Understanding Competitive Co-evolutionary Dynamics via Fitness Landscapes

Elena Popovici and Kenneth De Jong

George Mason University, Fairfax, VA 22030
epopovic@gmu.edu, kdejong@gmu.edu

Abstract. Co-evolutionary EAs are often applied to optimization and machine learning problems with disappointing results. One of the contributing factors to this is the complexity of the dynamics exhibited by co-evolutionary systems. In this paper we focus on a particular form of competitive co-evolutionary EA and study the dynamics of the fitness of the best individuals in the evolving populations. Our approach is to try to understand the characteristics of the fitness landscapes that produce particular kinds of fitness dynamics such as stable fixed points, stable cycles, and instability. In particular, we show how landscapes can be constructed that produce each of these dynamics. These landscapes are extremely similar when inspected with respect to traditional properties such as ruggedness / modality, yet they yield very different results. This shows there is a need for co-evolutionary specific analysis tools.

1 Introduction

Co-evolutionary EAs are often applied to optimization and machine learning problems with disappointing results. One of the contributing factors to this is the complexity of the dynamics exhibited by co-evolutionary systems. This is somewhat less of a problem with *cooperative* co-evolutionary EAs (see, for example, [1]), but is a central issue for *competitive* co-evolutionary EAs (see, for example, [2] or [3]). In addition, for applications like optimization and machine learning we are interested primarily in how the fitness of the best individuals evolves over time, a much more dynamic property than, say, population averages.

In this paper we focus on a particular form of competitive co-evolutionary EA and study the dynamics of the fitness of the best individuals in the evolving populations. Our approach is to try to understand the characteristics of the fitness landscapes that produce particular kinds of space and fitness dynamics such as stable fixed points, stable cycles, and instability. In particular, we show how landscapes can be constructed that produce each of these dynamics. The sensitivity of the dynamics with respect to the population size is investigated as well.

Additionally, it is pointed out that these landscapes are extremely similar with respect to traditional characterizations (e.g. ruggedness/modality) yet the behaviors the same algorithm exhibits on them can differ dramatically. This goes to show the need for different metrics / methodologies / properties to be designed / analyzed for co-evolutionary algorithms.

2 Cycling Behavior / Chasing Tails

2.1 Co-evolutionary Setup

The goal of this work was to generate the typical kinds of behaviors that the dynamical systems literature talks about (fixed points, stable cycles, and instability) with as simple a setup as possible. Therefore, we settled on using a basic competitive co-evolution model consisting of two populations with conflicting goals. Individuals in either of these populations are real numbers in the interval $[0, n]$. An individual in the first population (the X - population) can only be evaluated in combination with an individual from the second population (the Y - population) and vice-versa. When an x and a y are paired, they both receive the same fitness value, given by some function $f(x, y)$. However, the goal of the X individuals is to get as high values as possible, whereas the Y individuals are as good as small their value is.

During the run of the algorithm the two populations take turns. That is to say that during each generation only one of the populations is active, the other one is stalled. All the individuals in the active population are evaluated in combination with the current best individual from the stalled population. Once the active population is evaluated, selection and recombination take place on it. At this point, this population is stalled, and the formerly stalled one becomes active and goes through the same process, being evaluated against the current (possibly newly found) best individual in the other population. And the cycle keeps repeating.

2.2 Algorithm Behavior Generates Landscape Ideas

The way this type of algorithm behaves can be formalized a little bit as follows. For a fixed x value, there are some values for y that produce the minimum value for the function. We will be looking only at functions for which there is a unique y with this property for every x (denoted by $minY(x)$). Similarly, for a fixed y, there will be only one x among all $x - s$ that produces the minimum of the function for that y, called $maxX(y)$.

