A Dynamical Systems Analysis of Collaboration Methods in Cooperative Co-evolution.

Elena Popovici and **Kenneth De Jong**
George Mason University
Fairfax, VA
epopovic@gmu.edu
kdejong@gmu.edu

Abstract

Cooperative co-evolution is often used to solve difficult optimization problems by means of problem decomposition. In order to do so efficiently, one must have a good understanding of co-evolution's dynamical behavior. To build such understanding, we have constructed a methodology for analyzing co-evolution based on dynamical systems theory. In this paper we show how it can be applied to investigate the effects that collaboration methods have on performance and to identify a problem property relevant in this respect.

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. In this paper we focus on cooperative co-evolutionary algorithms (CCEAs) as a method of static function optimization (Potter & De Jong 2000). Their usage as such has been closely investigated ((Potter 1997), (Wiegand 2004)) in order to uncover their strengths and weaknesses.

This research pointed out that the way the problem was decomposed and the pieces assigned to different populations was a key issue for performance. However, the nature of the relationship between problem decomposition and problem difficulty was not a trivial one to investigate. Previous research on this topic ((Wiegand, Liles, & De Jong 2001), (Wiegand, Liles, & De Jong 2002)) was mainly in the context of analyzing the effects on performance of the inter-population collaboration mechanism used for evaluation. It suggested that these two aspects (problem decomposition and collaboration mechanism) are tightly coupled.

It is easy to see that when the function is linearly separable and the decomposition of the function matches its separability, using the simple single-best collaboration method is enough for tackling the problem. What is not clear is when would one need more complex schemes for evaluation. (Wiegand, Liles, & De Jong 2001) shows that the simple presence of non-linear interactions between the pieces of the problem represented in different populations is not enough to necessitate more complex evaluation schemes.

(Wiegand, Liles, & De Jong 2002) and (Wiegand 2004) take the analysis further to see what type of such cross-population epistasis can really cause problems for the single-best collaboration method. In the context of a study on pseudo-boolean functions, the hypothesis offered is that to blame is the *contradictory cross-population epistasis*, a situation in which "the linear effects of lower order components existing in different populations are opposite in magnitude and direction of the higher order blocks" and thus "the algorithm is tricked into believing that the lower order components accurately predict the total function value, when in fact they are misleading".

The analysis follows an epistasis-focused approach. Problems with and without contradictory cross-population epistasis are constructed and the results of running a CCEA with varied collaboration mechanisms are recorded. The observed performance agrees with the hypothesis. It seemed like this was the end of the story. In this paper we show that is not the case.

We introduce two functions for which the natural decomposition introduces contradictory cross-population epistasis, yet the effects on performance of the single-best collaboration method are quite opposite. What other property differed in these functions to generate such results? To answer this question, we used methods inspired from dynamical systems theory to investigate the run-time behavior of the algorithm and how it reflects on performance.

From the analysis we infer a different problem property to be responsible for the effects on the optimization performance of the single-best collaboration method. We use this knowledge to predict the results on a set of functions picked from the literature and show that the experiments agree with the prediction.

A Counter-example

We start by showing that contradictory cross-population epistasis is not necessarily a good indicator of difficulty. We do this by presenting two functions that both exhibit this kind of epistasis for the natural decomposition (one piece per function parameter), however for one of them the single-best collaboration method is all there is needed to solve the problem, whereas for the other this method does poorly and can be significantly improved upon.

These functions were initially introduced in (Popovici &

De Jong 2005) and we reproduce them here for completeness.

The first function used for the experiments reported here was the one pictured on the left side of 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 the right side of figure 1. 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:

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

To see the contradictory interaction of the X and Y components for the first function, consider any point along the diagonal ridge. Changing the x value while keeping the y constant will result in a decrease in function value, regardless of whether x is increased or decreased. Similarly, changing the y value while keeping the x constant will always result in a decrease in function value. However, if both x and y are changed together, in some cases the function value will increase (e.g. moving *up* along the diagonal) and in other cases it will decrease (e.g. moving *down* along the diagonal). Here's a concrete example:

$$\left.\begin{array}{l} oR(5,4) < oR(4,4) \\ oR(4,5) < oR(4,4) \end{array}\right\} \quad oR(5,5) > oR(4,4) \quad \text{and}$$

$$\left.\begin{array}{l} oR(3,4) < oR(4,4) \\ oR(4,3) < oR(4,4) \end{array}\right\} \quad oR(3,3) < oR(4,4) \ ,$$

where $oR = oneRidge$.

