Hao, Yiping; Wu, Zhijun

Article

Computation of sparse and dense equilibrium strategies of evolutionary games

Games

Provided in Cooperation with:
MDPI – Multidisciplinary Digital Publishing Institute, Basel

Suggested Citation: Hao, Yiping; Wu, Zhijun (2018) : Computation of sparse and dense equilibrium strategies of evolutionary games, Games, ISSN 2073-4336, MDPI, Basel, Vol. 9, Iss. 3, pp. 1-15, https://doi.org/10.3390/g9030046

games

MDPI

Article

Computation of Sparse and Dense Equilibrium Strategies of Evolutionary Games

Yiping Hao † and Zhijun Wu *,†

Department of Mathematics, Iowa State University, Ames, IA 50011, USA; hao.yiping@gmail.com
* Correspondence: zhijun@iastate.edu; Tel.: +1-515-294-8165
† These authors contributed equally to this work.

Received: 8 May 2018; Accepted: 4 July 2018; Published: 7 July 2018

Abstract: The evolution of social or biological species can be modeled as an evolutionary game with the equilibrium strategies of the game as prediction for the ultimate distributions of species in population, when some species may survive with positive proportions, while others become extinct. We say a strategy is dense if it contains a large and diverse number of positive species, and is sparse if it has only a few dominant ones. Sparse equilibrium strategies can be found relatively easily, while dense ones are more computationally costly. Here we show that by formulating a "complementary" problem for the computation of equilibrium strategies, we are able to reduce the cost for computing dense equilibrium strategies much more efficiently. We describe the primary and complementary algorithms for computing dense as well as sparse equilibrium strategies, and present test results on randomly generated games as well as a more biologically related one. In particular, we demonstrate that the complementary algorithm is about an order of magnitude faster than the primary algorithm to obtain the dense equilibrium strategies for all our test cases.

Keywords: evolutionary games; Nash equilibrium; Shapley-Snow algorithm; dense vs. sparse strategies; biodiversity

1. Introduction

Biological species (viruses, bacteria, insects, plants, or animals) develop, compete, reproduce, and evolve. Different species have different evolutionary strategies, some fail to survive while others are selected [1]. The evolution of biological species can be modeled mathematically as an evolutionary game, with the equilibrium strategies of the game as prediction for the ultimate distributions of species in population, when some species may survive with positive proportion, while others become extinct [2,3]. We say a strategy is dense if it contains a large and diverse number of positive species, and is sparse if it has only a few dominant ones.

Assume that populations may have mixed strategies, or in other words, assume that they are a polymorphic mixture of pure strategists. Let $x = (x_1, \ldots, x_n)^T$ be the strategy vector of a subpopulation, with x_i being the frequency or chance of the subpopulation to be species i, $\sum_i x_i = 1$. Let $y = (y_1, \ldots, y_n)^T$ be the composition profile of the population, with y_j being the fraction of species j, $\sum_j y_j = 1$. The average payoff for a subpopulation of strategy x in population of composition y would be $\sum_{i,j} x_i A_{i,j} y_j = x^T A y$, where $A_{i,j}$ is the payoff of the subpopulation of species i in population of species j, and $A = \{A_{i,j}\}$ is called the payoff matrix [4–6].

For an evolutionary game, a composition x^* is defined to be an equilibrium of the game, when the payoff of any subpopulation with the same strategy x^* is maximized:

$$x^{*T} A x^* \geq x^T A x^*, \qquad x \in S \tag{1}$$

where $S = \{x \in R^n : \sum_i x_i = 1,\ x_i \geq 0,\ i = 1, \ldots, n\}$ is the set of all possible strategies [4–6].

John Nash (1950, 1951) [7,8] showed that for an n-person non-cooperative game, there always exists an equilibrium strategy, now called Nash equilibrium bearing his name. However, computing an equilibrium strategy is hard in general. In fact, many types of games are proved to be NP-hard in terms of their computational complexities [9–12]. An evolutionary game is also NP-hard. For example, the evolutionary game on a graph, with the adjacency matrix of the graph as the payoff matrix, is equivalent to a maximum clique problem, which is NP-complete [13–15].

Many algorithms have been developed for computing equilibrium strategies of games, most notable, the Shapley-Snow algorithm [16], the Lemke-Howson algorithm [17], and the quadratic programming algorithm by Mangasarian and Stone [18]. The Shapley-Snow algorithm searches strategies with all possible numbers of nonzero components and reports all equilibrium ones. The Lemke-Howson algorithm follows a procedure similar to the simplex algorithm for linear programming along a special path of strategies that hopefully leads to an equilibrium strategy. The quadratic programming algorithm tries to find a strategy that satisfies a set of complementarity conditions necessary and sufficient for equilibrium strategies by solving a specially constructed quadratic program. The Lemke-Howson algorithm and the quadratic programming algorithm are efficient if they succeed, but they can only find one equilibrium strategy, which cannot be prescribed. The Shapley-Snow algorithm is more expensive, but can in principle find all possible equilibrium strategies.

We are interested in computing the equilibrium strategies of a given evolutionary game and in particular, the strategies with a sparse or dense set of nonzero components. The Shapley-Snow algorithm is thus a natural choice of the algorithm, for it can be modified straightforwardly to find the sparsest or densest equilibrium strategy, as we will describe in greater detail in the rest of the paper, while the Lemke-Howson algorithm and the quadratic programming algorithm cannot be modified easily to serve our purpose.

The problem of computing sparse or dense equilibrium strategies is challenging yet interesting both computationally and practically. Similar problems have been investigated in other closely related fields such as in regularization of least squares regression [19,20] and in signal reconstruction in compressive sensing [21–24]. In both cases, a sparse solution is sought for the problem by minimizing the l_1-norm of the solution. These studies have many applications in science and engineering, and have made great impacts in broad areas of scientific computing.

Computing sparse or dense equilibrium strategies is of great computational challenge, because the computation of such a strategy can be computationally very expensive when the problem size is large: Since the equilibrium strategies do not form a continuous set of strategies, there is no regularization schemes such as the l_1-norm minimization for least-squares regression and compressive sensing that can be used to obtain a sparse or dense equilibrium strategy. A general algorithm such as the Shapley-Snow algorithm has to be adopted with certain modifications. Yet, in the worst case, the computation time may still grow exponentially with increasing problem size.

Computing sparse or dense equilibrium strategies is of great interest in biological and social applications. For example, in ecological modeling, to preserve biodiversity, it is important to predict a population state with only a few species left or with most species kept in population [1,25]. In economic analysis, to optimize investment portfolio, it is crucial to know more risky investment plans when the funds are concentrated to only a few stocks or more balanced plans when the funds are more diversified. The portfolio optimization problem can be formulated as an evolutionary game with the sparsest equilibrium strategy corresponding to the most risky investment plan, and the densest one to the most balanced plan [26,27].

