K = 0$. In the following, without loss of generality, we assume that our rewards are all negative ($K = 0$).

The introduction of the sequence $\{O_t\}$ has been proposed earlier [3], [36] for connecting rewards to probabilities. We would like to emphasize that interpreting the factors $c(s_t a_t)$ as prior information in the probability factorization, may solve, at least for planning problems, the well-known issue of defining an appropriate reward function. Indeed, in a practical problem, we may have available statistics on how often a state is visited and how certain actions may be more likely than, or preferable to, others.

Note that when a state-action pair has zero probability, for example for forbidden states, or impossible actions, obstacles, etc., the reward function takes a value of $-\infty$. This is really not a problem in practice, because we can easily approximate such a value with a large negative number.

Note that our model includes a separate factor $p(a_t)$ for the priors on A_t, even if such information could be included in $c(s_t a_t)$. We have preferred, to be consistent with the Bayesian graph of Figure 1, to keep the two factors separate, one for marginal action priors and one for joint priors (rewards).

[1] Even if the notation should have capital letters for random variables as subscripts and lower case letters for their values in the functions, we use a compact notation with no subscripts when there is no ambiguity. We will include the subscripts for the messages only when necessary.

[2] In our definition, we assume that $\pi(s_t a_t)$ is normalized to be a valid pdf, even if normalization is irrelevant for the inference in a probabilistic graph.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

Omitting the sequence $\{O_t\}$ in the notation the factorization is more compactly written as

$$p(s_1a_1...s_Ta_T) = c(s_Ta_T)p(s_1)p(a_T) \prod_{t=1}^{T-1} p(s_{t+1}|s_ta_t)p(a_t)c(s_ta_t). \quad (2)$$

.

A. THE FACTOR GRAPH

Inference in a graphical model is easily handled with reference to an equivalent factor graph. Much like in a block diagram, variables are on the branches and factors are in the blocks.

We use here *Factor Graphs in Reduced normal form (FGrn)* [21], for wich the essential propagation rule are in Tables 3, 4, 5, 6 and 7.

Figure 2 shows the FGrn for the Bayesian model of Figure 1 for $T = 4$. The prior distributions $p(a_t)$ and $c(s_ta_t)$ are in the source nodes and the dynamics $p(s_{t+1}|s_ta_t)$ are in the SISO blocks. The junctions describe equality constraints, and the shaded blocks describe the mapping from single variables' space to a joint space, i.e., $p(s_ta_t|a'_t) = U(s_t)\delta(a_t - a'_t)$ and $p(s_ta_t|s'_t) = \delta(s_t - s'_t)U(a_t)$. Essentially, in a shaded block, the input variable is copied to the output and joined to the other variable that, in that edge, carries no information in the forward direction. Each edge has a direction and a *forward* f and a *backward* b message associated with it. In the following, each message has a subscript corresponding to the variable(s) it describes and an argument which is(are) the value(s) assumed by that(those) variable(s). Just as in any belief propagation network, all messages are proportional to probability distributions and their composition rules allow the agile derivation of inference algorithms. Note how the replicas $(S_tA_t)^i$, $i = 1 : 4$, of the same variable (S_tA_t) around the diverter block have different names and messages associated with them.

We will see how our estimation/control problem has a unique formulation on the factor graph. By changing the propagation rules for some of the blocks, we obtain the optimal solutions for various problem formulations.

B. INTRODUCING CONSTRAINTS

One of the main advantages of studying inference problems on graphs using messages, is that problem constraints are easily included in the flow. For example, looking at Figure 2:

- A known starting state $S_1 = \overline{s}_1$ can be included as a forward message $f_{S_1}(s_1) = \delta(s_1 - \overline{s}_1)$, where $\delta(x) = 1$ if $x = 0$, and is 0 otherwise;
- If we have no prior information on S_1, we set $f_{S_1}(s_1) = U(s_1)$;
- Knowledge of the initial action $A_1 = \overline{a}_1$ can be included as $f_{A_1}(a_1) = \delta(a_1 - \overline{a}_1)$;
- Knowledge of the final state (only) $S_T = \overline{s}_T$ is $b_{(S_TA_T)^4}(s_Ta_T) = \delta(s_T - \overline{s}_T)U(a_T)$;
- Knowledge of the state at time t_0 may be included as $f_{S_{t_0}}(s_{t_0}) = \delta(s_{t_0} - \overline{s}_{t_0})$;
- In a planning problem, a known map $m(s_t)$ can be associated to the factor $c(s_ta_t)$ with $f_{(S_tA_t)^3}(s_ta_t) \propto m(s_t)U(a_t)$;
- In the same planning problem, joint map-action information can injected as the message $f_{(S_tA_t)^3}(s_ta_t) \propto c(s_ta_t)$; if action and map are independent, and the action prior is $p(a_t)$, $f_{(S_tA_t)^3}(s_ta_t) \propto m(s_t)p(a_t)$, or equivalently $f_{(S_tA_t)^3}(s_ta_t) \propto m(s_t)U(a_t)$ and $f_{A_t}(a_t) = p(a_t)$; etc.

We denote collectively all the constraints available on the joint model as $K_{1:T}$, with the joint model written in compact form as $p(s_1a_1 \dots s_Ta_T|K_{1:T})$. Note that, in the above formulation, we have assumed a finite time segment $t = 1 : T$, but the model may as well represent a segment $t = t_0 + 1 : t_0 + T$, or one of many segments, of a longer process, where we have the freedom to introduce initial conditions in the forward messages at each starting time, and final conditions in the backward messages at the termination.

C. INFERENCE OBJECTIVES

Our inference aims at providing solutions to one or more of the following problems:

- Find the best *state* sequence (S): $s_1^* \to s_2^* \to \dots \to s_T^*$
- Find the best *action* sequence (A): $a_1^* \to a_2^* \to \dots \to a_T^*$
- Find the best *joint state-action* sequence (SA): $(s_1a_1)^* \to (s_2a_2)^* \to ... \to (s_Ta_T)^*$
- Find the best *state-action* sequence (SASA): $s_1^* \to a_1^* \to s_2^* \to a_2^* \to \dots \to s_T^* \to a_T^*$
- Find the best *action-state* sequence (ASAS): $a_1^* \to s_1^* \to a_2^* \to s_2^* \to \dots \to a_T^* \to s_T^*$
- Find the best *policy* distributions (P): $\pi^*(a_1|s_1), \pi^*(a_2|s_2) \dots \pi^*(a_T|s_T)$

We will see in the following how various cost functions determine the message composition rules across the blocks to solve the problems above, both in the probability space and in the log probability space depending on different optimization criteria. In the following discussion, we will concentrate mostly on the S sequence, on the SASA sequence and on the policy distribution (P), since the extensions to A, SA, and ASAS sequences are straightforward.

III. MARGINALIZATION AND THE SUM-PRODUCT

Standard inference in Bayesian model consists in marginalizing the global joint probability distribution to obtain distributions that are proportional to the posteriors on single variables [18]. More specifically, for states only, for actions only and for states and actions jointly, we want to compute

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

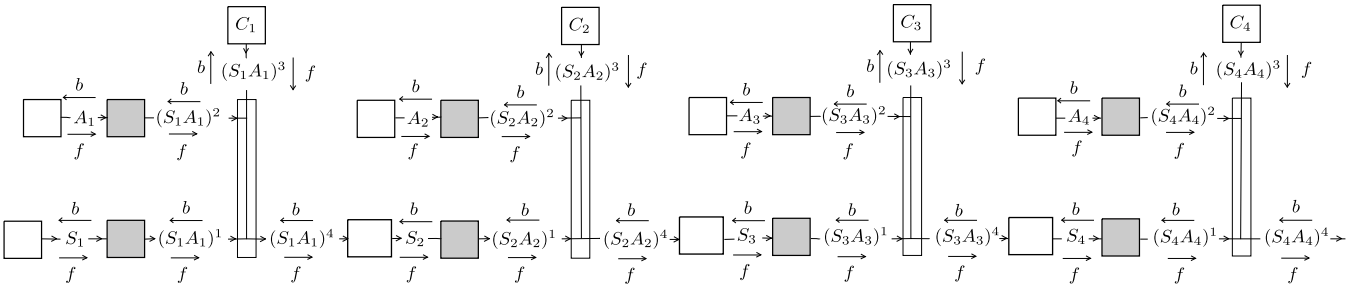


FIGURE 2. State-Action Model for $T = 4$ as a Factor Graph in Reduced normal form. The one-edge blocks are sources (priors); the two-edge white blocks represent the system dynamics; the shaded blocks map single variables to their joint space; the diverters connect the variables constrained to be equal.

the posteriors as

$$p(s_t|K_{1:T}) \propto \sum_{\substack{s_j,j\neq t,j=1:T \\ a_j,j=1:T}} p(s_1a_1 \dots s_Ta_T|K_{1:T}),$$
$$p(a_t|K_{1:T}) \propto \sum_{\substack{s_j,j=1:T \\ a_j,j\neq t,j=1:T}} p(s_1a_1 \dots s_Ta_T|K_{1:T}),$$
$$p(s_ta_t|K_{1:T}) \propto \sum_{s_j,a_j,j\neq t,j=1:T} p(s_1a_1 \dots s_Ta_T|K_{1:T}),$$

where $K_{1:T}$ is the information base on which action decisions are made. The *policy* distributions are obtained by fixing the state s_t at time t, and accounting for the foreseeable future until T

$$\pi^*(a_t|s_t) \triangleq p(a_t|s_t, K_{t:T}) = \frac{p(s_ta_t|K_{t:T})}{p(s_t|K_{t:T})}, \quad t = 1:T. \tag{3}$$

The policy distribution describes at time t how likely it is to take action a_t from state s_t, given all the available information (constraints, priors, etc.) about the future ($K_{t:T}$).

All the above functions can be obtained using forward and backward message propagation using the *Sum-product* rule [18], [32]. This approach essentially averages over the variables that are eliminated across each SISO block. With reference to Figure 2, by following the flow, for some of the backward messages we have

$$b_{(S_tA_t)^4}(s_ta_t) \propto \sum_{s_{t+1}} p(s_{t+1}|s_ta_t) b_{S_{t+1}}(s_{t+1}), \tag{4a}$$
$$b_{(S_tA_t)^1}(s_ta_t) \propto p(a_t) c(s_ta_t) b_{(S_tA_t)^4}(s_ta_t), \tag{4b}$$
$$b_{(S_tA_t)^2}(s_ta_t) \propto f_{(S_tA_t)^1}(s_ta_t) c(s_ta_t) b_{(S_tA_t)^4}(s_ta_t), \tag{4c}$$
$$b_{A_t}(a_t) = \sum_{s_t} b_{(S_tA_t)^2}(s_ta_t), \tag{4d}$$
$$b_{S_t}(s_t) = \sum_{a_t} b_{(S_tA_t)^1}(s_ta_t). \tag{4e}$$

For some of the forward messages

$$f_{S_{t+1}}(s_{t+1}) \propto \sum_{s_ta_t} p(s_{t+1}|s_ta_t) f_{(S_tA_t)^4}(s_ta_t),$$
$$f_{(S_tA_t)^1}(s_ta_t) = f_{S_t}(s_t)U(a_t),$$
$$f_{(S_tA_t)^2}(s_ta_t) = U(s_t)f_{A_t}(a_t),$$
$$f_{(S_tA_t)^4}(s_ta_t) \propto f_{(S_tA_t)^1}(s_ta_t) f_{(S_tA_t)^2}(s_ta_t) c(s_ta_t).$$

Note that going backward through a block, the message may not be normalized. Around the diverters, outgoing messages are the product of the incoming ones, and are not normalized. Posterior distributions are obtained by taking the product of forward and backward messages:

$$p(s_t|K_{1:T}) \propto f_{S_t}(s_t) b_{S_t}(s_t),$$
$$p(a_t|K_{1:T}) \propto f_{A_t}(a_t) b_{A_t}(a_t),$$
$$p(s_ta_t|K_{1:T}) \propto f_{(S_tA_t)^i}(s_ta_t) b_{(S_tA_t)^i}(s_ta_t), \quad i = 1:4.$$

For readers not too familiar with probability message propagation, it should be emphasized that this framework is a rigorous application of Bayes' theorem and marginalization. Also all messages can be normalized to be valid distributions, even if it is not strictly necessary (it is their shape that matters). However, it is often advised to keep messages normalized for numerical stability.

The policy distribution (3) at each t is derived as a consequence of the inference obtained from the probability flow as

$$\begin{aligned}\pi^*(a_t|s_t) &\propto \frac{f_{(S_tA_t)^1}(s_ta_t) b_{(S_tA_t)^1}(s_ta_t)}{f_{S_t}(s_t) b_{S_t}(s_t)} \\ &= \frac{f_{S_t}(s_t)U(a_t) b_{(S_tA_t)^1}(s_ta_t)}{f_{S_t}(s_t) b_{S_t}(s_t)} \\ &= \frac{b_{(S_tA_t)^1}(s_ta_t)}{b_{S_t}(s_t)},\end{aligned} \tag{5}$$

where we have used the edge with $i = 1$. It is easy to verify that the solution would have an equivalent expression for any other edges $i = 2, 3, 4$. Note also how the policy depends only on the backward messages. The reason for this is that by conditioning on s_t, all the information coming from the left side of the graph is blocked (i.e. is irrelevant).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

A. MAX POSTERIOR SEQUENCES

Optimal sequence values, for any $t = 1 : T$, can be obtained in parallel using maximization on the posteriors

$$\begin{aligned} s_t^* &= \operatorname*{argmax}_{s_t} p(s_t|K_{1:T}) = \operatorname*{argmax}_{s_t} f_{S_t}(s_t) b_{S_t}(s_t), \\ a_t^* &= \operatorname*{argmax}_{a_t} p(a_t|K_{1:T}) = \operatorname*{argmax}_{a_t} f_{A_t}(a_t) b_{A_t}(a_t), \\ (s_t a_t)^* &= \operatorname*{argmax}_{s_t a_t} p(s_t a_t|K_{1:T}) \\ &= \operatorname*{argmax}_{s_t a_t} f_{(S_t A_t)^1}(s_t a_t) b_{(S_t A_t)^1}(s_t a_t), \end{aligned}$$

In the above expression, the max posterior solutions are taken separately on each variable. Even if they are often used in the applications (for example in decoding convolutional codes - the algorithm, is named BCJR after its authors [37]), they may provide unsatisfactory sequences for path planning. In fact, the sequences that result from the individual (local) maximizations are unconstrained in time and may correspond to disconnected paths [30].

