http://pubsonline.informs.org/journal/moor

MATHEMATICS OF OPERATIONS RESEARCH
Vol. 45, No. 1, February 2020, pp. 34–62
ISSN 0364-765X (print), ISSN 1526-5471 (online)

# Polynomial Time Algorithms for Branching Markov Decision Processes and Probabilistic Min(Max) Polynomial Bellman Equations

**Kousha Etessami,[a] Alistair Stewart,[b] Mihalis Yannakakis[c]**

[a] School of Informatics, University of Edinburgh, Edinburgh EH8 9AB, United Kingdom; [b] Department of Computer Science, University of Southern California, Los Angeles, California 90089; [c] Department of Computer Science, Columbia University, New York, New York 10027

**Contact:** kousha@inf.ed.ac.uk, https://orcid.org/0000-0001-5700-3462 (KE); stewart.al@gmail.com (AS); mihalis@cs.columbia.edu (MY)

**Received:** January 22, 2016
**Revised:** December 8, 2017; June 18, 2018
**Accepted:** July 22, 2018
**Published Online in Articles in Advance:** December 5, 2019

**MSC2000 Subject Classification:** Primary: 90C40, 60J80, 68Q25, 90C53
**OR/MS Subject Classification:** Primary: analysis of algorithms, programming; secondary: dynamic programming, Markov

**https://doi.org/10.1287/moor.2018.0970**



**Abstract.** We show that one can compute the least nonnegative solution (also known as the *least fixed point*) for a system of probabilistic min (max) polynomial equations, to any desired accuracy $\epsilon > 0$ in time polynomial in both the encoding size of the system and in $\log(1/\epsilon)$. These are Bellman optimality equations for important classes of *infinite-state* Markov decision processes (MDPs), including branching MDPs (BMDPs), which generalize classic multitype branching stochastic processes. We thus obtain the first polynomial time algorithm for computing, to any desired precision, optimal (maximum and minimum) *extinction probabilities* for BMDPs. Our algorithms are based on a novel generalization of Newton's method, which employs linear programming in each iteration. We also provide polynomial-time (P-time) algorithms for computing an $\epsilon$-optimal policy for both maximizing and minimizing extinction probabilities in a BMDP, whereas we note a hardness result for computing an exact optimal policy. Furthermore, improving on prior results, we provide more efficient P-time algorithms for qualitative analysis of BMDPs, that is, for determining whether the maximum or minimum extinction probability is 1, and, if so, computing a policy that achieves this. We also observe some complexity consequences of our results for branching simple stochastic games, which generalize BMDPs.

**Funding:** This research was partially supported by the Royal Society and the National Science Foundation [Grants CCF-1320654, CCF-1703925, and CCF-1763970].
**Supplemental Material:** The online companion is available at https://doi.org/10.1287/moor.2018.0970.

**Keywords:** multitype branching processes • Markov decision processes • Bellman optimality equations • generalized Newton method • polynomial time algorithms

## 1. Introduction

Markov decision processes (MDPs) are fundamental models for stochastic dynamic optimization, with applications in many fields (see, e.g., Feinberg and Shwartz [18], Puterman [25]). They extend purely stochastic processes (Markov chains) with a controller (an agent) who can partially affect the evolution of the process and seeks to optimize some objective. For many important classes of MDPs, the task of computing the *optimal value* of the objective, starting at any state of the MDP, can be rephrased as the problem of solving the associated *Bellman optimality equations* for that MDP model. In particular, for finite-state MDPs where, for example, the objective is to maximize (or minimize) the probability of eventually reaching some target state, the associated Bellman equations are *max-linear* (*min-linear*) equations, and we know how to solve such equations in P-time using linear programming (see, e.g., Puterman [25]). The same holds for a number of other classes of finite-state MDPs.[1]

In many important settings, however, the state space of the processes of interest, both for purely stochastic processes and for controlled ones (MDPs), is not finite, even though the processes can be specified in a finite way. For example, consider multitype branching processes (BPs; Harris [20], Kolmogorov and Sevastyanov [23]), classic probabilistic models with applications in many areas. A BP models the stochastic evolution of a population of entities of distinct types. In each generation, every entity of each type $T$ produces a set of entities of various types in the next generation according to a given probability distribution on offspring for the type $T$. In a *branching Markov decision process* (*BMDP*; e.g., Pliska [24], Rothblum and Whittle [26]), there is a controller who can take actions that affect the probability distribution for the sets of offspring for each entity of each type.

Branching processes have been used to model phenomena in many fields, including biology (see, e.g., Kimmel and Axelrod [22]), population genetics (Haccou et al. [19]), physics and chemistry (e.g., particle

systems, nuclear chain reactions), medicine (e.g., cancer growth), marketing, and others. In many cases, the process is not purely stochastic, but there is the possibility of taking actions (e.g., adjusting the conditions of reactions, applying drug treatments in medicine, advertising in marketing, etc.) that can influence the probabilistic evolution of the process to bias it toward achieving desirable objectives. Branching Markov decision processes are useful models in such settings. For both BPs and BMDPs, the state space consists of all possible populations, given by the number of entities of the various types, so there is an infinite number of states. From the computational point of view, the usefulness of such infinite-state models hinges on whether their analysis remains tractable.

In recent years, there has been a body of research aimed at studying the computational complexity of key analysis problems associated with MDP extensions (and more general stochastic game extensions) of important classes of finitely presented but *countably infinite-state* stochastic processes, including controlled extensions of classic multitype branching processes (i.e., BMDPs), *stochastic context-free grammars*, and discrete-time *quasi-birth–death processes*. In Etessami and Yannakakis [11], the *recursive Markov decision process* (*RMDP*) model was studied, which is in a precise sense more general than all of these, and forms the MDP extension of *recursive Markov chains* (*RMCs*; Etessami and Yannakakis [10]; and equivalently, *probabilistic pushdown systems*; Esparza et al. [8]), or it can be viewed alternatively as the extension of finite-state MDPs with recursion. RMDPs consist of a set of MDPs that can call each other recursively, in the same way as recursive procedures do. These models arise in various areas, for example, performance evaluation, computational biology, and program analysis and verification.

A central analysis problem for all of these models, which forms the key to a number of other analyses, is the problem of computing their *optimal termination* (*extinction*) *probability*. For example, in the setting of multitype branching MDPs, these key quantities are the maximum (minimum) probabilities, over all control strategies (or policies), that starting from a single entity of a given type, the process will eventually reach extinction (i.e., the state where no entities have survived). From these quantities, one can compute the optimum probability for any initial population, as well as other quantities of interest.

One can indeed form Bellman optimality equations for the optimal extinction probabilities of BMDPs, and for a number of related important infinite-state MDP models. However, it turns out that these optimality equations are no longer max/min *linear*, but rather are max/min *polynomial* equations (Etessami and Yannakakis [11]). Specifically, the Bellman equations for BMDPs with the objective of maximizing (or minimizing) extinction probability are multivariate monotone max (or min) probabilistic polynomial systems of equations, which we call max/minPPSs, of the form $x_i = P_i(x_1, \ldots, x_n)$, $i = 1, \ldots, n$, where each $P_i(x) \equiv \max_j q_{i,j}(x)$ (respectively, $P_i(x) \equiv \min_j q_{i,j}(x)$) is the max (min) over a finite number of probabilistic polynomials, $q_{i,j}(x)$. A *probabilistic polynomial*, $q(x)$, is a multivariate polynomial where the monomial coefficients and constant term of $q(x)$ are all nonnegative and sum to $\leq 1$. We write these equations in vector form as $x = P(x)$. Then $P(x)$ defines a mapping $P : [0,1]^n \to [0,1]^n$ that is monotone, and thus (by Tarski's theorem) has a *least fixed point* (*LFP*) in $[0,1]^n$. The equations $x = P(x)$ can have more than one solution, but it turns out that the optimal value vector for extinction probabilities in the corresponding BMDP is precisely the LFP solution vector $q^* \in [0,1]^n$, that is, the (coordinate-wise) least nonnegative solution (Etessami and Yannakakis [11]). Intuitively, the reason that the optimal extinction probabilities are given by the least fixed point (as opposed to any other fixed point) is the following. For any integer $t \geq 0$, let $p(t)$ denote the vector of optimal probabilities of extinction by time $t$. Clearly, $p(t)$ converges monotonically from below to the optimal extinction probabilities as $t$ goes to infinity, meaning $\lim_{t\to\infty} p(t)$ is the vector of optimal extinction probabilities. For a max/minPPS $x = P(x)$, and for an integer $t \geq 0$, define the vector $P^t(0) \in [0,1]^n$ inductively by $P^0(0) := 0$, and $P^{t+1}(0) := P(P^t(0))$. It is not hard to show that $P^t(0) = p(t)$ for all $t$. Because of the monotonicity of $P(\cdot)$, the LFP solution of $x = P(x)$ is given by $\lim_{t\to\infty} P^t(0)$. Thus, $\lim_{t\to\infty} P^t(0) = q^* = \lim_{t\to\infty} p(t)$, so the LFP is indeed the vector of optimal extinction probabilities.

The same class of equations (max/minPPS) also models other stochastic processes besides BMDPs, including controlled stochastic context-free grammars and one-exit RMDPs (1-RMDPs), that is, RMDPs where the component MDPs have one exit (terminating state). These models arise in various areas. For example, RMDPs are natural models for recursive probabilistic programs, where the component MDPs of the RMDP correspond to the procedures of the program. There has been an extensive body of work over many years that has developed the theory, algorithms, and tools for the analysis and verification of nonrecursive probabilistic programs, which are modeled abstractly by ordinary finite-state Markov decision processes. Extending the scope of the work to handle recursive probabilistic programs requires the analysis of RMDPs. A central problem for this is the termination problem, computing the worst-case (or best-case) probability of termination of the RMDP; this is essential for the analysis and verification of more complex temporal properties. The

termination probabilities of one-exit RMDPs can be captured by max/minPPS; that is, for every 1-RMDP, one can construct efficiently a maxPPS (or minPPS) whose LFP gives the maximum (or minimum) termination probability of the 1-RMDP.

Already for pure stochastic multitype BPs, the extinction probabilities may be irrational values. The problem of *deciding* whether the extinction probability of a BP is $\geq p$, for a given probability $p$, is in PSPACE (Etessami and Yannakakis [10]), and likewise, deciding whether the optimal extinction probability of a BMDP is $\geq p$ is in PSPACE (Etessami and Yannakakis [11]). These PSPACE upper bounds appeal to decision procedures for the existential theory of reals for solving the associated (max/min)PPS equations. However, already for BPs, it was shown in Etessami and Yannakakis [10] that this quantitative *decision* problem is already at least as hard as the square-root sum (Sqrt-Sum) problem, as well as a (much) harder and more fundamental problem called PosSLP, which captures the power of unit-cost exact rational arithmetic. It is a long-standing open problem whether either of these decision problems is in NP, or even in the polynomial time hierarchy (for more information on these problems, see Allender et al. [1], Etessami and Yannakakis [10]). Thus, such *quantitative decision problems* are unlikely to have P-time algorithms in the standard Turing model, even in the purely stochastic setting, so we can certainly not expect to find P-time algorithms for the extension of these models to the MDP setting.[2] On the other hand, it was shown in Etessami and Yannakakis [10, 11] that for both BPs and BMDPs, the *qualitative* decision problem of deciding whether the optimal extinction probability is $q_i^* = 0$ or whether $q_i^* = 1$ can be solved in polynomial time.

The hardness of the quantitative decision problem does not, however, rule out the possibility of efficiently approximating the optimal extinction probabilities to any desired precision. A simple approach for approximation is to apply *value iteration*: Starting with $x = 0$, repeatedly apply $P$ to compute $P(0), P^2(0), \ldots, P^k(0), \ldots$. As $k \to \infty$, $P^k(0)$ converges (monotonically) to the LFP $q^*$. However, the convergence can be very slow, even for some simple examples of pure branching processes; specifically, it can be double exponential both in the number of types and in the number of bits of precision. The extinction probabilities of pure BPs are the LFP of a system of probabilistic polynomial equations (PPS), without max or min. Consider the equation $x = 0.5x^2 + 0.5$, which corresponds to a simple BP with one type; the LFP is 1 but, as shown in Etessami and Yannakakis [10], value iteration needs $2^{k-3}$ iterations to approximate it with $k$ bits of precision. Furthermore, there are pure multitype BPs, based on "nesting" this example, namely, the system with $n+1$ variables given by $x_0 = 0.5x_0^2 + 0.5$ and $x_i = 0.5x_i^2 + 0.5x_{i-1}$, for $i = 1, \ldots, n$, where approximating the extinction probability within less than $1/2$ (i.e., getting one bit of precision) using value iteration, starting from the 0 vector, requires a double-exponential number of iterations in $n$, specifically, at least $2^{2^n-3}$ iterations (see Esparza et al. [7], Stewart et al. [27]); for example, if $n = 10$, the value of $x_{10}$ remains close to 0 (i.e., very far from the LFP value 1) even after $2^{1000}$ iterations. It is also known that for ordinary finite-state MDPs, value iteration (as well as policy iteration) requires in some cases an exponential number of iterations.

Despite decades of theoretical and practical work on computational problems like extinction relating to multitype branching processes, and equivalent termination problems related to stochastic context-free grammars, until recently it was not even known whether one could obtain *any* nontrivial *approximation* of the extinction probability of a purely stochastic multitype branching process in P-time. Etessami et al. [12, 15] provided the first polynomial time algorithm for computing (i.e., approximating) to within any desired additive error $\epsilon > 0$ the LFP of a given PPS, and hence the extinction probability vector $q^*$ for a given purely stochastic BP, in time polynomial in both the encoding size of the PPS (or the BP) and in $\log(1/\epsilon)$. The algorithm works in the standard Turing model of computation. Our algorithm was based on an approach using Newton's method that was first introduced and studied in Etessami and Yannakakis [10]. In Etessami and Yannakakis [10], the approach was studied for more general *monotone* polynomial systems of equations (MPSs), and it was subsequently further studied in Esparza et al. [7]. The algorithm of Etessami et al. [12, 15] for PPSs first identifies and removes the variables that have value 0 or 1 in the LFP and then applies Newton's method in the resulting system. (The removal of the variables with value 0 or 1 is critical for the correctness and efficiency of the algorithm.)

Note that unlike PPSs and MPSs, the min/maxPPSs that define the Bellman equations for BMDPs are no longer differentiable functions (they are only piecewise differentiable). Thus, a priori, it is not even clear how one could apply a Newton-type method toward solving them.

### 1.1. Our Results

1. In this paper, we provide the first polynomial time algorithms for approximating the LFPs of both maxPPSs and minPPSs, and thus the first polynomial time algorithm for computing (to within any desired additive

error) the optimal value vector for BMDPs with the objective of maximizing or minimizing their extinction probability.

Our approach is based on a *generalized Newton method (GNM)* that extends Newton's method in a natural way to the (nondifferentiable) setting of max/minPPSs. The method is again iterative where each iteration involves the computation of the least (greatest) solution of a max-linear (min-linear) system of equations, both of which we show can be solved using linear programming. Our algorithms based on the GNM have the nice feature that they are relatively simple, although the analysis of their correctness and time complexity is rather involved. We show that if we first identify and remove the variables that have value 0 or 1 in the LFP and then apply the GNM with a suitable rounding of the computed points along the way, the algorithm computes the LFP of a maxPPS or minPPS within any desired number $j$ of bits of precision (i.e., within additive error $2^{-j}$) in polynomial time in the encoding size of the system and the number $j$ of bits of precision. We note that the two cases of maxPPS and minPPS are not symmetric, and separate analysis is required for them (and this holds also for the other results below). The reason for the asymmetry is that we seek a specific fixed point, the least one, and this behaves differently with respect to max and min.

2. We furthermore show that we can compute $\epsilon$-optimal (pure) strategies (policies) for both maxPPSs and minPPSs (and max/min BMDPs), for *any* given desired $\epsilon > 0$, in time polynomial in both the encoding size of the max/minPPS and in $\log(1/\epsilon)$. This result is, at first glance, rather surprising because there are only a bounded number of distinct pure policies for a max/minPPS, and computing an optimal policy is PosSLP-hard, as we show; thus, it is very unlikely that an optimal policy can be computed in P-time (Etessami et al. [12]).

3. We provide new algorithms for the *qualitative* analysis of max/minPPSs and BMDPs (i.e., identifying the variables that have value 0 or 1 in the LFP), which improve significantly on the running time of the previous P-time qualitative algorithms given in Etessami and Yannakakis [11]. This is important for the practical efficiency of our quantitative algorithms for the approximation of the LFP, which make crucial use of a preprocessing step that identifies and removes the variables with value 0 or 1 in the LFP. Polynomial time algorithms for the qualitative analysis were first established in Etessami and Yannakakis [11], but the running time was rather high, especially in the case of maxPPSs, which involved the solution of linear programs (LPs) with a cubic number of variables. This is improved substantially in our new qualitative algorithms, which solve LPs with a linear number of variables and constraints. (These improved qualitative results are provided in the online companion to this paper.)

4. Finally, we consider *branching simple stochastic games* (*BSSGs*), which are two-player, turn-based stochastic games, where one player wants to maximize, and the other wants to minimize, the extinction probability. The *value* of these games (which are determined) is characterized by the LFP solution of associated min-maxPPSs, which combine both min and max operators (see Etessami and Yannakakis [11]). We observe that our results easily imply a TFNP (Total Function NP) upper bound for $\epsilon$-approximating the *value* of BSSGs and computing $\epsilon$-optimal strategies for them.

### 1.2. Related Work

We have already mentioned some of the important relevant results. BMDPs and related processes have been studied previously in both the operations research (e.g., Denardo and Rothblum [6], Pliska [24], Rothblum and Whittle [26]) and computer science literature (e.g., Brázdil et al. [3], Esparza et al. [7], Etessami and Yannakakis [11]), but no efficient algorithms were known for the (approximate) computation of the relevant optimal probabilities and policies; the best known upper bound was PSPACE (Etessami and Yannakakis [11]).

In Etessami and Yannakakis [11], we introduced recursive Markov decision processes, a recursive extension of MDPs. We showed that for general RMDPs, the problem of computing the optimal termination probabilities, even within any nontrivial approximation, is undecidable. However, we showed for the important class of one-exit RMDPs, the optimal probabilities can be expressed by minPPSs (or maxPPSs), and in fact, the problems of computing (approximately) the LFP of a min/maxPPS and the termination probabilities of a max/min 1-RMDP, or BMDP, are all polynomially equivalent. We furthermore showed in Etessami and Yannakakis [11] that there are always pure, memoryless (and "stackless") optimal policies for both maximizing and minimizing termination probability for 1-RMDPs, and likewise pure memoryless "static" optimal policies for extinction probabilities of BMDPs, as well as for the more general turn-based simple stochastic games (one-exit recursive simple stochastic games (1-RSSGs) and BSSGs), which generalize 1-RMDPs and BMDPs.

In Etessami et al. [16], 1-RMDPs (and 1-RSSGs) with a different objective were studied, namely, optimizing the total expected reward in a setting with positive rewards. In that setting, things are much simpler: the

Bellman equations turn out to be max/min linear, the optimal values are rational, and they can be computed *exactly* in P-time using linear programming.

A work that is more closely related to this paper is Esparza et al. [9]. They studied more general monotone min-maxMPSs, that is, systems of monotone polynomial equations that include both min and max operators, and they presented two different iterative analogs of Newton's methods for approximating the LFP of a min-maxMPS, $x = P(x)$. Their methods are related to ours, but differ in key respects. Both of their methods use certain piecewise linear functions to approximate the min-maxMPS in each iteration, which is also what one does to solve each iteration of our generalized Newton method. However, the precise nature of their piecewise linearizations and how they solve them differ in important ways from ours, even when they are applied in the specific context of maxPPSs or minPPSs. They showed, working in the unit-cost *exact* arithmetic model, that, using their methods, one can compute $j$ "valid bits" of the LFP (i.e., compute the LFP within relative error at most $2^{-j}$) in $k_P + c_P \cdot j$ iterations, where $k_P$ and $c_P$ are terms that depend in *some* way on the input system, $x = P(x)$. However, they give no upper bounds on $k_P$, and their upper bounds on $c_P$ are exponential in the number $n$ of variables of $x = P(x)$. Note that MPSs are more difficult to solve: even without the min and max operators, we know that it is PosSLP-hard to approximate their LFPs within any nontrivial constant additive error $c < 1/2$, even for pure MPSs that arise from recursive Markov chains (Etessami and Yannakakis [10]).

Another subclass of RMDPs, called *one-counter MDPs* (a controlled extension of one-counter Markov chains and quasi-birth–death processes; Etessami et al. [17]) has been studied, and the approximation of their optimal termination probabilities was recently shown to be computable, but only in *exponential time* (Brázdil et al. [2]). This subclass is incomparable with 1-RMDPs and BMDPs, and does not have min/maxPPSs as Bellman equations.

