Centre for Computational Finance and Economic Agents

WP019-08

Working Paper Series

Philip Saks
Dietmar Maringer

Single versus Multiple Tree Genetic Programming for Dynamic Decision Making

July 2008

CCFEA

www.essex.ac.uk/ccfea

Single versus Multiple Tree Genetic Programming for Dynamic Decision Making

Philip Saks* Dietmar Maringer

July 15, 2008

Abstract

This paper considers genetic programming (GP) for dynamic decision making. Standard genetic programming only uses a single decision tree for decision making. In contrast, this paper proposes a general multiple tree framework for dynamic decision problems, where evaluation is contingent on the previous output of the program. The working hypothesis is that "recurrent" multiple trees are superior compared to conventional single trees for dynamic decision problems. To test this hypothesis, a single and a dual tree representation is considered. Both representations return Boolean values, but for the dual trees, evaluation is contingent on their previous output. Specifically, if the previous output was `FALSE`, the first tree is evaluated, otherwise it is the second.

The single and dual trees are applied within two different domains. The first domain consists of a coevolutionary predator-prey type environment where the single and dual trees are treated as different species. The objective of a predator is to capture the phenotypic behavior of a prey. Naturally, the objective of the prey is to evade the predator. It is found that the dual trees have greater expressive capabilities, since they can capture the dynamics of the single trees when acting as predators, while evading when acting as prey.

The second domain is closer related to finance. The single and dual trees are used to evolve successful trading strategies on artificial financial time series. Two different processes are constructed that exhibit some features also found in real financial data, i.e., mean-reversion and momentum effects. It is found that the single trees are unable to capture the dynamics of the mean-reverting process, but the dual trees succeed. For the trending series, both representation are capable of capturing the underlying dynamics, but the single trees have better out-of-sample performance compared to the dual trees. This is found to be a manifestation of *Ockham's razor*.

JEL classifications: C0, C45, C53, C63

Keywords: Genetic programming, coevolution and financial trading strategies.

*both: Centre for Computational Finance and Economic Agents, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, UK. {psaks,dmaring}[at]essex.ac.uk. Financial support from the EU Commission through MRTN-CT-2006-034270 COMISEF is gratefully acknowledged.

1 Introduction

Many control problems can be characterized as dynamic decision making. Control problems are often associated with mechanical systems, but the focus of this paper is more directed towards financial applications, specifically, trading. The majority of applications of genetic programming (GP) in trading use a single tree to represent a single decision rule [1, 22, 4, 17]. However, it might be useful to have different decision rules for different situations. Becker and Seshadri [2] suggest to separate decision making into two trees, i.e., a dual tree structure. Their motivation is to use mutualistic coevolution for evolving buy and sell rules separately, but they do not provide any insights behind or justification thereof. This paper addresses this issue, but it also goes further. It proposes a general multiple tree framework for dynamic decision making. Instead of representing a program as a single tree, it is represented as multiple trees. Multiple tree GP has previously been considered in the literature [18], but this paper proposes a different setup for temporal phenomena. Specifically, the previous output of the program determines what tree to evaluate currently. As this type of feedback is used extensively in recurrent neural networks for tackling dynamic problems [20], it might also prove beneficial in the context of GP. Consequently, the working hypothesis is that "recurrent" multiple trees are superior compared to conventional single trees for dynamic decision problems.

To test the hypothesis, this paper considers single and dual trees for two different problems. The first problem takes a general view of program representation, and a dynamic binary decision problem is constructed from basic Boolean operators. The single and dual trees compete directly within a predator-prey type environment, where the objective of the predator is to capture the phenotypic behavior of the prey. Naturally, the objective of the prey is to evade the predator. Both predators and preys evolve. Consequently, this is an example of competitive coevolution.

The second problem is closer related to finance. The single and dual trees are used to evolve trading strategies on artificial financial time series. The output of the programs dictates the market position, i.e., long or short. At this stage, artificial data is preferred over real data for a number of reasons. The true data-generating process of real data is unknown, and therefore exploitable patterns need not exist. Moreover, for the sake of validation it is important to clarify whether the methodology is capable of discovering patterns that do exist.

The remainder of the paper is outlined as follows. Section 2 introduces the multiple tree framework and discusses various implementation issues. Section 3 gives an introduction to natural coevolution and presents the predator-prey environment. Section 4 describes the artificial market environment where trading rules are evolved. Finally, Section 5 concludes.

2 Single and Multiple Tree Genetic Programming

Genetic programming is often described as a derivative method of genetic algorithms, where individuals are computer programs instead of binary strings. In

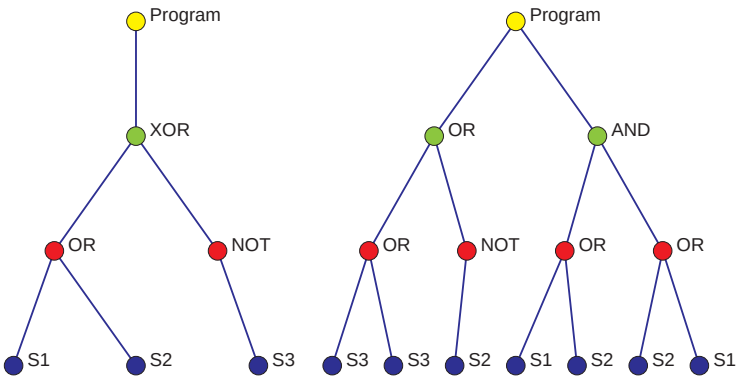

Figure 1: Examples of a single (left) and a dual tree (right), constructed using Boolean operators. The root node of the single tree is `XOR`, while the root nodes for the dual tree are `OR` and `AND`.

general, an individual program is represented as a single tree structure. However, it is perfectly feasible to generalize this concept such that an individual consists of multiple trees. Haynes et al. [12] have used this approach for evolving a team within a multi agent system. Hence, each team member is represented as a tree, and the entire team is one "individual". This methodology is applied to the predator-prey pursuit problem. The predator-prey pursuit problem was originally introduced by Benda et al. [3], and consists of four predators attempting to capture, or surround, a randomly moving prey inside a grid-world. The prey can see the predators and vice versa, but the four predators cannot communicate to plan a capture strategy. On a slightly different note, Langdon [18] uses multiple trees to evolve data structures such as stacks, queues and lists. In this context, each tree supports operations associated with the data structure. For example, a stack has the five operations; makenull, top, pop, push and empty. Finally, Becker and Seshadri [2] use a dual tree structure for evolving trading rules. Figure 1 provides an example of a single and a dual tree constructed from Boolean operators, i.e., the function set consists of `AND`, `OR`, `XOR` and `NOT`. For the dual tree, the "program" dictates whether to evaluate the first or the second tree.

While this paper only considers applications of single and dual tree programs, it should be emphasized that this paper proposes a generalized multiple tree approach for temporal decision making. Formally, an individual consists of k trees, where each tree (i) is a functional mapping of the information set at time t $(\Omega_{i,t})$ to an output $(b_{i,t})$, $f_i : \Omega_{i,t} \mapsto b_{i,t}$. The output of the program is denoted o_t. Since the actual return type of a tree might not be compatible with the output of the program, the transformation $g_i : b_{i,t} \mapsto o_{i,t}$ ensures this. At each point in time, the output of the program is simply the transformed return value from one of the trees. The function $h : \{o_{t-1}, \Omega_{1,t}, \ldots, \Omega_{k,t}\} \mapsto i_t \in \{1, \ldots, k\}$ determines what tree (i_t) to evaluate at time t, as a function of the previous program output and all current information sets. Since decision making is contingent on the previous output, this can be viewed as a form of recurrent GP. Consequently, the output of

the program at time t is,

$$i_t = h(o_{t-1}, \Omega_{1,t}, \ldots, \Omega_{k,t}) \quad (1)$$

$$o_t = g_{i_t}\left(f_{i_t}(\Omega_{i_t,t})\right). \quad (2)$$

The following sections provide examples of different applications of this methodology.

