Payoff-Monotonic Game Dynamics and the Maximum Clique Problem

Marcello Pelillo
pelillo@dsi.unive.it
Andrea Torsello
torsello@dsi.unive.it
Dipartimento di Informatica, Università Ca' Foscari di Venezia 30172 Venezia Mestre, Italy

Evolutionary game-theoretic models and, in particular, the so-called replicator equations have recently proven to be remarkably effective at approximately solving the maximum clique and related problems. The approach is centered around a classic result from graph theory that formulates the maximum clique problem as a standard (continuous) quadratic program and exploits the dynamical properties of these models, which, under a certain symmetry assumption, possess a Lyapunov function. In this letter, we generalize previous work along these lines in several respects. We introduce a wide family of game-dynamic equations known as *payoff-monotonic* dynamics, of which replicator dynamics are a special instance, and show that they enjoy precisely the same dynamical properties as standard replicator equations. These properties make any member of this family a potential heuristic for solving standard quadratic programs and, in particular, the maximum clique problem. Extensive simulations, performed on random as well as DIMACS benchmark graphs, show that this class contains dynamics that are considerably faster than and at least as accurate as replicator equations. One problem associated with these models, however, relates to their inability to escape from poor local solutions. To overcome this drawback, we focus on a particular subclass of payoff-monotonic dynamics used to model the evolution of behavior via imitation processes and study the stability of their equilibria when a regularization parameter is allowed to take on negative values. A detailed analysis of these properties suggests a whole class of annealed imitation heuristics for the maximum clique problem, which are based on the idea of varying the parameter during the imitation optimization process in a principled way, so as to avoid unwanted inefficient solutions. Experiments show that the proposed annealing procedure does help to avoid poor local optima by initially driving the dynamics toward promising regions in state space. Furthermore, the models outperform state-of-the-art neural network algorithms for maximum clique, such as mean field annealing, and compare well with powerful continuous-based heuristics.

1 Introduction

Research in computational complexity has shown that many problems of practical interest are inherently intractable, in the sense that it is not possible to find fast (i.e., polynomial time) algorithms that solve them exactly, unless the classes P and NP coincide, this being believed to be highly unlikely. In some cases, we can indeed find good approximate solutions in polynomial time (Papadimitriou & Steiglitz, 1982), but unfortunately, it turns out that certain important problems remain intractable even to approximate. This is the case with the maximum clique problem (MCP), a classic problem in combinatorial optimization that asks for the largest complete subgraph of a given graph. Indeed, the best polynomial-time approximation algorithm for the MCP achieves an approximation ratio of $n^{1-o(1)}$ (Boppana & Halldórsson, 1992), where n is the number of vertices in the graph, and Hastad (1996) has shown that this is actually the best we can achieve by proving that unless $NP = coR$, the MCP cannot be approximated within a factor of $n^{1-\varepsilon}$, for any $\varepsilon > 0$. Although this complexity result characterizes worst-case instances, it nevertheless indicates that the MCP is indeed a very difficult problem to solve.[1] Due to this pessimistic state of affairs and because of its important applications in such diverse domains such as computer vision, experimental design, information retrieval, and fault tolerance, much attention has recently gone into developing efficient heuristics for the MCP, for which no formal guarantee of performance may be provided but are nevertheless useful in practical applications. We refer to Bomze, Budinich, Pardalos, & Pelillo (1999) for a survey concerning algorithms, applications, and complexity issues of this important problem.

In the neural network community, there has been much interest around the maximum clique problem. Early attempts at encoding this and related problems in terms of a neural network were done in the late 1980s by Ballard, Gardner, and Srinivas (1987), Godbeer, Lipscomb, and Luby (1988), Ramanujam and Sadayappan (1988), Aarts and Korst (1989), and Shrivastava, Dasgupta, and Reddy (1990; see also Shrivastava, Dasgupta, & Reddy, 1992). However, little or no experimental results were presented, thereby making it difficult to evaluate the merits of these algorithms. Lin and Lee (1992) used a quadratic zero-one formulation from Pardalos and Rodgers (1990) as the basis for their neural network heuristic. For an n-vertex graph, they used a network with $2(n+1)$ computational nodes, and real-valued connection weights.

Grossman (1996) proposed a discrete, deterministic version of the Hopfield model for maximum clique, originally designed for an all-optical implementation. The model has a threshold parameter that determines the

[1] See Grötschel, Lovász, & Schrijver (1993) for classes of graphs for which the MCP can be solved in polynomial time.

character of the stable states of the network. The author suggests an annealing strategy on this parameter and an adaptive procedure to choose the network's initial state and threshold. On DIMACS graphs, the algorithm performs satisfactorily but does not compare well with more powerful heuristics such as simulated annealing.

Jagota (1995) developed several variations of the Hopfield model, both discrete and continuous, to approximate maximum clique. He evaluated the performance of his algorithms over randomly generated graphs as well as on harder graphs obtained by generating cliques of varying size at random and taking their union. Experiments on graphs coming from the Solomonoff-Levin, or "universal" distribution, are also presented in Jagota and Regan (1997). The best results were obtained using a stochastic steepest-descent dynamics and a mean field annealing algorithm, an efficient, deterministic approximation of simulated annealing. These algorithms, however, were also the slowest, and this motivated Jagota, Sanchis, and Ganesan (1996) to improve their running time. The mean field annealing heuristic was implemented on a 32-processor connection machine, and a two-temperature annealing strategy was used. Additionally, a reinforcement learning strategy was developed for the stochastic steepest-descent heuristic, to automatically adjust its internal parameters as the process evolves. On various benchmark graphs, all their algorithms obtained significantly larger cliques than other simpler heuristics but ran slightly slower. Compared to more sophisticated heuristics, they obtained significantly smaller cliques on average but were considerably faster.

Other attempts at solving the maximum clique problem using Hopfield-style neural networks can be found in Takefuji, Chen, Lee, and Huffman (1990), Funabiki, Takefuji, and Lee (1992), Wu, Harada, and Fukao (1994), Bertoni, Campadelli, and Grossi (2002), Pekergin, Morgül, and Güzelis (1999), Jagota, Pelillo, and Rangarajan (2000), and Wang, Tang, and Cao (2003). Almost invariably, all these works formulate the MCP in terms of an integer (usually 0-1) programming problem and use some variant of the Hopfield model to solve it.

In a recent series of papers (Pelillo, 1995, 1999, 2002; Bomze, 1997; Bomze, Pelillo, & Giacomini, 1997; Bomze, Pelillo, & Stix, 2000; Jagota et al., 2000; Bomze, Budinich, Pelillo, & Rossi, 2002; Pelillo, Siddiqi, & Zucker, 1999), a completely different framework has been developed. The approach is centered around a classic result from graph theory due to Motzkin and Straus (1965), and variations thereof, which allow us to formulate the MCP as a standard quadratic program—namely, a continuous quadratic optimization problem with simplex (or probability) constraints, to solve which replicator equations have been remarkably effective despite their simplicity. These are well-known continuous- and discrete-time dynamical systems developed and studied in evolutionary game theory, a discipline pioneered by J. Maynard Smith (1982) that aims to model the evolution of animal behavior using the principles and tools of noncooperative game theory

(Hofbauer & Sigmund, 1998). Evolutionary game-theoretic models are also gaining increasing popularity in economics since they elegantly get rid of the much-debated assumptions of traditional game theory concerning the full rationality and complete knowledge of players (Weibull, 1995; Samuelson, 1997; Fudenberg & Levine, 1998). Interestingly, these dynamical equations also turn out to be related to the so-called relaxation labeling processes, a class of parallel, distributed algorithms developed in computer vision to solve (continuous) constraint satisfaction problems (Rosenfeld, Hummel, & Zucker, 1976; Hummel & Zucker, 1983). An independent connection between dynamical systems such as relaxation labeling and Hopfield-style networks, and game theory has been described by Miller and Zucker (1992, 1999).

This letter substantially expands on previous work along these lines. We introduce a wide family of evolutionary game dynamics, of which replicator equations represent just a special instance, characterized by having the growth rates of strategies ordered by their expected payoffs, so that strategies associated with higher payoffs grow faster. It is shown that these payoff-monotonic models enjoy precisely the same dynamical properties as standard, first-order replicator equations. In particular, when the payoff matrix is symmetric, they possess a quadratic Lyapunov function, which is strictly increasing along any nonconstant trajectory; furthermore, it is shown that their asymptotically stable stationary points are in one-to-one correspondence with (strict) local solutions of standard quadratic programs. We then specialize our discussion to a parameterized family of such quadratic problems arising from the MCP, which include the Motzkin-Straus formulation as a special case, and show that a one-to-one correspondence exists between its local-global solutions and maximal-maximum cliques of the corresponding graph, provided that its parameter is positive (and less than 1). These properties therefore make any member of the payoff-monotonic family a potential heuristic for the MCP. In particular, we present extensive experimental results obtained with an exponential version of the standard replicator dynamics over hundreds of random as well as DIMACS benchmark graphs, and show that these dynamics are dramatically faster than their first-order counterpart and even more accurate. They also compare favorably with other simple neural network heuristics and obtain only slightly worse results than more sophisticated ones, such as mean field annealing.

As with standard replicator equations, however, these models are inherently unable to escape from inefficient local solutions or, in other words, from small maximal cliques. Although this is not necessarily a problem when dealing with graphs arising from graph isomorphism of maximum common subtree problems (Pelillo, 1999, 2002; Pelillo et al., 1999), it makes them unsuited for harder problem instances. In an attempt to overcome this drawback, in the second part of the article, we focus on a well-known subclass of payoff-monotonic dynamics that arises in modeling the evolution of

behavior by way of imitation processes. Following Bomze et al. (2002), we investigate the properties of our parameterized Motzkin-Straus program when its parameter is allowed to take on negative values. In this case, an interesting picture emerges: as its absolute value grows larger, local maximizers corresponding to maximal cliques disappear, that is, they become unstable under any imitation dynamics. We derive bounds on the parameter that affects the stability of the solutions, and these results, which generalize those presented in Bomze et al. (2002), suggest a whole family of annealed imitation heuristics, which consist of starting from a large negative value of the parameter and then properly reducing it during the optimization process. At each step, imitation dynamics are run in order to obtain a local solution of the corresponding objective function. The rationale behind this idea is that at large absolute values of the annealing parameter, only local solutions corresponding to large maximal cliques will survive, together with various spurious maximizers. As the value of the parameter is reduced, spurious solutions disappear and smaller maximal cliques become stable. A similar idea has been proposed by Gee and Prager (1994) in a different context.

Experiments conducted on both random and DIMACS graphs using an exponential imitation dynamics confirm the effectiveness of the proposed approach and the robustness of the annealing strategy. The overall conclusion is that the annealing procedure does help to avoid inefficient local solutions by initially driving the dynamics toward promising regions in state space and then refining the search as the annealing parameter is increased. Moreover, the algorithm outperforms state-of-the-art neural network heuristics for maximum clique, such as mean field annealing, and other heuristics based on Motzkin-Straus and related formulations.

2 Payoff-Monotonic Game Dynamics

Evolutionary game theory considers an idealized scenario whereby in a large population, pairs of individuals are repeatedly drawn at random to play a symmetric two-player game. In contrast to traditional game-theoretic models, players are not supposed to behave rationally or have complete knowledge of the details of the game. They act instead according to a preprogrammed behavior pattern, or pure strategy, and it is supposed that some evolutionary selection process operates over time on the distribution of behaviors. (We refer the reader to Hofbauer & Sigmund, 1998, and Weibull, 1995, for excellent introductions to this rapidly expanding field.)

Let $J = \{1, \ldots, n\}$ be the set of available pure strategies, and for all $i \in J$, let $x_i(t)$ be the proportion of population members playing strategy i, at time t. The state of the population at a given instant is the vector $\mathbf{x} = (x_1, \ldots, x_n)'$, where a prime denotes transposition. Clearly, population states are

constrained to lie in the standard simplex of the n–dimensional Euclidean space $\mathbb{R}^n$:

$$\Delta = \{\mathbf{x} \in \mathbb{R}^n \ : \ x_i \geq 0 \text{ for all } i \in J\,, \mathbf{e}'\mathbf{x} = 1\},$$

where $\mathbf{e} = \sum_i \mathbf{e}_i = (1, \ldots, 1)'$, and $\mathbf{e}_i$ denotes the ith standard basis vector in $\mathbb{R}^n$. The support of a population state $\mathbf{x} \in \Delta$, denoted by $\sigma(\mathbf{x})$, is defined as the set of indices corresponding to its positive components, which correspond to nonextinct strategies:

$$\sigma(\mathbf{x}) = \{i \in J \ : \ x_i > 0\}\,.$$

Given a subset of strategies $S \subseteq J$, the set of states where all strategies outside S are extinct, which corresponds to a face of Δ, is defined as

$$\Delta_S = \{\mathbf{x} \in \Delta \ : \ \sigma(\mathbf{x}) \subseteq S\},$$

and its (relative) interior is

$$int(\Delta_S) = \{\mathbf{x} \in \Delta \ : \ \sigma(\mathbf{x}) = S\}.$$

Clearly, $\Delta_J = \Delta$, and, accordingly, we shall write int(Δ) instead of int(Δ_J).

Let $A = (a_{ij})$ be the $n \times n$ payoff or utility matrix. Specifically, for each pair of strategies $i, j \in J\,, a_{ij}$ represents the payoff of an individual playing strategy i against an opponent playing strategy j. In biological contexts, payoffs are typically measured in terms of Darwinian fitness or reproductive success (i.e., the player's expected number of offspring), whereas in economic applications, they usually represent firms' profits or consumers' utilities. If the population is in state $\mathbf{x}$, the expected payoff earned by an i-strategist is:

$$\pi_i(\mathbf{x}) = \sum_{j=1}^{n} a_{ij} x_j = (A\mathbf{x})_i, \tag{2.1}$$

while the mean payoff over the entire population is

$$\pi(\mathbf{x}) = \sum_{i=1}^{n} x_i \pi_i(\mathbf{x}) = \mathbf{x}' A\mathbf{x}. \tag{2.2}$$

In evolutionary game theory, the assumption is made that the game is played over and over, generation after generation, and that the action of natural selection will result in the evolution of the fittest strategies. If

successive generations blend into each other, the evolution of behavioral phenotypes can be described by a set of ordinary differential equations. A general class of evolution equations is given by

$$\dot{x}_i = x_i g_i(\mathbf{x}), \tag{2.3}$$

where a dot signifies derivative with respect to time, and $g = (g_1, \ldots, g_n)$ is a function with open domain containing Δ. Here, the function g_i ($i \in J$) specifies the rate at which pure strategy i replicates when the population is in state $\mathbf{x}$. It is usually required that the growth function g is regular (Weibull, 1995), which means that it is C^1 and that $g(\mathbf{x})$ is always orthogonal to $\mathbf{x}$—$g(\mathbf{x})'\mathbf{x} = 0$. The former condition guarantees us that the system of differential equations 2.3 has a unique solution through any initial population state.[2] The condition $g(\mathbf{x})'\mathbf{x} = 0$ instead ensures that the simplex Δ is invariant under equation 2.3, namely, any trajectory starting in Δ will remain in Δ.