### 1.3. Organization of This Paper

Section 2 gives formal definitions and background on branching Markov decision processes and max and min probabilistic polynomial systems. In Section 3, we define the generalized Newton method for a maxPPS and minPPS, we analyze the method, and we show how to compute the LFP of a maxPPS or a minPPS to desired precision in polynomial time in the encoding size of the system and the desired number of bits of precision. In Section 4, we observe that computing an optimal policy is PosSLP-hard (thus, probably intractable) and show how to compute an $\epsilon$-optimal policy (for any given $\epsilon > 0$) of a maxPPS or minPPS in polynomial time in the size of the system and $\log(1/\epsilon)$. In Section 5, we show that the approximate computation problems of the value of, and $\epsilon$-optimal strategies for, BSSGs are in TFNP. Our improved polynomial time algorithms for the qualitative analysis of maxPPSs and minPPSs (and BMDPs) are presented in the online companion to this paper. Proofs for some of the supporting lemmas from the main paper are deferred to the appendix of this paper.

## 2. Definitions and Background

Throughout this paper, it will be convenient to compare a vector or matrix to the all 0, or all 1, vector/matrix. For a given vector/matrix $z$, we will use the notation $z \geq 0$, $z < 1, \ldots$, to denote that every entry of $z$ is, respectively, $\geq 0$, $<1$, .... The $l_\infty$ norm of a vector $z$ is $\|z\|_\infty := \max_i |z_i|$, and its associated matrix norm $\|A\|_\infty$ is the maximum absolute-value row sum of $A$, that is, $\|A\|_\infty := \max_i \sum_j |A_{i,j}|$.

For an $n$-vector of variables $x = (x_1, \ldots, x_n)$ and a vector $v \in \mathbb{N}^n$, we use the shorthand notation $x^v$ to denote the monomial $x_1^{v_1}, \ldots, x_n^{v_n}$. Let $\langle \alpha_r \in \mathbb{N}^n \mid r \in R \rangle$ be a multiset of $n$-vectors of natural numbers, indexed by the set $R$. Consider a multivariate polynomial $P_i(x) = \sum_{r \in R} p_r x^{\alpha_r}$, for some rational-valued coefficients $p_r$, $r \in R$. We shall call $P_i(x)$ a *monotone polynomial* if $p_r \geq 0$ for all $r \in R$. If in addition, we also have $\sum_{r \in R} p_r \leq 1$, then we shall call $P_i(x)$ a *probabilistic polynomial*.

**Definition 1.** A *probabilistic* (respectively, *monotone*) *polynomial system of equations*, $x = P(x)$, which we shall call a *PPS* (respectively, an *MPS*), is a system of $n$ equations, $x_i = P_i(x)$, in $n$ variables $x = (x_1, x_2, \ldots, x_n)$, where for all $i \in \{1, 2, \ldots n\}$, $P_i(x)$ is a probabilistic (respectively, monotone) polynomial.

A *maximum-minimum probabilistic polynomial system of equations*, $x = P(x)$, called a *max-minPPS* is a system of $n$ equations in $n$ variables $x = (x_1, x_2, \ldots, x_n)$, where, for all $i \in \{1, 2, \ldots, n\}$, either

- (max polynomial) $P_i(x) = \max\{q_{i,j}(x) : j \in \{1, \ldots, m_i\}\}$ or
- (min polynomial) $P_i(x) = \min\{q_{i,j}(x) : j \in \{1, \ldots, m_i\}\}$,

where each $q_{i,j}(x)$ is a probabilistic polynomial, for every $j \in \{1, \ldots, m_i\}$.

We shall call such a system a *maxPPS* (respectively, a *minPPS*) if for every $i \in \{1, \ldots, n\}$, $P_i(x)$ is a max polynomial (respectively, a min polynomial).

Note that we can view a PPS in $n$ variables as a maxPPS, or as a minPPS, where $m_i = 1$ for every $i \in \{1, \ldots, n\}$.

For computational purposes, we assume that all the coefficients are rational. We assume that the polynomials in a system are given in sparse form, that is, by listing only the nonzero monomial terms, with the coefficient and the nonzero exponents of each variable in the term given in binary. We let $|P|$ denote the total bit encoding length of a system $x = P(x)$ under this representation.

We use *max/minPPS* to refer to a system of equations, $x = P(x)$, that is either a maxPPS or a minPPS. Whereas Etessami and Yannakakis [11] also considered systems of equations containing both max and min equations (which we refer to as max-minPPSs), our primary focus will be on systems that contain just one or the other. (But we shall also obtain results about max-minPPSs as a corollary.)

As was shown in Etessami and Yannakakis [11], any max-minPPS, $x = P(x)$, has a least fixed point solution $q^* \in [0,1]^n$, that is, $q^* = P(q^*)$, and if $q = P(q)$ for some $q \in [0,1]^n$, then $q^* \leq q$ (coordinate-wise inequality). As observed in Etessami and Yannakakis [10, 11], $q^*$ may in general contain irrational values, even in the case of PPSs. The central results of this paper yield P-time algorithms for computing $q^*$ to within arbitrary precision, both in the cases of maxPPSs and minPPSs. As we shall explain, our P-time upper bounds for computing (to within any desired accuracy) the least fixed point of maxPPSs and minPPSs will also yield, as corollaries, FNP upper bounds for computing approximately the LFP of max-minPPSs.

**Definition 2.** We define a *policy* for a max/minPPS, $x = P(x)$, to be a function $\sigma : \{1, \ldots n\} \to \mathbb{N}$ such that $1 \leq \sigma(i) \leq m_i$.

Intuitively, for each variable, $x_i$, a policy selects one of the probabilistic polynomials, $q_{i,\sigma(i)}(x)$, that appear on the right-hand side (RHS) of the equation $x_i = P_i(x)$ and that $P_i(x)$ is the maximum/minimum over.

**Definition 3.** Given a max/minPPS $x = P(x)$ over $n$ variables and a policy $\sigma$ for $x = P(x)$, we define the PPS $x = P_\sigma(x)$ by

$$(P_\sigma)_i(x) = q_{i,\sigma(i)}(x)$$

for all $i \in \{1, \ldots, n\}$.

Obviously, because a PPS is a special case of a max/minPPS, every PPS also has a unique LFP solution (this was established earlier in Etessami and Yannakakis [10]). Given a max/minPPS, $x = P(x)$, and a policy, $\sigma$, we use $q^*_\sigma$ to denote the LFP solution vector for the PPS $x = P_\sigma(x)$.

**Definition 4.** For a maxPPS $x = P(x)$, a policy $\sigma^*$ is called *optimal* if for all other policies $\sigma$, $q^*_{\sigma^*} \geq q^*_\sigma$. For a minPPS $x = P(x)$, a policy $\sigma^*$ is optimal if for all other policies $\sigma$, $q^*_{\sigma^*} \leq q^*_\sigma$. A policy $\sigma$ is $\epsilon$-*optimal* for $\epsilon > 0$ if $\|q^*_\sigma - q^*\|_\infty \leq \epsilon$.

A nontrivial theorem, established in Etessami and Yannakakis [11], is that optimal policies always exist, and furthermore that they actually attain the LFP $q^*$ of the max/minPPS:

**Theorem 1** (Etessami and Yannakakis [11, theorem 2]). *For any max/minPPS, $x = P(x)$, there always exists an optimal policy $\sigma^*$, and furthermore, $q^* = q^*_{\sigma^*}$.*[3]

Probabilistic polynomial systems can be used to capture central probabilities of interest for several basic stochastic models, including multitype branching processes, stochastic context-free grammars, and the class of one-exit recursive Markov chains (1-RMCs; Etessami and Yannakakis [10]). MaxPPSs and minPPSs can be similarly used to capture the central optimum probabilities of corresponding stochastic optimization models: (multitype) branching Markov decision processes, context-free MDPs, and one-exit recursive Markov decision processes (Etessami and Yannakakis [11]). We now define BMDPs and 1-RMDPs.

A *branching Markov decision process* (BMDP) consists of a finite set $V = \{T_1, \ldots, T_n\}$ of types, a finite set $A_i$ of actions for each type, and a finite set $R(T_i, a)$ of probabilistic rules for each type $T_i$ and action $a \in A_i$. Each rule $r \in R(T_i, a)$ has the form $T_i \xrightarrow{p_r} \alpha_r$, where $\alpha_r$ is a finite multiset whose elements are in $V$, $p_r \in (0,1]$ is the probability of the rule, and the sum of the probabilities of all the rules in $R(T_i, a)$ is equal to 1: $\sum_{r \in R(T_i,a)} p_r = 1$.

Intuitively, a BMDP describes the stochastic evolution of entities of given types in the presence of a controller that can influence the evolution. A population $X$ is a finite set of entities of given types, and it can be represented by a vector $v(X) \in \mathbb{N}^n$, where $v_i(X)$ is the number of entities of $X$ of type $T_i$. Starting from an initial population $X_0$ at time (generation) 0, a sequence of populations $X_1, X_2, \ldots$ is generated, where $X_k$ is obtained from $X_{k-1}$ as follows. First, the controller selects for each entity of $X_{k-1}$ an available action for the type of the entity; then a rule is chosen independently and simultaneously for every entity of $X_{k-1}$ probabilistically according to the probabilities of the rules for the type of the entity and the selected action, and the entity is replaced by a new set of entities with the types specified by the right-hand side of the rule. The process is repeated as long as the current population $X_k$ is nonempty, and terminates if and when $X_k$ becomes empty. The objective of the controller is to either minimize the probability of termination (i.e., extinction of the

population), in which case the process is a minBMDP, or maximize the termination probability, in which case it is a maxBMDP. At each stage, $k$, the controller is allowed in principle to select the actions for the entities of $X_k$ based on the whole past history, may use randomization (a mixed strategy), and may make different choices for entities of the same type. However, it turns out that these flexibilities do not increase the controller's power, and there is always an optimal pure, memoryless strategy that always chooses the same action for all entities of the same type (Etessami and Yannakakis [11]).

For each type $T_i$ of a minBMDP (respectively, maxBMDP), let $q_i^*$ be the minimum (respectively, maximum) probability of termination if the initial population consists of a single entity of type $T_i$. From the given minBMDP (maxBMDP), we can construct a minPPS (respectively, maxPPS) $x = P(x)$ whose LFP is precisely the vector $q^*$ of optimal termination (extinction) probabilities (see theorem 20 of Etessami and Yannakakis [11]): the min/max polynomial $P_i(x)$ for each type $T_i$ contains one polynomial $q_{i,j}(x)$ for each action $j \in A_i$, with $q_{i,j}(x) = \sum_{r \in R(T_i,j)} p_r x^{\alpha_r}$.

**Example 1.** Suppose there are two types of entities (e.g., bacteria), $T_1, T_2$. For type $T_1$, we have three available actions, $a_1, a_2, a_3$. Under $a_1$, a type $T_1$ entity dies with probability 0.3 and produces two $T_1$ offspring with probability 0.7. We can write these rules succinctly as $R(T_1, a_1) = \{T_1 \xrightarrow{0.3} \emptyset,\ T_1 \xrightarrow{0.7} T_1T_1\}$, where $T_1T_1$ denotes the multiset $\{T_1, T_1\}$. Action $a_2$ increases the probability of death to 0.4 but also introduces the possibility that one of the offspring is mutated to the more resilient type $T_2$ with probability 0.1, that is, $R(T_1, a_2) = \{T_1 \xrightarrow{0.4} \emptyset,\ T_1 \xrightarrow{0.1} T_1T_2,\ T_1 \xrightarrow{0.5} T_1T_1\}$. Action $a_3$ has rules $R(T_1, a_3) = \{T_1 \xrightarrow{0.5} \emptyset,\ T_1 \xrightarrow{0.3} T_1T_2,\ T_1 \xrightarrow{0.2} T_1T_1\}$. For type $T_2$, we have two actions available, $b_1, b_2$. The rules are $R(T_2, b_1) = \{T_2 \xrightarrow{0.2} \emptyset,\ T_2 \xrightarrow{0.5} T_1T_2,\ T_2 \xrightarrow{0.3} T_2T_2\}$, and $R(T_2, b_2) = \{T_2 \xrightarrow{0.3} \emptyset,\ T_2 \xrightarrow{0.2} T_1T_2,\ T_2 \xrightarrow{0.5} T_2T_2\}$. The goal is to choose a strategy that maximizes the probability of extinction.

The corresponding maxPPS has two variables $(x_1, x_2)$ for the optimal extinction probabilities of the two types $(T_1, T_2)$ and has equations $x_1 = \max\{0.7x_1^2 + 0.3, 0.5x_1^2 + 0.1x_1x_2 + 0.4, 0.2x_1^2 + 0.3x_1x_2 + 0.5\}$ and $x_2 = \max\{0.5x_1x_2 + 0.3x_2^2 + 0.2, 0.2x_1x_2 + 0.5x_2^2 + 0.3\}$. To see the justification for these Bellman equations, suppose for example that we select action $a_1$ for $T_1$. Then, with probability 0.3, the process becomes extinct at this point, and with probability 0.7, there are two offspring of type $T_1$ and the process will become extinct iff both processes originating from the two offspring become extinct. Hence, in the case of $a_1$, the extinction probability $x_1$ satisfies $x_1 = 0.7x_1^2 + 0.3$. The expressions for the other actions $(a_2, a_3)$ are derived similarly, and naturally we want to select the action that yields the maximum value among them. The intuitive reason why the true optimal extinction probabilities are given by the *least* fixed point of the equations was explained in the introduction. The LFP of the system in this example, and the vector of maximum extinction probabilities, is $q^* \approx (0.7, 0.486)$. The optimal strategy is to use action $a_3$ for $T_1$ and $b_2$ for $T_2$. □

A *one-exit recursive Markov decision process* (1-RMDP) consists of a finite set of components, $A_1, \ldots, A_k$, where each component $A_i$ is essentially a finite-state MDP augmented with the ability to make recursive calls to itself and other components. Formally, each component $A_i$ has a finite set $N_i$ of nodes, which are partitioned into probabilistic nodes and controlled nodes, and a finite set $B_i$ of "boxes" (or supernodes), where each box is mapped to some component. One node $en_i$ is specified as the entry of the component $A_i$, and one node $ex_i$ as the exit of $A_i$.[4] The exit node has no outgoing edges. All other nodes and the boxes have outgoing edges; the edges out of the probabilistic nodes and boxes are labeled with probabilities, where the sum of the probabilities out of the same node or box is equal to one.

One-exit RMDPs serve as natural models for a class of recursive probabilistic programs. The components correspond to the recursive procedures, probabilistic nodes correspond to the probabilistic steps, controlled nodes correspond to the nonprobabilistic steps (e.g., branching steps), and the boxes correspond to the recursive calls. Execution of a 1-RMDP starts at some node, for example, the entry $en_1$ of component $A_1$. When the execution is at a probabilistic node $v$, then an edge out of $v$ is chosen randomly according to the probabilities of the edges out of $v$. At a controlled node $v$, an edge out of $v$ is chosen by a controller who seeks to optimize his objective. When the execution reaches a box $b$ of $A_i$ mapped to some component $A_j$, then the current component is suspended and a recursive call to $A_j$ is initiated at its entry node $en_j$; if the call to $A_j$ terminates, that is, reaches eventually its exit node $ex_j$, then the execution of component $A_i$ resumes from box $b$ following an edge out of $b$ chosen according to the probability distribution of the outgoing edges of $b$. Note that a call to a component can make further recursive calls, and thus, at any point, there is in general a stack of suspended recursive calls, and there can be an arbitrary number of such suspended calls. Thus, a 1-RMDP induces generally an infinite-state MDP. The process terminates when the execution reaches the exit of the component of the initial node and there are no suspended recursive calls.

There are two types of 1-RMDPs with a termination objective: In a min 1-RMDP (respectively, max 1-RMDP), the objective of the controller is to minimize (respectively, maximize) the probability of termination. In principle, a controller can use the complete past history of the process and also use randomization (i.e., a mixed strategy) to select at each point when the execution reaches a controlled node which edge to select out of the node. As shown in Etessami and Yannakakis [11], however, there is always an optimal strategy that is pure, stackless, and memoryless, that is, selects deterministically one edge out of each controlled node, the same one every time, independent of the stack of suspended calls and of the past history (including the starting node). From a given min or max 1-RMDP, we can construct efficiently a minPPS or maxPPS, whose LFP yields the minimum or maximum termination probabilities for all the different possible starting vertices of the 1-RMDP (Etessami and Yannakakis [11]): There is one variable $x_u$ for each node $u$, corresponding to the optimal termination probability starting at node $u$, and two variables $x_b, x'_b$ for each box $b$, corresponding to the optimal termination probabilities respectively when the recursive call for $b$ is made (is initiated) and when it returns. The exit nodes $ex_i$ have value $x_{ex_i} = 1$. The equation for each probabilistic node $u$ whose outgoing edges $(u, v)$ have probabilities $p_{uv}$ is $x_u = \sum_v p_{uv} x_v$; the equation for the variable $x'_b$ of a box $b$ is $x'_b = \sum_v p_{bv} x_v$; the equation for the variable $x_b$ of a box $b$ mapped to component $A_i$ with entry $en_i$ is $x_b = x_{en_i} x'_b$; the equation for a controlled node $u$ in a min 1-RMDP (respectively, max 1-RMDP) is $x_u = \min\{x_v | (u, v) \in E\}$ (respectively, $x_u = \max\{x_v | (u, v) \in E\}$. Conversely, from any given max/minPPS, we can efficiently construct a 1-RMDP whose optimal termination probabilities yield the LFP of the max/minPPS. The system of equations for a 1-RMDP has a particularly simple form. All max/minPPSs can be put in that form.

It is convenient to put max/minPPSs in the following simple form.

**Definition 5.** A maxPPS in *simple normal form* (SNF), $x = P(x)$, is a system of $n$ equations in $n$ variables $x_1, x_2, \ldots x_n$, where each $P_i(x)$ for $i = 1, 2, \ldots n$ is in one of three forms:

- Form L: $P(x)_i = a_{i,0} + \sum_{j=1}^{n} a_{i,j} x_j$, where $a_{i,j} \geq 0$ for all $j$, and such that $\sum_{j=0}^{n} a_{i,j} \leq 1$.
- Form Q: $P(x)_i = x_j x_k$ for some $j, k$.
- Form M: $P(x)_i = \max\{x_j, x_k\}$ for some $j, k$.

We define SNF for minPPSs analogously: only the definition of form M changes, replacing max with min.

In the setting of a max/minPPS in SNF, for simplicity in notation, when we talk about a policy, if $P_i(x)$ has form M, say $P_i(x) \equiv \max\{x_j, x_k\}$, then when it is clear from the context, we will use $\sigma(i) = k$ to mean that the policy $\sigma$ chooses $x_k$ among the two choices $x_j$ and $x_k$ available in $P_i(x) \equiv \max\{x_j, x_k\}$.

Using similar techniques as in Etessami and Yannakakis [10], it is easy to show that every max/minPPS can be transformed efficiently to one in SNF; see the appendix for the proof.

**Proposition 1** (See Etessami and Yannakakis [10, proposition 7.3]). *Every max/minPPS, $x = P(x)$, can be transformed in P-time to an "* equivalent*" max/minPPS, $y = Q(y)$ in SNF, such that $|Q| \in O(|P|)$. More precisely, the variables $x$ are a subset of the variables $y$, the LFP of $x = P(x)$ is the projection of the LFP of $y = Q(y)$, and an optimal policy (respectively, $\epsilon$-optimal policy) for $x = P(x)$ can be obtained in P-time from an optimal (respectively, $\epsilon$-optimal) policy of $y = Q(y)$.*

Thus, from now on, we assume, without loss of generality (w.l.o.g.), that *all max/minPPSs are in SNF normal form.*

The *dependency graph* of a max/minPPS $x = P(x)$ is a directed graph $G$ that has one node for every variable and has an edge $x_i \to x_j$ iff the variable $x_j$ appears in $P_i(x)$. We say that the system $x = P(x)$ is strongly connected if its dependency graph is strongly connected, that is, if every node has a directed path to every other node.

We now summarize some of the main previous results on PPSs and max/minPPSs.

**Proposition 2** (See Etessami and Yannakakis [11] or the Online Companion of This Paper). *There is a P-time algorithm that, given a minPPS or maxPPS, $x = P(x)$, over $n$ variables with LFP $q^* \in \mathbb{R}^n_{\geq 0}$, determines for every $i = 1, \ldots, n$ whether $q^*_i = 0$ or $q^*_i = 1$ or $0 < q^*_i < 1$.*[5]

Thus, given a max/minPPS, we can find in P-time all the variables $x_i$ such that $q^*_i = 0$ or $q^*_i = 1$, remove them and their corresponding equations $x_i = P_i(x)$, and substitute their values on the RHS of the remaining equations. This yields a new max/minPPS, $x' = P'(x')$, where its LFP solution, $q'^*$, is $0 < q'^* < 1$, which corresponds to the remaining coordinates of $q^*$. Thus, it suffices to focus our attention to systems whose LFP is strictly between 0 and 1.