In this paper, we show that the equilibrium strategies of an evolutionary game, including sparse and dense ones, can be computed by implementing a standard Shapley-Snow algorithm. We then show that by using a modified Shapley-Snow algorithm, sparse equilibrium strategies for an evolutionary game can be computed relatively easily, while dense ones are much more computationally

costly. However, we show that by formulating a "complementary" problem for the computation of equilibrium strategies, we can reduce the cost for computing dense equilibrium strategies and obtain them much more efficiently. We describe the primary and complementary algorithms and present test results on randomly generated games as well as a game related to allele selection in genetic studies. In particular, we demonstrate that the complementary algorithm is about an order of magnitude faster than the primary algorithm to obtain the dense equilibrium strategies for all our test cases.

Note that in biology, the diversity of a given population can be measured by the richness of species types and the evenness of species distribution. The richness of species in population x can be represented by the number of nonzero elements of x. The evenness of species can be measured with different standards [25]. In this paper, we will only check the standard deviation of nonzero elements of x for the evenness of species, for the focus of this study is on computation not biological analysis.

$$\text{Richness of species}: \quad \|x\|_0 = |\{x_i > 0 : \ i = 1, \ldots, n\}| \tag{2}$$

$$\text{Evenness of species}: \quad \sigma^2 = \sum\{(x_i - \mu)^2 : \ x_i > 0\}/\|x\|_0 \tag{3}$$

where $\mu = \sum\{x_i : \ x_i > 0\}/\|x\|_0$.

2. Shapley-Snow Algorithms

It is well known that a necessary and sufficient set of conditions for a strategy $x^* \in S$ to be a Nash equilibrium of (1) is that there is a parameter λ^* such that:

$$\begin{gathered} \lambda^* e - Ax^* \geq 0, \quad x^* \geq 0, \\ x^{*T}(\lambda^* e - Ax^*) = 0, \end{gathered} \tag{4}$$

which is equilvalent to

$$\begin{gathered} \lambda^* = A_i.x^*, \ if \ x_i^* > 0, \\ \lambda^* \geq A_i.x^*, \ if \ x_i^* = 0, \end{gathered} \tag{5}$$

where e is a vector of size n of all 1's, and $A_i.$ is the ith row of matrix A [5].

Let $p = \{i : \ x_i^* > 0\}$. Then, based on the conditions in (5), $\lambda^* = A_i.x^*$ for all $i \in p$. The latter can be written in a more compact form as $\lambda^* e_p = A_{pp}x_p^*$, where e_p is a vector of size $|p|$ of all 1's, $x_p^* = \{x_i^* : \ i \in p\}$, and $A_{pp} = \{A_{i,j} : \ i, j \in p\}$. It follows that if A_{pp} is nonsingular, then

$$\lambda^* = 1/e_p^T A_{pp}^{-1} e_p, \qquad x_p^* = A_{pp}^{-1} e_p / e_p^T A_{pp}^{-1} e_p. \tag{6}$$

In addition, if $q = \{i : \ x_i^* = 0\}$, then, $\lambda^* \geq A_i.x^*$ for all $i \in q$, which can be simplified to

$$\lambda^* e_q \geq A_{qp} x_p^*, \tag{7}$$

where $A_{qp} = \{A_{i,j} : \ i \in q, \ j \in p\}$ [5].

However, in order to find an equilibrium strategy x^*, we do not have the prior knowledge on p, the set of indices of positive components of x^*. Therefore, a simple way to do it is to enumerate all possible sets of indices p, to see if any gives rise to an equilibrium strategy. This is exactly the idea of the Shapley-Snow algorithm proposed by L. Shapley and R. Snow in early 1950s [16]. Let P_k be the set of all subsets of $\{1, 2, \ldots, n\}$ of size k. Then, a Shapley-Snow algorithm can be described formally as Algorithm 1.

Algorithm 1: A Complete Shapley-Snow Algorithm

For $k = 1, \ldots, n$ **do**
  For each $p \in P_k$ **do**
    If A_{pp} is nonsingular and $e_p^T A_{pp}^{-1} e_p \neq 0$ **then**
      $\lambda^* = 1/e_p^T A_{pp}^{-1} e_p$, $x_p^* = A_{pp}^{-1} e_p / e_p^T A_{pp}^{-1} e_p$
    End
    If $x_p^* > 0$ and $\lambda^* e_q \geq A_{qp} x_p^*$ **then**
      x^* is a Nash equilibrium, record x^*.
    End
  End
End

Note that the inverse of A_{pp} in the algorithm can actually be replaced by the adjoint matrix of A_{pp} in case A_{pp} is singular [16]. However, to simplify the representation and analysis, we just assume that A_{pp} is nonsingular and use A_{pp}^{-1} in our description and analysis throughout the paper.

3. Dense vs. Sparse Equilibria

The Shapley-Snow algorithm described in the previous section computes the equilibrium strategies of all possible sizes of p for a given evolutionary game. We therefore call it a Complete Shapley-Snow Algorithm. In this algorithm, the equilibrium strategies are computed in the order of increasing sizes of p, i.e., first, strategies with the smallest number of positive components, and then larger ones, and the largest in the end. Therefore, the algorithm can be modified easily to find only a sparse or dense equilibrium, say a strategy with the smallest or largest number of positive components. The following Algorithms 2 and 3 are the algorithms that can be implemented for such purposes.

Algorithm 2: A Sparse Shapley-Snow Algorithm

For $k = 1, \ldots, n$ **do**
  For each $p \in P_k$ **do**
    If A_{pp}^{-1} is nonsingular and $e_p^T A_{pp}^{-1} e_p \neq 0$ **then**
      $\lambda^* = 1/e_p^T A_{pp}^{-1} e_p$, $x_p^* = A_{pp}^{-1} e_p / e_p^T A_{pp}^{-1} e_p$
    End
    If $x_p^* > 0$ and $\lambda^* e_q \geq A_{qp} x_p^*$ **then**
      x^* is a Nash equilibrium, record x^*, **exit**.
    End
  End
End

Algorithm 3: A Dense Shapley-Snow Algorithm

For $k = n, \ldots, 1$ **do**
  For each $p \in P_k$ **do**
    If A_{pp}^{-1} is nonsingular and $e_p^T A_{pp}^{-1} e_p \neq 0$ **then**
      $\lambda^* = 1/e_p^T A_{pp}^{-1} e_p$, $x_p^* = A_{pp}^{-1} e_p / e_p^T A_{pp}^{-1} e_p$
    End
    If $x_p^* > 0$ and $\lambda^* e_q \geq A_{qp} x_p^*$ **then**
      x^* is a Nash equilibrium, record x^*, **exit**.
    End
  End
End

In Algorithm 2, the Sparse Shapley-Snow Algorithm, the strategies with small numbers of positive components are examined first. The algorithm exits once an equilibrium strategy with the smallest number of positive components is found. In Algorithm 3, the Dense Shapley-Snow Algorithm, the strategies with large numbers of positive components are examined first. It exits after an equilibrium strategy with the largest number of positive components is found. The former only solves some small linear systems of equations and does not require much computation, while the latter needs to solve relatively large linear systems and is more computationally expensive.