B. PROGRESSIVE MAX POSTERIOR SEQUENCES

Better solutions for the max posterior approach are obtained progressively in time following a forward procedure.[3] Looking at Figure 2, for the states-only (S) sequence

$$\begin{aligned} s_t^* &= \operatorname*{argmax}_{s_t} p(s_1^* ... s_{t-1}^* s_t | K_{t:T}) \\ &= \operatorname*{argmax}_{s_t} f_{S_t}(s_t|s_{t-1}^*) b_{S_{t-1}}(s_{t-1}), \end{aligned}$$

where the conditioned forward message comes from a one-step propagation

$$\begin{aligned} f_{S_t}(s_t|s_{t-1}^*) &= \sum_{a_{t-1}} p(s_t|s_{t-1}^* a_{t-1}) f_{(S_{t-1}A_{t-1})^4}(s_{t-1}^* a_{t-1}) \\ &= \sum_{a_{t-1}} p(s_t|s_{t-1}^* a_{t-1}) p(a_{t-1}) c(s_{t-1}^* a_{t-1}). \end{aligned}$$

Note again on the graph that knowledge of the state at time $t-1$ "breaks" the forward flow and only the backward flow drives the inference. Similarly, for the best State-Action (SASA) sequence, the *Progressive Max-posterior* algorithm using the messages on the graph in Figure 2 is

$$\begin{aligned} s_t^* &= \operatorname*{argmax}_{s_t} p(s_1^* a_1^* ... s_{t-1}^* a_{t-1}^* s_t | K_{t:T}) \\ &= \operatorname*{argmax}_{s_t} f_{S_t}(s_t|s_{t-1}^* a_{t-1}^*) b_{S_t}(s_t), \\ a_t^* &= \operatorname*{argmax}_{a_t} p(s_1^* a_1^* ... s_{t-1}^* a_{t-1}^* s_t^* a_t | K_{t:T}) \\ &= \operatorname*{argmax}_{a_t} f_{A_t}(a_t) b_{A_t}(a_t|s_t^*), \end{aligned}$$

where the conditioned forward and backward messages mean that we have considered their values when the conditioning variables on the left side of the graph are fixed. For the conditioned forward messages we have

$$f_{S_t}(s_t|s_{t-1}^* a_{t-1}^*) = p(s_t|s_{t-1}^* a_{t-1}^*).$$

[3]On a fixed time horizon, a similar procedure can be derived going backward in time. We prefer to maintain the framework causal and leave it out for brevity.

For the conditioned backward messages, we have

$$\begin{aligned} b_{A_t}(a_t|s_t^*) &\propto b_{(S_t A_t)^2}(s_t^* a_t) \\ &\propto c(s_t^* a_t) b_{(S_t A_t)^4}(s_t^* a_t) f_{(S_t A_t)^1}(s_t^* a_t). \end{aligned}$$

Since

$$\begin{aligned} b_{(S_t A_t)^1}(s_t a_t) &\propto b_{(S_t A_t)^4}(s_t a_t) f_{(S_t A_t)^2}(s_t a_t) c(s_t a_t) \\ &= b_{(S_t A_t)^4}(s_t a_t) p(a_t) U(s_t) c(s_t a_t), \\ b_{(S_t A_t)^4}(s_t a_t) &\propto \frac{b_{(S_t A_t)^1}(s_t a_t)}{p(a_t) c(s_t a_t)}, \end{aligned}$$

the backward messages can be rewritten as

$$\begin{aligned} b_{A_t}(a_t|s_t^*) &\propto \frac{b_{(S_t A_t)^1}(s_t^* a_t)}{p(a_t)} f_{(S_t A_t)^1}(s_t^* a_t) \\ &= \frac{b_{(S_t A_t)^1}(s_t^* a_t)}{p(a_t)} \delta(s_t - s_t^*) U(a_t) \\ &= \frac{b_{(S_t A_t)^1}(s_t^* a_t)}{p(a_t)}. \end{aligned}$$

Therefore, the SASA estimation simplifies to

$$s_t^* = \operatorname*{argmax}_{s_t} p(s_t|s_{t-1}^* a_{t-1}^*) b_{S_t}(s_t), \tag{6a}$$

$$a_t^* = \operatorname*{argmax}_{a_t} b_{(S_t A_t)^1}(s_t^* a_t) = \operatorname*{argmax}_{a_t} \pi^*(a_t|s_t^*). \tag{6b}$$

Note in all cases the crucial role played by the backward flow. We have successfully demonstrated this approach for path planning in our previous work [30]. In fact, in the progressive max posterior algorithm, the forward flow is not necessary. Action-only sequences and ASAS sequence can be obtained in a similar fashion and are omitted here for brevity.

C. SUM-PRODUCT IN THE LOG-SPACE

We have seen above how in the factorized model (2), prior distributions are related to rewards via the log transformation in (1). To provide a direct comparison to the dynamic programming approach, we consider now some of the Sum-product recursions in the log-space. We define the functions

$$\begin{aligned} Q_{(S_t A_t)^i}(s_t a_t) &\triangleq \log b_{(S_t A_t)^i}(s_t a_t), \qquad i = 1:4 \\ V_{S_t}(s_t) &\triangleq \log b_{S_t}(s_t), \end{aligned}$$

Note that there is a Q function for each message around the diverter. The choice of notations Q (*Q-function*) and V (*Value-function*) is intentional, as it leads to a direct comparison with DP. In this formulation, there is also a V-function for the action variables A_t, $V_{A_t}(a_t) \triangleq \log b_{A_t}(a_t)$. From the definition, it is evident that both the Q- and V-functions are negative (we have already pointed out above that this is not a limitation). We concentrate here mostly on the state S_t for which the backward recursions (4a, 4b, 4e), are written in the log-space as

$$Q_{(S_t A_t)^4}(s_t a_t) \propto \log \sum_{s_{t+1}} p(s_{t+1}|s_t a_t) e^{V_{S_{t+1}}(s_{t+1})}, \tag{7}$$

$$Q_{(S_t A_t)^1}(s_t a_t) \propto \log p(a_t) + R(s_t a_t) + Q_{(S_t A_t)^4}(s_t a_t),$$

$$V_{S_t}(s_t) \propto \log \sum_{a_t} e^{Q_{(S_t A_t)^1}(s_t a_t)}.$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

All messages can be propagated in the log-space: the product rule around the diverters of Figure 2 become sums and the backward propagation rules across the dynamics block and the shaded block are simply translated. For comparison with the formulations that follow, we re-write the equation (7) as

$$Q_{(S_tA_t)^4}(s_ta_t) \propto \log \sum_{s_{t+1}} e^{\log p(s_{t+1}|s_ta_t)+V_{S_{t+1}}(s_{t+1})}.$$

The main recursions for the Sum-product are summarized for later comparison in the first row of Tables 1 and 2. The same recursions, and some of the definitions in the log-space, have been reported in [3] that also notes how the transformation $y = \log \sum_{j=1}^{N} e^{x_j}$ is a *soft-max* ($y \sim \max(x_1, ..., x_N)$ when the x_is are large), in contrast to the the *hard-max* that is used in dynamic programming. Properties of this and other soft-max functions that arise in our analyses are included in Appendix A.

The best SASA sequence of equations (6) is equivalently written in the log-space as

$$\begin{aligned} s_t^* &= \operatorname*{argmax}_{s_t} \log p(s_t|s_{t-1}^*a_{t-1}^*) + V_{S_t}(s_t), \\ a_t^* &= \operatorname*{argmax}_{a_t} Q_{(S_tA_t)^1}(a_1, s_1^*), \end{aligned}$$

The policy distribution (5) is rewritten as

$$\pi^*(a_t|s_t) \propto e^{Q_{(S_tA_t)^1}(s_ta_t)-V_{S_t}(s_t)}.$$

IV. MAXIMUM A POSTERIORI AND THE MAX-PRODUCT

The max posterior rules, described above, are used extensively for inference in Bayesian networks, even though they do not necessarily solve the *global* maximum a posteriori problem

$$(s_1^*a_1^* \dots s_T^*a_T^*) = \operatorname*{argmax}_{s_1a_1 \dots s_Ta_T} p(s_1a_1 \dots s_Ta_T|K_{1:T}).$$

The Sum-product propagation rules solve *marginal* maximum a posteriori problems after summing on the eliminated variables, while the global optimization requires a different strategy for obtaining the solution. The *Max-product* algorithm [23], [32], by propagating maximum (or maxima) value messages in the graph, instead of computing averages across the blocks, provides the MAP solution. This is often named bi-directional Viterbi algorithm [32]. The detailed recursions are derived explicitely in Figure 3 for a model with $T = 4$. At a generic step t, the recursions for some of the backward messages are

$$\begin{aligned} b_{(S_tA_t)^4}(s_ta_t) &= \max_{s_{t+1}} p(s_{t+1}|s_ta_t) b_{S_{t+1}}(s_{t+1}), \\ b_{(S_tA_t)^1}(s_ta_t) &= p(a_t)c(s_ta_t)b_{(S_tA_t)^4}(s_ta_t), \\ b_{S_t}(s_t) &= \max_{a_t} b_{(S_tA_t)^1}(s_ta_t). \end{aligned}$$

Again the crucial role is played by the backward flow that, going through each SISO block, does not undergo a summation, but a max (in Max-product bayesian networks also the forward flow is computed using max rather that sum [32]; we focus here mostly on the backward flow). In the log-space, the backward recursions for the states are rewritten as

$$\begin{aligned} Q_{(S_tA_t)^4}(s_ta_t) &= \max_{s_{t+1}} \left[\log p(s_{t+1}|s_ta_t) + V_{S_{t+1}}(s_{t+1})\right], \\ Q_{(S_tA_t)^1}(s_ta_t) &= \log p(a_t) + R(s_ta_t) + Q_{(S_tA_t)^4}(s_ta_t), \\ V_{S_t}(s_t) &= \max_{a_t} Q_{(S_tA_t)^1}(s_ta_t). \end{aligned}$$

The best SASA sequence is computed in the forward direction in way similar to the Sum-product, in both the probability space and in the log-space, as follows

$$\begin{aligned} s_t^* &= \operatorname*{argmax}_{s_t} p(s_t|s_{t-1}^*a_{t-1}^*)b_{S_t}(s_t) \\ &= \operatorname*{argmax}_{s_t} p(s_t|s_{t-1}^*a_{t-1}^*)e^{V_{S_t}(s_t)} \\ &= \operatorname*{argmax}_{s_t} \log p(s_t|s_{t-1}^*a_{t-1}^*) + V_{S_t}(s_t), \\ a_t^* &= \operatorname*{argmax}_{a_t} b_{(S_tA_t)^1}(s_t^*a_t) = \operatorname*{argmax}_{a_t} Q_{(S_tA_t)^1}(s_t^*a_t). \end{aligned}$$

Note how the recursions are formally identical to the ones derived for the Sum-product algorithm, but the rules change across the shaded blocks. Also, the policy has the same formal expression (with a different meaning for Q and V)

$$\pi^*(a_t|s_t) \propto \frac{b_{(S_tA_t)^1}(s_ta_t)}{b_{S_t}(s_t)} = e^{Q_{(S_tA_t)^1}(s_ta_t)-V_{S_t}(s_t)}. \quad (8)$$

All the other sequences, S, A, SA, ASAS can be computed using the probability flow in the graph following the same formal approach, both in the Max-product and in the Sum-product, simply by changing some of the propagation rules. For brevity, we concentrate here only on some of the messages, even if a detailed analysis of other parts of the flow may reveal interesting aspects of the inference. All the propagation rules are reported in Tables 3, 4, 5, 6 and 7, also for the other algorithms described in the following. The main backup recursions are summarized for comparison in Tables 1 and 2 in the probability and in the log-space. We would like to emphasize that propagating information via probability distributions includes all the cases in which there may be deterministic values in the system, i.e., when the distributions are delta functions. Furthermore, in the Max-product algorithm, when multiple equivalent maxima are present, the distributions can carry multiple peaks. We will see that, in some of the simulation examples that follow, the Max-product messages provide a complete set of options in the policy distributions, even when more than one best action is available in a state.

V. THE SUM/MAX-PRODUCT

The unifying view provided by the graphical method is quite appealing and one wonders whether there may be a general rule that encompasses both the Sum-product and the Max-product. By looking at the recursions in the first two rows of Tables 1 and 2, we immediately observe that the Sum-product, both in the probability and in the log-space, can be seen as a *soft version* of the Max-product because of the soft-max functions. Therefore, we propose a general

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

TABLE 1. Summarized backup rules in probability space with $b(s_t a_t) \triangleq b_{(S_t A_t)^1}(s_t a_t)$; $b(s_t) \triangleq b_{S_t}(s_t)$; $c'(s_t a_t) \triangleq p(a_t)c(s_t a_t)$.

	$b(s_t a_t)$	$b(s_t)$
Sum product	$c'(s_t a_t) \sum_{s_{t+1}} p(s_{t+1}\|s_t a_t) b(s_{t+1})$	$\sum_{a_t} b(s_t a_t)$
Max product	$c'(s_t a_t) \max_{s_{t+1}} p(s_{t+1}\|s_t a_t) b(s_{t+1})$	$\max_{a_t} b(s_t a_t)$
Sum/Max product ($\alpha \geq 1$)	$c'(s_t a_t) \sqrt[\alpha]{\sum_{s_{t+1}} p(s_{t+1}\|s_t a_t)^\alpha b(s_{t+1})^\alpha}$	$\sqrt[\alpha]{\sum_{a_t} b(s_t a_t)^\alpha}$
DP	$c'(s_t a_t) e^{\sum_{s_{t+1}} p(s_{t+1}\|s_t a_t) \log b(s_{t+1})}$	$\max_{a_t} b(s_t a_t)$
Max-Rew/Ent ($\alpha > 0$)	$c'(s_t a_t) e^{\sum_{s_{t+1}} p(s_{t+1}\|s_t a_t) \log b(s_{t+1})}$	$\sqrt[\alpha]{\sum_{a_t} b(s_t a_t)^\alpha}$
SoftDP ($\beta > 0$)	$c'(s_t a_t) e^{\sum_{s_{t+1}} p(s_{t+1}\|s_t a_t) \log b(s_{t+1})}$	$e^{\frac{\sum_{a_t} b(s_t a_t)^\beta \log b(s_t a_t)}{\sum_{a'_t} b(s_t a'_t)^\beta}}$

TABLE 2. Summarized backup rules in log space with $Q(s_t a_t) \triangleq Q_{(S_t A_t)^1}(s_t a_t)$; $V(s_t) \triangleq V_{S_t}(s_t)$; $R'(s_t a_t) = \log p(a_t) + R(s_t a_t)$.