The decision problem of determining whether a coordinate $q^*_i$ of the LFP is $\geq 1/2$ (or whether $q^*_i \geq r$ for any other given bound $r \in (0, 1)$) is at least as hard as the Sqrt-Sum and the PosSLP problems even for PPS (without the min and max operator; Etessami and Yannakakis [10]), and hence it is highly unlikely that it can be solved in P-time.

The problem of approximating efficiently the LFP of a PPS was solved recently in Etessami et al. [12, 15] by using Newton's method after elimination of the variables with values 0 and 1.

**Definition 6.** For a PPS $x = P(x)$ we use $P'(x)$ to denote the Jacobian matrix of partial derivatives of $P(x)$, that is, $P'(x)_{i,j} := \frac{\partial P_i(x)}{\partial x_j}$. For a point $x \in \mathbb{R}^n$, if $(I - P'(x))$ is nonsingular, then we define one Newton iteration at $x$ via the operator:

$$\mathcal{N}(x) = x + \big(I - P'(x)\big)^{-1}\big(P(x) - x\big).$$

Given a max/minPPS $x = P(x)$ and a policy $\sigma$, we use $\mathcal{N}_\sigma(x)$ to denote the Newton operator of the PPS $x = P_\sigma(x)$; that is, if $(I - P'_\sigma(x))$ is nonsingular at a point $x \in \mathbb{R}^n$, then $\mathcal{N}_\sigma(x) = x + (I - P'_\sigma(x))^{-1}(P_\sigma(x) - x)$.

**Theorem 2** (Etessami et al. [15, theorem 3.2, corollary 4.5]). *Let $x = P(x)$ be a PPS with rational coefficients in SNF that has least fixed point $0 < q^* < 1$. If we conduct iterations of Newton's method by setting $x^{(0)} := 0$, and for $k \geq 0$, $x^{(k+1)} := \mathcal{N}(x^{(k)})$, then the Newton operator $\mathcal{N}(x^{(k)})$ is defined for all $k \geq 0$, and for any $j > 0$,*

$$\left\| q^* - x^{\left(j+4|P|\right)} \right\|_\infty \leq 2^{-j},$$

*where $|P|$ is the total bit encoding length of the system $x = P(x)$.*

*Furthermore, there is an algorithm (based on suitable rounding of Newton iterations) that, given a PPS $x = P(x)$ and given a positive integer $j$, computes a rational vector $v \in [0,1]^n$ such that $\|q^* - v\|_\infty \leq 2^{-j}$, and that runs in time polynomial in $|P|$ and $j$ in the standard Turing model of computation.*

The proof of the theorem involves a number of technical lemmas on PPSs and Newton's method, several of which we will also need in this paper, some of them in strengthened form, and which we include in the appendix. The following lemma summarizes key properties of the Newton operator for PPSs that are crucial for the correctness and the polynomial running time.

**Lemma 1** (Combining lemmas 3.9 and 3.5 and theorem 3.7 of Etessami et al. [15]). *Let $x = P(x)$ be a PPS in SNF with $0 < q^* < 1$. For any $0 \leq x \leq q^*$ and $\lambda > 0$, the operator $\mathcal{N}(x)$ is defined (i.e., the matrix $I - P'(x)$ is nonsingular), $\mathcal{N}(x) \leq q^*$, and if $q^* - x \leq \lambda(1 - q^*)$, then $q^* - \mathcal{N}(x) \leq \frac{\lambda}{2}(1 - q^*)$.*

If we knew an optimal policy $\tau$ for a max/minPPS, $x = P(x)$, then we would be able to solve the problem of computing the LFP for a max/minPPS using the algorithm in Etessami et al. [15] for approximating $q^*_\tau$, because we know $q^*_\tau = q^*$. Unfortunately, we do not know which policy is optimal. There are exponentially many policies, so it would be inefficient to run this algorithm using every policy. (And even if we did do so for each possible policy, we would only be able to $\epsilon$-approximate the values $q^*_\sigma$ for each policy $\sigma$ using the results of Etessami et al. [15], for, say, $\epsilon = 2^{-j}$ for some chosen $j$, and therefore we could only be sure that a particular policy that yields the best result is, say, $(2\epsilon)$-optimal, but it may not necessarily be optimal.) In fact, as we will see, it is probably impossible to identify an optimal policy in polynomial time.

Our goal instead will be to find an iteration $I(x)$ for a max/minPPS that has similar properties to the Newton operator for a PPS, that is, that can be computed efficiently for a given $x$ and for which we can prove a similar property to Lemma 1, that is, such that if $q^* - x \leq \lambda(1 - q^*)$, then $q^* - I(x) \leq \frac{\lambda}{2}(1 - q^*)$. Once we do so, we will be able to adapt and extend results from Etessami et al. [15] to get a polynomial time algorithm for the problem of approximating the LFP $q^*$ of a max/minPPS.

## 3. Generalizing Newton's Method Using Linear Programming

If a max/minPPS, $x = P(x)$, has no equations of form Q, then it amounts to precisely the Bellman equations for an ordinary finite-state Markov decision process with the objective of maximizing/minimizing reachability probabilities. It is well known that we can compute the exact (rational) optimal values for such finite-state MDPs, and thus the exact LFP, $q^*$, for such max-linear (min-linear) systems using linear programming (see, e.g., Courcoubetis and Yannakakis [5], Puterman [25]). Computing the LFP of max/minPPSs is clearly a generalization of this finite-state MDP problem to the infinite-state setting of branching and recursive MDPs.

If we have no equations of form M, we have a PPS, which we can solve in P-time, as shown recently in Etessami et al. [15]. The algorithm first preprocesses the system to identify the variables that have value 0 or 1 in the LFP, removes them from the system, and then applies Newton's method on the remaining system. Recall that an iteration of Newton's method works by approximating the system of equations by a linear system, which is a linearization of the system around the current point, and solving this linear system to obtain the new point.

For maxPPSs (or minPPSs) we employ a similar approach. We first identify the variables that have value 0 or 1 in the LFP using the algorithms of Etessami and Yannakakis [11] or the improved algorithms in the online companion of this paper. We remove these variables, substituting their values, and thereby obtain a reduced system on the remaining variables whose LFP $q^*$ satisfies $0 < q^* < 1$. We compute (approximately) the LFP of the remaining maxPPS (or minPPS) by an iterative algorithm, which we call *generalized Newton method* (GNM). For this purpose, we define an analogous "approximate" system of equations at the current point, which has both linear equations and equations involving the max (or min) function. We show that we can solve the equations that arise from each iteration of the GNM using linear programming. We then show that a polynomial (in fact, linear) number of iterations are enough to approximate the desired LFP solution, and that it suffices to carry out the computations with polynomial precision.

We begin by defining formally the max/min linear equations that should be solved by one iteration of the GNM, applied at a point $y$. Recall that we assume w.l.o.g. throughout that max/minPPSs and PPSs are in SNF.

**Definition 7.** For a max/minPPS $x = P(x)$ with $n$ variables, the *linearization of* $P(x)$ *at a point* $\mathbf{y} \in \mathbb{R}^n$ is a system of max/min linear functions, denoted by $P^y(x)$, that has the following form: if $P_i(x)$ has form L or M, then $P_i^y(x) = P_i(x)$, and if $P_i(x)$ has form Q, that is, $P_i(x) = x_j x_k$ for some $j,\ k$, then

$$P_i^y(x) = y_j x_k + x_j y_k - y_j y_k.$$

**Example 2.** Consider the following minPPS $x = P(x)$:

$$x_1 = 0.2x_2 + 0.3x_3 + 0.5; \quad x_2 = 0.4x_1 + 0.1x_3 + 0.5x_4; \quad x_3 = \min(x_2, x_5); \quad x_4 = x_1x_3; \quad x_5 = x_1^2.$$

Its linearization $x = P^y(x)$ at the point $y = (0.8, 0.3, 0.4, 0.2, 0.5)$ is

$$x_1 = 0.2x_2 + 0.3x_3 + 0.5; \quad x_2 = 0.4x_1 + 0.1x_3 + 0.5x_4; \quad x_3 = \min(x_2, x_5);$$
$$x_4 = 0.4x_1 + 0.8x_3 - 0.32; \quad x_5 = 1.6x_1 - 0.64. \quad \square$$

We define distinct iteration operators for a maxPPS and a minPPS, both of which we shall refer to with the overloaded notation $I(y)$. These operators will serve as the basis for a generalized Newton method to be applied to maxPPSs and minPPSs, respectively.

**Definition 8.** For a maxPPS $x = P(x)$ with LFP $q^*$ such that $0 < q^* < 1$, and for a real vector $y$ such that $0 \le y \le q^*$, we define the operator $I(y)$ to be the *unique* optimal solution, $a \in \mathbb{R}^n$, to the following mathematical program:[6]

$$\text{Minimize:} \quad \sum_i a_i, \quad \text{Subject to:} \quad P^y(a) \le a.$$

For a minPPS $x = P(x)$ with LFP $q^*$ such that $0 < q^* < 1$, and for a real vector $y$ such that $0 \le y \le q^*$, we define the operator $I(y)$ to be the *unique* optimal solution $a \in \mathbb{R}^n$ to the following mathematical program:

$$\text{Maximize:} \quad \sum_i a_i, \quad \text{Subject to:} \quad P^y(a) \ge a.$$

A priori, it is not obvious that the above definitions of $I(y)$ for maxPPSs and minPPSs are well defined, that is, that the mathematical programs are feasible and have unique optimal solutions. We will show in the following subsections that this is indeed the case. We will also show that the mathematical programs can be expressed as linear programs, and thus can be solved in polynomial time.

**Example 3.** For the minPPS $x = P(x)$ of the previous example and the vector $y = (0.8, 0.3, 0.4, 0.2, 0.5)$, the corresponding mathematical program is

$$\text{Maximize:} \quad \sum_{i=1}^{5} a_i$$
$$\text{Subject to:}$$
$$a_1 \le 0.2a_2 + 0.3a_3 + 0.5, \quad a_2 \le 0.4a_1 + 0.1a_3 + 0.5a_4, \quad a_3 \le \min(a_2, a_5),$$
$$a_4 \le 0.4a_1 + 0.8a_3 - 0.32, \quad a_5 \le 1.6a_1 - 0.64.$$

The third constraint can be written equivalently as the conjunction of the inequalities $a_3 \le a_2$ and $a_3 \le a_5$, which yields a linear program. The LP has a unique optimal solution $(0.85, 0.7, 0.7, 0.58, 0.72)$, and this vector is $I(y)$. Note that this vector satisfies $a = P^y(a)$. $\square$

Now we can give a polynomial time algorithm, in the Turing model of computation, for approximating the LFP for a max/minPPS to within any desired precision. First, compute the set of variables that have value 0 or 1 in the LFP using the P-time algorithms of Etessami and Yannakakis [11], or the improved P-time algorithms given in the online companion of this paper. Remove these variables from the system, yielding a remaining system whose LFP $q^*$ satisfies $0 < q^* < 1$. Then apply the following algorithm to compute iteratively a sequence of points $x^{(k)}, k = 0, 1, 2, \ldots$, starting from $x^{(0)} := 0$, rounding down every point along the computation to $h$ bits of precision. The number of iterations and the rounding parameter $h$ depend on the desired number of bits of precision in the approximation of the LFP.

**Algorithm 1** (Generalized Newton Method with Rounding)
Start with $x^{(0)} := 0$;
For each $k \geq 0$, compute $x^{(k+1)}$ from $x^{(k)}$ as follows:
1. Calculate $I(x^{(k)})$ by solving the following LP:

$$\text{Minimize} \sum_i x_i, \text{ subject to } P^{x^{(k)}}(x) \leq x, \text{ if } x = P(x) \text{ is a maxPPS;}$$

or

$$\text{Maximize} \sum_i x_i, \text{ subject to } P^{x^{(k)}}(x) \geq x, \text{ if } x = P(x) \text{ is a minPPS.}$$

2. For each coordinate $i = 1, 2, \ldots n$, set $x_i^{(k+1)}$ to be the maximum (nonnegative) multiple of $2^{-h}$ that is $\leq \max\{0, I(x^{(k)})_i\}$. (In other words, we round $I(x^{(k)})$ down to the nearest $2^{-h}$ and ensure it is nonnegative.)

**Example 4.** Consider the minPPS of the previous examples. Applying the algorithm from Etessami and Yannakakis [11] (or from the online companion of this paper) yields that all variables have value strictly between 0 and 1 in the LFP, and thus no variable is eliminated. We start the generalized Newton method with $x^{(0)} := 0$. The LP is the same as that of Example 3 except for the last two constraints (corresponding to the form Q equations of the minPPS), which are $a_4 \leq 0$ and $a_5 \leq 0$. Solving the LP yields the next point, $x^{(1)} \approx (0.54, 0.22, 0, 0, 0)$. The LP in the next iteration changes again only in the last two constraints and yields the next point, $x^{(2)} \approx (0.73, 0.47, 0.47, 0.25, 0.50)$. After a few more iterations, we get $x^{(5)} \approx (0.897, 0.795, 0.795, 0.713, 0.805)$ and $x^{(6)} \approx (0.8999, 0.7999, 0.7999, 0.7198, 0.8099)$. The actual LFP is $q^* = (0.9, 0.8, 0.8, 0.72, 0.81)$. In this example, value iteration takes about 200 iterations to reach the accuracy of $x^{(5)}$ and 400 iterations for $x^{(6)}$ (of course the iterations themselves are simpler).

We shall prove the following theorem.

**Theorem 3.** *Given any max/minPPS $x = P(x)$ with LFP $0 < q^* < 1$, if we use the above algorithm with rounding parameter $h = j + 2 + 4|P|$, then the iterations are all defined, and for every $k \geq 0$ we have $0 \leq x^{(k)} \leq q^*$. Furthermore, after $h - 1 = j + 1 + 4|P|$ iterations, we have*

$$\left\| q^* - x^{(j+1+4|P|)} \right\|_\infty \leq 2^{-j}.$$

**Corollary 1.** *Given any max/minPPS $x = P(x)$ with LFP $q^*$, and given any integer $j > 0$, there is an algorithm that computes a rational vector $v \leq q^*$ with $\|q^* - v\|_\infty \leq 2^{-j}$ in time polynomial in $|P|$ and $j$.*

The rest of this section is devoted to proving Theorem 3 and the corollary. The section is organized as follows. In Section 3.1, we give some basic properties of the linearization of a max/minPPS (their proofs are given in the appendix). In Section 3.2, we state the key properties of the operator $I(\cdot)$ for an iteration of the generalized Newton method. In Section 3.3, we analyze the operator for a maxPPS, and in Section 3.4 that for a minPPS, and we prove its key properties. Finally, in Section 3.5, we put everything together and prove Theorem 3 and Corollary 1, showing that the algorithm approximates the LFP within any desired precision in polynomial time in the Turing model.

### 3.1. Linearizations of Max/MinPPSs and Their Properties

Let $x = P(x)$ be a maxPPS or minPPS. For any policy $\sigma$, we can consider the linearization of the corresponding PPS, $x = P_\sigma(x)$.

**Definition 9.** $P^y_\sigma(x) := (P_\sigma)^y(x)$.

Note that the linearization $P^y(x)$ changes only equations of form Q, and using a policy $\sigma$ changes only equations of form M, so these operations are independent in terms of the effects they have on the underlying equations, and thus $P^y_\sigma(x) \equiv (P_\sigma)^y(x) = (P^y)_\sigma(x)$.

We first state some basic properties of linearizations of PPS (without max or min); see the appendix for the proofs.

**Lemma 2.** *Let $x = P(x)$ be any PPS. For any $y \in \mathbb{R}^n$, let $(P^y)'(x)$ denote the Jacobian matrix of $P^y(x)$. Then, for any $x \in \mathbb{R}^n$, we have $(P^y)'(x) = P'(y)$.*

**Lemma 3.** *If $x = P(x)$ is any PPS, then for any $x, y \in \mathbb{R}^n$, $P^y(x) = P(y) + P'(y)(x - y)$.*

An iteration of Newton's method on $x = P_\sigma(x)$ at a point $y$ solves a system of linear equations that can be expressed in terms of $P^y_\sigma(x)$. The next lemma establishes this basic fact in part (i). In part (ii) it provides us with conditions under which we are guaranteed to be doing *at least as well* as one such Newton iteration (see the appendix for the proof).

**Lemma 4.** *Let $x = P(x)$ be any max/minPPS. Suppose that the matrix inverse $(I - P'_\sigma(y))^{-1}$ exists and is nonnegative, for some policy $\sigma$ and some $y \in \mathbb{R}^n$. Then*

(i) *$\mathcal{N}_\sigma(y)$ is defined and is equal to the unique point $a \in \mathbb{R}^n$ such that $P^y_\sigma(a) = a$;*

(ii) *for any vector $x \in \mathbb{R}^n$, if $P^y_\sigma(x) \geq x$, then $x \leq \mathcal{N}_\sigma(y)$, and if $P^y_\sigma(x) \leq x$, then $x \geq \mathcal{N}_\sigma(y)$.*

### 3.2. Key Properties of the Iteration Operator of the GNM

Definition 8 defines the iteration operator $I(y)$ as the unique optimal solution of a mathematical program. We first observe that this mathematical program can be expressed as a linear program, for both maxPPSs and minPPSs.

**Proposition 3.** *Given a max/minPPS $x = P(x)$ with LFP $q^*$, and given a rational vector $y$, $0 \leq y \leq q^*$, the constrained optimization problem (i.e., mathematical program) defining $I(y)$ can be described by a LP whose encoding size is polynomial (in fact, linear) in both $|P|$ and the encoding size of the rational vector $y$. Thus, we can compute the (unique) optimal solution $I(y)$ to such an LP (assuming it exists and is unique) in P-time.*

**Proof.** For a maxPPS (minPPS), the definition of $I(y)$ asks us to maximize (minimize) a linear objective, $\sum_i a_i$, subject to the constraints $P^y(a) \leq a$ ($P^y(a) \geq a$, respectively). All of these constraints are linear, except the constraints of form M. For a maxPPS, if $(P^y(a))_i$ is of form M, then the corresponding constraint is an inequality of the form $\max\{a_j, a_k\} \leq a_i$. Such an inequality is equivalent to, and can be replaced by, the two linear inequalities $a_j \leq a_i$ and $a_k \leq a_i$. Likewise, for a minPPS, if $(P^y(a))_i$ is of form M, then the corresponding constraint is an inequality of the form $\min\{a_j, a_k\} \geq a_i$. Again, such an inequality is equivalent to, and can be replaced by, two linear inequalities $a_j \geq a_i$ and $a_k \geq a_i$.

Thus, for a rational vector $y$ whose encoding length is $\mathsf{size}(y)$, the operator $I(y)$ can be formulated (for both maxPPSs and minPPSs) as a problem of computing the unique optimal solution to a linear program whose encoding size is polynomial (in fact, linear) in $|P|$ and in $\mathsf{size}(y)$. □

The following proposition lists key properties of the operator $I(y)$. In particular (part (i)), the operator is well defined if $0 < y < q^*$, that is, the mathematical program is feasible and has a unique optimal solution. Furthermore (part (ii)), the iteration makes in some sense good progress toward the LFP $q^*$; this part is useful in establishing the speed of convergence of the GNM.

**Proposition 4.** *Let $x = P(x)$ be a max/minPPS with LFP $q^*$ such that $0 < q^* < 1$. For any $0 \leq y \leq q^*$,*

(i) *$I(y)$ is well defined, and $I(y) \leq q^*$, and*

(ii) *for any $\lambda > 0$, if $q^* - y \leq \lambda(1 - q^*)$, then $q^* - I(y) \leq \frac{\lambda}{2}(1 - q^*)$.*

We shall prove the proposition separately for maxPPSs and minPPSs in the following two subsections.

### 3.3. An Iteration of the Generalized Newton Method for MaxPPSs

In this subsection, we will analyze the operator $I(y)$ for a maxPPS and prove its key properties given in Proposition 4. For a maxPPS, $x = P(x)$, we know by Theorem 1 that there exists an optimal policy, $\tau$, such that $q^* = q^*_\tau \geq q^*_\sigma$ for any policy $\sigma$. The next lemma implies part (i) of Proposition 4 for maxPPSs.

