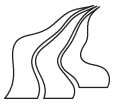
Cooperation Among Autonomous Adaptive Agents II

Group E4-119, DAT6/F10S


Aalborg Universitet
Department of Computer Science

10th June 2003

AALBORG UNIVERSITY
Department of Computer Science



Title:
Cooperation Among
Autonomous Adaptive Agents

Semester:
Spring 2003, DAT6/F10S

Project Period:
1st of Feb. 2003 to 11th of June 2003

Project Group:
E4-119

Group Members:
Michael Vengø Rydtoft
rydtoft@cs.auc.dk

Henrik Skriver Rasmussen
henriksr@cs.auc.dk

Supervisor:
Tomas Kocka
kocka@cs.auc.dk

Publications:
4 + 1 online

Number of Pages:
114

Abstract:

This report documents the work and findings on using Strongly Typed Genetic Programming to evolve cooperative behavior of autonomous agents.
We devise and implement a method for maintaining diversity allowing our multiobjective co-evolution to proceed without converging prematurely. A problem with noise is alleviated through the introduction of a modification to Competitive Fitness Sharing. In order to efficiently track progress in our co-evolutionary runs we conceive and utilize an objective measure of evolutionary progress.
The project has two primary goals. The first goal is to investigate whether cooperating agents can produce better results than non-cooperating agents solving the same problem. Tests in our extended pursuit environment with varying degrees of communication are performed. Results show that predators able to communicate evolve much better and more robust solutions against difficult prey than predators unable to communicate.
The second primary goal is to design and evaluate a method to avoid the phenomenon of disengagement which can occur in competitive co-evolution. Our method - named the Interleaved Approach - is compared with approaches found in literature and determined to be superior in the used environment. The Interleaved Approach is also compared with the regular co-evolutionary approach as well as single evolution. The Interleaved Approach outperforms both approaches.

Preface

This report is written by project group E4-119 on DAT6/F10S at the Department of Computer Science at Aalborg University during the spring semester of 2003.

Chapters, sections, subsections, figures and tables are numbered sequentially. Appendices are numbered alphabetically.

All references to literature in this report is referenced as: "Author Surname [43]", and all cross-references are of the form: "figure 5.2" or "section 5.2".

Acknowledgments

We would like to thank our supervisor Tomas Kocka for his great interest and involvement in this project.

We acknowledge the software package AToolBox as property of Kasper Kau (kasper@kau.dk) and thank him for its use in the building of the VisualGP application.

We acknowledge Sean Luke as the author of the open source ECJ system and the license agreement under which we use the ECJ system.

Henrik Skriver Rasmussen

Michael Vengø Rydtoft

Contents

Chapter 1

Evolutionary Algorithms

This chapter presents different approaches to evolutionary algorithms. Evolutionary algorithms are related to the family of greedy algorithms. Greedy algorithms are introduced in section 1.1. Section 1.2 presents varieties of evolutionary algorithms. Sections 1.4 and 1.5 introduce different methods for selection and breeding respectively. Section 1.6 introduces methods for tree generation and section 1.7 describes various problems commonly occurring when evolutionary algorithms are employed. Finally section 1.8 describes topics related specifically to co-evolution.

Evolutionary Algorithms seek to utilize the Darwinian concept of natural selection. This concept can be observed in nature where biological structures which are more adept at coping with their environment survive and reproduce at higher rates than less adept structures. Evolutionary algorithms use this concept to guide their search through very large search spaces. Basically they work by creating a set of individuals which offer potential solutions to a problem and measuring the degree to which these individuals are successful at solving the problem. This degree of success is referred to as fitness. The fitness is used as basis for selecting a subset of the individuals which are subsequently altered or "bred" to produce new individuals with new solutions. Thus the search is performed by moving to new solutions within the search space in the directions in which improvement is expected. This loop repeats until either a sufficiently fit solution has been produced or a predefined maximum number of iterations has been reached. The basic evolutionary loop can be seen in figure 1.1.

1.1 Greedy Algorithms

The general idea behind a greedy algorithm is to myopically always select the solution with the highest immediate gain. In EAs greediness can manifest itself through the concept of selection pressure - i.e. how high is the pressure to select the (currently) best individuals for the mating pool. In a maximally greedy EA the mating pool would always be filled solely with copies of the very best individual.

To further illustrate the idea of greediness we present a simple greedy algorithm

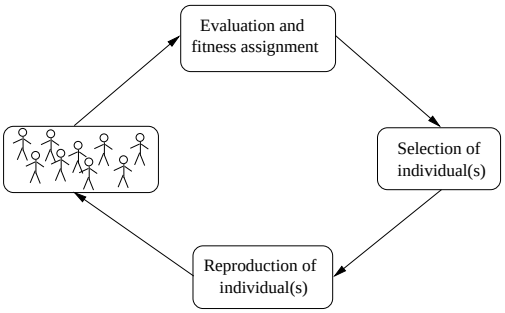


Figure 1.1: Basic evolutionary loop.

to solve the problem of finding a dominating set for a graph. A dominating set in graph **F** is a set of vertices in **F** which have all the other vertices in **F** as neighbors.

A greedy algorithm for finding a dominating in graph **F** set is:

- Initialize set **S** to be the empty set.
- Select the unmarked vertex in **F** with highest degree (i.e. most neighbors), mark it and add it to **S**.
- If **S** constitutes a dominating set terminate. Else repeat from step 2.

Note that this algorithm is not guaranteed to find the smallest possible dominating set.

1.2 Genetic Algorithms and Genetic Programming

Genetic Algorithms

In Genetic Algorithms (GAs) individuals are represented by fixed-length bit strings which encode the solution offered by each individual. When evaluated the bit string is interpreted in a predefined manner and fitness is assigned based on the capability of the individual. As the bit strings are generated pseudo-randomly, measures must be taken to ensure that the interpreter is generic enough to accept any kind of input. For instance three bits may be reserved to represent an integer between zero (binarily encoded 000) and six (binarily encoded 110) inclusive. The interpreter should be able to cope with instances in which the three bits are set to 111 corresponding to the integer 7. A potential drawback of using GAs is that designing the binary encoding of the behavior of individuals can be a comprehensive task.

Genetic Programming

One problem associated with Genetic Algorithms is the fixed length of the bit strings used to encode solutions. In Genetic Programming (GP) solutions are represented by variable length symbolic expressions or parse trees of varying shape and size.

This effectively extends the search space of possible solutions since often an infinite number of parse trees can be generated as potential solutions to a problem. In practice a maximum tree depth is usually defined in order to avoid too large solutions. A GP parse tree specifies a solution expressed in a particular language. This language is the union of a function set $\mathcal{F}$ and a terminal set $\mathcal{T}$. The leaves of the parse tree consist of members from $\mathcal{T}$ whereas the inner nodes are members from $\mathcal{F}$. In other words, what distinguishes $\mathcal{F}$ and $\mathcal{T}$ is that members of $\mathcal{F}$ have children whereas members of $\mathcal{T}$ do not.

When constructing parse tress we have to make sure that the trees are syntactically correct. For instance passing a boolean data type on to an arithmetic function used to compute the sum of two integers causes problems. This problem is similar to the problem in GAs in which all possible bit strings must be interpretable. One way to make sure all generated trees are syntactically correct is to use STGP which is introduced in section 1.2. Another way is to require that $\mathcal{F}$ and $\mathcal{T}$ satisfy a property of closure (see Koza [30]). This property can be expressed using $\mathcal{F}$ and $\mathcal{T}$.

The Closure Property:
Any function in $\mathcal{F}$ shall be able to accept as its argument any value and data type that may possibly be returned by any function in $\mathcal{F}$, and any value and data type that may possibly be assumed by any terminal in $\mathcal{T}$.

To ensure closure all functions are adapted to accept all possible data types. For instance an arithmetic function can interpret the boolean values "true" and "false" as respectively 1 and 0. Likewise, protected versions of some standard operators may be defined. A common example is protected division which allows division by zero.

Strongly Typed Genetic Programming

Strongly Typed Genetic Programming (STGP) is an extension of GP which effectively eliminates the before-mentioned closure constraint. This is done by requiring that each function specifies precisely the data types of its arguments and returned values. Having such a specification for each function STGP ensures that all generated parse trees or symbolic expressions satisfy the constraint that all arguments passed to functions are of the correct type.

Imposing such structural requirements reduces the search space of possible solutions. Montana [40] describes a problem with a terminal set of size two and a function set of size ten. With a maximum tree depth of five the size of the search space for this problem is 10^{11} using STGP while the size of the search space for the same problem using GP is 10^{38}. This significant reduction makes the problem more manageable using STGP.

1.3 Theory Behind Genetic Algorithms

Holland's Schema Theorem in Holland [22] mathematically characterizes the evolution over time of a GA population. As the name suggests the theorem is based on the concept of schemas. While a GA on the surface processes binary strings Holland states that it implicitly processes schemas which represent similarities between the binary strings. A schema is defined as a string of length L which is the same

length as the strings in the population. Each position in the schema consists of a symbol from the alphabet 0,1,*, where * represents a "don't care" character which is essentially a wildcard. A schema thus represents the set of binary strings whose corresponding bit-positions are of identical value to the 0 and 1 bits in the schema. Depending of the number of "don't care" characters in a schema it can represent up to 2^L binary strings where L is the string length. This number corresponds to the entire search space for the given problem. Schemas have two properties, namely their order and their defining length. The order of a schema is the number of fixed bit positions in a schema and the defining length is the distance between the outermost fixed bit positions in the schema.

The Building Block Hypothesis
Building blocks are a subclass of schemas and are defined as low-order, short defining-length, highly fit schemas, where the fitness of a schema is defined as the average fitness of the solutions it represents. A given solution can thus contain several building blocks. Building blocks can be seen as templates of solution parts. When combining such solution parts they form the total of a GA's solution to a given problem. The building blocks are interesting because of the following observation. The selection process chooses solutions with above average fitness for breeding. Therefore solutions which are members of highly fit schemas are selected more frequently than solutions which are members of less fit schemas. Furthermore, the crossover operator less frequently disrupts schemas with shorter defining length and the mutation operator less frequently destroys low-order schemas. The property of the crossover operator becomes obvious when comparing the two building blocks ****10 and 01**10*11 where the latter is less likely to be transferred undisrupted to a new individual than the former due to its longer defining length. Likewise with the mutation operator it is clear when comparing the two building blocks 1***01 and 1101*1 that it is less likely that the latter will survive mutation undisrupted due to its higher order. Therefore, the highly fit, short defining-length, low-order schemas named building blocks will gain and grow from the breeding process from generation to generation. This is known as the *Building Block Hypothesis*. GAs can thus be said to process such useful schemas or building blocks instead of strings. Holland [22] estimates that while a GA processes n strings each generation, it processes n^3 building blocks. He calls this *implicit parallelism*. This is naturally a very desired property in a search method.

The Schema Theorem
The Schema Theorem (see equation 1.1) states that building blocks grow exponentially over time while below average schemas decreases at a similar rate (Holland [22]). The Schema Theorem calculates an estimation of the instances of the schema s in the next population m(s, t+1) in terms of m(s, t), where m is the number of instances of s and t is the generation. The number m is expressed as the product of the expected number of selections of s and the survival probabilities. The survival probabilities are the probabilities with which the selected schemas remain unchanged during the processes of single-point crossover and mutation. The GA assigns a probability of selection to each string directly proportional to its fitness. The schema s can therefore be expected to be selected m(s, t) $\cdot$ $(\frac{f(s)}{\overline{f}})$ times, where $\overline{f}$ is the average population fitness and f(s) is the average fitness of those strings in the population that are elements of the schema s. P_c is the probability that single-point crossover will be applied on the individual and is used to calculate the probability

that single-point crossover will destroy the schema. P_m is the probability that an arbitrary bit of an arbitrary solution will by mutated and o(s) is the number of defined bits in the given schema. Finally d(s) is the distance between the leftmost and rightmost defined bit in s and l is the length of the strings in the population. The expression is an inequality due to the fact that a schema destroyed as a consequence of single-point crossover may be joined with a similar schema and through this join regain its original form.

$$m(s, t+1) \geq m(s, t) \cdot \frac{f(s)}{\overline{f}} \cdot (1 - p_c \frac{d(s)}{l-1}) \cdot (1 - p_m)^{o(s)} \tag{1.1}$$

The Schema Theorem can be roughly interpreted as saying that the more fit schemas will tend to grow in influence, especially schemas with a small number of defined bits, i.e. a high number of *, and especially when these defined bits are near each other within the bit string. However, the Schema Theorem is accused of being simplistic in its estimations and its description of the behavior of a GA (see Mahfoud [38]). The values f(s) and $\overline{f}$ do not stay constant through evolution as assumed since fitnesses tends to shift significantly. Furthermore, the Schema Theorem only takes into consideration schema losses and not schema gains as a consequence of the genetic operators even though crossover and mutation can construct schemas. Despite the simplifications the Schema Theorem describes interesting aspects of the behavior of GAs, e.g. that higher mutation probabilities increasingly disrupt higher order schemas while higher crossover probabilities increasingly disrupt schemas with higher defining length.

The literature on the subject of analyzing and describing the behavior of both GAs and GP is vast and much work is still being done within the area. Since the goals of this report is not of pronounced theoretical character we will not go into the most recent theories but only state that theory is a long way from being able to completely express the behavior of GAs and GP although it provides valuable insight.

1.4 Selection Methods

In order for evolution to occur individuals from one generation are chosen to serve as "parents" of individuals in the next generation. In general the individuals which best serve as parents for the next generation are those with good genetic structures. In spite of this it is not always beneficial to simply select the very best from a generation. Consult section 1.7 for elaboration on this issue. Several selection methods with varying degree of selection pressure have been proposed. This section will describe the most commonly used. All these selection methods favor individuals with higher fitness and therefore have the property and purpose to increase the average fitness of a population.

Fitness Proportionate Selection
For each individual i from a population of size S the probability of selection P_i is directly proportional to its fitness F_i. P_i can be calculated as

$$P_i = \frac{F_i}{\sum_{j=1}^{S} F_j} \tag{1.2}$$

Rank Based Selection
When using Rank Based Selection, selection is based on the rank of the fitness rather than its numerical value. The best individual is awarded rank N where N is the size of the population. Denoting the rank of individual *i* as R_i the probability P_i of selecting the individual is:

$$P_i = \frac{2 \cdot R_i}{N(N+1)} \tag{1.3}$$

Rank selection reduces the potentially dominating effects of having a single super individual with a numerically high fitness. In the event of a group of individuals with closely clustered fitness values Rank Based Selection exaggerates the difference favoring the individuals with higher fitness.

Tournament Selection
In Tournament Selection a group of predetermined size[1] is chosen randomly from the population and the one with the better numerical fitness is selected. Tournament Selection is not as sensitive to the fitness values as other selection methods besides Rank Based Selection since it is the ranking of the individuals which determines the outcome of the selection. Note that when using this method the worst individual from a generation has no chance of being selected. Tournament Selection is the selection method we will use in our work. Increasing the Tournament Size also has the effect of increasing the selection pressure. This makes the Tournament Size an attractive and easily adjustable parameter.

1.5 Breeding Methods

The evolutionary step in which the selected individuals are altered and introduced into the next generation is known as breeding. The most common breeding methods are explained below.

Crossover
The Crossover operator creates variance in the population by producing two new individuals from two old. Crossover can be nicely illustrated using GP parse trees. For each of the two trees one random node is selected and the subtrees below these nodes are swapped to produce two new individuals. Figure 1.2 illustrates this. Some crossover variants use more than one crossover point per tree. The crossover operator is often used more frequently than the other operators during an evolutionary run. Note that when STGP is employed the nodes at the crossover points must have the same return types. GAs differ from other search methods mainly by their assumption that individuals considered to be good are so because they contain some important good parts and the more good parts an individual contains the better it is. Crossover is a way to migrate possibly good parts to new individuals. It is assumed that applying crossover on two average individuals, containing only a few good parts each, will produce one more fit individual and one less fit individual. To be successful it is required that the good parts are to some extent independent and local in the individual and that the parts positively influence the

[1] This size is referred to as Tournament Size.

fitness. I.e. the good parts are contained within limited subtrees which are migrated completely with the crossover operator and these subtrees influence the new more fit individual to score better fitness due to its additional functionality.

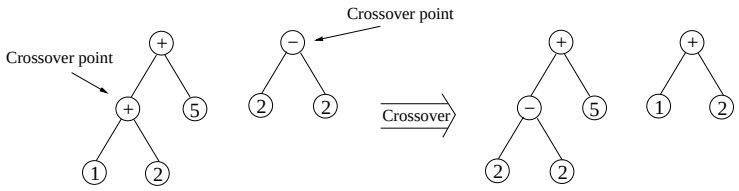


Figure 1.2: Crossover - note that the root node can be selected as crossover point.

Mutation
Mutation creates variance by mutating one part of an individual into something new. Using the GP parse tree as an example a mutation point is selected. The subtree emerging from this point is replaced by a new randomly generated subtree. This is illustrated in figure 1.3. Like with the crossover operator variants exist which use multiple mutation points. When using STGP the new randomly generated subtree will have same return value as the subtree it replaces. Note that the mutation operator has the property that it can introduce new genetic material into the population during evolution. It can thus be used as a simple means of maintaining diversity in the population.

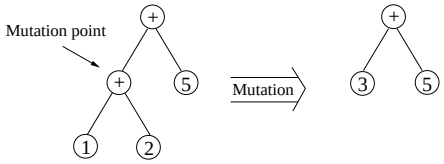


Figure 1.3: Mutation - the terminal '3' is inserted as new subtree.

Elitism
Elitism is a method (see Mahfoud [38]) which ensures that the best individual or individuals in the current population survive from generation to generation. Basic elitism copies the best individual(s) from the current population to the next population without altering them. In principle any standard selection method can be made elitist.

1.6 Tree Generation Methods

The initial generation of individuals in an EA population is randomly generated. As we are using STGP we will in this section briefly describe the FULL and GROW methods. These are two common approaches used to generate trees. These two methods are often both used when creating an initial population. The combination of using both, each of them used to create 50% of the initial trees, is so popular that

it has its own name. It is referred to as the Ramped Half-and-Half and is used in our work.

FULL
The FULL method generates trees for which the length of every non-backtracking path from the root node to any leaf is equal to some specified maximum depth. It does this by continuously selecting members from the function set $\mathcal{F}$ as nodes in the tree. When the maximum depth for a path is reached a member from the terminal set $\mathcal{T}$ is selected. The shape of the trees generated when using the FULL method is affected by the number of arguments taken by functions in $\mathcal{F}$. If the majority of functions have many arguments the generated trees are very wide. The FULL method is not influenced by the relative sizes of the sets $\mathcal{F}$ and $\mathcal{T}$.

GROW
The GROW method generates trees of varying shape and size. As with the FULL method a maximum depth is specified and no non-backtracking path from the root node to a leaf exceeds this depth. The path may, however, be shorter. Pseudocode for the GROW algorithm can be seen in table 1.1. As opposed to the FULL method the GROW method is influenced by the relative sizes of the sets $\mathcal{F}$ and $\mathcal{T}$. If the set $\mathcal{F}$ has significantly more members than $\mathcal{T}$ the algorithm behaves similarly to the FULL algorithm. This happens because nodes are selected from the union of the two sets and function nodes will have a high chance of being selected until the maximum depth is reached. If the terminal set $\mathcal{T}$ has many more members than $\mathcal{F}$ an abundance of short trees are generated as the chance of terminating branches early will be high. In some implementations a minimum depth is also specified. Thus terminal trees which are almost always useless and very small trees can be avoided.

1.7 Common Evolutionary Problems

In evolutionary algorithms the fitness of a population increases over time as the search is guided by a biased selection of more fit individuals. Choosing the right selection pressure is imperative in ensuring progress in the evolution. At the same time the genetic diversity of the population should be maintained in order to avoid stagnation in local optima. This section will discuss these two problems as well as possible solutions.

1.7.1 Convergence

One of the forces of an EA is its ability to guide the search for the optimal solution as opposed to a brute force search. The search is guided using selection biased by fitness. Individuals with high fitness scores will thus be selected more often for breeding and future generations will contain an increasing amount of such individuals. Additionally these individuals will be selected frequently for crossover with themselves so that many crossovers will be incestuous. The population will at some point contain so many copies or slight variations of these individuals that the population is said to have converged to some solution. In figure 1.4 individuals are depicted on a curve illustrating the fitness landscape. The guided search is taking

```
GROW Algorithm

S = functions F ∪ terminals T
d_max = maximum depth

grow(depth d_max) {
  if (d = d_max) {
    return random terminal ∈ T
  }
  else {
    get random element s ∈ S
    if (s ∈ T)
      return s;
    }
    else {
      for (each argument a of s) {
        a = grow(d+1);
      }
    return s;
    }
  }
}
```

Table 1.1: The GROW algorithm

place on several peaks where a number of individuals are climbing the peaks to reach the top. Once all individuals are located on the top of the highest explored peak the evolution has converged. The ability to converge ensures that a solution will be found at some point in the evolution and is thus a desired property. However, if the evolution converges to a (globally suboptimal) local optimum, which is the case if the evolution converges to the leftmost inhabited peak in figure 1.4, the evolutionary run has converged prematurely. The problem with premature convergence is to escape from this area of the search space and start exploring another one because all the individuals are almost identical and some breeding methods are rendered useless.

One major cause of converging prematurely is the concept of elitism which is also an important factor if convergence is to be ensured. Elitism occurs when the solutions with the highest fitness scores are copied directly into the next generation without modifications. Therefore elitism ensures that a globally optimal solution is saved once it has been found. However, also suboptimal solutions are saved if they are currently the best scoring solutions and the population will as described tend to converge towards these solutions.

Since it is very difficult to escape from a local optimum after having converged prematurely other measures are taken to prevent this phenomenon. The tournament size can be decreased when using tournament selection and the degree of elitism can be decreased. These are two of the factors which can help to decrease the selection pressure which again can help avoiding premature convergence. However, if

the selection pressure is too low the evolution may converge never or very late. If premature convergence has occurred mutation can guide the search in new directions. Often the evolution will re-converge and the situation is status quo, Koza [30]. If crossover is applied on two identical solutions with different crossover points the results can be very different from the parents. Another and more comprehensive method for avoiding premature convergence is to maintain diversity amongst the solutions in the population. The evolutionary search can proceed in many directions and avoid being locked at one place. This situation is illustrated in figure 1.4 where several peaks are being explored simultaneously. The following section introduces several methods for maintaining diversity.

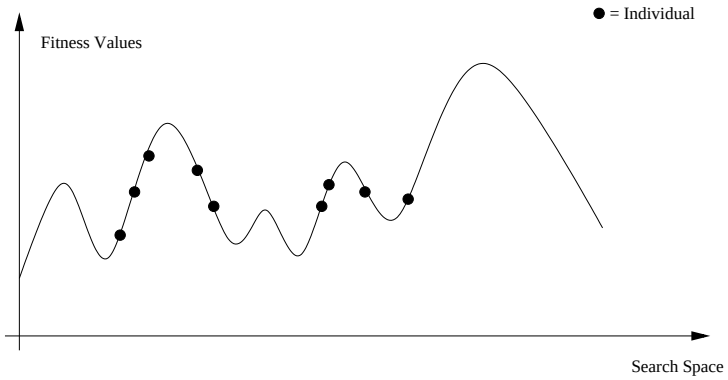


Figure 1.4: Individuals from a population climbing several peaks. If the individuals are climbing only the middle peak the population will converge prematurely. When diversity is maintained several peaks can be climbed simultaneously.

1.7.2 Maintaining Diversity

By diversity of a GP population we mean the extent of structural uniqueness among the parse trees of the individuals in the population. Keller [27] states that diversity drops during the runtime of an evolutionary algorithm as clusters of individuals build up because of fitness based selection mechanisms. If diversity is lost too soon the evolutionary search may converge to a local optimum. Generally speaking a tradeoff exists between exploration of new solutions and exploitation of already discovered material. Naturally the maintenance of diversity is most important for complex search spaces with multiple local optima. Several methods exist which assist in maintaining diversity. We will introduce some of these below. Sharing and newer crowding methods are so-called niching methods which promote the formation and maintenance of stable subpopulations in EAs (see Mahfoud [37]).

Crowding
Initial crowding techniques were introduced by De Jong [26] as a generalization of a similar method called pre-selection. Crowding inserts new elements in a population by replacing similar elements in the population with them. To make such a replacement an individual is compared to a random subpopulation of m individuals where m is known as the *crowding factor*. The individual from this subset with

the highest similarity is chosen as replacement. Over time this method tends to spread individuals over the more prominent peaks of the search space. As evolution progresses an increasing amount of individuals are similar to each other and the replacement of individuals with similar individuals tends to maintain diversity creating room for distinct subspecies within the population. Today De Jong's method is not considered a niching method as it is apparently incapable of consistently maintaining more than two peaks of even a simple multimodal function. In 1995 Mahfoud [38] presented a modified version of the crowding method called deterministic crowding which has better niching properties.

Sharing
Sharing methods are characterized as methods which require similar population individuals to share fitnesses. To induce such sharing a sharing algorithm typically alters the fitness of an individual based on its proximity to other population members. Specifically the shared fitness f' of individual i can be calculated as follows:

$$f'(i) = \frac{f(i)}{\sum_{j=1}^{S} sh(d(i,j))} \tag{1.4}$$

In the equation f designates the old fitness, S the population size and sh the sharing function which is a function of the distance d between individual i and j. sh returns 1 if the individuals are within some specified threshold of each other and 0 otherwise.

The idea behind the sharing approach is to avoid situations in which many or all individuals inhabit the same peak of the fitness landscape in which they are evaluated. Note that the smaller and thus possibly more local a peak in fitness is the fewer individuals it can "accommodate". A variation of this method is used in our work. Please consult section 1.8.1.

Island Approaches
In an island approach the population is split into several smaller subpopulations which are evolved independently from each other. Island approaches were originally conceived to enable distribution of the often computationally taxing search problem to several computers or CPUs. The approach has the attractive side effect that it can help to maintain diversity as the population is split. The subpopulations are typically not completely isolated. Usually a limited number of individuals are allowed to migrate from one population to another according to some predefined migration topology. Figure 1.5 which is borrowed from Holm [23] shows some commonly used migration topologies.

The assumption is that each subpopulation will investigate its own part of the search space converging to some local optimum in that area. If migration is performed on individuals of high fitness the population as a whole should converge to the best optimum of all explored optima. If migration is not used excessively the islands will help to prevent the entire population from converging quickly to suboptimal solutions.

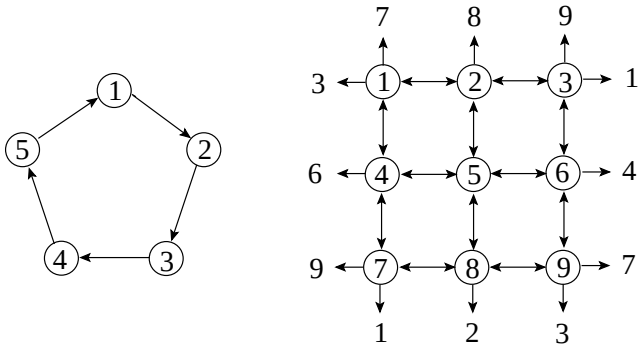


Figure 1.5: Island migration topologies. The nodes represent islands and the directed edges migration paths. The left topology has a ring structure whereas the right resembles a toroidal mesh.

1.8 Co-evolution: Problems and Solutions

The type of co-evolution employed in our work – in which the fitness landscapes of two or more populations of different species are coupled – is referred to as competitive co-evolution. The term co-evolution (non-competitive) is used for evolutions in which one or more populations of individuals from the same species are evolved and evaluated by competing amongst each other. Many of the same types of problems occur in competitive and non-competitive co-evolution. This section explains several general theories, methods and techniques from the literature used to alleviate such problems. Additionally we present our own proposals for possible solutions to some of the problems.

When competitive co-evolution is employed all competing populations are evolved in a parallel manner and the different populations are evaluated using members from the competing populations. Competitive co-evolution has been successfully applied to diverse problems, see for instance Sims [52], Floreano [13] and Reynolds[44]. Competitive co-evolution has been applied to the pursuit environment by Haynes [16] with discouraging results and by Cliff [6] where competitive co-evolution succeeded in producing good pursuers and good evaders. In Cliff [6] neural networks were used as opposed to Haynes who used genetic programming.

When individuals from one population are evaluated, i.e. when their fitness is being measured, against individuals from the opposing populations they are referred to as *students* and the individuals they are measured against are referred to as *teachers*. This terminology will be used throughout the remainder of this report.

Several problems commonly occur when competitive co-evolution is employed. One such problem concerns the way fitness is assigned. As members of one population is used as teachers for another population some members of the teaching population will only be beaten by few members of the student population. If these students beat few other teachers they will normally receive poor fitness compared to students which beat many teachers. Thus they will likely become extinct over time. However, these few student population members possess some useful trait

which enable them to beat the teachers few others can beat and they are thus worth preserving. Section 1.8.1 contains a fitness sharing method which deals with this problem. Another problem is the sheer number of evaluations necessary if each student is to be evaluated by each possible teacher. For two populations each of a thousand individuals one million evaluations are needed for all possible student teacher pairs. Section 1.8.2 describes a solution to this problem called *Shared Sampling* which locates a good subset of individuals to be used as teachers. This set is called the *Teacher Set*. Section 1.8.3 describes the concept of *Hall of Fame* which is used to avoid oscillatory patterns in the evolution where solutions to certain problems are forgotten and later rediscovered. The evaluation of individuals can also be problematic. Section 1.8.4 describes the *Lack of Objective Fitness* problem. Finally section 1.8.5 describes how the HoF concept can be combined with Multi Objective Evolutionary Algorithms.

1.8.1 Fitness Sharing and Competitive Fitness Sharing

The following simple fitness assignment method could be used to assign fitness to a student individual when it is being evaluated against a number of teacher individuals: A fitness value is assigned when an evaluation has occurred and all fitness values an individual attains against the teachers is summed to produce one overall fitness. If fitness is assigned as just described where fitness is based on the sum of scores across all the evaluations a problem arises. An individual which beats a high percentage of its opponents will using the simple fitness assignment receive a higher fitness than an individual which only beats few opponents. Intuitively this seems like a good property. However, there may exist an individual which is only able to beat one particular opponent which no other individual can beat. In this fitness model this individual will receive a low fitness and risk extinction due to selective pressure. This individual contains genetic material which should not be lost since it may be useful in later generations. A fitness assignment method which increases the survival chance of rare but important individuals is needed.

