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 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.

Additionally, it is pointed out that the differences between 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 Coevolutionary Setup

The co-evolutionary algorithm used for investigations in this paper is based on a very simple competitive model and yet, as it will be seen, it can exhibit a variety of interesting behaviors.

The setup consists of having 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 inside one generation only one of the populations is active, the other one is frozen. All the individuals in the active population are evaluated in combination with the current best individual from the frozen population. Once the active population is evaluated, selection and recombination take place on it. At this point, this population is frozen, and the formerly frozen 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.

3 Functions

With this algorithm setting in mind and the goal of studying what kinds of behaviors it can exhibit, three different functions were engineered, each causing the algorithm to have different dynamics, both in terms of space exploration and fitness changes.

3.1 Definitions

All three functions used are two dimensional ones operating on the square $[0, n]$ x $[0, n]$.

The first function was constructed with the goal of getting cyclic behavior from the particular algorithm used. It will be referred as *cyclingRidges*. The following two properties were considered as very likely to generate such behavior. For every x value, the y that produces the minimum value for the function is unique (denoted by $minY(x)$) and is located on the main diagonal. For every y value, the x that produces the maximum value for the function ($maxX(y)$) is unique and is located on the second diagonal.

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 sections, 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 sections, the function increases along the y axis from the diagonals towards the edges of the space with a slope of 1.

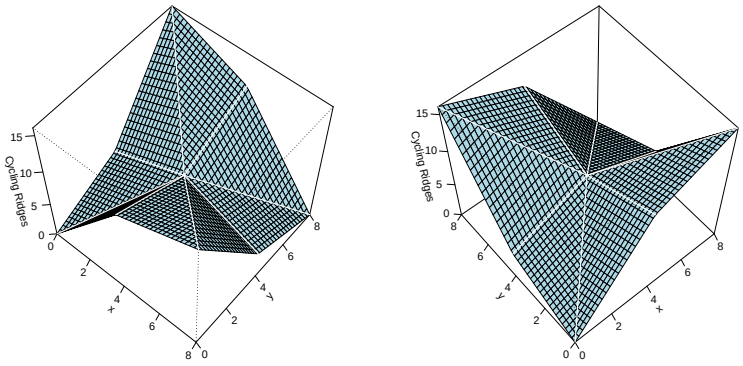

Fig. 1. Cycling Ridges function.

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 all three functions constructed here use. 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)
    {
        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.
- $ffaInv(y) = f(faInv(y), y) = 2 * faInv(y)$
- $ffcInv(y) = f(fbInv(y), y) = 2 * n - 2 * fbInv(j)$
- $ffcInv(y) = f(fcInv(y), y) = max(2 * fcInv(j), 2 * n - 2 * fcInv(j))$

The second and the third functions, *collapsingRidges* and *expandingRidges*, were obtained by changing the $minY(x)$ curve from a straight line into S-like curves. Such a curve is composed of two parts and each part 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. Their expressions are given below:

collapsingRidges:

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

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

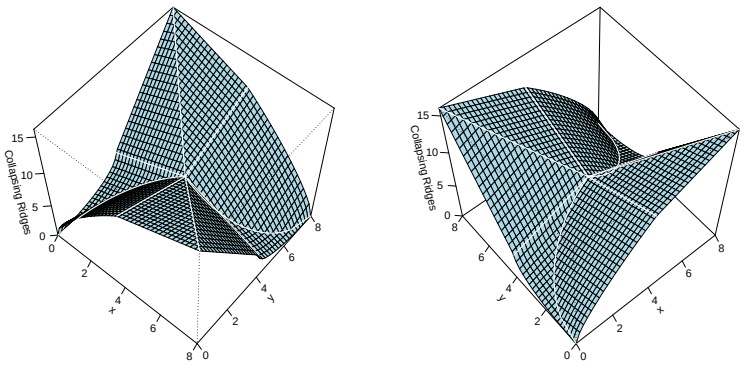

Fig. 2. Collapsing Ridges function.

The two $minY(x)$ curves for *collapsingRidges* and *expandingRidges* are the mirroring of one another with respect to the main diagonal. The reader can already inspect their shapes in figure 5 and figure 6.

The $maxX(y)$ curve is unchanged for all three functions, namely the straight line of the second diagonal. Therefore $fc(x)$ is unchanged as well.

Spatial views of *collapsingRidges* and *expandingRidges* are shown in figure 2 and figure 3.

3.2 Explanations

Why would these functions exhibit the cycling, collapsing and expanding behaviors that we seek? Here's the explanation. Consider plotting on the same chart the $minY(x)$ and $maxX(y)$ curves of the function to study and remember how the algorithm works.

