Understanding Cooperative Co-evolutionary Dynamics via Simple Fitness Landscapes

Elena Popovici
George Mason University
Fairfax, VA
epopovic@gmu.edu

Kenneth De Jong
George Mason University
Fairfax, VA
kdejong@gmu.edu

ABSTRACT

Cooperative co-evolution is often used to solve difficult optimization problems by means of problem decomposition. Its performance for such tasks can vary widely from good to disappointing. One of the reasons for this is that attempts to improve co-evolutionary performance using traditional EC analysis techniques often fail to provide the necessary insights into the dynamics of co-evolutionary systems, a key factor affecting performance. In this paper we use two simple fitness landscapes to illustrate the importance of taking a dynamical systems approach to analyzing co-evolutionary algorithms in order to understand them better and to improve their problem solving performance.

Categories and Subject Descriptors: I.2.m [Artificial Intelligence]: Evolutionary Computation

General Terms: Algorithms.

Keywords: cooperative co-evolution, dynamics, landscapes

1. INTRODUCTION

The goal of our research is to better understand the behavior of co-evolutionary systems in order to improve their applicability as a problem solving tool. To achieve this one would like to reuse as much of the existing evolutionary computation (EC) research as possible. However, it turns out that many of the intuitions we have about how simple evolutionary algorithms (EAs) work and how we can improve their performance do not transfer directly to co-evolution. Consequently, there is a need to develop new *rules of thumb* specific to co-evolutionary computation (Co-EC). In this paper we give examples of heuristics concerning improving the performance of regular EAs that do not apply in the same way to cooperative co-evolutionary algorithms (CCEAs). Additionally, we show how traditional EC analysis methods do not explain this lack of portability. We then provide a dynamical systems perspective which sheds light on the results as well as on specific ways of improving co-evolutionary performance.

GECCO'05, June 25–29, 2005, Washington, DC, USA.

2. BACKGROUND

The notion of *cooperative co-evolution* was introduced and made popular by [6], although systems using similar models of evolution pre-dated it, e.g. [3], [4]. Potter's definition of cooperative co-evolution [7] specifies the following characteristics: 1) a species represents a subcomponent of a potential solution; 2) complete solutions are obtained by assembling representative members of each of the species present; and 3) credit assignment at the species level is defined in terms of the fitness of the complete solutions in which the species members participate. While this wording doesn't explicitly talk about populations, Potter's implementation actually keeps each species in a separate population (as opposed to Holland's model, which uses a single population). To further make things concrete, Potter implements the third part of the definition as follows: from one interaction consisting of picking one subcomponent from each species and assembling them into a complete solution, each subcomponent gets the same payoff as the other subcomponents, namely a value giving the quality of the complete solution.

Wiegand [9] extended this work in a number of directions, one of which was to show the relationship between this notion of cooperative co-evolution and symmetric games in evolutionary game theory [2].[1] This provided a natural way to study cooperative co-evolution from a dynamical systems perspective and provided many new insights.[2] As we gain more practical experience in using cooperative co-evolutionary algorithms to solve difficult optimization problems, it is becoming increasingly clear how important this dynamical systems perspective is for the practitioner faced with making a variety of design decisions while building an effective system. In this paper we illustrate this by showing how traditional EC design heuristics fail to provide the necessary insights to improve the performance of a co-evolutionary system, and then showing how understanding the co-evolutionary dynamics improves things dramatically.

3. EXPERIMENTAL SETUP

The general philosophy adopted here is to keep things as simple as possible while still capturing the important phenomena. We begin by defining several simple fitness landscapes that will serve as our experimental task domains.

[1]Wiegand [9] actually suggested replacing the phrase Cooperative Co-evolutionary Algorithms with the more specific terminology of Multi-Population Symmetric Payoff Co-evolutionary Algorithms.
[2]Dynamical systems theory proved useful in analyzing competitive co-evolution as well[1]

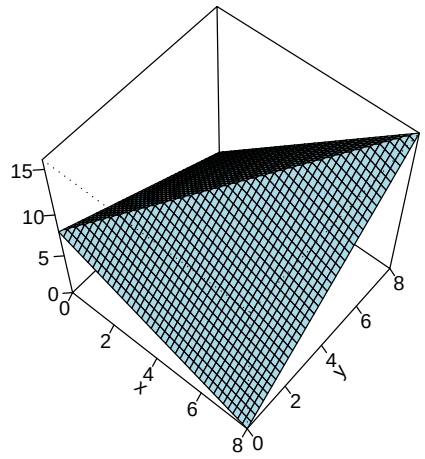

Figure 1: The *oneRidge* function.

