# Coevolutionary Optimization of Fuzzy Logic Intelligence for Strategic Decision Support

Rodney W. Johnson, Michael E. Melich, Zbigniew Michalewicz, Martin Schmidt

*Abstract*— **We present a description and initial results of a computer code that coevolves fuzzy logic rules to play a two-sided zero-sum competitive game. It is based on the TEMPO Military Planning Game that has been used to teach resource allocation to over 20,000 students over the past 40 years. No feasible algorithm for optimal play is known. The coevolved rules, when pitted against human players, usually win the first few competitions. For reasons not yet understood, the evolved rules (found in a symmetrical competition) place little value on information concerning the play of the opponent.**

## I. Introduction

The notion of big decisions, those that shape the future evolution of a business or organization, frequently attaches to the word *strategic*. For example, what a company chooses to do or avoid doing is shaped by its answer to the strategic question: *Are we a consulting company or a product company?* Or in the case of the US Navy the question has taken the form: *Are we an organization that provides prompt and sustained operations at sea, or are we operators of ocean-going naval combatants?*

The allocation of the people, capital, goodwill, and other assets will be different depending upon which choice is made. Product companies often expect to derive revenue streams from a developed set of loyal customers whose needs are understood through ongoing contact that informs new development and leads to sales of subsequent generations of products. Product companies usually see themselves lasting many product generations and hope to leverage their collective skills to increase margins. Consulting companies often are not tied to given products and find themselves working in a fee for service arrangement on their client's problem of the moment. Constantly finding new problems (and new clients) for the next engagement grows to dominate the marketing effort and tends to limit the profitability to the profit on hourly charges. Long-lived consultancies tend to attach to problems such as "accounting standards," the "tax code," and other complexes of regulation and custom.

How are strategic questions to be answered? First, any analysis will ask if there are customers and competitors, and how are they described. Second, whether the company can profitably obtain and serve the customers in the face of the existing and potential competition? Third, what should be done and in what order to become successful? Easily stated questions — but difficult to answer. Even describing what constitutes a good choice or set of choices is complicated by the tens to thousands of different actions that could be taken. In larger organizations that have had time to evolve in response to competitive and environmental pressures, the allocation of effort — the decision — is most explicitly presented in the budget. But budgets tend to describe inputs to the organization's activities and not the outputs. Businesses fall back on measures of profitability over some time period as the measure of their success. Military organizations look to wars and conflicts to characterize their success. Thus, analyzing strategic questions can be cast as asking: Will a particular sequence of investments, expressed as budgets, over many years produce a successful result in the face of competition and a changing environment?

Resource allocation in mission- or market-oriented large enterprises, either government departments or large businesses, is made difficult by the large number of possible investment plans that could be considered. This complexity is in addition to the normal uncertainties associated with a changing environment — changing competition, technical innovation, etc. For example, within the US Department of Defense it is not uncommon for tens of thousands of different categories to be examined annually. Decisions are then made to allocate funds and personnel for the forthcoming budget

Rodney W. Johnson is with Wayne E. Meyer Institute of Systems Engineering, Naval Postgraduate School, Monterey, CA 93943, e-mail: rwjohnso@nps.edu.

Michael E. Melich is with Wayne E. Meyer Institute of Systems Engineering, Naval Postgraduate School, Monterey, CA 93943, e-mail: melich@alumni.rice.edu.

Zbigniew Michalewicz is with the School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia, and Institute of Computer Science, Polish Academy of Sciences, ul. Ordona 21, 01-237 Warsaw, Poland, and Polish-Japanese Institute of Information Technology, ul. Koszykowa 86, 02-008 Warsaw, Poland, e-mail: zbyszek@cs.adelaide.edu.au

Martin Schmidt is with SolveIT Software Pty Ltd. PO Box 3161, Adelaide, SA 5000, Australia, and holds an adjunt research fellow position at the School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia, e-mail: martin.schmidt@solveitsoftware.com.

year as well as projections for six years in the future. Similar activities and associated complexities are found in non-governmental organizations [21].

In the early 1960s, the Department of Defense created a management system, the Planning, Programming, and Budgeting System (PPBS) of considerable complexity to rationalize its resource allocation problems. A major training program was instituted to teach the PPBS and a "game" was created by General Electric's "TEMPO think tank" to train people in the use of the new system. The Defense Management Resource Institute (DRMI) (see www.nps.navy.mil/drmi/98org.htm) has used the TEMPO game in its courses for nearly 40 years. Over 20,000 students from 125 countries have benefited from exposure to this game.

We became interested in resource allocation problems while conducting large scale, multi-nation "futures" studies. Our studies used scenario methods [30]. An integral part of the multi-year competitive decision environment was the allocation of national resources to defense. This forced trade-offs between investment in economic growth, foreign assistance, education, etc. This is a very complex set of decisions and we soon realized that we were trimming a very complex decision tree and had little hope of understanding what other options might offer. The work presented here reports one facet of our research program, initiated in 2000, to deal with aspects of resource allocation problems in a world where your competitors are also able to make choices.

## II. Coevolutionary Approaches to Games

Games are characterized by rules that describe the moves each player can make. These moves constitute the behavior of the players: the manner in which each allocates his resources. When a player makes a move, he receives a payoff; usually he tries to maximize the cumulative payoff over a period of time. In some games, such as chess, the payoff comes at the end of the game, but we can imagine a surrogate payoff, or evaluation function, that correlates with a player's chances of winning at each point in the course of the game.

Some games are competitive, others are cooperative, and still others are mixed, depending on the form of the evaluation function. If, for example, one player is gaining in payoff and the other player is losing payoff, it's a competitive game.

The evaluation function is a key ingredient in a game-playing system. Sometimes, however, we have no idea of how to create a good evaluation function; there may be no clear measure of performance beyond simply whether you win, lose, or draw.

As indicated in [23], the situation is similar to that of living creatures in nature, who are consummate problem solvers, constantly facing the most critical problem of avoiding being someone else's lunch. Many of their defensive and offensive survival strategies are genetically hard-wired. But how did these strategies begin? We can trace similarities across many species. For example, many animals use cryptic coloration to blend into their background. They may be only distantly related, such as the leafy sea dragon and the chameleon, and yet their strategy is the same: don't be noticed. Other animals have learned that there is "safety in numbers," including schooling fish and herd animals such as antelope. Furthermore, herding animals of many species have learned to seek out high elevations and form a ring looking outwards, so as to sight predators as early as possible. These complex behaviors were learned over many generations of trial and error, and a great deal of life and death.

This is a process of coevolution. It is not simply one individual or species against its environment, but rather individuals against other individuals, each competing for resources in an environment that itself poses its own threats. Competing individuals use random variation and selection to seek out survival strategies that will give them an edge over their opposition. Antelope learned to form a ring to spot predators more quickly; predators learned to hunt in teams, and use the tall grasses of the savanna to mask their approach. Each innovation from one side may lead to an innovation from another, an "arms race" wherein individuals evolve to overcome challenges posed by other individuals, which are in turn evolving to overcome new challenges, and so forth.

Note that the individual antelopes did not gather in a convention to discuss new ideas on survival and come up with the strategy of defensive rings on high ground. Nevertheless, the development of strategies is unmistakably a process of learning. When instinctual, they have been accumulated through random variation and selection, with no evaluation function other than life and death. The entire genome of the species is then the learning unit, with individuals as potential variations on a general theme.

It is not surprising that coevolutionary processes have been used by many researchers, whether in optimization or in game playing.

An example in optimization is Hillis's now famous example of minimizing a sorting network: a fixed sequence of operations for sorting a fixed-length string of numbers [15]. By an evolutionary search he had found a network that sorted 16 numbers with just 65 comparisons. Networks were scored on the fraction of all test cases (unsorted strings) that they sorted correctly. Hillis then noted that many of the sorting tests were

too easy and only wasted time. He therefore devised a method in which two populations coevolved: sorting networks and sets of test cases. The networks were scored according to the limited number of test cases presented (10 to 20), and the test sets were scored on how well they found problems in the networks. Variation and selection were applied to both populations; the test cases became more challenging as the networks improved. Hillis reported that the coevolutionary approach avoided stalling at local optima, and that it eventually found a network comprising only 61 comparisons. (This is just 1 short of the best network known to date, discovered by Green and using 60 comparisons [19].)