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

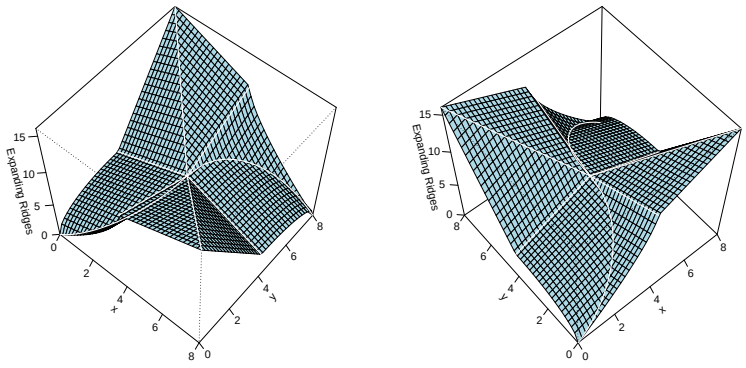

Fig. 3. Expanding Ridges function.

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 of 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).

It is easy to see that for *cyclingRidges* the cobweb immediately runs into a (rectangular) cycle whose position (edge size) is dependent on the initial starting point (y_0). For *collapsingRidges*, 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. The shape generated is an inward going spiral. The limit behavior of *expandingRidges* is also independent of the starting point, but this time an outward spiral can be observed, that gets closer and closer to the edges of the space.

4 Results and Analysis

4.1 Algorithm Settings

The general dynamics of the co-evolutionary algorithm used were described in section 2. 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 number of generations for which the algorithm was run was also 100.

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

For each function 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.

4.2 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.

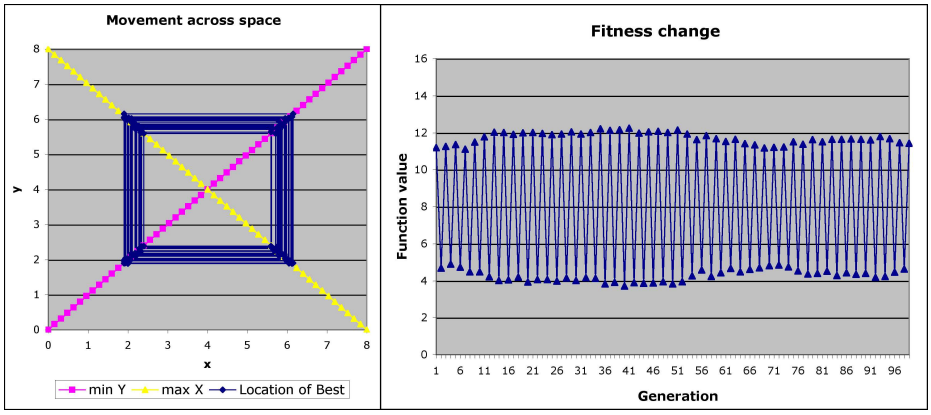

Fig. 4. The cyclingRidges function.

Therefore, for each function two plots are produced, one for each perspective, and they are shown in figures 4 - cyclingRidges, 5 - collapsingRidges and 6 - expandingRidges. There is a one-to-one correspondence between the points in the plots of such a pair.

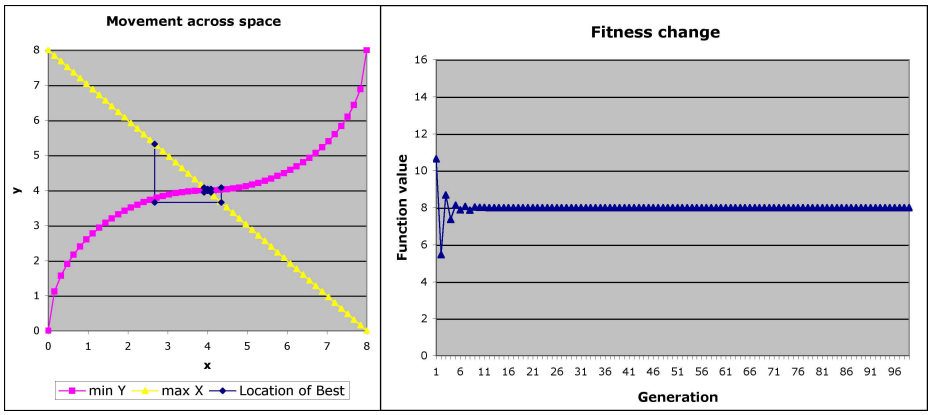

Fig. 5. The collapsingRidges function.