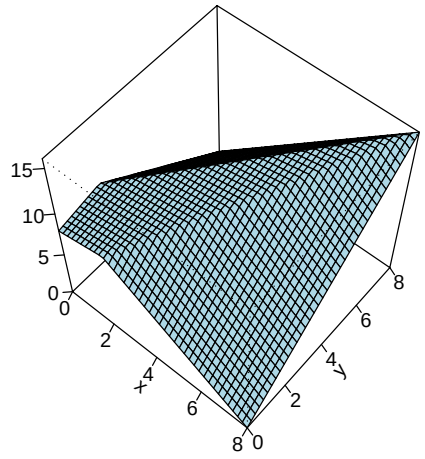

Figure 2: The *twoRidges* function.

The first function used for the experiments reported here was the one pictured in figure 1. Its mathematical expression is given by: $oneRidge_n(x,y) = n+2*min(x,y)-max(x,y)$. It has a single maximum of value $2*n$ at point (n,n) and one ridge going diagonally from $(0,0)$ to (n,n) along which the function value increases from n to the maximum $2*n$. The ridge separates two planar surfaces. The function has two minima of value 0 at $(0,n)$ and $(n,0)$. $n=8$ was used in all experiments.

The second function studied is much like the first, as can be seen in figure 2. It has the same maximum and the same two minima. The difference is that instead of a single ridge it has two ridges symmetrical w.r.t. the diagonal. The function is given by the following formula:

$$twoRidges_n(x,y) = \begin{cases} n + \frac{n-3x+4y}{2} & \text{if } y < \frac{4x-n}{3}; \\ n + \frac{x+y}{2} & \text{if } y < \frac{3x+n}{4}; \\ n + \frac{n+4x-3y}{2} & \text{otherwise.} \end{cases}$$

From a traditional EC perspective, both landscapes are similar with respect to properties such as continuity, modality, ruggedness, etc. But, as we shall see, these landscapes capture important differences for co-evolutionary algorithms.

Our task is to find the maximum of these functions. If we were to use a standard single-population EC approach, we might start out with a non-overlapping generational EA that uses a real-valued representation, binary tournament selection, and a gaussian mutation operator with a fixed sigma of 0.25 changing on average 75% of the genes. We might then tune our EA by experimenting with different population sizes (10, 100 and 1000) and with the inclusion of elitism. Figures 3 and 4 illustrate the typical kinds of insights obtained. In this case, increasing the population size improves performance, and introducing elitism has similar effects on performance without requiring larger population sizes. In the best-of-generation plots each point on a curve represents the mean of best fitness values for the corresponding generation over 100 runs. Every other 10 generations the 95% confidence intervals for the mean are displayed.

The standard way of attacking such problems using cooperative co-evolution is to decompose the problem "naturally" by defining each argument of the function to be a subcomponent (species) to be evolved in a separate population [6]. For our simple example landscapes this results in two populations, one evolving values for the x function parameter and the other evolving values for the y parameter. The simplest method of evaluating an individual in one population is to couple it with the current best member of the other population and the value of the function f at that point is assigned as fitness. This has been termed *single best collaboration strategy* by [9] and is the equivalent of what was termed in competitive co-evolution LEO (last elite opponent) evaluation [8]. It is also a case of symmetric payoff as presented by [9].

The two populations take turns in evolving. During each generation only one population of the two is active. The pseudo-code of the algorithm for one X generation is given below for clarity (the one for Y can easily be inferred):

- evaluate the X population using the current y_{best};
- select parents according to determined fitness values;
- breed;
- evaluate the new population using the same y_{best};
- determine x_{best} according to these new fitnesses.

At the begining of each run, both populations are initialized uniformly random across the domain. Co-evolution starts by evaluating the members of the initial X generation in conjunction with a random y individual. For the first Y generation (second generation of the run) the x_{best} used is the actual best x individual from the first (X) generation.

If we now use the same EA that we used for our single population example for each of the populations in our CCEA[3] and we perform the same tuning experiments as before, we obtain a rather surprisingly different set of results as illustrated in figures 5 and 6.[4] In particular, notice that on the *oneRidge* landscape increasing the population size and introducing elitism resulted in significant decreases in performance! Clearly getting an insight into how CCEAs work is more complex than for standard EC. In particular, one needs to take a closer look at the dynamics of CCEAs.

[3]This means having 10/100/1000 individuals *per* population for a total of 20/200/2000 in the system.

[4]All generations, both X and Y, are represented on the plots, as fitness represents the function value in a particular point, and for optimization purposes we are interested in finding the (x,y) pair with the highest function value.

