# DNA Computing Capabilities for Game Theory

David Harlan Wood (wood@cis.udel.edu)*.
*Department of Computer and Information Sciences*
*University of Delaware, Newark, DE 19716*

**Abstract.** Problems in game theory can be used for benchmark DNA computations. Large numbers of game strategies and chance events can be assembled into finite state machines. These many machines perform, in parallel, distinct plays of a game. Strategies can be exposed to selection and breeding. The computational capabilities of DNA are matched with aspects of game theory, but the most interesting problems are yet to be treated.

## 1. Introduction

Our main objective is to draw attention to DNA computing capabilities, rather than particular applications. Capabilities illustrated with game theory examples can potentially inspire new applications of DNA computing. At minimum, such examples can serve as benchmarks.

We wish to draw attention to the capabilities of DNA, and to leave enterprising applications to the future. Historically, computational capability has usually preceded significant applications by years. For example, the ENIAC computer preceded payroll and inventory applications and the desktop computer preceded spreadsheets and the Internet.

## 2. Game Theory

In a game[1], players make finite sequences of choices restricted only by a set of rules. Players receive payoffs depending on their choices and the choices of others, including chance events. A game strategy must provide decisions for every possible game situation. A strategy may use deterministic decisions (a pure strategy) or, more generally and more powerfully, probabilistic decisions (a mixed strategy).

### 2.1. Finding Adaptive Strategies for Games

DNA computing can be useful for seeking strategies that maximize expected payoffs. Throughout this paper, payoff means expected payoff obtained by averaging over all chance events and all strategies involved. In particular,

* This research was partially supported by NSF Grants Nos. 0130385 and 9980092, and DARPA/NSF grant No. 9725021

the strategies we seek depend on the strategies of the other players, who have no incentive to reveal them. Clearly, this is a difficult problem. As for definite procedures for finding good strategies for all games, none are known that can consistently outperform simple enumeration. Several characterizations of strategies are known to be $NP$-hard, even for symmetric two-person games(Conitzer and Sandholm, 2002).

### 2.1.1. *A Simple Three-Person Poker Game*

We use an example game (Nash and Shapley, 1950) to introduce some definitions and illustrate the nature of probabilistic (mixed) strategies. The game rules are given below.

1. Each of the three players starts by contributing *a* euros to the pot.
2. Each player is dealt a hand consisting of one card, with high and low cards being equally probable.
3. The players take turns in rotation.
4. The game ends if all players pass or when one player has bet (putting *b* euros into the pot) and each of the other players have chosen to call (putting *b* euros into the pot) or to fold (no additional cost).
5. Antes are retrieved if all players have passed.
6. Otherwise, the pot is divided equally among the highest hands of all players who have not folded.

As simple as this game is, it is known that good strategies must judiciously bluff (holding a low hand but not passing) and slow-play (holding a high hand but initially passing) (Nash and Shapley, 1950). Bluffing and slow-play avoid predictability that could be exploited by opponents. Such strategic misrepresentations are usually required in games involving private information (games of imperfect knowledge). In games of this type, including three-person poker, no deterministic strategy can outperform strategies that mix misrepresentation with apparent transparency.

### 2.1.2. *Games in Extensive Form*

The extensive form (Kuhn, 1953), or game tree, of the above game is shown in Fig. 1. It should be remarked that Fig. 1 shows only part of the full game tree. The root is shown at the top and has eight edges descending from it. The eight edges correspond to 2 2 2, 2 2 A, 2 A 2, 2 A A, A 2 2, A 2 A, A A 2, and A A A. There is one edge from the root for each possible combination of cards held by the three players. One edge from the root is shown in the

figure as a dotted line. Each of the edges from the root connects to a subtree identical to the one shown in Fig. 1.

From a player's point of view, the subtree in Fig. 1 is a complete characterization of the game. A player merely makes decisions while progressing down the subtree. During this progress the player knows his or her card but does not know the cards of the other players. Upon reaching a terminal node of the tree, payoffs can be calculated from the last two rules of the game.

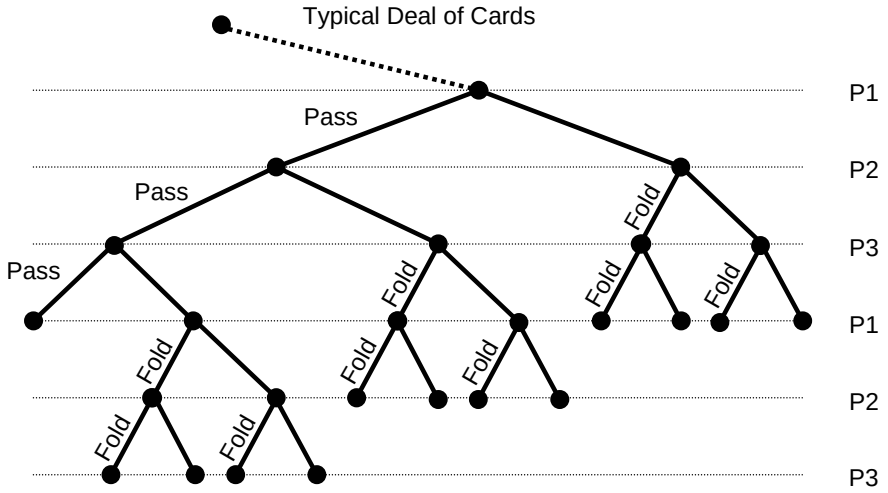


*Figure 1.* The game tree (extensive form) of 3-person poker.

2.1.3. *Mixed Strategies in Behavioral Form*

In three-person poker, Player 2 has four decision nodes. These are where the lines labeled P2 intersect the game tree in Fig. 1. Each of the other players has four decision nodes, too. Each different decision node (sometimes called an information set) corresponds to a distinct situation—a different history of prior choices by the other players. To find good strategies one adjusts the probabilistic choices to be made at each decision node.

A mixed strategy in behavioral form (Kuhn, 1953) is simply a set of probabilistic decision criteria, one for each decision node. In three-person poker, decisions depend on whether a high or low card is held. That is to say, for each decision node a mixed strategy consists of a high-card probability of passing and a low-card probability of passing. It is essential that probabilities are determined in advance, but decisions are only made during the play of the game. Since decisions are made knowing what card is held, such a strategy

can be used on any of the eight branches of the game tree. That is to say that only the part of the game tree shown in Fig. 1 is relevant to a strategy in behavioral form. We assume payoffs are computed at the end of the game.

#### 2.1.4. *Mixed Pure Strategies*

A pure strategy is a fixed series of choices taken at decision nodes. For example, in three-person poker the sixteen pure strategies of a player come from all $2^4$ combinations of two-way choices to be made at four decision nodes.

A *mixed* pure strategy, often just called a mixed strategy, is a probabilistic rule for choosing some one pure strategy to play a game. It is essential here that the probabilistic rule for selecting among the pure strategies is found in advance. This rule selects a single deterministic pure strategy before the game is played. If the game is played again, another probabilistic choice among the pure strategies is made.

#### 2.1.5. *Behavioral and Mixed Strategies are Equivalent*

Mixed strategies in behavioral form and mixed pure strategies are equivalent and interconvertible, if we assume total recall of all prior choices made playing a game (Kuhn, 1953).