Fitness Sharing (see section 1.7.2) is a fitness assignment method which takes into account similarities among individuals. An individual is penalized for having a common fitness by dividing its fitness with the sum of its similarities with each other individual in the population. A variant of this method called Competitive Fitness Sharing was proposed in Rosin [46]. When using Competitive Fitness Sharing each teacher is viewed as a separate resource of fitness to be shared among those students which can beat it. The fitness assigned to a certain student agent beating a teacher j in the set of teachers X is $\sum_{j \in X} \frac{1}{N_j}$, where N_j is the total number of students in the population beating teacher j.

This method rewards students which are able to beat teachers only few other students can beat even though they may not be able to beat many teachers. This is as mentioned desirable because the genome of such students is likely to contain important and rare substructures. Without the reward these individuals and their important substructures have a higher chance of becoming extinct before being able to pass the useful substructures on to new individuals. Thus competitive fitness sharing helps ensure the survival of important but rarely represented students. Additionally it helps to maintain diversity in the population. Rosin [46] concludes that

the method substantially improves performance on their suite of simple test problems[2].

1.8.2 Shared Sampling and the Teacher Set

When using co-evolution and competitive co-evolution it is not apparent which teacher individuals students from a population should be evaluated against. In single species evolution individuals are often evaluated against the same single opponent. This opponent may be the environment itself or a non-evolving agent. The latter was the case in our simple pursuit environment (detailed in Rydtoft and Rasmussen [43]) where all predators were evaluated against the hard-coded randomly moving prey. In co-evolution where individuals are all of the same species and measured by evaluation against each other one could evaluate every individual against every individual of the same generation or - even better - against every individual from the same generation and every past generation. For a population size of X individuals the first approach requires X^2 evaluations per generation while the second approach requires $X^2 \cdot Y$ where Y is the number of generation elapsed so far. Such numbers of evaluations are intractable for most problems. Only for very small population sizes or very few generations of evolution would the computational effort required be tolerable. In competitive co-evolution where two populations of different species compete the same problem exists.

Instead of testing all members of a student population against all members of the newest teacher population the computation time can be reduced drastically by only testing the student population against a limited sample of the teacher population. Shared Sampling is a method for choosing such a limited sample set of teacher individuals. Shared Sampling as described in Rosin [47] uses information available from the previous generation about the individuals who are to be teachers to obtain a more challenging teacher set than if they were picked randomly. The set of teachers produced by Shared Sampling is named the Teacher Set[3].

Shared Sampling chooses teachers which challenge as many segments of the student population as possible in an attempt to create the ideal teaching set. Note that the individuals which are used to decide this are members of the opponent Teacher Set which the population being sampled from is evaluated against. In this section OppTeacher refers to individuals from the opposing Teacher Set. The ideal teaching set is a set of individuals that work together by having, for each opponent, at least one individual in the set capable of beating that opponent. Shared sampling attempts to create this set by use of the Competitive Fitness Sharing also used for selection. The algorithm for selecting the Teacher Set is shown in table 1.2. Note that the Shared Sampling proposed in Rosin [47] is intended for a single species environment and the algorithm shown is adapted to our environment.

The algorithm works by selecting new members which have maximal competitive shared fitness within the population until the desired Teacher Set size is reached. Thus any new member is chosen to be one which competes well against OppTeachers which other members of the Teacher Set compete poorly against.

[2]These problems consisted of using competitive co-evolution strategies for the simple games Tic Tac Toe, Nim and Go.

[3]It is also often referred to as the Sample Set.

```
Teacher Set = ∅
For (all OppTeachers i) {
      beat[i] = 0
}
While (Teacher Set not full) {
   For (all possible teachers) {
      set teachers samp_fit = 0
      For (all OppTeachers which j beat) {
         samp_fit[j] += 1/(1+beat), beat is OppTeacher's beat value.
      }
   }
   Add possible teacher j with highest samp_fit to Sample-Set
   For (all OppTeachers i which j beat) {
         beat[i]++
   }
} End of While loop
```

Table 1.2: The Shared Sampling algorithm

Each OppTeacher has a Beat variable which is used to register the number of individuals in the Teacher Set which beat the OppTeacher. NumOfPossibleTeachers is the number of individuals from which we sample and which are not present in the Teacher Set. Each possible teacher j has a samp_fit[j] given by equation 1.5 where k is the k-th OppTeacher in its list of individuals it can beat.

$$samp\_fit[j] = \frac{1}{1 + beat[k]} \tag{1.5}$$

1.8.3 Hall of Fame

A possible difficulty of co-evolutionary approaches is caused by intransitive superiority (see Watson [55]). Basically intransitive superiority means that the facts that agent A beats agent B and agent B beats agent C do not imply that agent A will be able to beat agent C. An example of clear transitivity is equality for numbers while an example of clear intransitivity is the beat relation used in the game "Rock, Paper and Scissors" in which rock beats scissors, scissors beat paper and paper beats rock. In addition to lacking transitivity the rock-paper-scissor game contains a dominance loop. Except for such well understood and simple relationships transitivity or lack thereof can be very hard to detect. It is hypothesized to cause cycles in the evolution causing the evolutionary algorithm to visit parts of its search space more than once because each single step yields an apparent improvement when viewed myopically. The following simple example shows how such a cycle can occur: Consider two populations of separate species $spec_A$ and $spec_B$ each containing a single

individual A and B. Each individual is evaluated against the other. The following sequence of events creates a cycle in the search space for each species even though each single step yields an improvement for the individual causing it.

```
1  A beats B
2  B evolves into B'
3  B' beats A
4  A evolves into A'
5  A' beats B'
6  B' evolves into B
7  B beats A'
8  A' evolves into A
9  A beats B
```

To prevent the emergence of such circular dominance relations made possible by intransitive relationship among the species one could evaluate each individual against all of its previous opponents. Evaluating individuals against previous opponents has the effect that the individuals not only must evolve new solutions. At the same time they must also retain their ability to beat old opponents. This prevents individuals from being overfitted to the current generation of opponents. In the above example this would mean that e.g. individual A′ created in line 4 would not seem as an improvement over A as they would both be evaluated against individual B and B′. This would result in equal fitness for A′ and A as they would both win one match and lose the other. An individual A″ would be recognized as superior to both A and A′ if it is able to beat both B and B′. Using the myopic approach of only evaluating individuals using their current opponents would make distinguishing the performance of A′ and A″ impossible. As described in section 1.8.2 concerning Shared Sampling the amount of evaluations necessary to evaluate each individual against all previous opponents may well be intractable except for very simple problems.

The concept of *Hall of Fame* (HoF) described in Rosin [47] is used to compensate for the necessity to save all individuals from previous generations for testing purposes so that progress is ensured and to prevent useful genetic material from getting lost. The HoF concept can be considered as an extended kind of elitism over time which saves the best genetic material during the entire evolutionary run. Instead of saving all individuals in each generation the best individual is added to the Hall of Fame as a future evaluation source. All students in a generation are then tested against a sample of the current teachers[4] and all or a sample of the teachers from the HoF.

In Rosin [47] it is investigated whether sampling randomly from the HoF or updating fitness for the individuals in the HoF and picking the most fit individuals is the most beneficial method for choosing teachers. The fitness of the HoF teacher individuals is updated by playing the individuals against the new generation of individuals from the same population as the HoF. This is possible in their experiment since both populations are of the same species. Rosin concludes that the computational effort required to maintain the fitness is much greater than the benefit obtained over the random sampling. In our opinion, updating the fitness of HoF members by pitting them against new individuals is a very bad idea since the purpose of the HoF is to retain genetic material which at a specific point in the evolution was successful. Even though old individuals are saved they will most likely receive

[4] E.g. the Teacher Set.

poor fitness against the new opponents compared to younger individuals and thus they are not likely to be picked when using the performance based approach.

1.8.4 Lack of Objective Fitness

A general problem presents itself when evolved individuals from a co-evolutionary run are to be analyzed. The individuals from one population **A** receive fitness relative to the individuals from the other population **B** against whom they are evaluated. For instance if an individual from population **A** beats many individuals from population **B** it receives high fitness and is likely to survive evolution and reproduce at a high rate. But the fitness obtained by this individual is directly linked to the performance of the individuals of the opposing population. If they are all of low performance a seemingly adept individual from the other population may in fact prove to be of low quality. Thus any numerical fitness awarded to an individual participating in co-evolution provides no objective information regarding the capabilities of the individual as its fitness is intertwined with the fitnesses of the individuals from the competing population. We call a such fitness *subjective*. This means that comparing the subjective fitnesses of two individuals of the same species from two different co-evolutionary runs yields no information about which individual is better. To compare them an external *objective* metric which they can both be compared against individually could be used, e.g. their performance against some optimal opponent. The individual which according to this external metric performs best can then be seen as the better of the two. Such an external objective measure is often not available. Due to this dilemma it can be very difficult to measure or analyze the output of a co-evolutionary run and reach a conclusion about its quality by inspecting the fitness levels of the evolved individuals.

1.8.5 Combining HoF and MOEA

When combining the concept of Hall of Fame (HoF) with Multi Objective Evolutionary Algorithms (MOEA) an issue arises concerning which individuals should be picked for the HoF in each generation. It is no longer possible to pick the individual with the highest fitness unambiguously since no such individual exists. Instead a number of Pareto optimal individuals are available. Very early in the evolutionary run the HoF would reach its size limit if all individuals from each generation's Pareto front were added to the HoF. If the Pareto front contains many individuals then the inclusion of the entire front is also likely to result in unacceptable computational overhead.

With our extended pursuit environment this problem is greatly simplified as one of our objectives - the performance objective - clearly outweighs the size objective. Thus we will simply select for the HoF the individual with best performance fitness.

Chapter 2

Previous Work

This section serves to briefly describe our previous work performed during the DAT5/F9S semester fall 2002 and detailed in Rydtoft and Rasmussen [43]. The work described in this chapter forms the foundation of the work detailed in the remainder of this report.

2.1 The Simple Pursuit Environment

The simple pursuit environment described in Haynes [20] was chosen for our initial experiments because we deemed it a simple and yet interesting environment and because previous work in the environment provided a source for comparison. The first tests performed with the environment showed that we were able to evolve strategies similar in performance to those evolved by Haynes using the same Function and Terminal sets. Our strategy and that of Haynes were both able to capture the randomly moving prey approximately 2% of all matches. Our strategy was generally better at getting the predators close to the prey but as Haynes' strategy it suffered from a phenomenon which caused the predators to block each other's paths often leading to deadlocks. We subsequently performed a test in which the predators were able to share positions within the grid world so they would not be able to block each other. The best strategy evolved using this setting fared even worse as the predators had no way of detecting whether they had surrounded the prey and merely moved as close as possible disregarding whether they were standing on the same square as another predator. This caused them to end up in positions where all predators were next to the prey but not all directions were covered. In our last test we included a very simple form of communication allowing the predators to see each other if they were standing at the same square. The best strategy evolved using this setting had an impressive capture rate of 99%. Annoyingly the tree containing it consisted of 452 nodes making it utterly incomprehensible for us.

2.2 Reducing Tree Sizes

A general problem when evolving strategies in the simple pursuit environment was that the trees representing the strategies were unnecessarily large making it difficult to understand the strategies by examining the trees. This phenomenon - referred to as bloat - is very common and several ways of reducing it are documented in the literature (see for instance Koza [31], Blickle [3], Zhang [58] and Zitzler [61]). After examining various approaches we decided to use a multi-objective evolutionary algorithm known as SPEA2. We used our original performance related objective as well as an objective promoting small trees. Using the multi-objective approach we were - after reducing elitism by lowering the tournament size used by tournament selection - able to evolve a strategy with a capture rate of 100% represented by a tree consisting of merely 75 nodes. The strategy allowed the predators to capture the prey well within the 100 turn limit. A histogram showing the turns on which captures were achieved for 1000 test matches can be seen in figure 2.1.

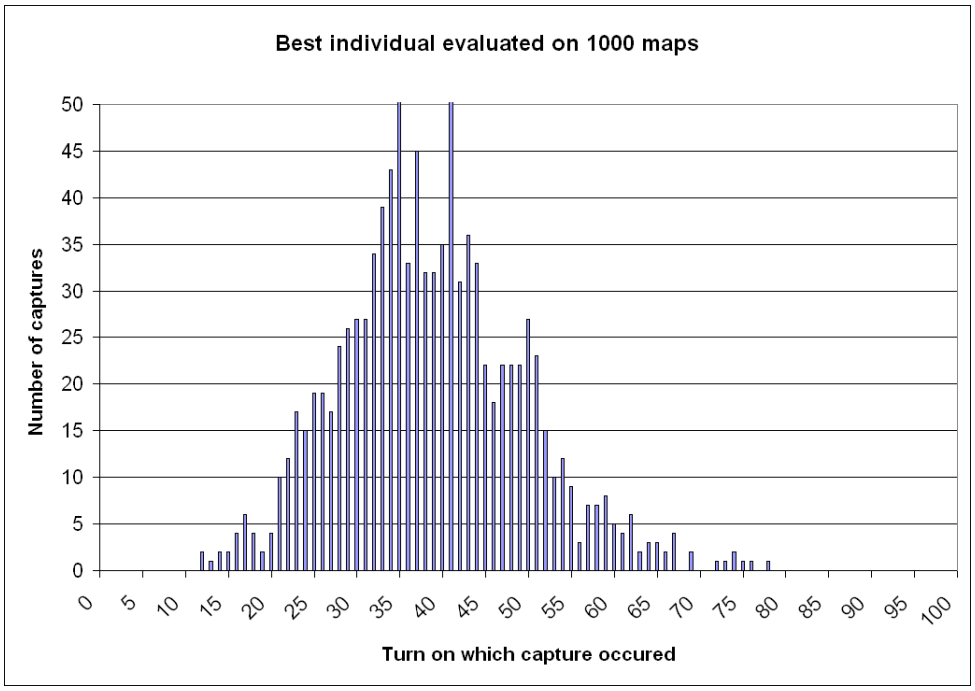


Figure 2.1: Histogram showing the turns on which captures were achieved for 1000 test matches.

2.3 The Extended Pursuit Environment

Because we were able to evolve a flawless strategy for the predators in the simple environment using very simple communication the necessity to extend the environment became apparent. Specifically we wanted an environment in which the

prey could be co-evolved along with the predators to produce a greater challenge. Haynes had already performed experiments with co-evolving prey agents in the simple environment with disappointing results (see Haynes [16]) so we decided to design a new extended environment with a larger grid world, different movement and maneuverability rules, new capture definition and new functions and terminals for both predators and prey. In addition numerous features were implemented to support the often problematic competitive co-evolution of two species. These features included Competitive Shared Fitness, Hall of Fame and Shared Sampling. The very first tests performed within the extended environment made apparent the existence of a disengagement problem which occurred when one of the two species became too dominant. The disengagement problem caused the evolution of the dominated species to halt as it was impossible to detect which members of the dominated species were most fit for survival as all members scored very low fitness against the superior opponent species. Effects of the problem can be seen in figure 2.2 in which the prey never counter advancements achieved in the predator population in generation 36. To avoid the problem of disengagement a novel approach of interleaved competitive co-evolution was devised in which the evolution of a dominating species would be halted until it lost its dominance. Further preliminary tests within the extended environment uncovered interesting strategies as well as a problem of lack of diversity. The solution to the diversity problem was left for the spring semester 2003 and is detailed in section 3.4.

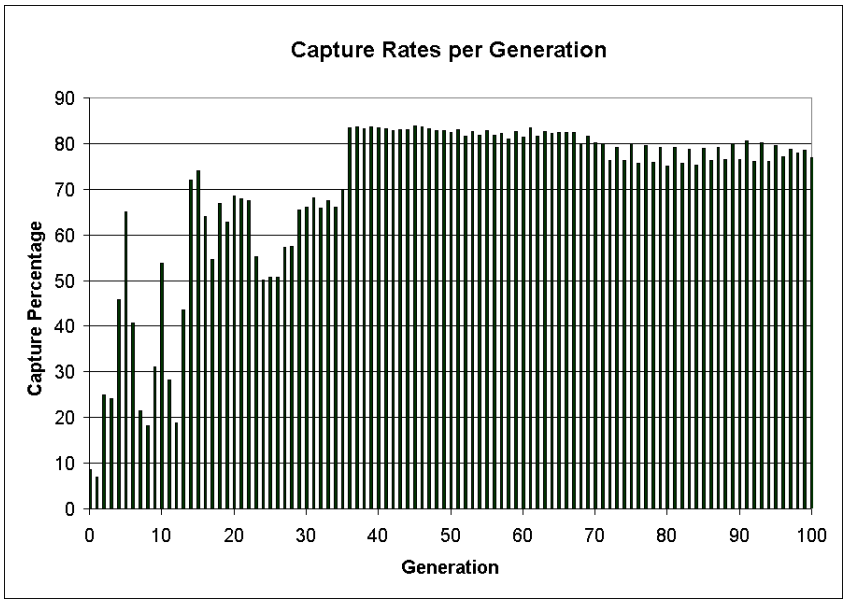


Figure 2.2: Predator dominance occurring at generation 36.

Chapter 3

Optimizing Evolution Flow

In this chapter we document the continued work concerned with adjusting and tuning our co-evolution to achieve satisfactory results. Additionally we describe some of the different statistical data we gather during an evolutionary run and how it is analyzed. This description is provided in section 3.1.

The primary problem identified in the previous semester is the lack of diversity which will be solved. The selection pressure is at first thought to be a major contributor to the lack of diversity. Test results support this suspicion and indicate that the selection pressure is too high causing various unwanted phenomena. A different selection strategy is proposed and tested in section 3.2. This selection strategy yields better results although not alleviating the problem with loss of diversity. However, the strategy allows for a much needed low selection pressure which is lower than previously attainable. In section 3.3 various techniques for maintaining diversity are presented and discussed with the intentions of applying one of these. The FOCUS method (De Jong [8]) is implemented and tested in section 3.4. Results using a modified version of the FOCUS approach are very satisfactory and the objective to maintain diversity is achieved. During our work we discover an additional problem related to noise in the extended pursuit environment. This problem and our solution is described in section 3.5.

3.1 Analyzing Co-evolutionary Data

The task of analyzing data from a co-evolutionary run is recognized (Cartlidge [5], Ficici [12]) as being complicated for several reasons. When analyzing a competitive co-evolutionary run it is difficult to tell whether the run is leading to increased learning, noisy oscillations or endless cycling through mediocre strategies - just to mention a few examples. In Ficici [12] and Cliff [6] the problem of analyzing and measuring the quality of an evolved agent behavior is also pointed out. The agent strategy can only be described in subjective and qualitative terms and no objective measures exist which can be used to characterize or judge the agent. This is also the case for the agents in our extended pursuit environment. However, several relevant properties can be deduced from statistical data generated from the run and

used e.g. as progress indicators. This section describes the various data generated from co-evolutionary runs in the extended pursuit environment along with how it is processed, evaluated and analyzed. The section lists the most important and useful generated data and gives examples of how particular types of data have been used to solve problems in the environment and in the evolution process. The interpretation of data and which information it provides is also explained. In subsection 3.1.2 an objective measure of progress in the co-evolution is introduced.

3.1.1 Statistical Data and Graphs

After each generation in a co-evolutionary run various statistical data is stored for analysis purposes. The data listed below is the most important data and its meaning and use will be explained in this section.

- Average tree size
- Node statistics
- HoF individuals
- Individuals in the Pareto Front
- Distribution of individuals with regards to performance and size
- Capture locations, capture turn and number of captures

Average Tree size

After each generation the average tree size for each population is stored and used to generate graphs of the type shown in figure 3.1. The tree sizes are not directly correlated to the complexity of the strategies of the individuals. Especially not if no measures are taken towards avoiding bloat. Since a size objective is also sought optimized in our co-evolution we can, however, deduce some information from the average tree sizes. Often the average tree sizes in conjunction with other statistics can give indications about the co-evolutionary run or the average tree sizes can be used to support or confirm deductions or conclusions. Although the average tree size alone will often not indicate evolutionary progress it is valuable. If for instance the average tree sizes decrease to very low values it is most likely that the co-evolution will not produce satisfactory results and that the selection process should be examined. In another scenario in which the average tree size for the predators has been stable for some time but then increases, the increase can indicate that the predators are exploring new search areas and that the strategies of the prey are forcing them to evolve more advanced solutions. However, the average tree size should always be used in context with other statistical data.

The graphs in 3.1 illustrate the development of the average tree sizes for a given run where the prey population is paused most of the time. The average tree size increases for both population during the beginning of the run. After generation 100 the prey population remains paused most of the time and thus does not grow or shrink very much in average tree size. In general the average tree size for the predator population fluctuates quite a lot. The relatively steady increase in size

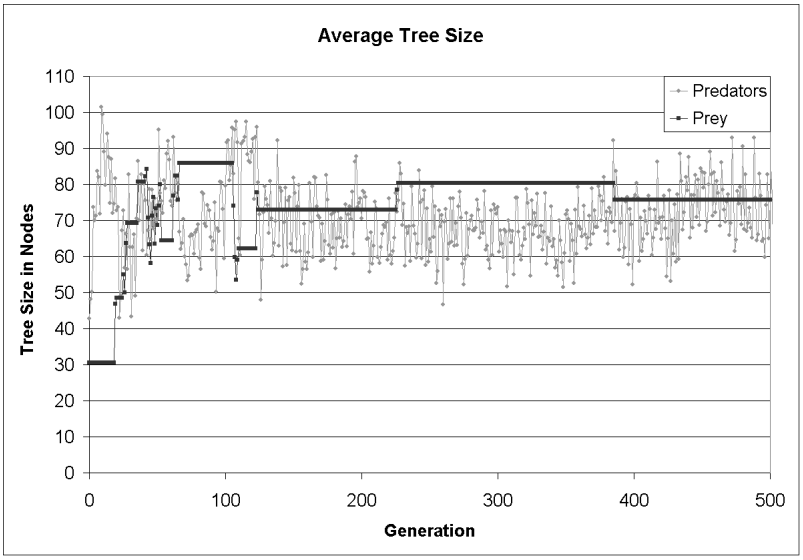


Figure 3.1: Graph illustrating the development of the average tree size for both the predator and prey population.

of the predators starting around 70 can be the cause of the prey being unpaused around generation 105. This can be an indication that the predator population has evolved strategies which the prey population now needs to evolve counter strategies against. The prey population seems to adjust fast to this change and is paused again shortly after. These speculations only serve to demonstrate how the graph can be used for analysis.

Node Statistics

For each generation and each population a histogram of node usage is stored. This data is used to produce graphs similar to the one shown in figure 3.2.

These graphs are useful when e.g. new communication functions are introduced and it is important to see the development of their use. The graphs also make it evident to us that certain functions which are practically bred out are later reintroduced during co-evolutionary runs. This property was assumed to be present in our co-evolution but these statistics confirm it. If a sudden increase occurs in the usage of a particular function, which has so far seldomly been used, it is a strong indication that the population has evolved strategies which take advantage of this function. The increase can therefore be related to co-evolutionary progress if at the same time that particular population is more successful. When examining the graphs in figure 3.2 it is evident that the two depicted functions are initially inserted equally many times when the solutions are randomly generated. The function incorporating the shout functionality is then for some reason less attractive to use and its frequency drops. Often when one of the depicted functions drops in usage the usage of the other increases. For these two functions this is not surprising as the functionality of the IfThenElse actually encompasses a subset of the functionality of the IfThenElseShout. Combined with other statistics the node statistics often

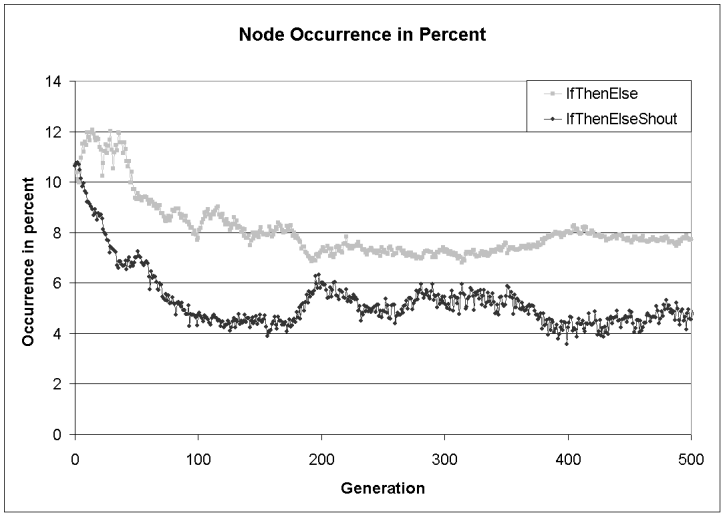


Figure 3.2: Graph depicting the occurrence in percent of the two nodes IfThenElse and IfThenElseShout (introduced in section 4.4) during 500 generations. The occurrence percentage for each generation is relative to the total amount of nodes used in the predators in that particular generation.

proves useful.

HoF individuals

When an individual is inserted into the Hall of Fame its entire solution tree along with characteristics such as tree size, number of successful matches and all of its fitness values are stored. The actual solution trees can for instance be used to test individuals from later generations against individuals from other co-evolutionary runs or to verify co-evolutionary progress. Often the HoF individuals serve as good solution candidates and can be considered part of the final results from the co-evolution. When examining a co-evolutionary run the number of successes for the individuals inserted into HoF in the latest generations gives an indication of how well the population is doing. If the HoF predators have a high number of captures it means that the population has strategies capable of beating a fairly high percentage of the prey strategies. However, a high number of captures is not always a correct indication of the quality of the predators. It also depends on the quality of the opposing prey strategies. If these are very poor a high number of captures is not an immediate indication of good predator strategies.

Individuals in the Pareto Front

In each generation information about the individuals in both Pareto fronts are stored along with the sizes of the Pareto fronts. Since we will be optimizing as much as three objectives[1] several individuals will be present in the Pareto front but not all of these are of interest with respect to the final solutions. For instance, individuals which are optimal with regards to size, i.e. trees consisting of only one node, are not of interest. Examining the individuals in the Pareto fronts often give an overall

[1]The third is introduced in section 3.4.

indication of the entire population since these individuals represent the most optimal solutions for each objective and also influence the breeding process and thus the further evolution. The development in the Pareto front can also be informative. If the size and make-up of the Pareto front stagnates/stabilizes over a period so has the evolution. The movement of the Pareto front also shows in which direction the evolution is proceeding, e.g. if the evolution is searching larger solutions.

Distribution of individuals with regards to performance and size

Since three objectives will be sought optimized the Pareto front will form a 3-dimensional surface in a coordinate space with the three objectives as axes. It is therefore more complicated to observe the movement of the Pareto front than when only two objectives were used. The third objective - which will be a measure of the diversity of a solution - is not in any way related to how interesting and advanced the actual solutions are. It serves to guarantee a satisfactory evolutionary flow and is therefore often excluded when depicting the Pareto front. This reduces the Pareto front to a 2-dimensional graph which is more easily perceived.

The same kind of graph can be made for the entire population. This produces an interesting distribution of the individuals with regards to performance fitness and size which can be used to examine the evolution. For instance it can be used to see if the population is diverse or in general contains too many small solutions. If additional information about which individuals are selected for breeding is embedded into the graph it can be used to verify that the selection pressure is adequate and verify whether desired individuals are selected. An example of a such graph can be seen in figure 3.3. Notice that individuals which may not seem to be in the Pareto front in this particular graph may be selected because they are optimized with respect to the diversity objective.

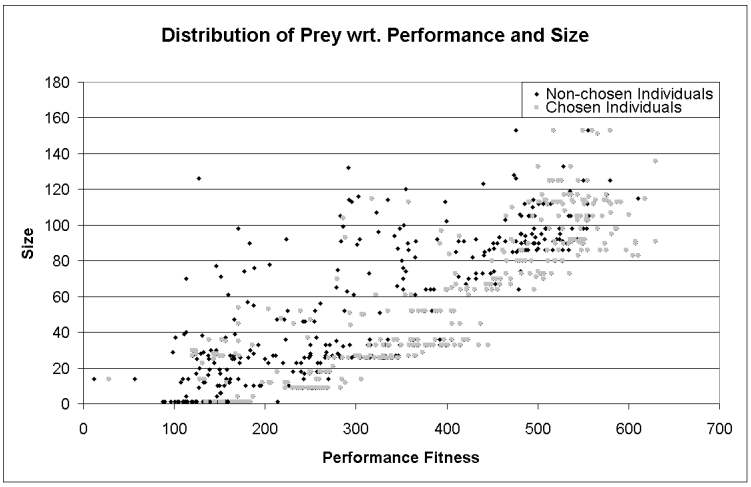


Figure 3.3: Graph depicting the distribution of prey individuals with regards to performance fitness and size from a given generation. The grey dots represent individuals which are selected for breeding.

Capture locations, capture turn and number of captures

For each generation the locations in the grid world where prey are caught are stored in a map. By examining this map it becomes obvious where captures occur and also if and how the strategies of both predators and prey change during an evolutionary run. The capture locations are also useful to detect potential exploitations of the design of the environment. During several tests the predators learn to drive the prey towards the walls in the grid world and thus trap and capture it. This became evident after studying the capture locations and effectively caused the introduction of functions for the prey to detect and avoid walls.

The turns in which prey are caught is also stored and used to produce the average, minimum and maximum value of this number for one generation. The total number of captures and evasions for each generation is also stored. If for instance the average turn in which capture occur is low and the number of captures is high this can indicate that the prey are too easy to catch and that the parameters for the evolution may need to be tuned.

An example capture location map can be seen in figure 3.4. The range of captures is mapped into a 256 color greyscale with highest number of captures represented by the color black and no captures by white. The shown map is from an early generation in which the prey have only been unpaused for a few generations. The cross-hair like appearance is due to a high number of simple prey strategies which cause the prey to simply move in a fixed direction. Predators have evolved to line up with the prey either vertically or horizontally and approach it once lined up correctly. Prey with simple evasive abilities are almost exclusively caught when close to walls. Especially the corners seem hazardous at this point.