A point $\mathbf{x}$ is said to be a *stationary* (or equilibrium) point of our dynamical system if $\dot{x}_i = 0$, for all $i = 1 \ldots n$. A stationary point $\mathbf{x}$ is said to be *Lyapunov stable* (or, more simply *stable*) if for any neighborhood U of $\mathbf{x}$ there exists a neighborhood W of $\mathbf{x}$ such that any trajectory in W remains also in U (formally, $\mathbf{x}(0) \in W$ implies $\mathbf{x}(t) \in U$ for all $t \geq 0$). It is said to be *asymptotically stable* if, in addition, such trajectories converge to $\mathbf{x}$.

Payoff-monotonic game dynamics represent a wide class of regular selection dynamics for which useful properties hold. Intuitively, for a payoff-monotonic dynamics, the strategies associated with higher payoffs will increase at a higher rate. Formally, a regular selection dynamics 2.3 is said to be payoff-monotonic if

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \;\Leftrightarrow\; \pi_i(\mathbf{x}) > \pi_j(\mathbf{x}) \tag{2.4}$$

for all $\mathbf{x} \in \Delta$.

Although this class contains many different dynamics, it turns out that they share a lot of common properties. To begin, they all have the same set of stationary points.

Proposition 1. *A point $\mathbf{x} \in \Delta$ is stationary under any payoff-monotonic dynamics if and only if $\pi_i(\mathbf{x}) = \pi(\mathbf{x})$ for all $i \in \sigma(\mathbf{x})$.*

Proof. See Weibull (1995).

[2] Indeed, to ensure existence and uniqueness of solutions to equation 2.3, it is sufficient that g is (locally) Lipschitz continuous, that is, there exists a constant K such that $\|g(\mathbf{x}) - g(\mathbf{y})\| \leq K\|\mathbf{x} - \mathbf{y}\|$ for all $\mathbf{x}$, $\mathbf{y}$ in any compact subset of the domain of g (see, e.g., Hirsch & Smale, 1974). It is well known that C^1 functions are also locally Lipschitz continuous.

A well-known subclass of payoff-monotonic game dynamics is given by

$$\dot{x}_i = x_i \left(\phi(\pi_i(\mathbf{x})) - \sum_{j=1}^{n} x_j \phi(\pi_j(\mathbf{x})) \right) \tag{2.5}$$

where $\phi(u)$ is an increasing function of u. These models arise in modeling the evolution of behavior by way of imitation processes, where players are occasionally given the opportunity to change their own strategies (Hofbauer, 1995; Weibull, 1995).

When ϕ is the identity function, that is, $\phi(u) = u$, we obtain the standard replicator equations,

$$\dot{x}_i = x_i \left(\pi_i(\mathbf{x}) - \sum_{j=1}^{n} x_j \pi_j(\mathbf{x}) \right), \tag{2.6}$$

whose basic idea is that the average rate of increase $\dot{x}_i / x_i$ equals the difference between the average fitness of strategy i and the mean fitness over the entire population.

Another popular model arises when $\phi(u) = e^{\kappa u}$, which yields:

$$\dot{x}_i = x_i \left(e^{\kappa \pi_i(\mathbf{x})} - \sum_{j=1}^{n} x_j e^{\kappa \pi_j(\mathbf{x})} \right), \tag{2.7}$$

where κ is a positive constant. As κ tends to 0, the orbits of this dynamics approach those of the standard, first-order replicator model, equation 2.6, slowed down by the factor κ; moreover, for large values of κ, the model approximates the so-called best-reply dynamics (Hofbauer, 1995; Hofbauer & Sigmund, 1998).

3 Payoff-Monotonic Dynamics and Quadratic Programming

In this section we explore the connections between payoff-monotonic dynamics and quadratic optimization problems. Consider the following standard quadratic program:[3]

$$\begin{array}{ll} \text{maximize} & \pi(\mathbf{x}) = \mathbf{x}' A \mathbf{x} \\ \text{subject to} & \mathbf{x} \in \Delta, \end{array} \tag{3.1}$$

[3] This terminology is due to Bomze (1998).

where A is an arbitrary $n \times n$ symmetric matrix. In evolutionary game theory, a symmetric payoff matrix arises in the context of doubly symmetric (or partnership) games, where the interests of the two players coincide (Hofbauer & Sigmund, 1998; Weibull, 1995).

A point $\mathbf{x}^* \in \Delta$ is said to be a *global* solution of program 3.1 if $\pi(\mathbf{x}^*) \geq \pi(\mathbf{x})$, for all $\mathbf{x} \in \Delta$. It is said to be a *local* solution if there exists an $\varepsilon > 0$ such that $\pi(\mathbf{x}^*) \geq \pi(\mathbf{x})$ for all $\mathbf{x} \in \Delta$ whose distance from $\mathbf{x}^*$ is less than ε, and if $\pi(\mathbf{x}^*) = \pi(\mathbf{x})$ implies $\mathbf{x}^* = \mathbf{x}$, then $\mathbf{x}^*$ is said to be a *strict* local solution. Note that the solutions of equation 3.1 remain the same if matrix A is replaced with $A + k\mathbf{e}\mathbf{e}'$, where k is an arbitrary constant. In addition, observe that maximizing a nonhomogeneous quadratic form $\mathbf{x}'Q\mathbf{x} + 2\mathbf{c}'\mathbf{x}$ over Δ is equivalent to solving equation 3.1 with $A = Q + \mathbf{e}\mathbf{c}' + \mathbf{c}\mathbf{e}'$ (Bomze, 1998).

A point $\mathbf{x} \in \Delta$ satisfies the Karush-Kuhn-Tucker (KKT) conditions for problem 3.1, that is, the first-order necessary conditions for local optimality, if there exist $n + 1$ real constants $\mu_1, \ldots, \mu_n$ and λ, with $\mu_i \geq 0$ for all $i = 1 \ldots n$, such that:

$$(A\mathbf{x})_i - \lambda + \mu_i = 0$$

for all $i = 1 \ldots n$, and

$$\sum_{i=1}^{n} x_i \mu_i = 0.$$

Note that since both x_i and μ_i are nonnegative for all $i = 1, \ldots, n$, the latter condition is equivalent to saying that $i \in \sigma(\mathbf{x})$ implies $\mu_i = 0$. Hence, the KKT conditions can be rewritten as

$$(A\mathbf{x})_i \begin{cases} = \lambda & \text{if } i \in \sigma(\mathbf{x}) \\ \leq \lambda & \text{if } i \notin \sigma(\mathbf{x}) \end{cases} \tag{3.2}$$

for some real constant λ. On the other hand, it is clear that $\lambda = \mathbf{x}'A\mathbf{x}$. A point $\mathbf{x} \in \Delta$ satisfying equation 3.2 will be called a *KKT point* throughout.

The following easily proved results establish a first connection between standard quadratic programs and payoff-monotonic dynamics.

Proposition 2. *If $\mathbf{x} \in \Delta$ is a KKT point for equation 3.1, then it is a stationary point of any payoff-monotonic dynamics. If $\mathbf{x} \in int(\Delta)$, then the converse also holds.*

Proof. The proof is a straightforward consequence of proposition 1 and the KKT conditions 3.2.

Clearly, not all equilibria of payoff-monotonic dynamics correspond to KKT points of equation 3.1 (e.g., think about the vertices of Δ), but if they are approached by an interior trajectory, then this comes true.

Proposition 3. *Let* $\mathbf{x} = \lim_{t\to\infty} \mathbf{x}(t)$ *be the limit point to a trajectory under any payoff-monotonic dynamics. If* $\mathbf{x}(0) \in int(\Delta)$, *then* $\mathbf{x}$ *is a KKT point for program 3.1.*

Proof. Since $\mathbf{x}$ is a limit point of a trajectory, it is a stationary point (see, e.g., Bhatia & Szegö, 1970), and hence by proposition 1, $\pi_i(\mathbf{x}) = \pi(\mathbf{x})$ for all $i \in \sigma(\mathbf{x})$. Suppose now, to the contrary, that $\pi_i(\mathbf{x}) > \pi(\mathbf{x})$ for some $j \notin \sigma(\mathbf{x})$. Because of payoff monotonicity and stationarity of $\mathbf{x}$, we have $g_j(\mathbf{x}) > g_i(\mathbf{x}) = 0$ for all $i \in \sigma(\mathbf{x})$, and by continuity, there exists a neighborhood U of $\mathbf{x}$ such that $g_j(\mathbf{y}) > 0$ for all $\mathbf{y} \in U$. Then, for a sufficiently large T, $g_j(\mathbf{x}(t)) > 0$ for all $t \geq T$, and since $x_j(t) > 0$ for all t (recall in fact that int(Δ) is invariant), we have $\dot{x}_j(t) > 0$ for $t \geq T$. This implies $x_j = \lim_{t\to\infty} x_j(t) > 0$, a contradiction.

The next proposition, which will be useful later, provides another necessary condition for local solutions of equation 3.1, when the payoff matrix has a particular structure.

Proposition 4. *Let* $\mathbf{x} \in \Delta$ *be a stationary point of any payoff-monotonic dynamics, and suppose that the payoff matrix* A *is symmetric with positive diagonal entries, that is,* $a_{ii} > 0$ *for all* $i = 1, \ldots, n$. *Suppose that there exist* $i, j \in \sigma(\mathbf{x})$ *such that* $a_{ij} = 0$. *For* $0 < \delta \leq x_j$ *let*

$$\mathbf{y}(\delta) = \mathbf{x} + \delta(\mathbf{e}_i - \mathbf{e}_j) \in \Delta.$$

Then $\mathbf{y}(\delta)' A \mathbf{y}(\delta) > \mathbf{x}' A \mathbf{x}$.

Proof. From the symmetry of A, we have:

$$\begin{aligned}
\mathbf{y}(\delta)' A \mathbf{y}(\delta) &= [\mathbf{x} + \delta(\mathbf{e}_i - \mathbf{e}_j)]' A [\mathbf{x} + \delta(\mathbf{e}_i - \mathbf{e}_j)] \\
&= \mathbf{x}' A \mathbf{x} + 2\delta(\mathbf{e}_i - \mathbf{e}_j)' A \mathbf{x} + \delta^2 (\mathbf{e}_i - \mathbf{e}_j)' A (\mathbf{e}_i - \mathbf{e}_j) \\
&= \mathbf{x}' A \mathbf{x} + 2\delta[(A\mathbf{x})_i - (A\mathbf{x})_j] + \delta^2 (a_{ii} - 2a_{ij} + a_{jj}).
\end{aligned}$$

But since $\mathbf{x}$ is a stationary point, we have $(A\mathbf{x})_i = (A\mathbf{x})_j$ (recall in fact that $i, j \in \sigma(\mathbf{x})$), and by the hypothesis $a_{ij} = 0$ we have

$$\mathbf{y}(\delta)' A \mathbf{y}(\delta) = \mathbf{x}' A \mathbf{x} + \delta^2 (a_{ii} + a_{jj}) > \mathbf{x}' A \mathbf{x},$$

which proves the proposition.

In an unpublished paper, Hofbauer (1995) showed that for symmetric payoff matrices, the population mean payoff $\mathbf{x}'A\mathbf{x}$ is strictly increasing along the trajectories of any payoff-monotonic dynamics. This result generalizes the celebrated "fundamental theorem of natural selection" (Hofbauer & Sigmund, 1998; Weibull, 1995), whose original form traces back to R. A. Fisher (1930). Here, we provide a different proof, adapting a technique from Fudenberg and Levine (1998).

Theorem 1. *If the payoff matrix A is symmetric, then $\pi(\mathbf{x}) = \mathbf{x}'A\mathbf{x}$ is strictly increasing along any nonconstant trajectory of any payoff-monotonic dynamics. In other words, $\dot{\pi}(\mathbf{x}(t)) \geq 0$ for all t, with equality if and only if $\mathbf{x} = \mathbf{x}(t)$ is a stationary point.*

Proof. See Hofbauer (1995), or Pelillo (2002) for a different proof.

Apart from the monotonicity result that provides a (strict) Lyapunov function for payoff-monotonic dynamics, the previous theorem also rules out complicated attractors like cycles, invariant tori, or even strange attractors. It also allows us to establish a strong connection between the stability properties of these dynamics and the solutions of equation 3.1. To this end, we need an auxiliary result.

Lemma 1. *Let $\mathbf{x}$ be a strict local solution of equation 3.1, and put $S = \sigma(\mathbf{x})$. Then $\mathbf{x}$ is the only stationary point of any payoff-monotonic dynamics in $int(\Delta_S)$. Moreover, $\mathbf{x}'A\mathbf{x} > \mathbf{y}'A\mathbf{y}$ for all $\mathbf{y} \in \Delta_S$.*

Proof. Clearly, since $\mathbf{x}$ is a strict local solution of equation 3.1, it is a KKT point and hence is stationary under any payoff-monotonic dynamics by proposition 2. Suppose by contradiction that $\mathbf{y} \in \text{int}(\Delta_S) \setminus \{\mathbf{x}\}$ is stationary too. Then it is easy to see that all points on the segment joining $\mathbf{x}$ and $\mathbf{y}$, which is contained in $\text{int}(\Delta_S)$ because of its convexity, consists entirely of stationary points. Hence, by theorem 1, $\dot{\pi} = 0$ on this segment, which means that π is constant, but this contradicts the hypothesis that $\mathbf{x}$ is a strict local solution of equation 3.1.