### 2.2. NASH EQUILIBRIA OF GAMES

It is the celebrated result of Nash (VanDamme, 1991) that for any game with only a finite number of pure strategies, there exist mixed strategies in equilibrium. When strategies are in equilibrium, no individual player is able to improve his or her expected payoff by changing strategy. For many games Nash equilibria give important insights; for other games Nash equilibria are more problematic[2].

There is a special advantage to zero-sum games, which includes poker. For these games, all Nash equilibrium strategies are interchangeable and have the same expected payoff[3]. For zero-sum games, there are methods for finding one Nash equilibrium efficiently (Koller and Pfeffer, 1997). This can be done in time polynomial in the number of nodes in the game tree, even in the worst case[4].

### 2.3. SEEKING ADAPTIVE STRATEGIES

Nash equilibria yield valuable insights into the possible strategies for a game. However, for all but the simplest poker variants, these strategies are apparently unknown or unplayed. Even if they were known, equilibrium strategies are indifferent to exploiting the mistakes of other players[5]. Given this fact, the goal of competitive players is evolving strategies that maximize expected payoffs against the strategies they encounter.

### 2.4. Examples of Computing Adaptive Poker Strategies

Example of adaptable poker strategies are given in the examples cited[6] and compared in Table 7. The first four examples in Table 7 use adaptive mixed strategies in behavioral form. That is, strategies make probabilistic decisions as a play of the game proceeds. The decision criteria in these strategies adapt to the (mostly) fixed strategies of opponents.

The last three examples in Table 7 cite highly simplified poker games. They all involve bluffing, however. An interesting aspect is that *populations* of strategies are used . Depending on the example, the strategies in the populations can be of pure, mixed, or behavioral type.

## 3. DNA Suitability for Encoding and Playing Games

This section matches some of the computational needs of game theory with capabilities of DNA computing (Garzon and Deaton, 1999). We often invoke illustrative poker examples, but our approaches are valid for a larger class of games. We show DNA can be used to address these aspects of game theory computations: (1) Strategies can be individually encoded, yet pair off with opponents in game tournaments, (2) Decisions discriminating among many alternatives can be made, and (3) Massive populations of strategies offer special advantages for game theory.

We avoid the details of specific DNA computing instantiations in order to focus on the underlying matches with game theory[8].

### 3.1. Finite State Machines

A player's strategy cannot play poker by itself, even if it knows what cards it is to use. A single play of a game requires one strategy from each player. The play of a game may also require a set of chance moves such as a deal of card hands. A strategy in behavioral form must act at a decision node. In computer pseudocode this can take the form shown in Fig. 2.

Let us regard this pseudocode as describing a "state." Thus, a game of poker can be played by a finite state machine (Kain, 1972). This is much less demanding that providing universal computation capability (a stored program computer with arbitrarily large memory).

An example of executing a single DNA-based finite state machine is found in Komiya et al. (2000). This machine has no decisions and ten states, of which six were demonstrated experimentally. Recently, repeated transitions in a two-state machine with four decisions is reported in Benenson et al. (2001). In principle, all input data DNA strands could be distinct, yet be processed simultaneously. There is also a project to execute many distinct

Table I. Example evolutions of strategies for some versions of poker[7].

| Game and source citation | Cards per player | Betting rounds | Ante, fold, stay, raise | Number of opponents using strategy type(s) | Versus main player and its strategy type | Convergence to strategy of type |
|---|---|---|---|---|---|---|
| Non-equilibrium poker (von Neumann and Morgenstern, 1944) | $\binom{5}{52}$ | 2 | hi or lo, lo – hi, 0, - | 1 erroneous strategy | adaptive behavioral strategy | maximizing mixed |
| Texas Hold'em poker (Billings et al., 2002) | $\binom{2}{52}$ private and $\binom{3}{52}+\binom{1}{52}+\binom{1}{52}$ public | 4 | all vary | 6-10 various strategies | adaptive behavioral strategy | "strong player" behavioral |
| Simplified Hold'em poker (Barone and While, 1998) (Barone and While, 1999) | $\binom{2}{52}$ private + $\binom{5}{52}$ public | 1 | ?, 0, 1, - | 1 adaptive behavioral plus 8 tight/loose while passive/aggressive | adaptive behavioral strategy | maximizing behavioral |
| Five-card draw poker (Kendall and Willdig, 2001) | $\binom{5}{52}$ with $\binom{2}{52}$ replacement | 3 | 0, 0, 5 levels 5 levels | 4 tight/loose while while passive/aggressive | adaptive behavioral strategy | maximizing pure |
| No-draw, high-low poker (Gintis, 2000, §3.16) | $\binom{1/2}{2}$ | 1 | 1, 0, varies, varies | 1 population of pure strategies | population of pure strategies | unique pure equilibrium |
| One-card, 2 round poker (Gintis, 2000, §4.11) | $\binom{1/2}{2}$ | 2 | 2, 0, 2, 2 | 1 population of mixed pure strategies | population of mixed pure strategies | unique mixed equilibrium |
| Simplified poker (Wood et al., 2001a) | $\binom{1/2}{2}$ | 1 | 0, 0, 1, 1 | 1 DNA population of behavioral strategies | DNA population of behavioral strategies | co-evolution to maximizing? |

```
110
120       // This node is labeled
130
140  Node(k) :
150
160       // Knowing the hand of cards held, and knowing the
170       // unique game history leading to this kth node,
180
190     GO TO Node(k+1) UNLESS
200     HistoryDependentRule(k) APPLIED TO Hand
210     IS SIMILAR TO Criterion(k)
220        THEN GO TO Node(k+1)
230
240       // Where Node(k+1) and Node(k+2) are found on this
250       // DNA game strand (in another player's strategy)
260
```

*Figure 2.* Computer pseudocode for a decision node of a behavioral strategy.

acyclic (tree structured) finite state machines of two different types simultaneously (Wood et al., 2001a). One type is a four-state machine with two "fuzzy logic" decisions. The other type is a three-state machine with one "fuzzy logic" decision. An experiment simultaneously executed $3^{15} \approx 10^7$ distinct machines of the second type, achieving all three states.

For playing games we could use any method for implementing finite state machines using DNA — as long as it provides a `GO TO` command[9] (and decisions, if we want to use strategies in behavioral form). Because of the `GO TO` command, any DNA encoded strategy of an individual player can physically cluster its decision nodes. This allows finite state machines with a complete set of game nodes to be assembled from one DNA strand per player.

### 3.2. Fuzzy Logic for Discriminating Decisions

Clearly, strategies in behavioral form need to take actions contingent on their assessment of the current situation. In poker, for example, each decision node will have to judge whether the prior history and a particular hand of cards warrants a bid. In particular, a mere stand of DNA has to discriminate among the millions of possible poker hands. Yet this turns out to be not unreasonable.

Probabilistic binding of near-complementary DNA strands is inherently a fuzzy decision (Deaton and Garzon, 2001). Fuzzy logic (Klir and Folger, 1988) methods provide for making definite decisions in situations that are not fully characterized. This is typical of games of imperfect information. For example, in poker the cards held by other players are not known.

3.2.1. *Discrimination via Near-Complementary DNA Sequences*
In natural systems, DNA is almost exclusively found in double stranded form. Each strand is a directed sequence of the bases A, C, G, and T. Naturally occurring double strands are bound together by mutual attractions of each A to a T and each C to a G. One easy way to separate complementary strands is to boil them. But as they cool, they will seek each other out to recombine. But all DNA sequences may bind to each other to a greater or lesser extent, or even to parts of themselves[10].