Standard GP has to adhere to the closure property, i.e., every function in the function set should be able to accept as argument, any value returned by an element from the union between the function and terminal set. Typically, a trading strategy consists of both numeric variables and Boolean operators, thus violating the closure constraint. To overcome this issue, Montana [21] proposes Strongly Typed Genetic Programming (STGP). STGP can handle an arbitrary number of data types, provided legal function compositions are specified in advance. Since this paper considers multiple trees in conjunction with STGP it has been necessary to implement a new framework in C++, instead of relying on a standard application. A selective description of the framework follows.

It has repeatedly been pointed out that programs in GP are represented as tree structures. A tree can easily be represented with a pointer based structure, where each parent node is connected to a number of children via pointers. However, in the context of GP, Keith and Martin [16] propose a more efficient linear genome which can be thought of as a "flattened" tree. When a pointer based tree structure is evaluated, it is traversed in a deterministic fashion. By arranging the nodes in the linear genome by the order in which they are evaluated in the pointer based tree structure, the pointers are made redundant. Hence, a linear genome is simply evaluated by traversing it from the beginning to the end. Once the genome has been implemented, it is straightforward to construct a multiple tree as a collection of genomes.

Another implementation issue relates to STGP. When a method is declared in an object-oriented language like C++, it can only return one data type. This poses a problem with respect to strong typing, because the functions and terminals return a mixture of types. To overcome this issue, the function and terminal nodes in the genome have to return a generic variable. A function which receives this variable, must then cast it into the type that it expects. This is the reason why valid function compositions have to be specified in advance for STGP to work. For example, if a Boolean function attempts to cast a numeric value, then an error will occur and possibly terminate the program execution.

In addition to strong typing, the framework gives the possibility of providing semantic constraints. Semantic constraints makes further restrictions to what constitutes valid function compositions. This is important when adding prior domain knowledge to a problem. In the context of trading rules, for example, it is not meaningful to compare prices and volume information. On a more general note, numeric values with different units of measures should not be compared directly. Using semantic constraints makes the search more efficient, since computational resources are not wasted on nonsensical solutions [4]. However, implementing strong typing and semantic constraints is more challenging, because

not only should valid function compositions be specified in advance, but the genetic operators should also take this information into account. For example, in the context of crossover, there is no guarantee that the selected subtree in the first parent can be swapped with any subtree in the second parent, such that the constraints are satisfied. Having multiple trees also raises some issues related to crossover. With multiple trees it is no longer given what trees are permitted to exchange genetic material, e.g., whether tree i in the first parent can crossover with tree j in the second parent, where $i \neq j$. Moreover, tree i in the first parent could crossover with tree i in the second, where $i = 1, \ldots, k$, such that k crossovers are performed [12]. Following Langdon [18], the crossover algorithm employed in this paper, uniformly selects one tree in the first parent and crossover is performed with the equivalent tree in the second parent. The crossover algorithm produces a single offspring, where most of the genetic material comes from only one of the parents. Performing crossover between equivalent trees eliminates any issues related to heterogenous primitives. The mutation algorithm uniformly selects a single tree to mutate. Within the chosen tree, a node is randomly selected and its subtree is replaced with a newly generated tree.

3 Predator-Prey Environment

3.1 Coevolution

The term *coevolution* was initially coined by Ehrlich and Raven [8] in their comprehensive study on foraging patterns of butterflies. It is found that various plants have evolved different chemical compounds, which do not assist basic metabolic processes, but serve as a defense against herbivores by reducing the palatability of the plant in which they are produced. However, insects might adapt to counter these new challenges, thus giving rise to an evolutionary "arms race" against the plants. Ehrlich and Raven do not give a precise definition of coevolution other than being "the examination of patterns of interaction between two major groups of organisms with a close and evident ecological relationship". Hence, Janzen [14] defines it as "an evolutionary change in a trait of the individuals in one population in response to a trait of the individuals in a second population, followed by an evolutionary response by the second population to a change in the first". This is a fairly strict definition since it requires that the evolution of a specific trait is induced by the other, and that both traits must evolve. In practice the evolution of a trait in one species tends to occur in response to traits from multiple species, e.g., a plant has most likely developed its defense mechanisms to counter a range of different insects instead of just one. This is known as *diffuse coevolution*. In its broadest sense coevolution is simply "reciprocal genetic changes that might be expected to occur in two or more ecologically interacting species" [10, p. 3].

The insect-plant relationship mentioned previously is an example of a *parasitic* relationship, but coevolution also gives rise to *mutualistic* relationships, where collaboration emerges instead of competition [24, p. 452]. An example of mutualism is given between acacia trees and ants. The trees provide shelter and

food for ant colonies, but in return the ants offer protection against herbivorous insects [13]. Mutualistic coevolution is not only found at a macro level between species, but it also exists on a genetic level. Dawkins [7] give an example, where genes coevolve to express certain traits in honey bees. Both parasitic and mutualistic coevolution are considered in the following.

3.2 Framework

The purpose of this paper it to examine the expressiveness of single trees versus dual trees. Both the single and the dual trees return Boolean values. As stated in Section 2, evaluation of the dual trees is contingent on the previous output of the program. Specifically, let $b_{i,t}$ be the truth value $(0,1)$ of tree i at time t, then the output of the program is given by,

$$o_t = \begin{cases} b_{1,t} & \text{if } o_{t-1} = 0 \\ b_{2,t} & \text{otherwise} \end{cases}. \tag{3}$$

All trees are constructed from the same primitives. The function set consists of `AND`, `OR`, `XOR` and `NOT`. The terminal set has three binary series; `S1`, `S2` and `S3`. Each of these series is a two-state Markov process where the current state only depends on the previous one. The conditional distribution is defined by the probabilities $p_{m,n}$ of moving from state m at time $t-1$ to state n at time t

$$\begin{aligned} P(s_t = 0 | s_{t-1} = 0) &= p_{0,0} \\ P(s_t = 1 | s_{t-1} = 0) &= p_{0,1} \\ P(s_t = 0 | s_{t-1} = 1) &= p_{1,0} \\ P(s_t = 1 | s_{t-1} = 1) &= p_{1,1} \end{aligned} \tag{4}$$

where $\sum_n p_{m,n} = 1$ for $m = 0,1$ and $p_{m,n} \geqslant 0 \ \forall m,n$. Figure 2 depicts the three processes used in the following experiments. Each process consists of 1000 samples and is generated using the parameters $p_{0,0} = p_{1,1} = 0.95$ and $p_{0,1} = p_{1,0} = 0.05$.

As mentioned previously, this section uses competitive coevolution to test the working hypothesis. The existing applications of competitive coevolution in the literature, are concerned with problems in which optimal strategies exist in *absolute* terms, i.e., strategies can be found which defeat many or all possible adversaries [11, 23, 15]. In contrast, this paper deals with the *relative* problem of comparing the single and dual tree representations. More specifically, it addresses the question of whether a single tree can capture the phenotypic behavior of a dual tree and vice versa.