Consider plotting on the same chart the $minY(x)$ and $maxX(y)$ curves of the function to study (their shape is not relevant for now). Then the behavior of the algorithm can be simulated like this: Let P_0^x be the initial X-population and y_0 a random y value with respect to which P_0^x is evaluated and then evolved. If the best individual in P_0^x is to be as good as theoretically possible, it should be given by $maxX(y_0)$, as good values for the X-population are big values of the function. This x can be graphically obtained by intersecting the horizontal line with equation $y = y_0$ with the $x = maxX(y)$ curve. Let x_1 be the x value thus found.

It is now the turn for P_0^y to be evaluated with respect to x_1 and then evolved. If the best individual in P_0^y is to be as good as theoretically possible, it should be given by $minY(x_1)$, as good values for the Y population are small values of the function. This y can be graphically obtained by intersecting the vertical line

with equation $x = x_1$ with the $y = minY(x)$ curve. Let y_2 be the y value thus found.

The process now continues in the same fashion, by alternatively drawing horizontal and vertical lines and intersecting them with the $maxX(y)$ and $minY(x)$ curves respectively, generating cobweb-like diagrams (the term was introduced in [3], although in a different kind of setup).

With this kind of mechanism in mind, it was immediately obvious how to generate cyclic behavior. Consider a function $f(x, y)$ for which $minY(x)$ is the main diagonal of the square $[0, n]$ x $[0, n]$ and $maxX(y)$ is the second diagonal. It is straightforward to see that applying the graphical technique above, the cobweb diagram quickly degenerates into a 4 point cycle (the 4 points residing on the diagonals and being the vertices of a square whose position/size are determined by the initial starting point).

2.3 Function Definition

And so the first function - named *cyclingRidges* - was constructed such that its $maxX(y)$ and $minY(x)$ are the two diagonals of the space as discussed.

To have these properties, the function was designed as follows. On the main diagonal, the function increases uniformly on $[0, n/2]$ from 0 to n, i.e. with a slope of 2 and then it decreases at the same rate from n to 0 on $[n/2, n]$. On the second diagonal, the function decreases uniformly on $[0, n/2]$ from $2n$ to n with slope 2 and then it increases at the same rate from n to $2n$ on $[n/2, n]$. The diagonals split the space into four areas. In the west and east areas, the function decreases along the x axis from the diagonals towards the corresponding edge of the space with a slope of 1. In the north and south areas, the function increases along the y axis from the diagonals towards the edges of the space with slope 1.

A three dimensional representation of the function can be seen in figure 1 from two perspectives. It's mathematical expression is described below (as Java code). The *ridges* function is the generic skeleton that the other two functions presented in this paper (see sections 3 and 4) use as well. The first three branches describe the values the function takes along the two curves that are defined by $minY(x)$ and $maxX(y)$. For *cyclingRidges* $minY(x)$ and $maxX(y)$ represent the two diagonals. The rest of the branches compute the value of the function in a certain point by adding/subtracting something to/from the value of the function in a corresponding point on one of the $minY(x)$ and $maxX(y)$ curves.

```
double ridges(double i, double j) {
    if (i <= n / 2 && j == fa(i))
        return (2 * i);
    else if (i >= n / 2 && j == fb(i))
        return (2 * n - 2 * i);
    else if (j == fc(i))
        return (Math.max(2 * i, 2 * n - 2 * i));
    else if (i <= n / 2)
    {
```