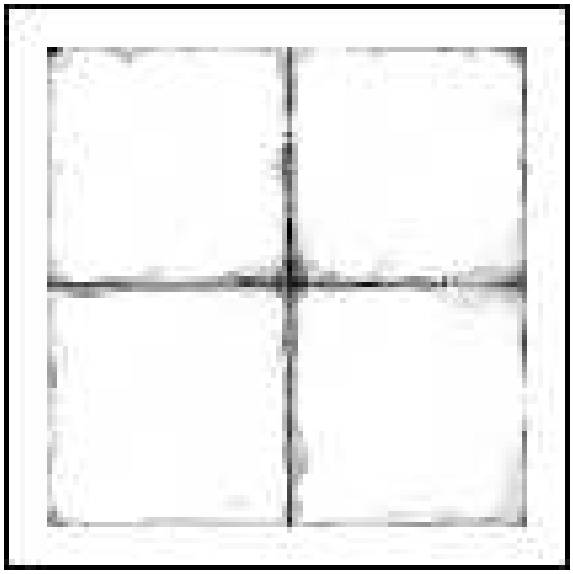


Figure 3.4: Map showing capture locations for an early generation. The higher number of captures at a grid location the darker its color.

3.1.2 An Objective Measure of Progress

One of our high priority tasks listed in the "Future Works" section of Rydtoft and Rasmussen [43] is the development of a general method for analyzing progress and results in competitive co-evolutionary runs.

We have devised an objective measure allowing the progress of one population during a competitive co-evolutionary run to be monitored by comparing it with the opposing population. Specifically, for each population member **A** being inserted into the HoF we pit this member against all members of the opposing HoF and the number of successes for **A** is stored. As new members are inserted into the opponent HoF they are also pitted against **A** and its number of successes is updated. If new HoF members beat more members from the opposing HoF than old HoF members our population is improving. Depicting the information in a graph with HoF individuals along the x-axis and their respective successes along the y-axis it is easy to see whether or not the evolution is progressing satisfactorily (i.e. if improvements are occurring). The slope of any part of the resulting graph can be seen as a rough indication of the speed with which improvements or possibly degrading occurs at that particular point in the evolution. Note that this method can only be used to monitor the progress within a single co-evolutionary run. Comparisons between two runs cannot be performed using this method alone. A graph depicting this information can be seen in e.g. figure 4.4 on page 57. Such graphs will be referred to as "HoF improvement graphs".

3.2 Adjusting Selection Pressure

In this section we investigate if the lack of diversity we are experiencing is caused or amplified by too high selection pressure. By examining test results it becomes evident that the selection pressure is too high which results in an unsatisfactory selection of individuals and in fluctuations in the capture rates of the predators. To alleviate this problem a different selection strategy is proposed and tested which produces a lower selection pressure resulting in a more satisfying co-evolutionary flow. The proposed approach is a modification of the original SPEA2 algorithm. The modification selects individuals from the entire population for breeding as opposed to the original approach which only selects individuals from the SPEA2 archive.

3.2.1 Original SPEA2 Approach

In figure 3.5 the capture rate for predators in a typical run is depicted. In this evolutionary run selection is performed only from the SPEA2 archive and the Tournament Size is 2. Populations are of size 700 and each SPEA2 archive is of size 70.

Examining the graph in figure 3.5 is can be seen during the early generations 10 to 50 that the capture rates do not increase at a steady pace as expected but fluctuate instead. This fluctuation is even more evident in the later generations 140 to 200 where the capture rates sometimes change almost 30% from one generation to the

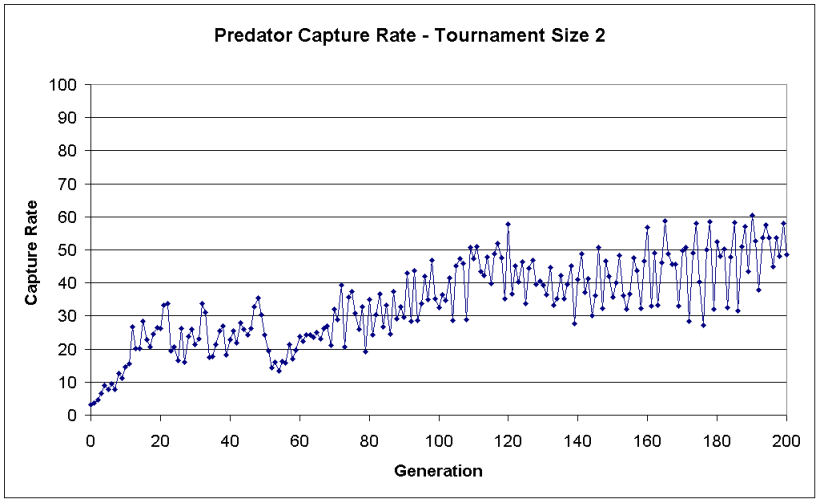


Figure 3.5: Capture rate for predators. Selection is only performed from the SPEA2 archive and the Tournament Size is 2.

next. Although small fluctuations can be unavoidable they are - to the extent seen in this run - unwanted since they may prevent the evolution from investigating potential solution areas. We believe that one reason for the fluctuations is a too high selection pressure.

This belief is strengthened when examining how individuals are selected for breeding and how this affects the population and thus the search as a whole. In generation 36, in the evolutionary run described above, 340 out of 700 predator individuals are below 20 nodes in size. The other part of the population has slightly better performance fitness and is larger in size. In the following generation the number of individuals which are below 20 nodes in size has increased with 48% from 340 to 505 individuals. Even though the predators far from dominate the prey their current solutions are being minimized with respect to size and fewer larger solutions are being explored. In figure 3.6 the average tree size is shown for the predator individuals. It is evident that the selection pressure forces down this average to a very low level which the evolution does not manage to leave within the 200 generations. Furthermore, these constantly small tree sizes are indications that the before mentioned selection towards small trees is not an isolated event but one which occurs repeatedly. This unfortunate selection happens because the small individuals are much better in the size objective than the larger individuals and the difference in performance fitness is not sufficiently large to get the larger individuals selected. Thus, due to the high selection pressure and size objective we select the smaller individuals too aggressively. This effectively slows down and disturbs the evolution.

3.2.2 Modified SPEA2 Approach

In order to lower the selection pressure in an attempt to avoid the fluctuations and to escape the small tree sizes, we propose selecting from the entire population in-

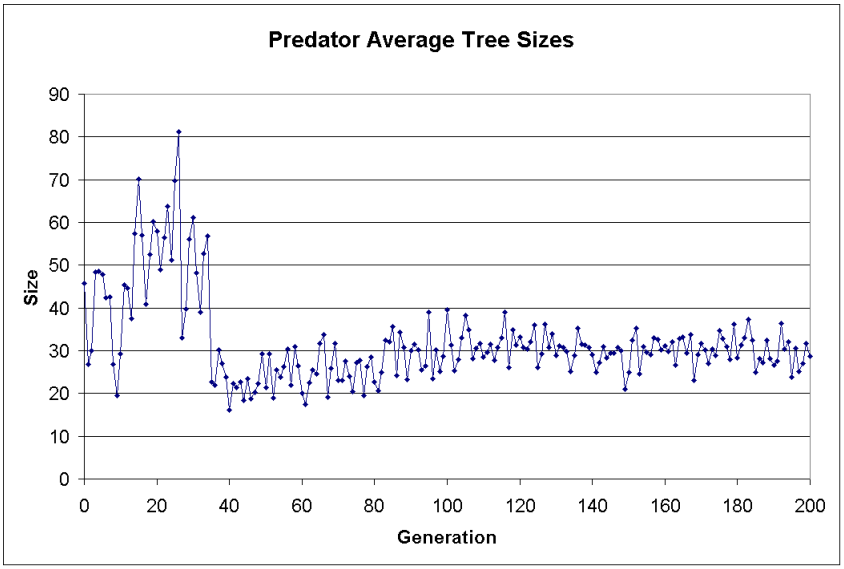


Figure 3.6: Average tree sizes for predators from the test performed with selection from the SPEA2 archive and Tournament Size 2.

stead of only from the SPEA2 archive. Currently we are using a Tournament Size of 2 which is the lowest possible setting for this parameter. To further lower the selection pressure we must select from a more varied pool of individuals. This should enable us to reach a lower selection pressure which can be further adjusted by altering the Tournament Size.

Identical tests are performed with various Tournament Sizes to examine the effect of selecting from the entire population and to identify an appropriate Tournament Size.

In figure 3.7 the capture rates for predators are shown from a performed test where the Tournament Size is 4 and selection is performed from the entire population. When compared to the corresponding graph in figure 3.5, the capture rates in figure 3.7 do not fluctuate to the same extent. The capture rates increase at a slower but more steady and constant pace as opposed to the captures rates in the test in which selection is only performed from the SPEA2 archive.

In figure 3.8 the distribution of predators with regards to performance fitness and size shows that the selected predators are now spread out over a much larger part of the Pareto front compared to the test in section 3.2.1. Furthermore, it is worth noticing that the majority of the population is no longer the smallest predators and that these no longer obtain nearly as good performance fitness scores as the larger predators. This also effectively increases the average tree sizes enabling the search to explore larger solutions when necessary. The average tree sizes for the predator population is shown in figure 3.9.

Based on test results it is found that a more appropriate selection pressure is applied in our environment when selecting individuals from the entire population for breeding. This more appropriate selection pressure does not cause major fluctu-

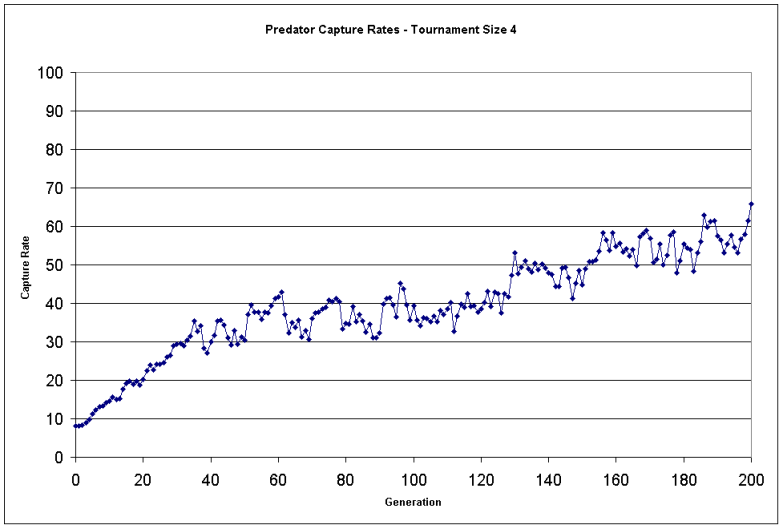


Figure 3.7: Capture rate for predators. Selection is performed from the entire population and the Tournament Size is 4.

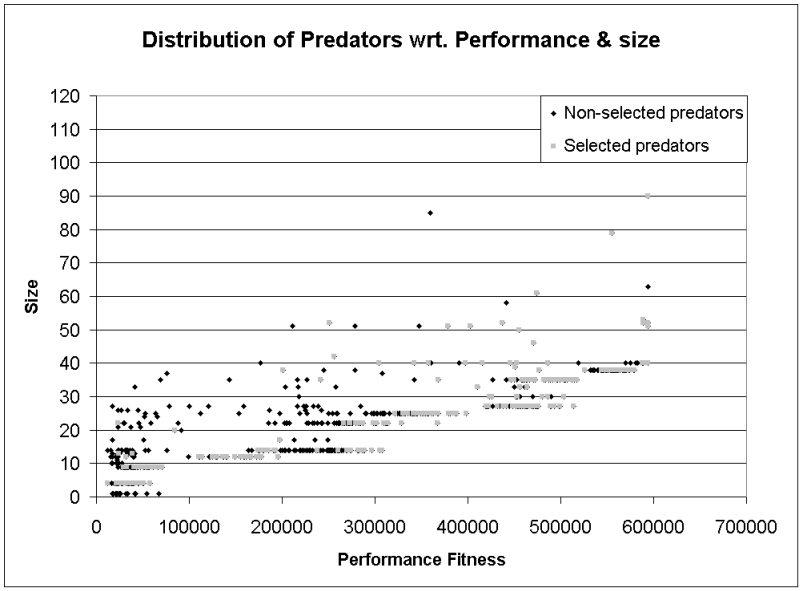


Figure 3.8: Distribution of non-selected and selected predators with regards to performance fitness and size. The predators are from generation 150.

ations in the capture rates and it does not lead to selecting small solutions to the same extent as with the original SPEA2 selection strategy. We will therefore continue to use this selection strategy. However, the results are not nearly satisfactory with respect to the diversity of the individuals. This is even more evident for the prey population. The following section will describe this problem in greater detail

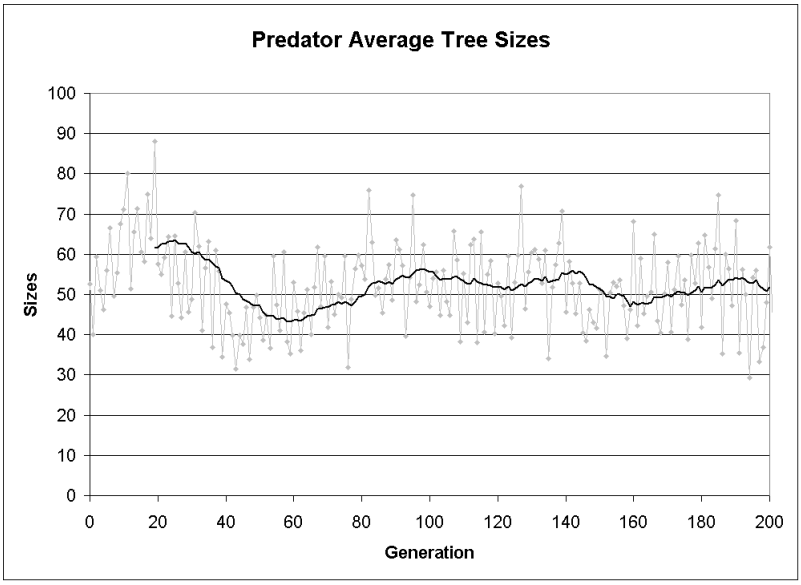


Figure 3.9: Average tree sizes for predators. Selection is performed from the entire population and the Tournament Size is 4.

and solutions are proposed.

3.3 Techniques For Maintaining Diversity

In Rydtoft and Rasmussen [43] we show that lack of diversity is present in our performed tests and we believe this problem to be a major reason that the learning cycles stop taking place in our co-evolutionary runs. In this chapter we document our effort for maintaining diversity in order to ensure a continuation in the learning cycles and hence to evolve more interesting strategies. In section 3.3 three different approaches for maintaining diversity are presented and discussed. The FOCUS approach presented in section 3.3.3 is chosen over the two other approaches – Fitness Uniform Selection Strategy and The Patchwork Model presented in 3.3.1 and 3.3.2 respectively. In section 3.4 tests are performed with an original and a modified version of the FOCUS approach and results document that the objective with regards to maintaining diversity has been obtained satisfactorily. Better strategies than previously witnessed are evolved.

3.3.1 Fitness Uniform Selection Strategy

Fitness Uniform Selection Strategy (FUSS) is proposed in Hutter [25]. FUSS generates selection pressure towards sparsely populated fitness regions but not necessarily towards higher fitness regions as opposed to other more standard selection schemes. In this section FUSS and its properties are presented and discussed with

the intent of using FUSS to solve the problem of maintaining diversity in the extended pursuit environment.

FUSS is based on the insight that a single sufficiently fit individual is usually appropriate as solution to a problem rather than having an entire population converged to some solution. This property is sought achieved by automatically creating a selection pressure towards sparsely populated fitness regions, which results in uniformly distributed fitness values. Furthermore, according to Hutter [25], FUSS has the effect of preserving the genetic diversity better than standard selection schemes such as Fitness Proportionate, Ranking and Tournament Selection which all favor individuals of better fitness. This is because the standard selection schemes use a distribution over individuals whereas FUSS uses a distribution over fitness values as bias when selecting.

FUSS is defined as follows: A fitness value f is randomly selected in the interval $[f_{min}, f_{max}]$, where f_{min}, f_{max} is the lowest and highest fitness values in the current population. Then the individual i in the population with a fitness value nearest to f is selected and a copy of this individual is added to the population, possibly after recombination and/or mutation. Population size is adjusted by deleting individuals from regions of high density. As such the FUSS algorithm is steady state rather than generational.

Since FUSS does not converge towards some fitness value the problem of premature convergence is avoided. However FUSS does apply selection pressure towards the higher fitness values. When the number of fit individuals is low the selection pressure towards higher fitness regions is increased and as the number of fit individuals grows the selection pressure is reduced accordingly. Since FUSS also favors low fitness individuals from sparsely populated areas, which as such are of no interest for the search, time may be wasted searching the wrong end of the fitness scale. This seemingly apparent slowdown is according to Hutter not necessarily a negative factor since it may later help in obtaining high fitness levels which are difficult to reach.

The problem of takeover of the highest fitness level is known from other selection schemes. In these the concept of "takeover time" specifies how long a discovered locally or globally optimal solution needs to spread out and dominate the population making it almost impossible to continue the search in other areas. This problem will not occur with FUSS since the fraction of very fit individuals in the population is always small and thus the average fitness is always lower than the best fitness.

3.3.2 The Patchwork Model

This section presents the Patchwork model which was first proposed in Krink [32]. According to Krink the Patchwork model allows better control of population diversity and of the selection pressure applied to the evolution. A major difference between the Patchwork model and more traditional EAs is the representation of individuals which will be described below. Furthermore, the structure used to model the population is different from previously encountered approaches.

The model used to contain the population is a combination of the island and the diffusion model with some slight modifications. As with the diffusion model a grid

interconnection topology is used where each grid cell holds one individual. This allows for parallel processing as each individual or grid cell can be assigned its own processor. However, in the Patchwork model each grid cell can contain several individuals as opposed to the diffusion model. Each grid cell can then be seen as a subpopulation and the Patchwork model can then be considered as a special case of the island model. Individuals can interact with each other within grid cells, e.g. mate depending on their desire. Like in the island model, individuals are able to migrate from one grid cell to another according to certain rules. This migration can result in grid cells without individuals - i.e. empty subpopulations - which is not allowed in the island model. Another difference between the Patchwork model and the two other approaches is the population size which may vary in the Patchwork model but is constant in the two other approaches.

In the Patchwork model individuals are modeled using a technique very similar to that of multi-agent systems, where a system is represented by autonomously interacting mobile agents. The individuals in the Patchwork model can be described as autonomous mobile agents which live in a virtual ecological niche where they interact with their environment through sensors and motors, making decisions based on collected local information. The idea to model the evolution of autonomous agents in an ecological niche has been introduced in ALife and ecology where one general approach is called SWARM (see Minar [39]). Each individual in the Patchwork approach consists of two parts, namely its *genome* and its *motivation network*. The genome is a set of three *chromosomes*, namely a solution chromosome, a chromosome of standard deviation and a chromosome of parameters. The solution chromosome represents the solution of the individual and is used to calculate its fitness. The second chromosome is related to the mutation of the individual and the third chromosome plays and important role in the motivation network of the individual. Based on local information gathered from sensors the motivation network of an individual decides which action to take, e.g. migrate, mate etc. The individuals with the highest fitness are preferred over those with lower fitness if conflicts associated with actions occur. We will not go into details with the functionality of the motivation network but merely state that applying the Patchwork model would require a substantial amount of implementation work.

3.3.3 FOCUS

The FOCUS algorithm described in De Jong [8] seeks to promote diversity using a multi-objective algorithm by introducing an objective which actively promotes diversity. Each individual is thus awarded fitness in this objective based on the degree to which it differs from other individuals in its population. This degree of difference is measured by - for each individual i in the population - measuring the structural difference between this individual and any other individual from the same generation. All of these pairwise comparisons are summed to produce one overall measure of diversity for the individual. The structural difference between two individuals i and j is measured by comparing their trees on a node-to-node basis. For each node in j not present at the same location in i the diversity measure is increased by one. After the comparison the measure is normalized through division with the node size of the smaller of the two trees. Thus for this objective a value of zero is the worst possible and infinity better than the possible best.

The FOCUS algorithm has a very strict selection criterion. Basically the mating pool is filled only with non-dominated individuals and in cases where multiple copies of the same non-dominated individual exist only one instance is allowed into the mating pool. This uniqueness criterion is according to De Jong introduced because the standard Pareto dominance criterion in this context may cause a proliferation of individuals occupying the same point of the trade-off surface. As a result of the strict selection the mating pool is typically very small and the search very greedy. The diversity objective, however, manages to ensure sufficient diversity in all test runs in De Jong's article. A flow chart for the FOCUS algorithm can be seen in figure 3.10.

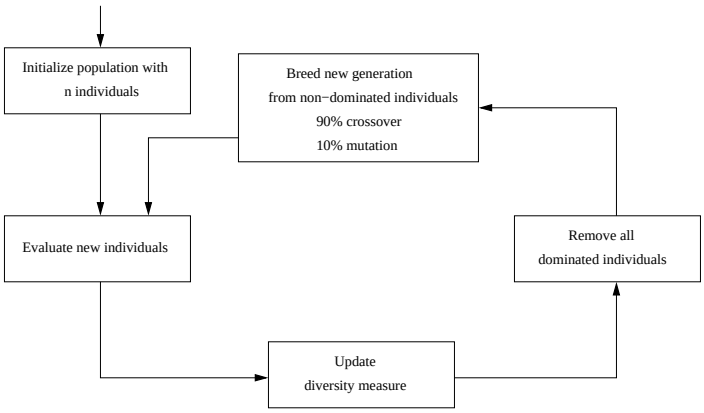


Figure 3.10: Flow of the FOCUS algorithm.

3.3.4 Discussion of Techniques

FUSS is described as a steady state algorithm in which no individual is ever deleted and pressure is generated towards sparsely populated regions of the search space. The FUSS scheme in its current form seems ill suited for a generational STGP implementation relying on SPEA2 multi-objective optimization. To use it we would have to rethink our evolution approach from the beginning and possibly have to solve the bloat problem again. An alternative approach would be to construct a FUSS variant suitable for our type of evolution. We prefer an approach more immediately applicable and therefore eliminate FUSS from consideration as a means to counter loss of diversity in our co-evolutionary model.

The remaining two approaches discussed in the preceding sections both seem attractive to us. Having consulted literature in which they are presented (in De Jong [8] and Krink [32]) we can find no obvious problems indicating that they should not work in our case. Implementing the Patchwork model will take some effort. As the ECJ system which we use (see Appendix B) does not support the approach numerous core classes will have to be modified or extended. For instance the breeding pipeline architecture of ECJ will need much attention as it not designed with local selection in mind. Implementing FOCUS will, however, be very easy. An additional feature of FOCUS which captivates us is that - unlike the Patchwork model - it is

designed explicitly to actively maintain diversity. The Patchwork model can help to increase diversity but it does not do so in a direct principled way (see Wiegand [57]). In our opinion the diversity maintaining properties of the Patchwork model are closer to being a positive side-effect. Thus a diversity loss can still take place in this model. The chance of this occurring is much smaller due to the fact that genetic material very slowly diffuses around within the population. However, there is nothing in the Patchwork which prevents a single best individual or the genetic material of that individual to slowly spread out. These arguments serve as reason for us to use an implementation of the FOCUS algorithm to promote and maintain diversity.

3.4 FOCUS Tests

As concluded in section 3.3.4 we will attempt to use the FOCUS algorithm as means of maintaining diversity. This section details two tests performed using FOCUS. Briefly the tests consist of the following:

- **Test Using FOCUS**
  Test using an exact implementation of De Jongs FOCUS algorithm.
- **Test Using Modified FOCUS**
  Test using a modification of FOCUS with lower selection pressure and alternate diversity measure.

3.4.1 Test Using FOCUS

In this test we employ the FOCUS method as it is described in De Jong [8]. That is, we initialize each population with 500 randomly generated individuals. These individuals are evaluated against a sample from the opposing population. In addition to being assigned performance and size fitness the individuals are assigned diversity fitness using the structural diversity measure described in the article. We also employ their version of the Pareto dominance relation and thus remove all individuals which are dominated as well as copies of non-dominated individuals. We use the lowest possible tournament size for Tournament Selection as the selection will be from a small mating pool consisting of only unique non-dominated individuals. The test is run for 100 generations. The function and terminal sets used for this and the following test are those shown in appendix D.

The results obtained are less than satisfactory. Capture and evasion rates for each generation can be seen in figure 3.11. In generation 15 the predators achieve a massive improvement with a capture rate of approximately 42.3% unpausing the prey population. The improvement is quickly lost and from this point nothing significant is evolved.

Rapid fluctuations can be observed in the average tree size of the predator population (see figure 3.12). In many cases the average tree size is below 2 signifying a population primarily consisting of individuals with a single terminal as their tree. After generation 1 the predator mating pool of unique non-dominated individuals

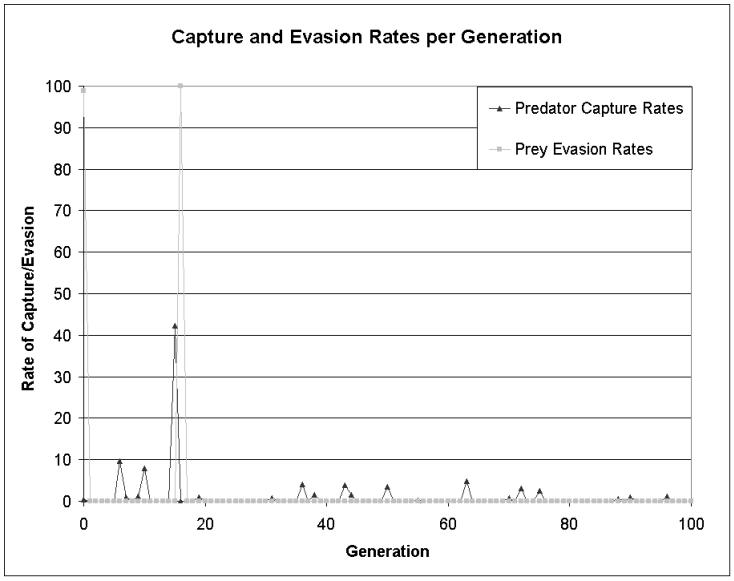


Figure 3.11: Capture rates for predator population and evasion rates for prey population.

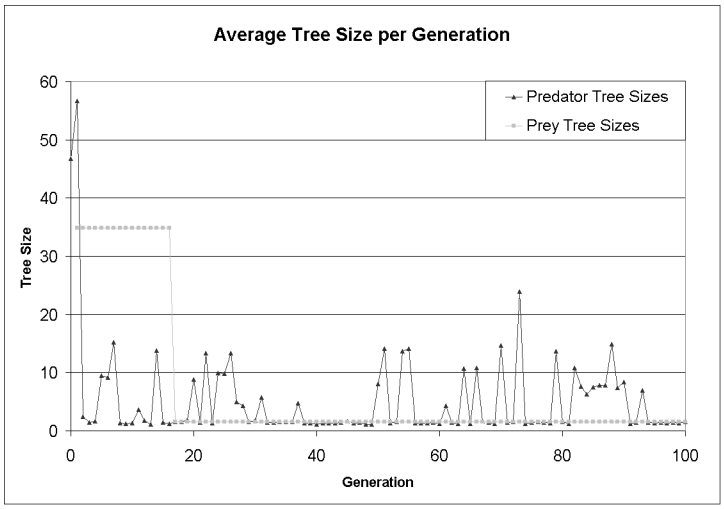


Figure 3.12: Average tree sizes for predator and prey population.

consists of three specimens. Their respective sizes are 1, 4 and 36 nodes. Selecting from this set, crossover will often be performed using at least one very small tree. This drastically reduces the average tree size. In the following generation only trees of sizes less than 10 nodes are present in the mating pool. Breeding from a such set produces a generation of very small and similar individuals all performing poorly. The later increases in average tree size occur when a large individual finds its way into the mating pool. A general problem throughout the evolution is that the mat-

ing pool always consists of at most 5 individuals of which a couple always have tree sizes of at most four nodes. Performing crossover with these small individuals splits the trees of the larger individuals and produce two offspring of reduced size and functionality. The evolution thus gets stuck in a part of the search space consisting of small individuals.

We had expected the diversity measure would prevent the evolution from being stuck as described above as individuals with larger trees should survive merely as a result of the diversity objective. Unfortunately small individuals usually achieve relatively nice diversity scores in a population consisting of individuals of mixed sizes. This is related to the fact that when diversity is calculated the diversity measure is divided with the size of the smaller of the two trees being compared. When one of the two trees is e.g. an individual with a tree consisting of a single terminal node, the diversity measure will always be divided by one. Only in generations in which a large part of the population consists of very small individuals the larger individuals will beat them with regards to diversity. This results in a subsequent generation with larger average tree sizes in which the small individuals will again obtain good diversity scores.

In conclusion, not only has the FOCUS approach failed to maintain diversity in our populations. It has additionally prevented the evolution of even the simplest strategy in either population. The best predator and prey individuals resulting from this run are clearly inferior to anything evolved earlier.

3.4.2 Test Using Modified FOCUS

It is interesting that the FOCUS approach performs so well on the Even 3-5 Parity problems it is applied to in De Jong [8] and yet shows none of the alleged properties[2] when used in our environment. One major difference between the two problem domains is that we use competitive co-evolution and thus have a relative measure of performance fitness while they use a basic evolutionary approach with an absolute fitness function. Furthermore our fitness function can be affected by noise as a predator or prey individual may beat its opponent as a result of a favorable starting position rather than a superior strategy. The fitness function for the simple Even Parity problems contain no such noise. We believe the complexity of our performance fitness both with regards to its relative nature and noise makes it necessary to reduce the degree of elitism. Another potential problem is that with the current measure of diversity individuals with smaller trees obtain good scores in two of the three objectives in generations with individuals of varied sizes.

To remedy the above-mentioned problems we suggest a variation of our combined SPEA2-FOCUS approach which allows for a lower degree of elitism to accommodate noisy environments while maintaining diversity and guiding the search sufficiently.