To formalize this within a predator-prey context, a prey generates a binary series which the predator seeks to imitate. If the classification accuracy of the predator is higher than a predefined threshold $\bar{\sigma}$, then the predator has managed to capture the dynamics of the prey. Therefore, a predator can either succeed or fail in capturing a particular prey, but there exist no optimal strategies that capture many different preys. Consequently, standard methods do not apply and a novel coevolutionary environment must be implemented. A procedure is

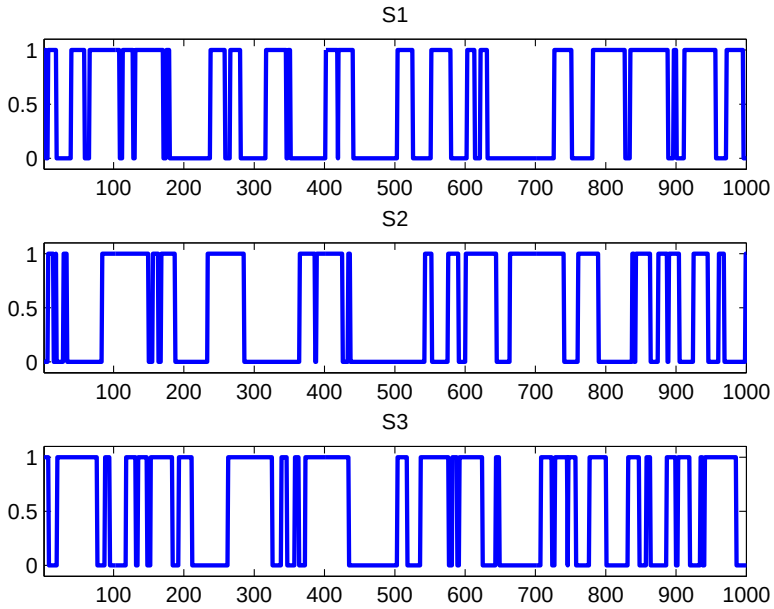

Figure 2: Three two-state Markov processes.

proposed where the predator evolution is nested within the prey evolution, thus coevolution is not simultaneous as usual; Algorithm 1 summarizes this concept.

The procedure requires a number of parameters: $\bar{g}_{\text{predator}}$ and $\bar{g}_{\text{prey}}$ denote the maximum number of predator and prey generations, and have related counters g_{predator} and g_{prey}. Then $\bar{c}_{\text{predator}}$ and $\bar{c}_{\text{prey}}$ are stopping criteria related to convergence, where c_{predator} and c_{prey} are their associated counters. Finally, $\bar{\sigma}$ is the success criterion for capturing a prey introduced previously.

The algorithm contains three **while**-loops nested within each other. The outer **while**-loop (ln. 2) is associated with the prey evolution. Inside is an **if**-statement (ln. 3) that ensures that fitnesses have been evaluated before making genetic operations. For each prey (ln. 7) a new predator population is initialized, and the second **while**-loop (ln. 10) is responsible for its nested evolution. The predator population evolves until convergence or the maximum number of generations is reached, or until the prey is caught, i.e., the classification accuracy is higher than the predefined threshold ($\sigma_{i,j} \geqslant \bar{\sigma}$). The predator fitness is assigned as the classification accuracy (ln. 18), and if a better predator has emerged the convergence counter c_{predator} is reset (ln. 22). Having terminated the nested predator evolution, the prey fitness is assigned as the inverse of the maximum fitness in the predator population ($f^i_{\text{prey}} = 1 - \max_j(\sigma_{i,j})$). Finally, if the average fitness of the prey population has increased, the counter c_{prey} is reset.

3.3 Results

Two experiments are presented in the following. The first is where the dual tree population is prey and the single tree population is predator, and vice versa

Algorithm 1 Predator Prey Coevolution

Require: $\bar{g}_{\text{predator}}$, $\bar{g}_{\text{prey}}$, $\bar{c}_{\text{predator}}$, $\bar{c}_{\text{prey}}$ and $\bar{\sigma}$

initialize prey population
while $g_{\text{prey}} < \bar{g}_{\text{prey}}$ **and** $c_{\text{prey}} \leqslant \bar{c}_{\text{prey}}$ **do**
  if $g_{\text{prey}} > 0$ **then**
    perform genetic operations on prey population
  end if
  evaluate forecasts of prey population
  for each prey i **do**
    initialize predator population
    $g_{\text{predator}} = 0$, $c_{\text{predator}} = 0$
    while $g_{\text{predator}} < \bar{g}_{\text{predator}}$ **and** $c_{\text{predator}} \leqslant \bar{c}_{\text{predator}}$ **and** $\sigma_{i,j} < \bar{\sigma}$ **do**
      if $g_{\text{predator}} > 0$ **then**
        perform genetic operations on predator population
      end if
      evaluate forecasts of predator population
      $j = 0$
      while $j < n_{\text{predators}}$ **and** $\sigma_{i,j} < \bar{\sigma}$ **do**
        assign fitness to predator j, as similarity with prey i
        $f^{j}_{\text{predator}} = \sigma_{i,j}$
        increment j
      end while
      if new superior predator has emerged **then**
        $c_{\text{predator}} = 0$
      end if
      increment c_{predator} and g_{predator}
    end while
    assign fitness to prey i
    $f^{i}_{\text{prey}} = 1 - \max_j(\sigma_{i,j})$
  end for
  if new superior prey population has emerged **then**
    $c_{\text{prey}} = 0$
  end if
  increment c_{prey} and g_{prey}
end while

for the second. The populations are initialized using the *ramped half-and-half* method and have a size of 250, but different maximum complexities are imposed, i.e., 100 and 50 nodes for the single and dual tree individuals, respectively. The stopping criteria are $\bar{g}_{\text{predator}} = 50$, $\bar{g}_{\text{prey}} = 20$, $\bar{c}_{\text{predator}} = 10$ and $\bar{c}_{\text{prey}} = 20$. The success criterion for capturing a prey is $\bar{\sigma} = 0.95$. A normal tournament selection with a size of five is used, and the crossover and mutation probabilities are 0.9 and 0.1, respectively. Moreover, the probability of selecting a function node during reproduction is 0.9.

Figure 3 depicts the evolution of the prey fitness for each of the two experiments. When the dual trees act as prey, the initial average fitness is 0.173, but already after seven generations it has converged to 0.432 where it remains fairly stable. At this point two-thirds of the population has attained the maximum fitness of 0.494. A solution with a fitness exceeding 0.5 is not stable, which is easily understood from a genotypic perspective. Consider the extreme scenario of a contrarian strategy with a fitness of 1, then it only requires the introduction of a `NOT`-function as root-node and the fitness would be zero. Hence, 0 and 1 are distant phenotypes, but close genotypes. The optimal dual tree strategy essentially appears random to the single trees. Despite having evolved an optimal strategy, there are always some dual trees that get caught. This is just an artifact of the destructive nature of the genetic operators, i.e., there is always a risk that the best parents can have unsuccessful offsprings.

When the single trees are prey, the situation is remarkably different. In generation zero the average fitness is 0.014 and it attains a value of only 0.031 after six generations. For all generations, at least 92% of the prey get caught, which clearly demonstrates that the dual trees are capable of capturing single tree dynamics. That a fraction of the prey do escape can be ascribed to early stopping of the predator evolution, because otherwise they would multiply within the population. Hence, individuals with fitnesses above 0.05 are not viable in this environment, but during the course of evolution a larger proportion of prey emerge which have fitnesses just below this critical value.