	$Q(s_t a_t)$	$V(s_t)$
Sum product	$R'(s_t a_t) + \log \sum_{s_{t+1}} e^{\log p(s_{t+1}\|s_t a_t) + V(s_{t+1})}$	$\log \sum_{a_t} e^{Q(s_t a_t)}$
Max product	$R'(s_t a_t) + \max_{s_{t+1}} (\log p(s_{t+1}\|s_t a_t) + V(s_{t+1}))$	$\max_{a_t} Q(s_t a_t)$
Sum/Max product ($\alpha \geq 1$)	$R'(s_t a_t) + \frac{1}{\alpha} \log \sum_{s_{t+1}} e^{\alpha(\log p(s_{t+1}\|s_t a_t) + V(s_{t+1}))}$	$\frac{1}{\alpha} \log \sum_{a_t} e^{\alpha Q(s_t a_t)}$
DP	$R'(s_t a_t) + \sum_{s_{t+1}} p(s_{t+1}\|s_t a_t) V(s_{t+1})$	$\max_{a_t} Q(s_t, a_t)$
Max-Rew/Ent ($\alpha > 0$)	$R'(s_t a_t) + \sum_{s_{t+1}} p(s_{t+1}\|s_t a_t) V(s_{t+1})$	$\frac{1}{\alpha} \log \sum_{a_t} e^{\alpha Q(s_t a_t)}$
SoftDP ($\beta > 0$)	$R'(s_t a_t) + \sum_{s_{t+1}} p(s_{t+1}\|s_t a_t) V(s_{t+1})$	$\frac{\sum_{a_t} Q(s_t a_t) e^{\beta Q(s_t a_t)}}{\sum_{a'_t} e^{\beta Q(s_t a'_t)}}$

rule that interpolates between the two solutions using a parametrized soft-max functions. We name this generalization the *Sum/Max-product algorithm*, that in the log-space gives

$$Q_{(S_t A_t)^4}(s_t a_t) = \frac{1}{\alpha} \log \sum_{s_{t+1}} e^{\alpha \left[\log p(s_{t+1}|s_t a_t) + V_{S_{t+1}}(s_{t+1})\right]},$$

$$Q_{(S_t A_t)^1}(s_t a_t) = \log p(a_t) + R(s_t a_t) + Q_{(S_t A_t)^4}(s_t a_t),$$

$$V_{S_t}(s_t) = \frac{1}{\alpha} \log \sum_{a_t} e^{\alpha Q_{(S_t A_t)^1}(s_t a_t)},$$

with $0 < \alpha \leq 1$.

In probability space, the updates are translated as

$$b_{(S_t A_t)^4}(s_t a_t) \propto \left[\sum_{s_{t+1}} p(s_{t+1}|s_t a_t)^\alpha b_{S_{t+1}}(s_{t+1})^\alpha\right]^{1/\alpha},$$

$$b_{(S_t A_t)^1}(s_t a_t) \propto p(a_t) c(s_t a_t) b_{(S_t A_t)^4}(s_t a_t),$$

$$b_{S_t}(s_t) \propto \left[\sum_{a_t} b_{(S_t A_t)^1}(s_t a_t)^\alpha\right]^{1/\alpha}.$$

Note that the function $y = \sqrt[\alpha]{\sum_{j=1}^N x_j^\alpha}$ is also a *soft-max* when the α is large. Therefore, both in the log-space and in the probability space, for $\alpha \to \infty$, the parametric soft-max functions converges to the hard max. For $\alpha = 1$, the equations become identical to those derived for the Sum-product algorithm. More details abouts the properties of this soft-max function are in Appendix A.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

The Max-product approach usually produces much more defined value functions and policies, in comparison to the Sum-product, as will be shown in some of the examples that follow. Interpolating between the two solutions provides a whole range of new solutions beyond the traditional Sum-product and Max-product approaches. The Sum/Max-product updates are added as the third row in Tables 1 and 2 and more details about block propagation rules are in Tables 3, 4, 5, 6 and 7. The policy is formally the same as in the Sum-product and the Max-product (8), but evidently, the messages, both in the probability space and in the log-space, carry different information.

The generalization of the Sum/Max-product has been derived as a straightforward interpolation between the Sum-product and the Max-product and such a function can span the whole range of solutions between the maximization of the marginals of the Sum-product algorithm to the maximization of the global posterior of the Max-product. *What is then, for each value of α, the function that the algorithm optimizes?*

In the middle part of Figure 3, we have reported the recursions of the Sum/Max-product algorithm in the probability space for $T = 4$. It is easily seen, by looking at the top of the same figure, that they match the recursions of the Sum-product algorithm as applied to the factorization

$$p(s_1 a_1 \dots s_T a_T)^\alpha = c(s_T a_T)^\alpha p(s_1)^\alpha p(a_T)^\alpha \prod_{t=1}^{T-1} p(s_{t+1}|s_t a_t)^\alpha p(a_t)^\alpha c(s_t a_t)^\alpha,$$

Therefore, in analogy to the Sum-product algorithm, the Sum/Max-product algorithm provides the posteriors

$$p(s_t|K_{1:T}) \propto \sum_{\substack{s_j, j\neq t, j=1:T \\ a_j, j=1:T}} p(s_1 a_1 \dots s_T a_T|K_{1:T})^\alpha,$$

$$p(a_t|K_{1:T}) \propto \sum_{\substack{s_j, j=1:T \\ a_j, j\neq t, j=1:T}} p(s_1 a_1 \dots s_T a_T|K_{1:T})^\alpha,$$

$$p(s_t a_t|K_{1:T}) \propto \sum_{s_j, a_j, j\neq t, j=1:T} p(s_1 a_1 \dots s_T a_T|K_{1:T})^\alpha,$$

if applied globally to the whole time horizon. The Progressive posteriors, just as in Subsection III-B, can be written with the distributions raised to the power α. The details are not repeated here for brevity.

Evidently the power of a distribution is not a normalized distribution, but this is not a problem in message propagation, as we mentioned earlier, because normalization is just a scale that is irrelevant for the inference. To better explain the generalization, recall that raising a probability distribution to a power greater than one, has the effect of sharpening the distribution around its maximum (or maxima, if multiple maxima are present). Therefore, raising the whole joint density to a large power has the effect of concentrating it on the global maximum a posteriori solution of the Max-product algorithm.

VI. DYNAMIC PROGRAMMING ON THE FACTOR GRAPH

The standard dynamic programming approach is based on the maximization of the expected sum of rewards [16], [31]. In previous sections, we have included rewards in factorization (2), but we have formulated the optimization as an estimation problem, i.e. the maximization of posterior probabilities, or marginals, which only implicitly involve the rewards. Evidently, one wonders whether the two approaches can be seen under a unified framework - after all Bellman backups resemble backward message combinations.

We show here that it is possible to map DP directly into the factor graph formulation if we consider rewards and their expectations as contributing to the probability messages, but in the log-space.

The dynamic programming algorithm [16] is derived as the solution to the following problem

$$\begin{aligned}
(a_1^* \dots a_T^*) &= \\
&= \operatorname*{argmax}_{a_1 \dots a_T} \mathbb{E}_{\sim p(s_1 a_1 \dots s_T a_T)} \left[\sum_{t=1}^{T} (R(s_t a_t) + \log p(a_t)) \right] \\
&= \operatorname*{argmax}_{a_1 \dots a_T} \sum_{s_1, \dots, s_T} p(s_1 a_1 \dots s_T a_T) \left[\sum_{t=1}^{T} (R(s_t a_t) + \log p(a_t)) \right] \\
&= \operatorname*{argmax}_{a_1 \dots a_T} \sum_{s_1, \dots, s_T} p(s_1) \prod_{t=1}^{T-1} p(s_{t+1}|s_t a_t) \left[\sum_{t=1}^{T} (R(s_t a_t) + \log p(a_t)) \right],
\end{aligned}$$

where $p(s_1 a_1 \dots s_T a_T)$ does not include the rewards and the priors on a_t appears in the log in the summation. This is slightly different from the pure sum of rewards. We introduce this slight modification because we want to obtain recursions that we can directly compare to the ones derived for the Sum-product and for the Max-product algorithms. In any case, this is not a crucial difference because $\log p(a_t)$ could be incorporated into $R(s_t a_t)$ and $p(a_t)$ can be assumed to be uniform.

We have reported in bottom part of Figure 3, the DP recursions for $T = 4$. Note that here the rewards appear as additive terms and there is a mix of maxima and sums. Formally

$$\begin{aligned}
Q_{(S_t A_t)^1}(s_t a_t) &= \log p(a_t) + R(s_t a_t) + \\
&\quad + \sum_{s_{t+1}} p(s_{t+1}|s_t a_t) V_{S_{t+1}}(s_{t+1}), \\
V_{S_t}(s_t) &= \max_{a_t} Q_{(S_t A_t)^1}(s_t a_t).
\end{aligned}$$

Translating the recursions in the probability space, we have

$$\begin{aligned}
b_{(S_t A_t)^1}(s_t a_t) &= p(a_t) c(s_t a_t) e^{\sum_{s_{t+1}} p(s_{t+1}|s_t a_t) \log b_{S_{t+1}}(s_{t+1})}, \\
b_{S_t}(s_t) &= \max_{a_t} b_{(S_t A_t)^1}(s_t a_t).
\end{aligned}$$

$$
\begin{array}{l}
p(s_1) \\
\quad p(a_1)c(s_1a_1) \\
\quad\quad p(s_2|s_1a_1) \\
\quad\quad\quad p(a_2)c(s_2a_2) \\
\quad\quad\quad\quad p(s_3|s_2a_2) \\
\quad\quad\quad\quad\quad p(a_3)c(s_3a_3) \\
\quad\quad\quad\quad\quad\quad p(s_4|s_3a_3) \\
\quad\quad\quad\quad\quad\quad\quad p(a_4)c(s_4a_4) \\
\quad\quad\quad\quad\quad\quad\quad b_{(S_4A_4)^1}(s_4a_4) = p(a_4)c(s_4a_4)b_{(S_4A_4)^4}(s_4a_4) \\
\quad\quad\quad\quad\quad\quad\quad b_{S_4}(s_4) = \max_{a_4} b_{(S_4A_4)^1}(s_4a_4) \\
\quad\quad\quad\quad\quad\quad b_{(S_3A_3)^4}(s_3a_3) = \max_{s_4} p(s_4|s_3a_3)b_{S_4}(s_4) \\
\quad\quad\quad\quad\quad b_{(S_3A_3)^1}(s_3a_3) = p(a_3)c(s_3a_3)b_{(S_3A_3)^4}(s_3a_3) \\
\quad\quad\quad\quad\quad b_{S_3}(s_3) = \max_{a_3} b_{(S_3A_3)^1}(s_3a_3) \\
\quad\quad\quad\quad b_{(S_2A_2)^4}(s_2a_2) = \max_{s_3} p(s_3|s_2a_2)b_{S_3}(s_3) \\
\quad\quad\quad b_{(S_2A_2)^1}(s_2a_2) = p(a_2)c(s_2a_2)b_{(S_2A_2)^4}(s_2a_2) \\
\quad\quad\quad b_{S_2}(s_2) = \max_{a_2} b_{(S_2A_2)^1}(s_2a_2) \\
\quad\quad b_{(S_1A_1)^4}(s_1a_1) = \max_{s_2} p(s_2|s_1a_1)b_{S_2}(s_2) \\
\quad b_{(S_1A_1)^1}(s_1a_1) = p(a_1)c(s_1a_1)b_{(S_1A_1)^4}(s_1a_1) \\
\quad b_{S_1}(s_1) = \max_{a_1} b_{(S_1A_1)^1}(s_1a_1)
\end{array}
$$

$$
\begin{array}{l}
p(s_1)^\alpha \\
\quad p(a_1)^\alpha c(s_1a_1)^\alpha \\
\quad\quad p(s_2|s_1a_1)^\alpha \\
\quad\quad\quad p(a_2)^\alpha c(s_2a_2)^\alpha \\
\quad\quad\quad\quad p(s_3|s_2a_2)^\alpha \\
\quad\quad\quad\quad\quad p(a_3)^\alpha c(s_3a_3)^\alpha \\
\quad\quad\quad\quad\quad\quad p(s_4|s_3a_3)^\alpha \\
\quad\quad\quad\quad\quad\quad\quad p(a_4)^\alpha c(s_4a_4)^\alpha \\
\quad\quad\quad\quad\quad\quad\quad b_{(S_4A_4)^1}(s_4a_4) = p(a_4)^\alpha c(s_4a_4)^\alpha b_{(S_4A_4)^4}(s_4a_4)^\alpha \\
\quad\quad\quad\quad\quad\quad\quad b_{S_4}(s_4) = (\sum_{a_4} b_{(S_4A_4)^1}(s_4a_4)^\alpha)^{\frac{1}{\alpha}} \\
\quad\quad\quad\quad\quad\quad b_{(S_3A_3)^4}(s_3a_3) = (\sum_{s_4} p(s_4|s_3a_3)^\alpha b_{S_4}(s_4)^\alpha)^{\frac{1}{\alpha}} \\
\quad\quad\quad\quad\quad b_{(S_3A_3)^1}(s_3a_3) = p(a_3)^\alpha c(s_3a_3)^\alpha b_{(S_3A_3)^4}(s_3a_3)^\alpha \\
\quad\quad\quad\quad\quad b_{S_3}(s_3) = (\sum_{a_3} b_{(S_3A_3)^1}(s_3a_3)^\alpha)^{\frac{1}{\alpha}} \\
\quad\quad\quad\quad b_{(S_2A_2)^4}(s_2a_2) = (\sum_{s_3} p(s_3|s_2a_2)^\alpha b_{S_3}(s_3)^\alpha)^{\frac{1}{\alpha}} \\
\quad\quad\quad b_{(S_2A_2)^1}(s_2a_2) = p(a_2)^\alpha c(s_2a_2)^\alpha b_{(S_2A_2)^4}(s_2a_2)^\alpha \\
\quad\quad\quad b_{S_2}(s_2) = (\sum_{a_2} b_{(S_2A_2)^1}(s_2a_2)^\alpha)^{\frac{1}{\alpha}} \\
\quad\quad b_{(S_1A_1)^4}(s_1a_1) = (\sum_{s_2} p(s_2|s_1a_1)^\alpha b_{S_2}(s_2)^\alpha)^{\frac{1}{\alpha}} \\
\quad b_{(S_1A_1)^1}(s_1a_1) = p(a_1)^\alpha c(s_1a_1)^\alpha b_{(S_1A_1)^4}(s_1a_1)^\alpha \\
\quad b_{S_1}(s_1) = (\sum_{a_1} b_{(S_1A_1)^1}(s_1a_1)^\alpha)^{\frac{1}{\alpha}}
\end{array}
$$