Moreover, for a sufficiently small $\varepsilon > 0$ we have

$$\begin{aligned}\mathbf{x}'A\mathbf{x} &> [(1-\varepsilon)\mathbf{x} + \varepsilon\mathbf{y}]'A[(1-\varepsilon)\mathbf{x} + \varepsilon\mathbf{y}] \\ &= (1-\varepsilon)^2\mathbf{x}'A\mathbf{x} + 2\varepsilon(1-\varepsilon)\mathbf{y}'A\mathbf{x} + \varepsilon^2\mathbf{y}'A\mathbf{y},\end{aligned}$$

but since $\mathbf{x}$ is stationary and $\sigma(\mathbf{y}) \subseteq \sigma(\mathbf{x})$, $\mathbf{y}'A\mathbf{x} = \mathbf{x}'A\mathbf{x}$, from which we readily obtain $\mathbf{x}'A\mathbf{x} > \mathbf{y}'A\mathbf{y}$.

Theorem 2. *A point $\mathbf{x} \in \Delta$ is a strict local solution of program 3.1 if and only if it is asymptotically stable under any payoff-monotonic dynamics.*

Proof. If $\mathbf{x}$ is asymptotically stable, then there exists a neighborhood U of $\mathbf{x}$ in Δ such that every trajectory starting in a point $\mathbf{y} \in U$ will converge to $\mathbf{x}$. Then, by virtue of Theorem 1, we have $\mathbf{y}' A\mathbf{y} > \mathbf{x}' A\mathbf{x}$ for all $\mathbf{y} \in U \backslash \{\mathbf{x}\}$, which shows that $\mathbf{x}$ is a strict local solution for equation 3.1.

On the other hand, suppose that $\mathbf{x}$ is a strict local solution of equation 3.1, and let $S = \sigma(\mathbf{x})$. By lemma 1, the function $V : \text{int}(\Delta_S) \to \mathbb{R}$ defined as $V(\mathbf{y}) = \pi(\mathbf{x}) - \pi(\mathbf{y})$ is clearly nonnegative in $\text{int}(\Delta_S)$, and it vanishes only when $\mathbf{y} = \mathbf{x}$. Furthermore, $\dot{V} \leq 0$ by theorem 1 and, again from lemma 1, $\dot{V} < 0$ in $\text{int}(\Delta_S) \backslash \{\mathbf{x}\}$, as $\mathbf{x}$ is the only stationary point in $\text{int}(\Delta_S)$. This means that V is a strict Lyapunov function for any payoff-monotonic dynamics, and hence $\mathbf{x}$ is asymptotically stable (see, e.g., Bhatia & Szegö, 1970; Hirsch & Smale, 1974).

The results presented in this section show that continuous-time payoff-monotonic dynamics can be usefully employed to find (local) solutions of standard quadratic programs. In the rest of the article, we focus the discussion on a particular class of quadratic optimization problems that arise in conjunction with the maximum clique problem.

4 A Family of Quadratic Programs for Maximum Clique

Let $G = (V, E)$ be an undirected graph with no self-loops, where $V = \{1, \ldots, n\}$ is the set of vertices and $E \subseteq V \times V$ is the set of edges. The order of G is the number of its vertices, and its size is the number of edges. Two vertices $i, j \in V$ are said to be adjacent if $(i, j) \in E$. The adjacency matrix of G is the $n \times n$ symmetric matrix $A_G = (a_{ij})$ defined as follows:

$$a_{ij} = \begin{cases} 1, & \text{if } (i, j) \in E, \\ 0, & \text{otherwise.} \end{cases}$$

The degree of a vertex $i \in V$ relative to a subset of vertices C, denoted by $\deg_C(i)$, is the number of vertices in C adjacent to it, that is,

$$\deg_C(i) = \sum_{j \in C} a_{ij}.$$

Clearly, when $C = V$ we obtain the standard degree notion, in which case we shall write $\deg(i)$ instead of $\deg_V(i)$.

A subset C of vertices in G is called a *clique* if all its vertices are mutually adjacent; that is, for all $i, j \in C$, with $i \neq j$, we have $(i, j) \in E$. A clique is said to be *maximal* if it is not contained in any larger clique and *maximum* if it is the largest clique in the graph. The *clique number*, denoted by $\omega(G)$, is defined as the cardinality of the maximum clique. The maximum clique problem is to find a clique whose cardinality equals the clique number.

In the mid-1960s, Motzkin and Straus (1965) established a remarkable connection between the maximum clique problem and the following standard quadratic program:

$$\begin{array}{ll} \text{maximize} & f(\mathbf{x}) = \mathbf{x}' A_G \mathbf{x} \\ \text{subject to} & \mathbf{x} \in \Delta \subset \mathbb{R}^n, \end{array} \tag{4.1}$$

where n is the order of G. Specifically, if $\mathbf{x}^*$ is a global solution of equation 4.1, they proved that the clique number of G is related to $f(\mathbf{x}^*)$ by the following formula:

$$\omega(G) = \frac{1}{1 - f(\mathbf{x}^*)}. \tag{4.2}$$

Additionally, they showed that a subset of vertices C is a maximum clique of G if and only if its characteristic vector $\mathbf{x}^C$, which is the vector of Δ defined as

$$x_i^C = \begin{cases} 1/|C|, & \text{if } i \in C \\ 0, & \text{otherwise,} \end{cases}$$

is a global maximizer of f on Δ.[4] Gibbons, Hearn, Pardalos, and Ramana (1997), and Pelillo and Jagota (1995), extended the Motzkin-Straus theorem by providing a characterization of maximal cliques in terms of local maximizers of f on Δ.

One drawback associated with the original Motzkin-Straus formulation, however, relates to the existence of "infeasible" solutions, that is, maximizers of f that are not in the form of characteristic vectors. Pelillo and Jagota (1995) have provided general characterizations of such solutions. To overcome this problem, consider the following family of standard quadratic programs:

$$\begin{array}{ll} \text{maximize} & f_\alpha(\mathbf{x}) = \mathbf{x}'(A_G + \alpha I)\mathbf{x} \\ \text{subject to} & \mathbf{x} \in \Delta \subset \mathbb{R}^n, \end{array} \tag{4.3}$$

where α is an arbitrary real parameter and I is the identity matrix, which includes as special cases the original Motzkin-Straus program (see equation 4.1) and the regularized version proposed by Bomze (1997) (corresponding to the cases $\alpha = 0$ and $\alpha = \frac{1}{2}$, respectively).

[4] In their original paper, Motzkin and Straus proved just the "only-if" part of this theorem. The converse direction is, however, a straightforward consequence of their result (Pelillo & Jagota, 1995).

Proposition 5. *Let* $\mathbf{x}$ *be a KKT point for program 4.3 with* $\alpha < 1$, *and let* $C = \sigma(\mathbf{x})$ *be the support of* $\mathbf{x}$. *If* C *is a clique of* G, *then it is a maximal clique.*

Proof. Suppose by contradiction that C is a nonmaximal clique. Hence, there exists $j \in V \backslash C$ such that $(i, j) \in E$ for all $i \in C$. Since $\alpha < 1$, we have for all $i \in C$:

$$(A_G\mathbf{x} + \alpha\mathbf{x})_i = (A_G\mathbf{x})_i + \alpha x_i = 1 - (1-\alpha)x_i < 1 = (A_G\mathbf{x})_j = (A_G\mathbf{x} + \alpha\mathbf{x})_j.$$

But due to equation 3.2, this contradicts the hypothesis that $\mathbf{x}$ is a KKT point for equation 4.3.

In general, however, the fact that a point $\mathbf{x} \in \Delta$ satisfies the KKT conditions does not imply that $\sigma(\mathbf{x})$ is a clique of G. For instance, it is easy to show that if for a subset C we have $\deg_C(i) = k$ for all $i \in C$ (i.e., C induces a k-regular subgraph), and $\deg_C(i) \leq k$ for all $i \notin C$, then $\mathbf{x}^C$ is a KKT point for equation 4.3 provided that $\alpha \geq 0$.

The following theorem, which generalizes an earlier result by Bomze (1997), establishes a one-to-one correspondence between local-global solutions of equation 4.3 and maximal-maximum cliques of G. By adapting the proof technique from Bomze (1997), it has also been proved previously in Bomze et al. (2002) using concepts and results from evolutionary game theory. Here we provide a different proof based on standard facts from optimization theory.

Theorem 3. *Let* C *be a subset of vertices of a graph* G, *and let* $\mathbf{x}^C$ *be its characteristic vector. Then, for any* $0 < \alpha < 1$, C *is a maximal (maximum) clique of* G *if and only if* $\mathbf{x}^C$ *is a local (global) solution of equation 4.3. Moreover, all solutions of the equation are strict and are characteristic vectors of maximal cliques of* G.

Proof. See Bomze et al. (2002) for a proof that requires several previous results from evolutionary game theory or appendix B for a self-contained proof which uses only basic concepts from optimization theory.

Corollary 1. *Let* C *be a subset of vertices of a graph* G *with* $\mathbf{x}^C$ *as its characteristic vector, and let* $0 < \alpha < 1$. *Then* C *is a maximal clique of* G *if and only if* $\mathbf{x}^C$ *is an asymptotically stable stationary point under any payoff-monotonic dynamics with payoff matrix* $A = A_G + \alpha I$.

Proof. The proof is obvious from theorems 2 and 3.

These results naturally suggest any dynamics in the payoff-monotonic class as a useful heuristic for the maximum clique problem, and this will be the subject of the next section.

5 Clique-Finding Payoff-Monotonic Dynamics

Let $G = (V, E)$ be a graph, and let A_G denote its adjacency matrix. By using

$$A = A_G + \alpha I \quad (0 < \alpha < 1) \tag{5.1}$$

as the payoff matrix, any payoff-monotonic dynamics, starting from an arbitrary initial state, will eventually be attracted with probability one by the nearest asymptotically stable point, which, by virtue of corollary 1, will then correspond to a maximal clique of G. Clearly, in theory, there is no guarantee that the converged solution will be a global solution of equation 4.3 and therefore that it will yield a maximum clique in G.

In practice, it is not unlikely, however, that the system converges toward a stationary point that is unstable, that is, a saddle of the Lyapunov function $\mathbf{x}'A\mathbf{x}$. This can be the case when the dynamics is started from the simplex barycenter and symmetry is not broken. Proposition 3 ensures, however, that the limit point of any interior trajectory will be at least a KKT point of program 4.3. The next proposition translates this fact in a different language.

Proposition 6. *Let $\mathbf{x} \in \Delta$ be the limit point of a trajectory of any payoff-monotonic dynamics starting in the interior of Δ. Then either $\sigma(\mathbf{x})$ is a maximal clique or it is not a clique.*

Proof. The proof is obvious from propositions 3 and 5.

The practical significance of the previous result reveals itself in large graphs: even if these are quite dense, cliques are usually much smaller than the graph itself. Now suppose we are returned a KKT point $\mathbf{x}$. Then we put $C = \sigma(\mathbf{x})$ and check whether C is a clique. This requires $\mathcal{O}(m^2)$ steps if C contains m vertices, while checking whether this clique is maximal would require $\mathcal{O}(mn)$ steps and, as stressed above, usually $m \ll n$. But proposition 6 now guarantees that the obtained clique C (if it is one) must automatically be maximal, and thus we are spared trying to add external vertices.

5.1 Experimental Results. To assess the ability of our payoff-monotonic models to extract large cliques, we performed extensive experimental evaluations on both random and DIMACS benchmark graphs.[5] For our simulations we used discretized versions of the continuous-time linear (see equation 2.6) and exponential (see equation 2.7) replicator dynamics (see appendix A for a description of our discretizations). We started the processes from the simplex barycenter and stopped them when a maximal clique (i.e., a strict local maximizer of f) was found. Occasionally, when

[5] Data can be found online at http://dimacs.rutgers.edu.

the system converged to a nonclique KKT point, we randomly perturbed the solution and let the game dynamics start from this new point. Since unstable stationary points have a basin of attraction of measure zero around them, the process is pulled away from it with probability one, to converge, eventually, to another (hopefully asymptotically stable) stationary point.

In an attempt to improve the quality of the final results, we used a mixed strategy as far as the regularization parameter α is concerned. Indeed, Bomze et al. (1997) showed that the original Motzkin-Straus formulation (i.e., $\alpha = 0$), which is plagued with the presence of spurious solutions, usually yields slightly better results than its regularized version. Accordingly, we started the dynamics using $\alpha = 0$ and, after convergence, we restarted it from the converged point using $\alpha = \frac{1}{2}$. This way, we are guaranteed to avoid spurious solutions, thereby obtaining a maximal clique.

In the first set of experiments we ran the algorithms over random graphs of order 100, 200, 300, 400, 500, and 1000 and with edge densities ranging from 0.25 to 0.95. For each order and density value, 100 different graphs were generated. Table 1 shows the results obtained in terms of clique size. Here, n refers to the graph order, ρ is the edge density, and the labels "RD linear" and "RD exp" indicate the first-order and the exponential replicator dynamics, respectively. The results are compared with the following state-of-the-art neural network heuristics for maximum clique: Jagota's continuous Hopfield dynamics (CHD) and mean field annealing (MFA) (Jagota, 1995; Jagota et al., 1996), the saturated linear dynamical network (SLDN) by Pekergin et al. (1999), an approximation approach introduced by Funabiki et al. (FTL) (1992), the iterative Hopfield nets (IHN) algorithm by Bertoni et al. (2002), and the Hopfield network learning (HNL) of Wang et al. (2003). Figure 1 plots the corresponding CPU timings obtained with a (nonoptimized) C++ implementation on a machine equipped with a 2.5 GHz Pentium 4 processor.