Instead of constructing our mating pool from a set of unique non-dominated individuals or even the SPEA2 archive we will continue to breed from the entire population as described in section 3.2. Raising the tournament size will make it possible to increase selection pressure while keeping it low will allow for a slower less greedy

[2]Maintaining diversity and guiding the search sufficiently in spite of an enormous degree of elitism.

search. We alter the diversity measure from the structural approach to an objective-based approach in which individuals are punished for having "common" size and performance fitness. Specifically we will define a bounding box around each individual and take the number of individuals within this bounding box as a measure of its lack of diversity. The *Bounding Box Diversity* which is to be maximized will for individual i thus be defined as the population size subtracted by the number of individuals within i's bounding box. To define a bounding box we will use a bounding box threshold with regards to size and a threshold with regards to performance. See figure 3.13 for a illustration of the Bounding Box Diversity. To avoid differentiating unnecessarily among individuals of similar diversity their diversity scores are mapped into a specified number of bins using a squashing function[3]. We will be using 10 bins.

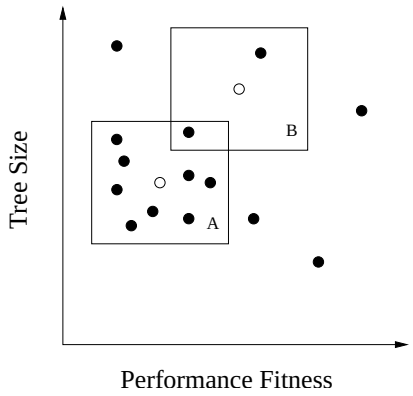


Figure 3.13: Individuals *A* and *B* are the outlined individuals in the center of the two bounding boxes. Individual *B* has better diversity than individual *A*.

To preserve good solutions once discovered we copy the individuals contained within the archive into the new population. An overview of the settings for this test can be seen in table 3.1.

The results from this test are very satisfying. First of all a high degree of diversity is maintained throughout the run. Figure 3.14 shows a plot of all predator individuals from the last generation. The individuals are spread nicely across the entire Pareto front.

The capture and evasion rates for the two populations can be seen in figure 3.15. The predator population rapidly increases in overall performance in two distinct parts of the evolution. The first part is approximately in the first 25 generations of the run and the second part from generation 70 to 92. The prey population is unpaused in generation 26 but paused again briefly in generation 43 and again later in generation 57. After being unpaused in generation 74 the prey are not paused again. Learning cycles in the prey population are not as evident as the learning

[3]The need for doing this became apparent in a test not detailed in this report. Briefly, if the diversity scores are not mapped into bins the Pareto front will be filled with a large amount of individuals with almost identical strategies, which have a small tradeoff between diversity and one of the other remaining objectives. In one particular bad case an abundance of size one individuals were inserted which differed slightly in performance and diversity.

Parameter	*Value*
Number of Predators	4
Population Size	**700**
Archive Size	**70**
Objectives	Performance Fitness, Tree Size, **Bounding Box Diversity**
Generations	**200**
Turns	300
Total Fitness Pool (both)	Teacher Set Size · Opponent Population Size · 1200
Teacher Set Size	50
HoF Sampling Size	**50**
Selection Method	Tournament Selection
Tournament Size	**4**
Breeding Methods	Crossover, Mutation
Crossover Probability	95%
Mutation Probability	5%

Table 3.1: Test settings for the modified FOCUS test. Altered parameters are in **boldface**. Some were altered (compared to the last tests from our previous semester) to speed up the test. Due to the lowered selection pressure the number of generations is increased to 200.

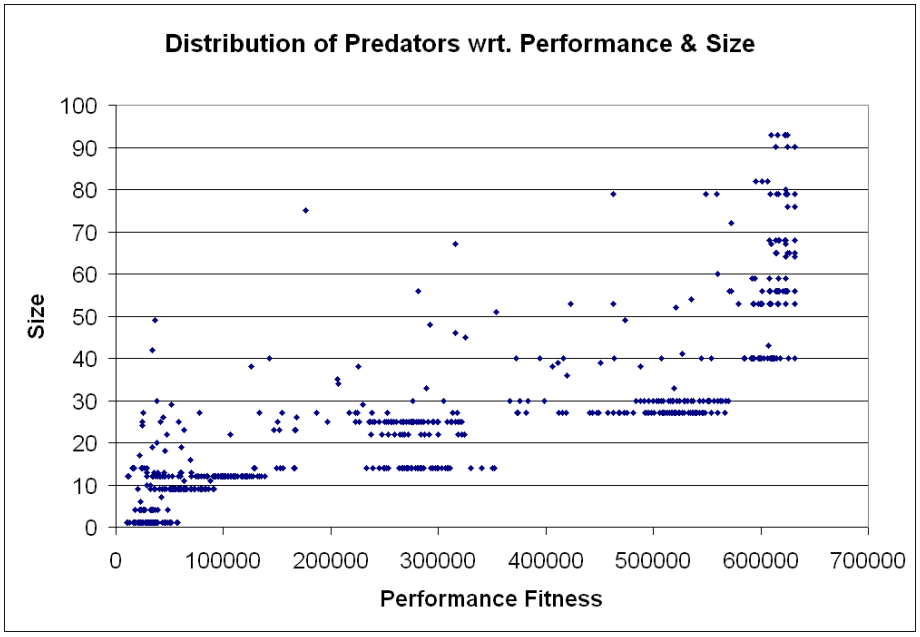


Figure 3.14: Distribution of predators with respect to performance and size.

cycles in the predator population. The prey does however manage to learn to evade both most predators and walls. The strategy of the best prey agent prefers to make the prey stay away from the walls. When pursued the prey will generally turn away from a wall before getting too close while still attempting to evade pursuing

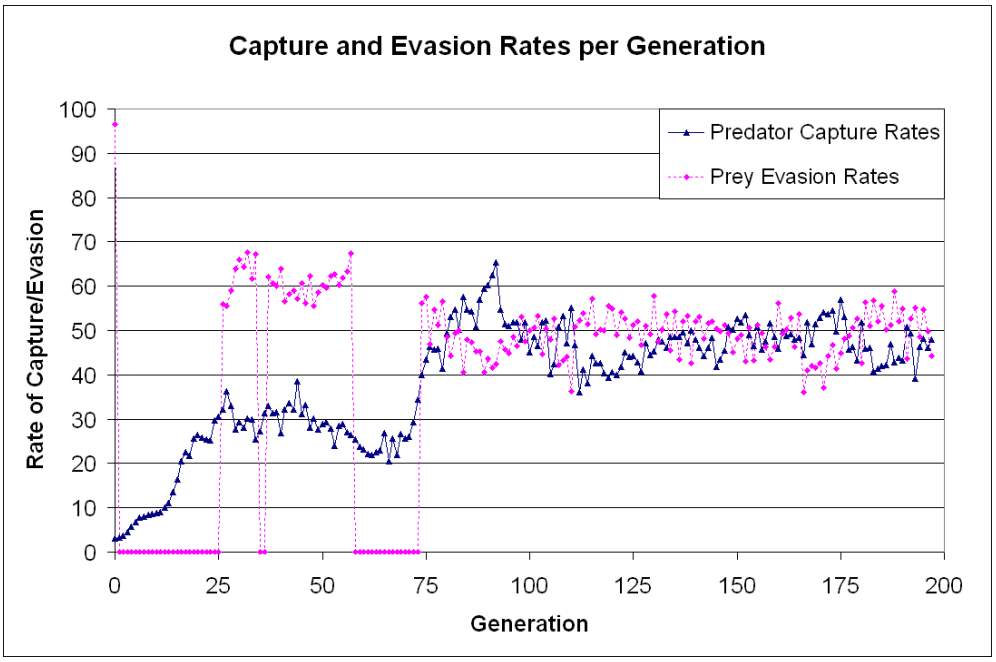


Figure 3.15: Capture and evasion rates per generation.

predators. It is generally very successful at doing both. Its strategy tree can be seen below:

```
 1 (ITE (LessThan (Md (60, 23)
 2                    (57, 39))
 3                (Md  CellOfNearestWall
 4                     (CellOfNearest South)))
 5       North
 6      (ITE (LessThan (Md  CellOfNearestWall
 7                          (CellOfNearest North))
 9                      (Md (CellOfNearest West)
10                           CellOfNearestWall))
11            East
12           (ITE (LessThan 17 (Md  CellOfNearestWall
13                                 (CellOfNearest East)))
14                 West
15                 South)))
```

It is interesting to observe that the strategy never references the position of the prey itself but only considers comparisons between the distance to the nearest predator and the nearest wall.

The best predator is generally able to beat the best prey[4]. It does this by approaching the prey from different directions and eventually coaxing the prey to approach the border where it will get stuck. Note that when tested against inferior predator packs the prey never gets stuck against a border. The predator strategy is very interesting since the predators do not actively seek to surround to prey. Rather predators will sometimes based on their relative distance to the prey and nearest peer choose

[4]Pitting them against each other in 100 matches produced 92 captures.

to move away from the prey or perpendicular to a direct line drawn from itself to the prey. When the prey later changes direction the predators which have "moved astray" will often find themselves on the other side of the prey relative to the predators which chose to follow the prey as closely as possible. This somewhat arbitrary way of surrounding the prey is interesting to observe. The strategy seems very different from most heuristics we ourselves would think of. We are excited that the predators have learned that a locally suboptimal decision for a single predator can later in the pursuit lead to a global optimum (i.e. capture). This may be the first step towards evolving explicit surrounding techniques. Observing the best predator strategy it is apparent that it is very dependent on having a high number of turns to capture the prey. To investigate this further we tried pitting them against each other in matches of varying length. The result can be seen in figure 3.16.

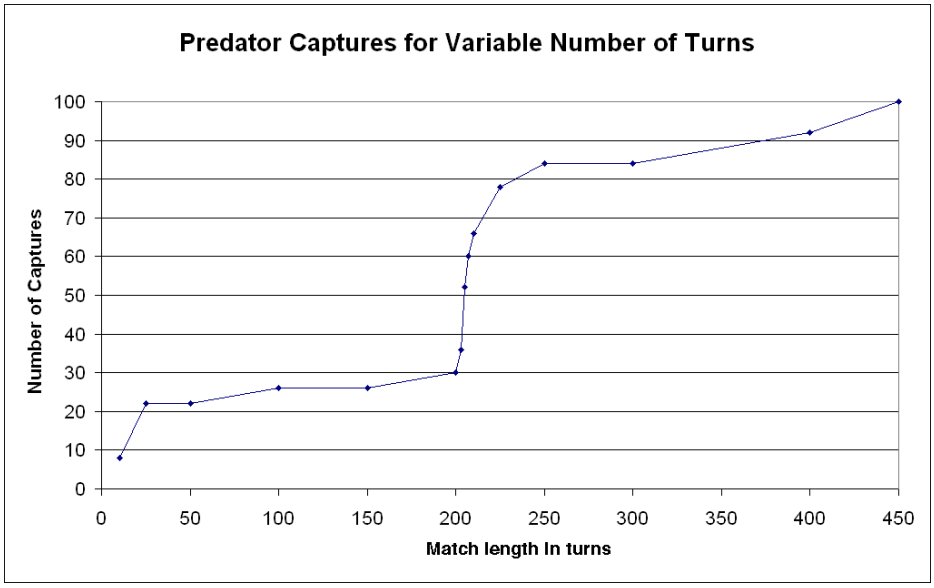


Figure 3.16: Capture rate for best predator vs. best prey using varied number of turns.

The predator strategy needs at least 225 turns to be effective. This is because the dispersion of the predators as a result of predators "moving astray" as described above takes some time to occur and once occurred additional turns are needed in order to approach the prey correctly. A strategy like this would never have evolved would the turn limit have been stricter. The strategy for the best predator can be seen below:

```
 1 (ITE (LessThan (Md (CellOfSelf North)
 2                    (CellOfPrey West))
 3                (Md (CellOfPrey East)
 4                    (CellOfSelf (ITE (LessThan (Md (CellOfPrey North)
 5                                                   (CellOfNearest East))
 6                                               (Md (CellOfPrey North)
 7                                                   (CellOfSelf Decelerate)))
 8                                    South
 9                                    West))))
10      (ITE (LessThan (Md (CellOfSelf North)
11                         (CellOfPrey East))
```

```
12                    (Md (CellOfPrey North)
13                        (CellOfSelf Continue)))
14            North
15            East)
16      (ITE (LessThan (Md (CellOfPrey North)
17                         (CellOfSelf South))
18                     (Md (CellOfPrey West)
19                         (CellOfSelf Continue)))
20            South
21            West))
```

Based on the results presented in this section we conclude that the Modified FOCUS approach allows the maintenance of diversity in our co-evolutionary runs. It will be used throughout the remainder of this report.

3.5 Handling Noise

Introducing diversity maintaining properties into the evolutionary model enables the evolution of better solutions than seen before. A general problem is, however, that in all runs performed the best evolved prey are still inferior to the best evolved predators. As one of our primary goals is to obtain more sophisticated predator strategies utilizing advanced communication we do need better prey in order to measure the effects of introducing advanced communication. Examining the distribution of prey individuals across multiple generations yields one hint as to why the prey are failing to evolve as well as the predators. Figures 3.17 and 3.18 show the distribution of prey individuals for two consecutive generations[5] of a typical test run.

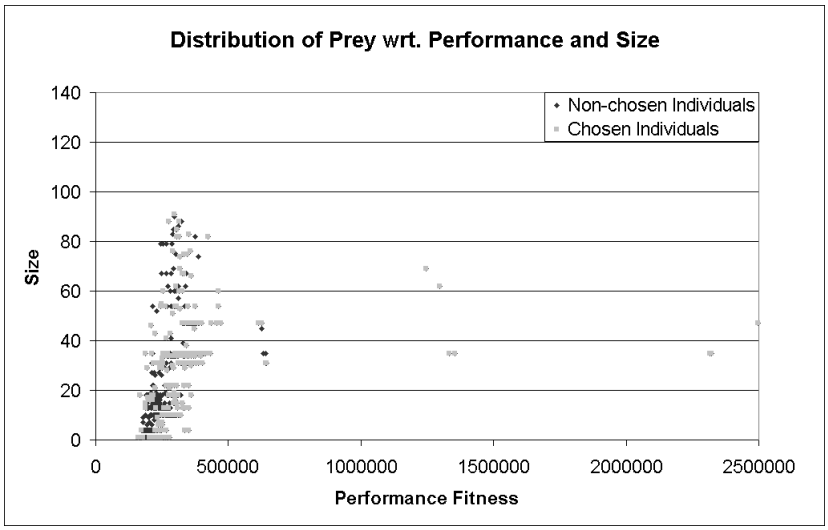


Figure 3.17: Distribution of prey individuals for generation 70.

[5]Specifically generation 70 and 71.

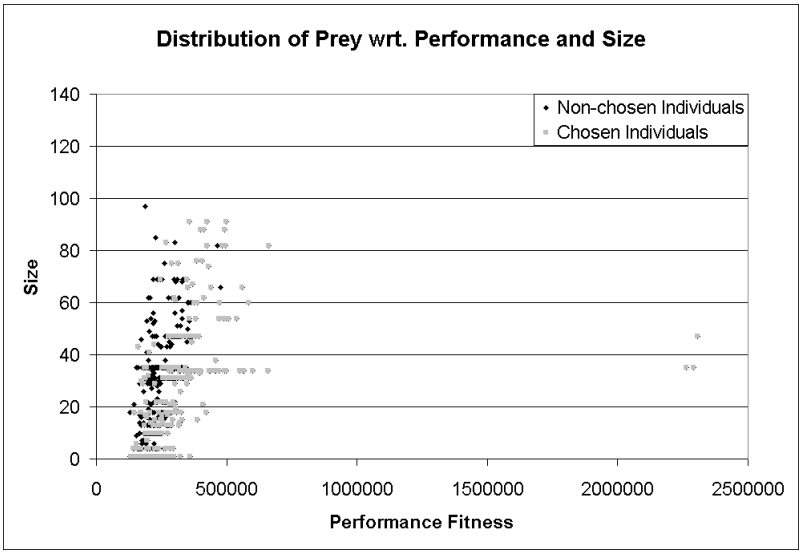


Figure 3.18: Distribution of prey individuals for generation 71.

The entire population in each figure is made up of the union of grey and black markers. The grey markers represent individuals which are selected for the mating pool whereas black markers represent individuals which are not selected. In 3.17 there are four noteworthy individuals with a performance fitness of approximately 130000 as well as a couple of individuals with a performance fitness of more than 230000. In the subsequent generation the small crowd of individuals with performance fitness around 130000 have disappeared even though they are members of the SPEA2-archive and thus copied directly to the new generation. One reason for this happening could be that the individuals depicted in figure 3.18 face a very different teacher set than those in figure 3.17 and therefore achieve very different performance fitness. However, we do not believe this is the case. After discovering the phenomenon we have observed that it usually happens many times each evolutionary run. In fact we have seen it happen when the opposing population is paused and its teacher set thus static. Another - in our opinion more likely - reason for the phenomenon is noise in our environment combined with the use of Competitive Fitness Sharing. The noise in our environment is due to the fact that the randomly assigned starting positions of the predator agents for each match[6] can affect whether or not the predators manage to capture the prey. Thus when a predator pack meets a prey individual with equal level of skill during evolution chance in the form of the randomly assigned starting positions dictates which of the two opponents wins the match. Even worse is it when an inferior prey by luck beats a very good predator that no other prey individuals can beat. Because of the Competitive Fitness Sharing this prey agent will receive an enormous amount of performance fitness for this single match. The four individuals in figure 3.17 have obtained their high fitness this way.

[6]The prey always starts in the middle of the grid world and is not assigned a position randomly.

3.5.1 Solution Approaches

It seems likely that the combination of a noisy environment and use of Competitive Fitness Sharing is prone to problems like the one described above. The higher the selection pressure being used is the more damaging mis-classifications due to noise can be. Even with our very low selection pressure (see section 3.2) we are experiencing problems. We will propose two ways of removing or greatly reducing the problem.

The first approach is simply to remove or reduce the noise in the environment. In our case noise can be reduced by for each match running the match multiple times and using a majority function to determine whether the prey or the predator pack is the winner. This is similar to what we did in the simple environment where all predator packs were evaluated against the randomly moving prey 10-15 times. Unfortunately reducing or removing noise is not always possible and even if possible some cost is usually associated with obtaining the extra degree of precision. In our case we would need to reduce the size of the Teacher Set to make possible the extra matches without unacceptable computational overhead.

The second approach is to reduce the effect of the Competitive Fitness Sharing without eliminating it completely. Thus we hope to still benefit from the positive consequences of using Competitive Fitness Sharing while reducing the problems caused by false classifications. Specifically we propose the use of a parameter which will smoothen the bonus awarded to individuals beating opponents which no or few other individuals can beat. We call this parameter the *Competitive Fitness Dampener* or CFD. Normally the fitness assigned to a student beating a teacher *j* while using Competitive Fitness Sharing is computed as $\sum_{j \in X} \frac{1}{N_j}$, where X is the set of teachers and N_j is the total number of students in the population beating teacher *j*. Including the CFD parameter the equation becomes:

$$\sum_{j \in X} \left(\frac{1}{N_j} \right)^{CFD} \tag{3.1}$$

The CFD parameter is a floating point value in the range]0;1]. Setting the value infinitely small corresponds to having no Competitive Fitness at all while setting it to 1 eliminates the effect of the CFD. The CFD could be used to exaggerate the effect of Competitive Fitness Sharing by setting it to a value above 1. The effect of using the parameter can be seen in figure 3.19. The y-axis displays the fitness awarded to each opponent beating the individual holding the fitness. Thus for each line the "Beat N" value represents that the opponent yielding the fitness depicted in the y-axis is beaten by a total of N individuals. As the CFD parameter approaches 0 the average difference in fitness for opponents beaten by different numbers of individuals is reduced. Thus false classifications of matches versus superior opponents will not boost fitness as excessively as when using pure Competitive Fitness Sharing.

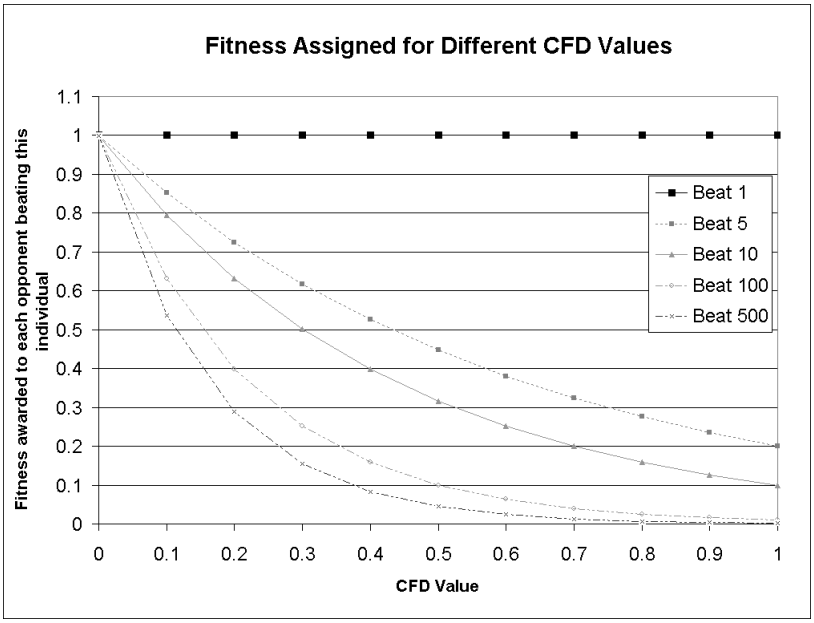


Figure 3.19: Effect on fitness from using various values of CFD.

3.5.2 Tests Using the Competitive Fitness Dampener

Tests are performed with three different values of the CFD. The values are 0.1, 0.25 and 0.5. With a CFD of 0.5 the flow of evolution is very similar to when the CFD is not being used. Examining figure 3.19 this is not surprising as the fitnesses assigned when using CFD 0.5 are quite similar to those assigned when using CFD 1.0 which is equivalent to no CFD. We observe that when lowering the value of the CFD, progress in the evolution is generally slowed down but the noise is also reduced. With a CFD value of 0.1 progress is slowed significantly down. With this value we believe that we loose too much of the positive effects of Competitive Fitness Sharing. With a CFD value of 0.25 progress is also slowed but the reduction in noise makes up for the slowdown. Using the 0.25 CFD we no longer witness the false classification problem described above. Furthermore the Pareto front and general distribution of prey individuals with regards to performance and size indicate that the evolution is progressing better when using the 0.25 CFD. Figure 3.20 shows the distribution of prey individuals before introduction of the CFD while figure 3.21 shows the distribution in a run using the 0.25 CFD.

The final strategies evolved using the 0.25 CFD are better than the best strategies described in section 3.4 when directly compared against each other. We have compared them in table 3.2. The individuals described in section 3.4 are referred to as "No CFD". Examining traces from the newly evolved individuals their strategies seem like refined versions of the strategies of the individuals from section 3.4. However, the new predator strategy does not have the tendency to arbitrarily spread out its predators which was evident in the older predator strategy. The tree for the new predator and prey are 73 and 118 nodes. The strategies can be seen in appendix F.1.

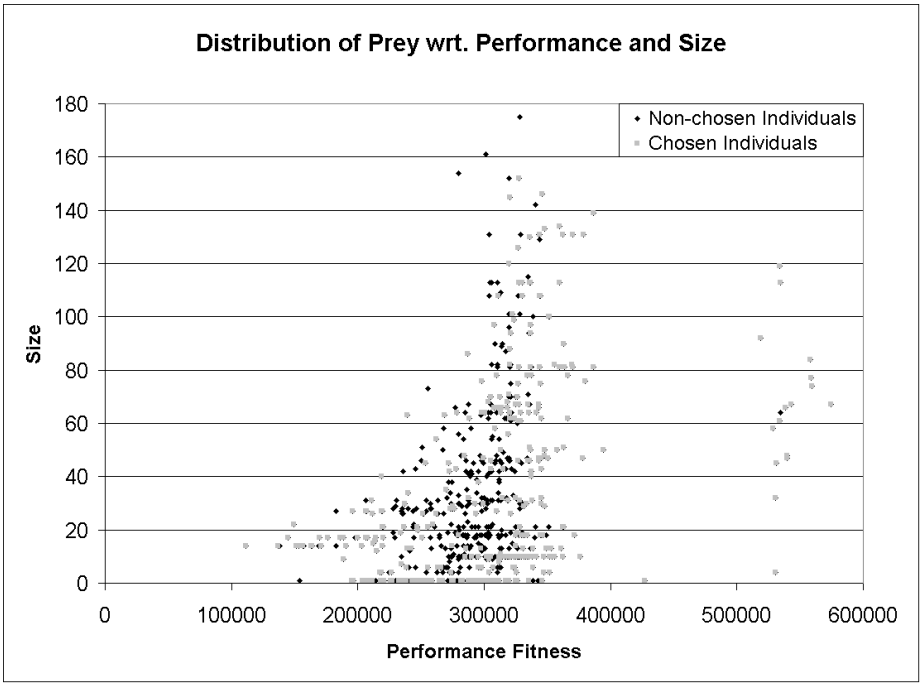


Figure 3.20: Distribution of prey Performance and Size before introduction of the CFD.

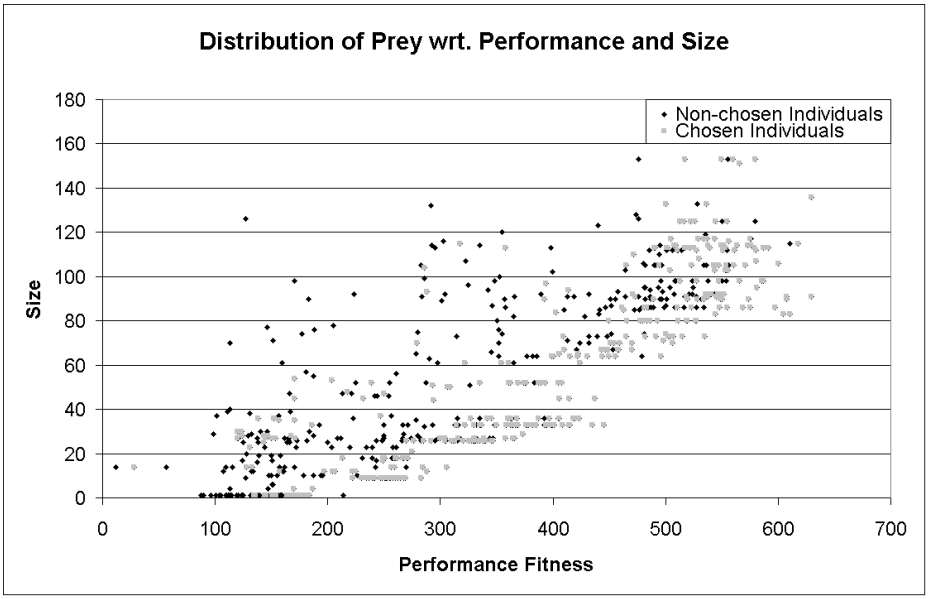


Figure 3.21: Distribution of prey Performance and Size after introduction of a 0.25 valued CFD.

	CFD Prey	No CFD Prey
CFD Predator	94%	98%
No CFD Predator	85%	92%

Table 3.2: Certainty in percent with which predator beats prey.

Note that while the newly evolved agents are better than the agents found earlier when compared directly against each other we cannot immediately infer that the new agents are universally better. In Rydtoft and Rasmussen [43] we showed by example that intransitive relationships can occur in our environment. Thus the mere fact that a pair consisting of a predator pack and a prey agent seem superior to another pair when directly compared does not prove that it is the case. In order to determine if some predator/prey pair is better than some other pair we need firstly to define precisely what we mean by better. Secondly we need a way to measure individuals according to this defined better-than relation. For predators we define the best possible strategy as the strategy which allows the predators to capture all prey strategies which can be created[7]. Let us call the set of all possible prey strategies $\mathcal{P}$. The best possible prey can be defined in a similar manner. Determining whether one predator strategy **A** is better than predator strategy **B** can then be done by comparing the number of members from $\mathcal{P}$ which **A** can beat with the number of members from $\mathcal{P}$ which **B** can beat. But what happens if **A** beats less prey than **B** but the prey that **A** beat are far more interesting and difficult? One might be tempted to say that a strategy which beats so seemingly interesting and difficult strategies is better than a strategy which beats many but poor strategies. Without an external objective measure it is utterly impossible to determine objectively if one pair of strategies is "better" than another pair. That being said our subjective measurement in table 3.2 as well as the flow of evolution in figure 3.21 lead us to conclude that the predator pack and prey individual evolved using the 0.25 CFD is more promising and "better" than what we have seen before.

[7] A purely theoretical measure since an intractable number of prey strategies can be created.

Chapter 4

Communication in the Extended Pursuit Environment

In chapter 3 we implemented and discussed features which promote diversity and reduce negative effects of noise in our co-evolutionary environment. We are now satisfied with the flow of our co-evolutionary runs. The issue regarding balance between our competing populations has remained unresolved. The predator population has in every run gained superiority and even though evolved prey strategies are very capable they are no match for the best individuals from the co-evolved predator population. In this chapter we introduce a change in our environment making it more difficult for the predators to capture the prey. This change is described in section 4.1. Subsequently tests are performed with varying degrees of communication. In section 4.2 a test is performed in which the predators have no means to detect each other - i.e. no communication. A new means of visual communication is provided for the predators in section 4.3 and new very interesting strategies are evolved. Section 4.4 describes the introduction of aural communication. No interesting strategies with aural communication are evolved even though multiple approaches are attempted.

4.1 The Burrowing Prey

As described in section 3.5 the introduction of the Competitive Fitness Dampener has improved our evolution flow with regards to the prey population and allowed us to produce more interesting strategies than previously. However, the predator population is still consistently outperforming the prey population and the improvements in the evolution-flow for the prey merely resulted in the evolution of an even better predator strategy. As described in section 3.5 a greater challenge for the predators is needed before the benefits of introducing advanced communication may be measured. We provide this increased challenge by making a change in the rules of our pursuit game.