Sebald and Schlenzig studied the design of drug controllers for surgical patients by coevolving a population of so-called "CMAC" controllers, chosen for effectiveness, against a population of (simulated) patients, chosen for presenting difficulties [31]. Many researchers have studied pursuit-evasion games, for example [29], [6], [9]. Various interesting approaches to constraint-satisfaction problems are reported in [26], [27], [24], [20]. With a bit of thought, what would appear to be a straightforward optimization problem can often be recast with advantage as a problem of coevolution.

In 1987 Axelrod studied the Iterated Prisoner's Dilemma (IPD) by an evolutionary simulation [2]. Strategies were represented as look-up tables giving a player's move — cooperate or defect — as a function of the past 3 moves (at most) on each side. Strategies competed in a round-robin format (everyone plays against every possible opponent) for 151 moves in each encounter. The higher-scoring strategies were then favored for survival using proportional selection, and new strategies were created by mutation and by one-point crossover. Axelrod made two observations. First, the mean score of the survivors decreased in the early generations, indicating defection, but then rose to a level indicating that the population had learned to cooperate. Second, many of the strategies that eventually evolved resembled the simple but effective strategy of "tit-for-tat" — cooperate on the first move, and then mirror the opponent's last move.

In 1993 Fogel studied the effect of changing the representation of strategies in the IPD, replacing Axelrod's look-up tables with finite state machines [10], [11]. The results were essentially the same as what Axelrod had observed. Harrald and Fogel, on the other hand, observed entirely different behavior in a version of IPD where the player could make moves on a continuous numeric scale from $-1$ (complete defection) to 1 (complete cooperation) [14]. Strategies were represented by artificial neural networks. In the vast majority of trials payoffs tended to decrease, not increase. Darwen and Yao observed similar results in a variation of IPD with eight options, rather than two or a continuous range [7]. They observed, first, that as the number of options to play increases, the fraction of the total game matrix that is explored decreases. Second, when the IPD had more choices, strategies evolved into two types, where the two types depended on each other for high payoffs and did not necessarily receive high payoffs when playing against members of their own type.

These observations are very interesting, yet they perhaps do not fully explain the degradation in payoff that is seen when a continuous range of options is employed. Hundreds of papers about the prisoner's dilemma are written each year, and very many of the contributions to this literature have involved evolutionary algorithms in different forms. These and many other studies indicate the potential for using coevolutionary simulation to study the emergence of strategies in simple and complex games.

In the late 1990s and into 2000, Chellapilla and Fogel [3], [4], [5] implemented a coevolutionary system that taught itself to play checkers at a level on par with human experts. The system worked like this. Each position was represented as a vector of 32 components, corresponding to the available positions on the board. Components could take on values from $\{-K, -1, 0, +1, K\}$, where $K$ was an evolvable real value assigned to a king, and 1 was the value for a regular checker. A 0 represented an empty square, positive values indicated pieces belonging to the player, and negative values were for the opponent's pieces. The vector components served as inputs to a neural network with an input layer, multiple hidden layers, and an output node. The output value served as a static evaluation function for positions — the more positive the value, the more the neural network "liked" the position, and the more negative, the more it "disliked" the position. Minimax was used to select the best move at each play based on the evaluations from the neural network.

The coevolutionary system started with a population of 15 neural networks, each having its weighted connections and $K$ value set at random. Each of the 15 parent networks created an offspring through mutation of the weights and $K$ value, and then the 30 neural networks competed in games of checkers. Points were awarded for winning $(+1)$, losing $(-2)$, or drawing (0). The 15 highest-scoring networks were selected as parents for the next generation, with this process of coevolutionary self-play iterating for hundreds of generations. The networks did not receive feedback about specific games or external judgments on the quality of moves. The only feedback was an aggregate score for a series of games.

The best neural network evolved (at generation

840) was tested by hand, using the screen name "Blondie24," against real people playing over the Internet in a free checkers website. After 165 games, Blondie24 was rated in the top 500 of 120,000 registered players on the site. The details of this research are in [3], [4], [5], [12].

There are $10^{20}$ possible positions in checkers, far too many to enumerate, and checkers remains an unsolved game: nobody knows for sure whether the game is a win for red, a win for white, or a draw. Chess, at $10^{64}$ positions, is still further from being solved. But Fogel and Hays have combined neural networks and coevolution to create a grandmaster-level chess-playing program, again without giving the simulated players any feedback about specific games [13].

Coevolution can be a versatile method for optimizing solutions to complex games, and a reasonable choice for exploring for useful strategies when there is little available information about the domain.

## III. The TEMPO game

The TEMPO Military Planning Game is a two-sided zero-sum competitive game. Teams of players compete in building force structures by dividing limited budgets, over a succession of budgeting periods ("years") between categories such as "acquisition" and "operation" of "offensive units" and "defensive units." The rules are no more complex than the rules of, say, Monopoly. However, players learn that the rules' apparent simplicity is deceptive: they pose challenging and difficult decision problems. No feasible algorithm for optimal play is known.

The full set of investment categories for the TEMPO game comprises: (1) operation of existing forces, (2) acquisition of additional or modified forces, (3) research and development ("R&D"), (4) intelligence, (5) counter-intelligence.

There are four types of forces: two offensive ("Offensive A and B") and two defensive ("Defensive A and B"). Each type comprises several weapon systems with varying acquisition and operation costs (measured in "dollars"), measures of effectiveness (in "utils"), and dates of availability (in "years"). A team's objective is to maximize its total "net offensive utils." A team's net offensive utils of type A are the total utils for its operating Offensive A units, minus the opposing team's Defensive A, but not less than zero, and likewise for type B. Thus there is no advantage in investing more in a defensive system than is necessary to counter the opponent's offensive systems of the same type. A team's total net offensive utils are the sum of its net offensive utils of types A and B.

R&D is current investment that buys the possibility in a future year of acquiring new weapon systems, possibly with better price/performance ratios than those now available. Investment in intelligence buys information about the opponent's operating forces and investment activities. Investment in counter-intelligence degrades the information the opponent obtains through intelligence.

Every year the probability of war (PWar) is announced. When this is low, players may well decide to invest heavily in R&D and acquisition of new units; when it is high, they may prefer to concentrate on operating existing units.

## IV. Initial experiments

In 2000 we performed experiments aimed at seeing whether with evolutionary methods we could obtain reasonable players for a TEMPO-like game. Being mindful of the usual tradeoff between programming convenience and execution speed, we began with a Lisp implementation, using Koza's Simple Lisp Code for Genetic Programming [18] before deciding to invest substantial programming effort. We used a very rudimentary version of the TEMPO game. There were only one offensive and one defensive weapon system ("OA1" and "DA1"). R&D was eliminated; however the operating and acquisition costs of the systems could vary from year to year. Intelligence and counter-intelligence were also eliminated, but each player was given the opponent's current inventory of the two weapon systems. A game ended with the outbreak of war, or after a specified number of years (typically 10). Finally, utils were equated with units, i.e., the util values were set at 1 per unit. This meant that operating and acquisition costs were effectively given in dollars per util, and operating and acquisition decisions could be made with a granularity of 1 util.

Koza's Simple Lisp Code is a generational, tree-based genetic-programming kernel written in Common Lisp. Individuals (candidate algorithms) are represented as computer programs in a simple Lisp-like language, written in terms of user-specified terminals (constants and variables) and functions. In addition to (1) the set of terminals and (2) the set of functions, the user must specify (3) the fitness measure, (4) a set of fitness cases, (5) a termination condition, and (6) a set of GP parameters such as population size and probability of mutation.

For the rudimentary TEMPO game, the terminals were random floating-point constants and variables describing the current state of the game. State variables included the current total available budget, PWar, current acquisition limits, prices, and operating costs for the offensive and defensive units, and both the player's and the opponent's current inventories of these units.