There are many single DNA strands with, say, 25 bases, namely $4^{25} \approx 10^{15}$ distinct sequences. Let us concentrate on some one sequence. Out of all the other strands, only one is complementary to our sequence. But many, many other strands are near-complements and can bind to our strand to a greater or lesser extent[11]. For example, there are $\binom{1}{25}3^1 + \binom{2}{25}3^2 + \binom{3}{25}3^3 + \binom{4}{25}3^4 + \binom{5}{25}3^5 + \binom{6}{25}3^6 = 109,366,867$ distinct strands having one to six mismatches to our strand. That is, we can use our one sequence to discriminate among very many near-complements. This can be useful for implementing fuzzy sets for DNA computing.

3.2.2. *An Example of Discrimination*
Figure 3 shows one approach to discriminating by using partial DNA binding in poker strategy strands [12]. A fixed decision criterion is encoded in a stretch of DNA. And a poker hand to be evaluated is encoded at the outgoing (3') end of a DNA sequence that is near-complementary to the encoded criterion. If the two strands bind well enough, it is possible for the 3' end of the hand to be extended as the complement of the sequence adjacent to the criterion strand. Otherwise, the 3' end does not get extended. Thus, the DNA sequence within the decision criterion identifies the poker hands that warrant bidding before going to the next player's turn. It should be remarked that each game node decision usually has different decisions to make. Because of its unique prior game history, each node can have its own distinct decision criterion.

3.2.3. *Self-Programming DNA by Selection*
Once poker hands are encoded, how do we know what DNA sequences to use for the decision criteria? Some experimentation is required[13]. In particular, considering strands of length 25 was only illustrative. When we want a criterion to match similar situations to a greater or lesser extent, we can use training by examples (Wood et al., 2001a). This is a standard approach to programming fuzzy decisions in conventional computers (Herrera et al., 1994; Pena-Reyes and Sipper, 2001).

Programming fuzzy decisions by training is an important advantage. To design near-complementary sequences that would probabilistically bind according to arbitrary specifications would be a daunting task. Here we take a much simpler approach. We simply test, in parallel, a huge sample of all

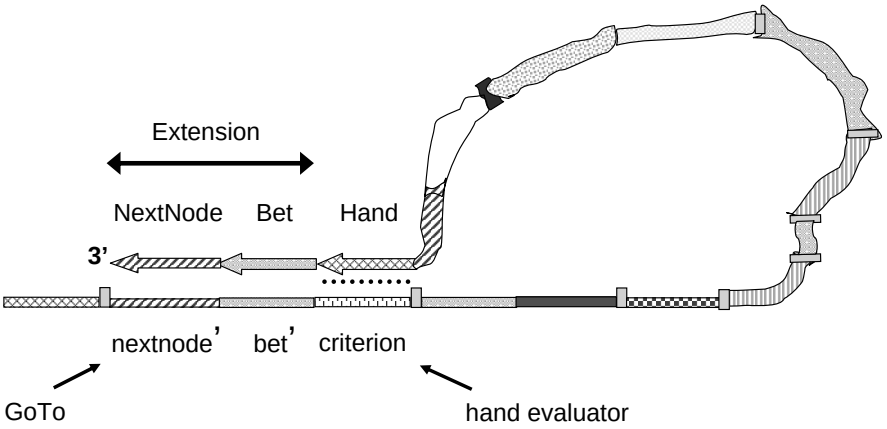


*Figure 3.* If the hand binds well enough with the criterion, a bet is made before proceeding to a decision node of the next player. This corresponds to lines 200–220 of the pseudocode in Fig. 2.

possible sequences. Those that give the desired results are selected; the remaining sequences are discarded. This kind of training has yet to be fully demonstrated using DNA, but results for two similar cases are encouraging (Wood et al., 2001a; Bi et al., 2002).

Correctly trained fuzzy decisions will presumably produce the same average payoffs as crisp probabilistically triggered decisions[14].

#### 3.2.4. *Complexity of Encoding Fuzzy Decisions*

Let us estimate the space complexity (number of DNA bases) for each decision in a strategy. An example would be making decisions based on a player's poker hand and the prior game history. A DNA strand segment containing

$$b = O(\lceil \lg(d)/2 \rceil) \tag{1}$$

bases is the absolute minimum to discriminate among $d$ possible alternatives. Two illustrative examples (Alspach, 1998) are 5-card poker hands, where $\lg(\binom{5}{52}/2) = \lg(2,598,960/2) \approx 11$ bases, and 7-card poker hands, where $\lg(\binom{7}{52}/2) = \lg(133,784,560/2) \approx 14$ bases. Note that Eq. 1 gives only an order estimate of the number of bases, while the numerical values given are the theoretical minimums.

### 3.3. Encoding Game Theory Strategies

On one hand, DNA can encode a strategy in behavioral form using fuzzy logic. Decisions could be made by probabilistic binding of dealt hands that are near-complements to decision criteria. Notice that a strategy in behavioral form can be encoded using only one DNA sequence[15].

On the other hand, a population of DNA stands could encode a mixture of pure strategies. Probabilistic selection of pure strategies could be implemented by the relative frequencies of individual pure strategy strands (Nash, 1996). Notice that encoding a mixed pure strategy requires an entire population of DNA strands; one strand is not enough.

The previous discussions of this section have indicated how a strategy in behavioral form can be encoded in a DNA strand. The number of decisions in a behavioral strategy is a fraction of the depth of the tree. Hence, the length of a behavioral strategy strand $s$ is

$$|s| = O(bd), \tag{2}$$

where $b$ is defined in Eq. 1 and $d$, is the depth of the game tree.

### 3.4. Exploiting Massive DNA Population Sizes

The following considerations are important because a population evolved to convergence theoretically needs to have only one distinct strategy in behavioral form. We may want to reduce population sizes for several reasons. The total amount of DNA available[16] constrains the number of games that can simultaneously be played. If too many distinct strategies are present, it may not be possible to form all possible strategy pairings. Smaller populations also permit game tournaments using more chance events, and also permit interacting subpopulations.

Prior to being fully evolved, larger populations allow greater diversity. This helps to explore the space of candidate strategies. This is a universal theme in evolutionary computation: trading off exploitation against exploration. In DNA computing, evaluating an entire population of candidates costs no more than evaluating a single candidate. So one might as well use a large population because it may help and no extra cost is incurred.

However, large populations have an extra advantage when seeking good game strategies. A population of strategies can, on the average, have optimal performance at equilibrium—performance better than any one individual strategy in the population. In his thesis, Nash (1996) pointed this out for populations of pure strategies. This observation remains true for a population of strategies in behavioral form because one is optimizing over a less constrained domain. A pure strategy is, after all, a special case of a strategy in behavioral form—it just happens to always make the same choices.

To put it colloquially, the final outcome of a game tournament can not distinguish whether strategies in a population are many and stupid (pure strategies), unique and smart (behavioral strategies), few and mediocre (incompletely evolved behavioral strategies), or all of the above.

3.4.1. *Size and Diversity Tradeoffs*

For simplicity we will discuss the case of two players. Modifications allowing more players and sampling of populations will be obvious. In this section, strategies will always refer to behavioral strategies.

First, some notation. For any collection $X$ of DNA strands, Let $\lfloor X \rfloor$ denote the number of distinct strands. Let $|X|$ denote the total number of strands in $X$. Let us assume that each of two game players has a population of DNA encoded strategies, calling them $P_1$ and $P_2$.