**Lemma 5.** *If $x = P(x)$ is a maxPPS with LFP solution $0 < q^* < 1$ and $y$ is a real vector with $0 \leq y \leq q^*$, then $x = P^y(x)$ has a least fixed point solution, denoted by $\mu P^y$, with $\mu P^y \leq q^*$. Furthermore, the operator $I(y)$ is well defined, $I(y) = \mu P^y \leq q^*$, and for any optimal policy $\tau$, $I(y) = \mu P^y \geq \mathcal{N}_\tau(y)$.*

**Proof.** Recall that (by Proposition 3) the mathematical program that "defines" $I(y)$ can be written equivalently as an LP:

$$\text{Minimize:} \quad \sum_i a_i, \qquad \text{Subject to:} \quad P^y(a) \le a. \tag{1}$$

First, we show that the LP constraints $P^y(a) \le a$ in the definition of $I(y)$ are *feasible*. We do so by showing that actually $P^y(q^*) \le q^*$. At any coordinate $i$, if $P_i(x)$ has form M or L, then $P_i^y(q^*) = P_i(q^*) = q_i^*$. Otherwise, $P_i(x)$ has form Q, that is, $P_i(x) = x_j x_k$, and then

$$\begin{aligned} P_i^y(q^*) &= q_j^* y_k + y_j q_k^* - y_j y_k \\ &= q_j^* q_k^* - \left(q_j^* - y_j\right)\left(q_k^* - y_k\right) \\ &\le q_i^* \qquad \left(\text{because } y \le q^*\right). \end{aligned}$$

Next we show that the LP (1) defining $I(y)$ is *bounded*. Recall that, by Theorem 1, there is always an optimal policy for any maxPPS, $x = P(x)$.

**Claim 1.** *Let $x = P(x)$ be any maxPPS, with $0 < q^* < 1$, and let $\tau$ be any optimal policy for $x = P(x)$. For any $y$ such that $0 \le y \le q^*$, we have that $\mathcal{N}_\tau(y)$ is defined, and for any vector $a$, if $P^y(a) \le a$, then $\mathcal{N}_\tau(y) \le a$. In particular, $\mathcal{N}_\tau(y) \le q^*$.*

**Proof.** Recall from our definition of an optimal policy that $q^* = q_\tau^*$ is also the least nonnegative solution to $x = P_\tau(x)$. So we can apply Lemma A.3 (in the appendix) using $x = P_\tau(x)$ and $y \le q^*$ to deduce that $(I - P'_\tau(y))^{-1}$ exists and is nonnegative. Thus, $\mathcal{N}_\tau(y)$ is defined. Now, by applying Lemma 4(ii), to show that $a \ge \mathcal{N}_\tau(y)$, all we need to show is that $P_\tau^y(a) \le a$. But recalling that $x = P(x)$ is a maxPPS, by the definition of $P^y(x)$ and $P_\tau^y(x)$, we have that $P_\tau^y(a) \le P^y(a) \le a$. We showed just before this claim that $P^y(q^*) \le q^*$, and thus $\mathcal{N}_\tau(y) \le q^*$. □

Thus, the LP (1) defining $I(y)$ is both feasible and bounded; hence, it has an optimal solution. To show that $I(y)$ is well defined, all that remains is to show that this optimal solution is unique. In the process, we will also show that $I(y)$ defines precisely the *least fixed point* solution of $x = P^y(x)$, which we denote by $\mu P^y$.

First, we claim that for any optimal solution $b$ to the LP (1), it must be the case that $P^y(b) = b$. Suppose not. Then there exists $i$ such that $P^y(b)_i < b_i$. Then we can define a new vector $b'$ such that $b'_i = P^y(b)_i$ and $b'_j = b_j$ for all $j \ne i$. By monotonicity of $P^y(x)$, it is clear that $P^y(b') \le b'$, and thus that $b'$ is a feasible solution to the LP (1). But $\sum_i b'_i < \sum_i b_i$, contradicting the assumption that $b$ is an optimal solution to the LP (1).

Second, we claim that there is a unique optimal solution. Suppose not: suppose $b$ and $c$ are two distinct optimal solutions to the LP (1). Define a new vector $d$ by $d_i = \min\{b_i, c_i\}$, for all $i$. Clearly, $d \le b$ and $d \le c$. Thus, by the monotonicity of $P^y(x)$, for all $i$, $P^y(d)_i \le P^y(b)_i = b_i$, and likewise, $P^y(d)_i \le P^y(c)_i = c_i$. Thus, $P^y(d) \le d$, and $d$ is a feasible solution to the LP. But because $b$ and $c$ are distinct, and yet $\sum_i b_i = \sum_i c_i$, we have that $\sum_i d_i < \sum_i b_i = \sum_i c_i$, contradicting the optimality of both $b$ and $c$.

We have thus established that $I(y)$ defines the unique *least fixed point* solution of $x = P^y(x)$, which we denote also by $\mu P^y$. Because $q^*$ is also a solution of the LP, we have $\mu P^y \le q^*$.

Finally, by Claim 1, it must be the case that $I(y) = \mu P^y \ge \mathcal{N}_\tau(y)$, where $\tau$ is any optimal policy for $x = P(x)$. □

We next establish part (ii) of Proposition 4 for maxPPSs.

**Lemma 6.** *Let $x = P(x)$ be a maxPPS with $0 < q^* < 1$. For any $0 \le y \le q^*$ and $\lambda > 0$, we have $I(y) \le q^*$, and furthermore, if*

$$q^* - y \le \lambda\left(1 - q^*\right),$$

*then*

$$q^* - I(y) \le \frac{\lambda}{2}\left(1 - q^*\right).$$

**Proof.** Let $\tau$ be an optimal policy (which exists by Theorem 1). The least fixed point solution of the PPS $x = P_\tau(x)$ is $q^*$. From our assumptions, Lemma 1 gives that $q^* - \mathcal{N}_\tau(y) \le \frac{\lambda}{2}(1 - q^*)$. However, by Lemma 5, $\mathcal{N}_\tau(y) \le I(y) \le q^*$. The claim follows. □

Proposition 4 for maxPPSs follows from Lemmas 5 and 6.

### 3.4. An Iteration of the GNM for MinPPSs

In this subsection, we will prove the key properties of the operator $I(y)$ for minPPSs (Proposition 4). Our proof of the minPPS version will be somewhat different, because it turns out we cannot use the same argument as

for maxPPSs, based on LPs, to prove that $I(y)$ is well defined. Fortunately, in the case of minPPSs, we can show that $(I - P_\sigma(y))^{-1}$ exists and is nonnegative for *any* policies $\sigma$ at those points $y$ that are of interest. And we can use this to show that there is *some* policy, $\sigma$, such that $I(y)$ is equivalent to an iteration of Newton's method at $y$ after fixing the policy $\sigma$. We shall establish the existence of such a policy using a policy improvement argument, instead of just using the LP, as we did for maxPPSs. (Note that the policy improvement algorithm may not be an efficient (P-time) way to compute it, and we do not claim it is. We use policy improvement only as an argument in the proof of the existence of a suitable policy $\sigma$.)

**Lemma 7.** *For a minPPS $x = P(x)$ with LFP $0 < q^* < 1$, for any $0 \le y \le q^*$ and any policy $\sigma$, $(I - P_\sigma(y))^{-1}$ exists and is nonnegative. Thus, $\mathcal{N}_\sigma(y)$ is defined.*

**Proof.** We show first that the LFP of $x = P_\sigma(x)$, denoted by $q^*_\sigma$, satisfies $q^* \le q^*_\sigma$. To see this, note that by Theorem 1, there is an optimal policy $\tau$ with $q^*_\tau = q^*$. But we defined an optimal policy for a minPPS as one with $q^*_\tau \le q^*_\upsilon$ for any policies $\upsilon$. Therefore, $q^* = q^*_\tau \le q^*_\sigma$.

Because $0 < q^* < 1$ and $0 \le y \le q^* \le q^*_\sigma$, we have $q^*_\sigma > 0$, $0 \le y \le q^*_\sigma$, and $y < 1$. It follows from Lemma A.3 of the appendix, applied to the PPS $x = P_\sigma(x)$, that $(I - P_\sigma(y))^{-1}$ exists and is nonnegative, and hence $\mathcal{N}_\sigma(y)$ is defined. □

**Lemma 8.** *Given a minPPS $x = P(x)$ with LFP $0 < q^* < 1$ and a vector $y$ with $0 \le y \le q^*$, there is a policy $\sigma$ such that $P^y(\mathcal{N}_\sigma(y)) = \mathcal{N}_\sigma(y)$.*

**Proof.** We use a policy (strategy) improvement "algorithm" to prove this. Start with any policy $\sigma_1$. At step $i$, suppose we have a policy $\sigma_i$.

For notational simplicity, in the following, we use the abbreviation $z = \mathcal{N}_{\sigma_i}(y)$. By Lemma 4, $P^y_{\sigma_i}(z) = z$. So we have $P^y(z) \le z$. If $P^y(z) = z$, then *stop*: we are done.

Otherwise, to construct the next strategy $\sigma_{i+1}$, take the smallest $j$ such that $(P^y(z))_j < z_j$. Note that $P_j(x)$ has form M, because otherwise $(P(x))_j = (P_{\sigma_i}(x))_j$. Thus, there is some variable $x_k$ with $P_j(x) = \min\{x_k, x_{\sigma_i(j)}\}$ and $z_k < z_{\sigma_i(j)}$. Define $\sigma_{i+1}$ to be

$$\sigma_{i+1}(l) = \begin{cases} \sigma_i(l) & \text{if } l \ne j, \\ k & \text{if } l = j. \end{cases}$$

Then $(P^y_{\sigma_{i+1}}(z))_j < z_j$, but for every other coordinate $l \ne j$, $(P^y_{\sigma_{i+1}}(z))_l = (P^y_{\sigma_i}(z))_l = z_l$. Thus,

$$P^y_{\sigma_{i+1}}(z) \le z \tag{2}$$

By Lemma 7, $\mathcal{N}_{\sigma_{i+1}}(y)$ is defined. Moreover, the inequality (2), together with Lemma 4(ii), yields that $\mathcal{N}_{\sigma_{i+1}}(y) \le z$. But $\mathcal{N}_{\sigma_{i+1}}(y) \ne z$ because $P^y_{\sigma_{i+1}}(z) \ne z$, whereas, by Lemma 4(i), we have $P^y_{\sigma_{i+1}}(\mathcal{N}_{\sigma_{i+1}}(y)) = \mathcal{N}_{\sigma_{i+1}}(y)$.

Thus, this algorithm gives us a sequence of policies $\sigma_1, \sigma_2 \ldots$ with $\mathcal{N}_{\sigma_1}(y) \ge \mathcal{N}_{\sigma_2}(y) \ge \mathcal{N}_{\sigma_3}(y) \ge \ldots$, where furthermore each step must strictly decrease at least one coordinate of $\mathcal{N}_{\sigma_i}(y)$. It follows that $\sigma_i \ne \sigma_j$, unless $i = j$. There are only finitely many policies. So the sequence must be finite, and the algorithm terminates. But it terminates only when we reach a $\sigma_i$ with $P^y(\mathcal{N}_{\sigma_i}(y)) = \mathcal{N}_{\sigma_i}(y)$. □

We note that the analogous policy improvement algorithm might fail to work for maxPPSs, as we might reach a policy $\sigma_i$ where $(I - P_{\sigma_i}(x))^{-1}$ does not exist or has a negative entry.

The next lemma shows that this policy improvement algorithm always produces a coordinate-wise minimal Newton iterate over all policies.

**Lemma 9.** *For a minPPS $x = P(x)$ with LFP $0 < q^* < 1$, if $0 \le y \le q^*$ and $\sigma$ is a policy such that $P^y(\mathcal{N}_\sigma(y)) = \mathcal{N}_\sigma(y)$, then*
(i) *for any policy $\sigma'$, $\mathcal{N}_{\sigma'}(y) \ge \mathcal{N}_\sigma(y)$;*
(ii) *for any $x \in \mathbb{R}^n$ with $P^y(x) \ge x$, we have $x \le \mathcal{N}_\sigma(y)$;*
(iii) *for any $x \in \mathbb{R}^n$ with $P^y(x) \le x$, we have $x \ge \mathcal{N}_\sigma(y)$;*
(iv) *$\mathcal{N}_\sigma(y)$ is the unique fixed point of $x = P^y(x)$;*
(v) *$\mathcal{N}_\sigma(y) \le q^*$.*

**Proof.** Note first that, by Lemma 7, for any policy $\sigma$, $(I - P'_\sigma(y))^{-1}$ exists and is nonnegative, and $\mathcal{N}_\sigma(y)$ is defined.

(i) Consider $P^y_{\sigma'}(\mathcal{N}_\sigma(y))$. Note that $P^y_{\sigma'}(\mathcal{N}_\sigma(y)) \ge P^y(\mathcal{N}_\sigma(y)) = \mathcal{N}_\sigma(y)$ by assumption. Thus, by Lemma 4(ii), $\mathcal{N}_\sigma(y) \le \mathcal{N}_{\sigma'}(y)$.

(ii) $P^y_\sigma(x) \ge P^y(x) \ge x$, so by Lemma 4(ii), $x \le \mathcal{N}_\sigma(y)$.

(iii) If $P^y(x) \leq x$, then there a policy $\sigma'$ with $P^y_{\sigma'}(x) \leq x$, and by Lemma 4(ii), $x \geq \mathcal{N}_{\sigma'}(y)$. So using part (i) of this lemma, $x \geq \mathcal{N}_{\sigma'}(y) \geq \mathcal{N}_\sigma(y)$.

(iv) By assumption, $\mathcal{N}_\sigma(y)$ is a fixed point of $x = P^y(x)$. We just need uniqueness. If $P^y(q) = q$, then by parts (ii) and (iii) of this lemma, $q \leq \mathcal{N}_\sigma(y)$ and $q \geq \mathcal{N}_\sigma(y)$, that is, $q = \mathcal{N}_\sigma(y)$.

(v) Consider an optimal policy $\tau$ for the minPPS, $x = P(x)$. From Lemma 1, if follows that $\mathcal{N}_\tau(y) \leq q^*_\tau = q^*$. And then part (i) of this lemma, gives us that $\mathcal{N}_\sigma(y) \leq \mathcal{N}_\tau(y) \leq q^*$. □

We can show now part (i) of Proposition 4. Recall the LP that "defines" $I(y)$, for a minPPS:

$$\text{Maximize:} \quad \sum_i a_i, \qquad \text{Subject to:} \quad P^y(a) \geq a. \tag{3}$$

**Lemma 10.** *For a minPPS $x = P(x)$ with LFP $0 < q^* < 1$, and for $0 \leq y \leq q^*$, there is a unique optimal solution, which we call $I(y)$, to the LP (3), and furthermore, $I(y) = \mathcal{N}_\sigma(y)$ for some policy $\sigma$, $P^y(I(y)) = I(y)$, and $I(y) \leq q^*$.*

**Proof.** By Lemma 8, there is a $\sigma$ such that $P^y(\mathcal{N}_\sigma(y)) = \mathcal{N}_\sigma(y)$. So $\mathcal{N}_\sigma(y)$ is a feasible solution of $P^y(a) \geq a$. Let $a$ by any solution of $P^y(a) \geq a$. By Lemma 9(ii), $a \leq \mathcal{N}_\sigma(y)$. Consequently $\sum_{i=1}^n a_i \leq \sum_{i=1}^n (\mathcal{N}_\sigma(y))_i$ with equality only if $a = \mathcal{N}_\sigma(y)$. So $\mathcal{N}_\sigma(y)$ is the unique optimal solution of the LP (3) and $I(y) = \mathcal{N}_\sigma(y)$. By Lemma 9(iv), $I(y) \leq q^*$. □

In the maxPPS case, we had an iteration that was at least as good as iterating with the optimal policy. Here we have an iteration that is at least as bad. Nevertheless, we shall see that it is good enough. In the maxPPS case, the analog of Lemma 1, Lemma 6, thus followed from Lemma 1. Here we crucially need the following stronger result for PPSs than Lemma 1; its proof is given in the appendix.

**Lemma 11.** *If $x = P(x)$ is a PPS and we are given $x, y \in \mathbb{R}^n$ with $0 \leq x \leq y \leq P(y) \leq 1$, and if the conditions*

$$\lambda > 0 \quad \text{and} \quad y - x \leq \lambda(1 - y) \quad \text{and} \quad (I - P'(x))^{-1} \text{ exists and is nonnegative} \tag{4}$$

*hold, then $y - \mathcal{N}(x) \leq \frac{\lambda}{2}(1 - y)$.*

(Note that we cannot conclude that $y - \mathcal{N}(x) \geq 0$.)

We can show now part (ii) of Proposition 4.

**Lemma 12.** *Let $x = P(x)$ be a minPPS with LFP $0 < q^* < 1$. For any $0 \leq x \leq q^*$ and $\lambda > 0$, if*

$$q^* - x \leq \lambda(1 - q^*),$$

then

$$q^* - I(x) \leq \frac{\lambda}{2}(1 - q^*).$$

**Proof.** By Lemma 10, there is a policy $\sigma$ with $I(x) = \mathcal{N}_\sigma(x)$. We then apply Lemma 11 to $x = P_\sigma(x)$, $x$, and $q^*$ instead of $y$. Observe that $P_\sigma(q^*) \geq P(q^*) = q^*$ and that $(I - P'_\sigma(x))^{-1}$ exists and is nonnegative. Thus, the conditions of Lemma 11 hold, and we can conclude that $q^* - \mathcal{N}_\sigma(x) \leq \frac{\lambda}{2}(1 - q^*)$. □

Proposition 4 for minPPSs follows from Lemmas 10 and 12.

### 3.5. Putting It Together

In this subsection, we use the properties shown in the previous subsections to analyze the algorithm and show Theorem 3 and Corollary 1. We show first that all iterations are well defined.

**Lemma 13.** *If we run the rounded-down GNM starting with $x^{(0)} := 0$ on a max/minPPS, $x = P(x)$, with LFP $q^*$, $0 < q^* < 1$, then for all $k \geq 0$, $x^{(k)}$ is well defined, and $0 \leq x^{(k)} \leq q^*$.*

**Proof.** The base case $x^{(0)} = 0$ is immediate for both claims.

For the induction step, suppose the claims hold for $k$, and thus $0 \leq x^{(k)} \leq q^*$. From Proposition 4, $I(x^{(k)})$ is well defined, and $I(x^{(k)}) \leq q^*$. Furthermore, because $x^{(k+1)}$ is obtained from $I(x^{(k)})$ by rounding down all coordinates, except setting to zero any that are negative, and because obviously $q^* > 0$, we have that $0 \leq x^{(k+1)} \leq q^*$. □

We analyze now the running time.

In Etessami et al. [15], we gave a polynomial time algorithm, in the standard Turing model of computation, for approximating the LFP of a PPS, $x = P(x)$, using Newton's method. The proof in Etessami et al. [15] uses induction based on the "halving lemma," Lemma 1. We of course now have suitable halving lemmas for maxPPSs and minPPSs, namely, Lemmas 6 and 12. In Etessami et al. [15], the following bound was used for the base case of the induction:

**Lemma 14** (Etessami et al. [15, theorem 3.14]). *If $0 < q^* < 1$ is the LFP of a PPS, $x = P(x)$, in $n$ variables, then for all $i \in \{1, \ldots, n\}$,*

$$1 - q_i^* \geq 2^{-4|P|}.$$

*In other words, $0 < q_i^* \leq 1 - 2^{-4|P|}$ for all $i \in \{1, \ldots, n\}$.*

We can easily derive from this an analogous lemma for the setting of max/minPPSs.

**Lemma 15.** *If $0 < q^* < 1$ is the LFP of a max/minPPS, $x = P(x)$, in $n$ variables, then for all $i \in \{1, \ldots, n\}$,*

$$1 - q_i^* \geq 2^{-4|P|}.$$

*In other words, $0 < q_i^* \leq 1 - 2^{-4|P|}$ for all $i \in \{1, \ldots, n\}$.*

**Proof.** Let $\tau$ be any optimal policy for $x = P(x)$. We know it exists, by Theorem 1. Lemma 14 gives that $1 - q_i^* \geq 2^{-4|P_\tau|}$. All we need to note is that $|P| \geq |P_\tau|$, which clearly holds using any sensible encoding for $P$ and $P_\tau$, in the sense that we should need no more bits to encode $x_i = x_j$ than to encode $x_i = \max\{x_j, x_k\}$ or $x_i = \min\{x_j, x_k\}$. □

For a vector $v > 0$, we will use the notation $v_{\min}$ to denote its minimum entry. Thus, the lemma says that if $q^* < 1$, then $(1 - q^*)_{\min} \geq 2^{-4|P|}$.

We bound now the distance of the iterates $x^{(k)}$ of the GNM from the LFP $q^*$.