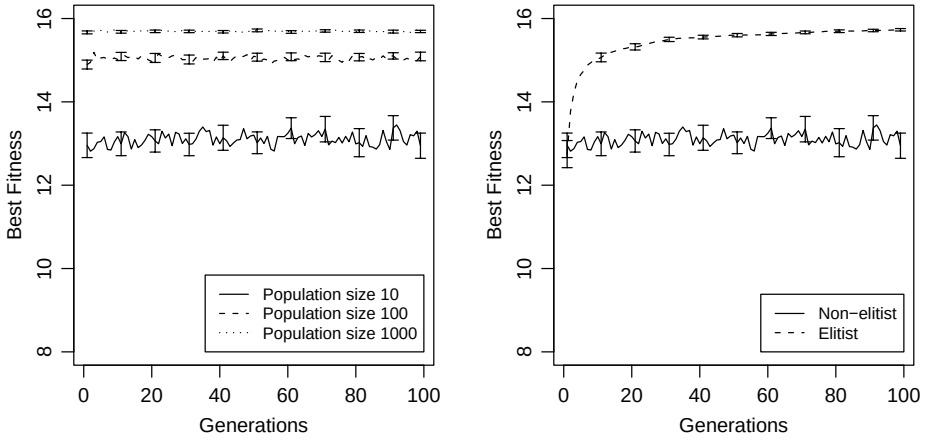

Figure 3: EA on *oneRidge*. Best-of-generation curves. Summary of 100 runs; means with 95% confidence intervals. Left: population size effects without elitism. Right: elitism effects for population size 10.

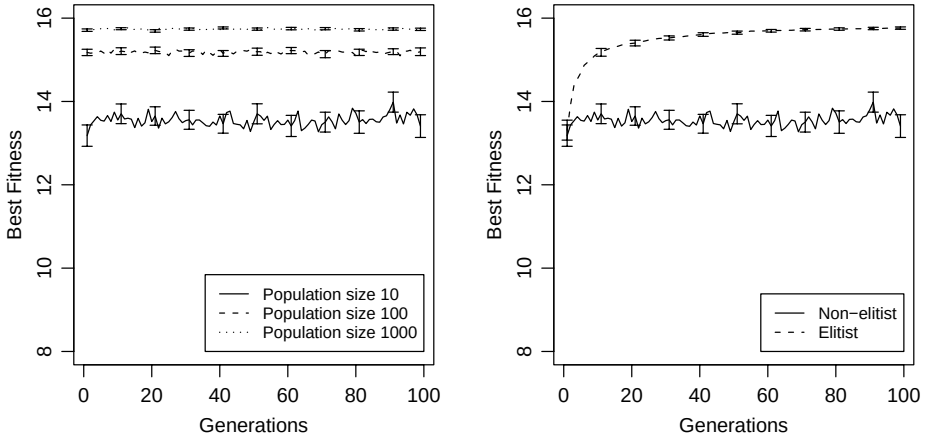

Figure 4: EA on *twoRidges*. Best-of-generation curves. Summary of 100 runs; means with 95% confidence intervals. Left: population size effects without elitism. Right: elitism effects for population size 10.

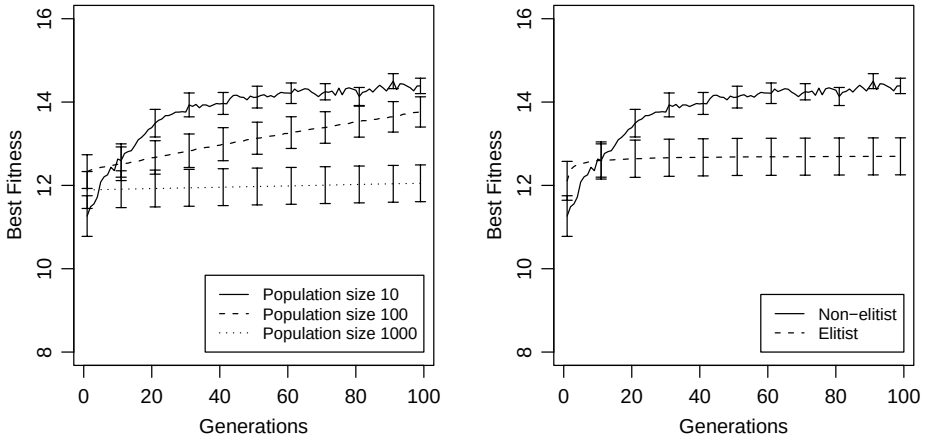

Figure 5: CCEA on *oneRidge*. Best-of-generation curves. Summary of 100 runs; means with 95% confidence intervals. Left: population size effects without elitism. Right: elitism effects for population size 10.