$$
\begin{array}{l}
\sum_{s_1} p(s_1)\,[\log p(a_1) + R(s_1a_1) + \sum_{s_2} p(s_2|s_1a_1)\,[\log p(a_2) + R(s_2a_2) + \sum_{s_3} p(s_3|s_2a_2)\,[\log p(a_3) + R(s_3a_3) + \sum_{s_4} p(s_4|s_3a_3)\,[\log p(a_4) + R(s_4a_4) + Q_{(S_4A_4)^4}(s_4a_4)]]]] \\
\quad Q_{(S_4A_4)^1}(s_4a_4) = \log p(a_4) + R(s_4a_4) + Q_{(S_4A_4)^4}(s_4a_4) \\
\quad V_{S_4}(s_4) = \max_{a_4} Q_{(S_4A_4)^1}(s_4a_4) \\
\quad Q_{(S_3A_3)^4}(s_3a_3) = \sum_{s_4} p(s_4|s_3a_3)V_{S_4}(s_4) \\
\quad Q_{(S_3A_3)^1}(s_3a_3) = \log p(a_3) + R(s_3a_3) + Q_{(S_3A_3)^4}(s_3a_3) \\
\quad V_{S_3}(s_3) = \max_{a_3} Q_{(S_3A_3)^1}(s_3a_3) \\
\quad Q_{(S_2A_2)^4}(s_2a_2) = \sum_{s_3} p(s_3|s_2a_2)V_{S_3}(s_3) \\
\quad Q_{(S_2A_2)^1}(s_2a_2) = \log p(a_2) + R(s_2a_2) + Q_{(S_2A_2)^4}(s_2a_2) \\
\quad V_{S_2}(s_2) = \max_{a_2} Q_{(S_2A_2)^1}(s_2a_2) \\
\quad Q_{(S_1A_1)^4}(s_1a_1) = \sum_{s_2} p(s_2|s_1a_1)V_{S_2}(s_2) \\
Q_{(S_1A_1)^1}(s_1a_1) = \log p(a_1) + R(s_1a_1) + Q_{(S_1A_1)^4}(s_1a_1) \\
V_{S_1}(s_1) = \max_{a_1} Q_{(S_1A_1)^1}(s_1a_1)
\end{array}
$$

FIGURE 3. Backward recursions for $T = 4$. Max-product (top); Sum/Max-product (middle); Dynamic programming (bottom). Note the presence of the backward message at the end of the chain that may carry information from further steps or may represent final constraints. The recursions must be read from right to left.

The crucial difference between DP and the Sum-product algorithm is in the fact that averages and maxima are taken in the log-space on the value function. Conversely in the Sum-product, they are taken in the probability space on the backward distributions. Therefore, *DP can be formulated in terms of probability messages traveling on the same factor graph of the Sum-product algorithm, but with a different combination rules*. All the propagation rules for DP in the FGrn of Figure 2 are reported in Tables 3, 4, 5, 6 and 7. The main DP recursions are also in Tables 1 and 2 for comparison.

The best SASA sequence, written both in the log-space and in the probability space, is immediately derived from the graph of Figure 2 as

$$
\begin{aligned}
s_t^* &= \operatorname*{argmax}_{s_t} p(s_t|s_{t-1}^* a_{t-1}^*) V_{S_t}(s_t) \\
&= \operatorname*{argmax}_{s_t} p(s_t|s_{t-1}^* a_{t-1}^*) \log b_{S_t}(s_t), \\
a_t^* &= \operatorname*{argmax}_{a_t} Q_{(S_t A_t)^1}(s_t^* a_t) = \operatorname*{argmax}_{a_t} b_{(S_t A_t)^1}(s_t^* a_t),
\end{aligned}
$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

The policy distribution has the same formal expression (8).

VII. SOFTDP

The presence of the max operator in the DP algorithm, suggests that, similarly to the Sum/Max-product approach, we could replace the max operator with a soft-max function to provide a different interpolation between a more entropic solution and the optimal DP algorithm. Using the soft-max function $r(x_1,\dots,x_N;\beta)=\sum_{i=1}^N x_i e^{\beta x_i}/\sum_{j=1}^N e^{\beta x_j}$ (common in neural network architectures), we propose the following *SoftDP* updates

$$\begin{aligned} Q_{(S_tA_t)^1}(s_ta_t) &= \log p(a_t) + R(s_ta_t) \\ &\quad + \sum_{s_{t+1}} p(s_{t+1}|s_ta_t)V_{S_{t+1}}(s_{t+1}), \\ V_{S_t}(s_t) &= \frac{\sum_{a_t} e^{\beta Q_{(S_tA_t)^1}(s_ta_t)} Q_{(S_tA_t)^1}(s_ta_t)}{\sum_{a'_t} e^{\beta Q_{(S_tA_t)^1}(s_ta'_t)}}. \end{aligned}$$

The parameter $\beta \geq 0$ can be used to control the sharpness of the soft-max function. If β is a large positive number, the soft-max tends to the maximum. When β is a small positive number, the soft-max function tends to return the mean. The soft-max is further discussed in Appendix A. We have not investigated the existence of a function that these recursions optimize for a finite value of β, as in the case of the Sum/Max-product algorithm. We leave it to further analyses. However, we observe that lowering the value of β shifts the policy distribution towards a smoother, i.e., more entropic, configuration. We show this effect in the simulations (§XI).

In the probability space, the recursion for the backward message $b_{(S_tA_t)^1}(s_ta_t)$ is the same as in DP, while the update for $b_{S_t}(s_t)$ becomes

$$b_{S_t}(s_t) \propto \exp\left[\frac{\sum_{a_t} \log b_{(S_tA_t)^1}(s_ta_t) b_{S_t}(s_t)^\beta}{\sum_{a'_t} b_{(S_tA_t)^1}(s_ta'_t)^\beta}\right].$$

We add this as a SoftDP option in our suite of algorithms with all the propagation rules on the FGrn specified in Tables 3, 4, 5, 6 and 7, and the main recursions included in Tables 1 and 2 for comparison.

VIII. MAXIMUM EXPECTED REWARD AND ENTROPY

In all the previous approaches to estimation and control, we have derived the policies *as consequences* of optimization on the graph of Figure 2. A different formulation can be adopted if we formally add to the Bayesian graph "policy" branches $\pi(a_t|s_t)$ that go from each state S_t to each action A_t and pose the problem as the functional optimization problem of finding the best $\pi(a_t|s_t)$, given the evidence $K_{1:T}$. The question is: how do we formalized the total reward function? Levine [3], in his excellent review, suggests that "less confident" behaviors with respect to the standard probabilistic inference (the Sum-product) could be obtained if we modify the function to optimize. In fact, he maintains that the recursions for the Sum-product approach derived above, may be too optimistic within the context of RL. The idea is to add an extra term to the rewards to account also for policy entropy. Levine shows that the modification can also be related to structural variational inference [3]. Entropy maximization is also a common criterion in practical uses of RL [33] and stochastic control [12]. Levine [3] proposes the following formulation:

$$\begin{aligned} &\{\pi^*(a_1|s_1)\dots\pi^*(a_T|s_T)\} = \\ &\quad \operatorname{argmax} \mathbb{E}_{\sim\hat{p}(s_1a_1\dots s_Ta_T)}\left[\sum_{t=1}^T R'(s_ta_t) - \log\pi(a_t|s_t)\right], \end{aligned}$$

where $R'(s_ta_t) = R(s_ta_t) + \log p(a_t)$ and

$$\begin{aligned} &\hat{p}(s_1a_1\dots s_Ta_T) = \\ &\qquad p(s_1)\pi(a_T|s_T)\prod_{t=1}^{T-1} p(s_{t+1}|s_ta_t)\pi(a_t|s_t). \end{aligned}$$

Note that here the policy distributions are included in the factorization. The extra term $\log\pi(a_t|s_t)$ pushes for entropy maximization. The backup recursions for the optimal policy distributions [3], in the factor graph notations, are

$$\begin{aligned} Q_{(S_tA_t)^1}(s_ta_t) &= \log p(a_t) + R(s_ta_t) \\ &\quad + \sum_{s_{t+1}} p(s_{t+1}|s_ta_t)V_{S_{t+1}}(s_{t+1}), \quad (9a) \\ V_{S_t}(s_t) &= \frac{1}{\alpha}\log\sum_{a_t} e^{\alpha Q_{(S_tA_t)^1}(s_ta_t)}, \quad (9b) \end{aligned}$$

with $\alpha = 1$. The optimal policy distributions are also shown to have the usual formal expression $\pi^*(a_t|s_t) \propto e^{\left(Q_{(S_tA_t)^1}(s_ta_t) - V(s_t)\right)}$. In our effort to provide more general approaches to the policy search, we have generalized the soft-max function to include an extra parameter α in (9) that for $\alpha\to\infty$ gives the maximum, and therefore the DP solution, and for $\alpha = 1$ Levine's Max-Reward/Entropy solution.

We have worked backward the recursion (9) for a generic α, and shown in Appendix B that the generalized recursions solve the following optimization problem

$$\begin{aligned} \{\pi^*(a_1|s_1)\dots\pi^*(a_T|s_T)\} &= \operatorname{argmax}\mathbb{E}_{\sim\hat{p}(s_1a_1\dots s_Ta_T)} \\ &\left[\sum_{t=1}^T R'(s_ta_t) - \frac{1}{\alpha}\log\pi_\alpha(a_t|s_t)\right], \quad (10) \end{aligned}$$

where

$$\begin{aligned} R'(s_ta_t) &= R(s_ta_t) + \log p(a_t), \\ \pi_\alpha(a_t|s_t) &= \frac{\pi(a_t|s_t)^\alpha}{\sum_{a'_t}\pi(a'_t|s_t)^\alpha}, \quad t = 1:T; \\ \hat{p}(s_1a_1\dots s_Ta_T) &= \\ &\quad p(s_1)\pi_\alpha(a_T|s_T)\prod_{t=1}^{T-1} p(s_{t+1}|s_ta_t)\pi_\alpha(a_t|s_t). \end{aligned}$$

The proof in Appendix B follows steps similar to those used by Levine [3] and where we show how the extra term gives rise to (iterative) simultaneous reward and entropy maximization. Note that when α is large, the extra term becomes

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

progressively irrelevant, and the distributions $\pi_\alpha(a_t|s_t)$ become more concentrated on the max value of the Q-function, thereby recovering the DP solution. Furthermore, when $\alpha < 1$, more weight is given to the extra term, the distributions $\pi_\alpha(a_t|s_t)$ becomes smoother and we have more entropic policy distributions. We demonstrate this effect in some of the simulations that follow. It should be mentioned that the criterion does not simply add an entropy term to the rewards, because the policy distribution affects also the reward as it appears in the factorization used in the expectation (see Appendix B for additional insights).

The propagation rules in the FGrn for the Max-Rew/Entropy approach are in Tables 3, 4, 5, 6 and 7, and the main recursions are added to Tables 1 and 2 also in the probability space as

$$b_{(S_tA_t)^1}(s_ta_t) = p(a_t)c(s_ta_t)\, e^{\sum_{s_{t+1}} p(s_{t+1}|s_ta_t)\log b_{S_{t+1}}(s_{t+1})}$$

$$b_{S_t}(s_t) = \left[\sum_{a_t} b_{(S_tA_t)^1}(s_ta_t)^\alpha\right]^{\frac{1}{\alpha}}.$$

IX. DETERMINISTIC SYSTEMS

The approach to optimal control in this paper is based on the assumption that the system description $p(s_{t+1}|s_ta_t)$ is stochastic. There are cases, however, in which the system transitions are deterministic, i.e., given s_ta_t, we have exact knowledge of s_{t+1} through a deterministic function $s_{t+1} = g(s_ta_t)$. The beauty of the stochastic framework is that these special cases are also included in the formulation and correspond to a transition probability function that is a delta function $p(s_{t+1}|s_ta_t) = \delta\left(s_{t+1} - g(s_ta_t)\right)$. Also, if no prior on the actions is available, $p(a_t) = U(a_t)$. The updates do not change, but some of them in the various methods may coincide, because the summations (expectations) in the updates disappear and the prior on A_t is irrelevant. More specifically, by looking at Tables 1 and 2, the updates for the Q-functions, and their probability-space counterparts, have the same (Bellman's) recursions

$$Q_{(S_tA_t)^1}(s_ta_t) = R(s_ta_t) + V_{S_{t+1}}(g(s_ta_t)),$$
$$b_{(S_tA_t)^1}(s_ta_t) = c(s_ta_t)b_{S_{t+1}}(g(s_ta_t)).$$

However, there are differences in the V-function updates. For the Sum-product and the Max-Rew/Ent ($\alpha = 1$), we have

$$V_{S_t}(s_t) = \log\sum_{a_t} e^{Q_{(S_tA_t)^1}(s_ta_t)},$$
$$b_{S_t}(s_t) = \sum_{a_t} b_{(S_tA_t)^1}(s_ta_t).$$

For the Max-product and DP, we have

$$V_{S_t}(s_t) = \max_{a_t} Q_{(S_tA_t)^1}(s_ta_t),$$
$$b_{S_t}(s_t) = \max_{a_t} b_{(S_tA_t)^1}(s_ta_t).$$

For the others, we have the parametrized soft-max function with various values of β and α. Therefore, by direct comparison, we can conclude that, when the system is deterministic: *DP* and *Max-product* coincide; *Mean-product* and *Max-Rew/Ent* ($\alpha = 1$) coincide (also recognized in [3]). The remaining cases are interpolations of the others. We have verified in our limited simulations that this is indeed the case and that the solutions in the various groups, even in this deterministic case, are different.