**Lemma 16.** *For a max/minPPS, $x = P(x)$, with LFP $q^*$ such that $0 < q^* < 1$, if we apply the rounded-down GNM with parameter $h$ starting at $x^{(0)} := 0$, then for all $k \geq 0$, we have*

$$\left\| q^* - x^{(k)} \right\|_\infty \leq \left(2^{-k} + 2^{-h+1}\right) 2^{4|P|}.$$

**Proof.** Because $x^{(0)} := 0$,

$$q^* - x^{(0)} = q^* \leq 1 \leq \frac{1}{(1 - q^*)_{\min}} (1 - q^*). \tag{5}$$

For any $k > 0$, if $q^* - x^{(k-1)} \leq \lambda(1 - q^*)$, then by Proposition 4 (which was proved separately for maxPPSs and minPPSs, in Lemmas 6 and 12, respectively), we have

$$q^* - I\left(x^{(k-1)}\right) \leq \left(\frac{\lambda}{2}\right)(1 - q^*). \tag{6}$$

Observe that after every iteration $k > 0$, in every coordinate $i$, we have

$$x_i^{(k)} \geq I\left(x^{(k-1)}\right)_i - 2^{-h}. \tag{7}$$

This holds simply because we are rounding down $I(x^{(k-1)})_i$ by at most $2^{-h}$, unless it is negative, in which case $x_i^{(k)} = 0 > I(x^{(k-1)})_i$. Combining the two inequalities (6) and (7) yields the following inequality:

$$q^* - x^{(k)} \leq \left(\frac{\lambda}{2}\right)(1 - q^*) + 2^{-h}1 \leq \left(\frac{\lambda}{2} + \frac{2^{-h}}{(1 - q^*)_{\min}}\right)(1 - q^*).$$

Taking inequality (5) as the base case (with $\lambda = \frac{1}{(1-q^*)_{\min}}$), it follows by induction on $k$, for all $k \geq 0$, that

$$q^* - x^{(k)} \leq \left(2^{-k} + \sum_{i=0}^{k-1} 2^{-(h+i)}\right) \frac{1}{(1 - q^*)_{\min}} (1 - q^*).$$

But $\sum_{i=0}^{k-1} 2^{-(h+i)} \leq 2^{-h+1}$ and $\frac{\|1-q^*\|_\infty}{(1-q^*)_{\min}} \leq \frac{1}{(1-q^*)_{\min}} \leq 2^{4|P|}$, by Lemma 15. Thus,

$$q^* - x^{(k)} \leq \left(2^{-k} + 2^{-h+1}\right) 2^{4|P|}.$$

Clearly, we have $q^* - x^{(k)} \geq 0$ for all $k$. Thus, we have shown that for all $k \geq 0$,

$$\left\| q^* - x^{(k)} \right\|_\infty \leq \left(2^{-k} + 2^{-h+1}\right) 2^{4|P|}. \quad \square$$

Combining Lemmas 13 and 16, we can prove Theorem 3.

**Proof of Theorem 3.** In Lemma 16, let $k := j + 4|P| + 1$ and $h := j + 2 + 4|P|$. We have $\|q^* - x^{(j+1+4|P|)}\|_\infty \leq 2^{-(j+1)} + 2^{-(j+1)} = 2^{-j}$. □

Corollary 1 follows readily.

**Proof of Corollary 1.** First, we use the algorithms given in Etessami and Yannakakis [11, theorems 11 and 13] or the faster algorithms given in the online companion of this paper to identify those variables $x_i$ with $q_i^* = 0$ or $q_i^* = 1$ in time polynomial in $|P|$. Then we remove these variables from the max/minPPS by substituting their known values into the equations for other variables. This gives us a max/minPPS with LFP $0 < q^* < 1$ and does not increase $|P|$. Then we use the iterated GNM, with rounding down, as outlined earlier in this section. In each iteration of the GNM, we solve an LP. Each LP has at most $n \leq |P|$ variables and at most $2n$ constraints, and the numerators and denominators of each rational coefficient are no larger than $2^{j+2+4|P|}$, so it can be solved in time polynomial in $|P|$ and $j$ using standard algorithms. We need only $1 + 2 + 4|P|$ iterations involving one LP each. Putting back the removed 0 and 1 values into the resulting vector gives us the full result $q^*$. This can all be done in polynomial time. □

## 4. Computing an $\epsilon$-Optimal Policy in P-Time

First let us note that we cannot hope to compute an optimal policy in P-time without a major breakthrough.

**Theorem 4.** *Computing an optimal policy for a max/minPPS is PosSLP-hard.*

**Proof.** Recall from Etessami and Yannakakis [10] and Etessami et al. [15] that the termination (extinction) probability vector $q^*$ of a branching process (or of a one-exit recursive Markov chain) can be equivalently viewed as the LFP of a purely probabilistic PPS, and vice versa.

It was shown in Etessami and Yannakakis [10, theorem 5.3] that, given a PPS (or equivalently, a BP or 1-RMC) and given a rational probability $p$, it is PosSLP-hard to decide whether the LFP is $q_1^* > p$, for a given rational $p$, as well as to decide whether $q_1^* < p$. (In fact, these hardness results hold already even if $p = 1/2$.)

The fact that computing an optimal policy for max/minPPSs is PosSLP-hard follows easily from this: for the case of maxPPSs (minPPSs, respectively), given a PPS $x = P(x)$, and given $p$, we simply add a new variable $x_0$ to the PPS and a corresponding equation:

$$x_0 = \max\{p, x_1\} \quad (\text{respectively, } x_0 = \min\{p, x_1\}). \tag{8}$$

It is clear that $q_1^* > p$ (respectively, $q_1^* < p$) holds for the original PPS if and only if in any optimal policy $\sigma$, for the augmented maxPPS (respectively, minPPS), the policy picks $x_1$ rather than $p$ on the RHS of Equation (8). So, if we could compute an optimal policy for a maxPPS (minPPS), we would be able to decide whether $q_1^* > p$ (respectively, whether $q_1^* < p$). □

Because we cannot hope to compute an optimal policy for max/minPPSs in P-time without a major breakthrough, we will instead seek to find a policy $\sigma$ such that $\|q_\sigma^* - q^*\|_\infty \leq \epsilon$ for a given desired $\epsilon > 0$, in time poly($|P|$, $\log(1/\epsilon)$). We have an algorithm for approximating $q^*$. Can we use a sufficiently close approximation, $q$, to $q^*$ to find such an $\epsilon$-optimal strategy? Once we have an approximation $q$, it seems natural to consider policies $\sigma$ such that $P_\sigma(q) = P(q)$. For minPPSs, this means choosing, for each type M variable $x_i$ with equation of the form $x_i = \min\{x_j, x_k\}$, the variable $x_j$ or $x_k$ that has the lowest value in the approximate vector $q$, and for maxPPSs, choosing the variable that has the highest value in $q$. It turns out that this works for minPPSs (provided that $q$ is sufficiently close to $q^*$), although for maxPPSs, we need to select the policy $\sigma$ more carefully.

Before getting into the details, we outline the basic approach for the algorithm and the proof. For most of this section, we focus on the case when the LFP $q^*$ satisfies $0 < q^* < 1$; at the end of the section, we will extend the policy to the variables that have value 0 or 1 in the LFP. We compute a sufficiently close approximation $q$ of the LFP $q^*$ of the given max/minPPS $x = P(x)$, and let $\sigma$ be a policy such that $P_\sigma(q) = P(q)$. We would like to show that the corresponding LFP $q_\sigma^*$ of the PPS $x = P_\sigma(x)$ is within distance $\epsilon$ of the LFP $q^*$ of the given max/minPPS. We know that $q$ is close to $q^*$; hence, it suffices to show that $q$ is sufficiently close also to $q_\sigma^*$. Toward this purpose, in the first part of the proof, we bound the distance $\|q_\sigma^* - q\|_\infty$ by the norm of $(I - P'_\sigma(x))^{-1}$ for a certain value of $x$ and $\|q^* - q\|_\infty$ (Lemma 19 below). In the second part of the proof, we then use a result from Etessami et al. [15] for PPSs to bound the norm of the matrix $(I - P'_\sigma(x))^{-1}$. For minPPSs, we show that the hypothesis of this result of Etessami et al. [15] is satisfied by any policy $\sigma$ such that $P_\sigma(q) = P(q)$, if $q$ is close enough to $q^*$. For maxPPSs, more effort is required, and we give an algorithm that chooses carefully a policy $\sigma$ so that the hypothesis is satisfied.

We start with a lemma on PPSs, which will then be applied in our case to the PPS $x = P_\sigma(x)$ for an appropriate policy $\sigma$. The proof is given in the appendix.

**Lemma 17.** *If $x = P(x)$ is a PPS with LFP $q^*$ and the matrix $(I - P'(\frac{1}{2}(q^* + y)))^{-1}$ exists, then*

$$q^* - y = \left(I - P'\left(\frac{1}{2}(q^* + y)\right)\right)^{-1}(P(y) - y). \tag{9}$$

The norm of the left-hand side, $\|q^* - y\|$, of Equation (9) in Lemma 17 is bounded by the product of the norms of the matrix and the vector $P(y) - y$ on the right-hand side. We can bound the norm of $P(y) - y$ for a PPS, and more generally for a max/minPPS, in terms of the distance of $y$ from the LFP (see the appendix for the proof).

**Lemma 18.** *If $x = P(x)$ is a max/minPPS with LFP $q^*$, and if $0 \le y \le q^*$, then $\|P(y) - y\|_\infty \le 2\|q^* - y\|_\infty$.*

From the previous two lemmas, we can derive the bound in the following lemma (see the appendix for the proof). For a square matrix $A$, $\rho(A)$ denotes its spectral radius. A basic property is that if $A$ is a nonnegative matrix and $\rho(A) < 1$, then the matrix $I - A$ is nonsingular, and $(I - A)^{-1} = \sum_{k=0}^{\infty} A^k$ is nonnegative (see, e.g., Horn and Johnson [21]).

**Lemma 19.** *For a max/minPPS $x = P(x)$, given $0 \le q \le q^*$ such that $q < 1$ and a policy $\sigma$ such that $P(q) = P_\sigma(q)$ and such that $\rho(P'_\sigma(\frac{1}{2}(q^* + q^*_\sigma))) < 1$, and thus $(I - P'_\sigma(\frac{1}{2}(q^* + q^*_\sigma)))^{-1}$ exists and is nonnegative,*

$$\|q^*_\sigma - q^*\|_\infty \le \left(2\left\|\left(I - P'_\sigma\left(\frac{1}{2}(q^*_\sigma + q^*)\right)\right)^{-1}\right\|_\infty + 1\right)\|q^* - q\|_\infty.$$

To apply Lemma 19, we need to show the existence of the matrix $(I - P'_\sigma(\frac{1}{2}(q^*_\sigma + q^*)))^{-1}$ and bound its norm. For this, we use the following fact for PPS, which is proved in Etessami et al. [15].

**Lemma 20** (Etessami et al. [15, theorem 5.1]). *If $x = P(x)$ is a PPS with LFP $q^* > 0$, then we have the following:*

(i) *If $q^* < 1$ and $0 \le y < 1$, then $\rho(P'(\frac{1}{2}(y + q^*))) < 1$, thus, $(I - P'(\frac{1}{2}(y + q^*)))^{-1}$ exists and is nonnegative, and*

$$\left\|\left(I - P'\left(\frac{1}{2}(y + q^*)\right)\right)^{-1}\right\|_\infty \le 2^{10|P|}\max\left\{2(1 - y)^{-1}_{\min}, 2^{|P|}\right\}.$$

(ii) *If $q^* = 1$, $x = P(x)$ is strongly connected (i.e., every variable depends directly or indirectly on every other), and $0 \le y < 1 = q^*$, then $\rho(P'(y)) < 1$, thus, $(I - P'(y))^{-1}$ exists and is nonnegative, and*

$$\left\|(I - P'(y))^{-1}\right\|_\infty \le 2^{4|P|}\frac{1}{(1 - y)_{\min}}.$$

Applying Lemma 20(i) on the PPS $x = P_\sigma(x)$ and completing the proof have some complications because of the following: although we assume that $0 < q^* < 1$, it need not be true for an arbitrary policy $\sigma$ that $0 < q^*_\sigma < 1$.

**Example 5.** Consider the following maxPPS $x = P(x)$:

$$x_1 = \max(x_2, x_4);\ x_2 = \max(x_1, x_3);\ x_3 = \max(x_2, x_5);\ x_4 = 0.25x_3 + 0.5x_5 + 0.25;\ x_5 = x_1x_4.$$

The LFP is $q^* = (0.5, 0.5, 0.5, 0.5, 0.25)$, and it is achieved by the optimal policy $\tau$ that selects $\tau(x_1) = x_4, \tau(x_2) = x_1, \tau(x_3) = x_2$. Consider, however, the policy $\sigma$ that selects $\sigma(x_1) = x_2, \sigma(x_2) = x_3, \sigma(x_3) = x_2$. The induced PPS $x = P_\sigma(x)$ is

$$x_1 = x_2;\ x_2 = x_3;\ x_3 = x_2;\ x_4 = 0.25x_3 + 0.5x_5 + 0.25;\ x_5 = x_1x_4.$$

Note that $P_\sigma(q^*) = q^*$. However, the LFP of the PPS $x = P_\sigma(x)$ is $q^*_\sigma = (0, 0, 0, 0.25, 0)$. □

But the following proposition obviously does hold.

**Proposition 5.** *Given a max/minPPS $x = P(x)$ with LFP $q^*$ such that $0 < q^* < 1$, for any policy $\sigma$,*

(i) *if $x = P(x)$ is a maxPPS, then $q^*_\sigma < 1$;*
(ii) *if $x = P(x)$ is a minPPS, then $q^*_\sigma > 0$.*

**Proof.** If $x = P(x)$ is a maxPPS, then clearly $q^*_\sigma \le q^* < 1$, because $\sigma$ can be no better than an optimal strategy. Likewise, if $x = P(x)$ is a minPPS, then $0 < q^* \le q^*_\sigma$ for the same reason. □

For maxPPSs, we may have that some coordinate of $q^*_\sigma$ is equal to 0 and for minPPSs we may have that some coordinate of $q^*_\sigma$ is equal to 1, even when $0 < q^* < 1$. This is the source of different complications for the max and min cases, and we give separate proofs for the two cases.

### 4.1. MinPPSs

For minPPSs, we shall show that if $y$ is a sufficiently close approximation to $q^*$, then any policy $\sigma$ with $P(y) = P_\sigma(y)$ is $\epsilon$-optimal. The maxPPS case will not be so simple: the analogous statement is false for maxPPSs.

**Theorem 5.** *If $x = P(x)$ is a minPPS, with LFP $0 < q^* < 1$, and $0 \le \epsilon \le 1$ and $0 \le y \le q^*$ such that $\|q^* - y\|_\infty \le 2^{-14|P|-3}\epsilon$, then for any policy $\sigma$ with $P_\sigma(y) = P(y)$, $\|q^* - q^*_\sigma\|_\infty \le \epsilon$.*

**Proof.** By Proposition 5, $q^*_\sigma \ge q^*$, and so $q^*_\sigma > 0$. Suppose for now that $q^*_\sigma < 1$ (we will show this later). Then, applying Lemma 20(i) for the case where we set $y := q^*$ and the PPS is $x = P_\sigma(x)$ yields that

$$\left\| \left( I - P'_\sigma\left(\frac{1}{2}(q^* + q^*_\sigma)\right)\right)^{-1} \right\|_\infty \le 2^{10|P_\sigma|} \max\left\{ \frac{2}{(1-q^*)_{\min}}, 2^{|P_\sigma|} \right\}.$$

Note that $|P_\sigma| \le |P|$. From Lemma 15, $(1-q^*)_{\min} \ge 2^{-4|P|}$. Thus,

$$\left\| \left( I - P'_\sigma\left(\frac{1}{2}(q^* + q^*_\sigma)\right)\right)^{-1} \right\|_\infty \le 2^{14|P|+1}.$$

Lemma 19 now gives that

$$\|q^* - q^*_\sigma\|_\infty \le \left(2^{14|P|+2} + 1\right)\|q^* - y\|_\infty \le \epsilon.$$

Thus, under the assumption that $q^*_\sigma < 1$, we are done.

To complete the proof, we now show that $q^*_\sigma < 1$. Suppose, for a contradiction, that for some $i$, $(q^*_\sigma)_i = 1$. Then, by results in Etessami and Yannakakis [10], $x = P_\sigma(x)$ (i.e., its dependency graph) has a bottom strongly connected component $S$ with $q^*_S = 1$. If $x_i$ is in $S$, then only variables in $S$ appear in $(P_\sigma)_i(x)$, so we write $x_S = P_S(x)$ for the PPS that is formed by such equations. We also have that $P'_S(1)$ is irreducible and that the least fixed point solution of $x_S = P_S(x_S)$ is $q^*_S = 1$. Take $y_S$ to be the subvector of $y$ with coordinates in $S$. Now, if we apply Lemma 20(ii) to the PPS $x_S = P_S(x)$, by taking the $y$ in its statement to be $\frac{1}{2}(y_S + 1)$, it gives that

$$\left\| \left( I - P'_S\left(\frac{1}{2}(y_S + 1)\right)\right)^{-1} \right\|_\infty \le 2^{4|P_S|} \frac{1}{\frac{1}{2}(1-y_S)_{\min}},$$

but $|P_S| \le |P|$ and $(1-y_S)_{\min} \ge (1-q^*)_{\min} \ge 2^{-4|P|}$. Thus,

$$\left\| \left( I - P'_S\left(\frac{1}{2}(y_S + 1)\right)\right)^{-1} \right\|_\infty \le 2^{8|P|+1}.$$

Lemma 17 gives that

$$1 - y_S = \left( I - P'_S\left(\frac{1}{2}(1 + y_S)\right)\right)^{-1}(P_S(y_S) - y_S).$$

Taking norms and rearranging gives

$$\|P_S(y_S) - y_S)\|_\infty \ge \frac{\|1 - y_S\|_\infty}{\left\|(I - P'_S(\frac{1}{2}(y_S+1)))^{-1}\right\|_\infty} \ge \frac{2^{-4|P|}}{2^{8|P|+1}} \ge 2^{-12|P|-1},$$

however, $\|P_S(y_S) - y_S)\|_\infty \le \|P_\sigma(y) - y\|_\infty$ and $P_\sigma(y) = P(y)$. We deduce that $\|P(y) - y\|_\infty \ge 2^{-12|P|-1}$. Lemma 18 states that $\|P(y) - y\|_\infty \le 2\|q^* - y\|_\infty$. We thus have $\|q^* - y\|_\infty \ge 2^{-12|P|-2}$. This contradicts our assumption that $\|q^* - y\|_\infty \le 2^{-14|P|-3}\epsilon$ for some $\epsilon \le 1$. □

### 4.2. MaxPPSs

Now we proceed to the harder case of maxPPSs. The main theorem in this case is the following.

**Theorem 6.** *If $x = P(x)$ is a maxPPS with $0 < q^* < 1$ and given $0 \le \epsilon \le 1$ and a vector $y$, with $0 \le y \le q^*$, such that $\|q^* - y\|_\infty \le 2^{-14|P|-3}\epsilon$, then we can compute a policy $\sigma$ such that $\|q^* - q^*_\sigma\|_\infty \le \epsilon$ in time polynomial in $|P|$ and $\log(1/\epsilon)$.*

We need a policy $\sigma$ such that we can apply Lemma 20 to $x = P_\sigma(x)$, and for which we can get good bounds on $\|P_\sigma(y) - y\|_\infty$. First we show that such policies exist. In fact, any optimal policy will do: for an optimal policy

$\tau$, $q^*_\tau > 0$ and Lemma 18 applied to $x = P_\tau(x)$ gives that $\|P_\tau(y) - y\|_\infty \leq 2^{-14|P|-2}\epsilon$. Unfortunately, the optimal policy might be hard to find (Theorem 4). Furthermore, if we select any policy $\sigma$ such that $P(y) = P_\sigma(y)$, it is possible that the corresponding LFP $q^*_\sigma$ of the PPS $x = P_\sigma(x)$ has some coordinates equal to 0, and thus we cannot apply directly Lemma 20. To prove the theorem, we will give below an algorithm that computes a suitable policy $\sigma$ such that $q^*_\sigma > 0$.

First, note that given a policy $\sigma$ and the PPS $x = P_\sigma(x)$, we can easily test in polynomial time whether the LFP is $q^*_\sigma > 0$ (see, e.g., Etessami and Yannakakis [10, theorem 2.2]). We shall make use of the following easy fact, shown in the appendix:

**Lemma 21.** *If $x = P(x)$ is a PPS with $n$ variables and with LFP $q^*$, then for any variable index $i \in \{1, \ldots, n\}$, the following are equivalent:*

(i) $q^*_i > 0$;
(ii) *there is a* $k > 0$ *such that* $(P^k(0))_i > 0$;
(iii) $(P^n(0))_i > 0$.