We have implemented the complete, sparse, and dense versions of the Shapley-Snow algorithm in Matlab on a 1.3 GHz MacBook Air, and tested them on a randomly generated set of evolutionary games. A total of 10 games are generated, each having 20 strategies and defined by a 20 by 20 randomly generated payoff matrix, i.e., all the entries of the matrix are set to a random number in between 0 and 1. The reason to test a set of randomly generated games is to avoid possible biases from specially structured games, and instead to have a relatively broad range of test cases in terms of numbers, sparsities, and densities of their equilibrium strategies.

From Table 1, we see that by using Algorithm 1, the Complete Shapley-Snow Algorithm, all equilibrium strategies are found for each of the games. The total number of equilibrium strategies for each game ranges from 9 to 65. However, the time spent to find all the strategies for each game is relatively long, and is about the same for all the games, in between 700 and 800 s. This is because the algorithm needs to exhaust all possible trial strategies, from those with 1 positive component, to those with 2, 3, ..., n. No matter how many equilibrium strategies are there for each game, the amount of work required to find them all is about the same.

Table 1. Performance of Complete Shapley-Snow Algorithm.

Game	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
# Equilibria	9	27	11	17	59	55	35	11	29	65
Time	723	838	723	737	737	818	726	714	770	769

Table legend: Game—Tested 10 games; # Equilibria—The number of equilibrium strategies found for each game; Time—Time in seconds spent on each game.

Table 2 shows the results for computing the sparsest equilibrium strategies for the tested games. Since the Sparse Shapley-Snow Algorithm exits immediately once it finds the equilibrium strategy of the smallest number of positive components, the algorithm does not require much computation time. Therefore, the sparser the first equilibrium strategy, the shorter the required computation time. Indeed, all the cases where there is only 1 positive component in the sparsest equilibrium take only 0.02 s to finish; The time is increased for the case where there are 2 positive components; and most cases where there are 3 positive components require about 0.44 s except for 1 case which takes 1.83 s. In general, the amount of computation required to find the sparsest equilibrium strategy is small because the algorithm only solves a set of small linear systems of equations before it finds the first and also the sparsest equilibrium strategy.

Table 2. Performance of Sparse Shapley-Snow Algorithm.

Game	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$\|\cdot\|_0$	3	1	3	3	2	1	1	1	3	1
μ	0.33	1	0.33	0.33	0.5	1	1	1	0.33	1
σ	0.42	0	0.14	0.41	0.41	0	0	0	0.20	0
Time	1.83	0.02	0.43	0.41	0.06	0.02	0.02	0.02	0.44	0.02

Table legend: Game—Tested 10 games; $\|\cdot\|_0$—The number of positive components in the sparsest equilibrium; μ—The average value of positive components; σ—The standard deviation of the positive components; Time—Time in seconds spent to obtain the sparest equilibria.

Table 3 is the opposite of Table 2. It shows the results for computing the densest equilibrium strategies of the tested games. Since the Dense Shapley-Snow Algorithm starts with the strategies with the largest number of positive components, it is much more time consuming because it requires the solution of many relatively large linear systems before it finds the first and densest equilibrium strategy. Therefore, the denser the first equilibrium strategy, the less time consuming to find it. As shown in Table 3, the densest equilibrium strategy found for Game 6 has 14 positive components, and requires 107.01 s, while the densest equilibrium strategy found for Game 1 has 6 positive components and requires 773.53 s. In general, computing the densest equilibrium strategy seems much more expensive than computing the sparsest one. Note also that in Tables 2 and 3, we have listed the average values and standard deviations of the positive components of the strategies. It seems that the standard deviations are in the same order of the corresponding average values. Therefore, the positive components seem unevenly distributed. However, since the games are only randomly generated and our focus is on the performance of the algorithms, we will not further discuss the richness and evenness of these strategies. Figure 1 shows a more direct comparison in computation time for computing all the equilibrium strategies, or only the sparsest, or only the densest for all the test cases.

Table 3. Performance of Dense Shapley-Snow Algorithm.

Game	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
$\|\cdot\|_0$	6	11	9	10	13	14	9	7	10	13
μ	0.17	0.09	0.11	0.1	0.08	0.07	0.11	0.14	0.1	0.08
σ	0.16	0.05	0.09	0.09	0.06	0.06	0.07	0.14	0.08	0.06
Time	774	482	684	596	203	107	675	744	580	204

Table legend: Game—Tested 10 games; $\|\cdot\|_0$—The number of positive components in the densest equilibrium; μ—The average value of positive components; σ—The standard deviation of the positive components; Time—Time in seconds spent to obtain the densest equilibria.

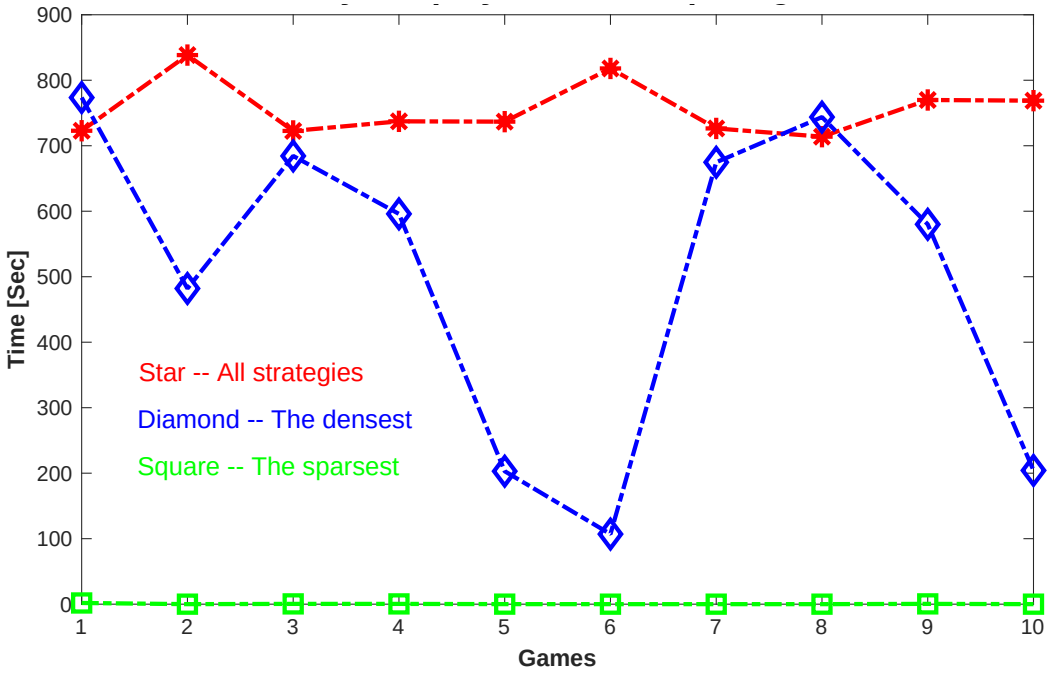

Figure 1. Performance Results: The computing times for obtaining all the equilibrium strategies, the densest, and the sparsest for 10 randomly generated games using the complete, sparse, and dense Shapley-Snow algorithms, respectively.