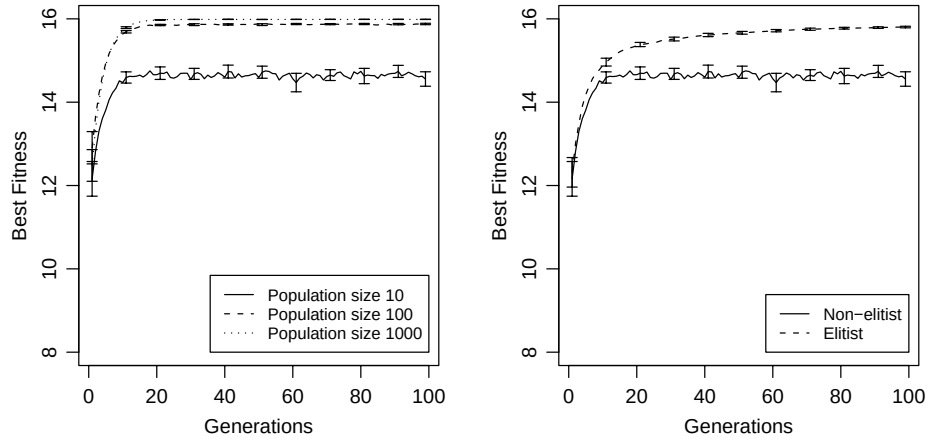

Figure 6: CCEA on *twoRidges*. Best-of-generation curves. Summary of 100 runs; means with 95% confidence intervals. Left: population size effects without elitism. Right: elitism effects for population size 10.

4. CO-EVOLUTIONARY DYNAMICS

In the literature the term *co-evolutionary dynamics* generally refers to population-level dynamics. In this paper we use the term to refer to dynamics of *individuals* rather than population(s). Specifically, we analyze the time trajectories of best-of-generation individuals across the search space.L There are two reasons for this. First of all, we are trying to analyze the performance of cooperative co-evolution for optimization, where the main concern is with the best individuals the algorithm produces. Second, we wanted to understand the behavior of basic CCEAs before moving to more complex ones, and the simplest one out there (that we picked for our initial experiments) uses a *single best collaboration strategy* for evaluation.

In addition to this, the fact that a basic CCEA alternates populations suggests a *line-search*-like way of operation. It therefore makes sense to plot the best individual of one generation (and therefore one population) by coupling it with its collaborator during fitness assessment (here, the best individual of the other population at the previous generation). We thus obtain one point of the search space per generation and we connect these points chronologically. It is obvious that the lines connecting them will only be vertical (when connecting an X generation with the following Y generation) and horizontal (when connecting a Y generation with the following X generation). An example of the result of this procedure for a single run can be seen in figure 7. Each inflection point on the trajectory represents one generation. The starting point is marked by an empty geometrical figure (in this case a circle) and the end point is marked by the same figure but filled.

This visualization technique was first introduced in [5] to analyze the dynamics of a *competitive* co-evolutionary EA. That work also pointed out that the shape of such trajectories is influenced by so-called *best response curves* of the domain. We give an intuitive description of this notion here. The evolution process in one generation within the corresponding population is basically a one dimensional search. The active population is trying to find the best collaborator for the current best individual from the other (static) population. The *bestResponseX* curve is simply obtained by plotting for each y value its best x collaborator.[5] The *bestResponseY* curve can be similarly constructed. For example, on the *oneRidge* function, one can easily see that both the X and the Y best response curves are identical, namely the diagonal $(0,0)-(n,n)$, while they are distinct lines on the *twoRidge* function. We use this methodology of combining trajectory plots and best response curves to provide additional insight into how the parameter tuning presented in the previous section affects problem solving performance.

Figures 8, 9, 10 show the space dynamics of the best individuals on *oneRidge* for population sizes 10, 100 and respectively 1000 and a non-elitist model. Each plot shows 2 or 3 sample runs from the 100 that were carried out for that particular setting. We tried to pick typical samples, while still showing the breadth of behaviors possible *and* keeping the pictures readable. Each run has a different line style and a different geometrical figure to denote the start and end of the run. We also show the best response curve(s).

One can see that for population size 10 the best individuals deviate widely from the best response curve, while for increased sizes they get closer to the best response. In the extreme case, for population size 1000, they end up on the curve. This causes them to no longer move, as each point on the diagonal is both an *X best response* and a *Y best response*. In terms of game theory, one can think of each of these points as a Nash equilibrium[2]. We now understand why increasing population size decreases performance. In order to reach the maximum point of the landscape, the algorithm has to actually avoid being permanently caught on the other points of the ridge. With a small population size, during one generation the evolutionary process is unlikely to come up with the exact best response. Even if it would occasionally do that, the trajectory is unlikely to end up in that point, unless the following generation (for the other population) will provide the exact best response as well. Therefore, the trajectory wanders around close to the diagonal, but doesn't get stuck on it. During this process, it is likely to visit areas close to the optimum, but it

[5]In this paper we only study functions for which the best response is always unique.

can subsequently leave them. Regardless of the initial starting conditions, most runs will do that. This explains the up and down oscillations on the best-of-generation curve for populations of size 10 in the left side of figure 5.