Given the maxPPS, $x = P(x)$, with $0 < q^* < 1$, and given a vector $y$ that satisfies the conditions of Theorem 6, we shall use the following algorithm to obtain the policy we need:

1. Initialize the policy $\sigma$ to any policy such that $P_\sigma(y) = P(y)$.
2. Calculate for which variables $x_i$ in $x = P_\sigma(x)$ we have $(q^*_\sigma)_i = 0$. Let $S_0$ denote this set of variables. (We can do this in P-time; see, for example, Etessami and Yannakakis [10, theorem 2.2].)
3. If for all $i$ we have $(q^*_\sigma)_i > 0$, that is, if $S_0 = \emptyset$, then terminate and output the policy $\sigma$.
4. Otherwise, look for a variable $x_i$, where $P_i(x)$ is of form M, with $P_i(x) = \max\{x_j, x_k\}$, and where $(q^*_\sigma)_i = 0$ but one of $x_j, x_k$, say, $x_j$, has $(q^*_\sigma)_j > 0$, and where, furthermore, $\|y_i - y_j\| \leq 2^{-14|P|-2}\epsilon$. (We shall establish that such a pair $x_i$ and $x_j$ will always exist when we are at this step of the algorithm.)

Let $\sigma'$ be the policy that chooses $x_j$ at $x_i$ but is otherwise identical to $\sigma$. Set $\sigma := \sigma'$ and return to Step 2.

**Example 6.** Consider the maxPPS of Example 5:

$$x_1 = \max(x_2, x_4);\ x_2 = \max(x_1, x_3);\ x_3 = \max(x_2, x_5);\ x_4 = 0.25x_3 + 0.5x_5 + 0.25;\ x_5 = x_1x_4.$$

Let $y$ be a sufficiently close approximation to the LFP $q^* = (0.5, 0.5, 0.5, 0.5, 0.25)$ and suppose that $y_2 > y_4$ and $y_3 > y_1$. The policy $\sigma$ that satisfies $P_\sigma(y) = P(y)$ selects $\sigma(x_1) = x_2, \sigma(x_2) = x_3, \sigma(x_3) = x_2$. As in Example 5, the LFP of the PPS $x = P_\sigma(x)$ is $q^*_\sigma = (0, 0, 0, 0.25, 0)$; the algorithm will compute only the set $S_0 = \{x_1, x_2, x_3, x_5\}$ of variables with value 0 in $q^*_\sigma$, not $q^*_\sigma$ itself. In Step 4, the algorithm will switch the choice for variable $x_1$ because $(q^*_\sigma)_4 > 0$ and $y_1 \approx y_4 \approx 0.5$, and will set $\sigma'(x_1) = x_4$. The new induced PPS $x = P_{\sigma'}(x)$ is

$$x_1 = x_4;\ x_2 = x_3;\ x_3 = x_2;\ x_4 = 0.25x_3 + 0.5x_5 + 0.25;\ x_5 = x_1x_4.$$

It has LFP $q^*_{\sigma'} \approx (0.29, 0, 0, 0.29, 0.085)$ and the new $S_0 = \{x_2, x_3\}$. Even though $x_3$ has a successor, $x_5$, with $(q^*_{\sigma'})_5 > 0$, the algorithm will not switch the choice for $x_3$ because $y_5 \approx q^*_5 = 0.25 \ll y_3 \approx 0.5$. Rather, it will switch the choice for variable $x_2$ and will set $\sigma''(x_2) = x_1$, because $(q^*_{\sigma'})_1 > 0$ and $y_1 \approx 0.5 \approx y_2$. The new induced PPS $x = P_{\sigma''}(x)$ is

$$x_1 = x_4;\ x_2 = x_1;\ x_3 = x_2;\ x_4 = 0.25x_3 + 0.5x_5 + 0.25;\ x_5 = x_1x_4$$

and has LFP $q^*_{\sigma''} = (0.5, 0.5, 0.5, 0.5, 0.25)$, and thus the new $S_0 = \emptyset$. So the algorithm will terminate and output $\sigma''$, which in this case is the optimal policy. □

**Lemma 22.** *The steps of the above algorithm are always well defined, and the algorithm always terminates in at most $n$ iterations with a policy $\sigma$ such that $q^*_\sigma > 0$ and $\|P_\sigma(y) - y\|_\infty \leq 2^{-14|P|-2}\epsilon$.*

**Proof.** First, to show that the steps of the algorithm are always well defined, we need to show that if $q^*_\sigma$ has some coordinate equal to 0, then Step 4 will find a pair of variables $x_i, x_j$ that satisfy the condition of Step 4 to switch the policy at $x_i$.

Suppose that $q^*_\sigma$ has some coordinate equal to 0. Let $\tau$ be an optimal policy; then $q^*_\tau = q^* > 0$. So by Lemma 21, $P^n_\tau(0) > 0$. For any variable $x_j$ with $(P_\tau(0))_j > 0$, the equation $x_j = P_j(x)$ must have form L and not M, so $(P_\sigma(0))_j > 0$ and $(q^*_\sigma)_j > 0$. There must be a least $k$, $k_{\min}$ with $1 < k_{\min} \leq n$, such that there is a variable $x_j$ with $(P^k_\tau(0))_j > 0$ but $(q^*_\sigma)_j = 0$. Let $x_i$ be a variable such that $(P^{k_{\min}}_\tau(0))_i > 0$ but $(q^*_\sigma)_i = 0$. We claim that $x_i$ is of type M.

Suppose that $x_i = P_i(x)$ has form Q. Then $P_i(x) = x_j x_l$ for some variables $x_j$, $x_l$. We have $0 < (P_\tau^{k_{\min}}(0))_i = (P_\tau^{k_{\min}-1}(0))_j (P_\tau^{k_{\min}-1}(0))_l$. So, $(P_\tau^{k_{\min}-1}(0))_j > 0$ and $(P_\tau^{k_{\min}-1}(0))_l > 0$. The minimality of $k_{\min}$ now gives us that $(q_\sigma^*)_j > 0$ and $(q_\sigma^*)_l > 0$. So, $(q_\sigma^*)_i = (q_\sigma^*)_j (q_\sigma^*)_l > 0$. This is a contradiction. Thus, $x_i = P_i(x)$ does not have form Q.

Similarly, $x_i = P_i(x)$ does not have form L. So, $x_i = P_i(x)$ has form M. There are variables $x_j$, $x_l$ with $P_i(x) = \max\{x_j, x_l\}$. Suppose w.l.o.g. that $(P_\tau(x))_i = x_j$. We have $P_\tau^{k_{\min}}(0))_i > 0$, and so $(P^{k_{\min}-1}(0))_j > 0$. By minimality of $k_{\min}$, we have that $(q_\sigma^*)_j > 0$. We have that $(q_\sigma^*)_i = 0$, and so $(P_\sigma(x))_i = x_l$.

Lemma 18 applied to the system $x = P_\tau(x)$ gives that $\|P_\tau(y) - y\|_\infty \le 2^{-14|P|-2}\epsilon$. So $|y_i - y_j| = |y_i - (P_\tau(y))_i| \le 2^{-14|P|-2}\epsilon$. Thus, Step 4 could use $x_i$ and change the policy $\sigma$ at $x_i$ (i.e., switch $\sigma(i)$) from $x_l$ to $x_j$.

Next, we need to show that the algorithm terminates.

**Claim 2.** *If Step 4 switches the variable $x_i$ with $P_i(x) = \max\{x_j, x_k\}$ from $(P_\sigma(x))_i = x_k$ to $(P_{\sigma'}(x))_i = x_j$, then*
(i) $q_{\sigma'}^* \ge q_\sigma^*$,
(ii) $(q_{\sigma'}^*)_i > 0$,
(iii) *the set of variables $x_l$ with $(q_{\sigma'}^*)_l > 0$ is a strict superset of the set of variables $x_l$ with $(q_\sigma^*)_l > 0$.*

**Proof.** Recall that Step 4 will switch only if $(q_\sigma^*)_i = 0$ and $(q_\sigma^*)_j > 0$.

(i) We show that, for any $t > 0$, $P_{\sigma'}^t(0) \ge P_\sigma^t(0)$. We use induction on $t$. The base case $t = 1$ is clear because the only indices $i$ where $P_i(0) \ne 0$ are when $P_i(0)$ has form L, in which case $P_i(0) = (P_{\sigma'}(0))_i = (P_\sigma(0))_i$. For the inductive case, note first that $P_\sigma(x)$ and $P_{\sigma'}(x)$ differ only on the $i$th coordinate. We have $(q_\sigma^*)_i = 0$, so for any $t$, $(P_\sigma^t(0))_i = 0$. Suppose that $P_{\sigma'}^t(0) \ge P_\sigma^t(0)$. Then, by monotonicity, $P_{\sigma'}^{t+1}(0) \ge P_{\sigma'}(P_\sigma^t(0))$. But $(P_{\sigma'}(P_\sigma^t(0)))_r = (P_\sigma^{t+1}(0))_r$ when $r \ne i$. Furthermore, $(P_{\sigma'}(P_\sigma^t(0)))_i \ge 0 = (P_\sigma^{t+1}(0))_i$. So $P_{\sigma'}(P_\sigma^k(0)) \ge P_\sigma^{k+1}(0)$. We thus have that $P_{\sigma'}^{t+1}(0) \ge P_\sigma^{t+1}(0)$.

We know that as $t \to \infty$, $P_{\sigma'}^t(0) \to q_{\sigma'}^*$ and $P_\sigma^t(0) \to q_\sigma^*$. So, $q_{\sigma'}^* \ge q_\sigma^*$.

(ii) We have $(q_{\sigma'}^*)_i = (q_{\sigma'}^*)_j$. By (i), $(q_{\sigma'}^*)_j \ge (q_\sigma^*)_j$. We chose $x_j$ such that $(q_\sigma^*)_j > 0$. So, $(q_{\sigma'}^*)_i > 0$.

(iii) If $(q_\sigma^*)_l > 0$, then by (i), $(q_{\sigma'}^*)_l > 0$. Also, $(q_\sigma^*)_i = 0$, and by (ii), $(q_{\sigma'}^*)_i > 0$. □

Thus, if at some stage of the algorithm we do not yet have $q_\sigma^* > 0$, then Step 4 always gives us a new $\sigma'$ with more coordinates having $(q_{\sigma'}^*)_i > 0$. This can happen at most $n$ times. Furthermore, note that if $\|P_\sigma(y) - y\|_\infty \le 2^{-14|P|-2}\epsilon$, then $\|P_{\sigma'}(y) - y\|_\infty \le 2^{-14|P|-2}\epsilon$. Our starting policy has $\|P_\sigma(y) - y\|_\infty = \|P(y) - y\|_\infty \le 2^{-14|P|-2}\epsilon$. The algorithm terminates in at most $n$ iterations and gives a $\sigma$ with $q_\sigma^* > 0$ and $\|P_\sigma(y) - y\|_\infty \le 2^{-14|P|-2}\epsilon$. □

We can now complete the proof of Theorem 6.

**Proof of Theorem 6.** Using the algorithm, we find a $\sigma$ with $\|y - P_\sigma(y)\|_\infty \le 2^{-14|P|-2}\epsilon$ and $q_\sigma^* > 0$. By Proposition 5, $q_\sigma^* < 1$. Also, $y < 1$ because $y \le q^*$ and $q^* < 1$. Applying Lemma 20(i) to the PPS $x = P_\sigma(x)$ and point $y$ gives that $(I - P'_\sigma(\frac{1}{2}(y + q_\sigma^*)))^{-1}$ exists and

$$\left\| \left( I - P'_\sigma\left(\frac{1}{2}\left(y + q_\sigma^*\right)\right)\right)^{-1} \right\|_\infty \le 2^{10|P_\sigma|} \max\left\{ \frac{2}{(1-y)_{\min}}, 2^{|P_\sigma|} \right\}.$$

We have $|P_\sigma| \le |P|$. From the fact that there always exists an optimal policy and from Lemma 15 (theorem 3.14 of Etessami et al. [15]), it follows that $(1 - q^*)_{\min} \ge 2^{-4|P|}$, and because $y \le q^*$, we have $(1-y)_{\min} \ge 2^{-4|P|}$. So

$$\left\| \left( I - P'_\sigma\left(\frac{1}{2}\left(y + q_\sigma^*\right)\right)\right)^{-1} \right\|_\infty \le 2^{14|P|+1}. \tag{10}$$

We cannot use Lemma 19 as stated because we need not have $P(y) = P_\sigma(y)$. We will use Lemma 17 instead. As observed above, the matrix $(I - P'_\sigma(\frac{1}{2}(y + q_\sigma^*)))^{-1}$ exists. Applying Lemma 17 to the PPS $x = P_\sigma(x)$, and taking norms, we get the inequality

$$\|q_\sigma^* - y\|_\infty \le \left\| \left( I - P'_\sigma\left(\frac{1}{2}\left(q_\sigma^* + y\right)\right)\right)^{-1} \right\|_\infty \|P_\sigma(y) - y\|_\infty. \tag{11}$$

From Lemma 22 we have

$$\|P_\sigma(y) - y\|_\infty \le 2^{-14|P|-2}\epsilon. \tag{12}$$

Combining (10), (11), and (12) yields

$$\|q_\sigma^* - y\|_\infty \le \frac{1}{2}\epsilon,$$

so $\|q_\sigma^* - q^*\|_\infty \le \|q_\sigma^* - y\|_\infty + \|q^* - y\|_\infty \le \frac{1}{2}\epsilon + 2^{-14|P|-3}\epsilon \le \epsilon$. □

We can extend the policy to the variables that have value 0 or 1 in the LFP and get an $\epsilon$-optimal policy for any maxPPS or minPPS.

**Theorem 7.** *Given a max/minPPS $x = P(x)$, and given $\epsilon > 0$, we can compute an $\epsilon$-optimal policy for $x = P(x)$ in time poly($|P|$, log$(1/\epsilon)$).*

**Proof.** First, we use the algorithms from Etessami and Yannakakis [11] or the more efficient algorithms from the online companion of this paper to detect variables $x_i$ with $q_i^* = 0$ or $q_i^* = 1$ in time polynomial in $|P|$. Then we can remove these from the max/minPPS by substituting the known values into the equations for other variables. This gives us a max/minPPS with least fixed point $0 < q^* < 1$ and does not increase $|P|$. To use either Theorem 6 or Theorem 5, it suffices to have a $y$ with $y \leq q^*$ with $q^* - y \leq 2^{-14|P|-3}\epsilon$. Theorem 3 says that we can find such a $y$ in time polynomial in $|P|$ and $14|P| - \log(\epsilon)$, which is polynomial in $|P|$ and $\log(1/\epsilon)$ as required. Now, depending on whether we have a maxPPS or minPPS, Theorem 6 or Theorem 5 shows that from this $y$, we can find an $\epsilon$-optimal policy for the max/minPPS with $0 < q^* < 1$ in time polynomial in $|P|$ and $\log(1/\epsilon)$. All that is left to show is that we can extend this policy to the removed variables $x_i$, where $q_i^* = 0$ or $q_i^* = 1$ while still remaining $\epsilon$-optimal.

We next show how this can be done. For a minPPS, if $q_i^* = 1$, then for any policy $\sigma$, $(q_\sigma^*)_i = 1$, so the choice made at such variables $x_i$ is irrelevant. Similarly, for maxPPSs, when $q_i^* = 0$, any choice at $x_i$ is optimal.

For a minPPS with $q_i^* = 0$, if $P_i(x)$ has form M, we can choose any variable $x_j$ with $q_j^* = 0$. There is such a variable: if $P_i(x) = \min\{x_j, x_k\}$ and $q_i^* = 0$, then either $q_j^* = 0$ or $q_k^* = 0$. Let $\sigma$ be a policy such that for each variable $x_i$ with $q_i^* = 0$, $(q^*)_{\sigma(i)} = 0$. We need to show that $(q_\sigma^*)_i = 0$ for all such variables. Suppose that, for some $k \geq 0$, $(P_\sigma^k(0))_i = 0$ for all $x_i$ such that $q_i^* = 0$. Then $P(P_\sigma^k(0))_i = 0$ for all $x_i$ with $q_i^* = 0$.

To see why this is so, note that whether $P_i(z) = 0$ depends only on which coordinates of $z$ are zero, and furthermore, if $P_i(z) = 0$ when the set of zero coordinates of $z$ is $S$, then for any vector $z'$ where the zero coordinates of $z'$ are $S' \supseteq S$, we have $P_i(z') = 0$. Because the coordinates $S$ that are zero in $q^*$ are a subset of the coordinates $S'$ that are zero in $P_\sigma^k(0)$, and we have $P_i(q^*) = q_i^* = 0$, we thus have $P(P_\sigma^k(0))_i = 0$.

If $P_i(x) = \min\{x_j, x_k\}$ and $q_i^* = 0$, then either $q_j^* = 0$ or $q_k^* = 0$. Suppose w.l.o.g. that $(P_\sigma(x))_i = x_j$. Then $q_j^* = 0$, so by assumption, $(P_\sigma^k(0))_j = 0$, and so $(P_\sigma(P_\sigma^k(0)))_i = 0$. We now have enough for $(P_\sigma^{k+1}(0))_i = 0$ for each variable $x_i$ with $q_i^* = 0$. We have $P_\sigma^0(0) = 0$, so by induction for all $k \geq 0$, $(P_\sigma^k(0))_i = 0$ for all $x_i$ with $q_i^* = 0$. From this, for each variable $x_i$ with $q_i^* = 0$, $(q_\sigma^*)_i = 0$.

The case of a maxPPS that has variables with $q_i^* = 1$ is not so simple. Although it is again the case that if a variable $x_i$ of type M with $P_i(x) = \max\{x_j, x_k\}$ has value $q_i^* = 1$ in the LFP, then at least one of the variables $x_j, x_k$ has also value 1 (i.e., $q_j^* = 1$ or $q_k^* = 1$), if both variables are 1 in $q^*$, the policy $\sigma$ cannot choose arbitrarily one of them for $x_i$, because then it is possible that in the resulting LFP $q_\sigma^*$ the variable $x_i$ does not have value 1 (it can get value 0 in fact). Thus, for a maxPPS, more care is needed to choose the policy for the variables with value $q_i^* = 1$. The P-time algorithm given in Etessami and Yannakakis [11] to compute the variables with $q_i^* = 1$ produces also a randomized policy for these variables (lemma 12 in Etessami and Yannakakis [11]). In the online companion for this paper, we give an improved algorithm that produces a *pure* (nonrandomized) policy for these variables (and does so much faster than the algorithm of Etessami and Yannakakis [11]). □

## 5. Approximating the Value of BSSGs in FNP

In this section, we briefly note that, as an easy corollary of our results for BMDPs, we can obtain a TFNP (total NP search problem) upper bound for computing (approximately) the *value* of BSSGs, where the objective of the two players is to maximize, and minimize, the extinction probability. For relevant definitions and background results about these games, see Etessami and Yannakakis [11]. It suffices for our purposes here to point out that, as shown in Etessami and Yannakakis [11], the value of these games (which are determined) is characterized by the LFP solution of associated min-maxPPSs, $x = P(x)$, where both min and max operators can occur in the equations for different variables. Furthermore, both players have optimal policies (i.e., optimal pure, memoryless strategies) in these games (see Etessami and Yannakakis [11]).

**Corollary 2.** *Given a max-minPPS $x = P(x)$, and given a rational $\epsilon > 0$, the problem of approximating the LFP $q^*$ of $x = P(x)$, that is, computing a vector $v$ such that $\|q^* - v\|_\infty \leq \epsilon$, is in TFNP, as is the problem of computing $\epsilon$-optimal policies for both players. (And thus, also, the problem of approximating the value and computing $\epsilon$-optimal strategies for BSSGs is in TFNP.)*

**Proof.** Let $x = P(x)$ be the max-minPPS whose LFP, $q^*$, we wish to compute. First guess pure policies $\sigma$ and $\tau$ for the max and min players, respectively. Then, fix $\sigma$ as max's strategy, and for the resulting minPPS (with LFP $q_\sigma^*$), use our algorithm to compute in P-time an approximate value vector $v_\sigma \leq q_\sigma^*$ such that $\|v_\sigma - q_\sigma^*\|_\infty \leq \epsilon/2$. Next,

fix $\tau$ as min's strategy, and for the resulting maxPPS (with LFP $q^*_\tau$), use our algorithm to compute in P-time an approximate value vector $v_\tau \leq q^*_\tau$ such that $\|v_\tau - q^*_\tau\|_\infty \leq \epsilon/2$. Finally, check whether $\|v_\sigma - v_\tau\|_\infty \leq \epsilon/2$. If not, then reject this "guess." If so, then output $\sigma$ and $\tau$ as $\epsilon$-optimal policies for max and min, respectively, and output $v := v_\sigma$ as an $\epsilon$-approximation of the LFP, $q^*$.