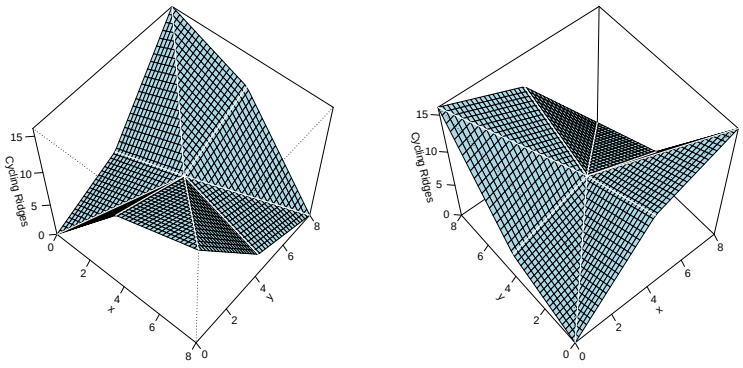

Fig. 1. The cyclingRidges function.

```
        if (j <= fa(i))
            return (2 * i + (fa(i) - j));
        else if (j <= n / 2)
            return (ffaInv(j) - (faInv(j) - i));
        else if (j <= fc(i))
            return (ffcInv(j) - (fcInv(j) - i));
        else
            return (Math.max(2 * i, 2 * n - 2 * i) + (j - fc(i)));
    }
    else
    {
        if (j <= fc(i))
            return (Math.max(2 * i, 2 * n - 2 * i) + (fc(i) - j));
        else if (j <= n / 2)
            return (ffcInv(j) - (i - fcInv(j)));
        else if (j <= fb(i))
            return (ffbInv(j) - (i - fbInv(j)));
        else
            return (2 * n - 2 * i + (j - fb(i)));
    }
}
```

Where:

- $y = fa(x) = x$ is the equation of the first half of the main diagonal
- $y = fb(x) = x$ is the equation of the second half of the main diagonal
- $y = fc(x) = n - x$ is the equation of the second diagonal

Why the distinction between the first and second half of the main diagonal was made will be evident when describing the other two functions.

- $faInv$, $fbInv$ and $fcInv$ are the inverses of fa, fb and fc respectively.
- $ffaInv(y) = f(faInv(y), y) = 2 * faInv(y)$
- $ffcInv(y) = f(fbInv(y), y) = 2 * n - 2 * fbInv(j)$
- $ffcInv(y) = f(fcInv(y), y) = max(2 * fcInv(j), 2 * n - 2 * fcInv(j))$

2.4 In Practice, Practice Agrees with Theory

Since evolutionary algorithms are stochastic processes, we wanted to test that the behavior of the algorithm on the engineered function is as predicted. The algorithm settings and the methodology presented here apply to sections 3 and 4 as well.

Algorithm Settings The general dynamics of the co-evolutionary algorithm used were described in subsection 2.1. Here we present the details.

The algorithm employs a real number representation and a generational model, with tournament selection of size two and gaussian mutation with a standard deviation of 0.25 applied at a rate of 0.95. Unless otherwise specified, the population size was set to 100 individuals and the algorithm was run for 100 generations.

As far as the fitness functions are concerned, $n = 8$ was used in all the experiments presented here.

For each experiment multiple runs were conducted (in the order of 10) and visually inspected (as it is hard to average trajectories) and typical runs are displayed in the charts.

Methodology In each case, the dynamics of the co-evolutionary process are inspected from two perspectives:

- how does the current best of each population move around in the space
- how does the fitness of the current best of each population change in time.

Therefore, two plots are produced, one for each perspective. For *cyclingRidges* they are shown in figure 2. There is a one-to-one correspondence between the points in the two plots.

The movement-across-the-space plot contains three data series. The dark colored one represents the data obtained from running the algorithm; each point is the pair of the current best in one population with the best of the other population at the previous generation; the lines are directed (with respect to time/generation number) always counter-clockwise for the functions investigated here. The light color trend shows for every y value which is the x that gives the highest function value (i.e. $maxX(y)$) and the medium shade trend shows for every x value which is the y that gives the lowest function value (i.e. $minY(x)$).