There are also cases when both the change in x and the change in y independently have the effect of increasing the function value, and such changes combined can either increase it or decrease it. Take for example the point $(4, 3)$. We have both following situations:

$$\left.\begin{array}{l} oR(3,3) > oR(4,3) \\ oR(4,4.5) > oR(4,3) \end{array}\right\} \quad oR(3,4.5) < oR(4,3) \quad \text{and}$$

$$\left.\begin{array}{l} oR(3.5,3) > oR(4,3) \\ oR(4,3.5) > oR(4,3) \end{array}\right\} \quad oR(3.5,3.5) > oR(4,3)$$

An infinite number of points with such behaviors exist. The same is true for $twoRidges$. As an example,

$$\left.\begin{array}{l} tR(5.5,6) > tR(7,6) \\ tR(7,6.5) > tR(7,6) \end{array}\right\} \quad tR(5.5,6.5) < tR(7,6) \quad \text{and}$$

$$\left.\begin{array}{l} tR(6.5,6) > tR(7,6) \\ tR(7,6.5) > tR(7,6) \end{array}\right\} \quad tR(6.5,6.5) > tR(7,6) \ ,$$

where $tR = twoRidges$. Additionally,

$$\left.\begin{array}{l} tR(7,6) < tR(5.5,6) \\ tR(5.5,6.5) < tR(5.5,6) \end{array}\right\} \quad tR(7,6.5) > tR(5.5,6)$$

and

$$\left.\begin{array}{l} tR(7,4) < tR(6,4) \\ tR(6,3) < tR(6,4) \end{array}\right\} \quad tR(7,3) < tR(6,4) \ .$$

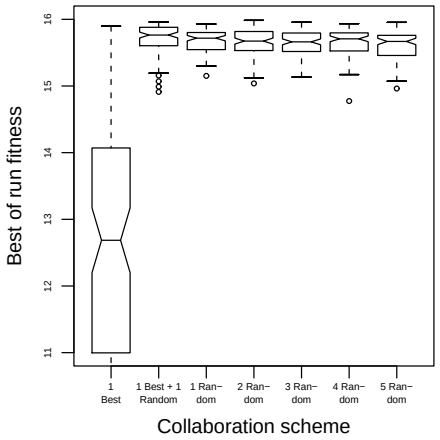

Figure 2: Best of run statistics for the different collaboration methods on the $oneRidge$ function. Maximization problem – bigger is better.

An infinite number of points with such behaviors exist for this function as well.

We ran a two-population CCEA on both problems, with one population evolving values for x and the other for y and the goal of maximizing the functions. In each case we experimented with 7 collaboration methods: single-best, single-best plus one random, one random, two, three, four and five random collaborators. These were selected per individual rather than per generation. The other settings of the algorithm were common ones and are detailed in section . The results are shown in figures 2 and 3. These plot the distribution over 100 runs of the best-of-run fitness. As these were maximization problems, bigger values are better.

Clearly, the single-best collaboration method has quite opposite performance on the two functions. For $oneRidge$, it is largely outperformed by all other methods. For $twoRidges$, all methods are in the same ballpark, with the single-best method however giving slightly better results then all other methods (its mean is statistically significantly larger then all other means with 95% confidence).

For two problems with the same type of contradictory cross-population epistasis, the single-best collaboration method is in one case the best method and in the other case the worst method. How do these functions differ to generate such behaviors for the CCEA? Our approach to answer this question is to look inside the black box of the algorithm and analyze its run-time dynamics.