We address two fundamental issues: (1) the total quantity of DNA in an experiment is limited, and (2) in a tournament, each distinct strategy should be paired with a competing strategy. The relationships

$$\lfloor P_1 \rfloor \leq |P_1| \quad \text{and} \quad \lfloor P_2 \rfloor \leq |P_2| \tag{3}$$

easily yield expressions for the quantities of DNA used when a tournament includes all parings of strategies:

$$\begin{aligned} \max(\lfloor P_1 \rfloor, \lfloor P_2 \rfloor)^2 &\leq \min(\max(|P_1|, \lfloor P_2 \rfloor)^2, \max(\lfloor P_1 \rfloor, |P_2|)^2) \\ &\leq \max(|P_1|, |P_2|)^2. \end{aligned} \tag{4}$$

The three expressions in Eq. 4 express three squared quantities of DNA to consider. Given one such a quantity as a goal, the sizes of the populations can be adjusted. First, decrease the size of both populations, if necessary, to meet the goal by deleting duplicate strategies. Second, increase the size of the smaller population, if necessary, to meet the goal by adding duplicate strategies. This guarantees that every distinct strategy in either population will have an opponent to pair with. Of the three candidate goals from Eq. 4, the largest one has the advantage that it can be met by modifying only one of the two populations. It may or may not be necessary to modify both populations to meet the other two candidate goals. This can be checked in advance using the expressions in Eq. 4.

3.4.2. *Techniques for Addressing Population Redundancy*

To use the guidance of the previous section, we need techniques to measure and adjust the diversity of a DNA population, *P.*

*Census Taking in DNA Populations* We give a very rough description of one approach to estimating $\lfloor P \rfloor$, the number of distinct DNA strategy strands in a population. We assume the DNA is single stranded and encoded so that no strategy strand is the complement of another.

Synthesize a census-taker population of single stranded DNA that consists of the complements of all possible strategy strands. This population is synthesized uniform randomness in the decision criteria regions, but is complementary in the remaining regions. When the two populations are mixed,

we measure the number of double strands and the number of single strands[17]. Any single strand has to be either an irrelevant census-taker or a redundant strategy. The subpopulation of double strands contains one copy of each distinct strategy strand.

*Inventory Control in DNA Populations* The protocol of the previous paragraph can be continued to obtain a controlled inventory, containing a limited number of each distinct strategy strand. It is simply a matter of extracting the double strands from the mixture[18].

*Increasing Redundancy in a Population* We assume strategy strands can be copied using PCR (Garzon and Deaton, 1999). Conventional PCR uses two primer sequences and obtains exponential increase by repeated doubling of strategy strands. Linear PCR uses one primer sequence and obtains repeated linear increases of strategy strands.

*The Intersection of Two Populations* Suppose we want to obtain strategies that are common to two populations. First, make an additional population consisting of the complements of the strands in the first population. When these complements are mixed with the second population, double strands may be formed. Each double strand contains one of the strategies that occur in both populations. Extract the double strands. Finally, separate the strategy strands from the complementary strands.

*Fuzzy Intersection of Two Populations* For clarity, we have greatly simplified our presentation, neglecting practical issues of multiple copies of strands, populations that we can only sample, strands that bind without being perfectly complementary, etc. For example, in the procedure in the previous paragraph many double strands can form that are only partially bound because they are only near-complementary. Lower temperatures can aggravate this complication. The desired separation can be approached by using two-dimensional temperature gradient gel electrophoresis (2d TGGE), because 2d TGGE uses a continuum of temperatures[19]. The DNA in a 2d TGGE gel is physically smeared out according to the meaningful but indistinct criterion of being near-complementary. This is an example of sorting using a fuzzy ordering. Conveniently, we can set boundaries on which parts of the gel we extract, interpreting this as an adjustably less stringent version of intersection.

### 3.5. Tracking Multiple Interacting Subpopulations

If relatively small populations of distinct strands suffice, surplus DNA capacity can be used to simultaneously evolve multiple populations, even interacting populations. Each strategy might contain some marker. The effect

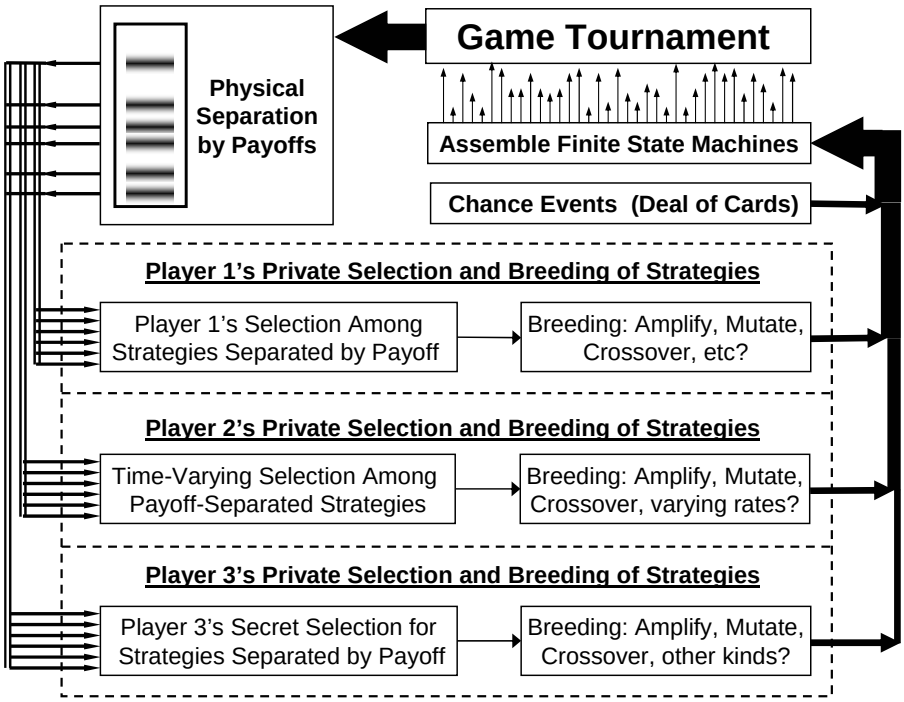


*Figure 4.* This is an outline of evolutionary computation of strategies. Beginning on the right hand side and proceeding counterclockwise, we have (1) Assembly of finite state machines from strategies and chance events, (2) A tournament of many simultaneous plays of the game, (3) Clustering game histories by payoffs, (4) Returning players' strategies separated by payoffs obtained, and (5) Private individualized selection and breeding by each player.

of the marker could be to inhibit tournament parings with strands bearing different markers. Any degree of inhibition could be used, from none (indiscriminate) to some (clannish) to total (speciation). This permits tracking *entire distributions* of evolutionary outcomes.

## 4. Evolution of Game Strategies via Learning

The prior section of this paper addressed some aspects of game theory and how they could be matched with DNA computing capabilities. These aspects were (1) Game playing by finite state machines, (2) Fuzzy logic for decisions that discriminate, (3) Encoding mixed strategies in behavioral form.

This section shows how each player can try to improve strategies by learning from experience. An evolutionary computation (Bäck et al., 1997) is outlined in Fig. 4. We are able to program most or all of the usual variants and elaborations of selection by fitness and breeding using mutation and crossover.

