Algorithm for Evolutionarily Stable Strategies Against Pure Mutations[*]

Sam Ganzfried
Ganzfried Research
sam@ganzfriedresearch.com

Abstract

Evolutionarily stable strategy (ESS) is an important solution concept in game theory which has been applied frequently to biology and even cancer. Finding such a strategy has been shown to be difficult from a theoretical complexity perspective. Informally an ESS is a strategy that if followed by the population cannot be taken over by a mutation strategy that is initially rare. We present an algorithm for the case where mutations are restricted to pure strategies. This is the first positive result for computation of ESS, as all prior results are for computational hardness.

1   Introduction

While Nash equilibrium has emerged as the standard solution concept in game theory, it is often criticized as being too weak: often games contain multiple Nash equilibria (sometimes even infinitely many), and we want to select one that satisfies other natural properties. For example, one popular concept that refines Nash equilibrium is evolutionarily stable strategy (ESS). A mixed strategy in a two-player symmetric game is an evolutionarily stable strategy if, informally, it is robust to being overtaken by a mutation strategy. Formally, mixed strategy x^* is an ESS if for every mixed strategy x that differs from x^*, there exists $\epsilon_0 = \epsilon_0(x) > 0$ such that, for all $\epsilon \in (0, \epsilon_0)$,

$$(1-\epsilon)u_1(x, x^*) + \epsilon u_1(x, x) < (1-\epsilon)u_1(x^*, x^*) + \epsilon u_1(x^*, x). \quad (1)$$

From a biological perspective, we can interpret x^* as a distribution among "normal" individuals within a population, and consider a mutation that makes use of strategy x, assuming that the proportion of the mutation in the population is ϵ. In an ESS, the expected payoff of the mutation is smaller than the expected payoff of a normal individual, and hence the proportion of mutations will decrease and eventually disappear over time, with the composition of the population returning to being mostly x^*. An ESS is therefore a mixed strategy of the column player that is immune to being overtaken by mutations. ESS was initially proposed by mathematical biologists motivated by applications such as population dynamics (e.g., maintaining robustness to mutations within a population of humans or animals) (Maynard Smith

and Price 1973; Maynard Smith 1982). A common example game is the 2x2 game where strategies correspond to an "aggressive" Hawk or a "peaceful" Dove strategy. A paper has recently proposed a similar game in which an aggressive malignant cell competes with a passive normal cell for biological energy, which has applications to cancer eradication (Dingli et al. 2009).

While Nash equilibrium is defined for general multiplayer games, ESS is defined specifically for two-player symmetric games. ESS is a refinement of Nash equilibrium. In particular, if x^* is an ESS, then (x^*, x^*) (i.e., the strategy profile where both players play x^*) is a (symmetric) Nash equilibrium (Maschler, Solan, and Zamir 2013). Of course the converse is not necessarily true (not every symmetric Nash equilibrium is an ESS), or else ESS would be a trivial refinement. In fact, ESS is not guaranteed to exist in games with more than two pure strategies per player (while Nash equilibrium is guaranteed to exist in all finite games (Nash 1950)). For example, while rock-paper-scissors has a mixed strategy Nash equilibrium (which puts equal weight on all three actions), it has no ESS (Maschler, Solan, and Zamir 2013) (that work considers a version where the payoffs are 1 for a victory, 0 for loss, and $\frac{2}{3}$ for a tie).

There exists a polynomial-time algorithm for computing Nash equilibrium in two-player zero-sum games, while for two-player general-sum and multiplayer games computing a Nash equilibrium is PPAD-complete (Chen and Deng 2006; 2005) and it is widely conjectured that no efficient (polynomial-time) algorithm exists. However, several algorithms have been devised that perform well in practice (Berg and Sandholm 2017; Porter, Nudelman, and Shoham 2008; Govindan and Wilson 2003; Sandholm, Gilpin, and Conitzer 2005; Lemke and Howson 1964). For ESS, there have been some recent hardness results as well (note that in general computing an ESS is at least as hard as computing a Nash equilibrium since it is a refinement). The problem of computing whether a game has an ESS was shown to be both NP-hard and CO-NP hard and also to be contained in Σ_2^P (the class of decision problems that can be solved in non-deterministic polynomial time given access to an NP oracle) (Etessami and Lochbihler 2008). Subsequently it was shown that the exact complexity of this problem is that it is Σ_2^P-complete (Conitzer 2013). Note that this result is for the complexity of determining whether an ESS exists in a

---

given game (as discussed above there exist games which have no ESS), and not for the complexity of actually computing an ESS in games for which one exists. Furthermore we are not aware of any algorithms or heuristics for approximating an ESS (while several exist for Nash equilibrium despite the computational hardness). In this work we describe the first positive result for ESS computation. We present an algorithm that computes an ESS that satisfies a less restrictive condition that only pure strategy mutations cannot successfully invade the population (while standard ESS rules out all mixed strategy mutations). The algorithm is based on modeling the problem as a mixed-integer non-convex quadratically-constrained feasibility program.