For population size 100, one generation of evolution approximates the best response much better. It is still unlikely that two consecutive generations will produce the exact best response, which would cause the trajectory to get stuck. Instead, the best individuals move slowly upwards, closely following the ridge. This steady but slow progress rate explains the shape of the corresponding best-of-generation curve in figure 5 (basically a constant slope line). Finally, for population size 1000 both populations immediately generate the best response and the trajectory stops almost in the same point where it started, therefore the flatness of its best-of-generation curve. Increasing the population size also increases the dependency of the best at a particular point in time on the initial starting point. Runs that start with low fitness will not reach high fitness levels. This is why the curve for size 1000 is positioned below the one for size 100 which is below the one for size 10. It is also due to this dependency that variance for best-of-generation over all runs increases with population size, and this is reflected in wider confidence intervals for the mean.

What is the explanation for the opposite effects of increasing population size on the *twoRidges* function? Figures 12, 13 and 14 hold the key. For this function the X best response and the Y best response curves do not coincide anymore. They are two different lines of equations $y = \frac{4x-n}{3}$ and $y = \frac{3x+n}{4}$ that intersect in the point of optimum fitness. If the two populations were to provide the exact best response each time (which actually happens for size 1000), the trajectory would do the following: move vertically up until intersecting the Y best response line, then move horizontally to the right until intersecting the X best response line, then up again, to the right again and so on. This kind of motion makes the trajectory converge to the point of maximum fitness located at (n, n). This is exactly what we see in figure 14. Additionally, it only takes a few generations to reach the optimum, therefore, regardless of the starting point, the best-of-run will approximate the optimum with high precision every time. Going back to the left side of figure 6, we now understand why there is a near lack of variance in the corresponding best-of-generation curve as it flattens at optimum fitness. Smaller population sizes will provide coarser approximations to the best responses, and therefore the trajectory will deviate from the two best response lines. This can slow its rate of reaching the optimum and make it harder to precisely approximate it. It is exactly what we see in figures 12 and 13 and they explain the curves in figure 6.

The effect of elitism is somewhat more subtle. Introducing elitism prevents the algorithm from making a move to a best individual that is worse than the previous best. In general, increasing fitness is done by moving closer to the best response curve of the current subcomponent. In the case of the *oneRidge* function, the two curves are one and the same line, so fitness of best individuals can be increased only by getting closer and closer to it. As both populations draw the trajectory towards the diagonal, it lands on a point on this line quite fast and never leaves it, due to its Nash equilibrium properties. This can be seen in figure 11 and explains the corresponding flat best-of-generation curve in the right side of figure 5. This behavior causes once again

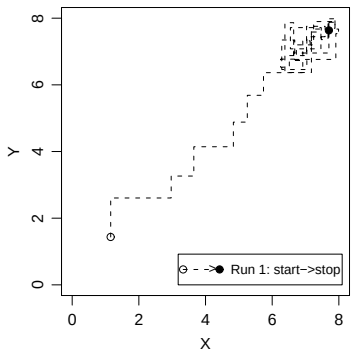

Figure 7: Best individuals' dynamics. OneRidge. Population size 10; no elitism. 1 run.

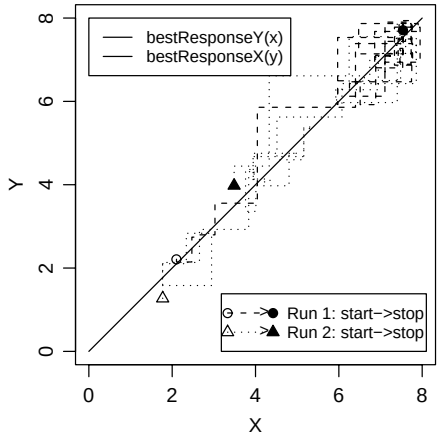

Figure 8: Best individuals' dynamics. OneRidge. Population size 10; no elitism. 2 runs.

the best at a certain generation to strongly depend on the random starting point. The result is a wider variance in performance over multiple runs and a lower average than in the non-elitist case, both observed in figure 5.

In the case of the *twoRidges* function, since the two best response curves are different, elitism does not generate lack of movement, as it draws the trajectory from one curve to the other repeatedly, making it advance towards the optimum (see figure 15). Since elitism doesn't allow for decreasing fitness, the trajectory doesn't take steps back, which makes it reach near-optimum values faster. This explains the trends of the corresponding best-of-generation curves in the right side of figure 6.[6]

[6]Elitism doesn't produce more accurate best responses, so the distribution of best-of-run fitnesses (investigated but omited for brevity) was the same as in the non-elitist case. For all other experiments, whenever best-of-generation curves were visually distinct (i.e. most mean confidence intervals disjunct), the medians of best-of-run distributions over 100 runs were statistically significantly different with 95% confidence and the relationships between them were consistent with the corresponding best-of-generation curves.