4. Complementary Formulation of Evolutionary Games

Consider the necessary and sufficient conditions in (4).

$$\begin{aligned} \lambda^* e - Ax^* \geq 0, \quad x^* \geq 0, \\ x^{*T}(\lambda^* e - Ax^*) = 0, \end{aligned} \tag{8}$$

which can be called a primary set of conditions. Let $y^* = \lambda^* e - Ax^*$. Assume that A is nonsingular. Then, $x^* = \lambda^* A^{-1} e - A^{-1} y^*$. Let $B = A^{-1}$. We can then obtain a so-called complementary set of conditions:

$$\begin{aligned} \lambda^* Be - By^* \geq 0, \quad y^* \geq 0, \\ y^{*T}(\lambda^* Be - By^*) = 0. \end{aligned} \tag{9}$$

Since $x^* y^* = 0,\ x^* \geq 0,\ y^* \geq 0$, we call x^* a primary equilibrium strategy and y^* a complementary.

Note that the conditions in (9) are equivalent to

$$\begin{aligned} \lambda^* B_i.e = B_i.y^*,\ if\ x_i^* = 0\ \&\ y_i^* > 0, \\ \lambda^* B_i.e = B_i.y^*,\ if\ x_i^* = 0\ \&\ y_i^* = 0, \\ \lambda^* B_i.e > B_i.y^*,\ if\ x_i^* > 0\ \&\ y_i^* = 0, \end{aligned} \tag{10}$$

where $B_i.$ is the ith row of matrix B.

Let $s = \{1, \ldots, n\}$ and $p = \{i :\ x_i^* > 0\ \&\ y_i^* = 0\}$ and $q = s \backslash p$. Then, based on the conditions in (10), $\lambda^* B_i.e = B_i.y^*$ for all $i \in q$. The latter can be written in a more compact form as $\lambda^* B_q.e = B_{qq} y_q^*$, where $y_q^* = \{y_i^* :\ i \in q\}$, and $B_{qq} = \{B_{i,j} :\ i, j \in q\}$. Since $x^* = \lambda^* Be - By^*$, $1 = \lambda^* e^T Be - e^T By^* = \lambda^* e^T Be - e^T B_{.q} y_q^*$. It follows that λ^* and y_q^* must satisfy the following equations.

$$1 = \lambda^* e^T Be - e^T B_{.q} y_q^*, \qquad (e^T BeB_{qq} - B_q.ee^T B_{.q}) y_q^* = B_q.e. \tag{11}$$

In addition, $\lambda^* B_i. > B_i.y^*$ for all $i \in p$, which can be simplified to

$$\lambda^* B_p.e > B_{pq} y_q^*, \tag{12}$$

where $B_{pq} = \{B_{i,j} :\ i \in p,\ j \in q\}$.

Since $x^* = \lambda^* Be - By^*$, we can derive x^* from y^* if we can find y^* using the conditions in (11) and (12). However, in order to find y^*, we do not have the prior knowledge on q. Therefore, we need to enumerate all possible set of indices q, to see if any gives rise to a complementary equilibrium strategy y^*. Once y^* is found, we can recover x^* immediately. We call this procedure a Complementary Shapley-Snow Algorithm. Let Q_k be the set of all subsets of $\{1, 2, \ldots, n\}$ of size k. Assume that $e^T Be \neq 0$. Then, the algorithm can be described formally as Algorithm 4.

Note that when a complementary equilibrium strategy y^* is found, $y_q^* \geq 0$, $x_q^* = 0$, and $x_p^* > 0$. It follows that $y_p^* = 0$, and x^* is a corresponding primary equilibrium strategy of $|p|$ positive components. Thus, as all the possible sizes of q are enumerated in Algorithm 4, the Complete Complementary Shapley-Snow Algorithm, all the possible complementary equilibrium strategies y^* can be found with $y_q^* \geq 0$ and $y_p^* = 0$, and hence are all the possible primary equilibrium strategies x^* with $x_p^* > 0$ and $x_q^* = 0$.

Algorithm 4: A Complete Complementary Shapley-Snow Algorithm

For $k = 1, \ldots, n$ **do**
  For each $q \in Q_k$ **do**
    If $(e^T B e B_{qq} - B_q . e e^T B_{.q})$ is nonsingular **then**
      $y_q^* = (e^T B e B_{qq} - B_q . e e^T B_{.q})^{-1} B_q . e$,
      $\lambda^* = (1 + e^T B_{.q} y_q^*)/e^T B e$.
    End
    If $y_q^* \geq 0$ and $\lambda^* B_p . e > B_{pq} y_q^*$ **then**
      y^* is a complementary equilibrium strategy.
    End
  End
End

5. Computing Dense Equilibria in Complementary Forms

Similar to the primary Shapley-Snow algorithm, the complete version of the complementary algorithm can be modified to compute the sparse as well as dense complementary equilibrium strategies of a given game. Assume that A is nonsingular. Let $B = A^{-1}$. In addition, assume that $e^T B e \neq 0$. We then have the Algorithms 5 and 6:

Algorithm 5: A Sparse Complementary Shapley-Snow Algorithm

For $k = 1, \ldots, n$ **do**
  For each $q \in Q_k$ **do**
    If $(e^T B e B_{qq} - B_q . e e^T B_{.q})$ is nonsingular **then**
      $y_q^* = (e^T B e B_{qq} - B_q . e e^T B_{.q})^{-1} B_q . e$,
      $\lambda^* = (1 + e^T B_{.q} y_q^*)/e^T B e$.
    End
    If $y_q^* \geq 0$ and $\lambda^* B_p . e > B_{pq} y_q^*$ **then**
      y^* is a complementary equilibrium strategy, **exit**.
    End
  End
End

Algorithm 6: A Dense Complementary Shapley-Snow Algorithm

For $k = n, \ldots, 1$ **do**
  For each $q \in Q_k$ **do**
    If $(e^T B e B_{qq} - B_q . e e^T B_{.q})$ is nonsingular **then**
      $y_q^* = (e^T B e B_{qq} - B_q . e e^T B_{.q})^{-1} B_q . e$,
      $\lambda^* = (1 + e^T B_{.q} y_q^*)/e^T B e$.
    End
    If $y_q^* \geq 0$ and $\lambda^* B_p . e > B_{pq} y_q^*$ **then**
      y^* is a complementary equilibrium strategy, **exit**.
    End
  End
End