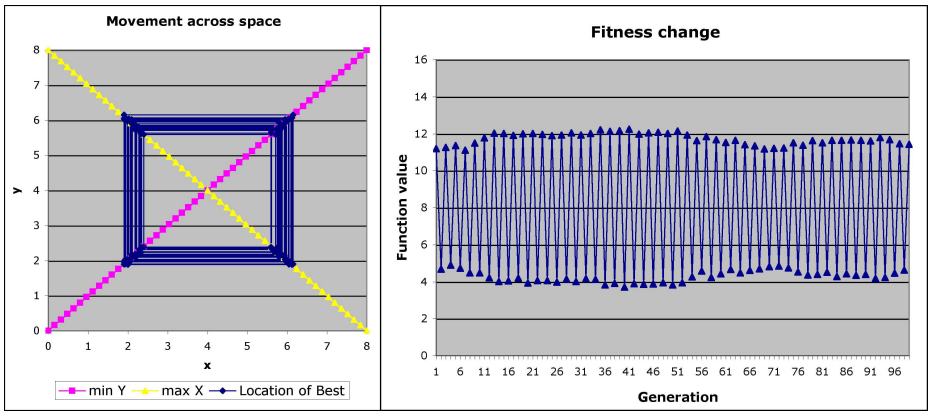

Fig. 2. Behavior on the cyclingRidges function.

Results and Discussion Researchers in co-evolution have previously introduced a series of concepts (such as mediocre stable states, arms race, etc.) to describe the behavior of the algorithms. The results of the experiments conducted here are presented in the light of these existing concepts.

The movement-across-the-space plot indicates that the experiment results closely follow the predicted behavior. There is still a certain degree of variance due to the stochastic nature of the algorithm, but the cycling trend is strong.

Figure 2 shows a type of cycling behavior in which the two populations take turns in "beating" each other by pulling each other's best through four regions, two of which are favorable to X and the other two favorable to Y. These regions are strongly dependent on the initial starting point (which one of them actually contains and is developed around) and are hardly ever escaped. The search space is not explored beyond them.

Tracing best fitness, the only changes observed over time are slight variations due to stochastic effects and alternative swaps at each generation between the local fitness range good for X and the one good for Y. Neither of the populations exhibits any significant change in fitness from the circumstantial initial preferred value found. This is also a side effect of the symmetry of the function on each diagonal with respect to the other. If it were not like that, each population would then have two preferred fitness areas that would alternatively be sampled.

This behavior can be regarded as a kind of stagnation, although it is a different type from what is referred in the literature as reaching a stable state. It is what one would call a "stable cycle", or a case of "dynamical stability". The two populations are just chasing each other's tail. Whether the best fitnesses produced by either population are very good or mediocre ones is circumstantial, depending on the starting point. In addition, the way the function is set up, the best fitnesses of the two populations fall in the same category, they are both either good or mediocre, neither X nor Y beats the other one at a higher degree.

We can talk about two kinds of mediocrity here: the fitness values *may happen to actually be* mediocre and the behavior *is* mediocre, i.e. performance is not improving at all after start-up.

3 Fixed-Point Behavior / Mediocre Stable States

Maybe an even more easily comprehendible kind of trajectory for a dynamical system is that of reaching not a stable cycle, but a stable fixed point.

3.1 Function Definition

It took a small change in the *cyclingRidges* function to transform it into one that generates collapsing to a point behavior. In particular, the change was modifying $minY(x)$ from a straight line into an S-shaped curve composed of two parts, each of which is a quarter of a (different) circle. As the reader might have already guessed, these pieces will be defined by the fa and fb functions that the *ridges* function in subsection 2.3 makes use of. Their expressions are given below:

- $fa(x) = sqrt(i * (n - i))$
- $fb(x) = n - sqrt(i * (n - i))$

The shape defined by these functions can be seen in figure 4 as the medium shade curve. The rest of the function definition is exactly the same as for *cyclingRidges*, with the mention that the expressions for $faInv$ and $fbInv$ will have to be recomputed accordingly. $fc(x)$ remains unchanged.

Applying the procedure described in subsect 2.2 the shape generated is an inward going spiral. Regardless of where the process starts, the trajectory gets closer and closer with each step to the $(n/2, n/2)$ point, converging to it in the limit. Therefore this function was called *collapsingRidges* and spatials views of it are shown in figure 3.

3.2 Experiment Results and Discussion

Experiments were conducted to test the behavior of the *collapsingRidges* function using the same setup as described in subsections 2.4 and 2.4.