The function set included operations that attempt

to allocate funds for the coming budgeting period to acquisition and operation of the offensive and defensive units. For example (AcOA1 $u$) allocates funds to acquiring at most $u$ units of "Offensive A1," subject to constraints: the number of units is a nonnegative integer, total expenditures do not exceed the available budget, and total units acquired do not exceed the acquisition limit for OA1. Arguments $u$ that attempt to violate these constraints incur a penalty; for example, if the requested number of units would exceed the acquisition limit, the player receives only the allowed number of units, but the budget is still decremented by the total cost of the number requested. Besides these TEMPO-specific operations, the function set included the elementary arithmetic operations $(+, -, \times, \div)$ and two general programming constructs: (if3 $n$ $x$ $y$ $z$) evaluates $n$ and then, depending on whether the result is negative, zero, or positive, evaluates and returns the value of $x$, $y$, or $z$; and (progn3 $x$ $y$ $z$) evaluates its three arguments in order and returns the value of the last.

Investment algorithms were evaluated for fitness by pitting each in games against a selection of others from the same population and adding up penalties according to the result: 0 for a win, 1/2 for a draw, and 1 for a loss. For simplicity, the fitness was $1/(1+F)$, where $F$ is the sum of the penalties.

A fitness case consisted of initial inventories of the two weapon systems, the maximum number of game years, and initial values, rates of increase or decrease, and volatilities for the budget, PWar (actually the corresponding odds) and the acquisition costs, acquisition limits, and operating costs of the weapon systems. (These latter parameters were updated from year to year within each game by random factors drawn from lognormal distributions determined by the corresponding rates of change and volatilities.) Six fitness cases were defined, and each player was evaluated by one game (each with a different opponent) for each fitness case.

The termination criterion for evolution was simply reaching the specified number of generations.

The main GP parameters for the run reported here were: population 12,000, number of generations 100, and the following probabilities for reproduction methods: crossover 0.89, copying 0.10, and mutation 0.01. Other parameters included method of selection (fitness-proportionate) and method of generation (ramped half-and-half, see [18] for the definition).

The question was whether anything reasonable would emerge in such a simple framework. And indeed, starting from an initial generation of completely random programs, an algorithm was evolved that allocated funds according to rudimentary sensible rules, which can be characterized as "dumb, but not crazy." If the budget and inventories are adequate, it is equivalent to:

```
(OPOA1 OA1INV)
(ACDA1 DA1ACLIM)
(OPDA1 DA1INV)
(ACOA1 OA1ACLIM)
```

Here OP means "operate," AC means "acquire," INV is "inventory," and ACLIM is "acquisition limit." Thus the algorithm would not attempt to acquire units beyond the appropriate acquisition limits or to operate units beyond the number in inventory. The actual code was nearly 100 lines of Lisp, mostly introns, which some hand editing reduced to:

```
(OPOA1 OA1INV)
(IF3 (+ PWAR BUDGET) 0 0
   (PROGN
      (ACDA1 DA1ACLIM)
      (IF3
         (OPDA1
            (IF3 DA1INV
               DA1ACCOST
               (OPDA1 DA1ACLIM)
               DA1INV))
         0
         (OPOA1 OA1OPCOST)
         (ACOA1 OA1ACLIM))))
```

This incorporates a check to ensure that an initial allocation to operation of offensive units has not exhausted available funds (by more than a fractional dollar) before further allocations are attempted. This result took just under 10 hours to obtain on a Pentium III processor.

We did not pursue the Lisp route further. We found this last result sufficiently encouraging to proceed to a C++ program and to a more substantial subset of the TEMPO game.

## V. Design of a New System

In an attempt to improve the evolution of human readable rules we attempted to design a coevolutionary system that evolves fuzzy logic rule-bases to play the TEMPO game.

There have been two basic approaches to such evolution of readable fuzzy logic rule-bases:

- Interpretability-oriented approach, where rules are based on symbols and then the symbols are translated into crisp/numeric values for the membership functions, the inference system, and the output membership function.
- Precision-oriented approach, where rules are based on crisp numbers that define all membership functions but then the rules have to be translated into more human interpretable form. Such translation would not

take place during the fuzzy logic calculations but only to output some more human readable rules.
As stated in [8]:
"There is a well-known tradeoff between numerical accuracy and linguistic interpretability. This tradeoff is the consequence of a well-known limitation of the human brain to represent a limited number of categories on a given domain. [...] On the other hand, the numerical accuracy is very important in the implementation of policies and control actions oriented to obtain a desired result from the system. This issue of accuracy is very critical when the models are used in dynamic way, where the predicted value is fed back and the small errors will be propagated and reflected as errors in the long term prediction."
The interpretability-oriented approach evolves readable rules yet the translation to crisp numbers via membership functions has to be defined. On the other hand, the precision-oriented approach is easy for internal calculations but readable rules have to be created at the end by a translation into more human readable rules. For both approaches there are techniques to reinforce the "missing" aspect (e.g., [32], [28], [8]).

In our design, a precision-oriented approach was chosen. Hence, the evolved rules are inherently crisp and "numerical" on the genotype level, which means that the evolutionary optimization works directly on the crisp parameters. The internal calculations still use fuzzy logic for all calculations, yet the evolved membership functions are based on numerical parameters. The translation of such exact rules will be explained in details later but is inherently a uniformly spaced segmentation of input ranges into symbols like "very low," "low," "medium," "high," and "very high." These symbols are then used for humans to attempt to understand *why* a fuzzy logic rule-base performs well or not (whatever the case might be). The new system has the following features:

Each individual can encode a maximum of $w$ rules for acquiring and operating weapons (called "weapon rules") and $q$ rules for buying intelligence or counter-intelligence (called "intel rules").

The chromosome is based on the structure given in Figure 1.

There are $m = w + q$ rules altogether; each $\text{Rule}_i$ is built from several fields (Figure 1 expands $\text{Rule}_3$):
- $U_3$ is a Boolean defining whether the rule $\text{Rule}_3$ is used,
- $B_{i3}$ are Booleans defining whether input $i$ is used,
- $C_{i3}$ are centers of the Gaussian in range $[0, 1]$ for input $i$,
- $S_{i3}$ are sigmas of the Gaussian in range $[0, \infty)$ for input $i$, and

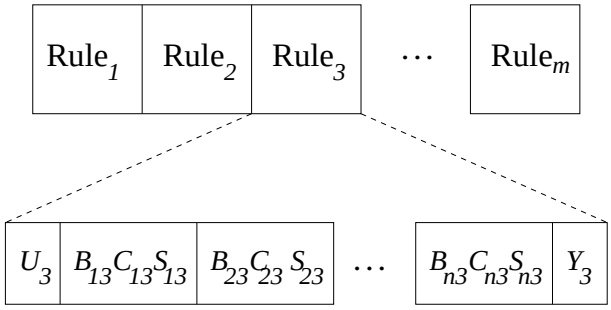


Fig. 1. A structure of a chromosome

- $Y_3$ is the output in range $[0, 1]$ of rule $\text{Rule}_3$.

There is one rule set for weapon allocations and a second similar rule set for buying intelligence. The first set has a total number of floating-point genes equal to $w(1 + 3n_w + 1)$; for the second set, the number is $q(1 + 3n_q + 1)$ (see Figure 1). The parameters of the weapon rule set used in the initial runs with fuzzy-logic system reported further in the paper were $w = 34$ and $n_w = 15$.[1] The intelligence rule set used $q = 16$ and $n_q = 7$.[2] The complete chromosome was encoded as one string of floating point numbers (booleans were represented as floating point numbers as well).

We use the Mamdani fuzzy logic system with Gaussian membership functions,[3] singleton fuzzyfier, product operation rule for fuzzy AND, and center of average defuzzyfication.

The weapon rules assign a value (a "desirability") to each weapon system; the intel rules assign a value to each intelligence/counter-intelligence category. The budget is allocated by linear scaling of these values, followed by normalization in order not to exceed the available budget.

We use two populations, X and Y, each consisting of *pop_size* individuals. This was 200 for the initial runs mentioned, with a fixed genotype length

[1]Note again, that each rule can use one or more of the available environmental parameters for weapon allocation and intelligence gathering. The environmental weapon related parameters are: probability of war, budget, weapon category (offensive or defensive), weapon type (A or B), weapon subtype (0, 1, 2, etc.), initial units available, maximum number of units available for acquisition, acquisition cost per unit, operations cost per unit, utils (i.e., effectiveness of weapon), utils per operation cost, utils per acquisition cost, year available, enemy offensive force change, and enemy defensive force change; so altogether there are $n_w = 15$ environmental weapon related parameters (maximum number of weapon inputs).