Let x^* and y^* be a pair of primary and complementary equilibrium strategies. Since x^* and y^* are complementary to each other, if x^* is the sparsest primary equilibrium strategy, y^* must be the densest among all complementary equilibrium strategies. Similarly, if x^* is the densest, y^* must be the sparsest. As we have discussed in previous sections, dense equilibrium strategies can be computationally

more costly than the sparse ones. Therefore, they may be computed more efficiently through their complementary strategies, which are sparse. In particular, the densest primary equilibrium strategy x^* may be computed through the corresponding complementary strategy y^*. The latter is the sparsest among all complementary equilibrium strategies and can be found efficiently using Algorithm 5, the Sparse Complementary Shapley-Snow Algorithm. Indeed, as we can see in Table 4, the time for computing the densest equilibrium strategy x^* using Algorithm 3, the primary dense Shapley-Snow algorithm is much longer than that for computing the corresponding complementary strategy y^* using Algorithm 5, the sparse complementary Shapley-Snow algorithm. More specifically, the computation time is reduced by 7 to 15 times in 10 tested games. Note that the 10 tested games are the same randomly generated ones as shown in Tables 1–3. Figure 2 displays a more direct comparison between the primary and complementary algorithms for computing the densest equilibrium strategies of these games.

Table 4. Sparse Complementary vs. Primary Dense Shapley-Snow.

Game	1	2	3	4	5	6	7	8	9	10
$\|\cdot\|_0$	6	11	9	10	13	14	9	7	10	13
PrimDense	774	482	684	596	203	107	675	744	580	204
SparseCompl	114	48	86	67	15	7	86	108	68	15

Table legend: Game—Tested 10 games; $\|\cdot\|_0$—The number of positive components in the densest equilibrium; PrimDense—Time in seconds using Algorithm 3, the primary dense Shapley-Snow algorithm; SparseCompl—Time in seconds using Algorithm 5, the sparse complementary Shapley-Snow algorithm.

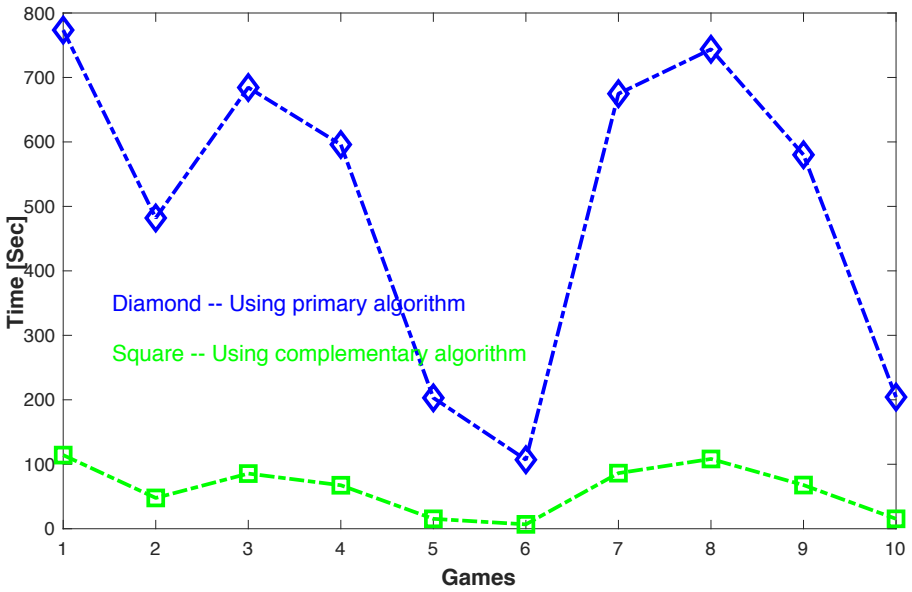

Figure 2. Performance Results: The computing times for obtaining the densest equilibrium strategies for 10 randomly generated games using the complementary vs. primary Shapley-Snow algorithms.

In order to obtain more statistical assessments, we have also tested the primary and complementary Shapley-Snow algorithms, i.e., Algorithm 3 and Algorithm 5, for computing the densest equilibrium strategies for an additional 100 randomly generated games, each again with 20 strategies. The results further confirm that the complementary algorithm outperforms the primary algorithm for all the test cases. More specifically, the average time required by the primary algorithm is 519.98 s, while the time by the complementary algorithm is only 53.88 s, about an order of magnitude faster. Figure 3 shows more detailed comparisons between the two algorithms. Note that the time required by both algorithms varies for different games, depending on the density of the densest equilibrium strategy of the game, but the relative difference in time between the two algorithms

remains about the same: In Figure 3, the x-axis corresponds to the time used by the primary algorithm, while the y-axis to that by the complementary algorithm. Each circle in the graph corresponds to the times required by the primary and complementary algorithms for one of the tested games. The density of the densest strategies found ranges from 4 to 16 and is color coded on the circles: The density, i.e., the richness or more specifically, the number of nonzero components of the strategy, denoted as $\|x^*\|_0$, increases in the order of 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, as the color of the circle changes in the order of dark blue, blue, light blue, brown, light brown, red, light red, pink, light pink, purple, light purple, orange, light orange, respectively. Note that all the circles are above the line $y = 0.0530x$ and below $y = 0.1381x$, showing that the computation time by the complementary algorithm can be as small as 5.3 percent of the time by the primary algorithm, and at most 13.81 percent. In average, the former is about 9.61 percent of the latter, as indicated by the median line $y = 0.096x$. Note also that in general, for both algorithms, the lower the density of the densest equilibrium strategy found, the longer the computation time. For the primary algorithm, this is because more relatively large systems need to be solved if the density of the densest strategy is lower. For the complementary algorithm, it means that the density of the corresponding complementary strategy is higher, which also requires the solution of more systems, although beginning with relatively small ones.