As stated earlier in Rydtoft and Rasmussen [43] we like to model our environ-

ment to closely resemble nature. In nature many species with natural enemies have evolved countermeasures increasing their chance of survival when being pursued by predators. Some prey species are very nimble and agile and highly maneuverable. Our prey species already has this characteristic. Other prey species may have very high stamina and thus be able to run for an extended duration of time tiring the pursuing predators which have only a limited reserve of energy. This is for instance the case in the predator/prey relationship between the African cheetah and gazelle. Another common prey ability is the ability to dig holes for hiding. Some prey species live in burrows underground and will rarely stray far from their burrow. As with high-stamina prey species this means that if the predators do not capture the prey very quickly they will loose their opportunity. We will provide a such advantage for our prey species. We will call prey using it "burrowing prey".

The burrowing prey will be realized by defining a rule which states that once a predator has moved sufficiently close to the prey a timer will start. This timer will decrement by one each turn and if it reaches zero before the prey has been captured it means that the prey has reached its burrow and is safe. Thus we will need to define two parameters. The first will specify the Manhattan distance between a predator and the prey which will make the prey feel threatened and seek shelter (i.e. which will start the timer). This parameter will be referred to as the "Burrow Distance". The other parameter specifies the number of turns the predators have to capture the prey before it reaches its burrow. This parameter is called the "Burrow Time".

Test of Burrowing Prey

Parameter	*Value*
Number of Predators	4
Population Size	700
Archive Size	70
Objectives	Perf. Fitness, Tree Size, Bounding Box Diversity
Generations	**550**
Turns	300
Total Fitness Pool (both)	Teacher Set Size · Opponent Population Size · 1200
Teacher Set Size	50
HoF Sampling Size	50
Selection Method	Tournament Selection
Tournament Size	4
Breeding Methods	Crossover, Mutation
Crossover Probability	**90%**
OneNode Mutation Probability	**5%**
SubTree Mutation Probability	**5%**
Dominance Threshold	**80%**
Competitive Fitness Dampener	**0.25**
Burrow Time	**30**
Burrow Distance	**8**

Table 4.1: Test settings for first test with burrowing prey.

The following test shows the burrowing prey species is much harder to catch than non-burrowing prey. Several tests have been performed and parameters have been tuned to accommodate the flow changes resulting from the introduction of the burrowing prey. The function and terminal sets used are those shown in appendix D. The parameters for the most successful test can be seen in table 4.1. We introduce a new mutation type - the one node mutation - which mutates a single node to another node adhering to the same STGP constraints. This type of mutation is very useful if for instance we need to swap a single West Tack with a North Tack. As the prey species is now more difficult to beat we increase the Dominance Threshold to 80%. Having it at the usual 70% makes it too difficult for the predators to evolve sufficiently to unpause the prey (i.e. be dominated by the prey population by no more than 70%), resulting in runs in which the prey population is paused in the vast majority of the generations. Note that the two new parameters "Burrow Time" and "Burrow Distance" have also been included with values of 30 and 8 respectively, meaning that once a predator gets within a Manhattan Distance of 8 squares from the prey the predator pack will need to capture the prey within the next 30 turns. The number of generations has been increased to 550. This is partly because the - otherwise beneficial - Competitive Fitness Dampener has a tendency to slightly slow the evolution and partly because the introduction of the burrowing prey makes each generation take much less computational time. This occurs because most matches are settled before the 300 turn limit as the prey is either captured or burrows itself.

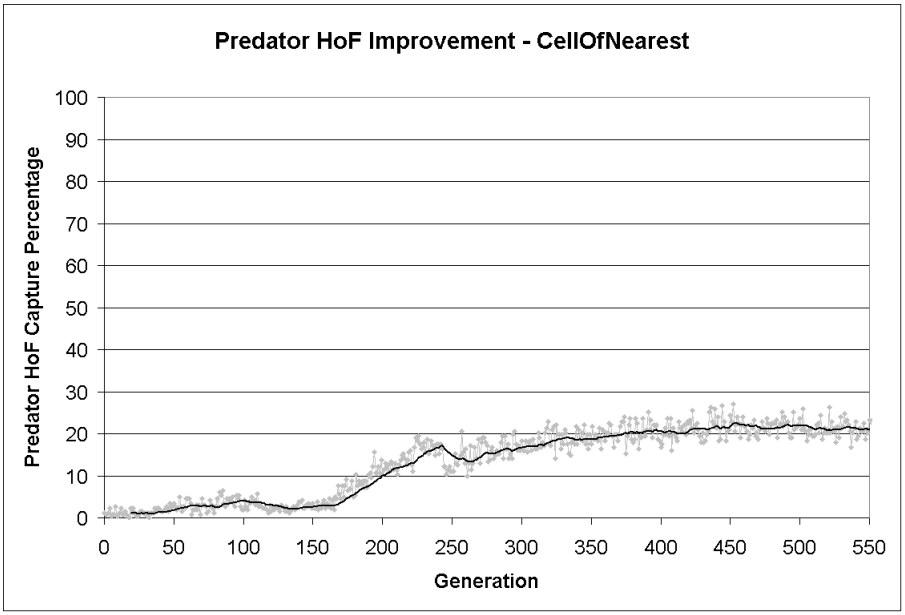


Figure 4.1: Performance improvement of predator HoF members. Trend line added due to noise.

We have measured the improvement of the predator population using the objective measure introduced in section 3.1.2. The results can be seen in figure 4.1. A steady improvement can be observed from generation 175 to approximately 230. After a

brief decline spanning 30 generations progress occurs again but at much slower rate. After generation 375 no significant changes occur. The best predator evolved is able to beat the best prey only 14% of the times they are matched against each other while it beats roughly 20% of the HoF prey. This is testament to the great increase in difficulty induced through the use of burrowing prey. The 14% can be compared with previous capture rates for evolved predators in table 3.2. Using burrowing prey we can now introduce advanced means of visual and aural predator communication.

4.2 Test Using No Communication

This section describes the first in a series of tests intended to collectively document benefits of introducing different means of communication. The predators used in the test in section 4.1 used the CellOfNearest() function. This function returns the cell of the predator closest to the calling predator and thus constitutes a form of visual communication. Before introducing advanced forms of communication we will make a test using predators without any form of pack communication. Thus the predators used in this test will be able to see the prey but will not be aware of each other. The test will use the same parameters as the test in section 4.1. Likewise the populations will use the same function and terminal sets with the exception that the predator population will not have access to the CellOfNearest() function.

The improvement graph for the predator HoF can be seen in figure 4.2. Improvement occurs earlier that in the test in section 4.1. This is likely due to the reduced search space of possible solutions resulting from the exclusion of the CellOfNearest() function. After generation 325 the trend line flattens and no further improvement can be identified. The best HoF predators from this test actually beat more HoF prey than those from section 4.1. However, this is likely caused by the insertion of a higher number of inferior prey into the HoF in this test.

The best predator beats the best prey in only 1% of the test matches. We have compared the best predator and prey from this test with the best found in section 4.1. The result is shown in table 4.2. The predators evolved while using the CellOfNearest() predator function are better but only slightly. This comparison is good indication of the weakness of the visual communication offered by the CellOfNearest() function. The next section will elaborate on this issue and provide details of a modification to the visual communication.

	CellOfNearest Prey	**No Communication Prey**
CellOfNearest Predator	14 %	5%
No Communication Predator	0%	1%

Table 4.2: Certainty in percent with which predator beats prey.

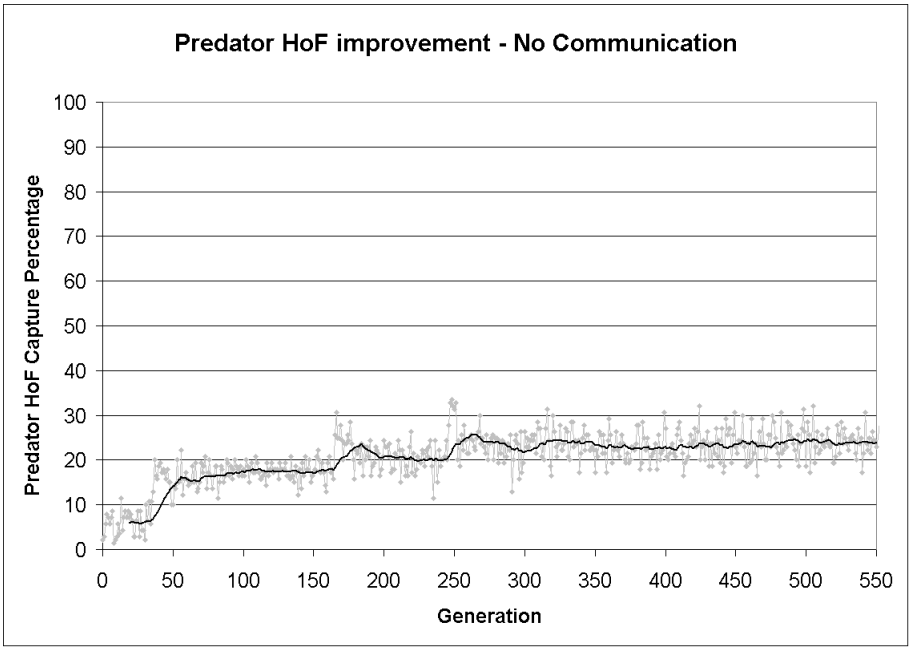


Figure 4.2: Performance improvement of predator HoF members. Trend line added due to noise.

4.3 Test Using Modified Visual Communication

This section describes a redesign of the visual communication in the extended pursuit environment and test results from evolutionary runs applying the related communication functions. The functions intended as a basis for visual communication are reevaluated and found to be unsatisfactory. A redesign is motivated and the modifications are described. Test results from evolutionary runs using visual communication show great increase in predator performance and in their ability to capture burrowing prey.

4.3.1 Redesign of Visual Communication Functions

In Rydtoft and Rasmussen [43] table 5.4 page 60 the four functions intended to be used for visual communication are listed. The names of the functions are CellOfNearest(), CellOfFarthest(), CellOfNearestRight() and CellOfNearestLeft(). The exact functionality of each of the functions can be found in the table referenced above. The essential functionality of all of these functions is to convey information about other predators' positions in the grid world to the predator using the functions. Although the functions do convey this information we have discovered that referencing with these functions makes it very difficult for predators to coordinate movement relative to each other. The weakness of the design was discovered while constructing heuristic predator strategies to be used during the fall semester exam presentation. Furthermore the comparison of the tests in section 4.1 and 4.2 shows

that the advantage of using CellOfNearest() as opposed to no pack communication is modest. When a predator uses the CellOfNearest() function the cell of the nearest predator is returned no matter in which direction the nearest predator is. After one single move this function might return a cell located in the directly opposite direction of the previous cell. This makes it difficult for the predator using the function to move relative to a fixed point. The CellOfNearestRight() and CellOfNearestLeft() functions have the same problem. The problem becomes apparent when a predator changes direction. Since e.g. the CellOfNearestLeft() returns the cell of the nearest predator to the left of the calling predator this may also change rapidly when the calling predator turns itself. These unwanted properties of the functions make it impossible for the predators to reference each other and move accordingly in a consistent manner. Therefore the functions are partly removed and partly redesigned to better facilitate visual communication amongst the predators.

The Modified CellOfNearestRight

Apart from the CellOfNearestRight() which is altered the other visual communication functions[1] are discarded. The CellOfNearestRight() (CONR) is redesigned to make referencing easier for the predators regardless of their direction. The new CONR function extends an imaginary line from the calling predator through the position of the prey. This line is independent of the direction in which the predator is currently facing. CONR then calculates the Manhattan distances from the prey to all predators located to the right of this line and returns the cell of the predator with the shortest distance to the prey. The function allows predators to reference each other in a counter-clockwise manner while moving and changing direction. A situation in the grid world can be seen in figure 4.3 where the functionality of CONR is illustrated. The CONR has the advantage that it incorporates the position of the prey which allows the predators to place themselves and move relative to both the prey and the other predators in an easy manner.

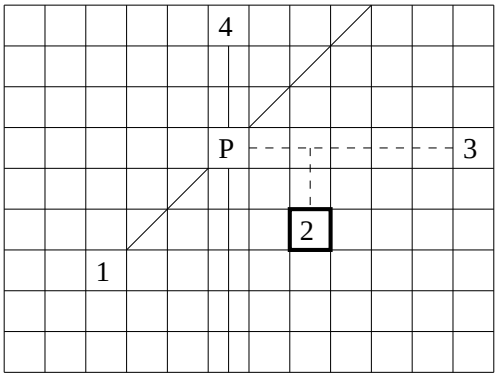


Figure 4.3: Figure illustrating the new CellOfNearestRight() function. Predator 1 is using the function and receives the cell of predator 2 which is located to the right of the line drawn from predator 1 through the prey. The cell of predator 2 is returned because its Manhattan distance to the prey is less than that of predator 3. If no predator is present to the right of the imaginary line no cell is returned.

[1] CellOfNearest(), CellOfFarthest(), CellOfNearestLeft().

4.3.2 Tests and Results

A test is performed in which predators are allowed use of the CellOfNearestRight() function. The test settings used are exactly the same as those listed in table 4.1. As it is evident from this table the co-evolution is continued for 550 generation. In figure 4.4 the improvement of the predators inserted into the HoF is depicted. The graph indicates a slow improvement until approximately generation 320 where a more steep increase takes place.

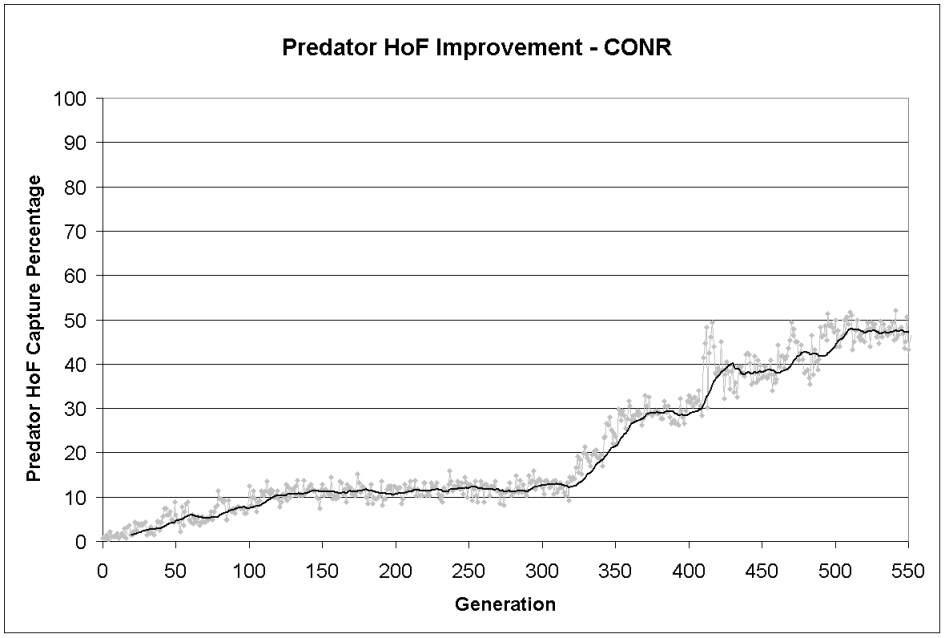


Figure 4.4: The figure depicts the capture percentage of predators in the HoF. The predators are using the CellOfNearestRight() function.

The best predator and prey strategies evolved during the run can be seen in appendix F.2. They are pitted against each other 100 times and against the individuals evolved in sections 4.1 and 4.2. The results are listed in table 4.3 and show that not only can the best newly evolved predator pack capture old prey nearly 100% of the time but the new prey is also able to evade old predator strategies almost flawlessly. This shows that introducing beneficial functionality for the predators has helped both populations to co-evolve to a new level of performance. Both the new prey and the new predator strategies are superior to the strategies evolved earlier.

	CellOfNearest Prey	**No Communication Prey**	**CONR Prey**
CellOfNearest Predator	14%	5%	1%
No Communication Predator	0 %	1%	0%
CONR Predator	99%	100%	65%

Table 4.3: Certainty in percent with which predators beat prey.

When examining the matches between the evolved CONR predator and co-evolved burrowing prey several interesting situations occur. Some of these will now be described.

It is evident from viewing the matches that predators use the CellOfNearestRight() to move and maneuver around the prey and coordinate in a much higher degree than previously seen. Previously evolved predators which start in the same quadrant of the map, e.g. to the south-west of the prey, would often all move either north or east to get aligned with the prey. The newly evolved predator strategy makes the predators fan out from such a starting position and approach the prey from several directions. Attempts to completely surround the prey have not been seen to be successful, however three out of four directions have been covered by predators forcing the prey to escape the only possible way out. The good predator strategies utilize the ability to accelerate and move two squares per turn.

Comparing predator strategies from earlier generations around 380 with the latest from 550 reveals interesting differences which indicate that evolution has been taking place. The younger predator strategies have learned to accelerate and rush the prey from several directions with some success. However, when predators rush towards the prey they do not slow down or stop when close to the prey. Instead they rush towards the prey merely to see it sidestep and the predator continues past it and then slows down. In the latest strategy the same pattern is still observed but the predators no longer rush past the prey. They decelerate so that they are precisely aligned with the prey after it changes direction and continue the chase, i.e they have learned to master their speed advantage and speed up and slow down at the right times which is very impressive. Both the early and the later predators rush towards the prey in very alike manners. The predators take turns in rushing the prey from different directions and if failing to catch it other predators are already rushing towards it. Although the predators do coordinate their attacks, members from earlier predator packs sometimes singlehandedly chase the prey before their teammates get close enough to assist in the capture. This is of course far from optimal since it lowers the chance of capture because the prey burrows. The later generation predators have to a high degree learned to wait until most of their team members are close before starting to rush the prey. A strategy which have been observed several times to be successful. This complex behavior is also testament to the usefulness of the CellOfNearest() function.

The prey initially stays in its starting position until a predator gets too close which the prey then reacts upon. This strategy has an upside and a downside to it. The downside is that since the prey stays in the middle until approached predators have an easier job surrounding it than they would have surrounding a roaming prey. The upside is that since the prey never starts to move until a predator gets close enough to activate its burrow timer the prey will never reach a wall and thus never get caught against a wall.

4.4 Test Using Visual and Aural Communication

Each test performed in the section uses parameters and functions/terminals identical to those of the test performed in section 4.3, with the exception that these tests

use additional functions for aural communication (messaging).

4.4.1 1st Approach with Communication

To enable aural communication among the predators we use the Shout(Msg m) and IsLastMessageOfType(Msg m) functions as described in Rydtoft and Rasmussen [43]. Basically the Shout() function registers its argument message as the last message shouted, each time overwriting the previous message, and the boolean IsLastMessageOfType() returns true if its argument message matches the last message shouted. This approach is depicted in figure 4.5.

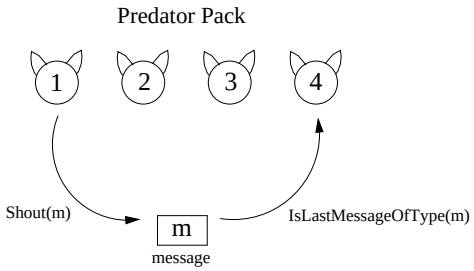


Figure 4.5: 1st communication approach.

We allow the test to continue for 900 generations to accommodate the increased search space resulting from the added functions. The improvement graph for the predator population can be seen in figure 4.6. Although improvement can be seen much noise is present in the graph. The final solutions evolved in this test are not impressive. They possess little of the ingenuity seen in solutions evolved previously in e.g. the test in section 4.3. It is surprising that this small increase in function set size has such detrimental effects on the evolution flow. The best evolved predator strategies do not make organized use of their messaging capabilities and additionally do not make as advanced use of the CellOfNearestRight() function as previous solutions.

Examining the late generation HoF predators it can be seen that they use the Shout() function often but rarely use the IsLastMessageOfType(). The abundance of shouting is likely related to the STGP constraints related to the function. The Shout function has a child of type Msg[2] and another child which must be of type Tack. The Shout() function returns this Tack itself when evaluated. The actual storing of the shouted message is simply implemented as a side-effect to the evaluation of the Shout node. Thus Shout() functions can be placed almost anywhere in a tree and even connected in series. Once introduced they can be difficult to get rid of through evolution. This problem should be resolved by enforcing stricter constraints on the shouting.

Another problem with this communication approach is the ability of the crossover operator to destroy communication functionality in the higher parts of a tree by introducing new subtrees. If for instance a predator has learned to shout the message

[2]This child is the actual message argument to the Shout function. I.e. the message being shouted.

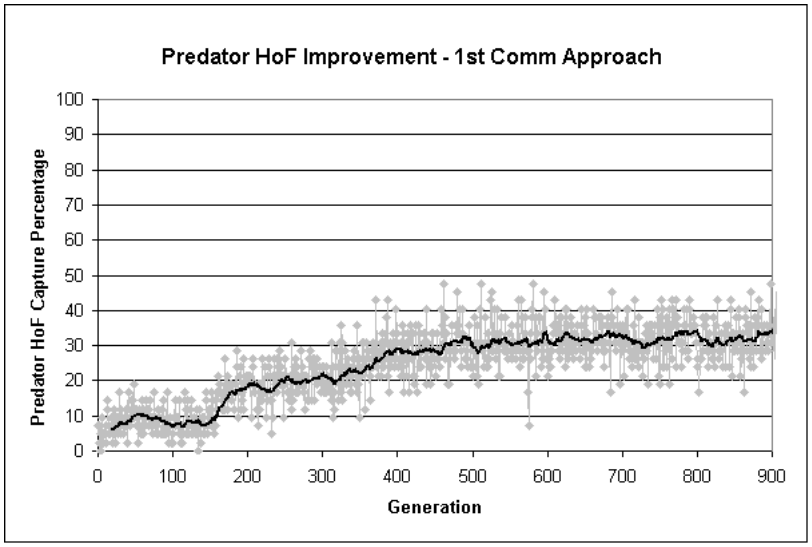


Figure 4.6: Predator HoF Improvement for the first communication approach.

B when it is closer than 20 squares to the prey and its tree gets a new branch which makes it always shout message **B** then the previous functionality is destroyed. An additional problem is that with only one message being stored at any time it is infeasible that one predator will be able to have its message "heard" by all the other predators in the pack. If this is to be possible each of the remaining predators will have to wait shouting until the next turn to avoid disrupting the stored message.

4.4.2 2nd Approach with Communication

To reduce excessive shouting we remove the existing Shout() function and introduce a new version with tighter constraints. The new function is called IfThenElseShout() (ITES) and behaves similarly to our ordinary IfThenElse(). The ordinary IfThenElse() evaluates a boolean child and based on the value of the boolean evaluates the subtree rooted at one of its remaining two children. In the new ITES version the first of these children will have a message attached which is shouted prior to evaluating the subtree rooted at that child[3]. Thus the semantics of the ITES are:

```
If (Boolean b == true) <shout message m> <eval child 1>
If (Boolean b == false) <eval child 2>
```

With this new function we reduce the shouting as messages can only occur at places in the tree which can hold an IfThenElseShout(). Additionally we make sure that the shouting occurs just after the evaluation of a boolean, i.e. after a decision has been made. We hope this will make the shouting of messages depend on their

[3]Technically the message is a child in itself which is evaluated just before the subtree rooted at the other child.

attached decisions.

We attempt to remove the ability of later shouts to destroy the meaning of earlier shouts. Thus for each predator only the first shout will count. We hope this will prevent situations in which crossovers will include disturbances by introducing new shouts closer to the leaves of the tree. Likewise we will store separately the messages for all predators so if for instance predator 1 shouts **A** and predator 2 shouts **B** predator 3 will still be able to detect that an **A** message was shouted. In predator 1's subsequent turn it will overwrite its message so each predator will have only one message stored at any one time. If a predator chooses not the shout during its turn its previous message will be erased. In order to better fit with the new semantics the IsLastMsgOfType() function will be renamed to WasMsgShouted(). The new message approach can be seen in figure 4.7.

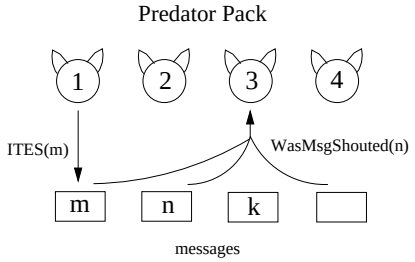


Figure 4.7: 2nd communication approach.

The predator HoF improvement graph for this run is similar to that of the previous run and has been omitted for the sake of brevity. Examining the statistics of the test run we can see that letting the first shouted message in a predator be the only one stored - while solving the problem with overwriting the meaning of the first message - creates new problems. A majority of the initial good predator solutions which spread within the population all shout the message **A** regardless what happens. The shouting of this message happens very early in the predators' tree and subsequent shoutings are all disregarded. As the evolution proceeds we reach a point in which nearly all predators inserted into the HoF have this characteristic. It seems to be very difficult to alter the early parts of the trees containing the first message. Any potentially useful communication functions developing further down the trees of these predators have no meaning whatsoever and it is evident that the best evolved predators do not utilize their ability to communicate.

4.4.3 3rd Approach with Communication

The design of the communication functions utilizing messaging is altered once more. To avoid the problems described above each predator in a pack will be allowed to shout all the possible messages as many times as desired. When a message **A** is shouted and afterward a message **C** is shouted both are remembered. The number of times a predator shouts the same message has no impact. Each type of message will only be registered once. When a predator has its turn it can check if any of the other predators shouted a given message and act upon this. Thus if a

predator checks for the presence of message **B** it will receive an acknowledgment if any of its brethren shouted this message during their last turn. When a predator performs this check messages which were shouted by this predator in its previous turn are not considered. However, if the predator shouts a message **A** in the top of its tree and deeper in the tree checks if anyone shouted **A**, then its own shouted **A** counts. The IfThenElseShout() is still used as shouting function as in the previous setting.

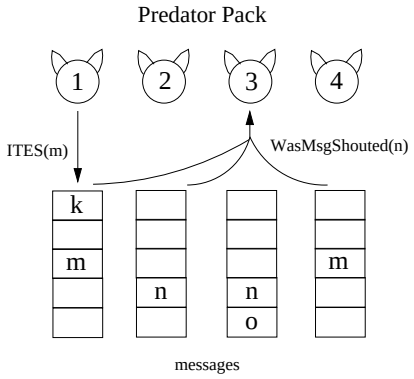


Figure 4.8: 3rd communication approach.

The HoF improvement graph for this run can be seen in figure 4.9. Notice that we have allowed this run to continue for 1200 generations. After generation 800 significant progress occurs and the resulting predator strategies are fairly competent. Many late generation predator strategies have the property that when one predator gets sufficiently close to the prey all predators will accelerate and attempt to capture the prey before it burrows. At first we thought this functionality to be due to communication but examining the strategies reveals that it is in fact clever use of the CellOfNearestRight() function which allows the predators to do this. Late generation predators make no organized use of the WasMsgShouted() function.

4.4.4 Additional Communication and Discussion

In addition to the aural communication attempts described in the three preceding sections other attempts have been made to promote the use of aural communication. We will not go into much detail with these attempts and their results but merely mention them briefly.

The Invisible Prey

In this attempt the prey is invisible until the predators get within a specified distance of it. This distance is longer than the Burrow Distance which allows the predators to see the prey without activating its burrow timer. The intuition is that predators should shout a message alerting the other predators of the position of the prey once it is discovered. The function LastPositionOfMsg(msg) allows the predators to extract the last position from which the argument message is shouted.

Evolved predators do not utilize messaging. In the best strategies evolved a preda-

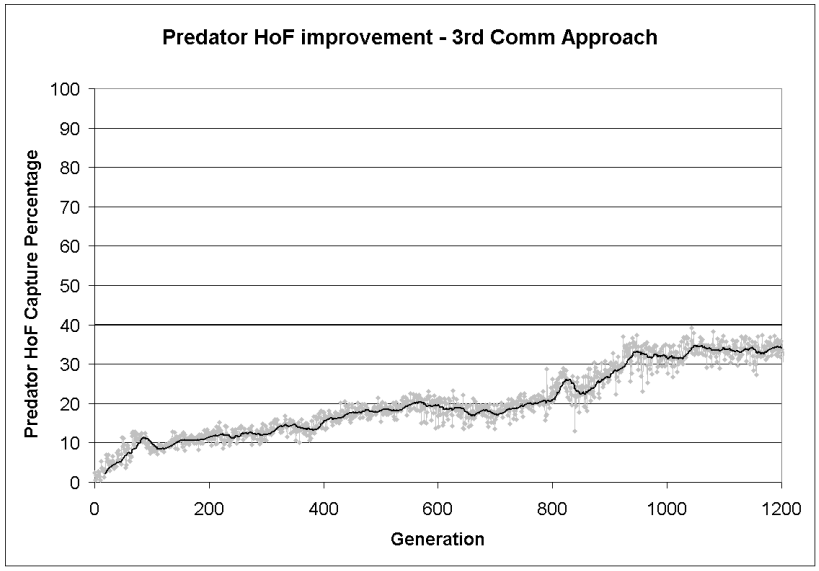


Figure 4.9: Predator HoF Improvement for the third communication approach.

tor will accelerate and attempt to capture the prey by itself once it detects its position.

Non-accelerating Predators

The acceleration ability is removed from the predators. The intuition behind this approach is that predators without the ability to accelerate will have greater need for surrounding the prey and will need the aural communication to do this. The learning curve for this approach seems too steep. The prey population consistently evades the predators and no advanced predator strategies are evolved.

Two-stage Addition of Functionality

In this attempt the evolution is performed in two stages. In the first stage the function set used in the test in section 4.3 is used. Once the predators have mastered the use of the CellOfNearestRight() function as well as the ability to accelerate and decelerate the messaging functionality is made available and introduced through mutation. The intuition behind this approach is that the messaging functionality is only needed at a very late stage in the evolution and that we will benefit from first introducing it once all other functions are being used appropriately. Also in this approach the predators fail to make organized use of the aural communication functionality.