### 4.1. Preparation for a Tournament

Each player pits his or her strategies against those of other players. On the right hand side of Fig. 4 we assemble finite state machines by randomly linking pairs of strategy strands, one from each of the players. Each pair is then linked to one of an assortment of chance events, for example dealt poker hands. Each resulting finite state machine is a single strand of DNA which contains chance events and one strategy for each player.

### 4.2. A Game Tournament

In a tournament (shown in the upper right hand corner of Fig. 4) all the finite state machines are executed in parallel. In principle, a milligram of DNA would suffice for a tournament with about $10^{15}$ individual plays of a game.

The result of a tournament is that each DNA strand encodes a finite state machine plus the entire history of one play of the game, including the payoff. A game history corresponds to one path from the root of the game tree (like the one in Fig. 1) to a terminal node. We emphasize how tournaments enable learning from experience. If it is of interest, a winner of the tournament can be determined by counting the game history strands in each of the payoff categories.

### 4.3. Parallel Fitness Evaluations by Payoffs

In the upper left corner of Fig. 4 we see that a game tournament is followed by separating DNA game history strands according to payoffs. One possibility is that the history strands are of different lengths according to which player won and how much money was won. A physical clustering by length is indicated, but other payoff encodings and separation techniques could be used. Each cluster of history strands is kept separate from the others; each cluster is processed in the same way. Within each cluster, strategies are cleaved from the history strands. Each player's strategies from that cluster are separately extracted and returned to that player.

This tournament design stringently suppresses some information: a player is not able to tell what situations their various strategies experienced. Other tournament designs would be possible, of course.

#### 4.3.1. *Player Strategies Grouped by Payoffs*

In this way each player receives, for each possible payoff (including losses), all of his or her strategies that achieved that payoff. That is to say, the strategies of this player have been physically separated according to their payoffs. Their strategy strands are returned to them with the rest of the play histories removed—but the returned strategies are grouped by payoffs. Players will want their strategies to adaptively learn from experience. Grouping gives

a measure of strategy fitnesses that can be used by evolutionary learning techniques.

### 4.4. Players' Private Strategy Evolutionary Techniques

Strategies separated by payoffs undergo evolution. As Fig. 4 indicates at the bottom, each player uses his or her unique private evolutionary approach[20]. A wide variety of combinations of these techniques and others[21]. are used in evolutionary computation (Bäck et al., 1997).

Each evolutionary approach has two parts in Fig. 4. First is selection: who will die and who will live to breed. Second is breeding: mutation, recombination, cloning, etc. Players freely choose their own private techniques for selection and breeding.

Most, if not all, of the aspects of evolutionary computation can be carried out in the laboratory on populations of DNA strands (Chen and Wood, 2000; Chen et al., 2000; Wood et al., 2001b; Vartanian et al., 1996). Thus, the laboratory can perform evolutionary computation in a programmable manner[22].

#### 4.4.1. *Evolution is Tolerant of Errors*

We see that DNA competing is especially suitable for evolutionary computation. But we also assert that evolutionary computation is especially suitable for DNA computing (Stemmer, 1995; Chen and Wood, 2000). Evolutionary computation is population oriented and inherently parallelizable. It is also designed to be tolerant of error and self-correcting. This is important because DNA laboratory manipulations have variability in their outcomes. Another characteristic of evolution is adaptation to changing circumstances. Co-evolution of strategies is just one venue profiting this robustness.

### 4.5. Other DNA Approaches to Evolution of Strategies

One particularly attractive alternative DNA computing approach would use continuous evolution (Ackermann et al., 1999; Schmitt and Lehman, 1999; Breaker et al., 1994). The virtue of this approach is that no external intervention is required. After all ingredients are combined, the system seeks an equilibrium. There is, however, a severe hurtle. The lack of intervention appears to greatly reduce the available vocabulary of computing commands.

## 5. Supplementary DNA Computing Techniques

Biochip readouts and microflow reactors are two supplementary resources for DNA computing.

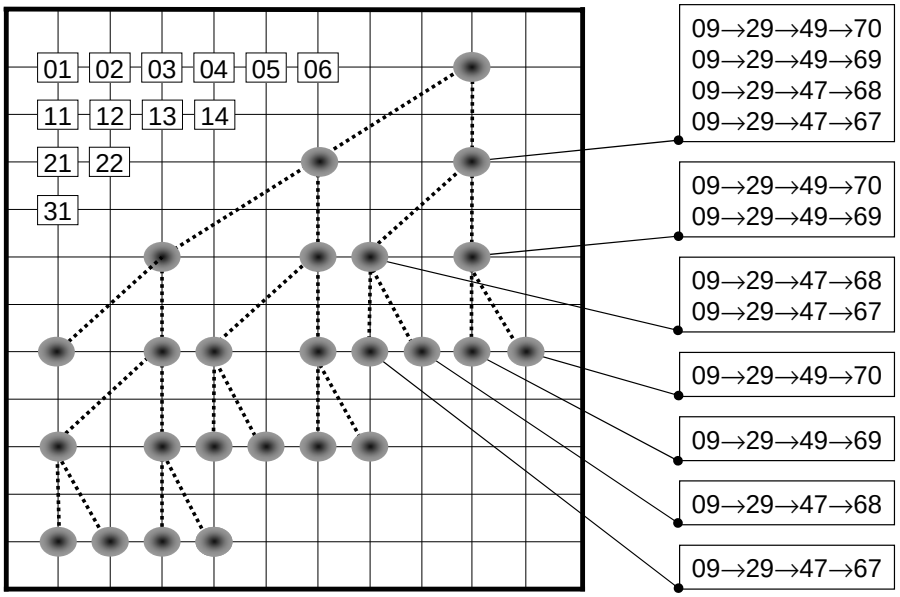


*Figure 5.* Many of the spots on this $10 \times 10$ biochip are unused. The spots encoding edges leading to the decision nodes of 3-person poker have been chosen to create a version of Fig.1. The dotted lines are superimposed for clarity. The numbering scheme for the spots and tree nodes is indicated in the upper left corner of the chip. For selected spots the right side of the figure shows what game histories will be found on that spot.

### 5.1. BIOCHIP READOUT AND SELECTION

There is a technique for observing the choices made at decision nodes by using a DNA biochip readout[23].

Let us require that each edge in the game tree corresponds to a unique preassigned DNA encoding, the complement of an "edge label" sequence. And let us require that each game history strand contains the complements of the labels for all of the edges traversed. For each edge in the tree, prepare a short, so-called "marker," a DNA strand containing one half of its edge label. This marker will be attached to what is known as a quantum-dot bead (Han et al., 2001) having its own unique color spectrum. We call these "colored q-beads." Han et al. (2001) estimates 10,000 distinguishable colored q-beads could be fabricated.

The mixture of all edge markers can be added to any population of game histories. Then the collection is allowed to hybridize to a DNA microarray. The microarray contains the other half of the edge label sequences, each at a known location. The q-beads are excited, and their emitted light is collected as a colored image. Figure 5 is a compressed version of the game tree for 3-person poker. Each spot on the chip represents and edge shows the colors of all edges that are part of some game history that includes this spot's edge.

Biochips with up to 10,000 spots can be fabricated (Wurmbach et al., 2001). However, a $10 \times 10$ grid of spots is available on a commercial biochip (Nanogen, 2002) on which desired sequences can be installed. Ten-thousand spots chips of this type are said to be possible (Heller, 2001).