We show the correctness of the algorithm. First, we need to show that an output is produced for at least one guess. Indeed, consider the guess where $\sigma, \tau$ are optimal strategies for the two players. Then $q^*_\sigma = q^* = q^*_\tau$, and both $v_\sigma$ and $v_\tau$ are $\leq q^*$ and within $\epsilon/2$ of $q^*$. Hence, $\|v_\sigma - v_\tau\|_\infty \leq \epsilon/2$, and the algorithm will output $\sigma, \tau$, and $v_\sigma$.

Second, we need to show that for every guess of the algorithm that results in an output, the output is correct; that is, $\sigma, \tau$ are $\epsilon$-optimal policies, and the value $v_\sigma$ that is output is within $\epsilon$ of $q^*$. First, note that $q^*_\sigma \leq q^* \leq q^*_\tau$. Because $v_\sigma \leq q^*_\sigma \leq q^*$, we have

$$\begin{aligned}\left\|q^* - q^*_\sigma\right\|_\infty \leq \left\|q^* - v_\sigma\right\|_\infty &\leq \left\|q^*_\tau - v_\sigma\right\|_\infty \\ &\leq \left\|q^*_\tau - v_\tau\right\|_\infty + \left\|v^*_\tau - v_\sigma\right\|_\infty \\ &\leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon.\end{aligned}$$

Hence, $\sigma$ is an $\epsilon$-optimal policy for the max player, and $v_\sigma$ is within $\epsilon$ of $q^*$. Because $v_\sigma \leq q^* \leq q^*_\tau$ and $\|q^*_\tau - v_\sigma\|_\infty \leq \epsilon$, it follows also that $\|q^*_\tau - q^*\|_\infty \leq \epsilon$, that is, $\tau$ is an $\epsilon$-optimal policy for the min player. □

It is worth noting that the problem of approximating the value of a BSSG game to within a desired $\epsilon > 0$, when $\epsilon$ is given as part of the input, is already at least as hard as computing the *exact* value of Condon's [4] finite-state simple stochastic games (SSGs), and thus one cannot hope for a P-time upper bound without a breakthrough. In fact, it was shown in Etessami and Yannakakis [11] that even the *qualitative* problem of deciding whether the value $q^*_i = 1$ for a given BSSG (or max-minPPS), which was shown there to be in NP∩coNP, is already at least as hard as Condon's [4] *quantitative* decision problem for finite-state simple stochastic games. (Whereas for finite-state SSGs, the qualitative problem of deciding whether the value is 1 is in P-time.)

## 6. Conclusions

We have provided the first polynomial time algorithms for computing optimal (maximum and minimum) extinction probabilities of branching MDPs to arbitrary desired accuracy $\epsilon > 0$, as well as for computing $\epsilon$-optimal policies for extinction. We have done so by providing a P-time algorithm for computing the least fixed point solution for systems of probabilistic max/min polynomial Bellman equations (max/minPPSs) to within desired accuracy $\epsilon > 0$. Our algorithms are based on a novel generalization of Newton's method applied to max/minPPSs.

Extinction probabilities are important quantities for the analysis of multitype branching processes, and they play a key role in various other analyses of such stochastic processes (see, e.g., Harris [20]). It may thus be expected that efficient algorithms for other analyses of BMDPs may be facilitated by the algorithms we have developed in this paper. Indeed, in a more recent work (Etessami et al. [14]) that builds directly on this paper,[7] we have shown that computing optimal *reachability* probabilities for BMDPs can be computed in P-time to desired precision. (By *reachability* probability in a BMDP, we mean the (optimal) probability that, starting from a given population, the population will eventually contain an object of a designated type.) We have done so by showing that optimal *nonreachability* probabilities constitute the *greatest fixed point* (*GFP*) of (different) max/minPPSs that we can associate with a BMDP, and by showing that a modification of the generalized Newton method developed in this paper can be used to compute the GFP of max/minPPSs to desired precision in P-time. It would be interesting to find other classes of infinite-state MDPs and other systems of max/min polynomial equations (perhaps even some nonmonotone ones) where variants of the GNM are applicable and yield efficient algorithms.

Finally, our focus in this paper has been on establishing provably polynomial time algorithms for optimal extinction probabilities for BMDPs. Our algorithms are relatively simple to implement, and it will be interesting to empirically evaluate their practical performance. For many MDP models, *value iteration* and *policy iteration* provide practically efficient iterative methods, although their worst-case behavior is (in some cases) known to be poor or is not adequately understood. It is indeed possible, and natural, to consider both value iteration and (suitable approximate versions of) policy iteration for BMDPs by exploiting their max/minPPS Bellman equations. It can be shown that both methods converge to the optimal extinction probabilities for BMDPs. Theoretically, these methods inherit worst-case lower bounds from finite-state MDPs with reachability objectives, as well as worst-case lower bounds that arise already for value iteration for multitype branching processes (Etessami and Yannakakis [10]), as explained in the introduction. More detailed (theoretical

and practical) analysis of the behavior of value and policy iteration methods for BMDPs, and comparison of their practical performance with our P-time algorithms, could be interesting.

## Appendix. Omitted Material from Sections 2–4

### Omitted Material from Section 2

**Proof of Proposition 1.** We can easily convert, in P-time, any max/minPPS into SNF, using the following procedure.

- For each equation $x_i = P_i(x) = \max\{p_1(x), \ldots, p_m(x)\}$, for each $p_j(x)$ on the right-hand side that is not a variable, add a new variable $x_k$, replace $p_j(x)$ with $x_k$ in $P_i(x)$, and add the new equation $x_k = p_j(x)$. Do similarly if $P_i(x) = \min\{p_1(x), \ldots, p_m(x)\}$.
- If $P_i(x) = \max\{x_{j_1}, \ldots, x_{j_m}\}$ with $m > 2$, then add $m-2$ new variables $x_{i_1}, \ldots, x_{i_{m-2}}$; set $P_i(x) = \max\{x_{j_1}, x_{i_1}\}$; and add the equations $x_{i_1} = \max\{x_{j_2}, x_{i_2}\}$, $x_{i_2} = \max\{x_{j_3}, x_{i_3}\}$, …, $x_{i_{m-2}} = \max\{x_{j_{m-1}}, x_{j_m}\}$. Do similarly if $P_i(x) = \min\{x_{j_1}, \ldots, x_{j_m}\}$ with $m > 2$.
- For each equation $x_i = P_i(x) = \sum_{j=1}^{m} p_j x^{\alpha_j}$, where $P_i(x)$ is a probabilistic polynomial that is not just a constant or a single monomial, replace every monomial $x^{\alpha_j}$ on the right-hand side that is not a single variable by a new variable $x_{i_j}$ and add the equation $x_{i_j} = x^{\alpha_j}$.
- For each variable $x_i$ that occurs in some polynomial with exponent higher than 1, introduce new variables $x_{i_1}, \ldots, x_{i_k}$, where $k$ is the logarithm of the highest exponent of $x_i$ that occurs in $P(x)$, and add equations $x_{i_1} = x_i^2$, $x_{i_2} = x_{i_1}^2$, …, $x_{i_k} = x_{i_{k-1}}^2$. For every occurrence of a higher power $x_i^l$, $l > 1$, of $x_i$ in $P(x)$, if the binary representation of the exponent $l$ is $a_k \ldots a_2 a_1 a_0$, then we replace $x_i^l$ by the product of the variables $x_{i_j}$ such that the corresponding bit $a_j$ is 1, and $x_i$ if $a_0 = 1$. After we perform this replacement for all the higher powers of all the variables, every polynomial of total degree >2 is just a product of variables.
- If a polynomial $P_i(x) = x_{j_1}, \ldots x_{j_m}$ in the current system is the product of $m > 2$ variables, then add $m-2$ new variables $x_{i_1}, \ldots, x_{i_{m-2}}$; set $P_i(x) = x_{j_1} x_{i_1}$; and add the equations $x_{i_1} = x_{j_2} x_{i_2}$, $x_{i_2} = x_{j_3} x_{i_3}$, …, $x_{i_{m-2}} = x_{j_{m-1}} x_{j_m}$.

Now all equations are of the form L, Q, or M.

The above procedure allows us to convert any max/minPPS into one in SNF by introducing $O(|P|)$ new variables and blowing up the size of $P$ by a constant factor $O(1)$. Furthermore, there is an obvious (and easy to compute) bijection between policies for the resulting SNF max/minPPS and the original max/minPPS. □

We will give in the rest of this section some lemmas on PPSs that we will need in this paper. As usual, we always assume, w.l.o.g., that PPSs are in SNF.

**Lemma A.1** (See Etessami et al. [15, lemma 3.3]). *For any PPS $x = P(x)$ in SNF with $n$ variables and any pair of vectors $a, b \in \mathbb{R}^n$, $P(a) - P(b) = P'(\frac{a+b}{2})(a-b)$.*

**Lemma A.2.** *Given a PPS $x = P(x)$ with LFP $q^* > 0$, if $0 \le y \le q^*$ and if $(I - P'(y))^{-1}$ exists and is nonnegative (in which case clearly $\mathcal{N}(y)$ is defined), then $\mathcal{N}(y) \le q^*$ holds.*[8]

**Proof.** In lemma 3.4 of Etessami et al. [15], it was established that when $(I - P'(y))$ is nonsingular, that is, $(I - P'(y))^{-1}$ is defined, and thus $\mathcal{N}(y)$ is defined, then

$$q^* - \mathcal{N}(y) = (I - P'(y))^{-1} \frac{P'(q^*) - P'(y)}{2} (q^* - y). \tag{A.1}$$

Now, because all polynomials in $P(x)$ have nonnegative coefficients, it follows that the Jacobian $P'(x)$ is monotone in $x$, and thus, because $y \le q^*$, we have that $P'(q^*) \ge P'(y)$. Thus, $(P'(q^*) - P'(y)) \ge 0$, and by assumption, $(q^* - y) \ge 0$. Thus, by the assumption that $(I - P'(y))^{-1} \ge 0$, we have by Equation (A.1) that $q^* - \mathcal{N}(y) \ge 0$, that is, that $q^* \ge \mathcal{N}(y)$. □

We also will need the following, which is a less immediate consequence of results in Etessami et al. [15]. Recall that for a square matrix $A$, $\rho(A)$ denotes its spectral radius.

**Lemma A.3.** *Given a PPS $x = P(x)$ with LFP $q^* > 0$, if $0 \le y \le q^*$, and $y < 1$, then $\rho(P'(y)) < 1$, and $(I - P'(y))^{-1}$ exists and is nonnegative.*

The proof of this lemma is more involved. We first recall several closely related results established in our previous papers. Recall that a PPS, $x = P(x)$, is called *strongly connected*, if its variable dependency graph $H$ is strongly connected.

**Lemma A.4** (Etessami and Yannakakis [10, lemma 6.5]).[9] *Let $x = P(x)$ be a strongly connected PPS, in $n$ variables, with LFP $q^* > 0$. For any vector $0 \le y < q^*$, $\rho(P'(y)) < 1$, and thus $(I - P'(y))^{-1}$ exists and is nonnegative.*

**Lemma A.5** (Etessami et al. [15, theorem 3.7]). *For any PPS $x = P(x)$ in SNF that has LFP $0 < q^* < 1$, for all $0 \le y \le q^*$, $\rho(P'(y)) < 1$, and $(I - P'(y))^{-1}$ exists and is nonnegative.*

**Proof of Lemma A.3.** Consider a PPS $x = P(x)$ with LFP $q^* > 0$ and a vector $0 \le y \le q^*$ such that $y < 1$. Note that all we need to establish is that $\rho(P'(y)) < 1$, because it then follows by standard facts (see, e.g., Horn and Johnson [21]) that $(I - P'(y))^{-1}$ exists and is equal to $\sum_{i=0}^{\infty} (P'(y))^i \ge 0$.

Let us first show that if $x = P(x)$ is strongly connected, then $\rho(P'(y)) < 1$. To see this, note that if $x = P(x)$ is strongly connected, then every variable depends on every other, and thus if there exists any $i \in \{1, \ldots, n\}$ such that $q_i^* < 1$, then it must be the case that for

all $j \in \{1,\ldots,n\}$, we have $q_j^* < 1$. Thus, either $q^* = 1$, or else $0 < q^* < 1$. If $q^* = 1$, then because $y < 1$, we have $y < q^*$, and thus, by Lemma A.4, we have $\rho(P'(y)) < 1$. If, on the other hand, $0 < q^* < 1$, then, because $0 \le y \le q^*$, by Lemma A.5, we have $\rho(P'(y)) < 1$.

Next, consider an arbitrary PPS, $x = P(x)$, that is not necessarily strongly connected. Recall the variable dependency graph $H$ of $x = P(x)$. We can partition the variables into sets $S_1,\ldots,S_k$, which form the strongly connected components (SCCs) of $H$. Consider the Directed Acyclic Graph (DAG), $D$, of SCCs, whose nodes are the sets $S_i$, and for which there is an edge from $S_i$ to $S_j$ iff in the dependency graph $H$ there is a node $i' \in S_i$ with an edge to a node in $j' \in S_j$.

Consider the matrix $P'(y)$. Our aim is to show that $\rho(P'(y)) < 1$. Because we assume $q^* > 0$, $0 \le y \le q^*$, and $y < 1$, it clearly suffices to show that $\rho(P'(y)) < 1$ holds in the case where we additionally insist that $y > 0$, because then for any other $z$ such that $0 \le z \le y$, we would have $\rho(P'(z)) \le \rho(P'(y)) < 1$.

So, assuming also that $y > 0$, consider the $n \times n$-matrix $P'(y)$. To keep notation clean, we let $A := P'(y)$. For the $n \times n$ matrix $A$, we can consider its underlying *dependency* graph, $H = (\{1,\ldots,n\}, E_H)$, whose nodes are $\{1,\ldots,n\}$, and where there is an edge from $i$ to $j$ iff $A_{i,j} > 0$. Notice, however, that because $y > 0$, this graph is precisely the same graph as the dependency graph $H$ of $x = P(x)$, and thus it has the same SCCs and the same DAG of SCCs, $D$. Let us sort the SCCs, so that we can assume $S_1,\ldots,S_k$ are topologically sorted with respect to the partial ordering defined by the DAG $D$. In other words, for any variable indices $i \in S_a$ and $j \in S_b$, if $(i,j) \in E_H$, then $a \le b$.

Let $S \subseteq \{1,\ldots,n\}$ be any nonempty subset of indices, and let $A[S]$ denote the principle submatrix of $A$ defined by indices in $S$. It is a well-known fact that $0 \le \rho(A[S]) \le \rho(A)$ (see, e.g., corollary 8.1.20 of Horn and Johnson [21]).

Because $A \ge 0$, $\rho(A)$ is an eigenvalue of $A$ and has an associated nonnegative eigenvector $v \ge 0$, $v \ne 0$ (again, see chapter 8 of Horn and Johnson [21]). In other words,

$$Av = \rho(A)v.$$

First, if $\rho(A) = 0$, then we are of course trivially done. So we can assume w.l.o.g. that $\rho(A) > 0$. Now, if $v_i > 0$, then for every $j$ such that $(j,i) \in E_H$, we have $(Av)_j > 0$, and thus, because $(Av)_j = \rho(A)v_j$, we have $v_j > 0$. Hence, repeating this argument, if $v_i > 0$, then for every $j$ that has a path to $i$ in the dependency graph $H$, we have $v_j > 0$.

Because $v \ne 0$, it must be the case that there is exists some SCC, $S_c$, of $H$ such that for every variable index $i \in S_c$, $v_i > 0$, and furthermore, such that $c$ is the maximum index for such an SCC in the topologically sorted list $S_1,\ldots,S_k$, that is, such that for all $d > c$, and for all $j \in S_d$, we have $v_j = 0$.

First, let us note that it must be the case that $S_c$ is a *nontrivial* SCC. Specifically, let us call an SCC $S_r$ of $H$ *trivial* if $S_r = \{i\}$ consists of only a single variable index, $i$, and furthermore, such that $\mathbf{0} = (A)_i = (P'(y))_i$, that is, that row $i$ of the matrix $A$ is all zero. This cannot be the case for $S_c$, because for any variable $i \in S_c$, we have $v_i > 0$, and thus $(Av)_i = \rho(A)v_i > 0$.

Let us consider the principal submatrix $A[S_c]$ of $A$. We claim that $\rho(A[S_c]) = \rho(A)$. To see why this is the case, note that $Av = \rho(A)v$, and for every $i \in S_c$, we have $(Av)_i = \sum_j a_{i,j}v_j = \rho(A)v_i$. But $v_j = 0$ for every $j \in S_d$ such that $d > c$, and furthermore, $a_{i,j} = 0$ for every $j \in S_{d'}$ such that $d' < c$.

Thus, if we let $v_{S_c}$ denote the subvector of $v$ corresponding to the indices in $S_c$, then we have just established that $A[S_c]v_{S_c} = \rho(A)v_{S_c}$, and thus that $\rho(A[S_c]) \ge \rho(A)$. But because $A[S_c]$ is a principal submatrix of $A$, we also know easily (see, e.g., corollary 8.1.20 of Horn and Johnson [21]) that $\rho(A[S_c]) \le \rho(A)$, so $\rho(A[S_c]) = \rho(A)$.

We are almost done. Given the original PPS, $x = P(x)$, for any subset $S \subseteq \{1,\ldots,n\}$ of variable indices, let $x_S = P_S(x_S, x_{D_S})$ denote the subsystem of $x = P(x)$ associated with the vector $x_S$ of variables in set $S$, where $x_{D_S}$ denotes the variables not in $S$.

Now, note that $x_{S_c} = P_{S_c}(x_{S_c}, y_{D_{S_c}})$ is itself a PPS. Furthermore, it is a *strongly connected* PPS, precisely because $S_c$ is a strongly connected component of the dependency graph $H$, and because $y > 0$. Moreover, the Jacobian matrix of $P_{S_c}(x_{S_c}, y_{D_{S_c}})$, evaluated at $y_{S_c}$, which we denote by $P'_{S_c}(y)$, is precisely the principal submatrix $A[S_c]$ of $A$. Because $x_{S_c} = P_{S_c}(x_{S_c}, y_{D_{S_c}})$ is a strongly connected PPS, we have already argued that it must be the case that $\rho(P'_{S_c}(y)) < 1$. Thus, because $P'_{S_c}(y) = A[S_c]$, we have $\rho(A[S_c]) = \rho(A) < 1$. This completes the proof. □

## Omitted Material from Section 3

**Proof of Lemma 2.** We need to show that the Jacobian $(P^y)'(x)$ of $P^y(x)$, evaluated anywhere, is equal to $P'(y)$. If $x_i = P_i(x)$ is not of form Q, then, for any $x \in \mathbb{R}^n$, $P_i(x) = P_i^y(x)$. So for any $x_j$, $\frac{\partial P_i^y(x)}{\partial x_j} = \frac{\partial P_i(x)}{\partial x_j}$. Otherwise, $x_i = P_i(x)$ has form Q, that is, $P_i(x) = x_j x_k$ for some variables $x_j, x_k$. Then $P_i^y(x) = y_j x_k + x_j y_k - y_j y_k$. In this case, $\frac{\partial P_i^y(x)}{\partial x_j} = y_k$ and $\frac{\partial P_i^y(x)}{\partial x_k} = y_j$. But when $x = y$, $\frac{\partial P_i(x)}{\partial x_j} = y_k$ and $\frac{\partial P_i(x)}{\partial x_k} = y_j$. Furthermore, clearly for any $x_l$, with $l \ne j$ and $l \ne k$, $\frac{\partial P_i(x)}{\partial x_l} = 0$ and $\frac{\partial P_i^y(x)}{\partial x_l} = 0$. We have thus established that $(P^y)'(x) = P'(y)$ for any $x \in \mathbb{R}^n$. □

**Proof of Lemma 3.** First, note that $P^y(x) = P^y(y) + (P^y)'(x)(x - y)$, because the functions $P_i^y(x)$ are all linear in $x$. Next, observe that $P_i(y) = P_i^y(y)$, for all $i$, and thus that $P(y) = P^y(y)$. Thus, to show that $P^y(x) = P^y(y) + P'(y)(x - y) = P(y) + P'(y)(x - y)$, all we need to show is that the Jacobian $(P^y)'(x)$ of $P^y(x)$, evaluated anywhere, is equal to $P'(y)$. But this was established in Lemma 2. □

**Proof of Lemma 4.**

(i) We define

$$a = y + \left(I - P'_\sigma(y)\right)^{-1}\left(P_\sigma(y) - y\right) \equiv \mathcal{N}_\sigma(y).$$

Then we can rearrange this expression, reversibly, yielding

$$\begin{aligned} a = y + (I - P'_\sigma(y))^{-1}(P_\sigma(y) - y) &\Leftrightarrow P_\sigma(y) - y - (I - P'_\sigma(y))(a - y) = 0 \\ &\Leftrightarrow P_\sigma(y) + P'_\sigma(y)(a - y) = a \\ &\Leftrightarrow P^y_\sigma(a) = a \quad \text{(by Lemma 3).} \end{aligned}$$

Uniqueness follows from the reversibility of these transformations.