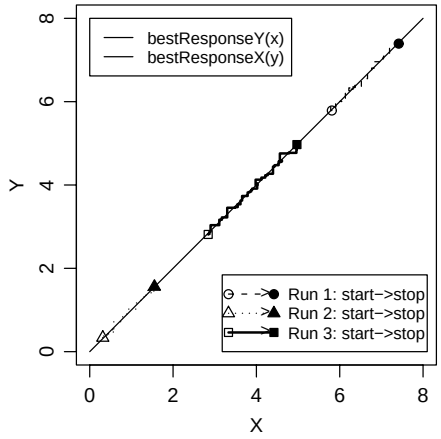

Figure 9: Best individuals' dynamics. OneRidge. Population size 100; no elitism. 3 runs.

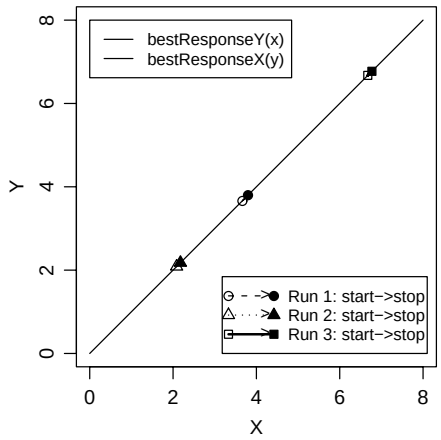

Figure 10: Best individuals' dynamics. OneRidge. Population size 1000; no elitism. 3 runs.

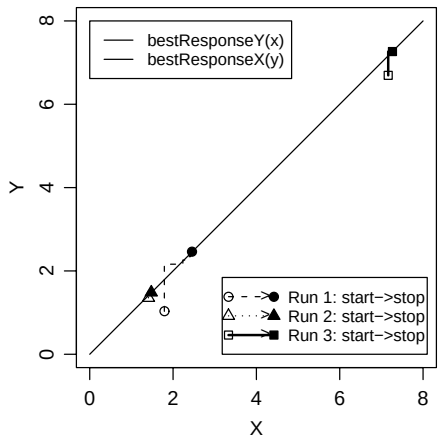

Figure 11: Best individuals' dynamics. OneRidge. Population size 10; elitism. 3 runs.

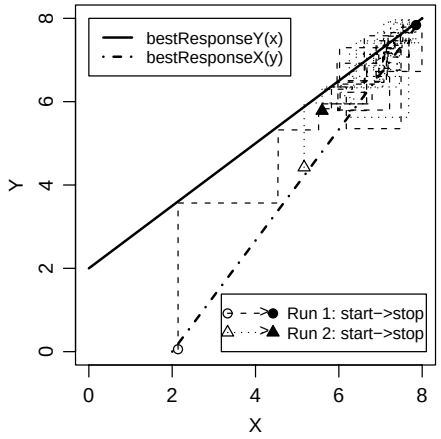

Figure 12: Best individuals' dynamics. TwoRidges. Population size 10; no elitism. 2 runs.

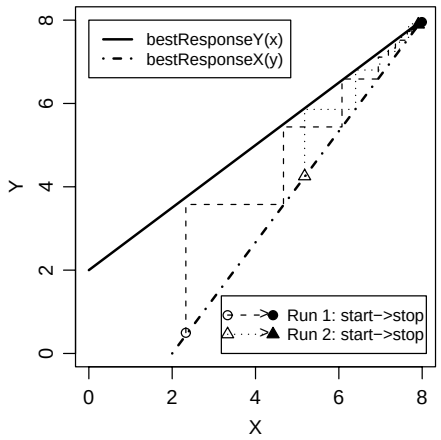

Figure 13: Best individuals' dynamics. TwoRidges. Population size 100; no elitism. 2 runs.

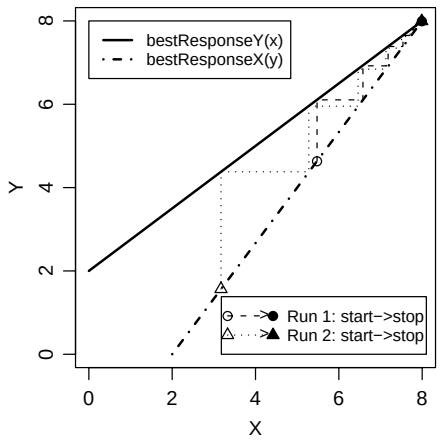

Figure 14: Best individuals' dynamics. TwoRidges. Population size 1000; no elitism. 2 runs.