#### 5.1.1. *Reading Out Decision Node Probabilities*

What do we see at a spot on the biochip having the label of a terminal node[24]? Every terminal node's spot contains only the colors of the q-beads labeling the edges connecting that terminal node back to the root. If a terminal node has $k$ siblings, its parent node will have $k-1$ additional colors. The parent node will contain the q-beads of its children in proportion to the frequencies of the choices made at the parent node.

Such frequencies can be obtained at any node in the following way. Given a particular node, concentrate on the colors corresponding to its children nodes. At this node, each of the children's colors is in proportion to the choices made at this node. The same proportional distribution of the children's colors is also found at the children nodes[25].

#### 5.1.2. *Selectable Recovery of Strategies from Biochips*

With certain biochips it is possible to recover the DNA from a particular spot (Mills et al., 2001). This capability is also available in commercial biochips with one hundred spots (Heller et al., 1998). For refining populations of strategy strands, biochips of this type are particularly useful selection tools.

### 5.2. MECHANIZING DNA COMPUTING

Ultimately DNA computing will be mechanized. That is, the DNA computer of the future will be a machine, not a biolaboratory. This will be essential to reduce errors, increase reproducibility, and facilitate multiple generations of evolutionary computation. Standard laboratory processes would likely have to be mechanized if DNA is to be used in more than milligram quantities.

There are special purpose microfluidic devices that already are starting to be used for DNA computing (van Noort et al., 2002; McCaskill, 2001; McCaskill et al., 2000)[26]. At first it might seem that "micro" is the wrong direction to go because we may want to scale up to processing larger quantities of DNA. But consider the history of electronic computers. The miniaturization and standardization of electroninc components have greatly facilitated the construction of powerful computers.

## 6. Summary of DNA Computational Capabilities

Evolutionary computation techniques, with or without using DNA, can seek game strategies maximizing expected payoffs against current adversaries. Exploiting selection while also exploring variation lets strategies evolve through learning from experience. Evolutionary computation is self-programming. That is to say, we only need to evaluate candidate solutions and do not need to provide details on how they are programmed. Evolutionary computation is designed to be robust under change or uncertainty. This is important since strategies need to adapt as opponents evolve their strategies. Evolutionary computation can be tolerant of errors. This is an advantage when using DNA laboratory techniques. DNA can implement evolutionary computation using selection and diversification. This includes essentially all selection schemes based on payoffs. Laboratory reproduction can include variable rates of mutation and crossover; selection can be varied according to payoffs; etc.

Inclusion of labels in DNA strategies means a DNA biochip can be used for readout. Configurations similar to the game tree in Fig. 1 are available. Variations in readout intensities report decisions made by populations of strategies. Furthermore, subpopulations can be extracted from selected spots for further processing. Microflow reactors are becoming the building blocks of future DNA computers.

We have presented several capabilities of DNA computing. DNA techniques are capable of (1) Forming finite state machines, (2) Implementing fuzzy decisions, (3) Encoding game strategies, (4) Staging massive game tournaments, (6) Physically separating game outcomes by payoff while preserving players' privacy, (5) Programming evolutionary computation using a variety of selection and breeding strategies, (6) Tolerating errors and adapting to change, (7) Reading out details of game histories using a DNA biochip, as well as extracting certain subsets, and (8) Mechanizing DNA computers.

## 7. Challenges in Game Theory

DNA computing capabilities, even the ones we have discussed, may find their most interesting applications outside of game theory. Nevertheless, applications of DNA computing to some problems in game theory may lead to further capabilities, insights, or inspiration.

Poker is too simple to fully stress computational capabilities. Likely, the most challenging problems in game theory will involve the dynamic competitive co-evolution of strategies for *n*-person non-zero sum games[27]. Analytic and semi-analytic computer-aided techniques have not yet fully illuminated such problems. Simulation can be used because of its generality, but it heavily taxes resources and interpretation.

The main advantages of DNA computing are the massive parallelism and information storage available. If we are seeking behavioral strategies, we do not at first know their encodings. They are just points in a gigantic search space. The space of all encodings of DNA strands with 100 variable bases is about $10^{60}$. Evolutionary computation typically maintains an evolving population of candidate strategies. The optimal population size depends on the problem being solved, and is usually not known in advance.

Some situations may grant us surplus population capacity. In these cases we can simultaneously simulate the evolutions of multiple populations. These populations may also be allowed to interact to an adjustable extent, as was discussed in Section 3.5. For problems of this type, we could obtain entire distributions of evolutionary dynamics. This could, at minimum, bring DNA computing speed and storage abreast or ahead of existing computers. And further scaling up is possible.

In this paper we have matched DNA computational capability with aspects of game theory. Simple poker games have been used for illustrative purposes only. Game theory problems could be used to benchmark the unprecedented capabilities of DNA computing. Hopefully such benchmarks will inspire even more interesting applications of DNA computing whether in game theory or another problem area.

## Notes

[1] A classic reference on game theory is (Osborne and Rubinstein, 1994). A recent book (Gintis, 2000) especially emphasizes evolving game strategies, as we do in this paper.

[2] Nash equilibria for arbitrary games have two significant drawbacks. First, and most fundamental, no general tractable general method for finding Nash equilibria is known. The existence of such a method is a noted open question (Papadimitriou, 2001). Second, Nash equilibria need not be unique or even discrete. Consider the three-person poker game given above. Nash and Shapley (1950) have shown that various ratios of anteing cost to betting cost determine equilibria that are (1) unique or (2) depend on one parameter or (3) depend on two parameters. Even for two-person games, merely determining if there is more than one Nash equilibrium is intractable (Gilboa and Zemel, 1989). Also, counting the total number of maximal connected sets of Nash equilibria is $\#P$-hard (Conitzer and Sandholm, 2002, Corollary 8).

[3] Guaranteed interchangeability assumes there are only a finite number of pure strategies. Three-person poker provides a counterexample (Nash and Shapley, 1950, p. 114).

[4] This is a significant improvement over prior techniques which were *exponential* in the number of nodes. Computation of one Nash equilibrium for a simplified poker with about $10^5$ nodes has been demonstrated. It is a sobering fact that five-card draw poker has about $10^{25}$ nodes in its game tree (Koller and Pfeffer, 1997).

[5] There is a famous analysis of a two-person simplified poker game (von Neumann and Morgenstern, 1944, Chapter 19). (This game is is cited in the top line of Table7.) The analysis demonstrates that bluffing is necessary for a player's "good strategy" (what we call a Nash

equilibrium strategy). The immediately following section (von Neumann and Morgenstern, 1944, §19.10.3) is perhaps too seldom cited:

> "Incorrect bluffing causes no losses against an opponent playing the good strategy; but the opponent could inflict losses by deviating appropriately from the good strategy. . . . hence no permanently optimal strategy exists there."

[6] There is an excellent overview of computer poker up to 1995 (Billings, 1995). More recent research has been summarized elsewhere (Kendall and Willdig, 2001).

[7] Notations like $\binom{5}{52}$ indicate that each player receives 5 cards from a deck of 52. We abuse this notation by using $\binom{1/2}{2}$ to indicate one of 2 players is dealt one of 2 cards.

[8] Some details of one particular approach to DNA implementation can be found in Wood et al. (2001a).

[9] The power of branching programs, which use only labels and `Go TO` commands is discussed by Winfree (1998) in the context of whiplash PCR. This interesting paper observes that at a given node, whiplash PCR can also extract data, particularly addresses of previously unreachable nodes. This implements we he calls write-once branching programs.