Since random graphs are notoriously easy to deal with, a second set of experiments was also performed on the DIMACS benchmark graphs (see Tables 2 and 3). Here, columns marked with n and ρ contain the number of vertices in the graph and the edge density respectively. Columns "Clique Size" contain the size of the cliques found by the competing algorithms, while the column "Time" reports the CPU timings for the proposed dynamics. The sizes of the cliques obtained are compared against several algorithms present in either the neural network or the continuous optimization literature, and against the best result over all algorithm featured on the DIMACS challenge (Johnson & Trick, 1996) (DIMACS best). The neural-based approaches include mean field annealing (MFA), the inverted neurons network (INN) model by Grossman (1996), and the IHN algorithm, while algorithms from the continuous optimization literature include the continuous-based heuristic (CBH) by Gibbons et al. (1997) and the QSH algorithm by Busygin, Butenko, and Pardalos (2002). The results are taken from the cited papers. No results are presented for CHD, SLDN, FTL, and

Table 1: Results of Replicator Dynamics (RD) and State-of-the-Art Neural Network–Based or Optimization-Based Approaches on Random Graphs with Varying Order and Density.

n	ρ	RD Linear	RD Exponential	CHD	MFA	SLDN	FTL	IHN	HNL
100	0.25	4.90 ± 0.56	4.81 ± 0.60	4.48	—	4.83	4.2	—	—
	0.50	8.01 ± 0.66	8.07 ± 0.64	7.38	8.50	8.07	8.0	9.13	9
	0.75	15.10 ± 1.05	15.15 ± 1.11	13.87	—	15.05	14.1	—	—
	0.90	28.50 ± 1.87	28.92 ± 1.74	27.92	30.02	—	—	—	30
	0.95	41.81 ± 1.80	42.04 ± 1.78	—	—	—	—	—	—
200	0.25	5.34 ± 0.68	5.35 ± 0.64	—	—	—	4.9	—	—
	0.50	9.04 ± 0.76	9.11 ± 0.77	—	—	—	8.5	10.60	11
	0.75	17.77 ± 1.35	18.05 ± 1.23	—	—	—	—	—	—
	0.90	36.74 ± 1.65	37.41 ± 1.55	—	—	—	—	—	39
	0.95	57.33 ± 2.33	58.24 ± 2.09	—	—	—	—	—	—
300	0.25	5.58 ± 0.62	5.61 ± 0.62	—	—	—	5.1	—	—
	0.50	9.54 ± 0.90	9.57 ± 0.88	—	—	—	8.9	11.60	11
	0.75	19.05 ± 1.27	19.40 ± 1.17	—	—	—	—	—	—
	0.90	40.74 ± 1.97	41.48 ± 1.81	—	—	—	—	—	46
	0.95	66.48 ± 2.69	67.43 ± 2.47	—	—	—	—	—	—
400	0.25	5.73 ± 0.65	5.73 ± 0.60	5.53	—	5.70	4.9	—	—
	0.50	9.99 ± 0.80	10.07 ± 0.77	9.24	10.36	9.91	8.9	12.30	—
	0.75	20.17 ± 1.16	20.42 ± 1.10	18.79	—	20.44	17.7	—	—
	0.90	43.63 ± 1.73	44.44 ± 1.64	43.24	49.94	—	—	—	—
	0.95	73.11 ± 2.37	74.25 ± 2.30	—	—	—	—	—	—
500	0.25	5.81 ± 0.69	5.74 ± 2.30	—	—	—	6.2	—	—
	0.50	10.14 ± 0.93	10.31 ± 0.91	—	—	—	9.4	12.80	12
	0.75	20.90 ± 1.32	21.31 ± 1.27	—	—	—	—	—	—
	0.90	46.10 ± 2.25	46.93 ± 2.29	—	—	—	—	—	56
	0.95	78.73 ± 2.66	79.94 ± 2.68	—	—	—	—	—	—
1000	0.25	6.23 ± 0.62	6.17 ± 0.57	6.03	—	6.17	5.8	—	—
	0.50	10.74 ± 0.80	10.83 ± 0.82	10.25	—	10.93	10.4	—	—
	0.75	22.88 ± 1.28	23.04 ± 1.35	21.26	—	23.19	21.4	—	—
	0.90	52.60 ± 2.02	53.15 ± 2.11	—	—	—	—	—	—
	0.95	93.75 ± 3.12	94.80 ± 2.96	—	—	—	—	—	—

HNL since the authors did not provide results on the DIMACS graphs. For the same reason, we did not report results on random graphs for INN, CBH, and QSH.

A number of conclusions can be drawn from these results. First, the exponential dynamics provides slightly better results than the linear one, being, however, dramatically faster, especially on dense graphs. These results confirm earlier findings reported in Pelillo (1999, 2002) on graph classes arising from graph and tree matching problems. As for the comparison with the other algorithms, we note that our dynamics substantially outperform CHD and FTL and are, overall, as effective as SLDN (for which no results on DIMACS graphs are available). Observe that these approaches do not incorporate any procedure to escape from poor local solutions and, hence,

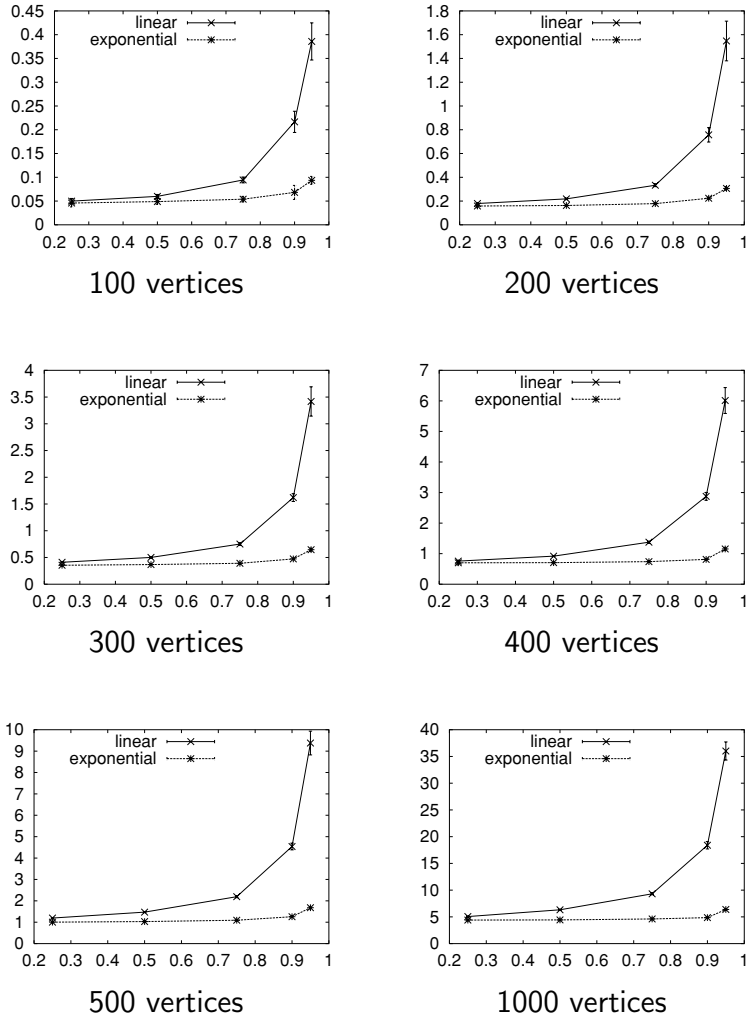

Figure 1: CPU time of replicator dynamics on random graphs with varying order and density. The x-axis represents the edge density, while the y-axis denotes time (in seconds).

are close in spirit to ours. Clearly our results are worse than those obtained with algorithms that do use some form of annealing or, in any case, are explicitly designed to avoid local optima, such as IHN, HNL, CBH, and QSH. Interestingly, however, the results are close to (and in some instances

Table 2: Results of Replicator Dynamics (RD) and State-of-the-Art Neural Network–Based or Optimization-Based Approaches on DIMACS Benchmark Graphs, Part I.

			Clique Size								*Time (secs)*	
Graph	n	ρ	DIMACS Best	RD Linear	RD Exponential	MFA	INN Average	IHN	CBH	QSH	RD Linear	RD Exponential
brock200_1	200	0.75	21	17	17	19	—	—	20	21	0.33	0.2
brock200_2	200	0.50	12	8	8	9	9.1	—	12	12	0.23	0.17
brock200_3	200	0.61	15	9	10	11	—	—	14	15	0.25	0.18
brock200_4	200	0.66	17	13	13	14	13.4	—	16	17	0.28	0.16
brock400_1	400	0.75	24	21	21	24	—	—	23	27	1.36	0.69
brock400_2	400	0.75	25	20	21	21	21.3	—	24	29	1.35	0.73
brock400_3	400	0.75	25	19	18	22	—	—	23	31	1.38	0.69
brock400_4	400	0.75	33	19	20	23	21.3	—	24	33	1.37	0.74
brock800_1	800	0.65	21	16	16	16	—	—	20	17	4.68	2.86
brock800_2	800	0.65	21	16	16	16	16.9	—	19	24	4.61	2.77
brock800_3	800	0.65	21	15	18	16	—	—	20	25	4.72	2.92
brock800_4	800	0.65	21	17	17	15	16.5	—	19	26	4.63	2.7
c-fat200_1	200	0.08	12	12	12	6	—	12	12	12	0.22	0.17
c-fat200_2	200	0.16	24	24	24	24	—	24	24	24	0.15	0.16
c-fat200_5	200	0.43	58	58	58	58	—	58	58	58	0.16	0.16
c-fat500-1	500	0.04	14	14	14	—	—	14	14	14	0.97	1.01
c-fat500-2	500	0.07	26	26	26	—	—	26	26	26	1	1.02
c-fat500-5	500	0.19	64	64	64	—	—	64	64	64	1.01	0.99
c-fat500-10	500	0.37	126	126	126	—	—	—	126	126	1.09	1
hamming6-2	64	0.90	32	32	32	—	—	32	32	32	0.09	0.03
hamming6-4	64	0.35	4	4	4	—	—	4	4	4	0.03	0.02

Table 2: Continued

			Clique Size								*Time (secs)*	
Graph	n	ρ	DIMACS Best	RD Linear	RD Exponential	MFA	INN Average	IHN	CBH	QSH	RD Linear	RD Exponential
hamming8-2	256	0.97	128	82	81	—	—	128	128	128	3.4	0.51
hamming8-4	256	0.64	16	10	10	—	16.0	16	16	16	0.44	0.27
hamming10-2	65536	0.99	512	312	297	—	—	512	—	—	459.04	34.48
hamming10-4	65536	0.83	36	32	32	—	32.0	36	—	—	24.27	6.05
johnson8-2-4	28	0.56	4	4	4	—	—	4	4	4	0.01	0.00
johnson8-4-4	70	0.77	14	14	14	—	—	14	14	14	0.04	0.02
johnson16-2-4	120	0.76	8	8	8	—	—	8	8	8	0.12	0.07
johnson32-2-4	496	0.88	16	16	16	—	—	16	16	16	5.11	1.36
keller4	171	0.65	11	7	7	—	8.4	—	10	11	0.21	0.14
keller5	776	0.75	27	15	15	—	16.7	—	21	24	5.43	2.64
keller6	3361	0.82	59	31	31	—	33.4	—	—	—	174.89	54.34

Table 3: Results of Replicator Dynamics (RD) and State-of-the-Art Neural Network–Based or Optimization-Based Approaches on DIMACS Benchmark Graphs, Part II.

			Clique Size								*Time (secs)*	
Graph	n	ρ	DIMACS Best	RD Linear	RD Exponential	MFA	INN Average	IHN	CBH	QSH	RD Linear	RD Exponential
p_hat300-1	300	0.24	8	6	6	—	7.0	8	8	7	0.41	0.39
p_hat300-2	300	0.49	25	24	24	—	21.9	25	25	24	0.69	0.43
p_hat300-3	300	0.74	36	32	33	—	33.1	36	36	33	1.04	0.43
p_hat500-1	500	0.25	9	8	8	—	—	9	9	9	1.18	1.07
p_hat500-2	500	0.50	36	34	34	—	—	36	35	33	2.07	1.14
p_hat500-3	500	0.75	50	47	48	—	—	49	49	46	3.75	1.23
p_hat700-1	700	0.25	11	9	9	—	8.4	11	11	8	2.39	2.1
p_hat700-2	700	0.50	44	43	43	—	42.0	44	44	42	4.31	2.2
p_hat700-3	700	0.75	62	58	59	—	58.2	61	60	59	7.35	2.43
p_hat1000-1	1000	0.25	10	8	8	—	—	10	—	—	4.89	4.5
p_hat1000-2	1000	0.50	46	42	44	—	—	46	—	—	8.02	4.36
p_hat1000-3	1000	0.75	66	61	63	—	—	68	—	—	14.07	4.8
p_hat1500-1	1500	0.25	11	9	9	—	9.4	—	—	—	11.03	9.71
p_hat1500-2	1500	0.50	65	61	61	—	60.2	—	—	—	22.96	10.37
p_hat1500-3	1500	0.75	94	88	88	—	86.2	—	—	—	36.72	10.89
san200_0.7_1	200	0.70	30	15	15	—	—	30	15	30	0.68	0.22
san200_0.7_2	200	0.70	18	12	12	—	—	15	12	18	0.96	0.23
san200_0.9_1	200	0.90	70	45	45	42	—	70	46	70	1.74	0.35
san200_0.9_2	200	0.90	60	36	36	—	—	41	36	60	1.11	0.3
san200_0.9_3	200	0.90	44	32	32	33	—	—	30	35	0.76	0.26

Table 3: Continued

			Clique Size								*Time (secs)*	
Graph	n	ρ	DIMACS Best	RD Linear	RD Exponential	MFA	INN Average	IHN	CBH	QSH	RD Linear	RD Exponential
san400_0.5_1	400	0.50	13	7	7	—	—	—	8	9	2.71	0.87
san400_0.7_1	400	0.70	40	20	20	—	—	40	20	40	2.41	0.95
san400_0.7_2	400	0.70	30	15	15	—	—	30	15	30	3.38	0.97
san400_0.7_3	400	0.70	22	12	12	—	—	—	14	16	4	0.89
san400_0.9_1	400	0.90	100	44	50	—	—	100	50	100	2.98	0.89
san1000	1000	0.50	10	8	8	—	—	10	—	—	29.95	6.91
sanr200_0.7	200	0.70	18	14	16	18	—	17	18	15	0.29	0.2
sanr200_0.9	200	0.90	42	37	39	41	—	41	41	37	0.72	0.24
sanr400_0.5	400	0.50	13	11	11	—	—	12	12	11	0.88	0.61
sanr400_0.7	400	0.70	21	18	18	—	—	21	20	18	1.24	0.69

even better than) those obtained with Jagota's MFA and Grossman's INN, which are also in this family.

6 Annealed Imitation Dynamics: Evolving Toward Larger Cliques

In an attempt to avoid inefficient local solutions, we now follow Bomze et al. (2002) and investigate the stability properties of equilibria of payoff-monotonic dynamics when the parameter α is allowed to take on negative values. Indeed, we shall restrict our analysis to imitation dynamics (see equation 2.5), but we first make a few observations pertaining to general selection dynamics.