Dynamics Analysis

Dynamical systems theory proved a good source of inspiration for constructing methods of analysis for co-evolutionary algorithms. In the literature the term *co-evolutionary dynamics* generally refers to population-level dynamics. For cooperative settings, (Wiegand 2004) introduced a technique for tracking the percentage of the best individual in the population. For a particular type of competitive settings

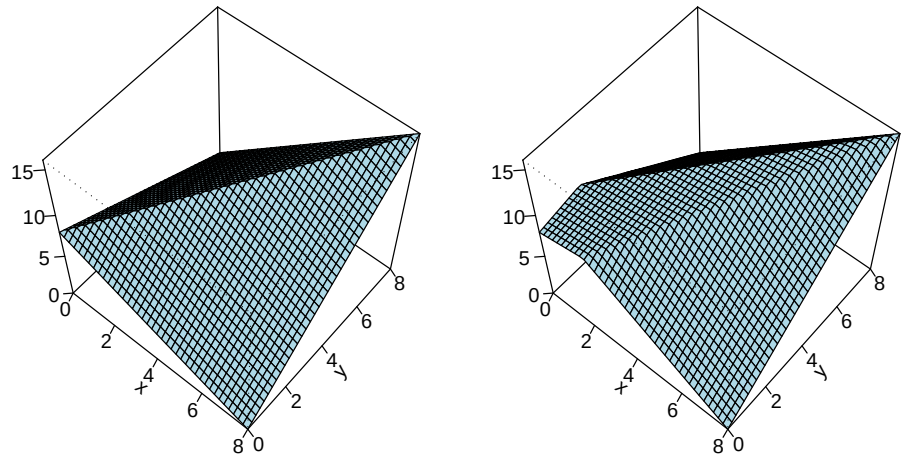

Figure 1: Left: The $oneRidge$ function. Right: The $twoRidges$ function.

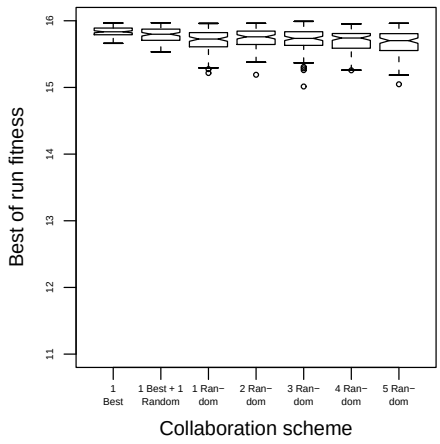

Figure 3: Best of run statistics for the different collaboration methods on the $twoRidges$ function. Maximization problem – bigger is better.

(namely, frequency based evolution), (Ficici, Melnik, & Pollack 2000) analyzes trajectories of the population state.

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. We employ a technique that has already been used in (Popovici & De Jong 2004) and (Popovici & De Jong 2005) to gain insight into the way co-evolutionary algorithms work, whether cooperative or competitive.

Before moving on to the analysis, we describe the details of the algorithm under investigation. We employed a two-population CCEA. Each population used a real-valued representation, a non-overlapping generational model except for elitism of one, binary tournament selection, and gaussian mutation with a fixed sigma of 0.25 operating at a rate of 75%. The population size was 10. When more than one collaborator was used for evaluation, the fitness was chosen to be the best of the obtained values. For the cases with more than one random individual, their selection was done with replacement. Populations took turns evolving, i.e. only one was active per generation (in other words, the update timing was sequential (Wiegand 2004)). The stopping criteria for all experiments was reaching 1000 function evaluations, therefore using more collaborators per individual meant fewer generations.

The $oneRidge$ function.

Figures 4, 5, 6 and 7 show typical runs[1] for the $oneRidge$ function for collaboration methods single-best, single-best plus one random, one random and five random. Two, three and four random are omitted for brevity and their features are basically an interpolation between one and five.

The plots are displaying one point per generation, marked with a black dot, and the points are connected chronologically by grey lines. Each point represents the best individual

[1]We generated plots for all 100 runs and visually investigated them. Figure 4 displays two runs, all other plots display a single run.

of that generation (and corresponding population) coupled with the collaborator from the other population to which it owes its fitness. The beginning of the run is marked with a circle and the end with a bullet[2]. As the total number of evaluations per run was kept constant, more collaborators per individual meant fewer generations, and this is reflected in the varying number of black dots in the plots.