X. INFINITE HORIZON CASE AND THE STEADY-STATE

We have presented the model in Figure 1 and the various algorithms that stem from the model with reference to a finite horizon scenario. However, all the analyses easily extends to an infinite-horizon framework simply by adding a discount factor $0 < \gamma \leq 1$ to the optimized functions and then to the updates. For example, the standard DP updates, in both spaces become

$$Q_{(S_tA_t)^1}(s_ta_t) = \log p(a_t) + R(s_ta_t) + \gamma\sum_{s_{t+1}} p(s_{t+1}|s_ta_t)V_{S_{t+1}}(s_{t+1}),$$
$$b_{(S_tA_t)^1}(s_ta_t) = p(a_t)c(s_ta_t)\, e^{\gamma\sum_{s_{t+1}} p(s_{t+1}|s_ta_t)\log b_{S_{t+1}}(s_{t+1})}.$$

Also, for the Sum-product, we have

$$Q_{(S_tA_t)^1}(s_ta_t) = \log p(a_t) + R(s_ta_t) + \gamma\log\sum_{s_{t+1}} e^{\log p(s_{t+1}|s_ta_t)+V_{S_{t+1}}(s_{t+1})}$$

In general, even if $\gamma = 1$, the backward recursions can be run to verify that a steady-state configuration for the Q, the V-function and the policy π^* can be found. The analysis of the mathematical conditions for convergence are beyond the scope of this paper. However, generally speaking, if all the states are reachable, a stable configuration should exist. We have verified experimentally that all the methods do converge (also for $\gamma = 1$), but they exhibit marked differences in the number of iterations required to reach the steady state equilibrium. The Max-product algorithm shows the fastest convergence with the Sum-product following in the list. DP and the other methods seem to show a much slower convergence speed. We show this effect in the simulations that follow.

XI. SIMULATIONS

We have simulated the various recursions on a path planning problem on two discrete grids. The first set of simulations is performed on a small 6x6 square grid shown in Figure 4, where we have one goal (bull's eye and green) and obstacles (dark gray). The states are the positions on the grid and the actions correspond to one of the nine possible single-pixel motions {up-left, up, up-right, left, center (still), right, down-left, down, down-right}. The reward function has the values 0 on the goal, -10 on the obstacles and -1 on other

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

TABLE 3. Forward distributions for the source blocks.

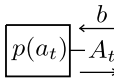
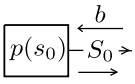
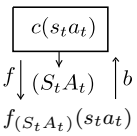
	 $f_{A_t}(a_t)$	 $f_{S_0}(s_0)$	 $f_{(S_t A_t)}(s_t a_t)$
Sum product Max product Sum/Max product DP Max-Rew/Ent SoftDP	$p(a_t)$	$p(s_0)$	$c(s_t a_t)$

TABLE 4. Propagation rules for action shaded blocks.

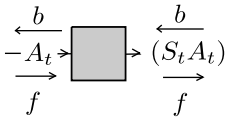
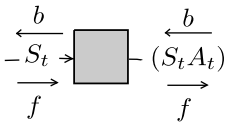
	 $b_{A_t}(a_t)$	$f_{(S_t A_t)}(s_t a_t)$
Sum product Max-Rew/Ent ($\alpha = 1$)	$\sum_{s_t} b_{(S_t A_t)}(s_t a_t)$	$f_{A_t}(a_t) U(s_t)$
Max product DP	$\max_{s_t} b_{(S_t A_t)}(s_t a_t)$	$f_{A_t}(a_t) U(s_t)$
Sum/Max product Max-Rew/Ent ($\alpha \neq 1$)	$\sqrt[\alpha]{\sum_{s_t} b_{(S_t A_t)}(s_t a_t)^\alpha}$	$f_{A_t}(a_t) U(s_t)$
SoftDP	$\exp\left[\frac{\sum_{s_t} \log b_{(S_t A_t)}(s_t a_t) b_{(S_t A_t)}(s_t a_t)^\beta}{\sum_{s'_t} b_{(S_t A_t)}(s'_t a_t)^\beta}\right]$	$f_{A_t}(a_t) U(s_t)$

TABLE 5. Propagation rules for state shaded blocks.

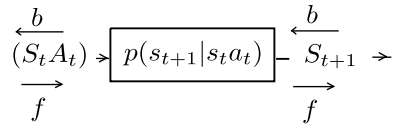
	 $b_{S_t}(s_t)$	$f_{(S_t A_t)}(s_t a_t)$
Sum product Max-Rew/Ent ($\alpha = 1$)	$\sum_{a_t} b_{(S_t A_t)}(s_t a_t)$	$f_{S_t}(s_t) U(a_t)$
Max product DP	$\max_{a_t} b_{(S_t A_t)}(s_t a_t)$	$f_{S_t}(s_t) U(a_t)$
Sum/Max product Max-Rew/Ent ($\alpha \neq 1$)	$\sqrt[\alpha]{\sum_{a_t} b_{(S_t A_t)}(s_t a_t)^\alpha}$	$f_{S_t}(s_t) U(a_t)$
SoftDP	$\exp\left[\frac{\sum_{a_t} \log b_{(S_t A_t)}(s_t a_t) b_{(S_t A_t)}(s_t a_t)^\beta}{\sum_{a'_t} b_{(S_t A_t)}(s_t a'_t)^\beta}\right]$	$f_{S_t}(s_t) U(a_t)$

pixel positions. The motion is stochastic with a transition function $p(s_{t+1}|s_t a_t)$ that has probability $1/2$ for the intended direction and the rest of the probability $(1/2)$ spread equally on the other eight directions. Built in the transition function are also re-normalizations when the transition is close to the boundaries: when some of the new projected states are outside the grid, their probabilities are set to zero, and the remaining probability is spread equally on the other pixels. No initial or final conditions are set on the model. The recursions are run until convergence to a steady state value function.

All the algorithms lead to policies that would allow an agent, starting from any position on the grid, to reach the goal in a finite number of steps. The values reported in the squares and the max policy arrows in Figure 4 reveal how the different solutions direct our potential agent in slightly

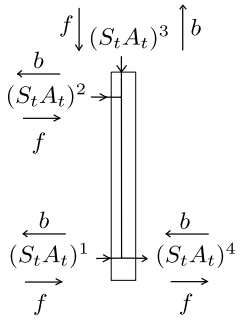
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

TABLE 6. Propagation rules for the dynamics block.



	$b_{(S_tA_t)}(s_ta_t)$	$f_{S_{t+1}}(s_{t+1})$
Sum product	$\sum_{s_{t+1}} p(s_{t+1}\|s_ta_t) b_{S_{t+1}}(s_{t+1})$	$\sum_{s_ta_t} p(s_{t+1}\|s_ta_t) f_{(S_tA_t)}(s_ta_t)$
Max product	$\max_{s_{t+1}} p(s_{t+1}\|s_ta_t) b_{S_{t+1}}(s_{t+1})$	$\max_{s_ta_t} p(s_{t+1}\|s_ta_t) f_{(S_tA_t)}(s_ta_t)$
Sum/Max product	$\sqrt[\alpha]{\sum_{s_{t+1}} p(s_{t+1}\|s_ta_t)^\alpha b_{S_{t+1}}(s_{t+1})^\alpha}$	$\sqrt[\alpha]{\sum_{s_ta_t} p(s_{t+1}\|s_ta_t)^\alpha f_{(S_tA_t)}(s_ta_t)^\alpha}$
DP SoftDP Max-Rew/Ent	$e^{\sum_{s_{t+1}} p(s_{t+1}\|s_ta_t) \log b_{S_{t+1}}(s_{t+1})}$	$e^{\sum_{s_ta_t} p(s_{t+1}\|s_ta_t) \log f_{(S_tA_t)}(s_ta_t)}$

TABLE 7. Propagation rules for the diverter.



Sum product Max product Sum/Max product DP Max-Rew/Ent SoftDP	$b_{(S_tA_t)^1}(s_ta_t) \propto f_{(S_tA_t)^2}(s_ta_t) f_{(S_tA_t)^3}(s_ta_t) b_{(S_tA_t)^4}(s_ta_t)$ $b_{(S_tA_t)^2}(s_ta_t) \propto f_{(S_tA_t)^1}(s_ta_t) f_{(S_tA_t)^3}(s_ta_t) b_{(S_tA_t)^4}(s_ta_t)$ $b_{(S_tA_t)^3}(s_ta_t) \propto f_{(S_tA_t)^1}(s_ta_t) f_{(S_tA_t)^2}(s_ta_t) b_{(S_tA_t)^4}(s_ta_t)$ $f_{(S_tA_t)^4}(s_ta_t) \propto f_{(S_tA_t)^1}(s_ta_t) f_{(S_tA_t)^2}(s_ta_t) b_{(S_tA_t)^3}(s_ta_t)$

different paths to avoid the obstacles. In the lower right corner of the figure, we also report the increments in reaching the steady-state solution for the various algorithms in a log-log graph (also the parameters are reported in the legend).

The algorithm is stopped only when all the increments in the value function are below 10^{-5}. It is noteworthy to see how the Max-product algorithm reaches the steady-state solution in a very limited number of steps (fastest convergence) and how the Sum-product and the Sum/Max-product algorithms converge at a much faster rate in comparison to the others.

The results of another set of simulations are reported in Figures 5, 6, 7, 9 and 8. Here, we have a grid extracted from a real dataset acquired at an intersection on the Stanford campus with pedestrians and bikes. The scene, with no agents, is simplified to 17×23 pixels, with goals (exits) and rewards assigned to various areas (semantic map) as shown in Figure 5. We assume that our agent is a pedestrian and the actions are the same nine actions we have used above for the smaller grid. The rewards are: $R(s_t) = 0$ (goals: bull's eye and green); -1 (pedestrian walkways: white); -10 (streets: light gray); -20 (grass: dark gray); -30 (obstacles: dark). The convergence behavior to a steady-state value function is similar to the one shown for the smaller grid. The number of iterations to reach a precision of 10^{-5} on all the states are: [Sum-product: 29; Max-product: 11; Sum/Max-product ($\alpha = 3$): 15; Soft DP ($\beta = 0.2$): 127; Soft DP ($\beta = 0.6$): 113; DP: 120; Max Rew/Ent ($\alpha = 0.2$): 187; Max Rew/Ent ($\alpha = 1$): 128; Max Rew/Ent ($\alpha = 6$): 120]. Note how quickly the Max-product and the probabilistic methods converge when compared to the others. The graph of the actual increments for the various algorithms is shown in the log scale in Figure 8. Figure 5 shows, for each state, the maximum policy directions for all the algorithms. The arrows point towards the preferences implied by the semantic information: pedestrians prefer walkways to streets; grass and obstacles are avoided. We observe a marked effect on the results of the Max-product algorithm that maintains the multiple maxima

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

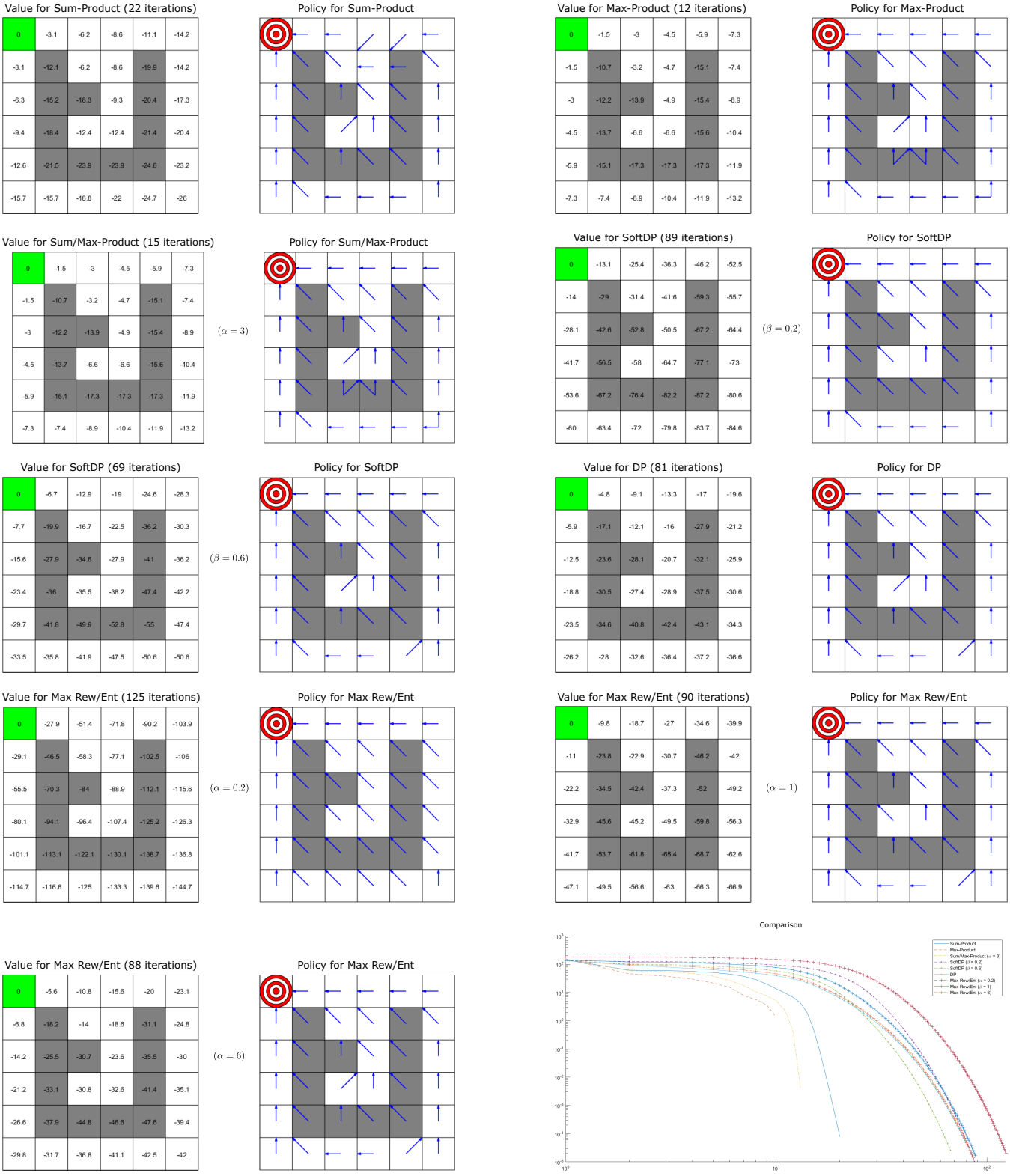


FIGURE 4. Max policy direction for the various algorithms (right column). At the top of each figure are also reported the number of iterations necessary to reach a steady-state value function (on the left columns the numerica values). The lowest right plot shows the value function increments as the iterations progress towards steady-state.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