As figure 4 shows, there is very strong agreement between the experiment results and the prediction, with a lot less variance than in the case of the *cyclingRidges* function.

The behavior of the algorithm on this function is a definite example of convergence to a mediocre stable state. The "stable state" is evident both on the movement plot, on which the trajectory goes into the saddle point $(n/2, n/2)$ and stays there, and on the fitness plot, which shows the fitness stabilizing at a value of n. This is mediocre, because it is half way between the minimum and the maximum possible. In addition, over time the fitness of the best individuals produced by the two populations gets worse and worse (lower for X and higher for Y).

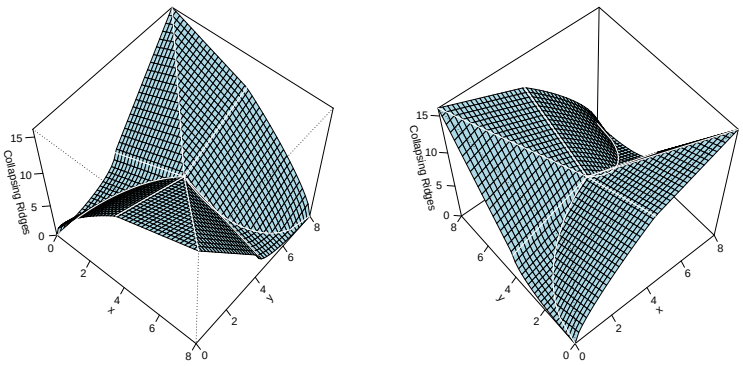

Fig. 3. The collapsingRidges function.

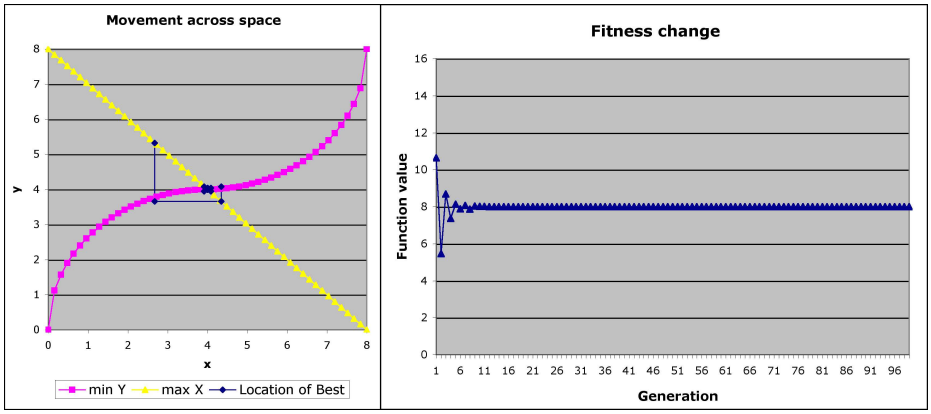

Fig. 4. Behavior on the collapsingRidges function.

4 Unstable Behavior / Arms Race

Both the cycling and the collapsing behaviors are examples of convergence to stability, whether it's a dynamic stability or a static one. The next challenge was to find a landscape on which the same algorithm would exhibit unstable behavior.

4.1 Function Definition

Once more, it took only a small change, on top of the same skeleton, to convert the previous function from one that generates collapsing behavior into one that generates expanding behavior. This third function is called *expandingRidges*. In particular, the change was replacing the S-shaped $minY(x)$ of *collapsingRidges* with its symmetric with respect to the main diagonal. The new expressions for the fa and fb functions are given below:

- $fa(x) = n/2 - sqrt((n/2) * (n/2) - i * i)$
- $fb(x) = n/2 + sqrt((n/2) * (n/2) - (n - i) * (n - i))$