The plots also display the so-called *best response curves*, a notion that was introduced in (Popovici & De Jong 2004). These are properties of the problem, independent of the algorithm. The $bestResponseX$ curve is obtained by plotting for each y the x that produces the best function value in combination with that particular y. $bestResponseY$ is similarly obtained as a function of x. For $oneRidges$, the two best response curves are one and the same, namely the main diagonal. This is displayed the figures by two thick superimposed lines of different color and pattern. Along them, the function value monotonically increases from 8 in $(0, 0)$ to 16 in $(8, 8)$. In this paper we deal only with functions for which the best responses are unique (i.e. for each x there is a single y that optimizes the function value, and similarly for y).

As one can see from all four plots, the dynamics of best individuals are strongly influenced by these curves. For the single-best collaboration method (figure 4), the trajectory evidently can move only horizontally and vertically. All points on the diagonal are equilibria points and once such a point is reached, the algorithm is stuck there. This happens fairly quickly after start-up and on a diagonal point close to the starting location. Unless the algorithm starts close to the global optimum, it will not get close to it. The best of a run is thus highly dependent on its start-up. This explains the high variance in performance over multiple runs and the generally poor results of the single-best collaboration strategy, as shown by the corresponding boxplot in figure 2.

Once a random collaborator is used, the trajectory of best individuals is no longer forced to move only horizontally and vertically, but it can take any direction. Additionally, elitism no longer guarantees ever increasing fitness. This means that the end of the run is not necessarily the best of the run as well. To figure out how close a run got to the global optimum, we need to track the inflection points of the trajectory, marked by small black dots.

Adding a random collaborator to the single best one has the effect that the trajectory can (and does) take big jumps towards areas of high fitness (upper right corner), regardless of where it started. The usage of the single best still keeps the trajectory fairly close to the diagonal corresponding to the best response curves (see figure 5). This results in big performance improvements compared to using only the single best as a collaborator.

While the best-of-run statistics showed that using a single random collaborator for each individual gives roughly the same performance as using both a random and the best, it gave us no clue as to why that is, and one might assume that the two variants of the algorithm behave the same internally.

[2]In figure 4 the trajectory for the second run starts with an empty triangle and ends with a filled triangle

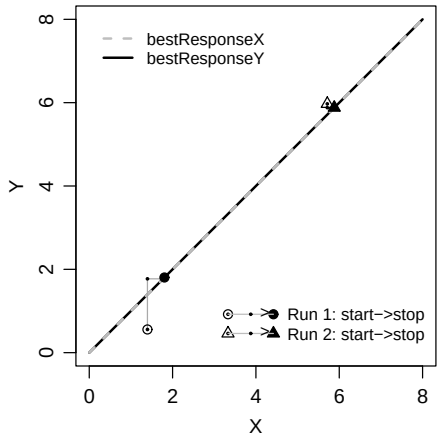

Figure 4: $oneRidge$ - Best-of-generation trajectories for the *single-best* collaboration method. Two runs.

By comparing figures 5 and 6 we can see that this is not the case. The algorithm using a single random collaborator per individual is a lot more explorative and it can visit areas of quite low fitness, far away from the diagonal or towards its lower-left end. However, in this wandering around, it inevitably hits points very close to the global optimum, thus giving good best-of-run statistics.

Using five random collaborators generates dynamics similar to those obtained when using a single random and the single best (compare figures 5 and 7). This is no surprise, as with a population size of 10, picking 5 random individuals has good chances of hitting the best.

Increasing the number of random collaborators from one to five has the effect of making the algorithm less explorative and more exploitative. The trajectory moves closer to the upper-right corner (good, higher function values) and closer to the diagonal (bad, danger of getting stuck). Combined, these two directions average out and best-of-run statistics are not statistically distinguishable.

The $twoRidges$ function.

Performing the same type of analysis for the $twoRidges$ function illuminates us why in this case the single-best collaboration method has the best performance rather than the worst, with respect to best-of-run statistics. For this function, the $bestResponseX$ and $bestResponseY$ curves differ and they are shown in figures 8 through 11 with thick lines, dashed grey and continuous black, respectively. The two best response curves intersect in the point that is the global optimum of the function.