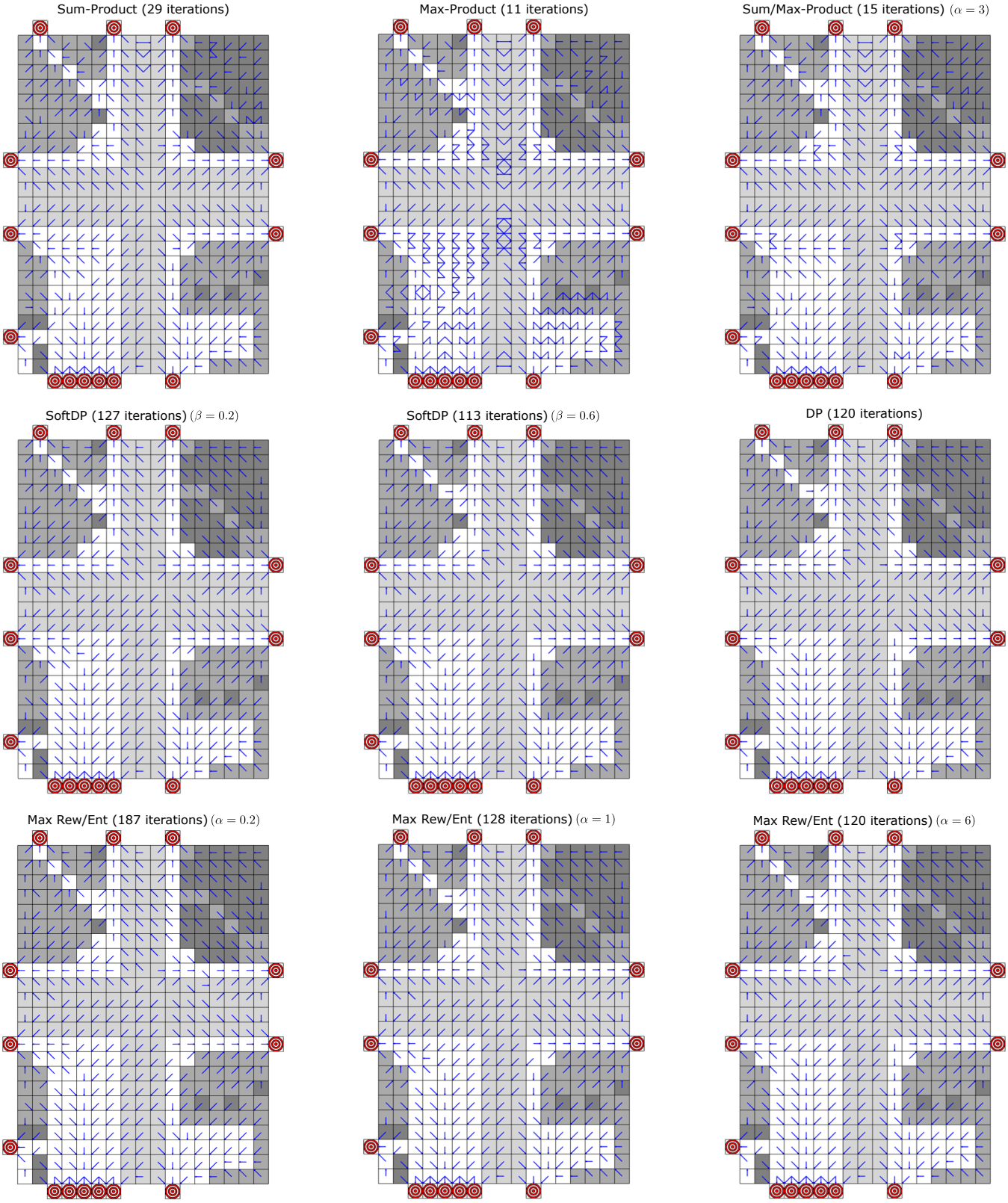


FIGURE 5. Max policy direction for the various algorithms (right column). At the top each figure are also reported the number of iterations necessary for the value function to reach its steady-state configuration.

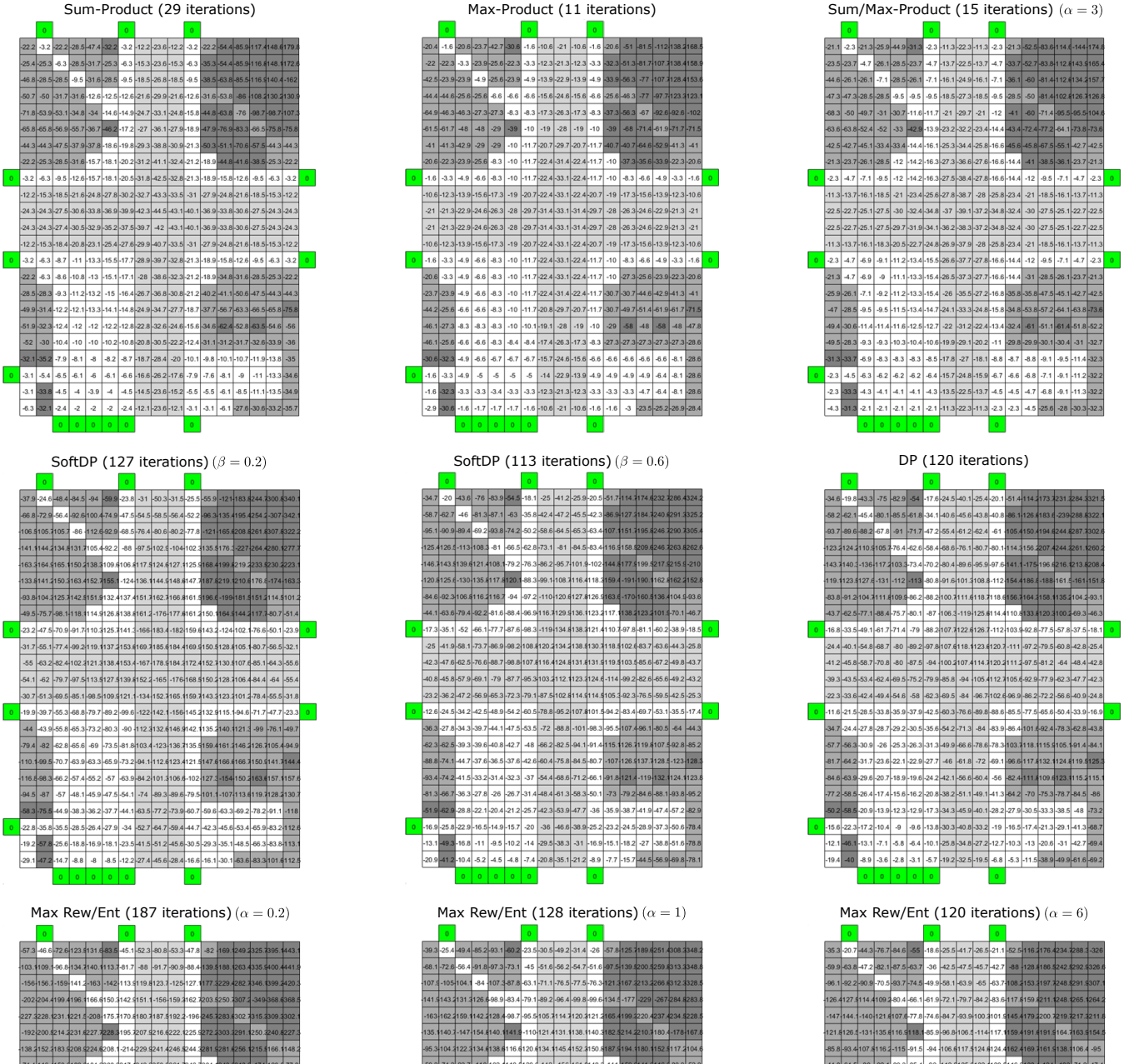


FIGURE 6. Numerical visualization of the value function for the various algorithms.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

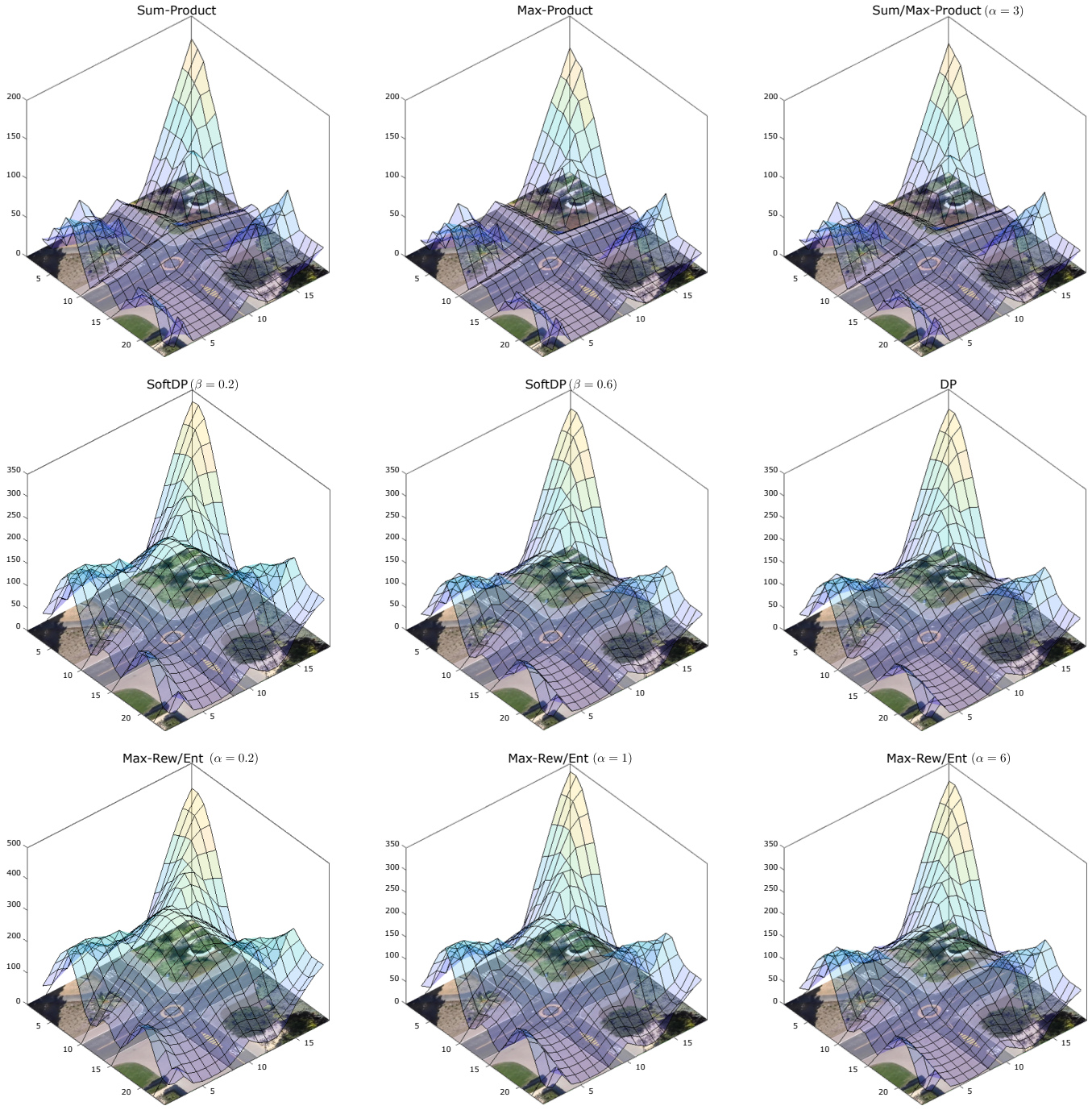


FIGURE 7. 3D Plots of $-V_{S_t}(s_t)$ for the various algorithms.

directions corresponding to the equivalent solutions. These multiple options appear smoothed out in the other methods. The Max-product requires the minimum number of steps to stabilize its configurations. Figure 7 shows also some of the negative value functions $-V(s_t)$ (they can be thought as potential functions) superimposed on the original scene for the various methods and for some hyper-parameter choices. The comparison clearly shows that the various algorithm lead to intriguingly smooth solutions, except for the Max-product that produces a very sharp value function with very well defined valleys.

Just as in the simulations on the smaller grid, we have included no paths on the map, because in all methods an agent that starts anywhere, will reach one of the goals in all cases. This can be easily verified by following the arrows in Figure 5. A much more revealing visualization of the differences among the various methods is displayed in Figure 9, where at a generic point on the map, we plot the policy distributions. In

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

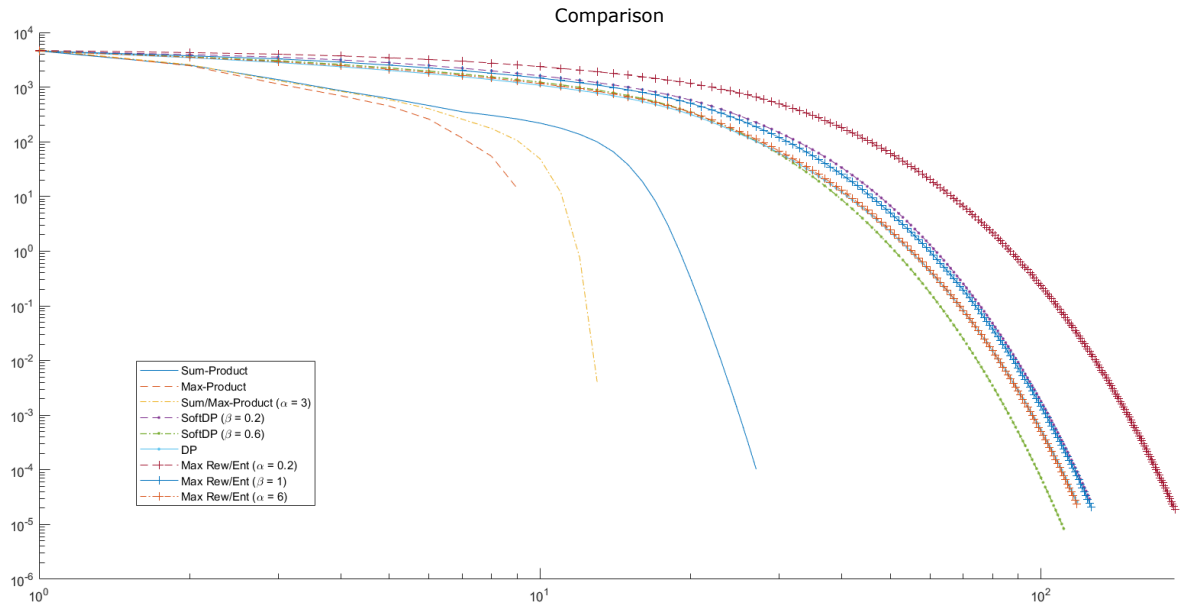


FIGURE 8. Comparison of the value function increments for the various algorithm for the example of Figure 5.