2 Evolutionarily stable strategies against pure mutations

We define an *Evolutionarily Stable Strategy against Pure Mutations* (ESSPM) as in Equation 1 except that we only require that the inequality holds for all pure strategies x that differ from x^*. In order to model the problem of computing an ESSPM as an optimization problem, we recall a widely-known alternative definition of ESS that has been proven to be equivalent to the initial one (Maschler, Solan, and Zamir 2013).

Definition 1. *A mixed strategy x^* in a two-player symmetric game is an evolutionarily stable strategy (ESS) if for every mixed strategy x that differs from x^* there exists $\epsilon_0 = \epsilon_0(x) > 0$ such that, for all $\epsilon \in (0, \epsilon_0)$,*

$$(1-\epsilon)u_1(x,x^*)+\epsilon u_1(x,x) < (1-\epsilon)u_1(x^*,x^*)+\epsilon u_1(x^*,x).$$

Theorem 1. *A strategy x^* is evolutionarily stable if and only if for each $x \neq x^*$ exactly one of the following conditions holds:*

- $u_1(x,x^*) < u_1(x^*,x^*)$
- $u_1(x,x^*) = u_1(x^*,x^*)$ *and* $u_1(x,x) < u_1(x^*,x)$

Definition 2. *A mixed strategy x^* in a two-player symmetric game is an evolutionarily stable strategy against pure mutations (ESSPM) if for every pure strategy x that differs from x^* there exists $\epsilon_0 = \epsilon_0(x) > 0$ such that, for all $\epsilon \in (0, \epsilon_0)$,*

$$(1-\epsilon)u_1(x,x^*)+\epsilon u_1(x,x) < (1-\epsilon)u_1(x^*,x^*)+\epsilon u_1(x^*,x). \tag{2}$$

Theorem 2. *A strategy x^* is evolutionarily stable against pure mutations if and only if for each pure strategy $x \neq x^*$ exactly one of the following conditions holds:*

- $u_1(x,x^*) < u_1(x^*,x^*)$
- $u_1(x,x^*) = u_1(x^*,x^*)$ *and* $u_1(x,x) < u_1(x^*,x)$

The proof of Theorem 2 follows from similar reasoning as the proof of Theorem 1 (Maschler, Solan, and Zamir 2013).

(Note that technically the statements of Theorem 1 and 2 can be logically weakened to require that only "at least" as opposed to "exactly" one of the conditions holds, since it is impossible for both to hold simultaneously.)

There are also known results that allow us to categorize ESS with respect to the more common solution concept Nash equilibrium (Maschler, Solan, and Zamir 2013):

Theorem 3. *If x^* is a an evolutionarily stable strategy in a two-player symmetric game, then (x^*, x^*) is a symmetric Nash equilibrium of the game.*

Theorem 4. *In a symmetric game, if (x^*, x^*) is a strict symmetric Nash equilibrium then x^* is an evolutionarily stable strategy.*

Recall that in a Nash equilibrium x^*, we have that $u_1(x,x^*) \leq u_1(x^*,x^*)$ for all mixed strategies x (and similarly for the other players). In a strict equilibrium this requirement changes to $u_1(x,x^*) < u_1(x^*,x^*)$.

Clearly every ESS is also an ESSPM, and so therefore ESS is a refinement of ESSPM. We can straightforwardly show a similar result to Theorem 3 for ESSPM. This shows that ESSPM is a refinement of Nash equilibrium.

Theorem 5. *If x^* is a an ESSPM in a two-player symmetric game, then (x^*, x^*) is a symmetric Nash equilibrium.*

Proof. Consider taking the limit as $\epsilon \to 0$ in Equation 2. From the continuity of the utility function (which is a standard assumption in game theory), this gives $u_1(x,x^*) \leq u_1(x^*,x^*)$ for all pure strategies x. This condition is sufficient to show that x^* is a Nash equilibrium; if a player could profitably deviate from x^* to a mixed strategy x', then for all the pure strategies in the support of x' he would obtain higher payoff against x^* than by following x^*, which contradicts the fact that $u_1(x,x^*) \leq u_1(x^*,x^*)$ for all pure strategies. So (x^*, x^*) is a Nash equilibrium, and is symmetric because the strategies for the players are the same. $\square$

We can also show a result similar to Theorem 4:

Theorem 6. *In a symmetric game, if (x^*, x^*) is a strict symmetric Nash equilibrium then x^* is an ESSPM.*

Proof. If (x^*, x^*) is a strict symmetric Nash equilibrium, then x^* satisfies the first condition from Theorem 2 for all mixed strategies, and therefore also satisfies the condition for all pure strategies x. Therefore, x^* is an ESSPM. $\square$

	A	B	C
A	(2,2)	(1,2)	(1,2)
B	(2,1)	(0,0)	(4,4)
C	(2,1)	(4,4)	(0,0)

Figure 1: Strategy A is evolutionarily stable against B or against C alone, but against a mutant $\frac{1}{2}$B + $\frac{1}{2}$C, A does worse than the mutant strategy.