[10] This is a big and sophisticated topic. See, for example, Wetmur (1997), for an introduction and then go on to Rose et al. (2001), and SantaLucia (1998).

[11] This varies with temperature. Roughly speaking, having fewer mismatches allows partial binding at higher temperatures, but the location of mismatches also has an effect.

[12] This approach is a variant of that found in Wood et al. (2001a).

[13] Often, experiments can simulated (Garzon and Oehmen, 2002).

[14] It is less clear how the variance of the two approaches might compare (fuzzy decisions are said to be less "brittle").

[15] Many copies of any particular DNA strand are needed for reliable laboratory procedures, of course. We will always assume this requirement is met.

[16] Standard laboratory techniques can process about a milligram of DNA. A milligram of DNA contains about $10^{18}$ DNA bases,which is roughly the same amount of data as a 1999 estimate of the size of the Internet (Lawrence and Giles, 1999)

[17] Using adsorption of UV light, for example.

[18] For example, single and double strands have different mobilities in gel electrophoresis.

[19] A similar application of 2d TGGE to DNA computing is found in Chen et al. (2000).

[20] Evolutionary computations can be designed in many ways. Aspects include selection according to fitness and breeding using mutation and recombination (exchanging blocks of structure, also called crossover). Additional issues include relative quantification of fitness, criteria for selection, including allowing the elite (most fit) strategies to survive unchanged, and many other aspects inspired by biological analogies.

[21] We add to these the possibility of adjusting the redundancy in a population, as discussed in subsection 3.4.2. Also, since an ideal strategy can successfully cope with all game situations, we might want to favor versatile strategies. We could seek these by taking fuzzy intersections of the sets of strategies that have been separated by payoffs.

[22] Our computer is programmed by changing laboratory protocols without substantially replacing the DNA involved. Contrast this with "stored program computers" which are programmed by computer instructions encoded in DNA. We have a hybrid system. Our finite state machines are stored program (and data) type computers, but our evolutionary computations are not directed by stored programs.

[23] This could also be done with biochip of universal design (but larger size) for reading out arbitrary graphs that are encoded in DNA in a special way. (Wood et al., 2003).

[24] We are now saying a spot represents a node, instead of saying an edge. Since each edge leads to a unique node, there is no contradiction.

[25] For a specific example, consider node 29 in fig. 5 having children nodes 47 and 49. The probabilities of branching left or right from node 29 are proportion to the quantities of colors 47 and 49 within node 29. They are also proportional to the quantities of colors 47 and 49 within nodes 47 and 49.

[26] Of course, present day robots can carry out standard processes by manipulating materials in the lab. Microfluidic devices offer specialized operations in an modular construction.

[27] For zero-sum games with only a finite number of pure strategies, all Nash equilibria have the same payoffs. This is not the case for non-zero games. Payoffs can be highly sensitive. Suppose you are playing a high-payoff Nash equilibrium strategy. Your opponent may play a non-equilibrium strategy that is slightly suboptimal for him or her but is disastrous for you. Gintis (2000), §4.30, discuses a simple example.

## Acknowledgements

I am grateful to my colleague Junghuei Chen for many very stimulating discussions of DNA computing.

## References

Ackermann, J., B. Wlotzka, and J. S. McCaskill: 1999, '*In vitro* DNA-based predator-prey system with oscillatory kinetics'. *Bulletin of Mathematical Biology* **60**(2), 329–354.

Alspach, B.: 1998, 'Poker computations'. http://www.math.sfu.ca/ alspach/computations.html.

Bäck, T., D. B. Fogel, and Z. Michalewicz (eds.): 1997, *Handbook of Evolutionary Computation*. Philadelphia: Institute of Physics Publishing.

Barone, L. and L. While: 1998, 'Evolving Adaptive Play for Simplified Poker'. In: *1998 IEEE International Conference on Computational Intelligence (ICEC '98)*. IEEE publications, pp. 108–113.

Barone, L. and L. While: 1999, 'An Adaptive Learning Model for Simplified Poker Using Evolutionary Algorithms'. In: *1999 Congress on Evolutionary Computation (CEC '99)*. IEEE publications, pp. 153–160.

Benenson, Y., T. Paz-Elizur, R. Adar, E. Keinan, Z. Livneh, and E. Shapiro: 2001, 'Programmable and autonomous computing machine made of biomolecules'. *Nature* **414**(1), 430–434.

Bi, H., J. Chen, R. Deaton, M. Garzon, H. Rubin, and D. H. Wood: 2002, 'A PCR-based protocol for *in vitro* selection of non-crosshybridizing oligonucleotides'. In (Hagiya and Ohuchi, pear), Springer-Verlag.

Billings, D.: 1995, 'Computer Poker'. M.Sc. Research Essay.

Billings, D., L. Peña, J. Schaeffer, and D. Szafron: 2002, 'The challenge of poker'. *Artificial Intelligence* **134**(1-2), 201–240. Special Issue on Games, Computers and Artificial Intelligence.

Breaker, R. R., A. Banerji, and G. F. Joyce: 1994, 'Continuous *in vitro* evolution of bacteriophage RNA polymerase promoters'. *Biochemistry* **33**(39), 11980–11986.

Chen, J., E. Antipov, B. Lemieux, W. Cedeño, and D. H. Wood: 2000, '*In vitro* selection for a OneMax DNA Genetic Algorithm'. In: D. Gifford and E. Winfree (eds.): *DNA Based*

*Computers V: DIMACS Workshop, June 14-15, 1999*, Vol. 54 of *DIMACS series in Discrete Mathematics and Theoretical Computer Science*. Providence, pp. 23–37, American Mathematical Society.

Chen, J. and D. H. Wood: 2000, 'Computation with biomolecules'. *Proceedings of the National Academy of Sciences, USA* **97**(4), 1328–1330. Commentary.

Conitzer, V. and T. Sandholm: 2002, 'Complexity results about Nash equilibria'. Technical Report CMU-CS-02-135, Carnegie Mellon University Computer Science Department, 5000 Forbes Avenue Pittsburgh, PA 15213.

Deaton, R. J. and M. H. Garzon: 2001, 'Fuzzy logic with biomolecules'. *Soft Computing* **5**(1), 2–9. Special issue on Biomolecular Approaches to Soft Computing.

Garzon, M. and R. Deaton: 1999, 'Biomolecular computing and programming'. *Transactions on Evolutionary Computation* **3**(3), 236–250.

Garzon, M. and C. Oehmen: 2002, 'Biomolecular computation in virtual test tubes'. In (Jonoska and Seeman, 2002), pp. 117–128, Springer.

Gilboa, I. and E. Zemel: 1989, 'Nash and correlated equilibria: Some complexity considerations'. *Games and Economic Behavior* **1**, 80–93.

Gintis, H.: 2000, *Game Theory Evolving*. Princeton, NJ: Princeton University Press.

Hagiya, M. and A. Ohuchi (eds.): to appear, 'DNA Computing, 8th International Workshop on DNA-Based Computers, DNA8, Hokkaido University, Japan, June 10-13, 2002, Revised Papers', Vol. 2568 of *Lecture Notes in Computer Science*. Berlin: Springer-Verlag.

Han, M., X. Gao, J. Z. Su, and S. Nie: 2001, 'Quantum-dot-tagged microbeads for multiplexed optical coding of biomolecules'. *Nature Biochemistry* **19**, 631–635.