Discussion

None of the attempted messaging approaches have been able to spur the evolution of communicating predators. We believe that there are multiple reason that we repeatedly fail to evolve predators utilizing aural communication. According to the Building Block Hypothesis GA/GP evolution relies on successful combination of small highly fit sub-solutions. These sub-solutions furthermore need to be

local meaning that they should keep their functionality when inserted into a new solution. With the messaging each communication event has two parts; the sending and receiving of messages. Often these parts are placed at different points in a tree and often only one of these parts will be transferred to a new tree. Probabilistically speaking there is a high chance that once evolved, messaging constructs relying on functionality from two separated parts of the tree will be destroyed by the genetic operators. Another reason that messaging fails to evolve can be the power of the CellOfNearestRight() function. In the test in section 4.4.3 we saw that clever use of the CellOfNearestRight() function can provide functionality of a type we had originally anticipated would be evolved through messaging. The CellOfNearestRight() is much easier to include in a small building block and does not rely on functionality from other parts of the solution. That the CellOfNearestRight() can be used the way described in section 4.4.3 means that the actual value of the messaging communication is less than anticipated. This combined with the fact that evolving messaging functionality is very complex compared to evolving the functionality made possible by the CellOfNearestRight() means that the chance of utilizing messaging is very small. The potential benefits to be harvested from simple message passing are too few to allow them to be evolved. The more difficult it is to use new functions and the less real advantage the functions offer the less likely it is that the functionality offered by the functions will be evolved and utilized.

Even though we have failed to evolve predators utilizing aural communication the results in section 4.3 confirm that predators using visual communication to coordinate their task are much more successful than non-communicating predators. This result confirms our hypothesis that cooperating agents can produce better results than non-cooperating agents solving the same problem. This result is of course not true if the problem to be solved is of a nature which inherently cannot benefit from teamwork.

Chapter 5

Co-evolutionary Disengagement and the Interleaved Approach

In this chapter the only two solution approaches to the disengagement problem that we have been able to find in literature are compared with our own Interleaved Competitive Co-evolution Approach. Additionally comparisons with an approach using no disengagement countermeasures are performed. After this comparisons are made between Interleaved Co-evolution, Regular Co-evolution and Stepwise Co-evolution. The comparisons are based on tests performed in the extended pursuit environment. Lastly the performance of the Interleaved Approach in an environment widely different from the extended pursuit environment are performed.

The chapter is structured as follows. In section 5.1 the phenomenon of disengagement is explained in general and within the extended pursuit environment. In 5.1.1, 5.1.2 and 5.1.3 we explain our own approach intended to overcome disengagement as well as the two approaches found in literature. Both of the latter solution approaches are implemented in the extended pursuit environment and tests are performed to compare all three approaches as well as the approach without countermeasures. These tests are documented in section 5.2. Section 5.3 describes the comparison between Interleaved Co-evolution and the two other mentioned co-evolutionary paradigms. Finally section 5.4 documents the test of Interleaved Co-evolution in a new environment.

5.1 Disengagement Countermeasures

The dynamics of competitive co-evolution are complicated and not all the phenomena which may hinder or misguide the evolution have been discovered. One phenomenon which has been found and documented is the disengagement phenomenon which is reported in e.g. Watson [55], Werfel [56], Cartlidge [5] and Rosin [45]. When two opposing populations are evolved against each other the

individuals from both populations are evaluated against all or a selected subset of the opposing population. Based on the outcome of the evaluations individuals are selected for breeding. Individuals which are considered of great value due to their evaluation results are thus selected and assumed beneficial for the further co-evolution of both populations. This fundamental assumption about the dynamics of the co-evolutionary approach is true if both populations are equally matched and thus guide each other to increasingly better solutions. This desired phenomenon is commonly referred to as the "arms race". The ability to differ amongst individuals in a population is therefore essential in order to be able to select good individuals, but not always guaranteed. If one population evolves solutions which none of the opposing individuals are able to counter a situation arises in which the dominated individuals all score identically poor fitnesses. This makes it impossible to guarantee the selection of the best individuals for breeding which again leads to the disappearance of the fitness gradient driving the evolution resulting in a period of degrading genetic drift. When this happens the populations are said to be disengaged from each other. Disengagement is particularly likely to occur in environments where one population at some point has an advantage over the other population. For instance in the pursuit environment the prey task of evasion will initially be easier to accomplish than the predator task of capture.

The disengagement phenomenon was first encountered during initial tests in the extended pursuit environment and is documented in Rydtoft and Rasmussen [43] in section 5.7.2 along with a proposed solution to the problem. The disengagement was evident early during evolution where the predators evolved a strategy which no prey seemed to be able to counter. A test in section 5.2.1 shows that the disengagement phenomenon is still present in our environment.

5.1.1 Interleaved Competitive Co-evolution

The Interleaved Competitive Co-evolution Approach was proposed in Rydtoft and Rasmussen [43] section 5.7.2. The flow of a co-evolutionary run using the Interleaved Approach can be seen in appendix C. The basic idea of the approach is to prevent further evolution of a population which is beginning to dominate its opponent population. This is done by maintaining the dominating population at its current generation while further evolving the dominated population. We usually refer to this as "pausing" the dominating population. The dominating population is paused until the dominated population has evolved sufficiently and is no longer disengaged. This means that for a period of time we will be evolving only one population against a static fitness landscape. Thus our approach actually makes the evolution a mixture of single and co-evolution. The criterion we use to determine when a population should be paused is the so-called "dominance threshold". Predator dominance is measured by for each generation counting the number of played matches which end with a capture and dividing with the total number of played matches. Prey domination is derived in the same way. Note that when added together these two values sum to 1. If for instance the prey dominance exceeds the dominance threshold the prey population is paused. The population is paused until the prey dominance no longer exceeds the dominance threshold. Dominance threshold values are always between 0.5 and 1.0 exclusive. We have experimented with alternative criteria for pausing. Most of these are related to the performance

of the very best individual in each population[1]. None of the alternative approaches have so far resulted in equally good results as when using the described criterion. A nice property of the described criterion is that both populations will never be paused simultaneously. This is not guaranteed with several of the alternatives we have explored.

One may question why this pausing approach is helpful. After all, once a population is dominant it should have no means of evolving further as improvements will not be noticeable in the fitness which is already at a maximum. However, while the dominating population does not evolve much new functionality it will still be able to quickly adjust to any progress made by the dominated population thus obscuring the progress and preventing the individuals which have progressed from spreading within the dominated population.

The test performed in section 4.3 has parameters identical to the comparison tests performed in section 5.2 and will serve as a representative test utilizing our Interleaved Approach.

5.1.2 The Phantom Parasite

To avoid disengagement Rosin [45] proposes the adding of a so-called "phantom parasite" to each Teacher Set of the competing populations. When disengagement begins to occur and one of the species is dominating the other, the purpose of the phantom parasites is to ensure that some non-optimal individuals survive in the dominating species to act as "pedagogical stepping stones" for the dominated species. This should in turn allow the dominated species to continue evolving in a direction which will allow them to beat first the pedagogical stepping stone teachers and later the dominating "optimal" teachers. The phantom parasites encompass no solution themselves (i.e. GP trees or GA binary strings) and no actual competitions are run between them and the opposing students. Instead they are used in conjunction with competitive fitness sharing to allow the survival of the pedagogical stepping stone teachers described above. Each generation, after computing competition outcomes against current teachers, the outcome against a phantom parasite is defined in the following way:

- Students which lose to some current teacher defeat the phantom parasite.
- Students which defeat all current teachers lose to the phantom parasite.

Used in conjunction with Competitive Fitness Sharing and possibly Shared Sampling for the next generation Teacher Set this should ensure the survival of some non-optimal teachers to be used to guide the evolution of the dominated population. Note that this approach automatically assumes that the imperfect stepping stone teachers will in fact guide the evolution towards eventual defeat of the "optimal" teachers. In our opinion this is a quite unsubstantiated assumption as there is no guarantee that the "optimal" and non-optimal teachers require similar strategies

[1] Disengagement is often initiated by having one or a few superior individuals spread through the population.

to defeat. Especially when diversity maintenance or niching schemes are being utilized many local and widely different optima may be investigated simultaneously. In a worst case scenario the non-optimal teachers could in fact drive the evolution away from a solution able to beat the "optimal" teachers.

5.1.3 Moderating Opponent Virulence

Inspired by the differences in virulence among natural parasite strains Cartlidge [5] proposes to moderate the virulence of Teacher individuals[2] in order to prevent disengagement. The virulence is adjusted as depicted in figure 5.1. Effectively this means that moderately virulent students (which in the subsequent generation will function as teachers for the opposing population) which beat some specified percentage below 100 of their teachers (Cartlidge uses 75%) are awarded maximum fitness whereas maximally virulent students beating 100% of their teachers receive less. Cartlidge uses the counting ones problem and the problem of constructing minimum length sorting networks to demonstrate that moderating virulence can prevent disengagement.

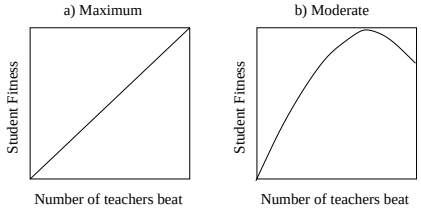


Figure 5.1: Fitness functions encouraging respectively maximal and moderate virulence.

This approach is very similar to Rosin's phantom parasite approach as both approaches punish individuals for being too good and actually guide the evolution towards the suboptimal moderately virulent solutions. We generally dislike the notion of discarding "too good" solutions and prefer our own pausing approach in which the good solutions are kept but not evolved further until the competing population has been allowed to catch up.

5.2 A Comparison on Disengagement Countermeasures

This section describes tests using the different countermeasures against disengagement which have just been described. As stated the test in section 4.3 will be used as documentation of our own Interleaved Approach and no new test will be performed using this approach.

[2]Referred to as parasites in the article.

5.2.1 Verifying Existence of Disengagement

The following test is performed to verify that disengagement occurs in the extended pursuit environment when no countermeasures are taken to prevent it. Test settings for this test are identical to the those specified in table 4.1[3] except that no pausing is performed.

Both populations are co-evolved for 550 generations. The predator HoF improvement graph from this test can be seen in figure 5.2. It is evident from viewing this graph that the predators are not able to evolve sufficiently advanced strategies which can capture a significantly large amount of the prey HoF members. The small incline during the first 200 generations indicates that the predators are slowly improving. This improvement stops at approximately generation 280 where the prey population has evolved strategies which very few of the predators can counter. Predators inserted after this point can not capture as many of the HoF prey as some of the previous HoF predators. This may well be an indication of genetic drift. When examining the corresponding improvement graph for the prey population it is clear that the prey population not at any point has problems with the evolved predator strategies. The prey inserted into the HoF during the beginning of the run very quickly learn to evade all of the Teacher Set predators and the graph shows no signs of changing. The graph has been omitted for the sake of brevity.

The graph in figure 5.2 is particularly interesting when it is compared to the corresponding graph in figure 4.4 section 4.3. The only difference between these two tests is that the Interleaved Approach (i.e. pausing) is employed in the test from section 4.3. The trend lines look almost the same until generation 300 where the test using pausing starts to improve due to the paused prey population while the non pausing test continues without change. The effect of pausing versus not pausing is very clear when comparing the two graphs. The fact that the prey population is paused while the predators start making rapid progress in approximately generation 325 in figure 4.4 indicates that the pausing is helping the predators to catch up with the superior prey population.

A comparison of the best individual evolved in the test described in this section and section 4.3 is shown in table 5.1. Both individuals from section 4.3 - the Interleaved individuals - outperform those evolved without pausing in the test in this section (i.e. regular).

	Interleaved Prey	**Regular Prey**
Interleaved Predator	65%	90%
Regular Predator	9%	46%

Table 5.1: Certainty in percent with which predator beats prey. The Interleaved Predator is earlier referred to as the CONR Predator.

Based on the results attained in this section we conclude that the Interleaved Approach produces more advanced strategies than when pausing is not employed in the extended pursuit environment. This again shows that the disengagement phenomenon is present in the extended pursuit environment. It will be interesting to

[3] I.e. the test with the modified CellOfNearestRight() function.

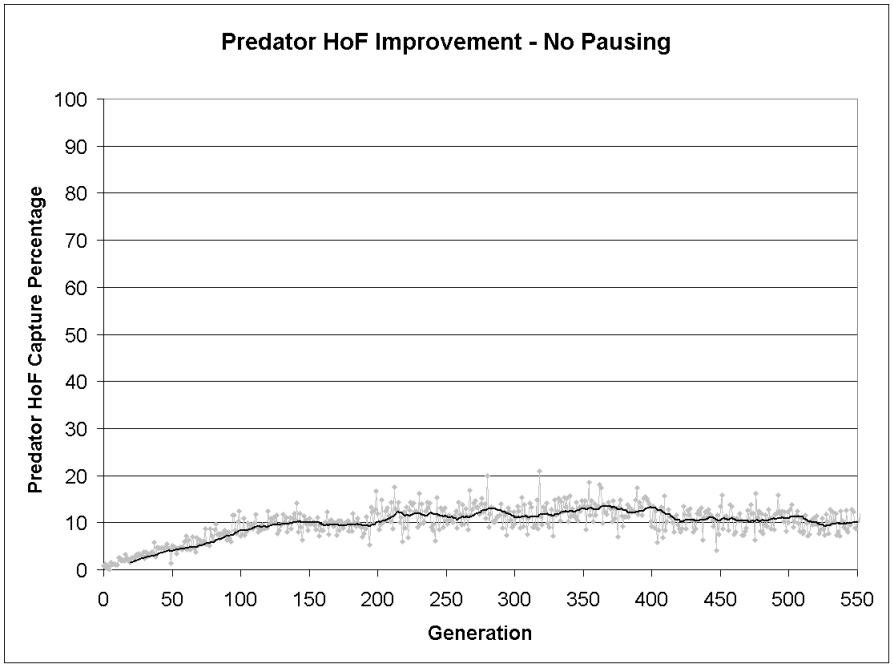


Figure 5.2: Capture percentage of predators in the HoF. The predators and prey are co-evolved without disengagement countermeasures.

see if and how the disengagement countermeasures found in literature will affect the co-evolution in our environment.

5.2.2 Test With the Phantom Parasite

The phantom parasite idea is very simple and nicely described in Rosin's article. As such it is also very easily implemented in our co-evolutionary system. We have performed a test in which it is used as countermeasure against disengagement. As with the test without disengagement countermeasures the parameters and function/terminal sets are identical to those used in the test in section 4.3.

The HoF predator improvement graph from this test can be seen in figure 5.3. Compared with the graph in figure 5.2 showing the test without disengagement countermeasures this graph looks much better. Although progress is for the most part slow it is fairly steady. Within the last 50 generations of the graph there seems to be signs of a small decline in performance. It is difficult to say if this is a beginning case of genetic drift as we have no data beyond generation 550. We choose to terminate the evolution at the same place as in the tests in sections 4.3 and 5.2 to allow for a fair comparison. The best predator and prey from this run are definitely better than those evolved without disengagement countermeasures. We have compared them with the best individuals from the other two tests as well as the test using moderated virulence. This comparison and a discussion of it can be found in section 5.2.4.

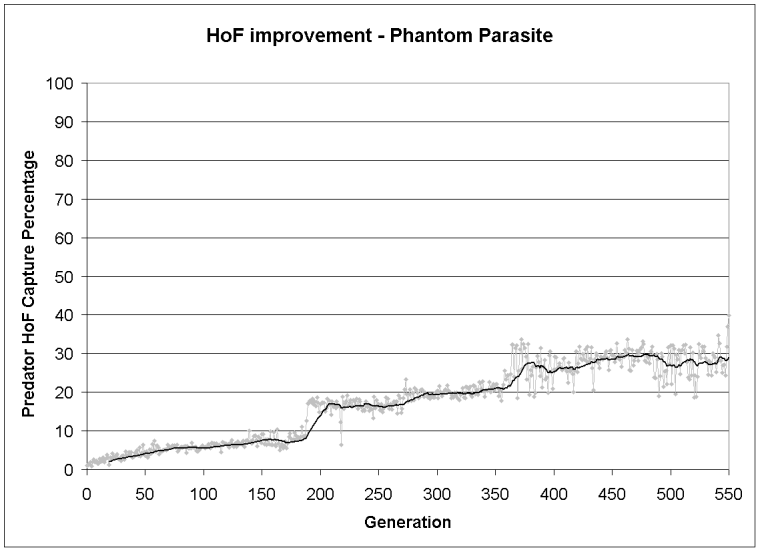


Figure 5.3: Performance improvement of predator HoF members. This co-evolutionary run uses the Phantom Parasite approach.

5.2.3 Test With the Moderate Virulence

Cartlidge [5] offers no exact specification or generally applicable description of how the moderate virulence approach should be implemented. Therefore the manner in which the moderate virulence approach is implemented in the extended pursuit environment is described. We invite the reader to examine the work of Cartlidge [5] and judge whether our implementation agrees with the description to a satisfactory degree.

Our implementation
In each generation an individual is evaluated against a constant number of opponent teachers. The fitness assigned to the individual depends on how many teachers the individual beats and on the virulence factor. Thus the Competitive Fitness Sharing is not applied together with the moderate virulence approach[4]. The virulence factor of an individual is a function of the number of teachers the individual can beat. It is calculated using the expression in equation 5.1 and the mapping between the "beat" percentage of an individual and the virulence factor is depicted in figure 5.4. The fitness assigned to an individual is multiplied by its virulence factor. The virulence factor is calculated in such a manner that individuals beating 75% of their opponents are assigned maximum fitness.

$$VirulenceFactor = \frac{-4}{225} \cdot x^2 + \frac{8}{3} \cdot x, x\epsilon[0; 100] \tag{5.1}$$

The x in equation 5.1 is the beat percentage of the given individual and ranges from 0 to 100.

[4]Combining Competitive Fitness Sharing with Cartlidge's approach is non-trivial, and we wish to employ an implementation as close to his approach as possible.

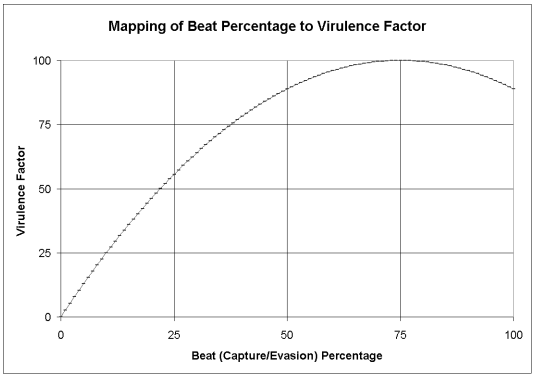


Figure 5.4: The graph shows the mapping of beat percentage values to their corresponding virulence factor values.

As with the other tests the parameters and function sets for this test are identical with those used in section 4.3. The improvement graph for the HoF predators from this test can be seen in figure 5.5. It is very unlike previous graphs of the same type and show large and rapid increases and decreases. The very rapid increase in performance in the beginning of the run is likely due to the decreased virulence of the otherwise taxing initial prey population. Due to the virulence factor the selected prey from early generations are very poor since even mediocre prey have a good chance of evading all early predators. The initial predator improvement changes at approximately generation 75 and the predators begin degrading. Examining statistics we can see that the predators selected for the HoF in generation 71 to 116 capture below 75% of the prey teachers. This means that no predators are present in the population at this time capable of beating the virulently optimal 75% of prey teachers. During this period degrading occurs. We thus believe that in this period disengagement occurs and the predator population is drifting.

After generation 120 we experience slightly more regular improvement for the predators. In general the predators reach very high capture percentage and far higher than previously attained in other tests. However, this positive statistical indication is not reflected in the evolved predator strategies which are generally poor when compared with the other approaches.

The average improvement graph for both the prey and predator HoF is depicted in figure 5.6. The prey improvement graph is like its counterpart from this test very uncommon and therefore included. Normally the prey improvement graph will increase very rapidly to around or above 90% and will remain there for the duration of the run. As can be seen in figure 5.6 this is not the case in this test. The prey are learning at a much slower rate, especially late in the run, and occasionally even degenerate. This occurs around generation 150, 210, 275 and 390 and is believed to happen for the same reason as for the predators.

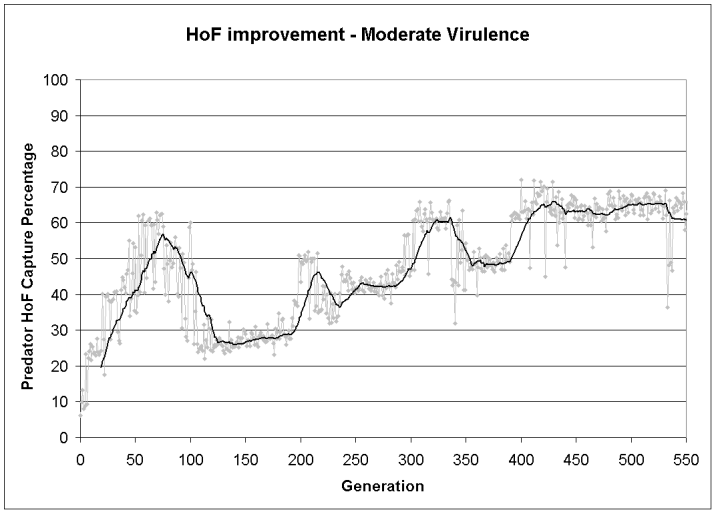


Figure 5.5: Performance improvement of predator HoF members. This co-evolutionary run uses the Moderate Virulence approach.

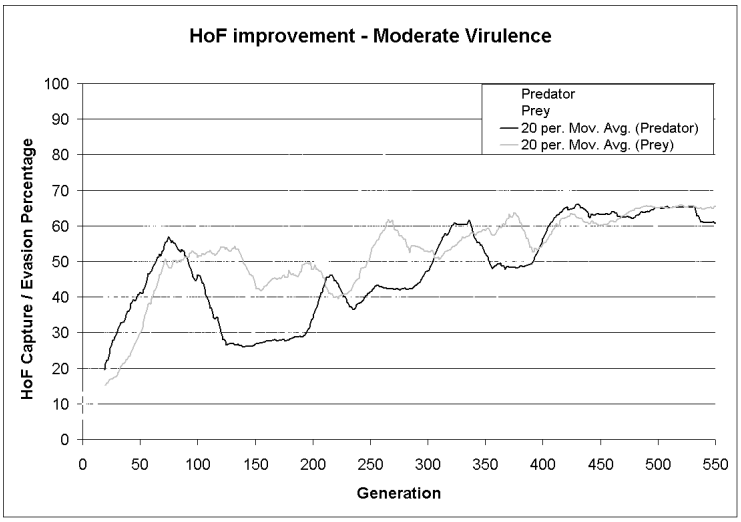


Figure 5.6: Average performance improvement of both prey and predator HoF members. This co-evolutionary run uses the Moderate Virulence approach. The average is found using the moving average method with a span of 20 values.

5.2.4 Discussion of Tests

For each of the four tests performed we have chosen the best predator and prey. To compare the quality of the evolved solutions we have matched each possible predator/prey pair against each other. The results of these tests are shown in table 5.2. Several interesting facts have become evident during these matches. Strategies

	Regular Prey	Interleaved Prey	Phantom Prey	Virulence Prey
Regular Predator	46%	9%	38%	1%
Interleaved Predator	90%	65%	61%	37%
Phantom Predator	79%	27%	22%	12%
Virulence Predator	4%	0%	0%	76%

Table 5.2: Certainty in percent with which predator beats prey. The Interleaved Predator is earlier referred to as CONR Predator. The Phantom prey and predator individuals are evolved with the Phantom Parasite approach and the Virulence prey and predator individuals are evolved with Moderate Virulence approach.

evolved using the regular approach - i.e. using no disengagement countermeasures - perform poorly and fail to evolve very interesting behavior. As already noted this is due to the presence of disengagement leading to evolutionary drift for the predator population and the lack of a proper "arms race" between the competing populations which in turn renders both populations without means to evolve.

Employing the Phantom Parasite approach enables an evolution flow with more improvement possible for both populations. The evolved strategies are much better than those evolved using no countermeasures. The predators make use of their ability to accelerate albeit to a lesser degree than those evolved using our Interleaved approach. Sometimes the predators will spread out and approach the prey from two sides. In all honesty the capabilities of the Phantom Parasite approach are a surprise to us. As discussed in section 5.1.3 we generally dislike approaches which may explicitly guide the evolutionary search towards sub-optimality. However, the Phantom Parasite approach has proved to us that locally suboptimal decisions may lead to a high degree of global optimality. Combining this with the fact that our problem is widely different from the simple Tick Tack Toe and Nim problems it is tested on in Rosin [45] we believe it to be an approach with general applicability.

The Moderate Virulence approach produces an evolution flow much different from what we have previously seen and the results are equally arcane. The moderately virulent best predator is extremely specialized towards the best moderately virulent prey and catches it with higher success than any of our other strategies. Unfortunately it is overfitted to a point where it is incapable of capturing any of the other prey strategies. Even the poorly evolved regular prey escapes this predator with 96% certainty. Interestingly the moderately virulent prey is very capable against anything but its predator counterpart and has very nice statistics against our other predators. Examining traces reveals that the moderately virulent prey always begins its matches by running north towards the northern wall. Many of our other evolved burrowing prey learn that the areas close to walls should be avoided and like to stay close to the center of the grid world. This in fact makes it easier for the predators to approach these prey from more than one side but the prey have the advantage of being in the open with no wall to hinder their escape. Other predator strategies than the virulence predator seem to be adapted to prey strategies which avoid walls and have to some extent forgotten how to beat prey which have not learned to respect the danger of walls. If more occurrences of prey strategies similar to the moderately virulent prey had been present in the late generation prey Teacher Sets of the other tests we firmly believe the other predator strategies would

have easily re-adapted to this kind of strategy too.

A potential problem with the Moderate Virulence approach in the extended pursuit environment regards the prey population which during the initial generations of the run are dominating the predator population. A very high percentage of the prey easily manage to evade almost all of the predators and only the very poorest strategies are caught. Thus in early generations these very poor prey strategies are most likely to score the best fitness values since they do not evade close to 100% of the opposing predators and are the closest to being "moderately" virulent. This is in our opinion not a desirable property of the approach. It may be the reason that the prey evolution is driven in a direction making the final prey strategies - while not very sophisticated - widely different from the other prey strategies. Note also that this problem is likely to persist in all environments, with similar degrees of disengagement as ours[5], if the Moderate Virulence approach is used on them.

The strategies evolved using our own Interleaved Approach perform very nicely when matched against the other strategies. Especially the Interleaved predator shows great prowess and is the predator pack with highest success rate against three of the four prey strategies. This shows the high degree of generality incorporated in the predator strategy evolved using the Interleaved Approach. Only the virulent prey strategy is beaten more efficiently by its own co-evolved predator pack. In conclusion we are very satisfied with the performance of our Interleaved Approach. We have shown the strategies evolved using it to be vastly superior to the regular approach with no disengagement countermeasures and also in this setting better than the Phantom Parasite and Moderate Virulence approach.

5.3 A Comparison on Evolutionary Paradigms

In this section single evolution is described and modeled within the extended pursuit environment. In single evolution only one population is evolved against a static teacher set. Following, tests are performed to compare the performance of single evolution and the quality of its solutions with that of the Interleaved Approach described in section 5.1.1 and the regular co-evolutionary approach. The interleaved and regular tests with which we compare the stepwise test results can be found in section 4.3.2 and section 5.2.1 respectively. The motivation for performing these tests is to examine the difference between these three approaches and determine if single evolution can produce results of quality similar to those evolved using the Interleaved Approach. Furthermore, in much of the literature cited in this paper it is generally assumed or implied that co-evolution is more efficient than single evolution and that co-evolution can even produce superior solutions which can not be produced through single evolution. No tests to support these assumptions are provided in any of the cited literature or - to our knowledge - elsewhere. A description of single evolution and how it is implemented in the extended pursuit environment can be found in section 5.3.1. In section 5.3.2 we present the test and the comparison of the test results from the three different approaches.

[5] E.g. environments in which one population from the outset of evolution has a great advantage over the other population.

5.3.1 Single Evolution in the Extended Pursuit Environment

In single evolution students are evolved against a static teacher set. In the extended pursuit environment where both populations are co-evolved, modeling an approach similar to the single evolutionary approach is done in the following manner. Each population takes turns evolving while the other population is paused. When the evolving population reaches a specific level of dominance the evolution is stopped and the other population starts to evolve. This means that a population is always evolved against a static opponent and two populations are never evolved at the same time as is the case with the other evolutionary approaches. This means that in the extended pursuit environment the predators are evolved against a static teacher set of prey, and as opposed to regular co-evolution the prey will not be able to adapt while the predators are evolving. This modified approach is named Stepwise Evolution to reflect that it is actually a series of single evolutions performed in steps. The dynamics of this approach are different from the two other approaches and it will be interesting to see the results and the effect of this evolution type. Figure 5.7 illustrates the three different paradigms with regards to how populations are evolved and when they may be paused. The Interleaved Approach can be seen as a combination of the two other approaches.

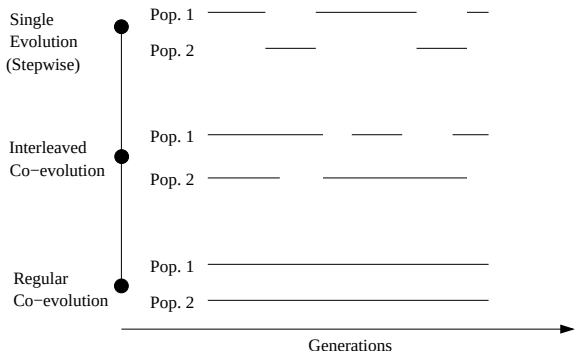


Figure 5.7: The figure illustrates the relation among the three different evolutionary paradigms wrt. the pausing of the two populations. The horizontal line represents a time line measured in generations. A solid line indicates that the population is being evolved and a break in the line means that the population is paused. The Single Evolution approach is also referred to as Stepwise Evolution in this report.

5.3.2 Stepwise Co-evolution Test

A test is performed with the same parameters and function sets as those used in section 4.3 and with the following modifications. The prey population is initially paused and when they do become unpaused the predator population is paused. The prey population is unpaused when the predator population dominates more than 20% which causes the predator population to be paused. The prey population

is paused again when it dominates more than 80%[6] and the cycle continues. The Stepwise Evolution test is executed for 1700 generations which is approximately twice the number of generations that the populations in the interleaved and regular co-evolutionary tests were evolved for. Comparisons are performed with results gathered after 850 and 1700 generations. This is because the interleaved test was run for 550 generations. Out of these 550 generations the predator population was evolved for 550 generations and the prey population for 282 summing to 832 generations or approximately 850 generations. We suspect stepwise evolution to be slow and the extended 1700 generation test is performed to determine if stepwise evolution can produce better solutions than the other approaches when alloted twice the number of generations.