4 Artificial Market Environment

4.1 Framework

It has been found that financial data may exhibit momentum and mean-reversion effects. For example, Brock et al. [5] find that a simple moving average strategy generates excess returns on the Dow Jones Industrial Average in the period from 1897 to 1986. Dacorogna et al. [6] report significant mean-reversion effects on high frequency foreign exchange data. Inspired by these findings, two stochastic processes are constructed that capture these dynamics.

The first process is a mean-reverting process based on the popular Relative Strength Index (RSI) technical indicator [26]. RSI is a standardized indicator in the interval between 0 and 1, where low values indicate that the market is oversold and vice versa for high values. Consequently, when the market is overbought (oversold) a sell (buy) order is issued – see Appendix A for details. Al-

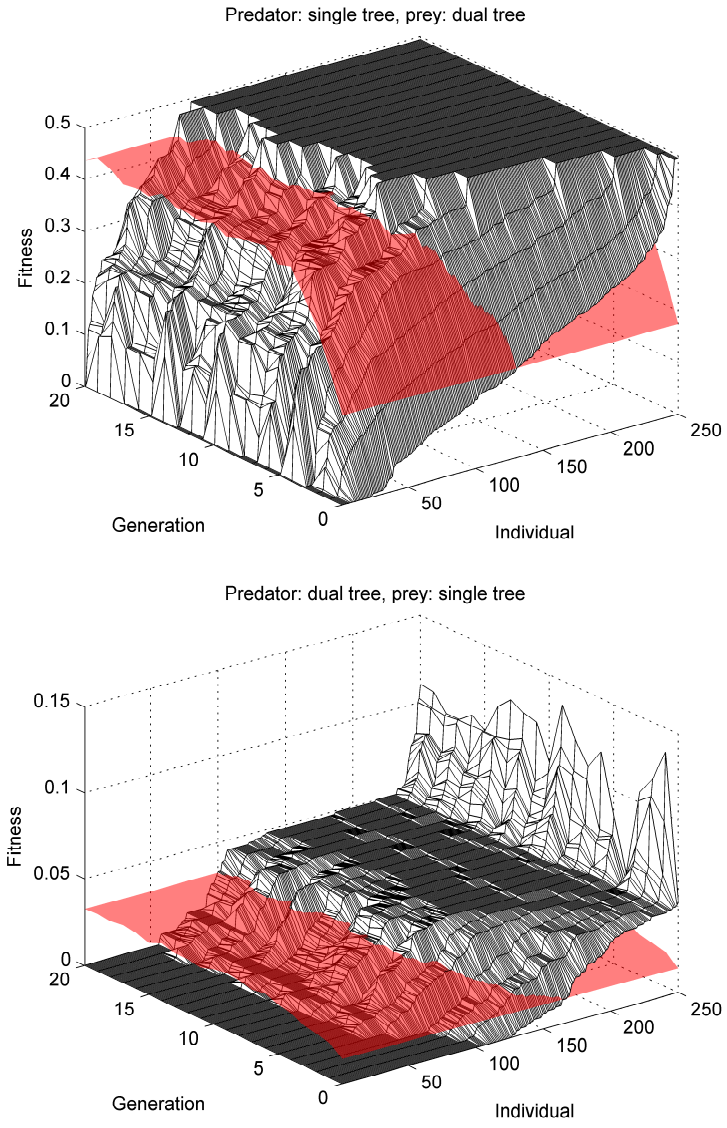

Figure 3: Evolution of prey fitness sorted in ascending order for each generation, where the red surfaces are the average fitnesses.

Algorithm 2 RSI Process

Require: $y_0 > 0$, $t_{\max}$, $\sigma > 0$, $\psi \in [0.5, 1)$, $\beta \in [0, 0.5]$ and $\ell > 0$

for $t = 1$ to $t_{\max}$ **do**
  if $RSI(\ell)_{t-1} < \beta$ **then**
    $s_t = 1$
  else if $RSI(\ell)_{t-1} > 1 - \beta$ **then**
    $s_t = -1$
  else
    $s_t = s_{t-1}$
  end if
  $y_t = y_{t-1} \cdot \left(1 + r_t + s_t \cdot \Phi^{-1}(\psi, 0, \sigma)\right) \qquad r_t \sim N(0, \sigma)$
end for

Algorithm 3 MA Process

Require: $y_0 > 0$, $t_{\max}$, $\sigma > 0$, $\psi \in [0.5, 1)$ and $\ell > 0$

for $t = 1$ to $t_{\max}$ **do**
  if $y_{t-1} > MA(\ell)_{t-1}$ **then**
    $s_t = 1$
  else
    $s_t = -1$
  end if
  $y_t = y_{t-1} \cdot \left(1 + r_t + s_t \cdot \Phi^{-1}(\psi, 0, \sigma)\right) \qquad r_t \sim N(0, \sigma)$
end for

gorithm 2 documents the RSI process, which requires a number of parameters. y_0 is the initial price and $t_{\max}$ is the simulated sample size. σ and $\psi \in [0.5, 1)$ control the volatility and conditional drift of the generated returns. $\beta \in [0, 0.5]$ defines thresholds outside which the market is overbought or oversold, and ℓ is the length parameter for the RSI indicator.

For each point in time t a state-variable s_t controls the regime or conditional drift of the price process. If the RSI of the previous period is below β (above $1-\beta$) the market is oversold (overbought) and triggers positive (negative) conditional drift, i.e., $s_t = 1$ ($s_t = -1$), otherwise the current regime is unchanged $s_t = s_{t-1}$. The price y_t is generated from conditional normally distributed returns with volatility σ, but the magnitude of conditional drift is dependent on ψ in the expression $s_t \cdot \Phi^{-1}(\psi, 0, \sigma)$, where Φ^{-1} is the inverse cumulative normal distribution function. In the limit as ψ approaches 1, the drift tends to infinity. Hence, the conditional drift is essentially a positive constant, but using the mapping of Φ^{-1} a more intuitive sense of the bias is obtained, since $\psi - 0.5$ gives its magnitude in probability terms, e.g., when $\psi = 0.55$ the probability of a positive (negative) return is 0.05 for $s_t = 1$ ($s_t = -1$).

The Moving Average (MA) process is listed in Algorithm 3 and has a similar structure to the RSI process, but instead of conditioning on the RSI indicator it employs a moving average of the price for regime switching. It simply has a positive (negative) conditional drift when the price is above (below) the ℓ-period moving average.

Figure 4 depicts samples of the two verification processes, each of which is shown with the purely random base process $y_t = y_{t-1} \cdot (1 + r_t)$, i.e., the equivalent trajectory where the conditional drift has been removed. For the first 358 samples the random base price depreciates, while the RSI process remains fairly stable due to the positive drift. Hereafter, the state becomes negative and the RSI process declines and the underlying appreciates slightly. Likewise, for the MA process the conditional drift is mostly negative, resulting in a larger depreciation compared to its purely random base where the drift is zero.

The verification processes have several convenient statistical properties. Firstly the return series do not contain any significant auto-correlations, as is the case with empirical data. Hence, traditional linear time-series models are useless in this setting. A Ljung-Box test using 20 autocorrelations cannot reject the null hypothesis of linear independence for either of the two samples depicted in Figure 4 [19]. The p-values are 0.27 and 0.78 for the RSI and MA processes, respectively.