For any regular selection dynamics $\dot{x}_i = x_i g_i(\mathbf{x})$, we have:

$$\frac{\partial \dot{x}_i}{\partial x_j} = \delta_{ij} g_i(\mathbf{x}) + x_i \frac{\partial g_i(\mathbf{x})}{\partial x_j}, \tag{6.1}$$

where δ_{ij} is the Kronecker delta, defined as $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise. Assuming without loss of generality that $\sigma(\mathbf{x}) = \{1, \ldots, m\}$, the Jacobian of any regular selection dynamics at a stationary point $\mathbf{x}$ has therefore the following block triangular form,

$$J(\mathbf{x}) = \begin{bmatrix} M(\mathbf{x}) & N(\mathbf{x}) \\ O & D(\mathbf{x}) \end{bmatrix}, \tag{6.2}$$

where the entries of $M(\mathbf{x})$ and $N(\mathbf{x})$ are given by $x_i \frac{\partial g_i(\mathbf{x})}{\partial x_j}$, O is the (possibly empty) matrix containing all zeros, and

$$D(\mathbf{x}) = \operatorname{diag}\{g_{m+1}(\mathbf{x}), \ldots, g_n(\mathbf{x})\}.$$

An immediate consequence of this observation is that we can already say something about the spectrum of $J(\mathbf{x})$, when $m < n$. In fact, the eigenvalues of $J(\mathbf{x})$ are those of $M(\mathbf{x})$ together with those of $D(\mathbf{x})$, and since $D(\mathbf{x})$ is diagonal, its eigenvalues coincide with its diagonal entries, that is, $g_{m+1}(\mathbf{x}), \ldots, g_n(\mathbf{x})$. This set of eigenvalues governs the asymptotic behavior of the external flow under the system obtained by linearization around $\mathbf{x}$ and is usually called transversal eigenvalues (Hofbauer & Sigmund, 1998).

Without knowing the form of the growth functions g_i, however, it is difficult to provide further insights into the spectral properties of $J(\mathbf{x})$, and therefore we now specialize our discussion to imitation dynamics (see equation 2.5). In this case, we have

$$g_i(\mathbf{x}) = \phi(\pi_i(\mathbf{x})) - \sum_{k=1}^{n} x_k \phi(\pi_k(\mathbf{x})), \tag{6.3}$$

where ϕ is a strictly increasing function, and hence

$$\frac{\partial \dot{x}_i}{\partial x_j} = \delta_{ij}\left[\phi(\pi_i(\mathbf{x})) - \sum_k x_k \phi(\pi_k(\mathbf{x}))\right] + x_i\left[a_{ij}\phi'(\pi_i(\mathbf{x})) - \phi(\pi_j(\mathbf{x})) - \sum_k x_k a_{kj}\phi'(\pi_k(\mathbf{x}))\right]. \tag{6.4}$$

When $\mathbf{x}$ is an equilibrium point, and hence $\pi_i(\mathbf{x}) = \pi(\mathbf{x})$ for all $i \in \sigma(\mathbf{x})$, the previous expression simplifies to

$$\frac{\partial \dot{x}_i}{\partial x_j} = \delta_{ij}[\phi(\pi_i(\mathbf{x})) - \phi(\pi(\mathbf{x}))] + x_i[a_{ij}\phi'(\pi(\mathbf{x})) - \phi(\pi_j(\mathbf{x})) - \phi'(\pi(\mathbf{x}))\pi_j(\mathbf{x})]. \tag{6.5}$$

Before we provide the main result of this section, we prove the following useful proposition, which generalizes an earlier result by Bomze (1986).

Proposition 7. *Let* $\mathbf{x}$ *be a stationary point of any imitation dynamics, equation 2.5. Then:*

a. $-\phi(\pi(\mathbf{x}))$ *is an eigenvalue of* $J(\mathbf{x})$*, with* $\mathbf{x}$ *as an associated eigenvector.*

b. *If* $\mathbf{y}$ *is an eigenvector of* $J(\mathbf{x})$ *associated with an eigenvalue* $\lambda \neq -\phi(\pi(\mathbf{x}))$*, then* $\mathbf{e}'\mathbf{y} = \sum_i y_i = 0$*.*

Proof. Recall from proposition 1 that $\mathbf{x}$ is an equilibrium point for equation 2.5 if and only if $\pi_i(\mathbf{x}) = \pi(\mathbf{x})$ for all $i \in \sigma(\mathbf{x})$. Hence, for $i = 1, \ldots, n$, we have:

$$\begin{aligned}\sum_{j=1}^{n} x_j \frac{\partial \dot{x}_i}{\partial x_j} &= x_i[\phi(\pi_i(\mathbf{x})) - \phi(\pi(\mathbf{x}))] \\ &\quad + x_i \sum_{j=1}^{n} x_j[a_{ij}\phi'(\pi(\mathbf{x})) - \phi(\pi_j(\mathbf{x})) - \pi_j(\mathbf{x})\phi'(\pi(\mathbf{x}))] \\ &= x_i[\phi(\pi_i(\mathbf{x})) - \phi(\pi(\mathbf{x})) + \phi'(\pi(\mathbf{x}))\pi_i(\mathbf{x}) - \phi(\pi(\mathbf{x})) - \phi'(\pi(\mathbf{x}))\pi(\mathbf{x})] \\ &= -x_i\phi(\pi(\mathbf{x})).\end{aligned}$$

In other words, we have shown that $J(\mathbf{x})\mathbf{x} = -\phi(\pi(\mathbf{x}))\mathbf{x}$, which proves part a of the proposition.

To prove part b, first note that the columns of $J(\mathbf{x})$ have a nice property: they all sum up to $-\phi(\pi(\mathbf{x}))$. Indeed, for all $j = 1, \ldots, n$ we have:

$$\begin{aligned}\sum_{i=1}^{n} \frac{\partial \dot{x}_i}{\partial x_j} &= \phi(\pi_j(\mathbf{x})) - \phi(\pi(\mathbf{x})) \\ &\quad + \sum_{i=1}^{n} x_i \left[a_{ij}\phi'(\pi(\mathbf{x})) - \phi(\pi_j(\mathbf{x})) - \pi_j(\mathbf{x})\phi'(\pi(\mathbf{x}))\right] \\ &= \phi(\pi_j(\mathbf{x})) - \phi(\pi(\mathbf{x})) + \phi'(\pi(\mathbf{x}))\pi_j(\mathbf{x}) - \phi(\pi_j(\mathbf{x})) - \phi'(\pi(\mathbf{x}))\pi_j(\mathbf{x}) \\ &= -\phi(\pi(\mathbf{x})).\end{aligned}$$

Now, the hypothesis $J(\mathbf{x})\mathbf{y} = \lambda\mathbf{y}$ yields

$$\lambda \sum_i y_i = \sum_i (J(\mathbf{x})\mathbf{y})_i = \sum_j y_j \sum_i J(\mathbf{x})_{ij} = -\phi(\pi(\mathbf{x})) \sum_j y_j,$$

which implies $\sum_i y_i = 0$, since $\lambda \neq -\phi(\pi(\mathbf{x}))$.

Since we analyze the behavior of imitation dynamics restricted to the standard simplex Δ, we are interested only in the eigenvalues of $J(\mathbf{x})$ associated with eigenvectors belonging to the tangent space $\mathbf{e}^{\perp} = \{\mathbf{y} \in \mathbb{R}^n : \mathbf{e}'\mathbf{y} = 0\}$. The previous result therefore implies that the eigenvalue $-\phi(\pi(\mathbf{x}))$ can be neglected in our analysis, and that the remaining ones, including the transversal eigenvalues, are indeed all relevant.

We now return to the maximum clique problem. Let a graph $G = (V, E)$ be given, and for a subset of vertices C, let

$$\gamma(C) = \max_{i \notin C} \deg_C(i) - |C| + 1 \,. \tag{6.6}$$

Note that if C is a maximal clique, then $\gamma(C) \leq 0$. The next theorem shows that $\gamma(C)$ plays a key role in determining the stability of equilibria of imitation dynamics.

Theorem 4. *Let C be a maximal clique of graph $G = (V, E)$, and let $\mathbf{x}^C$ be its characteristic vector. If $\gamma(C) < \alpha < 1$, then $\mathbf{x}^C$ is an asymptotically stable stationary point under any imitation dynamics (see equation 2.5) with payoff matrix $A = A_G + \alpha I$, and hence a (strict) local maximizer of f_α in Δ. Moreover, assuming $C \neq V$, if $\alpha < \gamma(C)$, then $\mathbf{x}^C$ becomes unstable.*

Proof. Assume without loss of generality that $C = \{1, \ldots, m\}$ and suppose that $\gamma(C) < \alpha < 1$. To simplify notations, put $\mathbf{x} = \mathbf{x}^C$. We shall see that the eigenvalues of $J(\mathbf{x})$ are real and negative. This implies that $\mathbf{x}$ is a *sink* and

hence an asymptotically stable point (Hirsch & Smale, 1974). The fact that $\mathbf{x}$ is a strict local maximizer of f_α in Δ follows directly from theorem 2.

As already noticed in the previous discussion, because of its block diagonal form, the eigenvalues of $J(\mathbf{x})$ are those of $M(\mathbf{x}) = \left(x_i \frac{\partial g_i(\mathbf{x})}{\partial x_j}\right)_{i,j=1,\ldots,m}$ together with the $n - m$ transversal eigenvalues $g_{m+1}(\mathbf{x}), \ldots, g_n(\mathbf{x})$, where:

$$g_i(\mathbf{x}) = \phi(\pi_i(\mathbf{x})) - \sum_{k=1}^{n} x_k \phi(\pi_k(\mathbf{x})).$$

Since C is a (maximal) clique, $\pi_k(\mathbf{x}) = \pi(\mathbf{x})$ for all $k \in C = \sigma(\mathbf{x})$, and therefore

$$g_i(\mathbf{x}) = \phi(\pi_i(\mathbf{x})) - \phi(\pi(\mathbf{x})).$$

But ϕ is a strictly increasing function, and hence $g_i(\mathbf{x}) < 0$ if and only if $\pi_i(\mathbf{x}) < \pi(\mathbf{x})$. Now, since C is a maximal clique, $\pi_i(\mathbf{x}) = (A_G\mathbf{x})_i = \deg_C(i)/m$ for all $i > m$, and $\pi(\mathbf{x}) = (m - 1 + \alpha)/m$. But for all $i > m$, we have $\deg_C(i) - m + 1 \leq \gamma(C) < \alpha$, and this yields $\pi_i(\mathbf{x}) < \pi(\mathbf{x})$. Hence, all transversal eigenvalues are negative.

It remains to show that the eigenvalues of $M(\mathbf{x})$ are negative too. When $A = A_G + \alpha I$, we have:

$$M(\mathbf{x})_{ij} = x_i \frac{\partial \dot{x}_i}{\partial x_j} = \frac{1}{m}[(a_{ij} + \alpha\delta_{ij})\phi'(\pi(\mathbf{x})) - \phi(\pi(\mathbf{x})) - \phi'(\pi(\mathbf{x}))\pi(\mathbf{x})].$$

Hence, in matrix form, we have

$$M(\mathbf{x}) = \frac{\phi'(\pi(\mathbf{x}))}{m}\left[\left(1 - \frac{\phi(\pi(\mathbf{x}))}{\phi'(\pi(\mathbf{x}))} - \pi(\mathbf{x})\right)\mathbf{e}\mathbf{e}' + (\alpha - 1)I\right],$$

where $\mathbf{e}\mathbf{e}'$ is the $m \times m$ matrix containing all ones, and the eigenvalues of $M(\mathbf{x})$ are

$$\lambda_1 = \frac{\phi'(\pi(\mathbf{x}))}{m}(\alpha - 1)$$

with multiplicity $m - 1$, and

$$\lambda_2 = \frac{\phi'(\pi(\mathbf{x}))}{m}\left[\left(1 - \frac{\phi(\pi(\mathbf{x}))}{\phi'(\pi(\mathbf{x}))} - \pi(\mathbf{x})\right)m + \alpha - 1\right] = -\phi(\pi(\mathbf{x}))$$

with multiplicity 1. Since $\alpha < 1$ and ϕ is strictly increasing, we have $\lambda_1 < 0$. Moreover, recall from proposition 7 that eigenvalue $\lambda_2 = -\phi(\pi(\mathbf{x}))$ is not

relevant to the imitation dynamics on the simplex Δ, since its eigenvector $\mathbf{x}$ does not belong to the tangent space $\mathbf{e}^{\perp}$. Hence, as far as the dynamics in the simplex is concerned, we can ignore it.

Finally, to conclude the proof, suppose that $\alpha < \gamma(C) = \max_{i>m} \deg_C(i) - m + 1$. Then there exists $i > m$ such that $m - 1 + \alpha < \deg_C(i)$ and hence, dividing by m, we get $\pi_i(\mathbf{x}) - \pi(\mathbf{x}) > 0$ and then $g_i(\mathbf{x}) = \phi(\pi_i(\mathbf{x})) - \phi(\pi(\mathbf{x})) > 0$, which implies that a transversal eigenvalue of $J(\mathbf{x})$ is positive, that is, $\mathbf{x}$ is unstable.

Theorem 4 provides us with an immediate strategy to avoid unwanted local solutions: maximal cliques that are not maximum. Suppose that C is a maximal clique in G that we want to avoid. By letting $\alpha < \gamma(C)$, its characteristic vector $\mathbf{x}^C$ becomes an unstable stationary point of any imitation dynamics under f_α, and thus will not be approached by any interior trajectory. Hence, if there is a clique D such that still $\gamma(D) < \alpha$ holds, there is a (more or less justified) hope to obtain in the limit $\mathbf{x}^D$, which yields automatically a larger maximal clique D. Unfortunately, two other cases could occur: (1) no other clique T satisfies $\gamma(T) < \alpha$, that is, α has a too large absolute value, and (2) even if there is such a clique, other attractors could emerge that are not characteristic vectors of a clique (note that this is excluded if $\alpha > 0$ by theorem 3). The proper choice of the parameter α is therefore a trade-off between the desire to remove unwanted maximal cliques and the emergence of spurious solutions.