When examining the HoF improvement graphs shown in figure 5.8 from the stepwise test it appears that the predator improvement is indeed slower than the Interleaved Approach but slightly faster than the regular co-evolution. The figure depicts the HoF improvement graphs for both the prey (upper) and the predator (lower) population. After generation 800 the predators have barely reached 20% captures. This number can be compared to the corresponding number from generation 550 from the Interleaved Approach. The predator HoF improvement graphs from the interleaved test and the regular test can be seen in figure 4.4 page 57 and figure 5.2 page 70 respectively. The predators from the Interleaved Approach had reached 48% captures in average which is considerably higher. It is also interesting to see that the prey improvement does increase almost at same pace as the other approaches but that it does not remain constantly above 90% as often seen in other tests. This could indicate either that the predators have evolved difficult solutions or that the evolution flow is slower or not as consistent as in the other tests. Since the predators have not evolved particularly advanced strategies as it is explained in the following paragraph, we believe it to be because the evolution flow becomes more erratic when a population evolves against a fixed teacher set for an extended duration after which the teacher set suddenly changes. Allowing the stepwise evolution to continue for twice as long as the two other approaches changes little with respect to the improvement graphs. A small increase in improvement can be seen for the predators while the prey are mostly paused. The solutions produced after extending the test are slightly better than those from the middle of the run. Towards the end of the test predators inserted into the HoF have captured 40 out 100 teacher prey in average while the prey inserted into the HoF manages to evade all 100 teacher predators. This testaments that the best prey have no problems finding solutions which can evade all predators which in turn indicates less advanced predators.

To examine the quality of the solutions produced from the Stepwise Evolution the best solutions from the two HoFs are matched against those from the Interleaved and the Regular approach. The results from these matches can be seen in table 5.3 which presents the percentage with which the given predator captured the prey based on 100 matches. Two predators and two prey are selected from the Stepwise test. Those which are labeled Stepwise are the HoF individuals which correspond - with regards to evolution time - to the individuals from the other approaches.

[6] In tests with a 50% threshold the predators were never paused. Several values are tested and the chosen values result in the most satisfactory evolution flow. These threshold values are also the same as those in the Interleaved test in section 4.3.

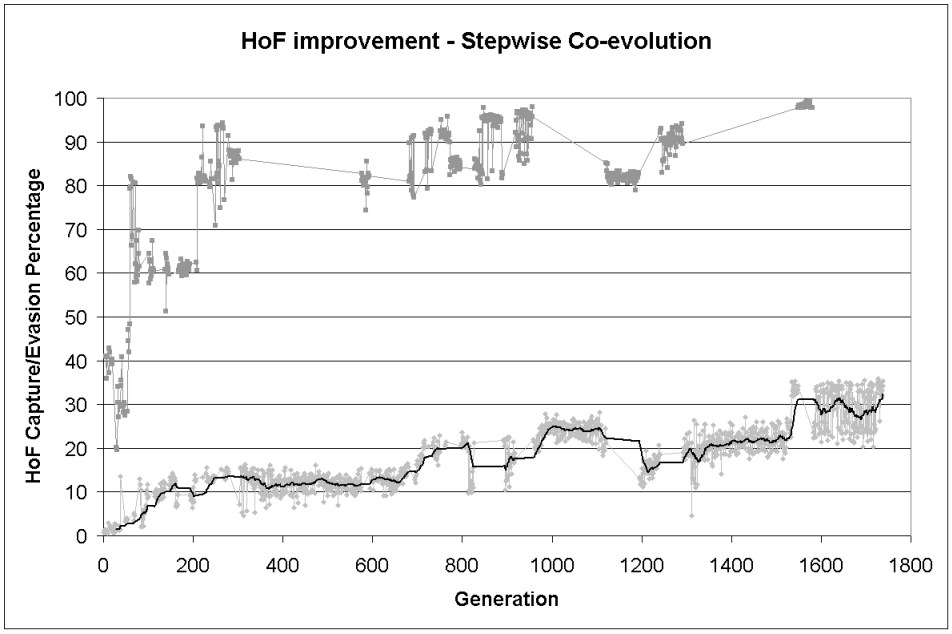


Figure 5.8: The HoF improvement graphs from the Stepwise Co-evolution test. The top line is the improvement graph for the HoF prey and the bottom line which is shown with a trend line is that of the HoF predators.

The Stepwise predator and prey are from generation 832 to match the number of generations from the interleaved run. Out of these 832 generations the predators were evolved for 573 generations and the prey for 260 generations. The individuals which are labeled Extended Stepwise are evolved for twice as long as those in the interleaved test.

	Regular Prey	Interleaved Prey	Stepwise Prey	Extended Stepwise Prey
Regular Predator	46%	9%	32%	5%
Interleaved Predator	90%	65%	60%	44%
Stepwise Predator	39%	0%	52%	0%
Extended Stepwise Predator	51%	0%	3%	28%

Table 5.3: Certainty in percentage with which the predators beat the prey.

As can be seen from table 5.3 all of the predators are fairly good at capturing the prey evolved with the regular approach. Notice that allowing the stepwise approach to evolve for twice as long does only produce a predator which is able to increase the capture percentage of the regular prey by 12%, from 39% to the 51% which is still far from the predator produced with the Interleaved approach. None of the stepwise predators are able to capture the interleaved prey but they perform somewhat better against their own prey from their respective generations. This indicates that the stepwise predators are somewhat over-fitted towards the prey

strategies they are evolved against. The extended stepwise prey strategy is able to evade the interleaved predator more than the interleaved prey which indicates a fairly advanced strategy. It should be kept in mind though that this prey is the result of a test twice as long as the interleaved test. If the interleaved predators were allowed to evolve for the same number of generations we are confident that a solution would be evolved which could produce a higher capture percentage. The stepwise predators have learned to accelerate and use this feature to capture prey but they do not use the CellOfNearestRight() function in the same manner as the interleaved predator.

One motivation for the comparison is to determine if there is truth to the claim that co-evolution is superior to and more efficient than single evolution. Our tests show that the regular approach is able to produce solutions which are slightly better than those produced with the stepwise approach in the same number of generations. Furthermore, it seems that the predators from the regular approach exhibit more generality than the stepwise predators because they are able to capture prey from all of the other approaches. The regular co-evolutionary approach has not proved to be significantly superior to the stepwise approach in our tests. However, keep in mind that the test using regular co-evolution experienced the phenomenon of disengagement which affects the evolution in a negative manner. Provided that disengagement was not present in this environment we believe that the regular approach would produce better and more general solutions.

The Interleaved Approach produces overall better solutions in the same number of generations as the regular co-evolutionary approach and the stepwise evolutionary approach. Additionally it outperforms the stepwise evolutionary approach even when the stepwise test is run for twice the number of generations. The Interleaved Approach can therefore be seen as a more efficient approach in this environment.

5.4 Interleaved Co-evolution and the Counting Ones Problem

In the preceding sections we have shown the Interleaved Co-evolutionary Approach to outperform the Phantom Parasite and Moderate Virulence Approach when employed in the extended pursuit environment as means to avoid disengagement. We have also compared the Interleaved Approach with other co-evolutionary paradigms - i.e. Regular and Stepwise co-evolution - and shown that the Interleaved Approach enables the evolution of better solutions in the extended pursuit environment. We have thus gathered extensive results testamenting to the usefulness of the Interleaved Approach but all results have been gathered through co-evolution in the extended pursuit environment. In this section we will investigate the generality of the Interleaved Approach by applying it to a co-evolutionary problem which is widely different from the extended pursuit environment. By choosing a problem different from the extended pursuit environment we hope to show that the Interleaved Approach has general applicability. The chosen problem is known as the *Counting Ones* problem and is used by Cartlidge [5] as benchmarking problem for the by now familiar Moderate Virulence Approach. The Counting Ones problem differs from the extended pursuit environment in several important aspects. Most

importantly no noise is present in the Counting Ones problem. Furthermore the use of Competitive Fitness Sharing is not necessary due to the simple betterThan relation inherent in the problem. For any N individuals in the Counting Ones problem a linear ordering of the individuals with respect to performance can always be produced. Thus no intransitive superiority relations exist in this problem which in turn eliminates the need for a Hall of Fame. An attractive property of the Counting Ones problem is the fact that the level of disengagement between the competing populations is easily adjustable from a level where no disengagement is present to levels of extreme disengagement.

Basically the Counting Ones problem contains two co-evolving populations each containing individuals made up by bit strings of length 100. The individuals compete against individuals from the opposing population. The winning individual in a competition is the individual with most high bits in its bit string (i.e. bits with value one). Disengagement is introduced by allowing the individuals from one of the populations to have multiple bit strings which are combined using bitwise OR. An individual with only one bit string is known as a *haploid* whereas individuals with two bit string are known as *diploids* and individuals with three bit strings *triploids*. A haploid will on average possess high bits in 50% of its loci while diploids and triploids will possess high bits in 75% and 87.5% of their loci respectively. Thus diploids and triploids will hold an inherent advantage over their haploid opponents. When two individuals are pitted against each other the winner will receive fitness. The fitness awarded for winning is constant. In case of a tie each individual is awarded fitness corresponding to half a win. At the beginning of the co-evolution all individuals are created with all bits low (i.e. zero).

The same basic parameters as those employed by Cartlidge [5] have been employed in our tests. They are summarized in table 5.4. As evident from the table breeding is exclusively performed using an a-sexual bit flipping mutation which with a 5% probability swaps the value of the bit it is applied to. Note that while Cartlidge employed GAs in his work we are still using STGP. GAs are the natural choice for a problem involving bit strings but the fact that we have a working and tested co-evolutionary STGP implementation containing an implementation of the Interleaved Approach means that we can save time by implementing Counting Ones as an STGP problem. The resulting individuals each contain a tree with a root function node and 10 intermediate function nodes each with 10 children. Each child contains one bit in the case of a haploid individual or two or three bits, as well as functionality for performing the bitwise OR, in the case of diploid or triploid individuals. With this setup the STGP constraints guarantee that each individual has the correct number of loci.

5.4.1 Haploid-Diploid Tests

The following tests pit a haploid population against a diploid population. As documentation of test results we will plot the absolute (objective) fitness of all individuals from both populations onto a graph in a manner similar to what was done in Cartlidge [5]. The objective fitness of an individual is the percentage of its loci containing high bits. As mentioned earlier the diploid population has a higher chance of evolving good solutions since they have higher chance of having high bits in

Parameter	*Value*
Population Size	25
Bit String Length (# of loci)	100
Objective	Majority function on number of high bits
Generations	**600**
Selection Method	Tournament Selection
Tournament Size	5
Breeding Method	Bit flipping
Bit Flip Probability	5%
Dominance Threshold	60% for diploid test and 55% for triploid test

Table 5.4: Test settings for Counting Ones tests.

many of their loci. Thus we expect disengagement to occur if countermeasures are not taken. A test is performed to verify this assumption. The upper graph in figure 5.9 shows the results of this test. When no countermeasures are taken the diploid population quickly dominates the haploids. As soon as disengagement occurs selection pressure is removed. All haploids score fitness zero and all diploids score maximal fitness. Having lost their fitness gradient the populations are pushed towards their expected average fitnesses which are 50% for the haploids and 75% for the diploids. Random fluctuations cause a brief re-engagement around generation 350. Shortly hereafter the populations disengage again and remain disengaged for the remainder of the run.

The lower graph in figure 5.9 shows that introducing the Interleaved Approach with a dominance threshold of 60% alleviates the disengagement problem. Figure 5.9 can be directly compared with figure 3 page 3 in Cartlidge [5]. Due to the consistent engagement both populations evolve better solutions than in the test with no countermeasures. The blocky appearance of especially the diploid population graph is due to the fact that the diploids are paused for extended periods of time. In generations 426-465 the haploid population has evolved sufficiently fit strategies to allow it to be paused continuosly for several generations. This can be more easily seen on the pausing graph below. The pausing of the haploids shows that the extended pausing of the diploid population has benefitted the haploids to an extend which allows them to briefly become dominating in an environment where they are otherwise clearly outmatched by design. The fact that none of the populations evolve individuals with an objective fitness of 100 can be attributed to the 5% mutation bias. As individuals get better it gets progressively more difficult to increase in performance due to the fact that most mutations will flip high bits to low bits.

Note that the dominance threshold for this test is reported as 60% in table 5.4. This is a low value which is necessary as it is imperative to quickly pause the strong diploid population when it slightly outperforms the haploid population. Also it is very important to keep the diploids paused until the haploids have comfortably caught up with them. We initially performed a test with a dominance threshold of 75%. The results from this test can be seen in figure 5.10. In the beginning of the run the co-evolution is proceeding nicely but at approximately generation 125 the diploid population is unpaused prematurely. This combined with a series of unfortunate mutations in the haploid population leads to disengagement near generation 200.

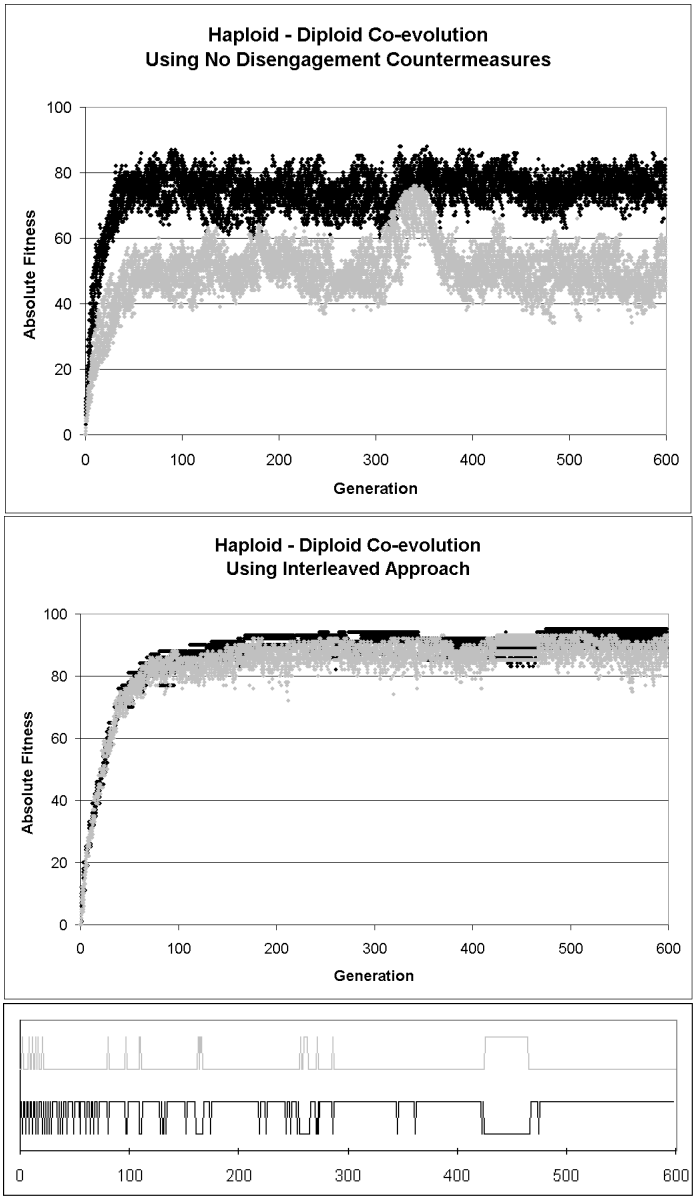


Figure 5.9: Haploid - Diploid tests. Lower test uses the Interleaved Approach with a dominance threshold of 60%. Haploids are grey and diploids black. The small graph in the bottom shows when the two populations are paused. When the grey line peaks the haploid population is paused. When the black line peaks the diploid population is paused.

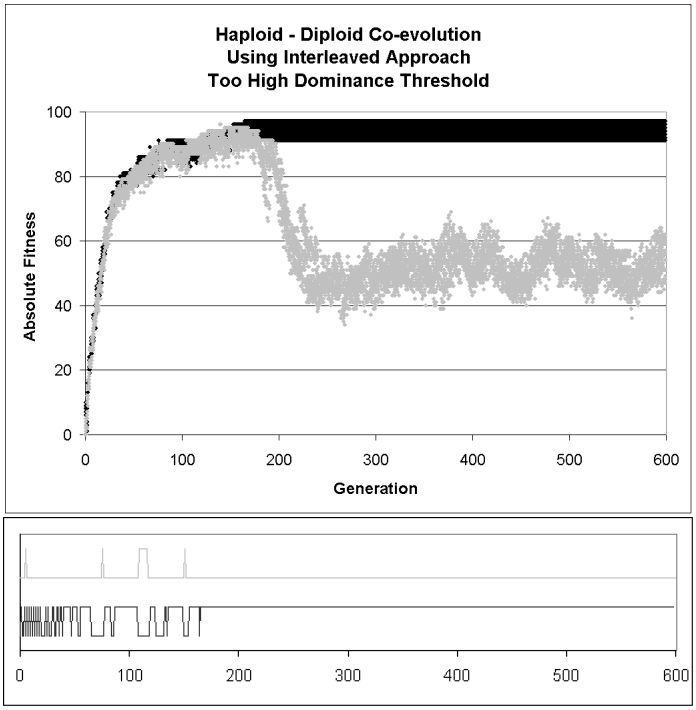


Figure 5.10: Haploid - Diploid test using the Interleaved Approach with a too high dominance threshold of 75%. Haploids are grey and diploids black. The small graph in the bottom shows when the two populations are paused. When the grey line peaks the haploid population is paused. When the black line peaks the diploid population is paused.

5.4.2 Haploid-Triploid Tests

The following tests pit a haploid population against a triploid population. The added level of difficulty due to the greater asymmetry between the populations means that the dominance threshold is lowered to 55%. Test results without and with the Interleaved Approach can be seen in figure 5.11. Figure 5.11 can be directly compared with figure 4 page 4 in Cartlidge [5]. The upper graph showing results of the run using no disengagement countermeasures shows that disengagement occurs quickly and definitively after a few generations. The populations do not re-engage after the initial disengagement. The lower graph shows that even in this very asymmetric environment the Interleaved Approach enables engaged co-evolution and the haploid population evolves solutions even better than those in the Haploid-Diploid test.

The results in this section show that the Interleaved Approach can be used to avoid disengagement in the Counting Ones problem. We have now successfully applied the Interleaved Approach to two very different environments. Based on this fact we conclude that the Interleaved Approach has general applicability.

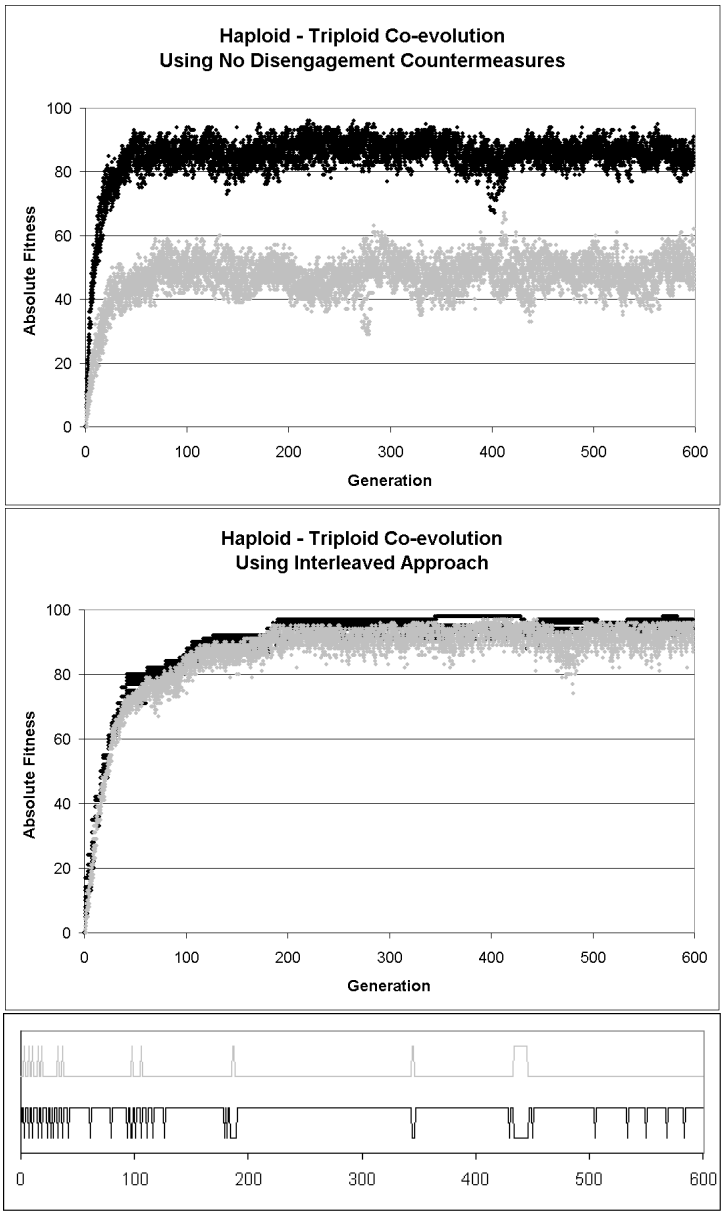


Figure 5.11: Haploid - Triploid tests. Lower test uses the Interleaved Approach with a dominance threshold of 55%. Haploids are grey and diploids black. The small graph in the bottom shows when the two populations are paused. When the grey line peaks the haploid population is paused. When the black line peaks the triploid population is paused.

Chapter 6

Conclusion

In this report we have used Strongly Typed Genetic Programming to evolve cooperative behavior of autonomous agents. Besides solving common problems such as lowering SPEA2 selection pressure, redesigning prey behavior for a greater predator challenge and reducing the effect of noise through the introduction of our Competitive Fitness Dampener we have attained four main achievements.

Experimental Achievements

- The evolution of an interesting strategy for homogeneous predator agents with which the predators utilize visual communication to coordinate the pursuit of taxing burrowing prey. The evolved predators are able to catch very capable prey agents with good success rates.
- The development of a diversity maintaining objective which successfully allows our co-evolutionary runs to continue without premature convergence.

Theoretical Achievements

- The devising and implementation of an objective measure capable of measuring progress or lack thereof during co-evolutionary runs.
- The development of a disengagement countermeasure shown to be more potent than two existing countermeasures when used in conjunction with the extended pursuit environment. Furthermore the testing of the countermeasure on a problem of completely different nature gives credence to our statement that the countermeasure has general applicability.

In the following we elaborate in more detail upon our main achievements in this project.

Evolution of Cooperation Among Predators

A powerful and advanced means of visual communication is introduced in section 4.3. Both predators and prey evolved when using this function are very capable. The predators master their abilities to accelerate and decelerate and through their

communication coordinate their attacks so the prey is approached by accelerating predators from at least two sides at once. The prey evolves into being very nimble and uses its high degree of maneuverability to sidestep approaching predators. The best evolved predator strategy is able to catch the best evolved prey with 65% certainty. Predators evolved with no communications skill cannot catch this prey at all and predators using simple visual communication catch it in only 1% of the matches. Introduction of aural (messaging) communication yields no improvement in the predator strategies. A discussion of why our predators do not benefit from the messaging functionality can be found in section 4.4.4.

Naturally we had hoped for interesting strategies using aural communication but we nevertheless feel that we have shown the benefits of cooperation through our visual communication. Based on our results we have shown that cooperating autonomous agents surely can produce better results than non-cooperating autonomous agents solving the same task. Naturally this fact will not be true if the task is of a type which inherently cannot benefit from teamwork.

Development of Diversity Measure

A heavily modified version of De Jong's [8] FOCUS algorithm is used as means of diversity maintenance. The main idea of the FOCUS algorithm is to introduce an objective which promotes diversity by rewarding individuals with high diversity. Our version uses a lower selection pressure than De Jong's, and instead of using a measure of diversity based on the structure of the GP trees our diversity measure is based on differences in the performance and size objective of individuals.

Results documented in section 3.15 show that using our version actively maintains the diversity of our populations and avoids any problems with premature convergence. Even in our longest co-evolutionary runs have we not experienced lack of diversity when using our diversity measure.

Development of Objective Measure for Monitoring Progress

Analyses of competitive co-evolutionary runs are often a demanding task due to the lack of an objective measure of the performance of evolved individuals. The method described in section 3.1.2 provides an objective measure of progress and flow of the competitive co-evolution and has been found valuable in the analysis of the results from our experiments. Information about progress and deterioration in the strategies of evolved individuals is updated continuously when applying the objective measure to a co-evolutionary run. Note that the measure can only be used to track progress or lack thereof within one co-evolutionary run. It cannot be used to compare results from different runs. The method is especially useful for lengthy runs which can be monitored and terminated if detrimental effects due to inaccurate parameters prevent progress.

Development and Investigation of a Disengagement Countermeasure

Our disengagement countermeasure - the Interleaved Approach - consistently helps to avoid disengagement when applied in the extended pursuit environment. See e.g. section 4.3 for results testamenting its effects. Comparison tests are performed with the only two disengagement countermeasures we can find in literature. These two countermeasures are the Moderate Virulence in Cartlidge [5] and Phantom Parasite in Rosin [45]. The results from these tests show that the Interleaved Approach

enables the evolution of the best and most general strategies of the three when utilized in the extended pursuit environment.

In section 5.3 the Interleaved Approach is compared with regular competitive co-evolution and a stepwise single evolution variant which evolves the competing populations in turns. While regular co-evolution slightly outperforms stepwise evolution both approaches are shown be inferior to the Interleaved Approach with respect to the quality of evolved predator and prey solutions.

To test the general applicability of the Interleaved Approach it is applied to the Counting Ones benchmark problem used to test the Moderate Virulence Approach in Cartlidge [5]. Test results show that the Interleaved Approach produces results of similar quality as the Moderate Virulence Approach in this environment. The fact that the Interleaved Approach performs so well in two very different co-evolutionary environments is indication that the Interleaved Approach has general applicability.

Appendix A

VisualGP

VisualGP is a small application which we have implemented to aid in visualizing predator and prey strategies. It takes as input trace files consisting of movement information from a match pitting a predator pack against a prey. The 300 or fewer turns are replayed continually one turn at a time. The user can adjust the playback speed, stop playback, pause playback or advance playback a single frame at a time. VisualGP has an interface similar to that of a VCR and has proven to be a valuable tool in examining agent behavior. Especially for cases in which the strategies of the agent are difficult to interpret. Figure A.1 shows a screenshot from VisualGP.

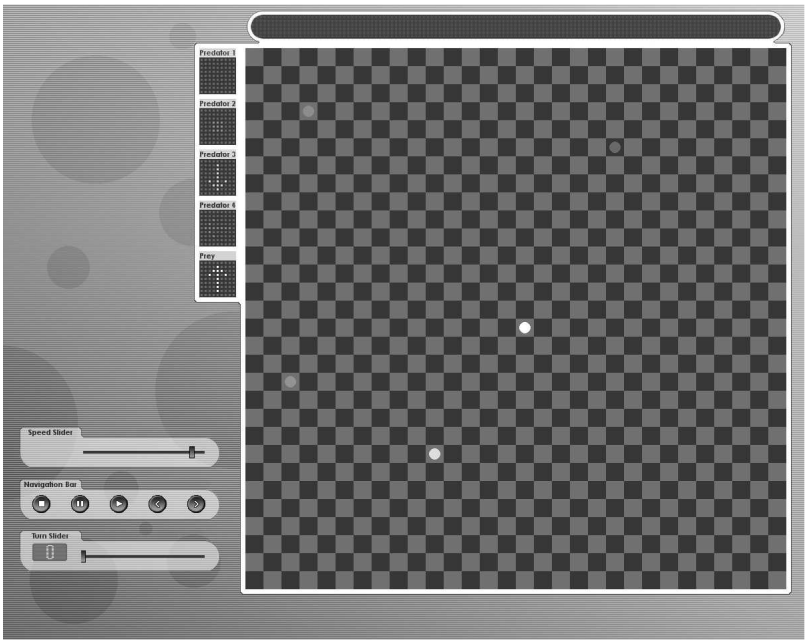


Figure A.1: A screenshot from VisualGP.

Appendix B

ECJ

The ECJ system by Sean Luke[1] is a Java-based evolutionary computation and genetic programming system. This chapter introduces the relevant parts of the ECJ system pertaining to the experiments in this report. Supported features and characteristics are listed and explained. Descriptions of the more special modules, e.g. used for Multi-Objective Evolutionary Algorithms and Co-Evolution are given subsequently. These modules are packages within the ECJ system and will thus be referred to as packages.

B.1 Contents and Structure of ECJ 9

The ECJ provides functionality and tools for a wide range of tasks within the field of EA and GP. A user is to implement only problem specific issues and various interfaces in order to utilize the system. Specifying which breeders, selection methods, evaluators, population size etc. are to be used is done in parameter files. This enables easy switches between different settings between evolutionary runs.

The topmost class in the evolutionary system is Evolve which handles checkpointing, random number generators, logging facilities and the parameter database. Evolve is the entry point and contains an EvolutionState object which holds the state of the evolutionary run at all times. Evolve contains the objects for initializing populations, evaluating individuals, breeding new individuals and so forth. These functionalities are embedded in the packages described in section B.2.

B.2 Supported Features

ECJ supports the following features:
Evolutionary Computation Features:

[1] http://www.cs.umd.edu/users/seanl/

- Genetic Algorithms/Programming style, Steady State and Generational evolution, with or without Elitism.
- Many selection operators. See section B.3.
- Multiple subpopulations and species. A population can consist of several subpopulations each of different species.
- Reading saved populations from files. Enables re-evaluation or further evolution of a population.
- Packages for STGP, co-evolution, and multi-objective optimization. See sections B.3 and B.3.

Genetic Programming Features:

- Supports Strongly Typed Genetic Programming. See section B.3.
- Ephemeral Random Constants. Defined in Koza I [30].
- Multiple tree forests. One individual can contain more than one tree of nodes.
- Six tree-creation algorithms. See section B.3.
- Extensive set of GP Breeding operators. See section B.3.

B.3 Packages

The Genetic Programming Package

The genetic programming package contains classes which implement tree-style genetic programming. The package supports STGP, ADF, ADM, ERC, and a large collection of Koza-I and Koza-II-style breeding and initialization methods.