That the processes have two different regimes, with different conditional drifts, implies that the unconditional distribution of the returns is a Gaussian mixture. Consequently, the models could easily be expanded to accommodate stylized facts from financial markets such as negative skewness and excess kur-

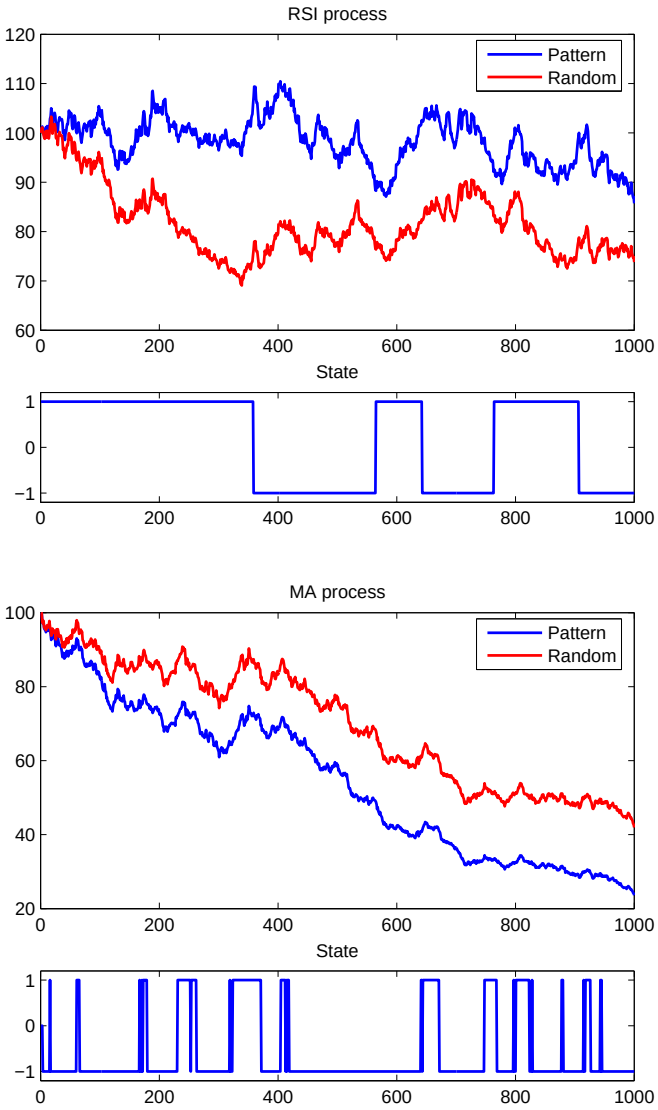

Figure 4: RSI process (top) and MA process (bottom). Each process is plotted together with its random equivalent with zero conditional drift, and s_t is depicted below.

Function	Arguments	Return Type
<, >	(price, price)	bool
<, >	(qrsi, qconst)	bool
BTWN	(price, price, price)	bool
BTWN	(qrsi, qconst, qconst)	bool
XOVER, XUNDR	(price, price)	bool
XOVER, XUNDR	(qrsi, qconst)	bool
AND, OR, XOR	(bool, bool)	bool
NOT	(bool)	bool

Table 1: Artificial market grammar.

tosis [9]. Specifically, the mean and variance of a Gaussian mixture is,

$$\mu_m = \sum_{j=1}^{k} w_j \cdot \mu_j \tag{5}$$

$$\sigma_m^2 = \sum_{j=1}^{k} w_j(\sigma_j^2 + \mu_j^2) - \mu_m^2 \tag{6}$$

where k is the number of mixtures, w_j is the probability of each mixture, and μ_j and σ_j^2 are the mean and variance of individual component j [25]. Hence, the expected information ratio of the true model is

$$IR_{\text{true}} = \kappa \frac{\Phi^{-1}(\psi, 0, \sigma)}{\sigma_m} \tag{7}$$

where κ is an appropriate positive scaling factor. In practice, however, only a noisy mapping of the true state is observed, and for this reason the expected information ratio is less than IR_{true}. Having introduced the artificial market data, the framework for evolving the trading strategies is described in the following.

Section 3 considered a Boolean universe, where the return values of the GP trees corresponded to the outputs of the programs. In this section trading strategies are evolved that take long and short positions, where a long (short) position corresponds to an output of +1 (−1). Consequently, the truth values of the trees need to be transformed to valid outputs. The transformation for the single trees is $o_t = 2 \cdot b_t - 1$, and for the dual trees it is,

$$o_t = \begin{cases} 2 \cdot b_{1,t} - 1 & \text{if } o_{t-1} = -1 \\ -2 \cdot b_{2,t} + 1 & \text{otherwise} \end{cases}. \tag{8}$$

All trees are constructed from the same grammar, which in addition to type constraints also have sematic restrictions. The grammar is documented in Table 1. It consists of Boolean operators, numeric comparators and three special functions; BTWN, XOVER and XUNDR. BTWN takes three arguments and returns TRUE if the value of the first is *between* the second and third. XOVER (XUNDR) reads *crosses-over* (*crosses-under*), and takes two numeric arguments (a,b) and returns TRUE

if $a_{t-1} < b_{t-1}$ **and** $a_t > b_t$ ($a_{t-1} > b_{t-1}$ **and** $a_t < b_t$). The terminal set comprises of the price and moving averages thereof (`price`), RSI indicators (`qrsi`) and numeric constants on the unit interval (`qconst`).[1] Moreover, Boolean terminals are also provided (`TRUE`, `FALSE`). As fitness measure, the annualized information ratio is used under the assumption that the artificial price series are sampled on a daily frequency.

4.2 Results

In this section 100 series, each consisting of 10000 samples, are generated for both the RSI and MA process. Both processes are generated using the parameters $\psi = 0.538$ and $\sigma = 0.01$. Moreover, for the RSI process $\beta = 0.3$ and $\ell = 20$, while for the MA process $\ell = 50$. The value of ψ is chosen such that under the assumption of a daily sampling frequency, IR_{true} is approximately 1.5. From a practitioner's perspective, a strategy with an information ratio of 1.5 is considered tradable and realistic.

The first 5000 samples of each series are used for evolving the trading rules, while the final 5000 samples are reserved for out-of-sample testing. Both the single and dual trees have a population size of 250, where a maximum complexity constraint of 50 nodes is imposed on each individual. Each run has a maximum of 50 generations, but is stopped prematurely if the *best-so-far* solution has remained the same for 25 generations.[2] Tournament selection is used where the tournament size is five. The probabilities of crossover and mutation are 0.9 and 0.1, respectively. The maximum mutation depth is three. Moreover, the probability of choosing a function node during crossover is 0.9.

Figure 5 provides an example of how the in-sample fitness evolves over time. At the beginning the average information ratio is approximately zero, but this is only natural due to the random initialization of the population. After ten generations the average fitness has reached a value around one, where it remains fairly stable for the remainder of the run. While the majority of the population has converged to good solutions, a significant part is clustered around zero and some individuals even have very negative performance. That individuals tend to be clustered around zero, is an artifact of the underlying data where the expected return of the *buy-and-hold* strategy is zero. Thus, trading models with little or no variation in their forecasts fall into this category. While it might appear that these individuals remain the same over time, it should be noted that this is not the case. Instead, it seems the population has reached a steady state, where the expected number of individuals in the three different groups remain the same.

Finally, the presence of individuals with highly negative performance are simply explained as negations of good phenotypes. On a genotypic level for the single trees, this can easily be achieved by adding a `NOT` function as root-node. This example merely illustrates the inner workings of the genetic programming

[1] The length parameters for the moving averages are; 10, 20, 50, 100 and 200. For the RSI indicators they are; 10, 20, 30, 40 and 50.