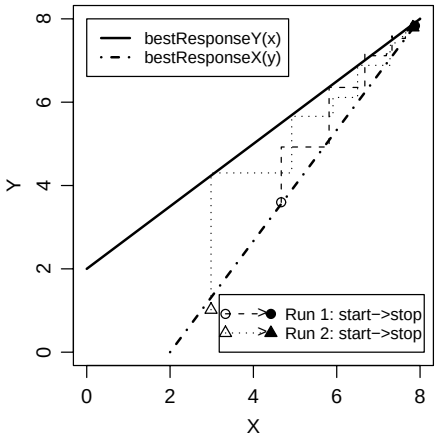

Figure 15: Best individuals' dynamics. TwoRidges. Population size 10; elitism. 2 runs.

5. IMPROVING CCEA PERFORMANCE

The key insight from the previous section is the critical role that the collaboration mechanism plays in determining the dynamics of the co-evolutionary system which in turn has a direct effect on problem solving performance. This observation is not new (see [10]). What is new is the use of a particular type of dynamical systems plots (namely ones depicting space trajectories of best individuals) to provide the insights necessary to make changes that improve overall performance. We illustrate this idea briefly in this section.

An obvious direction one might take is to reduce the one-dimensional line search characteristics of the single best collaboration strategy by requiring multiple collaborations in order to assess the fitness of an individual. This, of course, comes at the cost of an increase in the number of function evaluations required per generation (so one can afford fewer generations within the same time budget). The effectiveness of such a change can be studied both from a performance point of view and from a dynamical systems point of view.

For example, one might define the fitness of an individual to be the maximum of two function evaluations, one involving the best individual from the other population and a second using a randomly chosen one. Figure 16 shows the effects of this change on the co-evolutionary dynamics for both functions. Notice the significant difference in behavior for both landscapes in comparison to the earlier plots (figure 8 for *oneRidge* and figure 12 for *twoRidges*). One characteristic of this new collaboration scheme is that the trajectory no longer has only horizontal and vertical lines, but oblique ones as well, which tend to allow for larger jumps. While these jumps can take steps back sometimes, their main effect is that of pulling the whole trajectory into a smaller area of high fitness in the upper right corner of the domain.

One might also explore the added value, if any, of increasing the number of randomly chosen collaborators. As an example, Figure 17 illustrates the effects of using 4 randomly chosen collaborators. Notice the marginal effect this has on the dynamics.

Figure 18 illustrates how these modest changes in dynamics translate directly into modest performance improvements of the CCEA system.

6. CONCLUSIONS AND FUTURE WORK

In this paper we used two simple fitness landscapes to gain insights into the nature of cooperative co-evolution. We showed that heuristics for improving performance of regular EAs cannot be ported *as-is* to CCEAs. Additionally, we noted that regular EC analysis methods (such as best-of-generation curves) do not provide adequate insight as to why that is the case. However, switching to dynamical systems analysis methods immediately clarified the results. Moreover, we illustrated how to use these same methods with good results to investigate co-evolution specific heuristics for improving performance.

We believe that the key to efficiently using co-evolution (whether cooperative or competitive) as a problem solving tool is understanding its dynamical behavior. We are continuing to conduct further research on this topic. Some natural directions for extending the work presented here include:

- analyzing the frequency of interaction between populations (e.g. ranging from every evaluation to every couple of generations);
- analyzing additional fitness landscapes to expose more co-evolution behaviors;
- improving the dynamical systems visualization methods.

7. REFERENCES

[1] S. Ficici, O. Melnik, and J. Pollack. A game-theoretic investigation of selection methods used in evolutionary algorithms. In e. a. A. Zalzala, editor, *Proc. of CEC2000*. IEEE Press, 2000.

[2] J. Hofbauer and K. Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge University Press, 1998.

[3] J. H. Holland. Properties of the bucket brigade algorithm. In *ICGA*, pages 1–7, Pittsburgh, PA, 1985.

[4] P. Husbands and F. Mill. Simulated co-evolution as the mechanism for emergent planning and scheduling. In *ICGA*, pages 264–270, 1991.

[5] E. Popovici and K. De Jong. Understanding competitive co-evolutionary dynamics via fitness landscapes. In S. Luke, editor, *AAAI Fall Symposium. Artificial Multiagent Learning*. AAAI Press, 2004.

[6] M. Potter and K. De Jong. A cooperative coevolutionary approach to function optimization. In *Proc. of the Third Conference on Parallel Problem Solving from Nature*, pages 249–257, Jerusalem, Israel, 1994. Springer.

[7] M. Potter and K. De Jong. Cooperative coevolution:an architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.

[8] K. Sims. Evolving 3D morphology and behaviour by competition. In R. Brooks and P. Maes, editors, *Artificial Life IV Proc.*, pages 28–39, MIT, Cambridge, MA, USA, 6-8 July 1994. MIT Press.