We expect that in the case of the single-best collaboration method the trajectory will alternate vertical steps towards the $bestResponseY$ curves with horizontal steps towards the $bestResponseX$ curve. Indeed, this is what we see in figure 8 and it causes the trajectory to climb like on a ladder towards the global optimum.

Introducing randomness in the collaboration mechanism

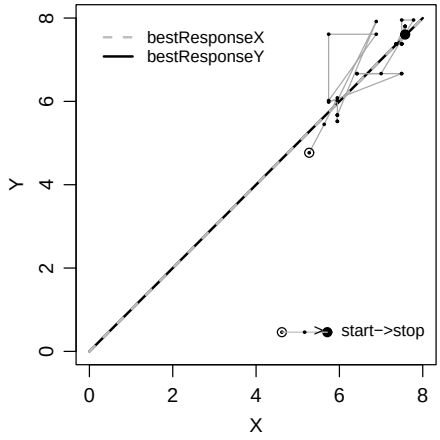

Figure 5: $oneRidge$ - Best-of-generation trajectories for the *single-best plus one random* collaboration method. One run.

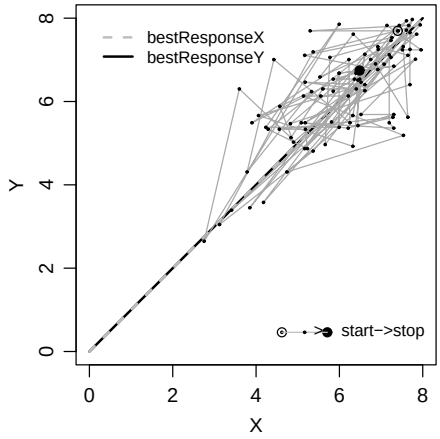

Figure 6: $oneRidge$ - Best-of-generation trajectories for the *one random* collaboration method. One run.

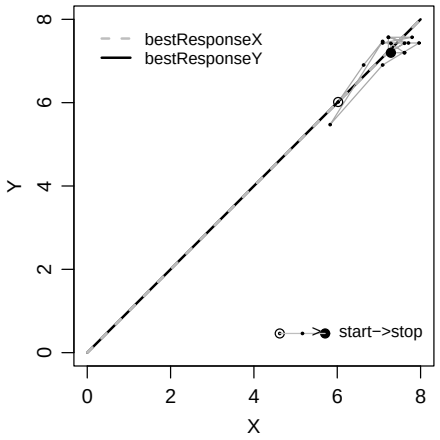

Figure 7: $oneRidge$ - Best-of-generation trajectories for the *five random* collaboration method. One run.

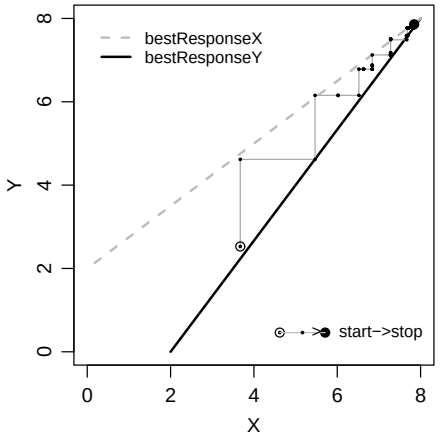

Figure 8: $twoRidges$ - Best-of-generation trajectories for the *single-best* collaboration method. One run.

causes the algorithm to move away from this focus towards the global optimum. In the case of the single-best plus single-random method (figure 9), the algorithm can take bigger steps toward high function values at the beginning, but fails to take the small steps required at the end to get really close to the optimum.

Using a single random collaborator (figure 10) generates again a lot of exploration at the cost of almost no exploitation. For this function however, due to the nature of its best response curves, exploitation alone is a guaranteed way of reaching the optimum, while exploration alone will get there only by luck. Increasing the number of random collaborators (figure 11) reduces exploration, but fails to increase exploitation to the level of using the single-best.