The Select Package

The select package contains objects which implement selection methods. Most popular is the Tournament Selection and a special variant named the SPEA2TournamentSelection used in this particular multi-objective EA and which is also used in the extended environment. Other selection methods are Best Selection, Greedy Selection and Fitness Proportionate Selection. A feature made possible by the pipeline design is the Multi Selection method which enables the use of more than one selection method in a evolutionary run.

The Breed Package

The breed package holds objects for breeding new individuals using the pipeline design specified within ECJ. All standard Koza breeders are included plus several additional. A special SPEA2 breeder is applied in the extended environment.

The Build Package

The build package holds objects which perform tree creation. Apart from the standard Koza tree creation algorithms the package includes Probability Tree Creation 1 and 2, RandomBranch and Uniform[2]. The PTC1 and 2 allow tree creation with various selection probabilities for each function and terminal.

The Koza Package

This package holds objects which define some of the algorithms used in Koza I [30] and Koza II [31] for breeding and tree creation. The three pipelines used for breeding Crossover, Mutation and Reproduction are versions of the genetic programming pipelines which perform sub-tree crossover, single point mutation and reproduction respectively. The tree creation is performed by the node builders FullBuilder, GrowBuilder and HalfBuilder which use the Koza I algorithms Full, Grow and Ramped Half-and-Half respectively. Additionally support for Koza-style node selection and Koza fitness is included.

The Coevolve Package

The competitive co-evolution provides very limited support primarily in the form of Java interfaces for this type of task. Several evaluation topologies are provided but are intended for single species environments.

The MOEA Package

This package provides basic multi-objective evolution and support for SPEA2 in the form of special breeder, evaluator, multi-fitness and tournament selection method objects.

[2]See http://www.cs.umd.edu/projects/plus/ec/ecj/docs/overview.html for more information

Appendix C

Interleaved Approach Flow

Figure C.1 shows the evolution flow in the extended pursuit environment when the Interleaved Approach is applied.

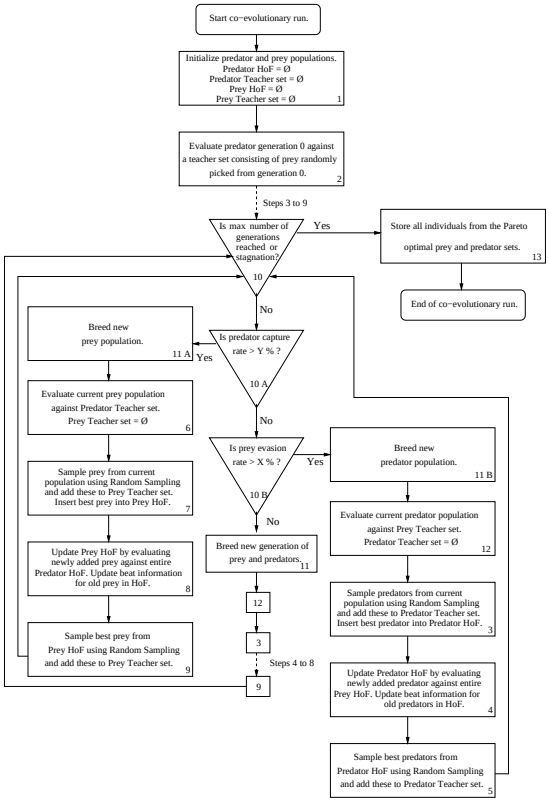


Figure C.1: The figure depicts the interleaved evolution flow as applied in the extended pursuit environment.

Appendix D

Functions and Terminals in the Extended Environment

This appendix serves as a reference of the predator and prey terminal and function sets.

D.1 Predator Functions and Terminals

The function set $\mathcal{F}$ for predator agents can be seen in table D.1.

Function Name	Argument(s)	Return Type	Description
IfThenElse()	Boolean, PredTack, PredTack	PredTack	Based on the Boolean argument the Then or the Else branch is executed.
LessThan()	Length, Length	Boolean	Compares two length values. Returns True if the first length argument is smaller than the second. Returns False otherwise.
MD()	Cell, Cell	Length	Calculates the Manhattan distance between the two argument cells.
CellOfSelf()	PredTack	Cell	Returns the position of the predator calling the function. If the PredTack argument is North the position just north of the Predator is returned.
MyLastCell()		Cell	Returns the cell where the calling predator was located in the previous turn.
CellOfNearest()	PredTack	Cell	Returns the cell of the nearest predator.
CellOfPrey()	PreyTack	Cell	Returns the position of the prey if it is visible. Otherwise returns a value which will make the calling function aware of the fact that a cell has been requested which cannot be revealed.

Table D.1: Predator functions for the extended pursuit environment.

The terminal set $\mathcal{T}$ for predator agents can be seen in table D.2.

Terminal Name	Possible Values
Boolean	True, False
Cell	(x,y) $\in$ integer [0;90]
Length	x $\in$ integer [0;30]
PredTack	North, South, East, West, Continue, Accelerate, Decelerate
PreyTack	North, South, East, West, Continue

Table D.2: Predator terminal nodes in the extended pursuit environment. The integer ranges pertaining to the Cell and Length data types are consequences of the grid size of 90 by 90 squares. Because of the way the function MD() calculates the distance between a prey and a predator the Length will never exceed 188. However, when length terminals are inserted in trees as constants they will not exceed 30. Note that the continue value corresponds to the stay value in the Simple Pursuit environment.

D.2 Prey Functions and Terminals

The function set $\mathcal{F}$ for prey agents can be seen in table D.3.

Function Name	Argument(s)	Return Type	Description
IfThenElse()	Boolean, PreyTack, PreyTack	PreyTack	Based on the Boolean argument the Then or the Else branch is executed.
LessThan()	Length, Length	Boolean	Compares two length values. Returns True if the first length argument is smaller than the second. Returns False otherwise.
MD()	Cell, Cell	Length	Calculates the Manhattan distance between the two argument cells.
CellOfSelf()	PreyTack	Cell	Returns the cell of the prey. If the PreyTack argument is North the position just north of the prey is returned.
CellOfNearest()	PredTack	Cell	Returns the cell of the nearest predator. The PredTack has the same effect as the PreyTack in the CellOfSelf function.
CellOfNearestWall()		Cell	Returns the cell of the nearest wall.

Table D.3: Prey functions for the extended pursuit environment.

The terminal set $\mathcal{T}$ for prey agents can be seen in table D.4.

Terminal name	Possible Values
Boolean	True, False
Cell	(x,y) $\in$ integer [0;90]
Length	x $\in$ integer [0;30]
PreyTack	North, South, East, West, Continue
PredTack	North, South, East, West, Accelerate, Decelerate, Continue

Table D.4: Prey terminals for the extended pursuit environment.

Appendix E

Modified Function and Terminal Set

The modified terminal set $\mathcal{T}$ for predator agents can be seen in table E.2. The various functions are introduced in different approaches described in the report.

E.1 Modified Predator Functions and Terminals

The modified function set $\mathcal{F}$ for predator agents can be seen in table E.1.

Function Name	Argument(s)	Return Type	Description
CellOfNearestRight()	PredTack	Cell	Returns the cell of the predator which is both to the right of a line drawn from the calling predator to the prey and closest to the prey.
IfThenElseShout()	Boolean. PredTack, PredTack, Msg	PredTack	Same functionality as the regular IfThenElse except that the argument msg is shouted if the boolean argument is true.
WasMsgShouted()	Msg	Boolean	Returns true if the argument msg was shouted.
LastPositionOfMsg()	Msg	Cell	Returns the cell of the predator which last shouted the argument msg.
Shout()	Msg		Shouts the argument msg.
IsLastMsgOfType()	Msg	Boolean	Returns true if the last msg shouted by any predator is the same as the argument msg.

Table E.1: Modified predator functions for the extended pursuit environment.

The modified terminal set $\mathcal{F}$ for predator agents can be seen in table E.2.

Terminal Name	Possible Values
Boolean	True, False
Cell	(x,y) $\in$ integer [0;90]
Length	x $\in$ integer [0;30]
PredTack	North, South, East, West, Continue, Accelerate, Decelerate
PreyTack	North, South, East, West, Continue
Msg	a $\in$ char [A;J]

Table E.2: Predator terminal nodes in the extended pursuit environment. The integer ranges pertaining to the Cell and Length data types are consequences of the grid size of 90 by 90 squares. Because of the way the function MD() calculates the distance between a prey and a predator the Length will never exceed 188. However, when length terminals are inserted in trees as constants they will not exceed 30. Note that the continue value corresponds to the stay value in the Simple Pursuit environment.

E.2 Modified Prey Functions

The modified function set $\mathcal{F}$ for the prey agent can be seen in table E.3. Most of the functions are repetitions from the function set of the predator. The prey has no need for the messaging functions and it has different functions used to locate predators. The CellOfNearestVerticalWall() and CellOfNearestHorizontalWall() functions make it possible for the prey to detect the walls of the grid world. If the prey cannot detect walls and is standing next to e.g. the southern wall while being approached by a predator from the north it will likely attempt to move south and be bounced back without realizing it. The cell returned from both these two functions is aligned with the prey and next the wall.

Function Name	Argument(s)	Return Type	Description
CellOfNearestVerticalWall()		Cell	Returns the cell of the nearest vertical wall.
CellOfNearestHorizontalWall()		Cell	Returns the cell of the nearest horizontal wall.
CellOf2ndNearest()	PredTack	Cell	Returns the cell of the second nearest predator. The PredTack has the same effect as the PreyTack in the CellOfSelf function.

Table E.3: Modified prey functions for the extended pursuit environment.

Appendix F

Evolved Strategies

F.1 Strategies Evolved Using CFD

The best predator evolved using the CFD has the following strategy which is of size 73.

```
(ITE (LessThan (Md (CellOfSelf West)
                   (CellOfPrey North))
               (Md (CellOfSelf North)
                   (CellOfPrey West)))
     (ITE (LessThan (Md (CellOfSelf Decelerate)
                        (CellOfPrey East))
                    (Md (MyLastCell (ITE (LessThan 25 (Md (45,64)
                                                          (CellOfPrey East)))
                                         East
                                         South))
                        (CellOfPrey North)))
          West
          (ITE (LessThan (Md (CellOfSelf West)
                             (CellOfPrey East))
                         (Md (CellOfPrey North)
                             (CellOfNearest South)))
               West
               South))
     (ITE (LessThan (Md (CellOfSelf North)
                        (CellOfPrey West))
                    (Md (CellOfSelf (ITE false
                                         East
                                         (ITE (LessThan 25 (Md (CellOfSelf West)
                                                               (CellOfPrey North)))
                                              East
                                              South)))
                        (CellOfPrey West)))
          North
          East))
```

The best prey evolved using the CFD has the following strategy which is of size 118.

```
(ITE (LessThan
     (Md (CellOfSelf West)
         (CellOfNearest South))
     (Md (CellOfSelf (ITE (LessThan
                          (Md (CellOfSelf North)
                              (CellOfSelf South))
                          (Md (CellOfSelf West)
                              (CellOfNearest South)))
```

```
                        (ITE (LessThan
                             (Md (CellOfSelf (ITE (LessThan
                                                  (Md (CellOfSelf North)
                                                      (CellOfSelf South))
                                                  (Md (CellOfSelf West)
                                                      (CellOfNearest South)))
                                                  (ITE (LessThan
                                                       (Md (CellOf2ndNearest Decelerate)
                                                           (77,29))
                                                       (Md (42,12)
                                                           (CellOfNearest East)))
                                                       West
                                                       East)
                                                  West))
                                 (77,29))
                             (Md (42,12)
                                 (CellOfNearest East)))
                             West
                             East)
                        West))
        (CellOfNearest Continue)))
    (ITE (LessThan
         (Md (CellOfSelf (ITE (LessThan
                              (Md (6,48)
                                  (CellOfNearest Decelerate))
                              (Md (CellOfSelf West)
                                  (CellOfNearest South)))
                              East
                              West))
             (CellOfNearest Accelerate))
         (Md (CellOfSelf South)
             (CellOfNearest West)))
         East
         West)
    (ITE (LessThan
         (Md (CellOfSelf West)
             (CellOfNearest East))
         (Md (CellOfNearestWall Accelerate)
             (89,71)))
         (ITE (LessThan
              (Md (CellOfSelf South)
                  (CellOfNearest South))
              7)
              (ITE (LessThan
                   (Md (17,89)
                       (CellOfNearest Accelerate))
                   (Md (38,74)
                       (76,33)))
                   North
                   West)
              South)
         North))
```

F.2 Strategies Evolved using CellOfNearestRight()

Best evolved predator strategy using CellOfNearestRight().

```
(ITE (LessThan
     (Md (CellOfSelf West)
         (CellOfPrey Continue))
     (Md (CellOfSelf West)
         (CellOfPrey West)))
     West (IfThenElse (LessThan
     (Md (CellOfSelf North) (CellOfPrey Continue))
     (Md (CellOfSelf (IfThenElse (LessThan (Md
         (CellOfSelf West) (CellOfPrey West)) (Md
         (CellOfPrey North) CellOfNearestRight)) (IfThenElse
         false Accelerate (IfThenElse (LessThan (Md
         (55,37) (CellOfSelf East)) 15) Continue Accelerate))
```

```
        East)) (CellOfPrey Continue))) North (IfThenElse
    (LessThan 18 (Md (CellOfPrey West) (CellOfPrey
        North))) East (IfThenElse (LessThan (Md (CellOfSelf
    (IfThenElse (LessThan 20 14) (IfThenElse
        false Decelerate (IfThenElse false (IfThenElse
        (LessThan (Md CellOfNearestRight (CellOfSelf
            North)) 26) South East) South)) Accelerate))
    (CellOfPrey Continue)) (Md (CellOfSelf West)
    (CellOfPrey Continue))) (IfThenElse (LessThan
    (Md (CellOfPrey West) (CellOfSelf South))
    (Md (MyLastCell East) (CellOfSelf (IfThenElse
        true Continue West)))) (IfThenElse (LessThan
    (Md (CellOfSelf West) (CellOfSelf (IfThenElse
        true South South))) (Md (CellOfPrey West)
    (CellOfSelf South))) North West) (IfThenElse
    (LessThan (Md (CellOfSelf (IfThenElse (LessThan
        (Md (CellOfSelf (IfThenElse true North West))
            (CellOfPrey Continue)) 19) North East)) (CellOfPrey
        Continue)) (Md (CellOfPrey West) (CellOfSelf
        East))) Accelerate Accelerate)) (IfThenElse
    false (IfThenElse (LessThan (Md (CellOfPrey
    West) CellOfNearestRight) (Md CellOfNearestRight
    (75,79))) North South) (IfThenElse (LessThan
    (Md (MyLastCell East) (CellOfPrey Continue))
    (Md (CellOfSelf South) (CellOfPrey East)))
    East South))))))
```

Best evolved prey strategy against predators using CellOfNearestRight().

```
(IfThenElse (LessThan (Md (CellOfNearest
     South) (CellOfNearest West)) (Md (CellOfSelf
     (IfThenElse (LessThan 5 (Md (CellOfSelf (IfThenElse
         (LessThan 5 2) (IfThenElse true (IfThenElse
         (LessThan 25 29) North Continue) (IfThenElse
         true South North)) South)) (CellOfNearest
         West))) (IfThenElse false North Continue)
         (IfThenElse false (IfThenElse (LessThan 5
             (Md CellOfNearestVerticalWall (CellOfNearest
                 West))) North (IfThenElse (LessThan 13 22)
             West (IfThenElse false West (IfThenElse (LessThan
             5 (Md (CellOfNearest Accelerate) (CellOf2ndNearest
             West))) North West)))) South))) (CellOfNearest
     West))) (IfThenElse (LessThan 5 (Md (CellOfSelf
     West) (CellOfNearest West))) (IfThenElse
     false West (IfThenElse false North Continue))
     (IfThenElse (LessThan 5 (Md (CellOfSelf East)
         (CellOfNearest West))) (IfThenElse (LessThan
         5 2) North (IfThenElse true East North))
         (IfThenElse false Continue South))) (IfThenElse
     (LessThan 9 16) (IfThenElse false West (IfThenElse
     (LessThan 5 (Md (CellOfSelf North) (CellOf2ndNearest
         West))) North West)) South))
```

Bibliography

[1] B. Jagannathan Benda, M. and R. Dodhiawala. Optimal cooperation of knowledge sources. In *Technical Report BCS-G2010-28*, Boeing AI Center, Boeing Computer Services, Bellevue, WA, 1986.

[2] Stefan Bleuler, Martin Brack, Lothar Thiele, and Eckart Zitzler. Multiobjective genetic programming: Reducing bloat using SPEA2. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 536–543, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27-30 2001. IEEE Press. ISBN 0-7803-6658-1. URL `citeseer.nj.nec.com/bleuler01multiobjective.html`.

[3] Tobias Blickle. Evolving compact solutions in genetic programming: A case study. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving From Nature IV. Proceedings of the International Conference on Evolutionary Computation*, volume 1141, pages 564–573, Berlin, Germany, 22-26 1996. Springer-Verlag. ISBN 3-540-61723-X. URL `citeseer.nj.nec.com/blickle96evolving.html`.

[4] E. Cantú-Paz. A survey of parallel genetic algorithms, 1997. URL `citeseer.nj.nec.com/article/cantu-paz97survey.html`.

[5] John Cartlidge and Seth Bullock. Learning lessons from the common cold: How reducing parasite virulence improves coevolutionary optimization. URL `citeseer.nj.nec.com/550008.html`.

[6] Dave Cliff and Geoffrey F. Miller. Co-evolution of pursuit and evasion II: Simulation methods and results. In Pattie Maes, Maja J. Mataric, Jean-Arcady Meyer, Jordan B. Pollack, and Stewart W. Wilson, editors, *From animals to animats 4*, pages 506–515, Cambridge, MA, 1996. MIT Press. URL `citeseer.nj.nec.com/cliff95coevolution.html`.

[7] Edwin D. de Jong. Multi-agent coordination by communication of evaluations. In *Modelling Autonomous Agents in a Multi-Agent World*, pages 63–78, 1997. URL `citeseer.nj.nec.com/dejong97multiagent.html`.

[8] Edwin D. De Jong, Richard A. Watson, and Jordan B. Pollack. Reducing bloat and promoting diversity using multi-objective methods. In L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors, *Proceedings of*

the Genetic and Evolutionary Computation Conference, GECCO-2001, pages 11–18, San Francisco, CA, 2001. Morgan Kaufmann Publishers. URL citeseer.nj.nec.com/dejong01reducing.html.

[9] Kenneth A. De Jong and Jayshree Sarma. Generation gaps revisited. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 19–28. Morgan Kaufmann, San Mateo, CA, 1993. URL citeseer.nj.nec.com/dejong92generation.html.

[10] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii, 2000. URL citeseer.nj.nec.com/deb00fast.html.

[11] G. Dworman. Games computers play: simulating characteristic function game playing agents with classifier systems, 1994. URL citeseer.nj.nec.com/dworman94games.html.

[12] S. Ficici and J. Pollack. Coevolving communicative behavior in a linear pursuer-evader game. URL citeseer.nj.nec.com/115449.html.

[13] D. Floreano, S. Nolfi, and F. Mondada. Competitive co-evolutionary robotics: From theory to practice, 1998. URL citeseer.nj.nec.com/floreano98competitive.html.

[14] Dario Floreano and Stefano Nolfi. God save the red queen! competition in co-evolutionary robotics. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 398–406, Stanford University, CA, USA, 13-16 1997. Morgan Kaufmann. URL citeseer.nj.nec.com/964.html.

[15] Claudia V. Goldman and Jeffrey S. Rosenschein. Emergent coordination through the use of cooperative state-changing rules. In *Proceedings of the Twelfth International Workshop on Distributed Artificial Intelligence*, pages 171–185, Hidden Valley, Pennsylvania, May 1993. URL citeseer.nj.nec.com/article/goldman94emergent.html.

[16] Thomas Haynes and Sandip Sen. Evolving behavioral strategies in predators and prey. In Sandip Sen, editor, *IJCAI-95 Workshop on Adaptation and Learning in Multiagent Systems*, pages 32–37, Montreal, Quebec, Canada, 20-25 1995. Morgan Kaufmann. URL citeseer.nj.nec.com/haynes96evolving.html.

[17] Thomas Haynes and Sandip Sen. Cooperation of the fittest. In John R. Koza, editor, *Late Breaking Papers at the Genetic Programming 1996 Conference Stanford University July 28-31, 1996*, pages 47–55, Stanford University, CA, USA, 28–31 1996. Stanford Bookstore. ISBN 0-18-201031-7. URL citeseer.nj.nec.com/haynes96cooperation.html.

[18] Thomas Haynes and Sandip Sen. Crossover operators for evolving A team. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 162–167, Stanford University, CA, USA, 13-16 1997. Morgan Kaufmann. URL citeseer.nj.nec.com/haynes97crossover.html.

[19] Thomas Haynes, Roger Wainwright, and Sandip Sen. Evolving cooperation strategies. In Victor Lesser, editor, *Proceedings of the First International Conference on Multi–Agent Systems*, page 450, San Francisco, CA, 1995. MIT Press. URL `citeseer.nj.nec.com/haynes94evolving.html`.

[20] Thomas Haynes, Roger Wainwright, Sandip Sen, and Dale Schoenefeld. Strongly typed genetic programming in evolving cooperation strategies. In L. Eshelman, editor, *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 271–278, Pittsburgh, PA, USA, 15-19 1995. Morgan Kaufmann. ISBN 1-55860-370-0. URL `citeseer.nj.nec.com/haynes95strongly.html`.

[21] Thomas D. Haynes and Roger L. Wainwright. A simulation of adaptive agents in hostile environment. In K. M. George, Janice H. Carroll, Ed Deaton, Dave Oppenheim, and Jim Hightower, editors, *Proceedings of the 1995 ACM Symposium on Applied Computing*, pages 318–323, Nashville, USA, 1995. ACM Press. URL `citeseer.nj.nec.com/haynes95simulation.html`.

[22] J.H. Holland. *Adaption in Natural and Artificial Systems*. MIT Press, Cambridge, Massachuseus, 1975/1992.

[23] Jørn Holm and Jens Dalgaard Nielsen. Genetic programming - applied to a real time game domain, 2001.

[24] Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, 1994. IEEE Service Center. URL `citeseer.nj.nec.com/horn94niched.html`.

[25] Marcus Hutter. Fitness uniform selection to preserve genetic diversity. (IDSIA-01-01):13 pages, January 2001. URL `citeseer.nj.nec.com/hutter01fitness.html`.

[26] K. A. De Jong. An analysis of the behavior of a class of genetic adaptive systems, 1975.

[27] R. Keller and W. Banzhaf. Explicit maintenance of genetic diversity on genospaces, 1994. URL `citeseer.nj.nec.com/keller94explicit.html`.

[28] Kenneth E. Kinnear, Jr. Generality and difficulty in genetic programming: Evolving a sort. In Stephanie Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, pages 287–294, University of Illinois at Urbana-Champaign, 17-21 1993. Morgan Kaufmann. URL `citeseer.nj.nec.com/kinnear93generality.html`.

[29] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. RoboCup: The robot world cup initiative. In W. Lewis Johnson and Barbara Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 340–347, New York, 5–8, 1997. ACM Press. ISBN 0-89791-877-0. URL `citeseer.nj.nec.com/kitano95robocup.html`.

[30] John R. Koza. *Genetic Programming - On the Programming of Computers by Means of Natural Selection*. The MIT Press - ISBN 0-262-11170-5, 1992.

[31] John R. Koza. *Genetic Programming II - Automatic Discovery of Reusable Programs*. The MIT Press - ISBN 0-262-11189-6, 1994.

[32] Thiemo Krink, Brian H. Mayoh, and Zbigniew Michalewicz. A PATCHWORK model for evolutionary algorithms with structured and variable size populations. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1321–1328, Orlando, Florida, USA, 13-17 1999. Morgan Kaufmann. ISBN 1-55860-611-4. URL `citeseer.nj.nec.com/krink99patchwork.html`.

[33] Thiemo Krink and Rasmus K. Ursem. Parameter control using the agent based patchwork model. In *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, pages –83, La Jolla Marriott Hotel La Jolla, California, USA, 6-9 2000. IEEE Press. ISBN 0-7803-6375-2. URL `citeseer.nj.nec.com/krink00parameter.html`.

[34] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 157–163, New Brunswick, NJ, 1994. Morgan Kaufmann. URL `citeseer.nj.nec.com/littman94markov.html`.

[35] Sean Luke. Two fast tree-creation algorithms for genetic programming. *IEEE Transactions on Evolutionary Computation*, 4(3):274–283, 2000. URL `citeseer.nj.nec.com/luke00two.html`.

[36] Sean Luke and Lee Spector. Evolving teamwork and coordination with genetic programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 150–156, Stanford University, CA, USA, 28–31 1996. MIT Press. URL `citeseer.nj.nec.com/luke96evolving.html`.

[37] Samir W. Mahfoud. A comparison of parallel and sequential niching methods. In Larry Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 136–143, San Francisco, CA, 1995. Morgan Kaufmann. URL `citeseer.nj.nec.com/mahfoud95comparison.html`.

[38] Samir W. Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, Urbana, IL, USA, 1995. URL `citeseer.nj.nec.com/mahfoud95niching.html`.

[39] N. Minar, R. Burkhart, C. Langton, and M. Askenazi. The swarm simulation system, a toolkit for building multi-agent simulations, 1996. URL `citeseer.nj.nec.com/minar96swarm.html`.

[40] David J. Montana. Strongly typed genetic programming. Technical Report #7866, 10 Moulton Street, Cambridge, MA 02138, USA, 7 1993. URL `citeseer.nj.nec.com/montana93strongly.html`.

[41] Jason Noble and Richard A. Watson. Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for pareto selection. In Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael

Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 493–500, San Francisco, California, USA, 7-11 2001. Morgan Kaufmann. ISBN 1-55860-774-9. URL `citeseer.nj.nec.com/noble01pareto.html`.

[42] Peter Nordin and Wolfgang Banzhaf. Complexity compression and evolution. In L. Eshelman, editor, *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 310–317, Pittsburgh, PA, USA, 15-19 1995. Morgan Kaufmann. ISBN 1-55860-370-0. URL `citeseer.nj.nec.com/nordin95complexity.html`.

[43] Michael V. Rydtoft & Henrik S. Rasmussen. Cooperation Among Autonomous Adaptive agents. Technical report, 2003.

[44] Craig W. Reynolds. Competition, coevolution and the game of tag, 1994.

[45] Christopher D. Rosin. *Coevolutionary search among adversaries*. PhD thesis, San Diego, CA, 1997. URL `citeseer.nj.nec.com/rosin97coevolutionary.html`.

[46] Christopher D. Rosin and Richard K. Belew. Methods for competitive co-evolution: Finding opponents worth beating. In Larry Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 373–380, San Francisco, CA, 1995. Morgan Kaufmann. URL `citeseer.nj.nec.com/rosin95methods.html`.

[47] Christopher D. Rosin and Richard K. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5(1):1–29, 1997. URL `citeseer.nj.nec.com/rosin96new.html`.

[48] Günter Rudolph. Evolutionary search under partially ordered finite sets. In M. F. Sebaaly, editor, *Proceedings of the International NAISO Congress on Information Science Innovations (ISI 2001)*, pages 818–822, Dubai, U. A. E., 2001. ICSC Academic Press. ISBN 3–906454–25–8. URL `citeseer.nj.nec.com/rudolph99evolutionary.html`.

[49] Jayshree Sarma and Kenneth De Jong. An analysis of the effects of neighborhood size and shape on local selection algorithms. In H. Voigt, W. Ebeling, and I. Rechenberg, editors, *Parallel Problem Solving from Nature – PPSN IV (Berlin, 1996)*, pages 236–244, Berlin, 1996. Springer. URL `citeseer.nj.nec.com/128357.html`.

[50] Franciszek Seredynski. Coevolutionary game-theoretic multi-agent systems. In *International Syposium on Methodologies for Intelligent Systems*, pages 356–365, 1996. URL `citeseer.nj.nec.com/189007.html`.

[51] R. Shipman, M. Shackleton, M. Ebner, and R. Watson. Neutral search spaces for artificial evolution: a lesson from life, 2000. URL `citeseer.nj.nec.com/shipman00neutral.html`.

[52] Karl Sims. Evolving 3D Morphology and Behavior by Competition. pages 28–39, 1994.

[53] Terence Soule and James A. Foster. Effects of code growth and parsimony pressure on populations in genetic programming. *Evolutionary Computation*, 6(4):293–309, Winter 1998. URL `citeseer.nj.nec.com/soule98effects.html`.

[54] N Srinivas and K Deb. Multi-objective function optimazation using non-dominated sorting genetic algorithms. In *Evolutionary Computation*, pages 221–248, 1995.

[55] R. A. Watson and J. B. Pollack. Coevolutionary dynamics in a minimal substrate. In Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, , and Edmund Burke, editors, *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*. Morgan Kaufmann, 2001. URL `citeseer.nj.nec.com/watson01coevolutionary.html`.

[56] J. Werfel, M. Mitchell, and J. P. Crutchfield. Resource sharing and coevolution in evolving cellular automata. *IEEE-EC*, 4(4):388, November 2000. URL `citeseer.nj.nec.com/werfel99resource.html`.

[57] R. Paul Wiegand. Applying Diffusion to a Cooperative Coevolutionary Model, 1999. URL `http://www.tesseract.org/paul/papers/ppsn99-dcga-rpw.pdf`.

[58] Byoung-Tak Zhang and Heinz Mühlenbein. Balancing accuracy and parsimony in genetic programming. *Evolutionary Computation*, 3(1):17–38, 1995. URL `citeseer.nj.nec.com/zhang95balancing.html`.

[59] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms on Test Functions of Different Difficulty. In Annie S. Wu, editor, *Proceedings of the 1999 Genetic and Evolutionary Computation Conference. Workshop Program*, pages 121–122, Orlando, Florida, 1999. URL `citeseer.nj.nec.com/zitzler00comparison.html`.

[60] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001. URL `citeseer.nj.nec.com/zitzler02spea.html`.

[61] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001. URL `citeseer.nj.nec.com/article/zitzler01spea.html`.

[62] Eckart Zitzler and Lothar Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999. URL `citeseer.nj.nec.com/zitzler99multiobjective.html`.