[2]The environmental intelligence related parameters are: probability of war, budget, intelligence category, offensive and defensive force change for weapon types A and B; so altogether there are $n_q = 7$ environmental intelligence related parameters (maximum number of intelligence inputs).

[3]We use one Gaussian spreading parameter for each input.

$l = 34(1 + 3 \cdot 15 + 1) + 16(1 + 3 \cdot 7 + 1) = 1,966$. The decoded phenotype has a varying number of rules and membership functions[4] with a maximum number (given by the maximum length $l$ of the chromosome).

Note that the fitness function created for the intial experiments (and discussed in section IV of the paper), was defined just as $1/(1 + F)$. It did not involve utilities; its value was determined by the number of wins, draws, and losses in a number of games. For the new system, a new fitness function was implemented that created more dragging towards improved and more compact rules. This required more gradual control by the evolutionary algorithm over the complexity of the rule-base. More precisely, the fitness $f$ of an individual is calculated as follows.

Each individual from each population is evaluated by letting it compete against $o$ randomly chosen opponents from the other population. (For the initial runs we had $o = 200$.) Thus each individual from population X plays $o$ games against opponents from Y and also has the expectation of being chosen at random about $o$ times to serve as an opponent for a player from Y. That is, a player from X plays at least $o$ games, but the expected number is $2o$ games, all of which contribute to the fitness score of X. The same is true of each player from Y. The fitness of each individual is computed from its average "net offensive utils" and average number of games won, with a penalty term that is linear in the number of parameters and rules used by the fuzzy logic system (for "pruning"). More precisely, the fitness $f$ of an individual is calculated as follows:

$$f = k \cdot r + u - p \cdot (w + w_i/n_w) - p \cdot (g + g_i/n_q),$$

where $r$ is the average number of wins (per game played), $u$ is the average net offensive utils, $w$ is the total number of weapon rules used, $w_i$ is the total number of weapon inputs used, $g$ is the total number of intelligence rules used, $g_i$ is the total number of intelligence inputs used, $n_w = 15$ is the maximum number of weapon inputs, $n_q = 7$ is the maximum number of intelligence inputs. The chosen penalty constants were $k = 10^8$, and $p = 10^6$.[5]

The components and constants included in the fitness function have the following roles:

- $k \cdot r$ should maximize the average number of games won by maximizing $r$.
- $u$ should maximize the average net offensive utils (per game).
- $p \cdot (w + w_i/n_w)$ should minimize the total number of weapon rules used $w$ and minimize the total number of weapon inputs used by weapon rules (expressed as a fraction $w_i/n_w$).
- $p \cdot (g + g_i/n_q)$ should minimize the total number of intelligence rules used $g$ and minimize the number of inputs used by intelligence rules (expressed as a fraction $g_i/n_q$).

Note that the fitness $f$ is constructed in such a way that its major component is the average number $r$ of wins; the constant $k$ is set so that simply winning is given much greater weight than sometimes winning by a large margin $u$. In other words, it is "better" to win 10 games by a small margin than winning 9 games by large margins and losing one game (even by a small margin). The constant $p$, on the other hand, is a prolixity penalty that controls the degree of parsimony pressure. Hence, more well-performing and compact rule-bases are preferred during the evolution, which is more in accordance to the well-known Ockhams Razor principle.[6] As the constant $p$ is set two orders of magnitude lower than $k$, again, winning is much more important than simplicity of rules.

The mutation operator could perform either small or large mutations of each parameter (floating point number) with a probability $p_{mut} = 0.7$. If mutation is selected then each gene has a probably of 0.5 to be mutated. If a gene is selected for mutation then there is a fixed probability of 0.1 for a "big mutation" and 0.9 for a "small mutation." A "big mutation" adds a random number in range $[-d, +d]$ to the gene, while the "small mutation" adds a random number in range $[-d/10, +d/10]$ ($d = 0.1$ is used). Notice that it might seem like overly strong mutation, but due to unexpressed parts of the chromosome which are not translated to the phenotype there are many unused intron-like segments in the chromosome.

The crossover operator is a standard two-point crossover with $p_{cross} = 0.3$.

For each population the environment changes from game to game, i.e., available weapons and effectiveness and prices change. This results in a dynamically changing environment in which the rule-bases have to make budget allocations.

Starting from random populations the coevolutionary system develops interesting fuzzy logic rule-bases. In order to get an understanding of some kind of "absolute" performance, the best-performing individual is played against a static "expert" based on simple heuris-

[4]Rules and inputs can be "pruned," i.e., be "NULL" and hence not used. Pruning reduced the effect of overlearning. The fitness reflects that a smaller rule-base is preferable using a static penalty approach.

[5]Parameter values were determined by preliminary experimentation.

[6]Of two equivalent theories or explanations, all other things being equal, the simpler one is to be preferred.

tics (expressed as fuzzy logic rules).[7] The "expert" uses the following rules:

**if** [UtilsPerOperationCost IS Very Low – Low]
**then** [Evaluation IS Very Low]
**if** [UtilsPerOperationCost IS Low – High]
**then** [Evaluation IS Medium]
**if** [UtilsPerOperationCost IS High – Very High]
**then** [Evaluation IS Very High]

In other words, the "expert" looks at the effectiveness of each weapon only and disregards any other available information. The performance against the "expert" is not included in any fitness calculations but is used to understand the quality of the evolved rule-bases during the evolution.

The rules can be presented in a form that can be understood easily by humans, which is one reason for choosing fuzzy logic. Here is an example:

RULE 1:
**if** [PWar IS Very Low – Low]
[CATEGORY IS DEFENSIVE]
[SUBTYPE IS 1 OR 2]
[Inventory IS Low]
[MaxAcquisitonUnits IS Low – Medium]
[AcquisitionCost IS Very Low]
[UtilsPerAcquisitionCost IS Very Low – Low]
**then** [Evaluation IS Low]

The terms of the "if" part refer to seven of the environment variables that are available for constructing a weapon rule. A term such as "AcquisitionCost IS Very Low" refers to the degree of membership of the acquisition cost in a certain fuzzy set represented internally by a Gaussian membership function with a given center $c$ and standard deviation $\sigma$. The program uses the actual numeric values of $c$ and $\sigma$ internally, and these are the quantities that mutation and crossover operate on. But for the human reader, expressions such as "Very Low" are presented, and are presumably more palatable than a pair of floating-point numbers.

The ranges of meaningful acquisition costs (normalized to the interval $[0, 1]$) are divided into subranges running from "Very Low" to "Very High" in the following way: "Very Low" refers to the range $[0, 1/8]$, "Low" to the range $(1/8, 3/8]$, "Medium" to the range $(3/8, 5/8]$, "High" to the range $(5/8, 7/8]$, and "Very High" to the range $(7/8, 1]$. The reason for this split-up is that the "imaginary center" for each category is placed as follows: "Very Low" is at 0, "Low" is at $1/4$, "Medium" is at $1/2$, "High" is at $3/4$, and "Very High" is at 1. Hence, the imaginary category centers have maximum distance to each other. A point in $[0, 1]$ is assigned to the category with the closest center. The human-readable output is generated using the categories of the points $c \pm \sigma$. If these are in the same category, a single label is used. In the example, the entire central part (width $2\sigma$) of the Gaussian for AcquisitionCost falls in the category "Very Low." A term such "AcquisitionCost IS Very Low – Low" would have been used if the points $c \pm \sigma$ had been in different categories.

The "Evaluation IS Low" in the "then" part of the rule refers to a "desirability" value. Again the program uses a specific number. The human reader is told that the number is in the low subrange of possible desirability values (the same category limits are used as for the "if" part).

In addition to the developed coevolutionary system, there is a game system that lets a human player play against a saved individual. The computer distributes its budget according to its rule base, while the human player interacts with the game system, currently through a spreadsheet interface.