Instead of keeping the value of α fixed, our approach is to start with a sufficiently large negative α and adaptively increase it during the optimization process, in much the same spirit as simulated or mean field annealing procedures. Of course, in our case, the annealing parameter has no interpretation in terms of a hypothetical temperature. The rationale behind this idea is that for values of α that are sufficiently negative, only the characteristic vectors of large maximal cliques will be stable, attractive points for the imitation dynamics, together with a set of spurious solutions. As the value of α increases, spurious solutions disappear, and at the same time, (characteristic vectors of) smaller maximal cliques become stable. We expect that at the beginning of the annealing process, the dynamics is attracted toward "promising" regions, and the search is further refined as the annealing parameter increases. In summary, a high-level description of the proposed algorithm is shown in Figure 2. Note that the last step in the algorithm is necessary if we also want to extract the vertices comprising the clique found, as shown in theorem 3.

It is clear that for the algorithm to work, we need to select an appropriate annealing schedule. To this end, we employ the following heuristic suggested in Bomze et al. (2002). Suppose that the underlying graph is a random one in the sense that edges are generated independent of each other with a certain probability q (in applications, q will be replaced by the actual graph density), and suppose that C is an

Algorithm

1. Start with a sufficiently large negative α.
2. Let $\mathbf{b}$ be the barycenter of Δ and set $\mathbf{x} = \mathbf{b}$.
3. Run any imitation dynamics starting from $\mathbf{x}$, under $A_G + \alpha I$ until convergence and let $\mathbf{x}$ be the converged point.
4. Unless a stopping condition is met, increase α and goto 3.
5. Select $\hat{\alpha}$ with $0 < \hat{\alpha} < 1$ (e.g., $\hat{\alpha} = \frac{1}{2}$), run any imitation dynamics starting from current $\mathbf{x}$ under $A_G + \hat{\alpha} I$ until convergence, and extract a maximal clique from the converged solution.

Figure 2: Annealed Imitation Heuristic.

unwanted clique of size m. Take $\delta > 0$ small, say 0.01, and consider the quantity

$$\overline{\gamma}_m = 1 - (1-q)m - \sqrt{mq(1-q)}\,\delta^{\nu}, \tag{6.7}$$

where $\nu = 1/2(n-m)$. Bomze et al. (2002) proved that $\gamma(C)$ exceeds $\overline{\gamma}_m$ with probability $1-\delta$. Thus, it makes sense to use $\overline{\gamma}_m$ as a heuristic proxy for the lower bound of $\gamma(C)$, to avoid being attracted by a clique of size m.

Furthermore, note that among all graphs with n vertices and m edges, the maximum possible clique number is the only integer c that satisfies the following relations:

$$\binom{c}{2} \leq m < \binom{c+1}{2}, \tag{6.8}$$

which, after some algebra, yields

$$\sqrt{\frac{8m+1}{4}} - \frac{1}{2} < c \leq \sqrt{\frac{8m+1}{4}} + \frac{1}{2}, \tag{6.9}$$

from which we get

$$c = \left\lfloor \sqrt{\frac{8m+1}{4}} + \frac{1}{2} \right\rfloor. \tag{6.10}$$

The previous results suggest us a sort of two-level annealing strategy: the level of clique size, which in turn induces that of the "actual" annealing parameter. More precisely, if we do not have any a priori information about the expected size of the maximum clique, we can use equation 6.10 to have an initial overestimation of it. By setting the initial value for α (step 1 of our algorithm) at some intermediate value between $\overline{\gamma}_c$ and $\overline{\gamma}_{c-1}$, for example, $\alpha = (\overline{\gamma}_c + \overline{\gamma}_{c-1})/2$, we expect that only the characteristic vectors of maximal cliques having size c will survive in f_α, together with many spurious solutions. After the initial cycle, we decrease c, recalculate $\overline{\gamma}_c$ and $\overline{\gamma}_{c-1}$, and update $\alpha = (\overline{\gamma}_c + \overline{\gamma}_{c-1})/2$ in step 4 as in the previous step. The whole process is iterated until either c reaches 1 or α becomes greater than zero.

7 Experimental Results

In this section we present experiments of applying our annealed imitation heuristics to the same set of random and DIMACS graphs used in the previous experiments. For each graph considered, the algorithms were run by using the two-level annealing schedule described at the end of the previous section. As for the internal cycle (step 3), we used both the (discretized versions of the) linear and exponential replicator dynamics. The processes were iterated until the Euclidean distance between two successive states became smaller than a threshold value. At the final cycle (step 5), the parameter $\hat{\alpha}$ was set to 1/2, and the dynamics were stopped when either a maximal clique (i.e., a local maximizer of $f_{1/2}$ on Δ) was found or the distance between two successive points was smaller than a fixed threshold. When the process converged to a saddle point, the vector was perturbed, and the algorithm restarted from the new perturbed point.

Table 4 and Figure 3 show the results obtained with our annealed imitation heuristics (AIH) on the random graphs, in terms of clique sizes and computation time, respectively, while Tables 5 and 6 show the results obtained on the DIMACS benchmark.

Several conclusions can be drawn from these experiments. First, both of our annealing heuristics perform, on average, significantly better than the

Table 4: Results of the Annealed Imitation Heuristics (AIH) and State-of-the-Art Neural Network–Based or Optimization-Based Approaches on Random Graphs with Varying Order and Density.

n	ρ	AIH Linear	AIH Exponential	CHD	MFA	SLDN	FTL	IHN	HNL
100	0.25	5.11 ± 0.53	5.22 ± 0.46	4.48	—	4.83	4.2	—	—
	0.50	8.43 ± 0.77	8.84 ± 0.63	7.38	8.50	8.07	8.0	9.13	9
	0.75	16.10 ± 0.92	16.43 ± 0.83	13.87	—	15.05	14.1	—	—
	0.90	29.53 ± 1.71	30.20 ± 1.52	27.92	30.02	—	—	—	30
	0.95	42.82 ± 1.74	42.94 ± 1.75	—	—	—	—	—	—
200	0.25	5.60 ± 0.60	5.87 ± 0.51	—	—	—	4.9	—	—
	0.50	9.43 ± 0.89	10.14 ± 0.55	—	—	—	8.5	10.60	11
	0.75	19.59 ± 1.14	19.96 ± 1.00	—	—	—	—	—	—
	0.90	38.62 ± 1.80	39.77 ± 1.50	—	—	—	—	—	39
	0.95	60.15 ± 1.79	60.60 ± 1.64	—	—	—	—	—	—
300	0.25	6.10 ± 0.54	6.30 ± 0.46	—	—	—	5.1	—	—
	0.50	9.98 ± 0.92	10.89 ± 0.60	—	—	—	8.9	11.60	11
	0.75	21.02 ± 1.11	21.78 ± 0.84	—	—	—	—	—	—
	0.90	43.84 ± 2.02	45.33 ± 1.57	—	—	—	—	—	46
	0.95	70.98 ± 2.35	72.09 ± 1.75	—	—	—	—	—	—
400	0.25	6.65 ± 0.50	6.43 ± 0.54	5.53	—	5.70	4.9	—	—
	0.50	11.15 ± 0.76	11.24 ± 0.73	9.24	10.36	9.91	8.9	12.30	—
	0.75	22.82 ± 0.89	22.80 ± 0.90	18.79	—	20.44	17.7	—	—
	0.90	48.68 ± 1.51	48.65 ± 1.51	43.24	49.94	—	—	—	—
	0.95	79.19 ± 2.00	79.00 ± 2.03	—	—	—	—	—	—
500	0.25	6.28 ± 0.68	6.68 ± 0.55	—	—	—	6.2	—	—
	0.50	10.53 ± 0.84	11.73 ± 0.71	—	—	—	9.4	12.80	12
	0.75	22.70 ± 1.24	23.91 ± 0.85	—	—	—	—	—	—
	0.90	49.75 ± 2.10	52.26 ± 1.46	—	—	—	—	—	56
	0.95	84.87 ± 2.16	86.50 ± 2.51	—	—	—	—	—	—
1000	0.25	6.61 ± 0.73	7.17 ± 0.45	6.03	—	6.17	5.8	—	—
	0.50	11.32 ± 0.85	12.73 ± 0.78	10.25	—	10.93	10.4	—	—
	0.75	24.88 ± 1.40	26.63 ± 1.03	21.26	—	23.19	21.4	—	—
	0.90	56.90 ± 2.40	60.46 ± 1.60	—	—	—	—	—	—
	0.95	101.98 ± 3.10	104.93 ± 3.47	—	—	—	—	—	—

corresponding plain replicator dynamics, where no annealing strategy is used, while paying a time penalty that is in most cases limited. Moreover, similar to what we found in section 5.1 for the plain processes, the exponential version of AIH provides larger cliques than its linear counterpart, thereby improving the results reported in Bomze et al. (2002).

As for the comparison with other neural-based clique finding algorithms, we see that AIH performs substantially better than CHD and FTL, which were already outperformed by the plain dynamics, SLDN, mean field annealing (MFA), and the inverted neurons network (INN). Furthermore, it provides results comparable to those obtained with the continuous-based heuristic (CBH) and Hopfield network learning (HNL).

Table 5: Results of the Annealed Imitation Heuristics (AIH) and State-of-the-Art Neural Network–Based or Optimization-Based Approaches on DIMACS Benchmark Graphs, Part I.

			Clique Size								*Time (secs)*	
Graph	n	ρ	DIMACS Best	AIH Linear	AIH Exponential	MFA	INN Average	IHN	CBH	QSH	AIH 1st	AIH Exponential
brock200_1	200	0.75	21	20	20	19	—	—	20	21	1.39	0.85
brock200_2	200	0.50	12	10	10	9	9.1	—	12	12	0.50	0.27
brock200_3	200	0.61	15	12	13	11	—	—	14	15	1.07	0.43
brock200_4	200	0.66	17	16	16	14	13.4	—	16	17	1.77	0.38
brock400_1	400	0.75	24	21	24	24	—	—	23	27	1.98	1.73
brock400_2	400	0.75	25	22	24	21	21.3	—	24	29	2.06	3.34
brock400_3	400	0.75	25	23	24	22	—	—	23	31	4.05	2.36
brock400_4	400	0.75	33	23	23	23	21.3	—	24	33	6.57	1.59
brock800_1	800	0.65	21	18	20	16	—	—	20	17	10.00	6.66
brock800_2	800	0.65	21	18	18	16	16.9	—	19	24	10.06	7.03
brock800_3	800	0.65	21	18	19	16	—	—	20	25	18.46	5.85
brock800_4	800	0.65	21	18	18	15	16.5	—	19	26	18.52	21.35
c-fat200-1	200	0.08	12	10	12	6	—	12	12	12	0.17	0.20
c-fat200-2	200	0.16	24	22	24	24	—	24	24	24	0.21	0.18
c-fat200-5	200	0.43	58	58	58	58	—	58	58	58	0.20	0.17
c-fat500-1	500	0.04	14	12	14	—	—	14	14	14	1.03	1.31
c-fat500-2	500	0.07	26	24	26	—	—	26	26	26	1.10	1.23
c-fat500-5	500	0.19	64	62	64	—	—	64	64	64	1.59	1.21
c-fat500-10	500	0.37	126	124	126	—	—	—	126	126	1.27	1.18
hamming6-2	64	0.90	32	32	32	—	—	32	32	32	0.06	0.03
hamming6-4	64	0.35	4	4	4	—	—	4	4	4	0.02	0.02

Table 5: Continued

			Clique Size								*Time (secs)*	
Graph	n	ρ	DIMACS Best	AIH Linear	AIH Exponential	MFA	INN Average	IHN	CBH	QSH	AIH 1st	AIH Exponential
hamming8-2	256	0.97	128	128	128	—	—	128	128	128	2.22	0.65
hamming8-4	256	0.64	16	16	16	—	16.0	16	16	16	0.39	0.36
hamming10-2	65536	0.99	512	512	512	—	—	512	—	—	405.76	150.89
hamming10-4	65536	0.83	36	33	33	—	32.0	36	—	—	155.06	16.61
johnson8-2-4	28	0.56	4	4	4	—	—	4	4	4	0.01	0.01
johnson8-4-4	70	0.77	14	14	14	—	—	14	14	14	0.03	0.05
johnson16-2-4	120	0.76	8	8	8	—	—	8	8	8	0.12	0.07
johnson32-2-4	496	0.88	16	16	16	—	—	16	16	16	4.54	1.69
keller4	171	0.65	11	8	9	—	8.4	—	10	11	0.67	0.92
keller5	776	0.75	27	15	16	—	16.7	—	21	24	31.83	679.00
keller6	3361	0.82	59	31	31	—	33.4	—	—	—	1444.15	869.99

Table 6: Results of the Annealed Imitation Heuristics (AIH) and State-of-the-Art Neural Network–Based or Optimization-Based Approaches on DIMACS Benchmark Graphs, Part II.