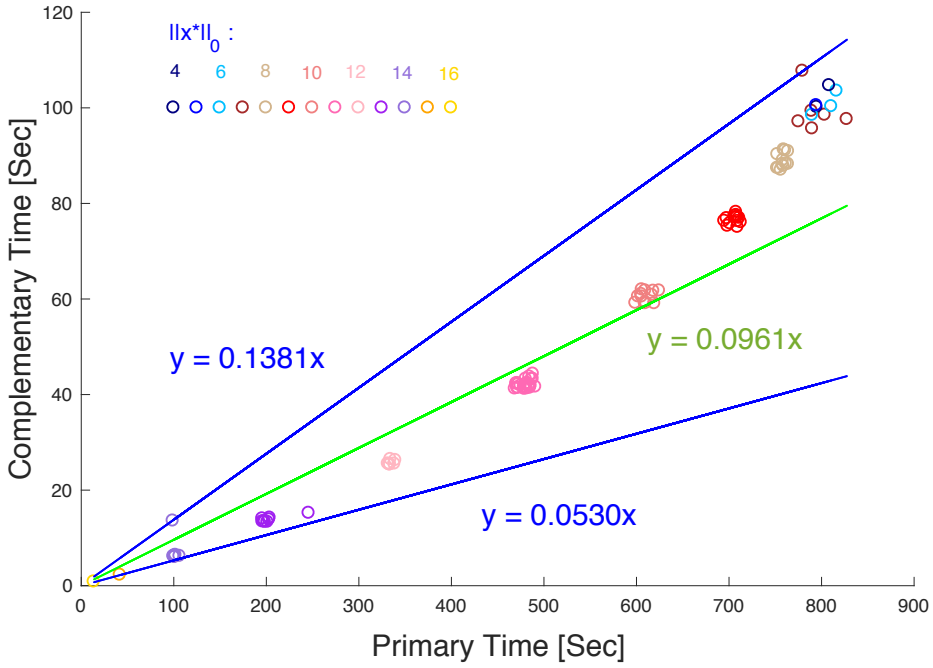

Figure 3. Performance Results: Shown in the figure are the computing times for obtaining the densest equilibrium strategies for 100 randomly generated games using the complementary vs. primary Shapley-Snow algorithms (i.e., Algorithm 5 vs. Algorithm 3). The x-axis corresponds to the time used by the primary algorithm, while the y-axis to that by the complementary algorithm. Each circle in the graph corresponds to the times required by the primary and complementary algorithms for one of the tested games. The density of the densest strategies found ranges from 4 to 16 and is color coded on the circles: The density, i.e., the richness or more specifically, the number of nonzero components of the strategy, denoted as $\|x^*\|_0$, increases in the order of 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, as the color of the circle changes in the order of dark blue, blue, light blue, brown, light brown, red, light red, pink, light pink, purple, light purple, orange, light orange, respectively. Note also that all the circles are above the line $y = 0.0530x$ and below $y = 0.1381x$, showing that the computation time by the complementary algorithm can be as small as 5.3 percent of the time by the primary algorithm, and at most 13.81 percent. On average, the former is about 9.61 percent of the latter, as indicated by the median line $y = 0.096x$.

Finally, as a special case, we have also tested a game related to allele selection in genetic studies [5,28–32]. Assume that there are n alleles at a given genetic locus. Let $x \in S$ be a selection strategy for the alleles, with x_i being the allele frequency for allele i. Let $G = (V, E)$ be a graph representing allele matching, where $V = \{1, 2, \ldots, n\}$ is a set of nodes corresponding to the alleles, and E is a set of links between the nodes. If allele i and j can make a successful genotype, then there is a link between node i and j. Then, an evolutionary game can be defined for allele selection with the adjacency matrix A for graph G as the payoff matrix, where $A_{i,i} = 1/2$ for all i, $A_{i,j} = 1$ for all $(i, j) \in E$ and $i \neq j$, and $A_{i,j} = 0$ for $(i, j) \notin E$ and $i \neq j$.

We have considered an allele selection game with $n = 20$, i.e., 20 alleles and A being the adjacency matrix for the graph shown in Figure 4. It is easy to verify that for this game, $x^* \in S$, with $x_i^* = 1/16$ for $i = 1, \ldots, 16$ and $x_i^* = 0$ for $i = 17, \ldots, 20$, is an equilibrium strategy, as it satisfies all the necessary and sufficient conditions for equilibrium strategies as stated in (4). It turns out that x^* is also the densest equilibrium strategy for this game. We have thus applied both primary (dense) and (sparse) complementary Shapley-Snow algorithms to compute the densest equilibrium strategy for this game and recorded the computing times. Our results showed that both algorithms recovered the densest equilibrium strategy x^* accurately. However, the complementary Shapley-Snow took only 0.89, 0.93, 0.88 s in three runs, while the primary Shapley Snow required 14.11, 14.20, 14.45 s, respectively, again suggesting that the complementary algorithm can be more or less an order of magnitude more efficient than the primary one for computing the densest equilibrium strategy for a game.

Note that in all our test cases, we should be able to obtain the same set of equilibrium strategies for each game using either primary or complementary algorithms, including the sparsest and densest ones. In other words, if we obtain a set of equilibrium strategies x^* for a given game using the primary algorithm, we should be able to obtain all the corresponding complementary strategies y^* using the complementary algorithm. In this way, we can obtain from all y^* found by the complementary algorithm the same set of equilibrium strategies x^* found by the primary algorithm, and vice versa. Otherwise, the sparsest equilibrium strategy x^* found in the primary dense algorithm may not necessarily be found in the sparse complementary algorithm, if the latter cannot find the corresponding complementary strategy y^*. This property is important, but is not so obviously shown in the descriptions of the algorithms. We therefore provide a more formal verification in the following.

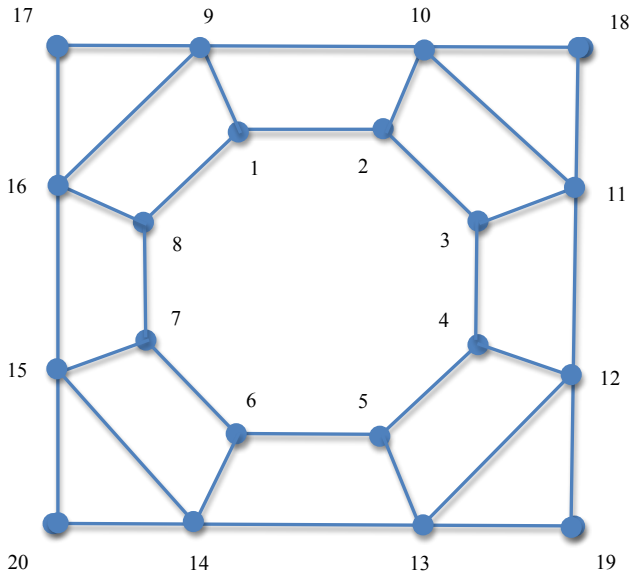

Figure 4. Genetic Matching Graph: There are 20 nodes in the graph, with node i representing allele i, $i = 1, \ldots, 20$. If allele i and j make a successful genotype, there is a link between node i and j.

As required by the algorithms, we assume that payoff matrix A is nonsingular. Let $s = \{1, \ldots, n\}$. We then show that for each equilibrium strategy x^* with positive components x_p^* that is found by the primary Shapley-Snow algorithm, a corresponding complementary strategy y^* with $y_p^* = 0$ and $y_q^* \geq 0$ can be found by the complementary Shapley-Snow algorithm, where $p = \{i : \ x_i^* > 0\}$, and $q = s \backslash p$. Note that in the primary algorithm, an equilibrium strategy x^* with positive components x_p^* is found if A_{pp} is nonsingular and $e_p^T A_{pp}^{-1} e_p \neq 0$. On the other hand, the corresponding complementary strategy y^* with $y_p = 0$ and $y_q^* \geq 0$ is obtained by the complementary algorithm if $(e^T B e B_{qq} - B_q . e e^T B_{.q})$ is nonsingular and $e^T B e \neq 0$, where $B = A^{-1}$. Therefore, all we need to show is that for the given pair of p and q, A_{pp} is nonsingular and $e_p^T A_{pp}^{-1} e_p \neq 0$ if and only if $(e^T B e B_{qq} - B_q . e e^T B_{.q})$ is nonsingular and $e^T B e \neq 0$, which follows immediately from the following two theorems.