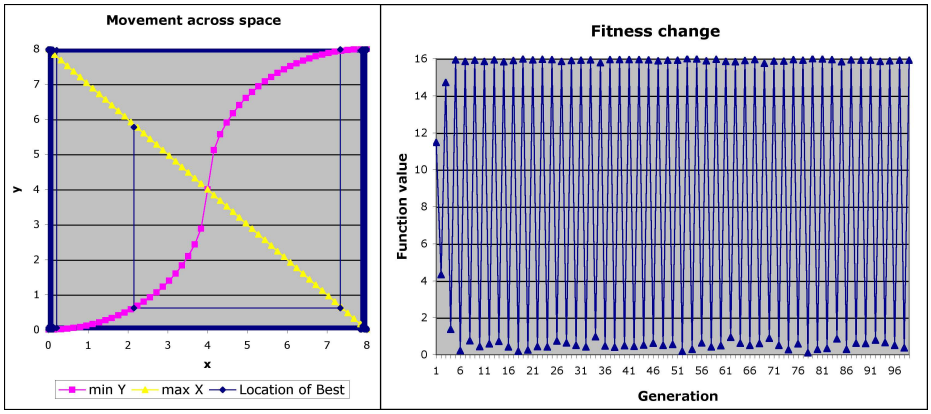

Fig. 6. The expandingRidges function.

Movement-across-the-space type plots contain three data series. The dark colored one represents the data obtained from running the algorithm and the lines are directed (with respect to time/generation number) always counter-clockwise for the functions investigated. 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. $minY(x)$).

4.3 In Practice, Practice Agrees with Theory

The most visible thing from the movement-across-the-space type plots is that they closely follow the predicted behavior. There is still a (fairly low) degree of variance due to the stochastic nature of the algorithm, but the cycling, collapsing and respectively expanding trends are strong.

It is interesting to note that the highest degree of trajectory variance is in the cycling case, and this trend will show again in a later experiment presented in subsection 4.4. One explanation 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 define 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.

4.4 Population Size Effects

As some variance in the cycling/collapsing/expanding nature of the trajectories has been noticed (and was expected), an additional experiment was conducted to observe the effects of a dramatically smaller population size, namely 10.

As predicted, 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.

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. Explanations for this behavior have been suggested in subsection 4.3.

4.5 Chasing Tails, Mediocre Stable States and Arms-Race Dynamics

A number of buzz-words have emerged in the field of co-evolutionary algorithms trying to characterize their behavior and point presumably desirable kinds of

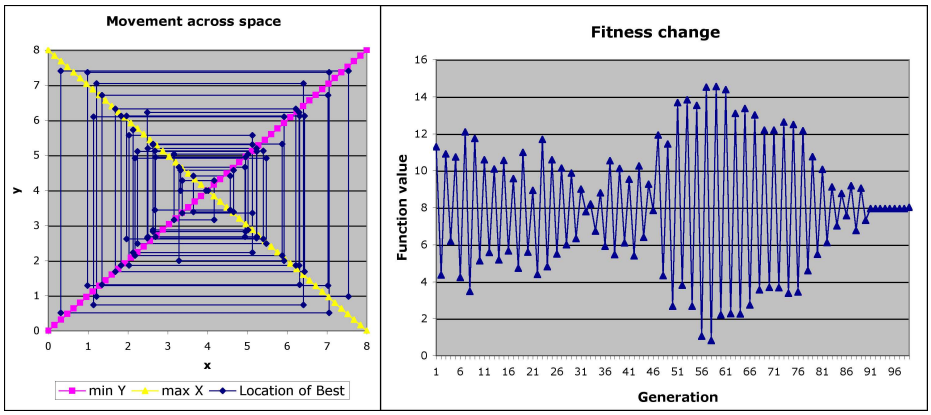

Fig. 7. The cyclingRidges function. Behavior for population size 10.

behavior. The results of the conducted experiments are presented in the light of these existing notions.

cyclingRidges Figure 4 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 kind of 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. The two populations are just chasing each other's tail. Whether the best fitnesses produced by either population are very good, mediocre or lousy 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, mediocre or lousy, neither X nor Y beats the other one at a higher degree.

collapsingRidges The behavior of the algorithm on the second function is a definite example of convergence to a mediocre stable state, as can be easily

seen in figure 5. 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).

expandingRidges Finally, the third function is an example of an 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 (see figure 6) 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 Conclusions

This paper took a new approach at analyzing dynamics of competitive co-evolutionary algorithms. These dynamics were studied on real-valued function landscapes, as opposed to most existing co-evolution studies which focused on game domains. In addition, the dynamics were observed from two perspectives - movement over space and fitness change - and the joint analysis of the insights each of them gives facilitates a better understanding of the process.

One very important thing to be noted as a result of this work is how useless traditional methods of characterizing fitness landscapes are when dealing with co-evolution. The differences between the three 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.

The work presented opens directions of research that hold promise for identifying landscape properties that are relevant to co-evolutionary algorithms. These will have to initially be investigated on fabricated functions, and then taken one (big) step further to apply them in 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