Figure 1 gives an example of a game that has an ESSPM but no ESS, which shows that the two are not equivalent and that ESSPM is in fact a strictly weaker solution concept (note that it can be shown relatively easily that ESSPM and ESS are equivalent for 2x2 games). Strategy A is evolutionarily stable against both pure strategies B or C (both attain 2 against A but obtain 0 against themselves while A obtains 1). However, consider the mixed-strategy mutation that plays B and C with probability $\frac{1}{2}$. This strategy obtains expected payoff $\frac{1}{2}2 + \frac{1}{2}2 = 2$ against A (which is the same

payoff that A obtains against itself). Against itself this strategy obtains expected payoff $\frac{1}{4}(0 + 4 + 4 + 0) = 2$, while A obtains payoff of $\frac{1}{2}(1 + 1) = 1$ against it.

3 Algorithm

Given the formulation from Theorem 2, we can cast the problem of computing an ESSPM as the following feasibility program:

$$u_1(x, x^*) \leq u_1(x^*, x^*) - \epsilon_{x1} + M_{x1}y_x \text{ for all } x \in [1, \ldots, m] \tag{3}$$

$$u_1(x, x^*) \leq u_1(x^*, x^*) + M_{x2}(1 - y_x) \text{ for all } x \in [1, \ldots, m] \tag{4}$$

$$u_1(x^*, x^*) \leq u_1(x, x^*) + M_{x3}(1 - y_x) \text{ for all } x \in [1, \ldots, m] \tag{5}$$

$$u_1(x, x) \leq u_1(x^*, x) - \epsilon_{x2} + M_{x4}(1 - y_x) \text{ for all } x \in [1, \ldots, m] \tag{6}$$

$$x_i^* \geq 0 \text{ for all } i \in [1, \ldots, m] \tag{7}$$

$$\sum_i x_i^* = 1 \tag{8}$$

$$y_i \text{ binary for all } i \in [1, \ldots, m] \tag{9}$$

We assume that we are initially given an $m \times m$ matrix A of utilities to player 1 (we assume that the game is symmetric since ESS is defined only for symmetric games, so payoffs to player 2 are given by matrix $B = A^T$). If v and w are pure strategies, the expression $u_1(v, w)$ corresponds to the v'th row and w'th column of matrix A. In general if v and w are vectors of mixed strategies (i.e., probability distributions over pure strategies), then the utility for player 1 is given by $u_1(v, w) = v^T A w$. Without loss of generality we can assume that all values in A are nonnegative and between 0 and 1 (note that any affine transformation does not affect strategic aspects of the game, so we can add a sufficiently large constant to all entries and normalize by dividing by the largest entry to achieve this condition).

The variables in the formulation are x_i^* and y_i for $1 \leq i \leq m$. The x_i^* correspond to the ESSPM for player 1 (and equivalently player 2) that we are seeking to compute, and y_i correspond to indicator variables denoting which of the conditions holds from Theorem 2 for the given component.

We set ϵ to be a very small floating point number slightly larger than 0, such as $\epsilon = 0.00001$. M denotes a constant that exceeds the maximum difference in absolute value of utility between two strategy profiles. We would prefer to have M be as small as possible to make the inequality tighter, so we will set $M = 1 + \epsilon$, given our assumption that all payoffs are between 0 and 1. Note that our framework is flexible enough to allow for different selections for the different ϵ_{xi} and M_{xi} parameters corresponding to the different constraints if desired, though for simplicity we set them all to be the same ϵ and M as described. The equivalence of the feasibility program to the conditions of Theorem 2 follows from a rule for representing either-or constraints in a mixed-integer program by adding in auxiliary binary indicator variables (Bisschop 2006). If $y_x = 0$, then the first constraint ensures that $u_1(x, x^*) \leq u_1(x^*, x^*) - \epsilon_{x1}$, which

is equivalent to $u_1(x, x^*) < u_1(x^*, x^*)$ since ϵ_{x1} is negligible (note that strict inequalities must be converted to weak inequalities to be solved by most standard optimization algorithms). (Numerical precision considerations will be elaborated on further below.) And if $y_x = 1$ then the constraint states $u_1(x, x^*) \leq u_1(x^*, x^*) - \epsilon_{x1} + M_{x1}$, which essentially makes the constraint inactive since it will be true for all strategies x^* since we have chosen M_{x1} to be sufficiently large. The translation between the other constraints and the theorem can be obtained similarly. Note that this formulation is a quadratically-constrained mixed-integer feasibility program (QCMIP). It is quadratically constrained because $u_1(x^*, x^*)$ equals $x^{*T}Ax^*$, which is a quadratic form involving a product of the variables x^*. It is mixed integral because there are both continuous and binary variables. And it is a feasibility program because there is no objective function, though there are several candidates that could potentially be used that may help aid performance in practice.