[2] The stopping criterion measures both fitness and complexity, such that an individual a is preferred to b, if a has greater fitness, or equal fitness and less complexity, compared to b.

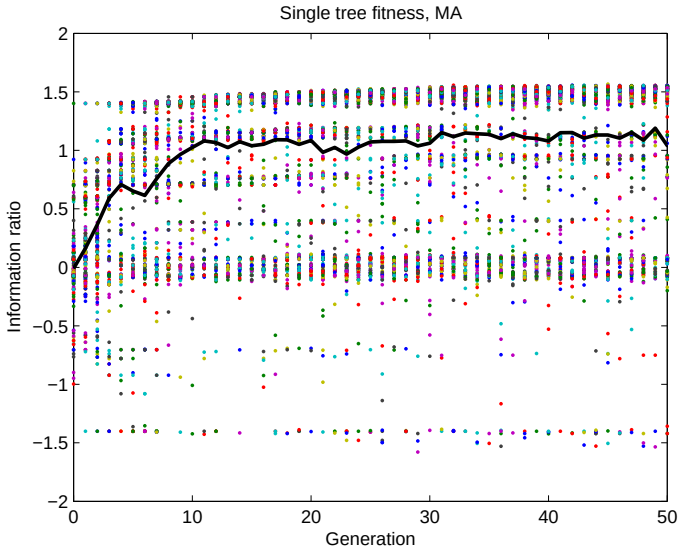

Figure 5: In-sample information ratio as a function of generation. Each dot represents the fitness of an individual, while the solid line is the average fitness of the population.

framework. However, it is the out-of-sample performance and not the in-sample fitness that essentially matters.

Figure 6 depicts kernel distribution estimates of the fitness obtained across the 100 experiments for the single and dual tree programs on both the RSI and MA processes. In addition, the information ratio distributions of the true state classifications are shown. According to the experiment settings, it is seen that the average information ratios of the true models are 1.5 in each of the four panels.

For the RSI processes the average in-sample fitnesses are 1.04 and 1.25 for the single and dual tree method, respectively. That these numbers are below the theoretical value could suggest that inadequate computational resources have been allocated to solving this complicated problem. As expected, the performance deteriorates out-of-sample. The average information ratios drop to 0.23 and 0.82 for the single and dual trees, respectively. Using a standard t-test the null of zero-mean performance is strongly rejected at all usual levels of significance ($p < 10^{-10}$). Moreover, it is found that the dual trees have significantly better out-of-sample performance compared to the single trees. The null of equal mean-performance against the one-sided alternative is strongly rejected using a two-sample t-test ($p < 10^{-10}$).

For the MA processes the results are quite different. The two representations have almost identical in-sample performance, where the average information ratios are 1.64 and 1.65 for the single and dual trees, respectively. In the test period the performances deteriorate to 1.33 and 1.18. Again, these results are highly significant ($p < 10^{-10}$). Unlike the RSI processes, the single trees fair sig-

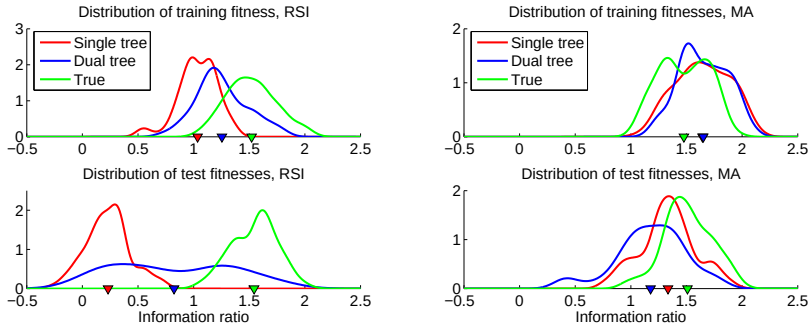

Figure 6: In-sample and out-of-sample fitness distributions for the RSI (top) and MA (bottom) process . Each panel contains the kernel density estimates from 100 experiments for the single and dual tree representation, as well as the true underlying model. The triangles mark the average fitnesses.

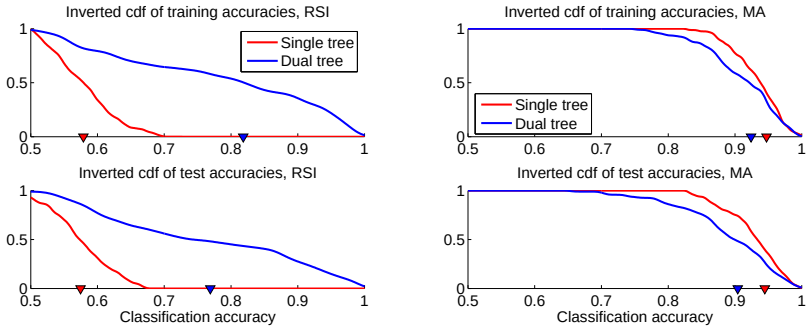

Figure 7: Non-parametric estimates for the probability that the classification accuracy is *greater* than x. The triangles mark the median classification accuracies.

nificantly better out-of-sample on the MA processes compared to the dual trees ($p = 1.45 \cdot 10^{-4}$).

To further understand these results, it is instructive to take a closer look at the phenotypic behaviors of the individuals. Using simulated data gives the luxury of knowing the true states underlying the processes. By evaluating the classification accuracies of the evolved solution, a more complete picture emerges. The classification accuracy is the proportion of periods where the forecasts are equal to the true regimes.

Figure 7 shows non-parametric estimates of the probabilities that the classification accuracies are *greater* than x. During the training period for the RSI processes, the median accuracy of the single trees is only 0.58, while the equivalent number for the dual trees is 0.82. Clearly, this explains the poor generalization of the single trees. In Section 3, a predator had successfully captured a prey when the classification error was less than 0.05. From this perspective, none of the single trees are capable of capturing the underlying dynamics of the RSI processes, whereas 22% of the dual trees do.

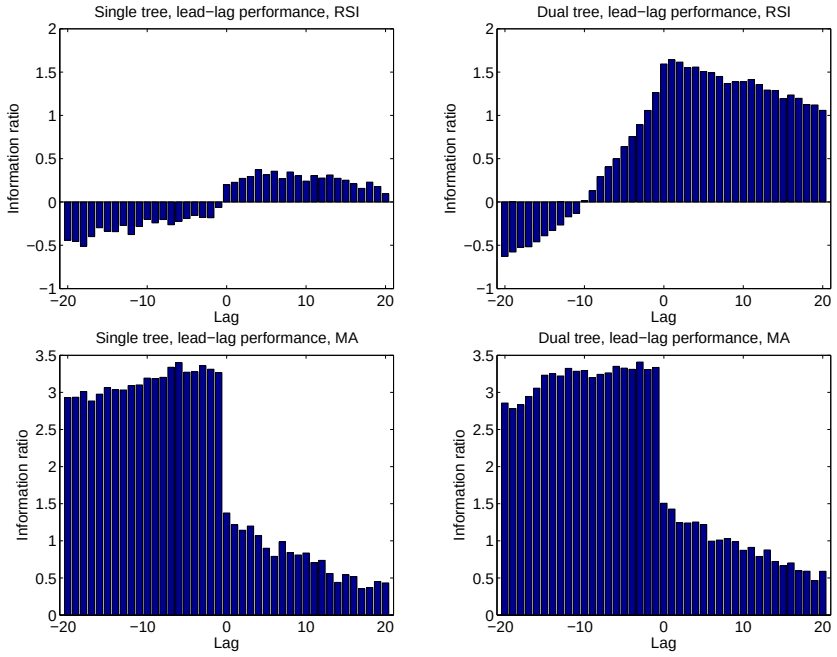

Figure 8: Out-of-sample lead-lag performances of the best individuals on the RSI data (top) and MA data (bottom). The left and the right column are the single and dual trees, respectively.