Heller, M., C. Edman, and S. Esener: 1998, 'Electric field assisted self-Assembly of DNA structures: A potential nanofabrication technology'. *Sixth Foresight Conference on Molecular Nanotechnology*.

Heller, M. J.: 2001, 'Microelectronic array devices: New designs for DNA diagnostic, biomedical research, and pharmacogenomic applications'. In: *Lab-on-a-Chip and Microarrays*. Newton Upper Falls, MA. abstract only.

Herrera, F., M. Lozano, and J. Verdegay: 1994, 'Generating fuzzy rules from examples using genetic algorithms'. *Proc. IPMU'94 (5th Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems)* pp. 675–680.

Jonoska, N. and N. Seeman (eds.): 2002, 'DNA Computing: 7th International Workshop on DNA-Based Computers, DNA7, Tampa, FL, USA, June 10-13, 2001. Revised Papers', Vol. 2340 of *Lecture Notes in Computer Science*. Springer.

Kain, R. Y.: 1972, *Automata Theory: Machines and Languages.* New York: McGraw Hill.

Kendall, G. and M. Willdig: 2001, 'An Investigation of an Adaptive Poker Player'. In: *Australian Joint Conference on Artificial Intelligence*. pp. 189–200.

Klir, G. and T. Folger: 1988, *Fuzzy sets, uncertainty and information*. Prentice Hall International.

Koller, D. and A. Pfeffer: 1997, 'Representations and Solutions for Game-Theoretic Problems'. *Artificial Intelligence* **94**(1–2), 167–215.

Komiya, K., K. Sakamoto, H. Gouzo, S. Yokoyama, M. Arita, A. Nishikawa, and M. Hagiya: 2000, 'Successive state transitions with I/O interface by molecules'. In (Winfree and Gifford, 2000), pp. 21–30, American Mathematical Society.

Kuhn, H. W.: 1953, 'Extensive games and the problem of information'. In: H. W. Kuhn and A. W. Tucker (eds.): *Contributions to the Theory of Games II*, Annals of Mathematical Studies. Princeton, NJ: Princeton University Press, pp. 193–216.

Lawrence, S. and C. Giles: 1999, 'Accessibility of information on the web'. *Nature* **400**, 107—109.

McCaskill, J. S.: 2001, 'Optically programming DNA computing in microflow reactors'. *Biosystems* **59**(2), 125–138.

McCaskill, J. S., R. Penchovsky, M. Gohlke, J. Ackermann, and T. Rucker: 2000, 'Steady flow micro-reactor module for pipelined DNA computation'. In (Winfree and Gifford, 2000), pp. 239–246, American Mathematical Society.

Mills, Jr., A. P., M. Turberfield, A. J. Turberfield, B. Yurke, and P. M. Platzman: 2001, 'Experimental aspects of DNA neural network computation'. *Soft Computing* **5**(1), 10–18. Special issue on Biomolecular Approaches to Soft Computing.

Nanogen: 2002, 'NanoChip Microarray'. http://nanogen.com/products/nanochip_micro.htm.

Nash, Jr., J. F.: 1996, 'Motovation and interpretation'. In: J. F. Nash, Jr. (ed.): *Essays on Game Theory*. Cheltenham, UK: Edward Edgar Publishing, Limited, pp. 31–33. Previously unpublished appendix from Ph. D. Thesis.

Nash, Jr., J. F. and L. S. Shapley: 1950, 'A simple three-person poker game'. In: H. W. Kuhn and A. W. Tucker (eds.): *Contributions to the Theory of Games I*, Annals of Mathematical Studies. Princeton, NJ: Princeton University Press, pp. 105–116.

Osborne, M. and A. Rubinstein: 1994, *A Course in Game Theory*. Boston: MIT Press.

Papadimitriou, C. H.: 2001, 'Algorithms, games, and the Internet'. In: *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing: STOC'01, July 6–8, 2001, Hersonissos, Crete, Greece*. New York, pp. 749–753, ACM Press.

Pena-Reyes, C.-A. and M. Sipper: 2001, 'Fuzzy CoCo: Balancing accuracy and interpretability of fuzzy models by means of coevolution.'. *IEEE Transactions on Fuzzy Systems* **9**(5), 727–737.

Rose, J. A., R. J. Deaton, M. Hagiya, and A. Suyama: 2001, 'The fidelity of the tag-antitag system'. In (Jonoska and Seeman, 2002), pp. 138–149, Springer.

SantaLucia, Jr., J.: 1998, 'A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics'. *Proceedings of the National Academy of Sciences, USA* **95**(4), 1460–1465.

Schmitt, T. and N. Lehman: 1999, 'Non-unity molecular heritability demonstrated by continuous evolution in vitro'. *Chemistry and Biology* **12**(6), 857–869.

Stemmer, W. P. C.: 1995, 'The Evolution of Molecular Computation'. *Science* **270**, 1510–1510.

van Noort, D., F.-U. Gast, and J. S. McCaskill: 2002, 'DNA computing in microreactors'. In (Jonoska and Seeman, 2002), pp. 128–137, Springer.

VanDamme, E.: 1991, *Nash Equilibrium*. New York: Springer-Verlag.

Vartanian, J. P., M. Henry, and S. WainHobson: 1996, 'Hypermutagenic PCR involving all four transitions and a sizeable proportion of transversions'. *Nucleic Acids Research* **24**(14), 2627–2631.

von Neumann, J. and O. Morgenstern: 1944, *Theory of Games and Economic Behavior*. Princeton, NJ: Princeton University Press.

Wetmur, J. G.: 1997, 'Physical chemistry of Nucleic acid hybridization'. In: H. Rubin and D. H. Wood (eds.): *Preliminary Proceedings of the Third Annual Workshop on DNA Based Computers, held at the University of Pennsylvania, June 23-25, 1997*. pp. 1–14.

Winfree, E.: 1998, 'Whiplash PCR for $O(1)$ computing'. In: L. Kari, H. Rubin, and D. H. Wood (eds.): *Preliminary Proceedings of the Fourth Annual Workshop on DNA Based Computers*. Piscataway NJ, pp. 175–188, DIMACS. Available from http://www.dna.caltech.edu/~winfree/old_html/Papers/whiplash.ps.

Winfree, E. and D. K. Gifford (eds.): 2000, 'DNA Based Computers V: DIMACS Workshop, June 14-15, 1999', Vol. 54 of *DIMACS series in discrete mathematics and theoretical computer science*. Providence: American Mathematical Society.

Wood, D. H., H. Bi, S. O. Kimbrough, D. Wu, and J. Chen: 2001a, 'DNA starts to learn poker'. In (Jonoska and Seeman, 2002), pp. 92–103, Springer.

Wood, D. H., J. Chen, E. Antipov, B. Lemieux, and W. Cedeño: 2001b, 'A design for DNA computation of the OneMax problem'. *Soft Computing* **5**(1), 19–24. Special issue on Biomolecular Approaches to Soft Computing.

Wood, D. H., C. L. T. Clelland, and C. Bancroft: 2003, 'Universal biochip readout of directed Hamiltonian path problems'. In (Hagiya and Ohuchi, pear), Springer-Verlag.

Wurmbach, E., T. Yuen, B. J. Ebersole, and S. C. Sealfon: 2001, 'Gonadotropin-releasing hormone receptor-coupled gene network organization'. *Journal of Biological Chemistry* **276**(50), 47195–47201.