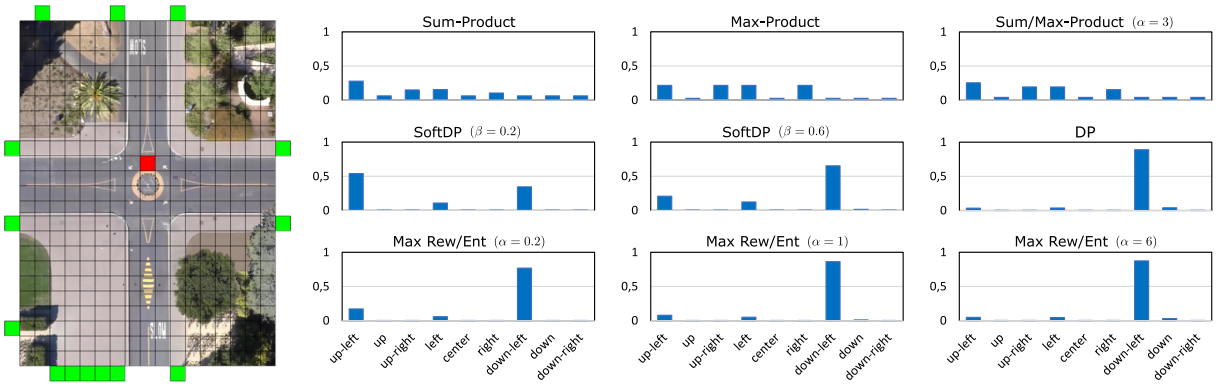


FIGURE 9. Policy distributions at a generic point (red on the map) for the various methods for different parameter choices. The goals are depicted in green.

the first column, the policies for the probabilistic methods are shown with the Max-product clearly producing a rather sharp behavior with all the multiple equivalent options. Recall that the map has multiple goals and the agent in that position has more than one option to achieve optimality (see also Figure 5 in that position). In the second column, we report the results of the DP approach in its standard form (bottom graph) and in its soft parametrized versions. Note how, for the two values $\beta = 0.3$ and $\beta = 0.6$, an agent may be led to consider more options with respect to DP and if we look also at the maximum policy on the map of Figure 5, it may even result in a different path. In the third column, we report the results of the Max Rew/Ent algorithm for various values of the parameter α. We notice, as expected from the theory, that when $\alpha < 1$, the policy distribution is more entropic and that when α increases, the distribution tends to the DP policy.

XII. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, we have provided a unified view of a set of solutions to the MDP problem using probabilistic formalism on a factor graph. The various algorithms span a gamut from a standard Sum-product (marginalization), to Dynamic programming to Max-Reward/Entropy methods. We have shown how the various approaches can be viewed as different combination rules through two of the blocks on a Factor Graph in Reduced Normal form, showing that estimation and control may merge naturally using this framework. Our review of the classical methods has been augmented with original parametric generalizations, providing a whole suite of algorithms that can be easily implemented using the same belief propagation framework. The resulting set of choices, presented here, both in the probability and in the log space, may enhance the options for decision makers that may need to control the sharpness of their solutions by adopting more or less entropic cost functions. The set of solutions provided

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

here, may also be useful in designing on-line reinforcement learning algorithms that may require V- or Q-functions that seek a balance between exploration and exploitation in their current model knowledge. We have included in this paper typical results on discrete grids that reveal marked differences among some of the methods. Our computational results suggest that the Max-product algorithm, optimal maximum a posteriori solution, together with other probabilistic methods, such as the Sum-product and its Sum/Max-product generalizations, shows faster converge to the steady-state configuration in comparison to other reward-based methods, that are typically derived in the log space. Another practical advantage of having a unified formulation for the various algorithms is software tranferability: a unified algorithm can easily accomodate risk propensity of decision makers. For the path planning problem on grids, we have produced simulation software that can be obtained on request. Further work on the topic will be devoted to continuous spaces, approximations based on heuristic serach, and applications involving interacting agents.

.

APPENDIX A SOFT-MAX FUNCTIONS

We review here some of the soft-max functions that are used in the recursions discussed in the paper. For all the functions, we consider the ranked set of real numbers $x_1, x_2, \ldots, x_N$, with $x_1 \leq x_2 \leq \cdots \leq x_N$.

Theorem A.1. *Consider the following expression*

$$s(x_1, \ldots, x_N) = \log \sum_{j=1}^{N} e^{x_j}.$$

This function has the property that when $x_N \gg x_{N-1}$, $s(x_1, \ldots, x_N) \to \max(x_1, \ldots, x_N) = x_N$.

Proof. The function can rewritten as

$$\begin{aligned} s(x_1, \ldots, x_N) &= \log \left(e^{x_1} + e^{x_2} + \cdots + e^{x_N}\right) \\ &= \log e^{x_N} \left(e^{x_1 - x_N} + e^{x_2 - x_N} + \cdots + 1\right) \end{aligned}$$

When x_N is large, the differences also become large negative numbers. Therefore, the first $N-1$ terms inside the parenthesis tend to zero and $s(x_1, \ldots, x_N) \to x_N$. □

Theorem A.2. *A parametrized soft-max function can be defined as the expression*

$$g(x_1, \ldots, x_N; \alpha) = \frac{1}{\alpha} \log \sum_{j=1}^{N} e^{\alpha x_j}$$

where $\alpha \geq 1$. *This function has the property that*

$$\lim_{\alpha \to \infty} g(x_1, \ldots, x_N; \alpha) = \max(x_1, \ldots, x_N) = x_N.$$

Proof. The function can be bounded as

$$\frac{1}{\alpha} \log e^{\alpha x_N} \leq g(x_1, \ldots, x_N; \alpha) \leq \frac{1}{\alpha} \log N e^{\alpha x_N},$$

$$x_N \leq g(x_1, \ldots, x_N; \alpha) \leq \frac{\log N}{\alpha} + x_N,$$

that for $\alpha \to \infty$ achieves the maximum x_N. Note that for $\alpha > 1$, from the bound, the soft-max always exceeds the maximum value, i.e., tends to x_N from the right. □

It is useful to look at the expression when $0 < \alpha < 1$.

Theorem A.3. *When* α *is a very small positive number*

$$g(x_1, \ldots, x_N; \alpha) \simeq \frac{1}{\alpha} \log N + \mu,$$

where $\mu = (1/N) \sum_{i=1}^{N} x_i$ *is the arithmetic mean. The function diverges for* $\alpha \to 0$, *but for small values of* α, *the function tends to become independent on any specific* x_i.

Proof. The function can be written as

$$\begin{aligned} g(x_1, \ldots, x_N; \alpha) &= \frac{1}{\alpha} \log \left(\sum_{i=1}^{N} e^{\alpha(x_i - \mu)} e^{\alpha \mu} \right) \\ &= \frac{1}{\alpha} \log \sum_{i=1}^{N} e^{\alpha(x_i - \mu)} + \mu. \end{aligned}$$

When α approaches zero, the exponents become $\simeq 1$ and we have the result. □

Theorem A.4. *Another parametric soft-max function is*

$$h(x_1, \ldots, x_N; \alpha) = \left(\sum_{j=1}^{N} x_j^{\alpha} \right)^{\frac{1}{\alpha}},$$

where here $x_i \geq 0, i = 1 : N$. *Here too, for* $\alpha \to \infty$, $h(x_1, \ldots, x_N; \alpha) \to x_N$.

Proof. From the bounds

$$\begin{aligned} (x_N^{\alpha})^{\frac{1}{\alpha}} \leq h(x_1, \ldots, x_N; \alpha) \leq (N x_N^{\alpha})^{\frac{1}{\alpha}}, \\ x_N \leq h(x_1, \ldots, x_N; \alpha) \leq N^{\frac{1}{\alpha}} x_N, \end{aligned}$$

as α increases $N^{\frac{1}{\alpha}} \to 1$ and the function tends to x_N. □

Theorem A.5. *The function* $h(x_1, \ldots, x_N; \alpha)$, *just as* $g(x_1, \ldots, x_N; \alpha)$, *diverges for* $\alpha \to 0$, *but for small* α

$$h(x_1, \ldots, x_N; \alpha) \simeq N^{\frac{1}{\alpha}},$$

which, again as in g, doas not depend on any of the x_i.

Proof. easily seen as $x_i^{\alpha} \simeq 1$ for small α. □

Theorem A.6. *Another soft-max functions can be defined as*

$$r(x_1, \ldots, x_N; \alpha) = \frac{\sum_{i=1}^{N} x_i e^{\alpha x_i}}{\sum_{j=1}^{N} e^{\alpha x_j}}.$$

This function is well-known in the neural network literature, where the vector function $e^{\alpha x_i} / \sum_j e^{\alpha x_j}$ *tends to a distribution concentrated on the maximum. By taking the expectation with such a distribution, we get the soft-max. Therefore, when* $\alpha \to \infty$, $r(x_1, \ldots, x_N; \alpha) \to x_N$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

Proof. The function can re-written using the ranked set as

$$\begin{aligned} r(x_1, \dots, x_N; \alpha) &= \frac{\sum_{i=1}^{N-1} x_i e^{\alpha x_j} + x_N e^{\alpha x_N}}{\sum_{j=1}^{N-1} e^{\alpha x_j} + e^{\alpha x_N}} \\ &= \frac{\sum_{i=1}^{N-1} x_i e^{\alpha (x_j - x_N)} + x_N}{\sum_{j=1}^{N-1} e^{\alpha (x_j - x_N)} + 1}. \end{aligned}$$

For $\alpha \to \infty$ both summations tend to zero, because the exponents are negative, and we have the result. □

Theorem A.7. *This soft-max function, when* $\alpha \to 0^+$ *does not diverge, but tends to the arithmetic mean* $r(x_1, \dots, x_N; \alpha) \to 1/N \sum_{i=1}^{N} x_i$.

Proof. Trivial, because for $\alpha = 0$ all the exponentials are equal to one. □

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

APPENDIX B OPTIMIZING REWARD AND ENTROPY

To better understand the nature of the function being optimized in (10), and how it gives rise to an entropy term, let us write it explicitely for $T = 4$, using the compact otation $R'(s_t a_t) = R(s_t a_t) + \log p(a_t)$. The function to optimize is

$$\begin{aligned}\sum_{s_1 \dots s_4} \sum_{a_1 \dots a_4} \; & p(s_1)\pi_\alpha(a_1|s_1)p(s_2|s_1a_1)\pi_\alpha(a_2|s_2)p(s_3|s_2a_2)\pi_\alpha(a_3|s_3)p(s_4|s_3a_3)\pi_\alpha(a_4|s_4) \\ & \left[R'(s_1a_1) - \frac{1}{\alpha}\log\pi_\alpha(a_1|s_1) + R'(s_2a_2) - \frac{1}{\alpha}\log\pi_\alpha(a_2|s_2) + R'(s_3a_3) \right. \\ & \left. - \frac{1}{\alpha}\log\pi_\alpha(a_3|s_3) + R'(s_4a_4) - \frac{1}{\alpha}\log\pi_\alpha(a_4|s_4) \right],\end{aligned} \tag{11}$$

Starting from the last term, in (11) we identify the backward recursions

$$\begin{aligned}& \underbrace{\frac{1}{\alpha}\sum_{s_4} p(s_4|s_3a_3) \underbrace{\left[\underbrace{\sum_{a_4}\pi_\alpha(a_4|s_4)\alpha R'(s_4a_4)}_{} + \overbrace{\sum_{a_4}\pi_\alpha(a_4|s_4)\log\frac{1}{\pi_\alpha(a_4|s_4)}}^{\mathcal{H}(\pi_\alpha(a_4|s_4))} \right]}_{V(s_4)}}_{Q(s_3a_3)} \\ & \underbrace{\frac{1}{\alpha}\sum_{s_3} p(s_3|s_2a_2) \underbrace{\left[\underbrace{\sum_{a_3}\pi_\alpha(a_3|s_3)(\alpha R'(s_3a_3) + \alpha Q(s_3a_3))}_{} + \overbrace{\sum_{a_3}\pi_\alpha(a_3|s_3)\log\frac{1}{\pi_\alpha(a_3|s_3)}}^{\mathcal{H}(\pi_\alpha(a_3|s_3))} \right]}_{V(s_3)}}_{Q(s_2a_2)} \\ & \underbrace{\frac{1}{\alpha}\sum_{s_2} p(s_2|s_1a_1) \underbrace{\left[\underbrace{\sum_{a_2}\pi_\alpha(a_2|s_2)(\alpha R'(s_2a_2) + \alpha Q(s_2a_2))}_{} + \overbrace{\sum_{a_2}\pi_\alpha(a_2|s_2)\log\frac{1}{\pi_\alpha(a_2|s_2)}}^{\mathcal{H}(\pi_\alpha(a_2|s_2))} \right]}_{V(s_2)}}_{Q(s_1a_1)} \\ & \frac{1}{\alpha}\sum_{s_1} p(s_1) \underbrace{\left[\underbrace{\sum_{a_1}\pi_\alpha(a_1|s_1)(\alpha R'(s_1a_1) + \alpha Q(s_1a_1))}_{} + \overbrace{\sum_{a_1}\pi(a_1|s_1)\log\frac{1}{\pi_\alpha(a_1|s_1)}}^{\mathcal{H}(\pi_\alpha(a_1|s_1))} \right]}_{V(s_1)}.\end{aligned} \tag{12}$$

Note how the value function $V(s_t)$ (not optimized here) is written as a recursive superposition of reward and policy entropy. The parameter α controls the balance between the two terms and the power of the distribution. Note that the policy function multiplies also the reward term. Therefore, the optimized policy distribution will shape, in a non trivial way, the effects of the rewards with respect to the entropy.