			Clique Size								*Time (secs)*	
Graph	n	ρ	DIMACS Best	AIH Linear	AIH Exponential	MFA	INN Average	IHN	CBH	QSH	AIH 1st	AIH Exponential
p_hat300-1	300	0.24	8	6	7	—	7.0	8	8	7	0.58	1.05
p_hat300-2	300	0.49	25	24	25	—	21.9	25	25	24	0.70	2.15
p_hat300-3	300	0.74	36	33	36	—	33.1	36	36	33	1.00	0.98
p_hat500-1	500	0.25	9	8	8	—	—	9	9	9	1.71	3.98
p_hat500-2	500	0.50	36	33	36	—	—	36	35	33	2.13	6.07
p_hat500-3	500	0.75	50	43	49	—	—	49	49	46	3.20	5.68
p_hat700-1	700	0.25	11	8	9	—	8.4	11	11	8	3.45	3.70
p_hat700-2	700	0.50	44	38	44	—	42.0	44	44	42	4.60	11.01
p_hat700-3	700	0.75	62	56	60	—	58.2	61	60	59	6.25	7.94
p_hat1000-1	1000	0.25	10	9	10	—	—	10	—	—	7.19	8.89
p_hat1000-2	1000	0.50	46	43	43	—	—	46	—	—	9.35	10.65
p_hat1000-3	1000	0.75	66	60	64	—	—	68	—	—	12.43	32.35
p_hat1500-1	1500	0.25	11	9	10	—	9.4	—	—	—	16.60	27.84
p_hat1500-2	1500	0.50	65	60	64	—	60.2	—	—	—	23.09	31.57
p_hat1500-3	1500	0.75	94	89	92	—	86.2	—	—	—	46.56	50.67
san200_0.7_1	200	0.70	30	15	15	—	—	30	15	30	3.60	1.54
san200_0.7_2	200	0.70	18	12	12	—	—	15	12	18	3.65	1.39
san200_0.9_1	200	0.90	70	46	46	42	—	70	46	70	21.84	3.10
san200_0.9_2	200	0.90	60	37	38	—	—	41	36	60	3.02	3.64
san200_0.9_3	200	0.90	44	36	35	33	—	—	30	35	6.57	5.59
san400_0.5_1	400	0.50	13	7	7	—	—	—	8	9	2.22	2.80

Table 6: Continued

			Clique Size								*Time (secs)*	
Graph	n	ρ	DIMACS Best	AIH Linear	AIH Exponential	MFA	INN Average	IHN	CBH	QSH	AIH 1st	AIH Exponential
san400_0.7_1	400	0.70	40	20	20	—	—	40	20	40	21.83	9.53
san400_0.7_2	400	0.70	30	15	15	—	—	30	15	30	16.14	6.18
san400_0.7_3	400	0.70	22	12	12	—	—	—	14	16	14.82	4.90
san400_0.9_1	400	0.90	100	50	51	—	—	100	50	100	90.28	14.45
san1000	1000	0.50	10	8	8	—	—	10	—	—	27.46	19.37
sanr200_0.7	200	0.70	18	18	18	18	—	17	18	15	0.65	0.35
sanr200_0.9	200	0.90	42	38	41	41	—	41	41	37	4.18	1.74
sanr400_0.5	400	0.50	13	12	12	—	—	12	12	11	1.26	1.63
sanr400_0.7	400	0.70	21	18	21	—	—	21	20	18	4.34	2.34

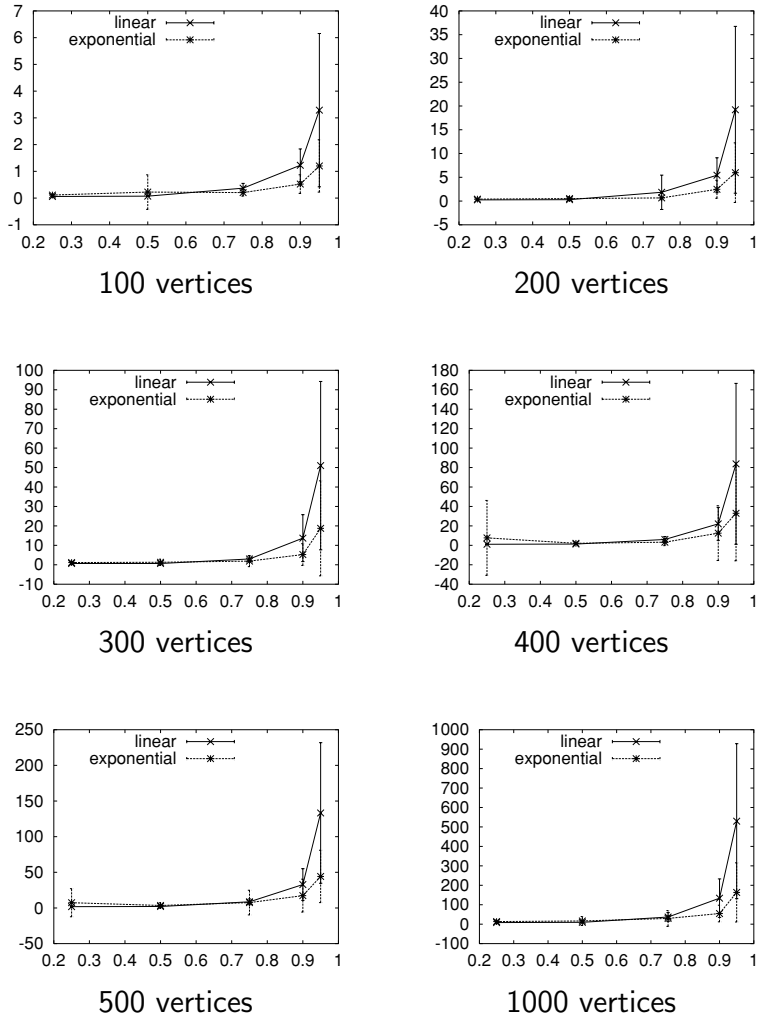

Figure 3: CPU time of the annealed imitation heuristics on random graphs with varying order and density. The x-axis represents the edge density, while the y-axis denotes time (in seconds).

The comparison with the iterative Hopfield nets (IHN) and the QSH algorithm is not as straightforward and depends on the specific graph class. More specifically, on `c-fat`, `hamming`, and `johnson` graphs, the proposed AIH approaches perform as well as IHN and QSH. The annealed dynamics outperform QSH on the `sanr` graphs and provide slightly better results on the `p_hat` graphs, while QSH performs better on the `brock`

and the keller graphs. Note, however, that we do not provide results for the largest instances such as hamming10-2, hamming10-4, keller6, san1000, and all the p_hat's with 1000 or more vertices. As for IHN, the approach performs slightly better than AIH on p-hat, although no result for the largest instances is given, and gives comparable results on the sanr graphs. However, we did not provide results for the brock and the keller graphs, which are notoriously hard instances (Brockington & Culberson, 1996).

The sanc graphs deserve a separate discussion. Indeed, these graphs were already found to be particularly hard for Motzkin-Straus-based approaches, like the proposed dynamics and CBH (Bomze et al. 2002). As expected, our approaches fail to provide good results and are substantially outperformed by IHN and QSH.

8 Conclusions

In this letter, we have introduced a wide family of game dynamic equations known as *payoff-monotonic* dynamics and have shown how their dynamical properties make any member of this family a potential heuristic for solving standard quadratic programs and, in particular, the maximum clique problem (MCP). Such systems can easily be implemented in a parallel network of locally interacting computational units and can be coded in a few lines of any high-level programming language. We have shown experimentally that an exponential version of the classic (linear) replicator dynamics is particularly effective at finding maximum or near-maximum cliques for several graph classes, being dramatically faster and even more accurate than its linear counterpart. However, these models are inherently unable to avoid poor local solutions in harder graph instances.

In an attempt to avoid local optima, we have focused on a particular subclass of these dynamics used to model the evolution of behavior via imitation processes, and have developed the annealed imitation dynamics. This is a class of heuristics for MCP whose basic ingredients are (1) a parameterized continuous formulation of the problem, (2) an instability analysis of equilibria of imitation dynamics, and (3) a principled way of varying a regularization parameter during the evolution process. Experiments on various benchmark graphs have shown that the annealed imitation class contains algorithms that substantially outperform classic neural network algorithms for maximum clique, such as mean field annealing, and compares well with sophisticated MCP heuristics from the continuous optimization literature.

Appendix A: Discrete-Time Replicator Dynamics

The results presented in section 3, and in particular theorem 1, show that continuous-time payoff-monotonic dynamics can be usefully employed to

find local solutions of standard quadratic programs. In practical computer implementations, however, we need a way of discretizing the models. A customary way of doing this is given by the following difference equations:

$$x_i(t+1) = \frac{x_i(t)\pi_i(t)}{\sum_{j=1}^{n} x_j(t)\pi_j(t)} \tag{A.1}$$

and

$$x_i(t+1) = \frac{x_i(t)e^{\kappa\pi_i(t)}}{\sum_{j=1}^{n} x_j(t)e^{\kappa\pi_j(t)}}, \tag{A.2}$$

which correspond to well-known discretizations of equations 2.6 and 2.7, respectively (Cabrales & Sobel, 1992; Gaunersdorfer & Hofbauer, 1995; Hofbauer & Sigmund, 1998; Weibull, 1995). Note that model A.1 is the standard discrete-time replicator dynamics, which have already proven to be remarkably effective in tackling maximum clique and related problems, and to be competitive to other more elaborated neural network heuristics (Bomze, 1997; Bomze et al., 1997, 2000; Pelillo, 1995, 1999; Pelillo et al., 1999). Equation A.2 has been used in Pelillo (1999, 2002) as a heuristic for graph and tree isomorphism problems.

As their continuous counterparts, these dynamics are payoff-monotonic, that is,

$$\frac{x_i(t+1) - x_i(t)}{x_i(t)} > \frac{x_j(t+1) - x_j(t)}{x_j(t)} \Leftrightarrow \pi_i(t) > \pi_j(t).$$

It is a well-known result in evolutionary game theory (Weibull, 1995; Hofbauer & Sigmund, 1998) that the fundamental theorem of natural selection (see theorem 1) also holds for the first-order linear dynamics (see equation A.1)—namely, the average consistency $\mathbf{x}'A\mathbf{x}$ is a (strict) Lyapunov function for equation A.1, provided that $A = A'$. In other words:

$$\mathbf{x}(t)'A\mathbf{x}(t) < \mathbf{x}(t+1)'A\mathbf{x}(t+1)$$

unless $\mathbf{x}(t)$ is a stationary point. Unfortunately, unlike the continuous-time case, there is no such result for the discrete exponential dynamics, equation A.2. That is, there is no guarantee that for any fixed value of the parameter, κ, the dynamics increases the value of $\mathbf{x}'A\mathbf{x}$. Indeed, with high values of this parameter, the dynamics can exhibit an oscillatory behavior (Pelillo, 1999). However, a recent result by Bomze (2005) allows us to define an adaptive approach that is guaranteed to find a (local) maximizer for $\mathbf{x}'A\mathbf{x}$.

We define the ε-stationary points as

$$\mathrm{Stat}_\varepsilon = \left\{ \mathbf{x} \in \Delta : \sum_i x_i ((A\mathbf{x})_i - \mathbf{x}' A\mathbf{x})^2 < \varepsilon \right\}.$$

Clearly, this set is composed of the union of open neighborhoods around the stationary points of any payoff-monotonic dynamics, and for $\varepsilon \to 0$ shrinks toward the stationary points themselves. Let $\bar{m}_A = \max_{ij} |a_{ij}|$, $\mathrm{span}(A) = \max_{ij} a_{ij} - \min_{ij} a_{ij}$, and, for any given ε, define $\kappa_A(\varepsilon)$ as the unique $\kappa > 0$, which satisfies

$$\kappa \exp(2\bar{m}_A \kappa) = \frac{2\varepsilon}{\mathrm{span}(A)(\varepsilon + 2\bar{m}_A^2)}.$$

Theorem 5. *Suppose $A = A'$. Then for arbitrary $\varepsilon > 0$, for any positive $\kappa \leq \kappa_A(\varepsilon)$, the objective function $\mathbf{x}' A\mathbf{x}$ is strictly increasing over time along the parts of trajectories under equation A.2, which are not ε-stationary, that is,*

$$\mathbf{x}(t)' A\mathbf{x}(t) < \mathbf{x}(t+1)' A\mathbf{x}(t+1) \quad \text{if} \quad \mathbf{x}(t) \notin Stat_\varepsilon.$$

Proof. See Bomze (2005).

This means that for each point $\mathbf{x} \in \Delta$ we can find a κ for which one iteration of equation A.2 increases $\mathbf{x}' A\mathbf{x}$. That is, by setting at each iteration $\kappa = \kappa_A(\varepsilon)$, we are guaranteed to increase $\mathbf{x}' A\mathbf{x}$ along the trajectories of the system. Note, however, that this estimate of $\kappa_A(\varepsilon)$ is not tight. In particular, our experience shows that it severely underestimated the value of κ, slowing the convergence of the dynamics considerably.

In order to obtain a better estimate of the parameter κ and improve the performance of the approach, in our experiments we employed the adaptive exponential dynamics described in Figure 4, which, as the next proposition shows, has $\mathbf{x}' A\mathbf{x}$ as a Lyapunov function.

Proposition 8. *If the payoff matrix A is symmetric, then the function $\mathbf{x}' A\mathbf{x}$ is strictly increasing along any nonconstant trajectory of the adaptive exponential dynamics defined above. In other words, $\mathbf{x}(t)' A\mathbf{x}(t) \leq \mathbf{x}(t+1)' A\mathbf{x}(t+1)$ for all t, with equality if and only if $\mathbf{x} = \mathbf{x}(t)$ is a stationary point.*

Proof. By construction, the function is guaranteed to grow as long a κ that increases $\mathbf{x}' A\mathbf{x}$ can be found. Theorem 5 guarantees that such a κ can indeed be found.

Algorithm

1. Start with a sufficiently large κ and from an arbitrary $\mathbf{x}(0) \in \Delta$. Set $t \leftarrow 0$.
2. While $\mathbf{x}(t)$ is not stationary do
3. Compute $\mathbf{x}(t+1)$ using equation A.2;
4. While $\mathbf{x}'(t+1)A\mathbf{x}(t+1) \leq \mathbf{x}'(t)A\mathbf{x}(t)$ do
5. Reduce κ;
6. Recompute $\mathbf{x}(t+1)$ using equation A.2;
7. Endwhile;
8. $t \leftarrow t+1$;
9. Endwhile;

Figure 4: Adaptive exponential (discrete-time) replicator dynamics.