The analysis of the dynamics thus explained why we observed the results in figures 2 and 3 and it provided additional understanding of how the collaboration methods

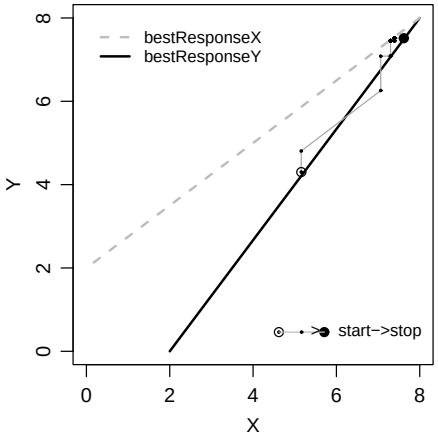

Figure 9: $twoRidges$ - Best-of-generation trajectories for the *single-best plus one random* collaboration method. One run.

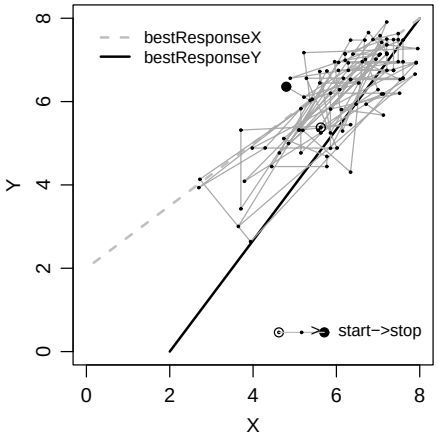

Figure 10: $twoRidges$ - Best-of-generation trajectories for the *one random* collaboration method. One run.

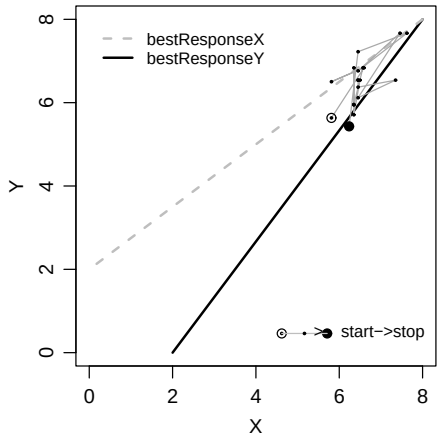

Figure 11: $twoRidges$ - Best-of-generation trajectories for the *five random* collaboration method. One run.

under investigation work. Specifically, the best response curves, which are a property of the problem, have a strong impact on the dynamics of the algorithm. Different collaboration methods interact with this property in different ways, affecting optimization performance.

In particular, the overlapping of the best response curves in more than one point is bound to cause problems for the single-best scheme, as it makes the algorithm get stuck quickly in one of these points. Which of the points will attract the trajectory largely depends on the initial configuration, rather than the function value at those points. Thus, for functions that have different values across these overlapping areas, the single-best collaboration scheme will exhibit poor performance at finding the optimum.

Predictive Power

The key insight from the previous section is that one needs to take a close look at the run-time dynamics of a coevolutionary system to understand the effects that certain parameters of the system (in this case the collaboration mechanism)[3] have on its problem solving performance. Additionally, we have identified some problem features (namely the best response curves) that have a direct influence on the behavior of the algorithms.

In this section we show how the knowledge gained from this analysis of dynamics can be used to predict effects of collaboration methods on new problem domains.

We use for that two functions from the literature that were previously analyzed with respect to collaboration methods in (Wiegand, Liles, & De Jong 2001), namely $rosenbrock$ and $offAxisQuadratic$. We reproduce them below for completeness:

$$rosenbrock(x, y) = 100(x^2 - y^2)^2 + (1 - x)^2,$$
$$x, y \in [-2.048, 2.048];$$

[3] In (Popovici & De Jong 2005) the same was shown for the population size.

$$offAxisQuadratic(x, y) = x^2 + (x + y)^2,$$
$$x, y \in [-65.536, 65.536].$$

The task for the cooperative co-evolutionary algorithm will be to minimize these functions.