Following the approach in Levine [3], using our modified cost function, in (11) we search for the best policy distribution starting from re-writing the last term using the KL-divergence

$$\begin{aligned}& \tfrac{1}{\alpha}\textstyle\sum_{s_4} p(s_4|s_3a_3)\left[\sum_{a_4}\pi_\alpha(a_4|s_4)\left(\alpha R'(s_4a_4) - \log\pi_\alpha(a_4|s_4)\right)\right] = \\ & \tfrac{1}{\alpha}\textstyle\sum_{s_4} p(s_4|s_3a_3)\left[\sum_{a_4}\pi_\alpha(a_4|s_4)\left(\log\frac{e^{\alpha R'(s_4a_4)}}{\pi_\alpha(a_4|s_4)}\frac{\sum_{a'_4}e^{\alpha R'(s_4a'_4)}}{\sum_{a'_4}e^{\alpha R'(s_4a'_4)}}\right)\right] = \\ & \tfrac{1}{\alpha}\textstyle\sum_{s_4} p(s_4|s_3a_3)\left[-\mathcal{D}_{KL}\left(\pi_\alpha(a_4|s_4)\left\|\frac{e^{\alpha R'(s_4a_4)}}{\sum_{a'_4}e^{\alpha R'(s_4a'_4)}}\right.\right) + \log\underbrace{\sum_{a'_4}e^{\alpha R'(s_4a'_4)}}_{e^{\alpha V(s_4)}}\right] = \\ & \textstyle\sum_{s_4} p(s_4|s_3a_3)\left[-\mathcal{D}_{KL}\left(\pi_\alpha(a_4|s_4)\left\|\frac{e^{\alpha R'(s_4a_4)}}{e^{\alpha V(s_4)}}\right.\right) + V(s_4)\right].\end{aligned} \tag{13}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3148127, IEEE Access

The optimum value is obtained when the $\mathcal{D}_{KL}(.\|.) = 0$, i.e., when $\pi_\alpha(a_4|s_4) = \frac{e^{\alpha R'(s_4a_4)}}{e^{\alpha V(s_4)}}$, and the optimal policy distribution is

$$\pi^*(a_4|s_4) \propto \frac{e^{R'(s_4a_4)}}{e^{V(s_4)}} = \frac{e^{Q(s_4a_4)}}{e^{V(s_4)}},$$

where we have defined $Q(s_4a_4) = R'(s_4a_4)$. Now the optimized expression $\sum_{s_4} p(s_4|s_3a_3)V(s_4)$ is carried over

$$\underbrace{R'(s_3a_3) + \sum_{s_4} p(s_4|s_3a_3)V(s_4)}_{Q(s_3a_3)} - \frac{1}{\alpha}\log \pi_\alpha(a_3|s_3).$$

Taking the expectation, we obtain (14)

$$\begin{aligned}
&\tfrac{1}{\alpha}\textstyle\sum_{s_3} p(s_3|s_2a_2)\left[\sum_{a_3}\pi_\alpha(a_3|s_3)\left(\alpha Q(s_3a_3) - \log\pi_\alpha(a_3|s_3)\right)\right] = \\
&\tfrac{1}{\alpha}\textstyle\sum_{s_3} p(s_3|s_2a_2)\left[\sum_{a_3}\pi_\alpha(a_3|s_3)\left(\log\frac{e^{\alpha Q(s_3a_3)}}{\pi_\alpha(a_3|s_3)}\frac{\sum_{a'_3}e^{\alpha Q(s_3a'_3)}}{\sum_{a'_3}e^{\alpha Q(s_3a'_3)}}\right)\right] = \\
&\tfrac{1}{\alpha}\textstyle\sum_{s_3} p(s_3|s_2a_2)\left[-\mathcal{D}_{KL}\left(\pi_\alpha(a_3|s_3)\left\|\frac{e^{\alpha Q(s_3a_3)}}{\sum_{a'_3}e^{\alpha Q(s_3a'_3)}}\right.\right) + \log\underbrace{\sum_{a'_3}e^{\alpha Q(s_3a'_3)}}_{e^{\alpha V(s_3)}}\right] = \\
&\tfrac{1}{\alpha}\textstyle\sum_{s_3} p(s_3|s_2a_2)\left[-\mathcal{D}_{KL}\left(\pi_\alpha(a_3|s_3)\left\|\frac{e^{\alpha Q(s_3a_3)}}{e^{\alpha V(s_3)}}\right.\right) + \alpha V(s_3)\right].
\end{aligned} \tag{14}$$

where $\mathcal{D}_{KL} = 0$ when $\pi_\alpha(a_3|s_3) = e^{\alpha Q(s_3a_3)}/e^{\alpha V(s_4)}$. The best policy distribution is then $\pi^*(a_3|s_3) \propto e^{Q(s_3a_3)}/e^{V(s_4)}$. Carrying over $\sum_{s_3} p(s_3|s_2a_2)V(s_3)$, we have

$$\underbrace{R'(s_2a_2) + \sum_{s_3} p(s_3|s_2a_2)V(s_3)}_{Q(s_2a_2)} - \frac{1}{\alpha}\log \pi_\alpha(a_2|s_3).$$

Following similar steps, we have $\pi^*(a_2|s_2) \propto \frac{e^{Q(s_2a_2)}}{e^{V(s_2)}}$ and

$$\underbrace{R'(s_1a_1) + \sum_{s_2} p(s_2|s_1a_1)V(s_2)}_{Q(s_1a_1)} - \frac{1}{\alpha}\log \pi_\alpha(a_1|s_1).$$

The last step is

$$\begin{aligned}
&\tfrac{1}{\alpha}\textstyle\sum_{s_1} p(s_1)\left[\sum_{a_1}\pi_\alpha(a_1|s_1)\left(\alpha Q(s_1a_1) - \log\pi_\alpha(a_1|s_1)\right)\right] = \\
&\tfrac{1}{\alpha}\textstyle\sum_{s_1} p(s_1)\left[\sum_{a_1}\pi_\alpha(a_1|s_1)\left(\log\frac{e^{\alpha Q(s_1a_1)}}{\pi_\alpha(a_1|s_1)}\frac{\sum_{a'_1}e^{\alpha Q(s_1a'_1)}}{\sum_{a'_1}e^{\alpha Q(s_1a'_1)}}\right)\right] = \\
&\tfrac{1}{\alpha}\textstyle\sum_{s_1} p(s_1)\left[-\mathcal{D}_{KL}\left(\pi_\alpha(a_1|s_1)\left\|\frac{e^{\alpha Q(s_1a_1)}}{\sum_{a'_1}e^{\alpha Q(s_1a'_1)}}\right.\right) + \log\underbrace{\sum_{a'_1}e^{\alpha Q(s_1a'_1)}}_{e^{\alpha V(s_1)}}\right] = \\
&\tfrac{1}{\alpha}\textstyle\sum_{s_1} p(s_1)\left[-\mathcal{D}_{KL}\left(\pi_\alpha(a_1|s_1)\left\|\frac{e^{\alpha Q(s_1a_1)}}{e^{\alpha V(s_1)}}\right.\right) + \alpha V(s_1)\right],
\end{aligned} \tag{15}$$

which is minimized when $\pi_\alpha(a_1|s_1) = e^{\alpha Q(s_1a_1)}/e^{\alpha V(s_1)}$, with the optimal policy distribution $\pi^*(a_1|s_1) \propto e^{Q(s_1a_1)}/e^{V(s_1)}$.Therefore, the recursions at a generic time step t are

$$\begin{aligned}
Q_{(S_tA_t)^1}(s_ta_t) &= \log p(a_t) + R(s_ta_t) \\
&\quad + \sum_{s_{t+1}} p(s_{t+1}|s_ta_t)V_{S_{t+1}}(s_{t+1}), \\
V_{S_t}(s_t) &= \frac{1}{\alpha}\log\sum_{a_t} e^{\alpha Q_{(S_tA_t)^1}(s_ta_t)},
\end{aligned}$$

with the optimal policy distribution:

$$\pi^*(a_t|s_t) \propto e^{Q(s_ta_t) - V(s_t)}.$$

REFERENCES

[1] M. Toussaint and A. Storkey, "Probabilistic inference for solving discrete and continuous state markov decision processes," *Proceedings of the 23rd International Conference on Machine Learning*, vol. 2006, pp. 945–952, 01 2006.

[2] M. Toussaint, "Probabilistic inference as a model of planned behavior," *Kunstliche Intelligenz*, vol. 3, 01 2009.

[3] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," *arXiv:1805.00909v3 [cs.LG] 20 May 2018*, 2018.

[4] E. Todorov, "General duality between optimal control and estimation," in *Proceedings of the47th IEEE Conference on Decision and Control, Cancun, Mexico, Dec. 9-11*, 2008.

[5] H. J. Kappen, V. Gomez, and M. Manfred, "Optimal control as a graphical model inference problem," *Machine Learning*, vol. 87, p. 159–182, 2012.

[6] T. Verbelen, P. Lanillos, and C. Buckley, Christopher L. De Boom, Eds., *Abtive Inference, First International Workshop, IWAI 2020Co-located with ECML/PKDD 2020Ghent, Belgium, September 14*. Springer, 2020.

[7] C. L. Buckley, C. S. Kim, S. McGregor, and A. K. Seth, "The free energy principle for action and perception: a mathematica review," *Journal of Mathematica Phychology*, vol. 81, pp. 55–79, 2017.

[8] T. Parr and K. J. Friston, "Generalised free energy and active inference: can the future cause the past?" *BioARXIV*, 2018.

[9] Z. Alexandre, S. Oleg, and P. Giovanni, "An information-theoretic perspective on the costs of cognition," *Neuropsychologia*, vol. 123, pp. 5–18, 2018.

[10] T. Parr and K. J. Friston, "The anatomy of inference: Generative models and brain structure," *Frontiers in Computational Neuroscience*, vol. 12, 2018.

[11] R. Kaplan and K. J. Friston, "Planning and navigation as active inference," *Biological Cybernetics*, vol. 112, pp. 323–347, 2018.

[12] B. D. Ziebart, A. Bagnell, and A. K. Dey, "Modeling interaction via the principle of maximum causal entropy," in *Proceedings of the 27th International Conference on Machine Learning (ICML), Haifa, Israel*, 2010.

[13] M. Baltieri and C. L. Buckley, "An active inference implementation of phototaxis," *ARXIV*, 2017.

[14] S. Nair, Y. Zhu, S. Savarese, and F.-F. Li, "Causal induction from visual observations for goal directed tasks," *arXiv:1910.01751v1 [cs.LG] 3 Oct 2019*, 2019.

[15] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2006.

[16] D. Bertsekas, *Reinforcement Learning and Optimal Control*. Athena Scientific, 2019.

[17] H. Attias, "Planning by probabilistic inference," in *Proc. of the 9th Int. Workshop on Artificial Intelligence and Statistics*, 2003, p. .

[18] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[19] J. Forney, G.D., "Codes on graphs: normal realizations," *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 520–548, 2001.

[20] H. A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Magazine*, vol. 21, no. 1, pp. 28 – 41, jan. 2004.

[21] F. A. N. Palmieri, "A comparison of algorithms for learning hidden variables in bayesian factor graphs in reduced normal form," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 11, pp. 2242–2255, Nov 2016.

[22] G. Di Gennaro, A. Buonanno, and F. A. N. Palmieri, "Optimized realization of bayesian networks in reduced normal form using latent variable model," *Soft Computing, Springer*, pp. 1–12, Mar. 2021.

[23] H.-A. . Loeliger, J. Dauwels, J. Hu, S. Korl, P. Li, and F. Kschischang, "The factor graph approach to model-based signal processing," *Proceedings of the IEEE*, vol. 95, no. 6, pp. 1295 –1322, june 2007.

[24] F. Castaldo and F. Palmieri, "Target tracking using factor graphs and multi-camera systems," *IEEE Transactions on Aerospace and Electronic Systems (TAES)*, vol. 51, no. 3, pp. 1950 – 1960, 2015.

[25] F. Castaldo, F. Palmieri, and C. Regazzoni, *Application of Bayesian Techniques to Behavior Analysis in Maritime Environments*. Springer, Cham, 2014, vol. 37, ch. 5, pp. 175–183.

[26] P. Coscia, F. Castaldo, F. A. N. Palmieri, L. Ballan, A. Alahi, and S. Savarese, "Point-based path prediction from polar histograms," in *Proceedings of the 19th International Conference on Information Fusion (FUSION 2016)*, 2016, pp. 1961–1967.

[27] P. Coscia, F. A. N. Palmieri, P. Braca, L. M. Millefiori, and P. Willett, "Unsupervised maritime traffic graph learning with mean-reverting stochastic processes," in *2018 21st International Conference on Information Fusion (FUSION)*, 2018, pp. 1822–1828.

[28] P. Coscia, P. Braca, L. M. Millefiori, F. A. N. Palmieri, and P. K. Willett, "Multiple ornstein-uhlenbeck processes for maritime traffic graph representation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 54, pp. 2158–2170, 2018.

[29] P. Coscia, F. Castaldo, F. A. N. Palmieri, A. Alahi, S. Savarese, and L. Ballan, "Long-term path prediction in urban scenarios using circular distributions," *Image and Vision Computing*, vol. 69, pp. 81–91, 2018.

[30] F. A. N. Palmieri, K. R. Pattipati, G. Fioretti, G. Di Gennaro, and A. Buonanno, "Path planning using probability tensor flows," *IEEE Aerospace and Electronic Systems Magazine*, vol. 36, no. 1, Jan. 2021, preliminary version on arXiv:2003.02774.

[31] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction (second edition)*. MIT Press, 2018.

[32] D. Barber, *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[33] B. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. Bagnell, M. Hebert, A. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," 12 2009, pp. 3931–3936.

[34] B. Millidge, A. Tschantz, A. K. Seth, and C. L. Buckley, "On the relationship between activeinference and control as inference," in *Proceedings of First International Workshop, IWAI 2020, Co-located with ECML/PKDD 2020, Ghent, Belgium, September 14*, 2020.

[35] A. Imohiosen, J. Watson, and J. Peters, "Active inference or control as inference?a unifying view," in *Proceedings of First International Workshop, IWAI 2020, Co-located with ECML/PKDD 2020, Ghent, Belgium, September 14*, 2020.

[36] H. Kappen, V. Gomez, and M. Opper, "Optimal control as a graphical model inference problem," in *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling*, 2013.

[37] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. IT-20, no. 2, pp. 284–287, Mar. 1974.