## VI. New Experiments

Initial experience with the coevolution code immediately demonstrated the utility of the approach — it proceeded to win first games with most of those who played against the derived rules. It was also clear early on, that the coevolved rules did not value information about the opponent's choices. That is, no rules for buying intelligence or counter-intelligence were of sufficient value to be included in the evolved set. Similar behavior had been seen in the play of the TEMPO paper game when we were using it to teach our students. We attributed this either to avoidance of excessive inputs — a common human strategy for coping with information overload — or to the "gaming of the game" that occurs when you know approximately when the game will be over. Another possibility was that since the initial version of the TEMPO code provided information that was not quantitative on what the opponent was getting with the investments made, there was truly little value.

We did some preliminary investigations to determine if we could configure the game so that there might be value to buying intelligence. We gave player X a larger budget and immediate access to all weapons as they became available, while player Y had a smaller budget and was delayed one year in having investment opportunity on the various weapons. This coupled with a reduction in the prolixity penalty did produce a few

[7] We use the term "expert" as applicable to any system that incorporated rules of thumb derived by consultation with human experts, regardless of whether the system actually exhibited any particular level of expertise. Our "expert" incorporates a rule of thumb that human players find useful (e.g., "more bang per buck is better") but we do not actually claim that it a very strong player. It wins handily in the early generations, against opponents that play more or less at random, but within the first hundred generations or so, evolved players usually arise that can beat the "expert" more than half the time.

"weak" rules for the purchase of intelligence by the disadvantaged player.

We also modified the rule inputs to enhance the value of the information a player could obtain by buying intel. A rule input was provided indicating whether the opponent had bought counterintelligence. The opponent's operating forces were given in total utils rather than number of weapon units and these values were given as absolute current values, rather than as changes relative to the previous year. This last change was motivated by the fact that the rules incorporate no "memory" of previous years' decisions. Finally, an input giving the initially available number of units of a weapon system was replaced with one giving the player's current inventory.

The results of a coevolution have been used in a course at NPS, "Economics for Defense Managers." Students played the game on line through the spreadsheet interface. Many of the students needed three or four tries before achieving an outcome that they were willing to submit for grading. Thus we continue to see human-competitive play in the coevolved rules. One of our colleagues, an economist with previous experience with the DRMI paper form of the game, was able through prolonged and concerted effort to beat the machine by a small margin on a first try. During play, he was ascribing all manner of sophisticated motivations to the machine for its moves. He was dismayed to learn afterward that he had been competing against a set of precisely three rules: the one shown above in section V and the following two others.

RULE 2:
**if** [Budget IS Low – Medium]
[EnemyCounterintel IS NOT BOUGHT]
[SUBTYPE IS 1]
[Utils IS Low]
[UtilsPerAcquisitionCost IS Very High]
[YearAvailable IS Medium]
**then** [Evaluation IS Low]

RULE 3:
**if** [Budget IS Low]
[CATEGORY IS OFFENSIVE]
[TYPE IS B]
[Utils IS Very High]
[YearAvailable IS Medium - High]
[EnemyOffensiveUtils IS Unkn. OR Very Low]
[EnemyDefensiveUtils IS Unkn. OR Very Low]
**then** [Evaluation IS Very High]

Such a low number of rules is not atypical. The above three rules (RULE1, RULE2, and RULE3) constitute a complete rule-base after a completed coevolutionary run. However, it was a bit surprising that the system did *not* evolve any "intelligence rules." We were hoping to see rules like:

RULE 4:
**if** [PWar IS Low]
[IntelligenceCategory IS OffensiveForceIntel]
**then** [Evaluation IS High]

However, we did not get any (we return to this topic later in the paper).

Some analysis revealed that RULE1 acts as a baseline rule stating that unless a weapon has special characteristics it is not worth investing in. RULE2 states that certain defense weapons are worth investing in. It might seem counter-intuitive that this is the case since both RULE1 and RULE2 state "Evaluation IS Low," but analysis revealed that the Evaluation from RULE1 is significantly lower than from RULE2. Both have the same human-readable output "Evaluation IS Low" but the actual value in the phenotype is very different. Hence, this is an example where the human-readable output can actually be misleading the interpretation of the rule-base. RULE3 on the other hand declares that certain offensive weapons are desirable.

Figure 2 shows how the number of rules used by the best player during the coevolution varied over the first 600 generation. The actual run went to generation 1927, but instances of 3-rule best players were already appearing before generation 500.

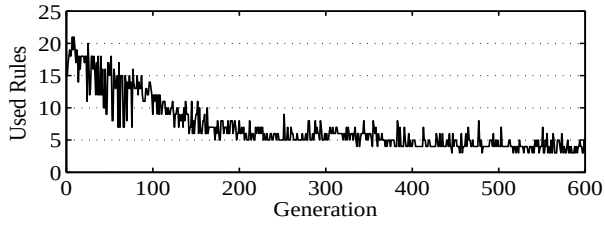


Fig. 2. Number of rules used by best player as a function of generation number

Figure 3 shows how the performance of the best player, playing against the static "expert," varied over the first 600 generations.

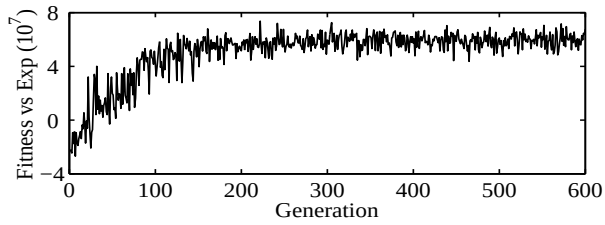


Fig. 3. Fitness of best player, playing against "expert," as a function of generation number

Each player's fitness was computed from the outcomes of $o = 100$ games against the "expert."

The initial efforts had immediately highlighted another problem. It had taken approximately 2 weeks of computation on a single 3 GHz processor to coevolve the initial rules. To properly investigate issues of the sort just described would require faster computational turnaround. We embarked on porting the coevolutionary code to the Processing Graph Method Tool (PGMT), a parallel computing program support system developed at the Naval Research Laboratory (see [17], [1]). An application under PGMT is represented as a data-flow graph (similar to a Petri net) whose processing nodes can run in parallel on separate processors. The mapping of nodes to physical processors takes place at runtime. This flexibility facilitates moving an application, without rewriting, from one parallel-processing system to a very different one — from a small, heterogeneous network of workstations, say, to a large, homogeneous, high-performance shared-memory multi-processor system. (TEMPO examples were run on two machines of the latter type at NRL's Distributed Center for High Performance Computing: Silicon Graphics Origin 3800 and Altix systems. We have also installed PGMT on a cluster at UNC-Charlotte and conducted runs there).

With the availability of the PGMT port, we are beginning to be able to experiment with somewhat larger problems than previously, in particular to increase the number of weapon systems from two to a dozen or so. Doing so, with further relaxation of parsimony pressure, seems to encourage appearance of rule sets (now larger than three rules) containing intel rules with High or Very High in their "then" parts.

We have done a few runs varying just the prolixity setting $p$. This (unsurprisingly) confirmed the expectation that the average number of used rules tends to be lower when the prolixity setting is higher.[8] To track convergence of runs we have also been recording diversity levels of both populations during a run. The diversity measure is computed as follows. Let $P = pop\_size$ be the number of individuals in the population and $G$ be the number of genes in the rule set, and let $x_{ig}$ $(0 \le i < P,\ 0 \le g < G)$ be the floating-point gene value for gene number $g$ of individual $i$. Define the average $a_g$ of gene number $g$ over the population by $a_g = (1/P)\sum_{i=0}^{P} x_{ig}$.
Then the diversity $d$ is the average magnitude of the difference of the gene values from the mean:
$d = (1/PG)\sum_{i=0}^{P}\sum_{g=0}^{G} |x_{ig} - a_g|$.
Figures 4 and 5 display the diversity of the two competing populations (players X and Y, respectively) for three single runs in which we varied the prolixity penalty. Each figure contains three plots, one of each setting of the prolixity penalty $p$.

[8]Prolixity settings of $10^3$, $10^4$, $10^5$, and $10^6$ resulted in the number of rules decreasing from about 30 to about 4.