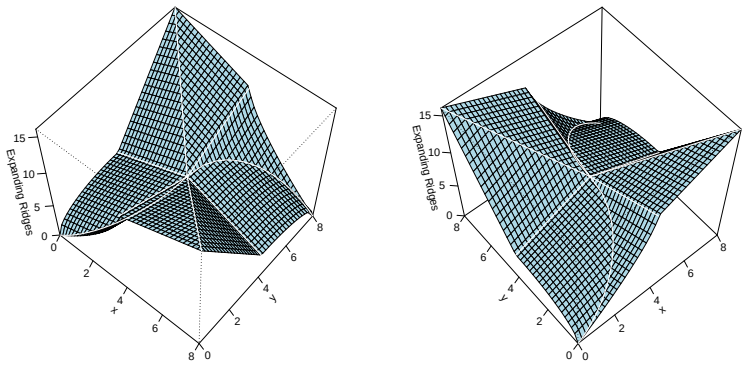

Fig. 5. The expandingRidges function.

The new shape of $minY(x)$ can be seen in figure 6 as the medium shade curve and spatials views of the function are shown in figure 5.

The limit behavior of *expandingRidges* is also independent of the starting point, but this time, by applying the procedure described in subsection 2.2, an

outward spiral can be observed, that gets closer and closer to the edges of the space (as it cannot get out).

4.2 Experiment Results and Discussion

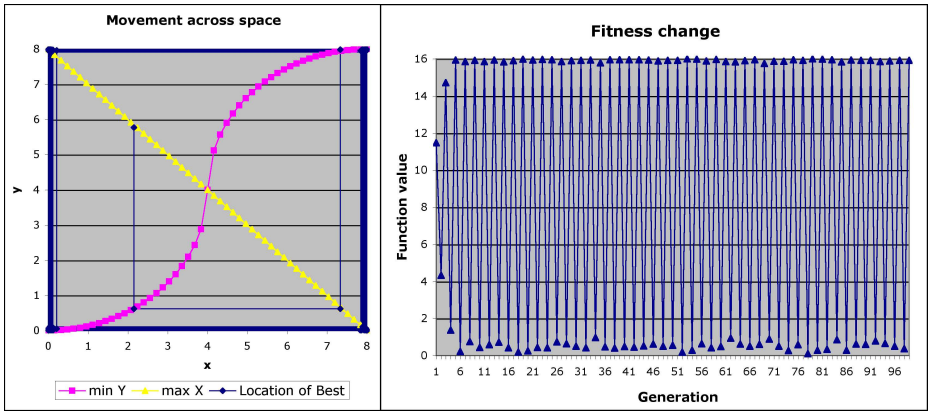

Fig. 6. Behavior on the expandingRidges function.

With the same experiment settings as before, in practice, the practice agreed with the theory once more. And again, there was very little variance in the expanding trend of the space trajectory of the system, as figure 6 shows.

The effect of the algorithm on this landscape can be related to the notion of arms race, one type of behavior that is considered desirable, especially when co-evolution is used as an optimization technique. The space trajectory drawn by the current best individuals of the two populations has the shape of a spiral expanding towards the boundaries of the space and then cycling there because it is not allowed to go out. If the space were infinite, it would keep expanding for ever.

The same trend is visible on the fitness plot. While the populations still take turns in "beating" each other, the function values that the best individuals are reaching are getting better and better over time until they reach the maximum (for X) and minimum (for Y) after which there is just light variance in the values due to the stochastic nature of the algorithm. Again, if the space were infinite, unlimited growth in function values would be seen.

5 Population Size Effects

It was interesting to note that while the cycling, collapsing and respectively expanding trends are clearly visible on the three landscapes, the highest degree

of trajectory variance is in the cycling case and this trend showed up again in one additional experiment presented here.

The experiment was conducted to observe the effects of a dramatically smaller population size, namely 10 and, as could be expected, there is increasing variance due to sampling error.

For the *collapsingRidges* function, the effect is slightly slower convergence to the $(n/2, n/2)$ point, and sometimes the trajectory temporarily escapes it only to be drawn back in. Similarly, in the *expandingRidges* case, it takes a bit more time for the trajectory to reach the edges of the space and it can be from time to time pulled back in for a while only to consequently expand again.