We start by determining the best response curves for these functions. For the $rosenbrock$ they look like in figure 12 and they overlap along the curve $y = x^2, x \in (0, \sqrt{2.048})$. On this curve lies the point where the function reaches its minimum/optimum, $(1, 1)$. Along the curve the function value decreases from 1 in $(0, 0)$ to 0 in $(1, 1)$ and then increases to 0.185833 in $(\sqrt{2.048}, 2.048)$.

Our CCEA using the single-best collaboration method will soon after start-up get stuck somewhere on this area of overlapping, the position depending on the start-up position. It will get close to the optimum only by chance (if it starts close to it), therefore we predict poor performance for this case. Introducing a random collaborator is likely to increase exploration and visit several points along the overlapping curve, thus increasing the chances of getting closer to the optimum. We therefore expect these methods to give better results than the single-best, in a manner similar to the $oneRidge$ function.

Indeed, by running our CCEA with the same 7 collaboration methods as before, we get the results plotted in figure 14, which confirm our expectations. Note that as these were minimization problems, smaller values are better.

For the $offAxisQuadratic$ function, the best response curves look like in figure 13. They intersect in a single point, $(0, 0)$, and this is where the function reaches its minimum. The trajectory of best individuals for our CCEA using the single-best collaboration method will alternately move horizontally towards the $bestResponseX$ curve and vertically towards the $bestResponseY$ curve. This will direct the algorithm smoothly towards the point of intersection of the best response curves, thus towards the global optimum. Introducing random collaborators can only disrupt this focus. Exploitation alone will solve the problem, exploration is not needed and in fact can be harmful. We predict the results will be similar to the ones obtained for the $twoRidges$ function, and figure 15 confirms this.

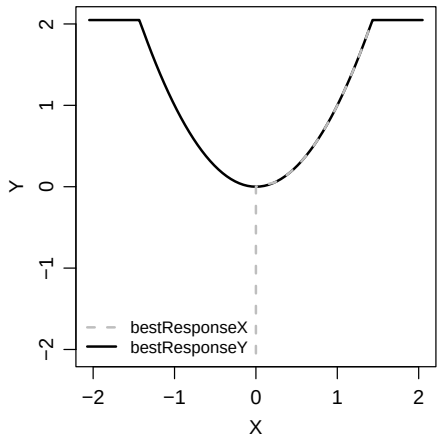

Figure 12: Best response curves for the $rosenbrock$ function.

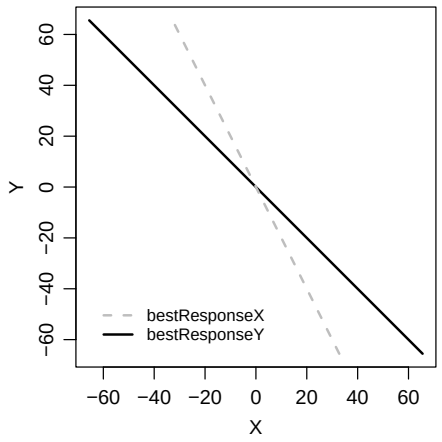

Figure 13: Best response curves for the $offAxisQuadratic$ function.

Discussion and Conclusions

To efficiently use co-evolution (whether cooperative or competitive) as a problem solving tool, one must understand the effect that the combination of problem properties and algorithm properties has on the system's performance. The previous body of co-evolution literature has shown that such systems can display quite complex phenomena. Therefore, we believe there are benefits to taking a *divide-and-conquer* approach to analyzing the dependency problem properties + algorithm properties → system performance. Namely, we split it into two more basic ones that should be easier to analyze: problem properties + algorithm properties → run-time system properties and run-time system properties → system performance.

In this paper we show how the use of this approach provides new insights into the way the collaboration scheme used by the algorithm affects optimization performance.