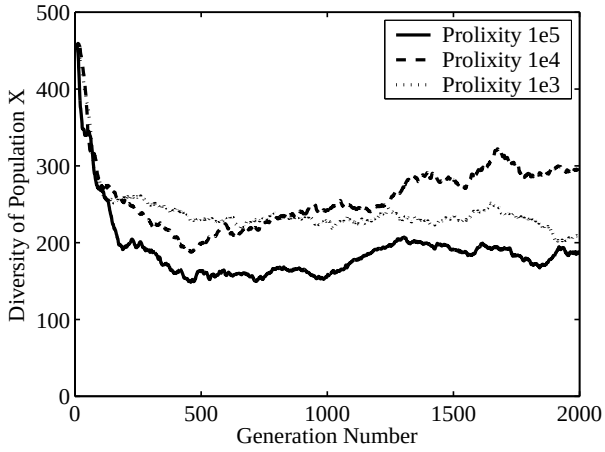


Fig. 4. Diversity of the first population as a function of generation number

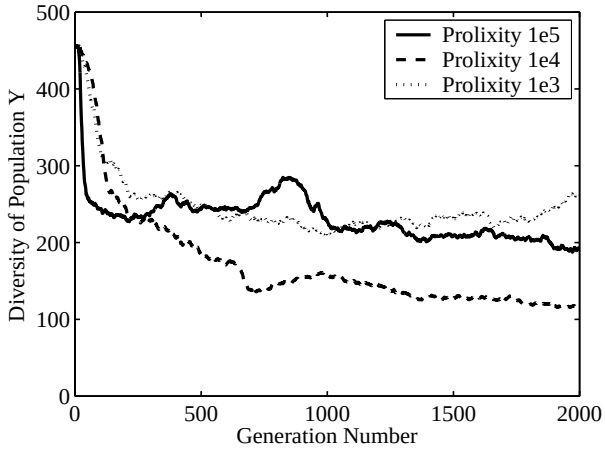


Fig. 5. Diversity of the second population as a function of generation number

We were hoping to see a relationship between diversity of the population and the prolixity level, but none is apparent in the two figures. The population diversity decreased with the number of generations until some equilibrium appeared to be reached. But the apparent equilibrium values do not go monotonically with prolixity, nor is the ordering consistent between the two figures. In fact the diversities in the two populations may stay quite apart from each other: while the dashed line in figure 5, for example, hovers around 120, that in figure 4 stays near 300. We currently do not understand fully this relationship; investigations are under way to analyze and explain it. However, there is considerable variability from run to run — Figure 6 shows

the solid line from Figure 4 (prolixity 1e5) together with four other traces: the results of four other runs with the same parameters. This variability indicates that any systematic correlation between prolixity and divergence is likely to be washed out in the random variation from run to run. Moreover, further experiments described below, though undertaken for other reasons than to study the relation between prolixity penalty and diversity, have revealed that the convergence properties of the coevolutionary system are not as simple as they appeared.

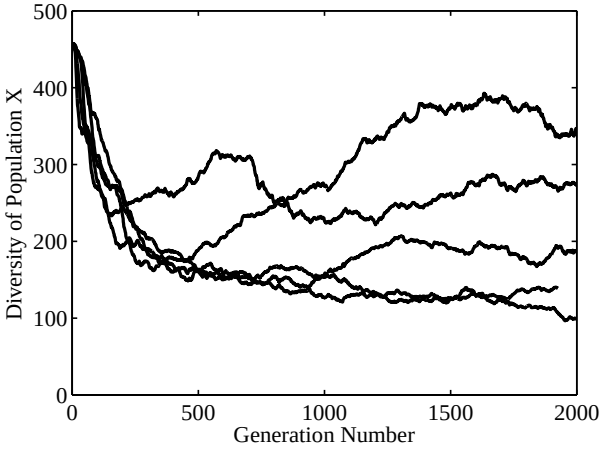


Fig. 6. Diversity of the first population — five runs

Our most ambitious run to date has used 12 weapon types (A, B,...,L), or 72 weapon systems in all, and has run for over 17,000 generations with a population of 2,500. (The prolixity penalty was $p = 10^5$.) The diversity of the two populations is plotted as a function of generation number in Figure 7. In contrast to the behavior shown in Figures 4 and 5 the diversities, after apparently converging over the first few hundred generations, then start to peak up again, reaching a maximum for player Y at about 7,000 generations before starting to drop off again. It is somewhat surprising that the diversity reaches values greater than the original random population, but we do have evidence that some of the Gaussian centers are drifting out of the ranges in which they were originally randomly chosen.

Apparently, after an initial shakeout in which the "losers" are replaced, some rule-sets get recombined or mutated, creating innovation that leads to an increase in diversity. If one strategy is "better" that means it exploits a weakness of the opposition. If that happens, the "better" strategy may start to dominate the population, reducing diversity. Then the opposing population may find a good specific defense and start to defeat the former "better" solution. This flip-flopping may continue ad infinitum.

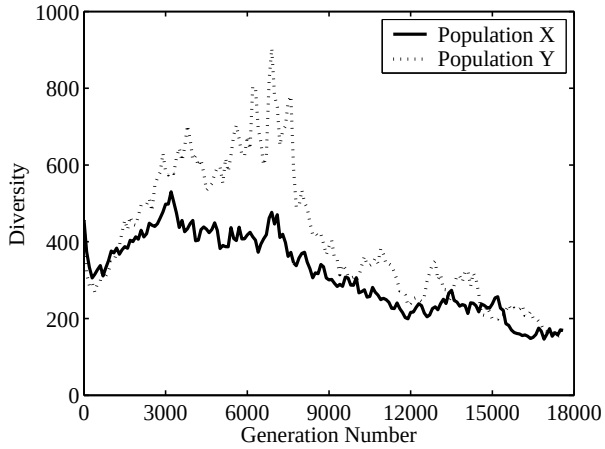


Fig. 7. Diversity of both populations as a function of generation number (run with 12 weapon types)

The following plot is for a run in which we varied the ratios of crossover to mutation ($p_{cross}/p_{mut}$) from 0.3/0.7 to 0.7/0.3. Otherwise the distribution of weapon characteristics and the evolution parameters were as in the run of Figure 7.

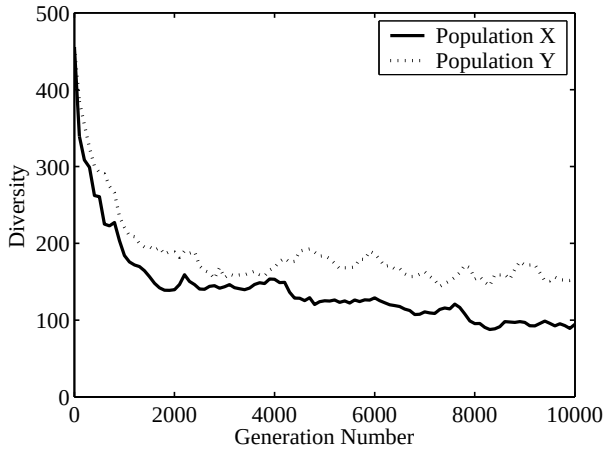


Fig. 8. Diversity of both populations as a function of generation number (run with modified $p_{cross}/p_{mut}$)

The next two plots are from two long runs, with 2 weapon types, in which we attempted to promote intel purchase by (a) lowering the costs of intelligence and counterintelligence to make them essentially free (Figure 9) or (b) making intelligence essentially free but making counterintelligence prohibitively expensive (Figure 10).