Theorem 1. *Let p and q be two subsets of indices, and $p = s \backslash q$. Assume that A is nonsingular. Then, B_{qq} is nonsingular if and only if A_{pp} is nonsingular, where $B_{qq} = \{B_{i,j} : \ i, j \in q\}$ and $A_{pp} = \{A_{i,j} : \ i, j \in p\}$.*

Proof. If A_{pp} is nonsingular, then $C_{qq} = A_{qq} - A_{qp} A_{pp}^{-1} A_{pq}$ is a Shur's complement of A [33]. Since A is nonsingular, $det\{A\} = det\{A_{pp}\} det\{C_{qq}\} \neq 0$. It follows that $det\{C_{qq}\} \neq 0$ and C_{qq} is nonsingular. Since $B_{qq} = C_{qq}^{-1}$, B_{qq} is nonsingular. Reversely, if B_{qq} is nonsingular, then $D_{pp} = B_{pp} - B_{pq} B_{qq}^{-1} B_{qp}$ is a Shur's complement of B. Since B is nonsingular, $det\{B\} = det\{B_{qq}\} det\{D_{pp}\} \neq 0$. It follows that $det\{D_{pp}\} \neq 0$ and D_{pp} is nonsingular. Since $A_{pp} = D_{pp}^{-1}$, A_{pp} is nonsingular. □

Theorem 2. *Let A be a payoff matrix. Assume that A is nonsingular, $B = A^{-1}$, and $e^T B e \neq 0$. Let $E_{qq} = e^T B e B_{qq} - B_q . e e^T B_{.q}$. Then, E_{pp} is nonsingular if and only if A_{pp} is nonsingular and $e_p^T A_{pp}^{-1} e_p \neq 0$.*

Proof. If $e^T B e \neq 0$, it suffices to show that

$$F_{qq} = B_{qq} - \frac{B_q . e e^T B_{.q}}{e^T B e} \tag{13}$$

is nonsingular if and only if A_{pp} is nonsingular and $e_p^T A_{pp}^{-1} e_p \neq 0$.

Let $u_q = B_q . e$ and $v_q = B_{.q}^T e / e^T B e$. Then

$$F_{qq} = B_{qq} - u_q v_q^T. \tag{14}$$

By Sherman-Morrison-Woodbery Formula [33], F_{qq} is nonsinglar if and only if B_{qq} is nonsingular and $1 - v^T B_{qq}^{-1} u \neq 0$. In addition,

$$F_{qq}^{-1} = B_{qq}^{-1} + \frac{B_{qq}^{-1} u v^T B_{qq}^{-1}}{1 - v^T B_{qq}^{-1} u}. \tag{15}$$

By Theorem 1, B_{qq} is nonsingular if and only if A_{pp} is nonsingular. In addition,

$$\begin{aligned} & 1 - v^T B_{qq}^{-1} u \qquad (16) \\ = \ & 1 - \frac{e^T B_{.q} B_{qq}^{-1} B_q . e}{e^T B e} \\ = \ & \frac{e^T (B - B_{.q} B_{qq}^{-1} B_q .) e}{e^T B e} \\ = \ & \frac{e_p^T (B_{pp} - B_{pq} B_{qq}^{-1} B_{qp}) e_p}{e^T B e} \end{aligned}$$

Since $B_{pp} - B_{pq}B_{qq}^{-1}B_{qp}$ is a Shur's complement of B, $(B_{pp} - B_{pq}B_{qq}^{-1}B_{qp})^{-1} = (B^{-1})_{pp}$. But $B^{-1} = A$. Therefore, $(B_{pp} - B_{pq}B_{qq}^{-1}B_{qp})^{-1} = A_{pp}$ and

$$1 - v^T B_{qq}^{-1} u = \frac{e_p^T A_{pp}^{-1} e_p}{e^T B e}. \tag{17}$$

It follows that $1 - v^T B_{qq}^{-1} u \neq 0$ if and only if $e_p^T A_{pp}^{-1} e_p \neq 0$. □

Again, an equilibrium strategy x_p^* can be found in either the primary algorithm under the conditions A_{pp} is nonsingular and $e_p^T A_{pp}^{-1} e_p \neq 0$ or in the complementary algorithm with its complementary strategy y_q^* under the conditions B_{qq} is nonsingular and $e^T B e \neq 0$. Based on the above two theorems, the two sets of conditions are equivalent and therefore, x_p^* can indeed be found by either algorithm, and so is y_q^*.

6. Concluding Remarks

In this paper, we have considered the problem of computing the equilibrium strategies of a given evolutionary game, complete, sparse, or dense. We are particularly interested in computing the dense equilibrium strategies, for they may represent more diverse ecological conditions in biology or more even distributions of funds in financial investment. If such system states can be determined, further analysis on related properties such as the stabilities of the states and their dynamic behaviors may provide great insights into maintaining the diversities of the system and preventing certain species from extinction. In any case, the dense equilibrium strategies are more costly to compute than the sparse ones as we have demonstrated in the paper. We have therefore formulated a complementary version of the game for a given primary one, and shown that computing a dense strategy for the primary version then becomes computing a sparse strategy in the complementary version, which can be done much more efficiently.

We have implemented the primary and complementary Shapley-Snow algorithms in Matlab for all the complete, sparse, and dense versions. We have tested these algorithms on randomly generated games and analyzed their performance. In particular, we have shown that the complementary algorithm is on average about 10 times faster than the primary algorithm for finding the densest equilibrium strategies for the given games. We have also particularly tested a more realistic game related to allele selection in genetic studies. The results on this game are consistent with those for randomly generated games: The computing time to find the densest equilibrium strategy for the game using the complementary algorithm is more or less an order of magnitude more efficient than using the primary algorithm.

In our algorithms, the conditions under which a primary equilibrium strategy x^* is found are different from those under which the corresponding complementary equilibrium strategy y^* is found. Therefore, it is not clear if y^* can always be found by the complementary algorithm given x^* found by the primary algorithm, and vice versa. In other words, if a set of equilibrium strategies x^* is obtained using the primary algorithm, it is not clear if all the corresponding complementary strategies y^* can be found using the complementary algorithm, and vice versa. In any case, in the end of Section 5, we have provided a formal justification showing that the two sets of conditions are equivalent, and the two algorithms should be able to produce and only produce all the corresponding equilibrium strategies. Note that for our complementary algorithms to work, we did assume the payoff matrix A to be nonsingular, which will limit to some extent the applicability of the algorithms.