Appendix B: Proof of Theorem 3

Theorem 3. *Let C be a subset of vertices of a graph G, and let $\mathbf{x}^C$ be its characteristic vector. Then, for any $0 < \alpha < 1$, C is a maximal (maximum) clique of G if and only if $\mathbf{x}^C$ is a local (global) solution of equation 4.3. Moreover, all solutions of the equation 4.3 are strict and are characteristic vectors of maximal cliques of G.*

Proof. Suppose that C is a maximal clique of G, and let $|C| = m$. We shall prove that $\mathbf{x}^C$ is a strict local solution of program 4.3. To this end, we use standard second-order sufficiency conditions for constrained optimization

(Luenberger, 1984). Let $A_G = (a_{ij})$ be the adjacency matrix of G and, for notational simplicity, put

$$A = A_G + \alpha I.$$

First, we need to show that $\mathbf{x}^C$ is a KKT point for equation 4.3. It is easy to see that since C is a maximal clique, we have:

$$(A_G \mathbf{x}^C)_i \begin{cases} = \frac{m-1}{m} & \text{if } i \in C \\ \leq \frac{m-1}{m} & \text{if } i \notin C. \end{cases}$$

Hence, if $i \in C$, then

$$(A\mathbf{x}^C)_i = (A_G\mathbf{x}^C)_i + \alpha x_i^C = \frac{m-1}{m} + \frac{\alpha}{m}, \tag{B.1}$$

and if $i \notin C$,

$$(A\mathbf{x}^C)_i = (A_G\mathbf{x}^C)_i \leq \frac{m-1}{m} < \frac{m-1}{m} + \frac{\alpha}{m}\,. \tag{B.2}$$

Therefore, conditions 3.2 are satisfied and $\mathbf{x}^C$ is a KKT point. Note that the Lagrange multipliers μ_i's defined in section 3 are given by

$$\mu_i = \frac{m-1+\alpha}{m} - (A\mathbf{x}^C)_i.$$

To conclude the first part of the proof, it remains to show that the Hessian of the Lagrangian associated with program 4.3, which in this case is simply $A_G + \alpha I$, is negative definite on the following subspace:

$$\Gamma = \{\mathbf{y} \in \mathbb{R}^n \,:\, \mathbf{e}'\mathbf{y} = 0 \text{ and } y_i = 0 \text{ for all } i \in \Upsilon\},$$

where

$$\Upsilon = \{i \in V \,:\, x_i^C = 0 \text{ and } \mu_i > 0\}\,.$$

But from equation B.2, $\Upsilon = V \backslash C$. Hence, for all $\mathbf{y} \in \Gamma$, we have:

$$\begin{aligned} \mathbf{y}' A\mathbf{y} &= \sum_{i=1}^{n} y_i \sum_{j=1}^{n} a_{ij} y_j + \alpha \sum_{i=1}^{n} y_i^2 \\ &= \sum_{i \in C} y_i \sum_{j \in C} a_{ij} y_j + \alpha \sum_{i \in C} y_i^2 \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i \in C} y_i \left(\sum_{j \in C} y_j - y_i \right) + \alpha \sum_{i \in C} y_i^2 \\
&= (\alpha - 1) \sum_{i \in C} y_i^2 \\
&= (\alpha - 1) \mathbf{y}' \mathbf{y} \\
&\leq 0
\end{aligned}
$$

with equality if and only if $\mathbf{y} = \mathbf{0}$, the null vector. This proves that $A_G + \alpha I$ is negative definite on Γ, as required.

To prove the inverse direction, suppose that $\mathbf{x}^C \in \Delta$ is a local solution to equation 4.3 and hence a KKT point. By proposition 2, $\mathbf{x}^C$ is also a stationary point for payoff-monotonic dynamics, and since A has positive diagonal entries $a_{ii} = \alpha > 0$, all the hypotheses of proposition 4 are fulfilled. Therefore, it follows that C is a clique (i.e., $a_{ij} > 0$ for all $i, j \in C$); otherwise $\mathbf{x}^C$ could not be a local solution of equation 4.3. On the other hand, from proposition 5, C is also a maximal clique.

Furthermore, if $\mathbf{x}$ is any local solution, and hence a KKT point of equation 4.3, then necessarily $\mathbf{x} = \mathbf{x}^S$ where $S = \sigma(\mathbf{x})$. Geometrically, this means that $\mathbf{x}$ is the barycenter of its own face. In fact, from the previous discussion, S has to be a (maximal) clique. Therefore, for all $i \in S$,

$$(A_G \mathbf{x})_i + \alpha x_i = 1 - (1 - \alpha) x_i = \lambda,$$

for some constant λ. This amounts to saying that x_i is constant for all $i \in \sigma(\mathbf{x})$, and $\sum_i x_i = 1$ yields $x_i = 1/|S|$. From what we have seen in the first part of the proof, this also shows that all local solutions of equation 4.3 are strict.

Finally, as for the "global/maximum" part of the theorem, simply notice that at local solutions $\mathbf{x} = \mathbf{x}^S$ of equation 4.3, $S = \sigma(\mathbf{x})$ being a maximal clique, the value of the objective function f_α is $1 - (1 - \alpha)/|S|$.

Acknowledgments

We thank Manuel Bomze for many stimulating discussions and Claudio Rossi for his help in early stages of this work.

References

Aarts. E., & Korst. J. (1989). *Simulated Annealing and Boltzmann machines*, New York: Wiley.

Ballard, D. H., Gardner, P. C., & Srinivas, M. A. (1987). *Graph problems and connectionist architectures* (Tech. Rep. No. TR 167). Rochester, NY: University of Rochester.

Bertoni, A., Campadelli, P., & Grossi, G. (2002). A neural algorithm for the maximum clique problem: Analysis, experiments and circuit implementation. *Algorithmica*, *33*(1), 71–88.

Bhatia, N. P., & Szegö, G. P. (1970). *Stability theory of dynamical systems.* Berlin: Springer-Verlag.

Bomze, I. M. (1986). Non-cooperative two-person games in biology: A classification. *Int. J. Game Theory*, *15*(1), 31–57.

Bomze, I. M. (1997). Evolution towards the maximum clique. *J. Global Optim.*, *10*, 143–164.

Bomze, I. M. (1998). On standard quadratic optimization problems. *J. Global Optim.*, *13*, 369–387.

Bomze, I. (2005). Portfolio selection via replicator dynamics and projections of indefinite estimated covariances. *Dynamics of Continuous, Discrete and Impulsive Systems B*, *12*, 527–564.

Bomze, I. M., Budinich, M., Pardalos, P. M., & Pelillo, M. (1999). The maximum clique problem. In D.-Z. Du & P. M. Pardalos (Eds.), *Handbook of combinatorial optimization (Suppl. Vol. A)*, (pp. 1–74). Boston: Kluwer.

Bomze, I. M., Budinich, M., Pelillo, M., & Rossi, C. (2002). Annealed replication: A new heuristic for the maximum clique problem. *Discr. Appl. Math.*, *121*(1–3), 27–49.

Bomze, I. M., Pelillo, M., & Giacomini, R. (1997). Evolutionary approach to the maximum clique problem: Empirical evidence on a larger scale. In I. M. Bomze, T. Csendes, R. Horst, & P. M. Pardalos (Eds.), *Developments in global optimization* (pp. 95–108). Dordrecht: Kluwer.

Bomze, I. M., Pelillo, M., & Stix, V. (2000). Approximating the maximum weight clique using replicator dynamics. *IEEE Trans. Neural Networks*, *11*(6), 1228–1241.

Boppana, R., & Halldórsson, M. M. (1992). Approximating maximum independent sets by excluding subgraphs. *BIT*, *32*, 180–196.

Brockington, M., & Culberson, J. C. (1996). Camouflaging independent sets in quasi-random graphs. In D. Johnson & M. Trick (Eds.). *Cliques, coloring and satisfiability: Second DIMACS implementation challenge* (pp. 75–88). Providence, RI: American Mathematical Society.

Busygin, S., Butenko, S., & Pardalos, P. M. (2002). A heuristic for the maximum independent set problem based on optimization of a quadratic over a sphere. *J. Comb. Optim.*, *6*, 287–297.

Cabrales, A., & Sobel, J. (1992). On the limit points of discrete selection dynamics. *J. Econom. Theory*, *57*, 407–419.

Fisher, R. A. (1930). *The genetical theory of natural selection.* New York: Oxford University Press.

Fudenberg, D., & Levine, D. K. (1998). *The theory of learning in games*. Cambridge, MA: MIT Press.

Funabiki, N., Takefuji, Y., & Lee, K.-C. (1992). A neural network model for finding a near-maximum clique. *J. Parallel Distrib. Comput.*, *14*, 340–344.

Gaunersdorfer, A., & Hofbauer, J. (1995). Fictitious play, Shapley polygons, and the replicator equation. *Games Econom. Behav.*, *11*, 279–303.

Gee, A. W., & Prager, R. W. (1994). Polyhedral combinatorics and neural networks. *Neural Computation*, *6*, 161–180.

Gibbons, L. E., Hearn, D. W., Pardalos, P. M., & Ramana, M. V. (1997). Continuous characterizations of the maximum clique problem. *Math. Oper. Res.*, *22*, 754–768.

Godbeer, G. H., Lipscomb, J., & Luby, M. (1988). *On the computational complexity of finding stable state vectors in connectionist models (Hopfield nets)* (Tech. Rep. No. 208/88). Toronto: University of Toronto.

Grossman, T. (1996). Applying the INN model the maximum clique problem. In D. Johnson & M. Trick (Eds.), *Cliques, coloring and satisfiability: Second DIMACS Implementation Challenge* (pp. 122–145). Providence, RI: American Mathematical Society.

Grötschel, M., Lovász, L., & Schrijver, A. (1993). *Geometric algorithms and combinatorial optimization*. Berlin: Springer-Verlag.

Hastad, J. (1996). Clique is hard to approximate within $n^{1-\varepsilon}$. In *Proc. 37th Ann. Symp. Found. Comput. Sci.* (pp. 627–636). Los Alamitos, CA: IEEE Computer Society Press.

Hirsch, M. W., & Smale, S. (1974). *Differential equations, dynamical systems, and linear algebra*. New York: Academic Press.

Hofbauer, J. (1995). *Imitation dynamics for games*. Collegium Budapest. Unpublished manuscript.

Hofbauer, J., & Sigmund, K. (1998). *Evolutionary games and population dynamics*. Cambridge: Cambridge University Press.

Hummel, R. A., & Zucker, S. W. (1983). On the foundations of relaxation labeling processes. *IEEE Trans. Pattern Anal. Machine Intell.*, *5*, 267–287.

Jagota, A. (1995). Approximating maximum clique with a Hopfield neural network. *IEEE Trans. Neural Networks*, *6*, 724–735.

Jagota, A., Pelillo, M., & Rangarajan, A. (2000). A new deterministic annealing algorithm for maximum clique. In *Proc. IJCNN'2000: Int. J. Conf. Neural Networks* (pp. 505–508). Piscataway, NJ: IEEE Press.

Jagota, A., & Regan, K. W. (1997). Performance of neural net heuristics for maximum clique on diverse highly compressible graphs. *J. Global Optim.*, *10*, 439–465.

Jagota, A., Sanchis, L., & Ganesan, R. (1996). Approximately solving maximum clique using neural networks and related heuristics. In D. Johnson & M. Trick (Eds.), *Cliques, coloring and satisfiability: Second DIMACS Implementation Challenge* (pp. 169–204). Providence, RI: American Mathematical Society.

Johnson, D., & Trick, M. (1996). *Cliques, coloring and satisfiability: Second DIMACS Implementation Challenge.* Providence, RI: American Mathematical Society.

Lin, F., & Lee, K. (1992). A parallel computation network for the maximum clique problem. In *Proc. 1st Int. Conf. Fuzzy Theory Tech.* Baton Rouge, LA.

Luenberger, D. G. (1984). *Linear and nonlinear programming*. Reading, MA: Addison-Wesley.

Maynard Smith, J. (1982). *Evolution and the theory of games*. Cambridge: Cambridge University Press.

Miller, D. A., & Zucker, S. W. (1999). Efficient simplex-like methods for equilibria of nonsymmetric analog networks. *Neural Computation*, *4*, 167–190.

Miller, D. A., & Zucker, S. W. (1999). Computing with self-excitatory cliques: A model and an application to hyperacuity-scale computation in visual cortex. *Neural Computation*, *11*, 21–66.

Motzkin, T. S., & Straus, E. G. (1965). Maxima for graphs and a new proof of a theorem of Turán. *Canad. J. Math.*, *17*, 533–540.

Papadimitriou, C. H., & Steiglitz, K. (1982). *Combinatorial optimization: Algorithms and complexity*, Englewood Cliffs, NJ: Prentice Hall.

Pardalos, P. M., & Rodgers, G. P. (1990). Computational aspects of a branch and bound algorithm for quadratic zero-one programming. *Computing*, *45*, 131–144.

Pekergin, F., Morgül, Ö., & Güzelis, C. (1999). A saturated linear dynamical network for approximating maximum clique. *IEEE Trans. Circuits and Syst.—I*, *46*(6), 677–685.

Pelillo, M. (1995). Relaxation labeling networks for the maximum clique problem. *J. Artif. Neural Networks*, *2*, 313–328.

Pelillo, M. (1999). Replicator equations, maximal cliques, and graph isomorphism. *Neural Computation*, *11*(8), 2023–2045.

Pelillo, M. (2002). Matching free trees, maximal cliques, and monotone game dynamics. *IEEE Trans. Pattern Anal. Machine Intell.*, *24*(11), 1535–1541.

Pelillo, M., & Jagota, A. (1995). Feasible and infeasible maxima in a quadratic program for maximum clique. *J. Artif. Neural Networks*, *2*, 411–420.

Pelillo, M., Siddiqi, K., & Zucker, S. W. (1999). Matching hierarchical structures using association graphs. *IEEE Trans. Pattern Anal. Machine Intell.*, *21*(11), 1105–1120.

Ramanujam, J., & Sadayappan, P. (1988). Optimization by neural networks. *Proc. IEEE Int. Conf. Neural Networks* (pp. 325–332). Piscataway, NJ: IEEE Press.

Rosenfeld, A., Hummel, R. A., & Zucker, S. W. (1976). Scene labeling by relaxation operations. *IEEE Trans. Syst. Man and Cybern.*, *6*, 420–433.

Samuelson, L. (1997). *Evolutionary games and equilibrium selection*, Cambridge, MA: MIT Press.

Shrivastava, Y., Dasgupta, S., & Reddy, S. (1990). Neural network solutions to a graph theoretic problem. In *Proc. IEEE Int. Symp. Circuits Syst.* (pp. 2528–2531). Piscataway, NJ: IEEE Press.

Shrivastava, Y., Dasgupta, S., & Reddy, S. M. (1992). Guaranteed convergence in a class of Hopfield networks. *IEEE Trans. Neural Networks*, *3*, 951–961.

Takefuji, Y., Chen, L., Lee, K., & Huffman, J. (1990). Parallel algorithms for finding a near-maximum independent set of a circle graph. *IEEE Trans. Neural Networks*, *1*, 263–267.

Wang, R. L., Tang, Z., & Cao, Q. P. (2003). An efficient approximation algorithm for finding a maximum clique using Hopfield network learning. *Neural Computation*, *15*, 1605–1619.

Weibull, J. W. (1995). *Evolutionary game theory.* Cambridge, MA: MIT Press.

Wu, J., Harada, T., & Fukao, T. (1994). New method to the solution of maximum clique problem: Mean-field approximation algorithm and its experimentation. *Proc. IEEE Int. Conf. Syst., Man, Cybern.* (pp. 2248–2253). Piscataway, NJ: IEEE Press.

Received January 27, 2005; accepted September 9, 2005.