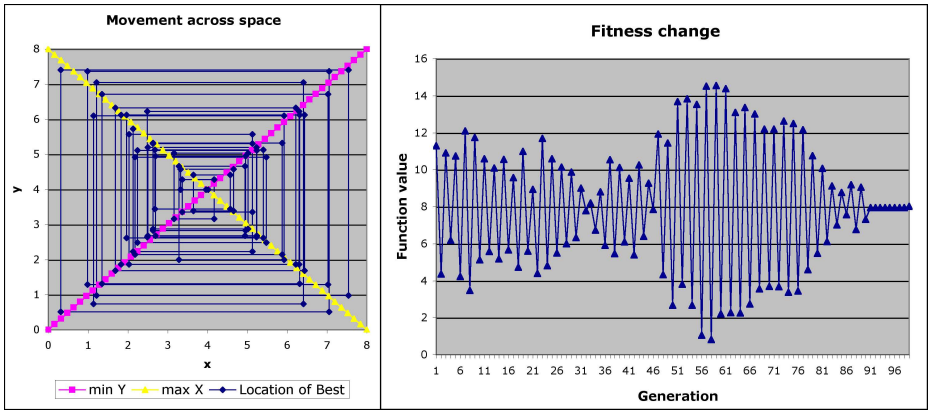

Fig. 7. The cyclingRidges function. Behavior for population size 10.

The *cyclingRidges* function shows a more drastic behavior difference in this case. As it can be seen in figure 7, the trajectory travels all over the place, highly exploring the space, and in the particular movement plot shown it even ends up stabilizing (maybe just for a while?) in the saddle point in the middle of the space.

One explanation for this behavior is the fact that for this function the gradients of both sides of all ridges are fairly close, yielding comparable probabilities of slipping off on either side. For the *collapsingRidges* and *expandingRidges* functions, along the ridges defines by the $minY(y)$ curves the gradient is much higher on one side than on the other, yielding higher probability of slipping on the side on which being close to the ridge will pull towards convergence (or respectively divergence) anyway.

6 Conclusions

This paper provided some new insights into the nature of the complex dynamics that co-evolutionary systems can exhibit. It showed that a very simple co-evolutionary algorithm can exhibit very different dynamics on similar landscapes. Thus, one very important thing that the results suggest is that traditional methods of characterizing fitness landscapes are not very helpful when dealing with co-evolution. The differences between the three crafted functions with respect to notions such as ruggedness / modality or deception are so slight or inexistent, and yet the resulting behaviors of the algorithm are so drastically different. It is hard both formally and for the human eye to distinguish what is fundamentally different between these landscapes, especially the last two.

Additionally, interesting results were obtained concerning the sensitivity of the dynamics with respect to the population size employed by the algorithm. On some landscapes, the behavior of the algorithm is robust with regard to this setting (e.g. the *cyclingRidges* and *expandingRidges* functions), whereas on others, drastic changes in terms of dynamics can occur due to the higher variance that a small population size causes.

The presented work reveals promising directions of research for identifying landscape properties relevant to co-evolutionary algorithms. These properties will have to initially be studied on customly designed functions. Additional co-evolutionary algorithms must be investigated on such functions. The goal is to match the right algorithm to a domain whose properties are identified (and hope to be able to do this for real world domains).

References

1. Wiegand, R.P.: An Analysis of Cooperative Coevolutionary Algorithms. PhD thesis, George Mason University, Fairfax, Virginia (2004)
2. Ficici, S., Pollack, J.: Challenges in coevolutionary learning: Arms–race dynamics, open–endedness, and mediocre stable states. In et al, A., ed.: Proceedings of the Sixth International Conference on Artificial Life, Cambridge, MA, MIT Press (1998) 238–247
3. Ficici, S., Pollack, J.: Game–theoretic investigation of selection methods used in evolutionary algorithms. In et al, Z., ed.: Proceedings of the 2000 Congress on Evolutionary Computation. (2000) 880–887