[9] R. P. Wiegand. *An Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, George Mason University, Fairfax, VA, 2004.

[10] R. P. Wiegand, W. Liles, and K. De Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In L. Spector, editor, *Proc. of GECCO 2001*, pages 1235–1242. Morgan Kaufmann, 2001.

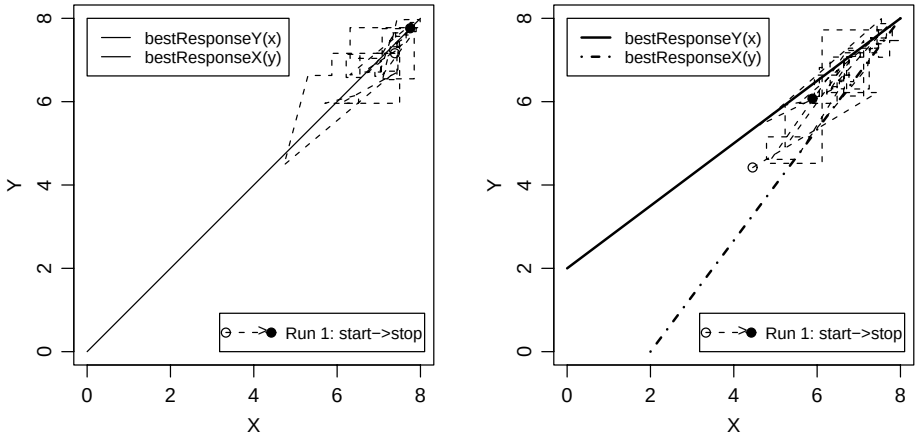

Figure 16: Best individuals' dynamics. Population size 10; no elitism; collaboration: best + 1 random. Left: *oneRidge*. **Right:** *twoRidges*.

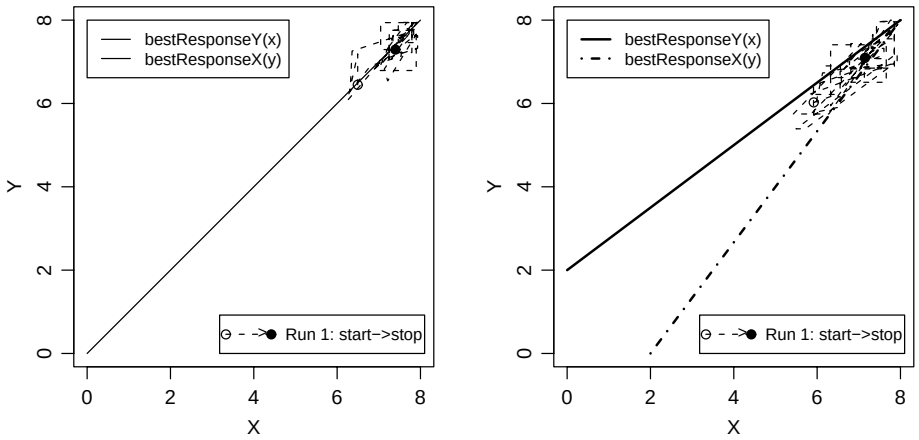

Figure 17: Best individuals' dynamics. Population size 10; no elitism; collaboration: best + 4 random. Left: *oneRidge*. **Right:** *twoRidges*.

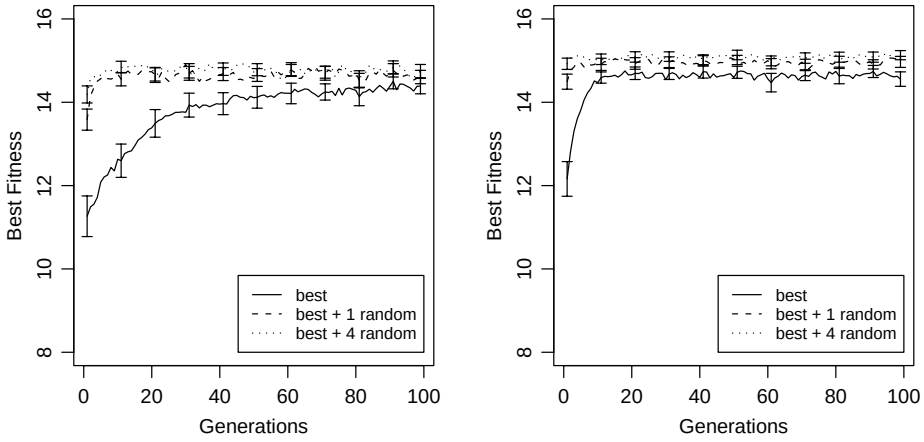

Figure 18: Collaboration effects for best-of-generation. Population size 10; no elitism. Summary of 100 runs; means with 95% confidence intervals. Left: *oneRidge*. **Right:** *twoRidges*.