(ii) First, we shall observe that the result of applying Newton's method to solve $x = P^y_\sigma(x)$ with any initial point $x$ gives us $\mathcal{N}_\sigma(y) = a$ in a single iteration. Recalling from Lemma 2 that the equality $(P^y_\sigma)'(x) = P'_\sigma(y)$ holds between the Jacobians, one iteration of Newton's method applied to $x = P^y_\sigma(x)$ can be equivalently defined as

$$\begin{aligned} x + (I - P'_\sigma(y))^{-1}(P^y_\sigma(x) - x) &= x + (I - P'_\sigma(y))^{-1}(P_\sigma(y) + P'_\sigma(y)(x - y) - x) \\ &= (I - P'_\sigma(y))^{-1}(x - P'_\sigma(y)x + P_\sigma(y) + P'_\sigma(y)(x - y) - x) \\ &= (I - P'_\sigma(y))^{-1}(P_\sigma(y) - P'_\sigma(y)y) \\ &= (I - P'_\sigma(y))^{-1}((I - P'_\sigma(y))y + P_\sigma(y) - y) \\ &= y + (I - P'_\sigma(y))^{-1}(P_\sigma(y) - y) \\ &= \mathcal{N}_\sigma(y). \end{aligned}$$

We thus have $\mathcal{N}_\sigma(y) = x + (I - P'_\sigma(y))^{-1}(P^y_\sigma(x) - x)$. By assumption, $(I - P'_\sigma(y))^{-1}$ is a nonnegative matrix. So if $P^y_\sigma(x) - x \geq 0$, then $\mathcal{N}_\sigma(y) \geq x$, whereas if $P^y_\sigma(x) - x \leq 0$, then $\mathcal{N}_\sigma(y) \leq x$. □

**Proof of Lemma 11.** First, we show that $P'(y)(1 - y) \leq (1 - y)$. Clearly, for any PPS, $P(1) \leq 1$. Note that because by assumption $y \leq P(y)$, we have $(1 - y) \geq (1 - P(y)) \geq (P(1) - P(y))$. Then, by Lemma A.1 (lemma 3.3 of Etessami et al. [15]),

$$(1 - y) \geq P(1) - P(y) = P'\left(\frac{1 + y}{2}\right)(1 - y) \tag{A.2}$$

$$\geq P'(y)(1 - y). \tag{A.3}$$

Again, by Lemma A.1, $P(y) - P(x) = \frac{1}{2}(P'(x) + P'(y))(y - x)$, and thus

$$P(x) = P(y) - \frac{1}{2}(P'(x) + P'(y))(y - x). \tag{A.4}$$

Thus,

$$\begin{aligned} y - \mathcal{N}(x) &= y - x - (I - P'(x))^{-1}(P(x) - x) \\ &= y - x - (I - P'(x))^{-1}\left(P(y) - x - \frac{1}{2}(P'(x) + P'(y))(y - x)\right) \quad \text{(by (A.4))} \\ &\leq y - x - (I - P'(x))^{-1}\left(y - x - \frac{1}{2}(P'(x) + P'(y))(y - x)\right) \\ &= (y - x) - (I - P'(x))^{-1}\left((y - x) - \frac{1}{2}(P'(x) + P'(y))(y - x)\right) \\ &= \left(I - (I - P'(x))^{-1}\left(I - \frac{1}{2}(P'(x) + P'(y))\right)\right)(y - x) \\ &= \left((I - P'(x))^{-1}(I - P'(x)) - (I - P'(x))^{-1}\left(I - \frac{1}{2}(P'(x) + P'(y))\right)\right)(y - x) \\ &= (I - P'(x))^{-1}\left(I - P'(x) - \left(I - \frac{1}{2}(P'(x) + P'(y))\right)\right)(y - x) \\ &= (I - P'(x))^{-1}\left(-P'(x) + \frac{1}{2}(P'(x) + P'(y))\right)(y - x) \\ &= (I - P'(x))^{-1}\frac{1}{2}(P'(y) - P'(x))(y - x) \\ &\leq \frac{\lambda}{2}(I - P'(x))^{-1}(P'(y) - P'(x))(1 - y) \quad \text{(by (4), and because } (P'(y) - P'(x)) \geq 0) \\ &\leq \frac{\lambda}{2}(I - P'(x))^{-1}(I - P'(x))(1 - y) \quad \text{(because by (A.3), } P'(y)(\mathbf{1} - y) \leq (\mathbf{1} - y)) \\ &= \frac{\lambda}{2}(1 - y). \quad \square \end{aligned}$$

## Omitted Material from Section 4

**Proof of Lemma 17.** Lemma A.1 tells us that for any PPS, $x = P(x)$ (assumed to be in SNF), and any pair of vectors $a, b \in \mathbb{R}^n$, we have $P(a) - P(b) = P'((a+b)/2)(a-b)$. Applying this lemma with $a = q^*$ and $b = y$, we have that

$$q^* - P(y) = P'\left(\left(\frac{1}{2}\right)(q^* + y)\right)(q^* - y).$$

Subtracting both sides from $q^* - y$, we have that

$$P(y) - y = \left(I - P'\left(\frac{1}{2}(q^* + y)\right)\right)(q^* - y). \tag{A.5}$$

Multiplying both sides of Equation (A.5) by $(I - P'((1/2)(q^* + y)))^{-1}$, we obtain

$$q^* - y = \left(I - P'\left(\frac{1}{2}(q^* + y)\right)\right)^{-1}(P(y) - y)$$

as required. □

**Proof of Lemma 18.** We show first the lemma for a PPS, and then extend it to a max/minPPS. Suppose that $x = P(x)$ is a PPS (recall, in SNF). By Lemma A.1, we have that $q^* - P(y) = P'(\frac{1}{2}(y + q^*))(q^* - y)$. Because $\frac{1}{2}(y + q^*) \le 1$, $\|P'(\frac{1}{2}(y + q^*))\|_\infty \le 2$: If the $i$th row has $x_i = P_i(x)$ of type L, then $\sum_{j=1}^n |p_{i,j}| \le 1$, and if $x_i = P_i(x)$ has type Q, that is, $P_i(x) = x_j x_k$ for some $j, k$, then $\sum_{j=1}^n |\frac{\partial P_i(x)}{\partial x_j}(\frac{1}{2}(y + q^*))| = \frac{1}{2}(y_j + q_j^*) + \frac{1}{2}(y_k + q_k^*) \le 2$. So we have that $\|q^* - P(y)\|_\infty \le \|P'(\frac{1}{2}(y + q^*))\|_\infty \|q^* - y\|_\infty \le 2\|q^* - y\|_\infty$.

Because $y \le q^*$, we know also that $P(y) \le q^* = P(q^*)$ because $P(x)$ is monotone. If $(P(y))_i \le y_i$, then $y_i - P(y)_i \le q_i^* - P(y)_i \le \|q^* - P(y)\|_\infty \le 2\|q^* - y\|_\infty$. If $P_i(y) \ge y_i$, $P_i(y) - y_i \le q_i^* - y_i \le \|q^* - y\|_\infty$. So $\|P(y) - y\|_\infty \le 2\|q^* - y\|_\infty$ as required.

Suppose now that $x = P(x)$ is a max/minPPS. Then it has some optimal policy, $\tau$, and from the above, $\|P_\tau(y) - y\|_\infty \le 2\|q^* - y\|_\infty$. For type L and type Q variables $x_i$, we have $(P_\tau)_i(y) = P_i(y)$. It thus only remains to show that $|P_i(y) - y_i| \le 2\|q^* - y\|_\infty$ when $x_i$ is of type M.

If $P_i(y) \ge y_i$, then this follows easily: as before, we have that $P_i(y) - y_i \le q_i^* - y_i \le \|q^* - y\|_\infty$. Suppose that instead we have $P_i(y) \le y_i$. Then we consider the two cases (min and max) separately to bound $|P_i(y) - y_i| = y_i - P_i(y)$.

Suppose $x = P(x)$ is a minPPS, and that $P_i(x) = \min\{x_j, x_k\}$. Because $q^* = P(q^*)$, we have

$$0 \le y_i - P_i(y) \le q_i^* - P_i(y) = \min\left\{q_j^*, q_k^*\right\} - P_i(y). \tag{A.6}$$

We can assume, w.l.o.g., that $P_i(y) \equiv \min\{y_j, y_k\} = y_j$. (The case where $P_i(y) = y_k$ is entirely analogous.) Then, by (A.6), we have

$$0 \le y_i - P(y)_i \le \min\left\{q_j^*, q_k^*\right\} - y_j \le q_j^* - y_j \le \left\|q^* - y\right\|_\infty.$$

Suppose now that $x = P(x)$ is a maxPPS and that $P_i(x) \equiv \max\{x_j, x_k\}$. Again, we are already assuming that $P_i(y) \le y_i$. Because $q^* = P(q^*)$, we have

$$0 \le y_i - P_i(y) \le q_i^* - P_i(y) = P_i(q^*) - \max\{y_j, y_k\}. \tag{A.7}$$

We can assume w.l.o.g. that $P_i(q^*) \equiv \max\{q_j^*, q_k^*\} = q_j^*$. (Again, the case when $P_i(q^*) = q_k^*$ is entirely analogous.) Then, by (A.7), we have

$$0 \le y_i - P_i(y) \le q_j^* - \max\{y_j, y_k\} \le q_j^* - y_j \le \left\|q^* - y\right\|_\infty$$

This completes the proof of the Lemma for all max/minPPSs. □

**Proof of Lemma 19.** We will apply Lemma 17 to the PPS $x = P_\sigma(x)$ (which has LFP $q_\sigma^*$), with $q$ in place of $y$. For this we need to show that $(I - P'_\sigma(\frac{1}{2}(q_\sigma^* + q)))^{-1}$ exists. Note that because $0 \le q \le q^*$, we have $0 \le P'_\sigma(\frac{1}{2}(q_\sigma^* + q)) \le P'_\sigma(\frac{1}{2}(q_\sigma^* + q^*))$, and thus $0 \le \rho(P'_\sigma(\frac{1}{2}(q_\sigma^* + q))) \le \rho(P'_\sigma(\frac{1}{2}(q_\sigma^* + q^*)) < 1$. Therefore, $(I - (P'_\sigma(\frac{1}{2}(q_\sigma^* + q)))^{-1}$ also exists and is nonnegative. Lemma 17 gives $q_\sigma^* - q = (I - P'_\sigma(\frac{1}{2}(q_\sigma^* + q)))^{-1}(P_\sigma(q) - q)$. Taking norms, we obtain the following inequality:

$$\left\|q_\sigma^* - q\right\|_\infty \le \left\|\left(I - P'_\sigma\left(\frac{1}{2}(q_\sigma^* + q)\right)\right)^{-1}\right\|_\infty \left\|P_\sigma(q) - q\right\|_\infty. \tag{A.8}$$

Using the fact that $P_\sigma(q) = P(q)$ and Lemma 18, we have

$$\begin{aligned}
\|q^* - q^*_\sigma\|_\infty &\le \|q^* - q\|_\infty + \|q^*_\sigma - q\|_\infty \\
&\le \|q^* - q\|_\infty + \left\|\left(I - P'_\sigma\left(\frac{1}{2}(q^*_\sigma + q)\right)\right)^{-1}\right\|_\infty \|P_\sigma(q) - q\|_\infty \\
&= \|q^* - q\|_\infty + \left\|\left(I - P'_\sigma\left(\frac{1}{2}(q^*_\sigma + q)\right)\right)^{-1}\right\|_\infty \|P(q) - q\|_\infty \\
&\le \|q^* - q\|_\infty + \left\|\left(I - P'_\sigma\left(\frac{1}{2}(q^*_\sigma + q)\right)\right)^{-1}\right\|_\infty 2\|q^* - q\|_\infty \\
&= \left(2\left\|\left(I - P'_\sigma\left(\frac{1}{2}(q^*_\sigma + q)\right)\right)^{-1}\right\|_\infty + 1\right)\|q^* - q\|_\infty \\
&\le \left(2\left\|\left(I - P'_\sigma\left(\frac{1}{2}(q^*_\sigma + q^*)\right)\right)^{-1}\right\|_\infty + 1\right)\|q^* - q\|_\infty
\end{aligned}$$

The last inequality follows because $q \le q^*$, and

$$0 \le \left(I - P'_\sigma(q^*_\sigma + q)\right)^{-1} = \sum_{i=0}^{\infty}\left(P'_\sigma(q^*_\sigma + q)\right)^i \le \sum_{i=0}^{\infty}\left(P'_\sigma(q^*_\sigma + q^*)\right)^i = \left(I - P'_\sigma(q^*_\sigma + q^*)\right)^{-1}. \quad \square$$

**Proof of Lemma 21.**

(i) $\Rightarrow$ (ii) From Etessami and Yannakakis [10], $P^k(0) \to q^*$ as $k \to \infty$. It follows that if $(P^k(0))_i = 0$ for all $k$, then $q^*_i = 0$.

(ii) $\Rightarrow$ (iii) From Etessami and Yannakakis [10], $P^k(0)$ is monotonically nondecreasing in $k$; that is, if $m \ge l > 0$; then $P^m(0) \ge P^l(0)$. Thus, if $(P^k(0))_i > 0$ for some $k \le n$, then $(P^n(0))_i > 0$.

Whether $P_i(x) > 0$ depends only on whether each $x_j > 0$ or not and not on the value of $x_j$. So, for any $k$, whether $(P^{k+1}(0))_i > 0$ depends only on the set $S_k = \{x_j | (P^k(0))_j > 0\}$. Because $P^{k+1}(0) \ge P^k(0)$, we have $S_{k+1} \supseteq S_k$. If ever we have that $S_{k+1} = S_k$, then for any $j$, $(P^{k+2}(0))_j > 0$ whenever $(P^{k+1}(0))_j > 0$, so $S_{k+2} = S_{k+1} = S_k$. Proper containment $S_{k+1} \supset S_k$ can occur only for $n$ values of $k$ as there are only $n$ variables to add. Consequently, $S_{n+1} = S_n$ and so $S_m = S_n$ whenever $m > n$. So if we have a $k > n$ with $(P^k(0))_i > 0$, then $(P^n(0))_i > 0$.

(iii) $\Rightarrow$ (i) By monotonicity and an easy induction, $q^* \ge P^k(0)$ for all $k > 0$. In particular, $q^* \ge P^n(0)$. So $q^*_i \ge (P^n(0))_i > 0$. $\square$

## Endnotes

[1] A preliminary extended abstract for this paper appeared in Etessami et al. [13].

[2] Let us mention, however, that, more recently, in Etessami et al. [15], we also showed that the quantitative decision problem for the extinction probability of BPs, and for the LFP of PPSs, is in fact polynomial time equivalent to PosSLP.

[3] Theorem 2 of Etessami and Yannakakis [11] is stated in the more general context of one-exit recursive simple stochastic games and shows that also for max-minPPSs, both the max player and the min player have optimal policies that attain the LFP $q^*$.

[4] The restriction to having only one entry node is not important; any multientry RMDP can be efficiently transformed to a one-entry RMDP. The restriction to one-exit is very important: multiexit RMDPs lead to undecidable termination problems, even for any nontrivial approximation of the optimal values (Etessami and Yannakakis [11]).

[5] The online companion of this paper provides more efficient P-time algorithms for this.

[6] Note that we do not constrain the variables $a$ to be nonnegative in the mathematical programs corresponding to the operator $I(y)$ for both maxPPSs and minPPSs.

[7] An extended abstract for this paper appeared earlier in Etessami et al. [13].

[8] Note that the lemma does not claim that $\mathcal{N}(y) \ge 0$ holds. Indeed, it may not.

[9] Lemma 6.5 of Etessami and Yannakakis [10] is actually a more general result, relating to strongly connected MPSs that arise from more general RMCs.

## References

[1] Allender E, Bürgisser P, Kjeldgaard-Pedersen J, Miltersen PB (2009) On the complexity of numerical analysis. *SIAM J. Comput.* 38(5): 1987–2006.

[2] Brázdil T, Brozek V, Etessami K, Kucera A (2011) Approximating the termination value of one-counter MDPs and stochastic games. *Proc. 38th Internat. Colloquium Automata, Languages and Programming*, Lecture Notes in Computer Science, vol. 6756 (Springer, New York), 332–343.

[3] Brázdil T, Brozek V, Forejt V, Kucera A (2008) Reachability in recursive Markov decision processes. *Inform. Comput.* 206(5):520–537.

[4] Condon A (1992) The complexity of stochastic games. *Inform. Comput.* 96(2):203–224.

[5] Courcoubetis C, Yannakakis M (1998) Markov decision processes and regular events. *IEEE Trans. Automatic Control* 43(10):1399–1418.

[6] Denardo E, Rothblum U (2005) Totally expanding multiplicative systems. *Linear Algebra Appl.* 406:142–158.

[7] Esparza J, Kiefer S, Luttenberger M (2010) Computing the least fixed point of positive polynomial systems. *SIAM J. Comput.* 39(6):2282–2355.

[8] Esparza J, Kučera A, Mayr R (2006) Model checking probabilistic pushdown automata. *Logical Methods Comput. Sci.* 2(1):1–31.

[9] Esparza J, Gawlitza T, Kiefer S, Seidl H (2008) Approximative methods for monotone systems of min-max-polynomial equations. *Proc. 35th Internat. Colloquium Automata, Languages and Programming*, Lecture Notes in Computer Science, vol. 5125 (Springer, New York), 698–710.
[10] Etessami K, Yannakakis M (2009) Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *J. ACM* 56(1):1–66.
[11] Etessami K, Yannakakis M (2015) Recursive Markov decision processes and recursive stochastic games. *J. ACM* 62(2):1–69.
[12] Etessami K, Stewart A, Yannakakis M (2012) Polynomial-time algorithms for multi-type branching processes and stochastic context-free grammars. *Proc. 44th ACM Sympos. Theory Comput.* (ACM, New York).
[13] Etessami K, Stewart A, Yannakakis M (2012) Polynomial-time algorithms for branching Markov decision processes, and probabilistic min(max) polynomial Bellman equations. *Proc. 39th Internat. Colloquium Automata, Languages and Programming*, Lecture Notes in Computer Science, vol. 7391 (Springer, New York).
[14] Etessami K, Stewart A, Yannakakis M (2015) Greatest fixed points of probabilistic min/max polynomial equations, and reachability for branching Markov decision processes. *Proc. 42nd Internat. Colloquium Automata, Languages Programming*, Lecture Notes in Computer Science, vol. 9135 (Springer, New York).
[15] Etessami K, Stewart A, Yannakakis M (2017) A polynomial-time algorithm for computing extinction probabilities of multi-type branching processes. *SIAM J. Comput.* 46(5):1515–1553.
[16] Etessami K, Wojtczak D, Yannakakis M (2008) Recursive stochastic games with positive rewards. Aceto L, Damgård I, Goldberg LA, Halldórsson MM, Ingólfsdóttir A, Walukiewicz I, eds. *Automata, Languages and Programming*, Lecture Notes in Computer Science, vol. 5125 (Springer, Berlin), 711–723.
[17] Etessami K, Wojtczak D, Yannakakis M (2010) Quasi-birth–death processes, tree-like QBDs, probabilistic 1-counter automata, and pushdown systems. *Performance Evaluation* 67(9):837–857.
[18] Feinberg E, Shwartz A, eds. (2002) *Handbook of Markov Decision Processes: Methods and Applications* (Springer, New York).
[19] Haccou P, Jagers P, Vatutin VA (2005) *Branching Processes: Variation, Growth, and Extinction of Populations* (Cambridge University Press, Cambridge, UK).
[20] Harris TE (1963) *The Theory of Branching Processes* (Springer-Verlag, Berlin).
[21] Horn RA, Johnson CR (1985) *Matrix Analysis* (Cambridge University Press, Cambridge, UK).
[22] Kimmel M, Axelrod DE (2002) *Branching Processes in Biology* (Springer, New York).
[23] Kolmogorov AN, Sevastyanov BA (1947) The calculation of final probabilities for branching random processes [in Russian]. *Doklady* 56: 783–786.
[24] Pliska S (1976/77) Optimization of multitype branching processes. *Management Sci.* 23(2):117–124.
[25] Puterman ML (1994) *Markov Decision Processes* (Wiley, Hoboken, NJ).
[26] Rothblum U, Whittle P (1982) Growth optimality for branching Markov decision chains. *Math. Oper. Res.* 7(4):582–601.
[27] Stewart A, Etessami K, Yannakakis M (2015) Upper bounds for Newton's method on monotone polynomial systems, and P-time model checking of probabilistic one-counter automata. *J. ACM* 62(4):1–33.