To implement our algorithm we explored the QVXPY and Gurobi software packages (Park and Boyd 2017; Gurobi Optimization, Inc. 2014) and will present analysis in Section 4. Beyond finding a suitable optimization solver, there are several further challenges for applying our algorithm. A game may contain no ESSPM, in which case our algorithm should return that the optimization model is infeasible. Due to degrees of approximation error and numerical precision there is some chance of a false negative (our algorithm outputs that the model is infeasible while an ESSPM actually exists). Note that if we were solving for a different solution concept that is known to exist, such as Nash equilibrium, we would know for sure that an output of infeasibility would indicate a false negative; however for ESSPM we cannot be sure whether the game actually has no ESSPM or we are in a false negative (and so our algorithm is *incomplete*). We will explore the extent of this issue in the experiments. A second issue is that according to the alternative definition of ESSPM given by Theorem 2, the conditions apply just to $x \neq x^*$, which we do not encode directly in our algorithm, which considers deviations for all x since it cannot readily tell whether $x = x^*$. This can be problematic for the case when x^* is a pure strategy, and the conditions for $x = x^*$ would be enforced when they should not be (which would lead to a false negative). To address this we will first apply an efficient preprocessing procedure to determine if a pure strategy ESSPM exists, which is described in Algorithm 1.

Algorithm 1 Preprocessing procedure for computing pure-strategy ESSPM

Inputs: Payoff matrix M with m pure strategies per player.
    for $i = 1$ to m **do**
        for $j = 1$ to m, $j \neq i$ **do**
            Test whether the conditions of Theorem 2 hold using $x^* = i$, $x = j$.

        If the conditions held for all j, output i as pure-strategy ESSPM.

Conditions of Theorem 2 did not hold for any pure strategy, so there is no pure ESSPM

Since our algorithm has elements of approximation error and numerical imprecision, we must develop a metric to evaluate the quality of the candidate solution. For approximating Nash equilibrium the standard metric is the maximum amount a player can gain by deviating from the candidate equilibrium strategy profile x^*, i.e., $\epsilon = \max_i \max_j (u_i(x_j, x^*_{-i}) - u_i(x^*, x^*_{-i}))$ (where $u_i(x, y)$ is the utility to player i when player i follows x and the opponent(s) follow y, and x^*_{-i} is the strategy vector for x^* excluding player i). It is common for Nash equilibrium algorithms to be analyzed by their convergence with respect to this ϵ (Gilpin, Peña, and Sandholm 2012). As one notable example an ϵ-Nash equilibrium has been computed for two-player limit Texas hold 'em for a sufficiently small ϵ that a human cannot differentiate between the approximation and an exact equilibrium over a lifetime of human play (Bowling et al. 2015). For ESSPM we propose a similar metric, given by Algorithm 2. One challenge is that it may not be clear which of the conditions to consider, since small amounts of numerical instability can make it impossible to satisfy a hard equality. To address this we include a precision parameter δ, which we set equal to 10^{-7}. Note that this could be set equal to or different from the ϵ_{xi} of our algorithm (in general it seems most appropriate to set δ to be smaller than the ϵ_{xi}).

Algorithm 2 Procedure to compute degree of approximation error of candidate ESSPM strategy

---

Inputs: Payoff matrix with m pure strategies per player, candidate ESSPM x^*, degree of precision δ

    for $i = 1$ to m **do**
        if $u_1(i, x^*) - u_1(x^*, x^*) > \delta$ **then**
            $\theta[i] \leftarrow u_1(i, x^*) - u_1(x^*, x^*)$
        else if $u_1(i, x^*) - u_1(x^*, x^*) > (-1) \cdot \delta$ **then**
            if $u_1(i, i) - u_1(x^*, i) > 0$ **then**
                $\theta[i] \leftarrow u_1(i, i) - u_1(x^*, i)$
        else
            $\theta[i] \leftarrow 0$
    return $\max_i \theta[i]$

---

4 Applying the algorithm to solve the Mutation-Population Game

We first implemented the algorithm using the CVXPY non-convex quadratically-constrained quadratic program (QCQP) Suggest-and-Improve framework (Park and Boyd 2017), since it is the only publicly-available software we are aware of for solving our QCMIP formulation (which is non-convex). We considered all of their algorithms and found best performance with the coordinate-descent algorithm, because it focuses more on trying to find and maintain feasibility as opposed to the other algorithms which are more focused on optimality. This is appropriate for our problem since it is a feasibility program with no objective function. We create an initial solution for the algorithm by solving the semidefinite program (SDP) relaxation, which we then attempt to improve iteratively using coordinate descent. For the parameters we set $\epsilon_{xi} = 0.00001$ and $M_{xi} = 1 + \epsilon_{xi}$.