An observation: It is not hard to see that, at least when there are 3 or more weapon types, non-transitive orderings of strategies can occur. Consider a player who buys Offensive A and Defensive B versus a player

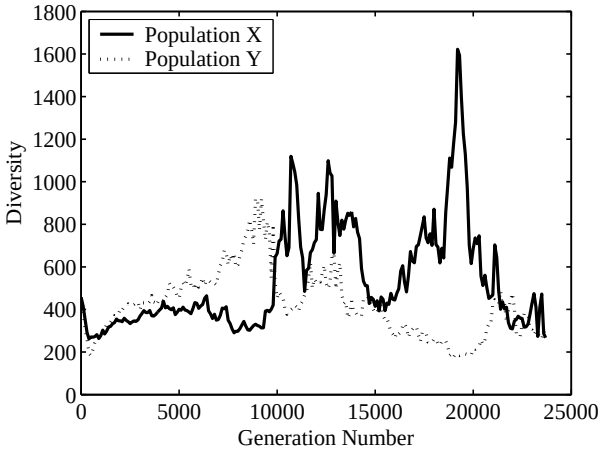


Fig. 9. Diversity of both populations as a function of generation number (run with "free" intelligence and counterintelligence)

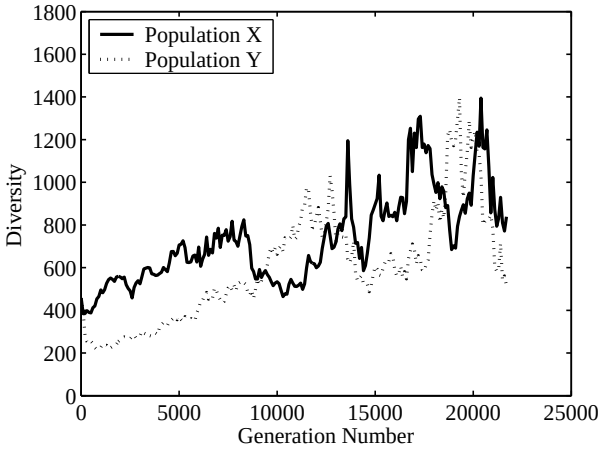


Fig. 10. Diversity of both populations as a function of generation number (run with "free" intelligence and unaffordable counterintelligence)

who buys Offensive B and Defensive C. The first player will get credit for all his offensive utils, and his defense will counter some of the second player's offense. The second player will get credit for only part of his offensive utils, and his defense will be wasted. But the second player will similarly win against a player who buys Offensive C and Defensive A. And the latter will in turn beat the original player who bought Offensive A and Defensive B.

One of our colleagues (Mike Sovereign, an economist) played against the best player from generation 11,106 of population Y of the run of figure 7. He observed that the machine player was playing an absolutely balanced strategy, allocating equal amounts to each offensive type, and equal amounts to each defensive type. The machine player never bought intel/counterintel, despite the fact that the intel rule-set contained a rule with a higher desirability value in its right-hand side than any rule in the weapon-rule set. Sovereign could therefore see its allocations at every stage. He also played a balanced strategy for several rounds, waiting to see whether the machine player would break the symmetry. He finally concentrated a fair number of utils in a single offensive system and shortly thereafter won by essentially the number of excess utils in that offensive type.

Another colleague (Chip Franck, another economist) speculated that what is going on with regard to the diversity is this: We know from the theory of 2-person 0-sum games that if one adopts an optimal (minmax) mixed strategy, then it doesn't matter what the opponent does (as long as he avoids dominated strategies). (For example, in rock-paper-scissors, if one plays the three moves randomly with probabilities (1/3, 1/3, 1/3) then there is a guaranteed expected outcome of 1/3 wins, 1/3 draws, 1/3 losses, regardless of my opponent's strategy.) Thus, if one side in the tempo game evolves something analogous to an optimal mixed strategy, that may take much of the evolutionary pressure off the other side and permit the other side to evolve a quite disparate population without much penalty.

## VII. Conclusions and Future Work

Four years ago when we started this work we didn't know if resource allocation problems of the type represented by the TEMPO game would be approachable using coevolutionary computation methods. And, even though the initial LISP experimental system suggested an affirmative answer, the nature of what we could learn from a coevolutionary system was not obvious to us. We have learned a number of things, which also suggest future research.

The environment, e.g., PWar, budget size, sequence of available weapon types, cost per util, etc., is important, but knowing what your opponent is doing — via intelligence information — is a far slipperier component in a sequential game in which the order of decisions matters.

Though the derived rules can beat most human players immediately, the human players are able to learn the "manner of play" of the machine codes. This suggests that "intel" may after all be important for the machine players to win consistently. Human players usually do buy intel, but the machine does not. The human players use the information they get from intel purchases to good advantage. This works, of course, when the machine rules are not changed from competition to competition. During coevolution, on the other hand, the rules faced by a player do change from competition to competition, since the identity of the

opponent changes.

A reviewer of this paper wrote of "...conceiving, and evolving, solutions that can also learn or adapt their own strategy. As long as an evolved strategy is fixed it will exhibit mid-or long-term weakness when confronted to human counterparts. The mechanism to perform adaptation might be subject to (co-)evolution." And indeed a primary need is to solve the mystery of why the code produces rules that allocate most effort to evaluating weapon characteristics and ignore available information about the opponent's behavior.

To explore the conditions under which intelligence on the opponent's inventory of weapons is useful we have coevolved rules where knowing should matter. In particular we have expanded the number of weapons types to 12, drawing the cost and utils and availability from the same distribution, and we have tried lowering the cost of intelligence and preventing the purchase of counterintelligence. It was expected that a player should base its investment decisions on the offensive weapons the opponent owns and operates as well as on the "bang per buck" for a given weapon. When a set of evolved rules that did seem as though it ought to do so competed against a human player, it appeared that in actual play the rules did not lead to buying and using intelligence information. Further, several of us have played against the coevolved rules and we are now generally able to win.

One might ask whether the currently used representation for the rules is expressive enough. We have made a start toward exploring alternative representations. In particular, a student in a recent class project [22] tried replacing fuzzy-logic rules with LISP-like expression trees as evaluators for the "desirability" of weapon systems, while otherwise retaining the architecture of the current system. (The system was thus quite different from the LISP-based system mentioned in Section IV.) It was found that the tree-based system performed quite creditably against the existing system, and the idea is well worth pursuing further.

Another idea worth trying is to evolve a population against opponents with more "human-like" strategies than arise in a purely coevolutionary setting. (Two reviewers have made roughly similar suggestions.) For example, we could hand-craft rule sets based on strategies that humans have found effective. (In particular they would buy and make plausible use of intel.) Then we would seed one of the two populations with instances of these rule sets. Two sorts of questions arise. First, how do the seeded individuals fare in their own population? Do they thrive and propagate themselves? What do they evolve into? Second, how does their presence as opponents affect the other population? Will adaptable opponents on one side encourage the evolution of adaptable players on the other side?

Our colleague, when playing the 12 weapons type game, perceived that to compute the potential outcome of a competition required much more analysis than for the two weapons case. Further, if the weapons types don't have the same "bang for buck" then the decisions become even more challenging. Thus, we see in this case evidence of the effects of information overload. There are three standard techniques for dealing with information overload: filtering (discarding information), clustering (creating a hierarchy), and random selection of what to do [25]. We are interested in how clustering might arise and how effective it might be in reducing the computational load, though at the expense of reducing the information used for making decisions. The current system has PWar and budgets as externally supplied values. A hierarchical competition where the higher-level system that determines these values interacts with the current game is of considerable interest.

The prolixity penalty appears likely to be useful as a proxy for the information handling capacity of the "decision maker." As mentioned, teams playing the paper TEMPO game will sometimes deliberately forego intelligence entirely, possibly to cope with information overload. Once, in an attempt to help a poorly performing team, an instructor gave them free, accurate, completely detailed information about the opponents' decisions. This conferred no advantage — to the contrary, they lost decisively. The more rules an organization uses to make decisions, the greater the demand for information processing capability. Since most large organizations use fairly simple metaphors for making decisions, and these metaphors can be captured as "if-then" rules, it is possible to imagine exploring alternative rules using different fitness functions to determine why organizations have come to the rules they use. This is very much like the "inverse problem" where given a result we have to find a potential cause of that result [16]. This is why "1R" and "Naive Bayes" [33] work so well so often. This leads us to consider putting an explicit cost function into the coevolution that would pay for the increasing demand for information processing.