For the MA processes the results are again different: both methods capture the underlying dynamics well, and, according to criteria above, 48% and 30% of the single and dual trees succeed, respectively. Moreover, the median accuracies are 0.95 and 0.92. During the training period the single and dual trees had equivalent average fitnesses, but, using the classification accuracy as a measure, it becomes clear that the dual trees have overfitted the data slightly.

So far, the focus has primarily been on general inferences across the many experiments. However, it is instructive to consider the dynamics of individual trading strategies in more detail. Appendix B illustrates the best single tree and dual tree on the RSI processes, and the best single and dual tree on the MA processes. Three of these strategies have discovered the true underlying models, but the single tree on the RSI process has a sub-optimal classification accuracy of 0.71.

Figure 8 shows the out-of-sample *lead-lag plot* of the best individuals on their respective domains. The lead-lag plot, measures the performance of a trading strategy as the forecasts are shifted along the temporal dimension while the predictand remains fixed. It can therefore be viewed as a sensitivity analysis for the timing of a given strategy. The three strategies that have discovered the true underlying models, have information ratios around 1.5 at lag zero. As mentioned previously, this is in line with the experimental setup.

As the forecasts are lagged, the performance decays. This is intuitive since trading decisions are based on older information. However, as the forecasts are shifted forwards in time the information ratios on MA processes improve significantly, while for the RSI processes there is a deterioration in performance. The latter might seem counterintuitive, because using information from the future should imply better results, *ceteris paribus*. An explanation is found in the mean-reverting effects of the RSI processes. Consider the following scenario. There has recently been a large depreciation in the price of an asset, which causes an investor to believe that the market is oversold and consequently the asset is bought. Had the buy order been issued earlier, it would have resulted in a substantial loss due to a long exposure during the initial depreciation.

In a trending market, which the MA process represents, the situation differs. In this context, an investor interprets a recent depreciation as the beginning of a downtrend and therefore a short position is taken. Had this position been taken earlier, it would have resulted in additional gains, because the investor had a short position whilst the market depreciated.

5 Conclusion

This paper proposes a general multiple tree approach for dynamic decision problems, in which the current evaluation is contingent on the previous output of the program. The hypothesis is that this methodology is superior to a traditional single tree, where evaluation is unconditional, for temporal phenomena. The hypothesis is tested using single and dual trees in two different contexts.

The first problem considers a coevolutionary predator-prey type environment, where the prey generates a binary process which the predator seeks to imitate. In this setting there are no clear optimal strategies. Hence, a framework is developed in which the predator evolution is nested within the prey evolution. It is found that the dual trees have greater expressive capabilities, since they can capture the dynamics of the single trees when acting as predators, while evading when acting as prey. Moreover, this holds despite the maximum complexity constraint of the single trees being twice that of the dual trees.

The second problem is closer related to finance. Single and dual trees are used to evolve trading strategies on artificial financial time series. Two types of processes are considered. The first has a trending behavior, while the second exhibits mean-reversion effects. Both processes are highly nonlinear, thus rendering traditional linear time series analysis useless. Both the single and the dual trees generate significant positive returns out-of-sample. However, the single trees are incapable of capturing the true underlying nature of the mean-reverting processes, but the dual trees succeed. For the trending series, both representation are capable of capturing the underlying dynamics, but the single trees have significantly better out-of-sample performance compared to the dual trees. The single tree is nested as a special case of the dual trees, when the two trees have identical phenotypic behaviors. However, this is not easily achieved in a noisy environment, which explains the discrepancy in performance.

In conclusion, the "recurrent" multiple trees nest the single tree as a special case. Theoretically it is a superior framework, but in practice the added flexibility is not always a benefit. This depends on the nature of the problem, and is basically a manifestation of *Ockham's razor*. This paper has used artificial data, but future research will consider real financial applications.

References

[1] Allen, F. and Karjalainen, R. [1999], 'Using genetic algorithms to find technical trading rules', *Journal of Financial Economics* **51**, 245–271.

[2] Becker, L. A. and Seshadri, M. [2003], GP-evolved technical trading rules can outperform buy and hold, *in* 'Proceedings of the Sixth International Conference on Computational Intelligence and Natural Computing'.

[3] Benda, M., Jagannathan, V. and Dodhiawalla, R. [1985], On optimal cooperation of knowledge sources, Technical report, Boeing AI Center, Boeing Computer Services, Bellevue, WA.

[4] Bhattacharyya, S., Pictet, O. V. and Zumbach, G. [2002], 'Knowledge-intensive genetic discovery in foreign exchange markets', *IEEE Transactions on Evolutionary Computation* **6**(2), 169–181.

[5] Brock, W., Lakonishok, J. and LeBaron, B. [1992], 'Simple technical trading rules and the stochastic properties of stock returns', *Journal of Finance* **74**(5).

[6] Dacorogna, M. M., Gencay, R., Müller, U. A., Olsen, R. B. and Pictet, O. V. [2001], *An Introduction to High-Frequency Finance*, Academic Press.

[7] Dawkins, R. [2006], *The Selfish Gene*, 30th anniversary edn, Oxford University Press.

[8] Ehrlich, P. R. and Raven, P. H. [1992], 'Butterflies and plants: A study in coevolution', *Evolution* **18**, 586–608.

[9] Franses, P. H. and van Dijk, D. [2004], *Non-linear time series models in empirical finance*, Cambridge University Press.

[10] Futuyma, D. J. and Slatkin, M., eds [1983], *Coevolution*, Sinauer Associates.

[11] Haynes, T. and Sen, S. [1995], Evolving behavioral strategies in predators and prey, *in* S. Sen, ed., 'IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems', Morgan Kaufmann, Montreal, Quebec, Canada, pp. 32–37.

[12] Haynes, T., Sen, S., Schoenefeld, D. and Wainwright, R. [1995], Evolving a team, *in* E. V. Siegel and J. R. Koza, eds, 'Working Notes for the AAAI Symposium on Genetic Programming', pp. 23–30.

[13] Janzen, D. H. [1966], 'Coevolution of mutualism between ants and acacias in Central America', *Evolution* **20**, 249–275.

[14] Janzen, D. H. [1980], 'When is it coevolution?', *Evolution* **34**(3), 611–612.

[15] Jin, N. [2007], Constraint-based co-evolutionary genetic programming for bargaining problems, PhD thesis, University of Essex.

[16] Keith, M. J. and Martin, M. C. [1994], Genetic programming in C++: Implementation issues, *in* K. L. Kinnear, ed., 'Advances in Genetic Programming', The MIT Press, pp. 285–310.

[17] Kozhan, R. and Salmon, M. [2007], On uncertainty, market timing and the predictability of tick by tick exchange rates, Technical report, Warwick Business School.

[18] Langdon, W. B. [1998], *Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming*, Kluwer Academic Publishers.

[19] Ljung, G. M. and Box, G. E. P. [1978], 'On a measure of lack of fit in time series models', *Biometrika* **65**(2), 297–303.