The focus of this paper is on computation and especially on computation of dense equilibrium strategies. Many related issues have not yet been addressed. For example, we have only concerned ourselves with the richness of the strategies but not with the evenness, which can be important in practice. Further development of algorithms that can find both rich and even equilibrium strategies can be interesting. In economic application, the payoff of a strategy may correspond to the risk of investment. The higher the payoff, the lower the risk. Therefore, it would be interesting to find

not only a dense strategy but also a dense strategy with the highest possible payoff. Then, a more extensive search for such a strategy may be required in the current algorithms. Finally, we have not considered the evolutionary stability of equilibrium strategies, either sparse or dense, while it is of great concern in both biological and economic applications. We will investigate how to compute sparse and dense equilibrium strategies efficiently while at the same time justifying their stabilities in our future research efforts.

Author Contributions: Y.H. and Z.W. developed and analyzed the algorithms. Y.H. performed all computational work. Y.H. and Z.W. finished the paper.

Funding: This work is partially funded by the NIH/NIGMS grant R01GM072014 and by the NSF/DMS grant DMS0914354.

Acknowledgments: The authors would like to thank the anonymous referees and the associate editors for their patiently reading the paper and providing many valuable suggestions for the revision of the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hall, B.; Hallgrimsson, B. *Strickberger's Evolution*, 5th ed.; Jones & Bartlett Learning: Burlington, MA, USA, 2013.
2. Smith, M.J.; Price, G.R. The logic of animal conflict. *Nature* **1973**, *246*, 15–18. [CrossRef]
3. Smith, M.J. *Evolution and the Theory of Games*; Cambridge University Press: Cambridge, UK, 1982.
4. Weibull, J.W. *Evolutionary Game Theory*; MIT Press: Cambridge, MA, USA, 1995.
5. Hofbauer, J.; Sigmund, K. *Evolutionary Games and Population Dynamics*; Cambridge University Press: Cambridge, UK, 1998.
6. Webb, J.N. *Game Theory: Decisions, Interaction and Evolution*; Springer: Berlin, Germany, 2007.
7. Nash, J. Equilibrium points in n-person games. *Proc. Natl. Acad. Sci. USA* **1950**, *36*, 48–49. [CrossRef] [PubMed]
8. Nash, J. Non-cooperative games. *Ann. Math.* **1951**, *54*, 286–295. [CrossRef]
9. Daskalakis, C.; Papadimitriou, C.H. On oblivious PTAS's for Nash equilibrium. In Proceedings of the 41st Annual ACM Symposium on Theory of Computing, (STOC 2009), Bethesda, MD, USA, 31 May–2 June 2009; ACM: New York, NY, USA, 2009; pp. 75–84.
10. Gilboa, I.; Zemel, E. Nash and correlated equilibria: Some complexity considerations. *Games Econ. Behav.* **1989**, *1*, 80–93. [CrossRef]
11. Conitzer, V.; Sandholm, T. Complexity results about Nash equilibria. In Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003), Acapulco, Mexico, 9–15 August 2003; pp. 765–771.
12. Chen, X.; Deng, X.; Teng, S. Sparse games are hard. In Proceedings of the 2nd International Workshop on Internet and Network Economics, (WINE2006), Patras, Greece, 15–17 December 2006; pp. 262–273.
13. Motzkin, T.S.; Straus, E.G. Maxima for graphs and a new proof of a theorem of Turan. *Can. J. Math.* **1965**, *17*, 533–540. [CrossRef]
14. Pardalos, P.M.; Ye, Y.; Han, C. Algorithms for the solution of quadratic knapsack problems. *Linear Algebra Its Appl.* **1991**, *152*, 69–91. [CrossRef]
15. Bomze, I.M.; Budinich, M.; Pardalos, P.M.; Pelillo, M. The maximum clique problem. In *Handbook of Combinatorial Optimization*; Du, D.Z., Pardalos, P.M., Eds.; Springer: Boston, MA, USA, 1999; pp. 1–74.
16. Shapley, L.; Snow, R. Basic solutions of discrete games. *Ann. Math. Stud.* **1950**, *24*, 27–35.
17. Lemke, C.E.; Howson, J.T., Jr. Equilibrium points and bimatrix games. *J. SIAM* **1964**, *12*, 413–423. [CrossRef]
18. Mangasarian, O.L.; Stone, H. Two-person, nonzero-sum games and quadratic programming. *J. Math. Anal. Appl.* **1964**, *9*, 348–355. [CrossRef]
19. Tibshirani, R. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B* **1996**, *58*, 266–288.
20. Zou, H.; Hastie, T. Regularization and variable selection via the elastic net. *J. R. Stat. Soc. Ser. B* **2003**, *67*, 301–320. [CrossRef]
21. Donoho, D.L. For most large underdetermined systems of linear equations the minimal 1-norm solution is also the sparsest solution. *Commun. Pure Appl. Math.* **2006**, *59*, 797–829. [CrossRef]

22. Candes, E.J.; Romberg, J.K.; Tao, T. Stable signal recovery from incomplete and inaccurate measurements, *Commun. Pure Appl. Math.* **2006**, *59*, 1207–1223. [CrossRef]
23. Candes, E.J.; Wakin, M.B.; Boyd, S.P. Enhancing sparsity by reweighted L1 minimization. *J. Fourier Anal. Appl.* **2008**, *14*, 877–905. [CrossRef]
24. Zhang, Y. Theory of compressive sensing via l1-minimization: A non-RIP analysis and extensions. *J. Oper. Res. Soc. China* **2003**, *1*, 79–105. [CrossRef]
25. Gaston, K.; Spicer, J. *Biodiversity: An Introduction*; Wiley-Blackwell: Hoboken, NJ, USA, 2004.
26. Markowitz, H.M. Portfolio selection. *J. Financ.* **1952**, *7*, 77–91.
27. Elton, E.; Gruber, M. *Modern Portfolio Theory and Investment Analysis*; Wiley: Hoboken, NJ, USA, 2014.
28. Vickers, G.T.; Cannings, C. On the number of stable equilibria in a one-locus, multi-allelic system. *J. Theor. Biol.* **1988**, *131*, 273–277. [CrossRef]
29. Vickers, G.T.; Cannings, C. Patterns of ESS's I. *J. Theor. Biol.* **1988**, *132*, 387–408. [CrossRef]
30. Cannings, C.; Vickers, G.T. Patterns of EES's II. *J. Theor. Biol.* **1988**, *132*, 409–420. [CrossRef]
31. Bomze, I.M. Evolution towards the maximum clique. *J. Glob. Optim.* **1997**, *10*, 143–164. [CrossRef]
32. Burger, R. *The Mathematical Theory of Selection, Recombination, and Mutation*; Wiley: Hoboken, NJ, USA, 2000.
33. Horn, R.; Johnson, C. *Matrix Analysis*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2013.