Exploring questions such as these will require a large and growing computational environment. Fortunately, the choice of PGMT has facilitated our ability to move between different multi-processor environments with a minimal amount of recoding. Our research has now moved from our preliminary trials on desktop computing machines to networks of processors. We thus far have done our computation in 2-3 GHz desktop computers, on a loosely coupled network of 2 Sun and 3 Silicon Graphics workstations, on 28 processors in a tightly coupled network of 128 processor in the SGI 3800, and

we are prepared to do experiments in other networked environments. We have also worked on a cluster of computers at UNCC. The use of PGMT has permitted us to move efficiently from one computing environment to another. Once PGMT has been installed bringing our TEMPO codes into operation has taken between 4-18 man-hours. Our expectation is that this combination of a very flexible representation of the resource allocation problem and the computational environment that PGMT provides will permit research on a growing family of poorly understood problems encountered in large living systems [25].

Our planned future work includes incorporating investment in R&D into the game, analysis of the developed coevolutionary system with selection of different fitness functions for the competitors, and the possibility of investigating a non-zero sum game (note that we can already set the environment variables in such a way, that the game is not symmetrical: the players start with different budgets, different weapons are available at different iterations, etc.). These are potential research questions that TEMPO can be used to explore. There are also other issues of proper representation, e.g., whether fuzzy logic is the best way to co-evolve a hierarchical system. We hope that within the next year we should have answers for some of these questions.

## Acknowledgements

This work was initially supported by Dr. William Mularie of DARPA/ISO and subsequently by the Office of the Secretary of Defense. We greatly appreciate their interest and encouragement for our research on the complexities of using scenarios for planning. The parallel TEMPO game coevolutionary examples were run on resources provided under the auspices of the High Performance Computing Modernization Program Office. We are grateful for their support.

Parts of material from chapter 14 of [23] were used in section II (background information on coevolutionary approaches to games) with the authors' permission.

We thank Prof. Michael Sovereign and Prof. Raymond ("Chip") Franck for their interest and for much thoughtful commentary. Thanks to Professor Franck for giving one of our machine players exposure in the classroom, and to Professor Sovereign for good-naturedly serving as a guinea pig by playing more than once against the machine.

The authors extend thanks to Yunjun Mu for writing code for the TEMPO Fuzzy-Logic coevolutionary system; to Wendell Anderson and Roger Hillson for an education in PGMT and for support in parallelizing the system; to John Thornton, also for support in the parallelization and for thought-provoking discussions as well; and to Neal Wagner, for running and analysing several experiments. Finally, we thank anonymous reviewers, who provided us with excellent comments, and David Fogel for his useful suggestions.

## References

[1] W. Anderson, "Processing Graph Method (PGMT) User's Manual," US Naval Research Laboratory, October 2002.

[2] R. Axelrod, "Evolution of strategies in the iterated prisoner's dilemma," in *Genetic Algorithms and Simulated Annealing*, L. Davis, ed., Pitman, London, pp.32-41, 1987.

[3] K. Chellapilla and D.B. Fogel, "Evolution, neural networks, games, and intelligence," *Proceedings of the IEEE*, Vol.87, No.9, pp.1471-1496, 1999.

[4] K. Chellapilla and D.B. Fogel, "Evolving neural networks to play checkers without expert knowledge," *IEEE Transactions on Neural Networks*, Vol.10, No.6, pp.1382-1391, 1999.

[5] K. Chellapilla and D.B. Fogel, "Evolving an expert checkers playing program without using human expertise," *IEEE Transactions on Evolutionary Computation*, Vol.5, No.4, pp.422-428, 2001.

[6] D. Cliff and G. F. Miller (1996), "CoEvolution of Neural Networks for Control of Pursuit and Evasion," University of Sussex, U.K. [Online]. Available: http://www.cogs.susx.ac.uk/users/davec/pe.html

[7] P.J. Darwen and X. Yao, "Why more choices cause less cooperation in iterated prisoner's dilemma," *Proceedings of the 2001 Congress on Evolutionary Computation*, IEEE, Piscataway, NJ, pp.987-994.

[8] J. Espinosa and J. Vandewalle, "Constructing fuzzy models with linguistic integrity from numerical data-AFRELI algorithm," *IEEE Transactions on Fuzzy Systems*, Vol.8, No.5, pp.591-600, 2000.

[9] D. Floreano. and S. Nolfi, "God save the Red Queen! Competition in co-evolutionary robotics," *Genetic Programming 1997*, J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba, and R.L. Riolo, eds., Morgan Kaufmann, San Mateo, CA, pp.398-406, 1997.

[10] D.B. Fogel, "Evolving behaviors in the iterated prisoner's dilemma," *Evolutionary Computation*, Vol.1, No.1, pp.77-97, 1993.

[11] D.B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, Piscataway, NJ, 1995.

[12] D.B. Fogel, *Blondie 24: Playing At The Edge of AI*, Morgan Kaufmann, San Francisco, CA, 2002.

[13] D.B. Fogel, T.J. Hays, S.L. Hahn, and J. Quon, "A self-learning evolutionary chess program," *Proceedings of the IEEE*, Vol.92, No.12, pp.1947-1954, 2004.

[14] P.G. Harrald and D.B. Fogel, "Evolving continuous behaviors in the iterated prisoner's dilemma," *BioSystems*, Vol.37, pp.135-145, 1996.

[15] W.D. Hillis, "Co-evolving parasites improve simulated evolution as an optimization procedure," *Artificial Life II*, C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, eds., Addison-Wesley, Reading, MA, pp.313-324, 1992.

[16] E.T. Jaynes, *Probability Theory: The Logic of Science*, Cambridge University Press, 2003.

[17] D. Kaplan, "Introduction to the Processing Graph Method," U.S. Naval Research Laboratory, March 1997.

[18] J.R. Koza, *Genetic Programming*, Cambridge, MA: MIT Press, 1992.

[19] D.E. Knuth, *Sorting and Searching*, Vol.3 of *The Art of Computer Programming*, Addison-Wesley, New York, NY, 1973.

[20] R. Le Riche, C. Knopf-Lenoir, and R.T. Haftka, "A segregated genetic algorithm for constrained structural optimization," *Proceedings of the Sixth International Confer-*

*ence on Genetic Algorithms*, L.J. Eshelman, ed., Morgan Kaufmann, San Mateo, CA, pp.558-565, 1995.
[21] D. Lovallo and D. Kahneman, "Delusions of success," *Harvard Business Review*, vol.81, no.7, pp.57-63, July 2003.
[22] J. Merritt, "Function tree strategy for TEMPO," UNC-Charlotte Project Report, May 2005.
[23] Z. Michalewicz and D.B. Fogel, *How to Solve It: Modern Heuristics*, 2nd edition, Springer, Berlin, 2004.
[24] Z. Michalewicz and G. Nazhiyath, "GENOCOP III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints," *Proceedings of the 1995 IEEE Conference on Evolutionary Computation*, IEEE Press, Piscataway, NJ, pp.647-651, 1995.
[25] J.G. Miller, *Theory of Living Systems*, University of Colorado Press Niwot, Colorado, 1995.
[26] J. Paredis, "Co-evolutionary constraint satisfaction," *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature*, Y. Davidor, H.-P. Schwefel, and R. Manner, eds., Lecture Notes in Computer Science, vol.866, Springer, Berlin, pp.46-55, 1994.
[27] J. Paredis, "The symbiotic evolution of solutions and their representations," *Proceedings of the Sixth International Conference on Genetic Algorithms*, L.J. Eshelman, ed., Morgan Kaufmann, San Mateo, CA, pp.359-365, 1995.
[28] C.-A. Peña-Reyes and M. Sipper, "Fuzzy CoCo: A cooperative-coevolutionary approach to fuzzy modeling," *IEEE Transactions on Fuzzy Systems*, Vol.9, No.5, pp.727-737, 2001.
[29] C. Reynolds, "Competition, coevolution and the game of tag," *Proceedings of Artificial Life IV*, R. Brooks and P. Maes, eds., MIT Press, Cambridge, MA, pp.56-69, 1994.
[30] P. Schwartz, *The Art of the Long View: Planning for the Future in an Uncertain World*, Currency/Doubleday, New York, NY, 1991.
[31] A.V. Sebald and J. Schlenzig, "Minimax design of neural-net controllers for uncertain plants," *IEEE Transactions on Neural Networks*, Vol.5, No.1, pp.73-82, 1994.
[32] J. Valente de Oliveira, "Semantic constraints for membership function optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, Vol.29, No.1, pp.128-138, 1999.
[33] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, 1999.