We tested this implementation on the well-studied Mutation-Population game (Figure 2) (Maschler, Solan, and Zamir 2013). In this game there are two types of animals: hawks (who are aggressive), and doves (who are peaceful). When an animal invades the territory of another animal of the same species, a hawk will aggressively repel the invader, while a dove will yield and be driven out of the territory. The game has one symmetric Nash equilibrium in which Dove is selected with probability $\frac{1}{5}$ and Hawk $\frac{4}{5}$. This strategy profile also constitutes an ESS and an ESSPM. (Note that the game also has two asymmetric Nash equilibria (Dove, Hawk) and (Hawk, Dove), both of which are neither ESS nor ESSPM.)

	Dove	Hawk
Dove	(4,4)	(2,8)
Hawk	(8,2)	(1,1)

Figure 2: The Mutation-Population Game

When we ran our algorithm on this game, the initial solution obtained from the SDP relaxation was $(0.637, 0.363)$, and the solution after the first iteration of the algorithm was $(0.245, 0.755)$. However, the solution varied drastically between iterations as the algorithm progressed and did not converge over 1000 iterations, though the degree of violation of the constraints generally decreased. We therefore concluded that the CVXPY-QCQP framework was not suitable for solving our problem, despite the fact that good performance has been observed for previous benchmark domains (Park and Boyd 2017)).

We next decided to run experiments with Gurobi's mixed-integer programming solver (Gurobi Optimization, Inc. 2014). Gurobi does not have a solver for non-convex quadratically-constrained programs; however, we can apply a known technique that approximates products of variables using piecewise linearization by adding additional constraints known as "SOS2" constraints (Bisschop 2006). This approach introduces an additional layer of approximation due to the piecewise-linear approximation of quadratic variables, and also introduces an additional parameter to use for the number of breakpoints in the linearization. However, it is able to quickly and effectively solve the mutation-population game. For example, using $k = 20$ breakpoints with $\epsilon = 10^{-5}$ it outputs a solution of $(0.19972, 0.80028)$ in 0.242 seconds, which is extremely close to the optimal solution and produces an approximation error of 1.4×10^{-4}.

5 Experiments

Given the results of the preceding section we decided to use Gurobi's solver for the remainder of our experiments. First, in Section 5.1 we further explore the effect of varying the choices of parameters k and ϵ on the runtime and performance. In Section 5.2 we look at performance on uniform-random symmetric 2x2 games. In addition to runtime and performance we also explore the false negative rate of our algorithm. It is known that all 2x2 games that satisfy a certain condition contain an ESS (and therefore ESSPM). The

condition is that $a_{11} \neq a_{21}$ and $a_{12} \neq a_{22}$, where A is the payoff matrix (van Damme 1987). This implies that an ESSPM will exist with probability 1 in uniformly-generated 2x2 games, since there is probability zero on the exact payoff values needed being equal. Therefore, any infeasibility output of our algorithm will be a false negative. This does not hold for games with more than two strategies, as the rock-paper-scissors example in the introduction demonstrates. At the other extreme, it has been shown that as the number of pure strategies approaches infinity the probability of existence of an ESS with support of size two converges to 1 for games with payoffs chosen according to a distribution with "exponential and faster decreasing tails" which includes uniform (the *support* of a randomized strategy is the set of pure strategies that are played with nonzero probability) (Hart, Rinott, and Weiss 2008). In Section 5.2 we also explore 3x3 games and discuss further scalability, and in Section 5.3 we explore a generalized game class generated by the GAMUT repository (Nudelman et al. 2004).

5.1 Mutation-Population Game

In the Mutation-Population Game we consider the effect of selection of parameters k (number of breakpoints) and ϵ on the runtime and error (computed by Algorithm 2). Table 1 shows the effect of varying k keeping $\epsilon = 10^{-5}$ fixed. For the values of k up to 30 the runtime is under one second, before increasing to 1.7 seconds for $k = 50$ and 6.6 seconds for $k = 100$. Interestingly for $k = 5$ our algorithm outputs infeasibility, while it produces low errors for larger k which decrease monotonically. This demonstrates that choosing too few breakpoints can be problematic and lead to a false negative. Using 10–30 breakpoints leads to low runtimes and relatively small errors. These experiments were using up to 64 cores (threads) for the algorithm.