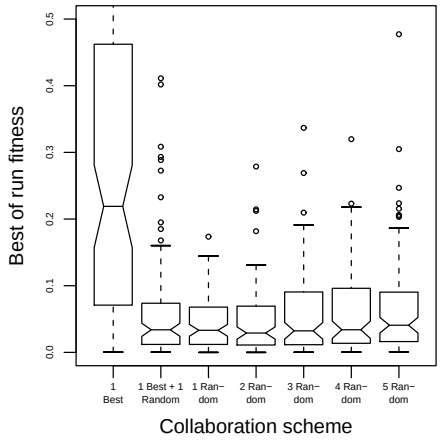

Figure 14: Best of run statistics for the different collaboration methods on the $rosenbrock$ function. Minimization problem – smaller is better.

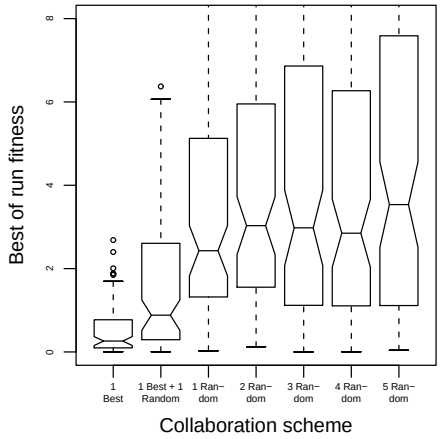

Figure 15: Best of run statistics for the various collaboration methods on the $offAxisQuadratic$ function. Minimization problem – smaller is better.

Namely, the collaboration scheme (an algorithm property) in combination with the nature of the best-response curves (a problem property) determine the type of best-of-generation trajectories (a run-time property), which in turn determine optimization power (system performance).

This paper is not the end of the story either, but rather a beginning. We intend to further use the methodology for understanding effects of various algorithm parameters and for discovering important problem properties. For example, here we studied the effects of collaboration methods in isolation from other algorithm parameters, and when varying several of them at the same time the dynamics may change. In (Popovici & De Jong 2005), the effects of varying the population size were studied in isolation. An immediate next step could be to investigate the collective impact of the collaboration method and the population size on performance. Such knowledge can then be used to build heuristics on how to tune the algorithms to the problems.

The functions used in these experiments were all continuous. Additionally, with the exception of *offAxisQuadratic*, which had a large number of local optima, the functions were very smooth. We believe that the influence of the best-responses transfers to more rugged domains, although larger population sizes may be required to observe the same kind of results.

This paper exposed one type of predictive analysis: infer the effects of a certain algorithm when the problem properties can be identified. However, for more complex problems, it may be difficult to identify their properties. We believe the techniques introduced in this paper may still be of use in such cases. The best-of-generation trajectories may be observed at run-time and the heuristics constructed from previous knowledge be applied to infer what are the properties of the problem and then accordingly tune the algorithm.

References

Ficici, S. G.; Melnik, O.; and Pollack, J. B. 2000. A game theoretic investigation of selection methods used in evolutionary algorithms. In *Genetic and Evolutionary Conference*.

Popovici, E., and De Jong, K. 2004. Understanding competitive co-evolutionary dynamics via fitness landscapes. In Luke, S., ed., *AAAI Fall Symposium on Artificial Multi-agent Learning*. AAAI Press.

Popovici, E., and De Jong, K. 2005. Understanding cooperative co-evolutionary dynamics via simple fitness landscapes. In *Genetic and Evolutionary Computation Conference*. to appear.

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

Potter, M. 1997. *The Design and Analysis of a Computational Model of Cooperative Coevolution*. Ph.D. Dissertation, George Mason University, Computer Science Department.

Wiegand, R. P.; Liles, W.; and De Jong, K. 2001. An empirical analysis of collaboration methods in cooperative

coevolutionary algorithms. In Spector, L., ed., *Proceedings of GECCO 2001*, 1235–1242. Morgan Kaufmann. Errata available at http://www.tesseract.org/paul/papers/gecco01-cca-errata.pdf.

Wiegand, R. P.; Liles, W. C.; and De Jong, K. A. 2002. The effects of representational bias on collaboration methods in cooperative coevolution. In *Proceedings of the Seventh Conference on Parallel Problem Solving from Nature*, 257–268. Springer.

Wiegand, R. P. 2004. *An Analysis of Cooperative Coevolutionary Algorithms*. Ph.D. Dissertation, George Mason University, Fairfax, VA.