[20] Mandic, D. and Chambers, J. [2001], *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*, John Wiley & Sons.

[21] Montana, D. J. [1994], Strongly typed genetic programming, Technical report, Bolt Beranek and Newman.

[22] Neely, C., Weller, P. and Dittmar, R. [1997], 'Is technical analysis in the foreign exchange market profitable? a genetic programming approach', *Jounal of Financial and Quantitative Analysis* **32**, 405–426.

[23] Rosin, C. D. [1997], Coevolutionary search among adversaries, PhD thesis, University of California, San Diego.

[24] Roughgarden, J. [1979], *Theory of Population Genetics and Evolutionary Ecology: An Introduction*, Macmillan Publishing.

[25] Trailovic, L. and Pao, L. Y. [2002], Variance estimation and ranking of Gaussian mixture distributions in target tracking applications, Technical report, University of Colorado, Boulder.

[26] Wilder, J. W. [1978], *New Concepts in Technical Trading Systems*, Trend Research.

A Relative Strength Index

The Relative Strength Index (RSI) by Wilder [26] is a popular indicator for capturing reversals. RSI is a so-called oscillator, i.e., a momentum indicator that is standardized to lie in an interval between 0 and 1 or 0 to 100. The RSI at time t with length parameter n is formally defined,

$$RSI(t,n) = 100\frac{RS(t,n)}{1+RS(t,n)} \tag{9}$$

$$RS(t,n) = \frac{EG(t,n)}{EL(t,n)} \tag{10}$$

where EG and EL are expected gains and losses computed from weighted averages,

$$EG(t,n) = \frac{\max(C_t - C_{t-1},0) + (n-1)EG(t-1,n)}{n} \tag{11}$$

$$EL(t,n) = \frac{\max(C_{t-1} - C_t,0) + (n-1)EL(t-1,n)}{n} \tag{12}$$

and C denotes the closing price. Assuming negative times prior to initialization, it holds, at time $t = 0$,

$$EG(0,n) = \frac{1}{n}\sum_{i=-n+1}^{0}\max(C_i - C_{i-1},0) \tag{13}$$

$$EL(0,n) = \frac{1}{n}\sum_{i=-n+1}^{0}\max(C_{i-1} - C_i,0) \tag{14}$$

In its original form Wilder [26] proposed the length input n of the indicator to be 14 days, where the market is said to be overbought if RSI is above 70 and oversold if it is below 30. Consequently, when the market is overbought (oversold) a sell (buy) order is issued.

B Evolved Strategies

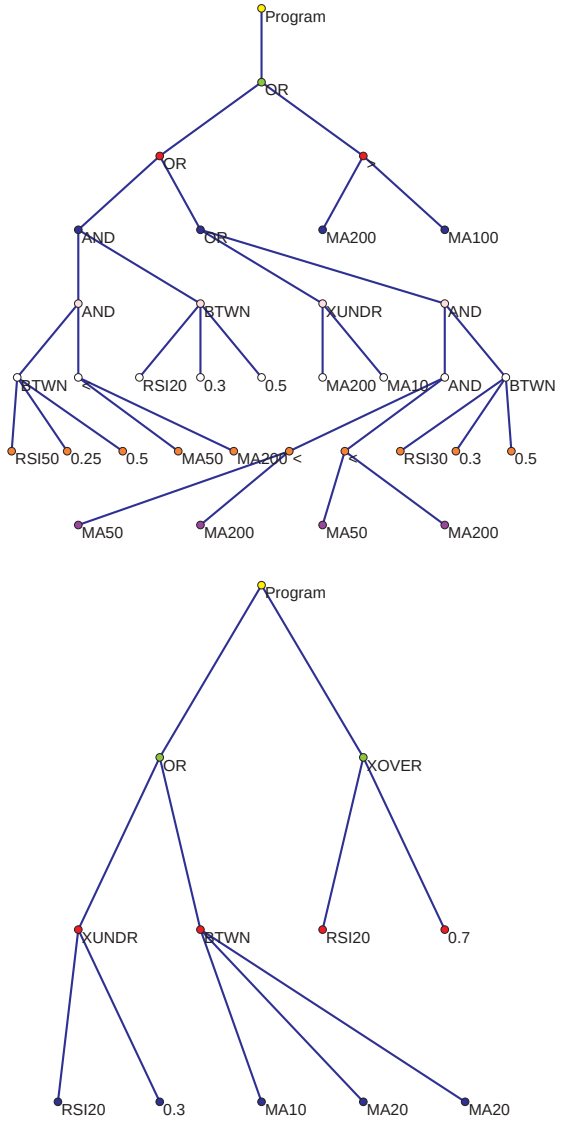

Figure 9: Single (top) and dual tree (bottom) with maximum in-sample classification accuracy on RSI processes.

Figure 9 shows the single and dual tree with highest in-sample classification accuracy on the RSI processes. The single tree has a suboptimal classification accuracy of 0.71. It approximates the mean-reversion effects of the RSI process, via some non-trivial combination of moving averages and RSI indicators. The dual tree has discovered the true model and is more interpretable. A long position is initiated when `OR(XUNDR(RSI20, 0.3)), BTWN(MA10, MA20, MA20))` is `TRUE`, while a short position is initiated when `XOVER(RSI20,0.7)` is `TRUE`. Be-

sides being the correct solution, it provides a typical example of *introns*. An intron is a non-coding region of DNA, and in genetic programming it is associated with redundant code such as `BTWN(MA10, MA20, MA20)`, which is essentially `FALSE`, and therefore irrelevant in conjunction with `OR`.

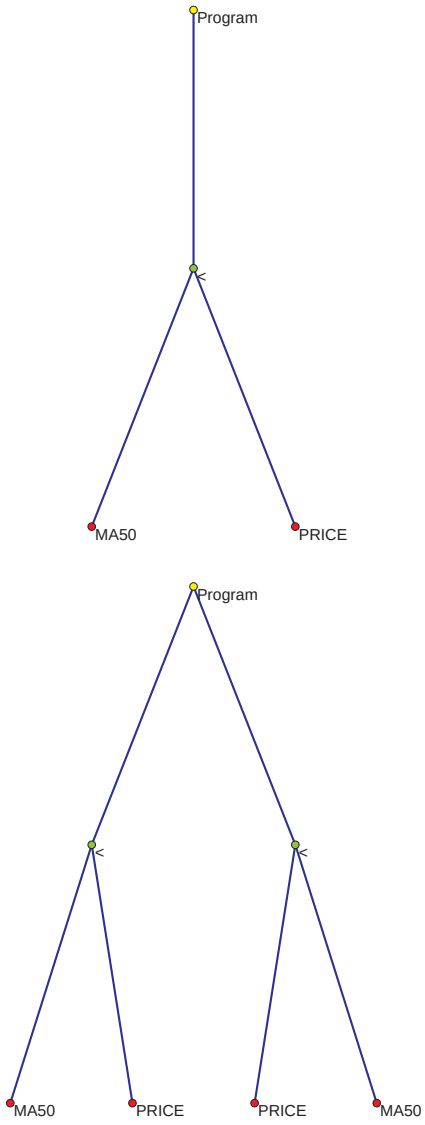

Figure 10: Single (top) and dual tree (bottom) with maximum in-sample classification accuracy on MA processes.

Figure 10 shows the single and dual tree with the highest in-sample classification accuracy on the MA processes. Both trees have discovered the true model, and have no excessive code. The two trees of the dual tree are identical, which implies that evaluation is independent of the previous program output. In effect, the dual tree is reduced to a single tree.