We next explored the effect of varying ϵ while keeping $k = 20$ fixed (Table 2). Changing ϵ seems to have minimal effect on the runtime for this game. For the three largest ϵ values our algorithm outputs that the model is infeasible. This demonstrates that careful selection of ϵ in addition to k is needed to prevent a false negative. The results also show that the error does not necessarily decrease monotonically with smaller ϵ (though this conclusion is drawn from a small sample size).

5.2 Uniform random games

We next considered 2x2 symmetric games with all payoffs generated uniformly at random (we generated 4 payoffs for the matrix A for player 1, and then assume that the payoff matrix for player 2 is A^T). We generated and solved 10,000 games, using 20 breakpoints with $\epsilon = 10^{-5}$. Of these, the pre-processing step confirmed that 7506 had a pure-strategy ESSPM. Of the remaining games, our algorithm computed an optimal solution for 2454, and output that 40 were infeasible. The average running time for the games with an optimal (non-pure) solution was 0.18s, and the average running times for the infeasible cases was 0.26s. The average approximation error for the optimal solutions was 1.4×10^{-4}.

These results lead to several immediate observations. First, a very large percentage games contain a pure-strategy

ESSPM (close to 75% in our sample). Note that this is not surprising, since we are considering symmetric games. If the payoff matrix is A, then the game has a pure-strategy ESSPM whenever $a_{11} > a_{21}$ or $a_{22} > a_{12}$, which is exactly 75%. Note that for randomly-generated games the possibility of identical payoffs for different strategy profiles has probability zero. This may make it difficult for the second condition of Theorem 2 to hold (the probability of it holding when x^* is a pure strategy is 0, though it could be possible for mixed strategies). However, rather than trying to optimize our algorithm just for random games we elect to keep our general-purpose framework as the second condition can be important for many games we are interested in.

The observation that our algorithm outputs infeasibility for 40 games (which constitutes 1.6% of the games in our sample that did not contain a pure-strategy ESSPM) shows that false negatives can occur even for reasonably chosen values of the parameters $k = 20$ and $\epsilon = 10^{-5}$. On the positive side, the runtimes and average approximation errors are quite low. So for the games that our algorithm does output a valid solution, the solution is obtained quickly with low error. And furthermore a valid solution is found for a very large percentage of the game instances (98.4% of the games that do not contain a pure-strategy ESSPM). If our algorithm outputs infeasibility for a specific game we are interested in, we can always solve it again with larger values of k and/or smaller values of ϵ until a solution is found (note that this would only work for 2x2 games, and for larger games we could not be sure whether a solution exists and may have to give up eventually).

We also experimented on 3x3 uniform-random symmetric games using the same parameter values $k = 20$, $\epsilon = 10^{-5}$. Of the 1000 games in our sample, our algorithm output that 709 had a pure-strategy ESSPM, it found an optimal solution for 39, and output that 252 were infeasible. The average running time for the games with an optimal (non-pure) solution was 12.7s, and the average running times for the infeasible cases was 14.7s. The average approximation error for the optimal solutions was 3.3×10^{-4}.

These results indicate as before that a large fraction of random 3x3 games have a pure-strategy ESSPM (the theoretical value should be $\frac{19}{27} \equiv 0.704$, and in general for m pure strategies should be $1 - \left(\frac{m-1}{m}\right)^m$). In contrast to the $m = 2$ results, for $m = 3$ we observe that our algorithm outputs infeasibility for a very large fraction of the games with no pure-strategy ESSPM (86.7% in contrast to 1.6% for $m = 2$ using the same parameter values). Unlike the $m = 2$ case we can not be sure for a given game whether our algorithm has produced a false negative or whether no ESSPM exists, so we cannot conclude whether this disparity is due to an increased false negative rate or nonexistence of ESSPM. For the games that our algorithm did solve optimally the running time was only 12.7s on average (and the running time to determine infeasibility was only 14.7s on average), and the approximation error was quite low for the games which our algorithm output an optimal solution.

We also performed experiments for 3x3 games using $k = 10$, with $\epsilon = 10^{-5}$ as before. Over 1000 games our algo-

# Breakpoints	5	10	20	30	50	100
Time(s)	0.04	0.08	0.13	0.54	1.69	6.6
Error	INFEASIBLE	0.001	1.4×10^{-4}	5.5×10^{-5}	1.3×10^{-5}	5.9×10^{-6}

Table 1: Running time and approximation error for Mutation-Population Game for different breakpoints with $\epsilon = 10^{-5}$.

ϵ	1×10^{-1}	1×10^{-2}	1×10^{-3}	1×10^{-4}	1×10^{-5}	1×10^{-6}
Time(s)	0.13	0.31	0.28	0.35	0.13	0.25
Error	INFEASIBLE	INFEASIBLE	INFEASIBLE	6.6×10^{-5}	1.4×10^{-4}	1.9×10^{-4}

Table 2: Running time and approximation error for Mutation-Population Game for different values of ϵ using 20 breakpoints.

#Strategies	#Games	k	ϵ	#Pure	#Optimal	#Infeasible	Optimal avg. runtime	Infeasible avg. runtime	Avg. error
2	10,000	20	10^{-5}	7506	2454	40	0.18s	0.26s	1.4×10^{-4}
3	1000	20	10^{-5}	709	39	252	12.7s	14.7s	3.3×10^{-4}
3	1000	10	10^{-5}	712	68	220	1.2s	1.7s	0.002
4	1000	10	10^{-5}	671	48	281	6.3s	9.0s	0.004
5	1000	10	10^{-5}	656	50	293	29.7s	70.2s	0.005

Table 3: Results for uniform random games.

rithm output that 712 had a pure-strategy ESSPM, found an optimal solution for 68, and output that 220 were infeasible. The average running time for the games with an optimal (non-pure) solution was 1.2s, and the average running times for the infeasible cases was 1.7s. The average approximation error for the optimal solutions was 0.002. Using fewer breakpoints decreased the infeasibility rate over this sample from 86.7% to 76.4%, while leading to a significant reduction in running time, but an order of magnitude increase in approximation error.

We also experimented on 4x4 games using $k = 10$, $\epsilon = 10^{-5}$. Over 1000 games the algorithm output that 671 had a pure-strategy ESSPM, found an optimal solution for 48, and output that 281 were infeasible. The average running time for the games with an optimal (non-pure) solution was 6.3s, and the average running times for the infeasible cases was 9.0s. The average approximation error for the optimal solutions was 0.004. For 5x5 games with the same parameters we found 656 pure-strategy ESSPM, 50 optimal solutions, and 293 infeasible. The average runtime for the optimal cases was 29.7s, and 70.2s for the infeasible cases. The average error for the optimal solutions was 0.005.

5.3 Experiments on games from GAMUT repository

In addition to uniform randomly-generated games we also experimented on the class of Chicken games created using the GAMUT generator (Nudelman et al. 2004). These games generalize the Mutation-Population game that was previously considered. For a payoff matrix A, Chicken games satisfy $a_{21} > a_{11} > a_{12} > a_{22}$. We generated 1,000 games from this class where, as before, we normalize payoffs to be between 0 and 1 before applying our algorithm. We only consider Chicken games with $m = 2$ pure strategies per player, as GAMUT does not support a generalized version. Results using $k = 20, \epsilon = 10^{-5}$ are given in Table 4. Note that none of the games have a pure-strategy ESSPM, while

our algorithm incorrectly outputs infeasibility for 1.8% of the games (we know that an ESSPM exists in nondegenerate 2x2 games). The average runtimes and errors of our algorithm are low for this class.

Several popular games such as Prisoner's Dilemma and Battle of the Sexes contain a pure-strategy ESS (and therefore ESSPM), so experimenting on them is not useful. GAMUT also supports a class called "Hawk and Dove," however the way this is generated produces only games that have a pure-strategy ESS. (Note that several different definitions of "Hawk and Dove games" have been proposed in the literature. For example, while the Population-Mutation game we considered does not fall into GAMUT's class, it is considered a Hawk-Dove game by Maschler et al. (Maschler, Solan, and Zamir 2013).)

6 Conclusion

We have presented an optimization formulation and algorithm that constitutes the first positive result for computation of Evolutionarily Stable Strategies (ESS) (previously the known computational results were all negative hardness results). ESS is a well-studied refinement of Nash equilibrium that is biologically-motivated and has been applied to cancer. We consider a restriction where robustness is guaranteed only against pure-strategy mutations and not necessarily against all mixed-strategy mutations, which we call Evolutionarily Stable Strategies against Pure Mutations (ESSPM). The algorithm is based on a new quadratically-constrained mixed integer feasibility program formulation.

We experimented with different parameter settings for the Population-Mutation Game, which were then used for experiments in uniform-random games and in a generalized class of Chicken games created by the GAMUT generator. We observe that for certain classes a large fraction of games has a pure-strategy ESSPM, and that our algorithm can sometimes output a false negative infeasibility for 2x2

#Pure	#Optimal	#Infeasible	Optimal avg. runtime	Infeasible avg. runtime	Avg. error
0	982	18	0.13s	0.19s	1.3×10^{-4}

Table 4: Results for 1,000 games of Chicken generated by GAMUT, using parameter values $k = 20$, $\epsilon = 10^{-5}$.

games where an ESS is known to exist. However, in several experiments the false negative rate is small, and the running times and errors produced by our algorithm are generally quite low for the games that are solved. We considered two commercial solvers CVXPY-QCQP and Gurobi, focusing our experiments on Gurobi after preliminary analysis indicated that it was better suited for our setting. The algorithm takes two input parameters, the number k of breakpoints for the piecewise-linear approximations of variable products, and the degree ϵ of constraint approximation to use. Our experiments investigated the effect of selection of these parameters on performance.

We would like to further tune the parameters for epsilon and the number of breakpoints and further explore the cases of infeasibility that we encountered. It is not clear for the games with more than 2 actions whether the infeasibility was due to the approximation error of the algorithm or to the lack of existence of an ESSPM. Eventually we would like to generalize our approach to compute an ESS that is robust to all mutation strategies (as in the standard definition of ESS), not just to pure strategy mutations as we have done. It is possible that this can be accomplished within the same mixed-integer non-convex quadratically-constrained optimization framework. Our approach could potentially apply to generalizations to multiple players by approximating the products of variables in a similar way, though the standard definition of ESS is just for two-player symmetric games. We would also like to see how our algorithm can perform on a problem generated from real data, perhaps one that has application to medicine such as the cancer eradication game described in the introduction, and to experiment with our algorithm on games with larger number of actions per player.

References

Berg, K., and Sandholm, T. 2017. Exclusion method for finding Nash equilibrium in multiplayer games. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Bisschop, J. 2006. *AIMMS—Optimization Modeling*. Paragon Decision Technology.

Bowling, M.; Burch, N.; Johanson, M.; and Tammelin, O. 2015. Heads-up limit hold'em poker is solved. *Science* 347(6218):145–149.

Chen, X., and Deng, X. 2005. 3-Nash is PPAD-complete. *Electronic Colloquium on Computational Complexity* Report No. 134.

Chen, X., and Deng, X. 2006. Settling the complexity of 2-player Nash equilibrium. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*.

Conitzer, V. 2013. The exact computational complexity of evolutionarily stable strategies. In *Conference on Web and Internet Economics (WINE-13)*, 96–108.

Dingli, D.; Chalub, F.; Santos, F.; Segbroeck, S. V.; and Pacheco, J. 2009. Cancer phenotype as the outcome of an evolutionary game between normal and malignant cells. *British Journal of Cancer* 101(7):1130–1136.

Etessami, K., and Lochbihler, A. 2008. The computational complexity of evolutionarily stable strategies. *International Journal of Game Theory* 37(1):93–113.

Gilpin, A.; Peña, J.; and Sandholm, T. 2012. First-order algorithm with $\mathcal{O}(\ln(1/\epsilon))$ convergence for ϵ-equilibrium in two-person zero-sum games. *Mathematical Programming* 133(1–2):279–298. Conference version appeared in AAAI-08.

Govindan, S., and Wilson, R. 2003. A global Newton method to compute Nash equilibria. *Journal of Economic Theory* 110:65–86.

Gurobi Optimization, Inc. 2014. Gurobi optimizer reference manual version 6.0.

Hart, S.; Rinott, Y.; and Weiss, B. 2008. Evolutionarily stable strategies of random games, and the vertices of random polygons. *Annals of Applied Probability* 18(1):259–287.

Lemke, C., and Howson, J. 1964. Equilibrium points of bimatrix games. *Journal of the Society of Industrial and Applied Mathematics* 12:413–423.

Maschler, M.; Solan, E.; and Zamir, S. 2013. *Game Theory*. Cambridge University Press.

Maynard Smith, J., and Price, G. 1973. The logic of animal conflict. *Nature* 2461:13–18.

Maynard Smith, J. 1982. *Evolution and the Theory of Games*. Cambridge University Press.

Nash, J. 1950. *Non-cooperative games*. Ph.D. Dissertation, Princeton University.

Nudelman, E.; Wortman, J.; Leyton-Brown, K.; and Shoham, Y. 2004. Run the GAMUT: A comprehensive approach to evaluating game-theoretic algorithms. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

Park, J., and Boyd, S. 2017. General heuristics for nonconvex quadratically constrained quadratic programming. *arXiv preprint arXiv:1703.07870 [math.OC]*.

Porter, R.; Nudelman, E.; and Shoham, Y. 2008. Simple search methods for finding a Nash equilibrium. *Games and Economic Behavior* 63(2):642–662.

Sandholm, T.; Gilpin, A.; and Conitzer, V. 2005. Mixed-integer programming methods for finding Nash equilibria. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 495–501.

van Damme, E. 1987. *Stability and Perfection of Nash Equilibrium*. Springer-Verlag.