Time-dependent Collaboration Schemes for Cooperative Coevolutionary Algorithms

Liviu Panait and **Sean Luke**
Department of Computer Science, George Mason University
4400 University Drive MSN 4A5, Fairfax, VA 22030, USA
{lpanait, sean}@cs.gmu.edu

Abstract

Cooperative coevolutionary algorithms represent a popular approach to learning via problem decomposition. Since they were proposed more than a decade ago, their properties have been studied both formally and empirically. One important aspect of cooperative coevolutionary algorithms concerns how to select collaborators for computing the fitness of individuals in different populations. We argue that using a fixed number of collaborators during the entire search may be suboptimal. We experiment with a simple ad-hoc collaboration scheme that varies the numbers of collaborators over time. Empirical comparisons in a series of problem domains indicate that decreasing the numbers of collaborators over time fares better than fixed collaboration schemes. We conclude with a brief discussion of our findings and suggest directions for future research.

Introduction

Coevolutionary algorithms (CEAs) are popular augmentations of traditional evolutionary algorithms (EAs). The basic elements of these augmentations lay in the adaptive nature of fitness evaluation in coevolutionary systems: individuals are assigned fitness values based on direct interactions with other individuals. Traditionally, CEAs may be either competitive or cooperative. In *competitive* coevolutionary algorithms, individuals compete against one another during fitness assessment — an individual's increase in fitness usually results in a decrease in the fitness of other individuals. In *cooperative* coevolutionary algorithms (CCEAs), individuals are rewarded when they perform well together, and punished when they perform poorly (Bull, 1997; Husbands and Mill, 1991; Potter, 1997). Applications of these methods include optimization of inventory control systems (Eriksson and Olsson, 1997), learning constructive neural networks (Potter and De Jong, 2000), and rule learning (Potter and De Jong, 1998; Potter et al., 1995).

A standard approach (Potter, 1997) to applying cooperative coevolutionary algorithms (or CCEAs) starts by identifying a static decomposition of the problem representation into components, each represented by a separate population of individuals. CCEAs do not evaluate components of the entire solution in isolation from each other, but rather only the quality of a complete solution is assessed. The fitness of an individual in a population is determined by testing that individual in combination with individuals from the other populations (in order to assemble one or more complete solutions that can be evaluated). Aside from this collaborative assessment, each population follows its own independent evolution process in parallel with other populations.

An example may serve to clarify how cooperative coevolutionary algorithms work. Suppose we are optimizing a three argument function $f(x,y,z)$. One might assign individuals in the first population to represent the x argument, individuals in the second population to represent y, and in the third to represent z. Each population is evolved separately, except that when evaluating an individual in some population (e.g., x), collaborating individuals are chosen from the other populations (y and z) in order to obtain an objective function value with a complete solution, $f(x,y,z)$. The fitness of x might simply be assessed as $f(x,y,z)$, where $<y,z>$ is a tuple containing one individual form each of the other populations (we refer to this tuple as a *collaborator*). Alternatively, two collaborators $<y_1,z_1)$ and $<y_2,z_2>$ may be used to assess the fitness of x as the average of $f(x,y_1,z_1)$ and $f(x,y_2,z_2)$. The collaborators may be either chosen at random, or as containing the best individuals in the populations at the previous generation, or using some other selection procedure. Additionally, the fitness of an individual may be aggregated using average, maximum or minimum of performance with multiple collaborators. We assume that learning in the populations is performed concurrently. That is, all populations advance to the next generation at the same time.

The effects of different settings of cooperative coevolutionary algorithms onto the performance of the search have been the concern of several research studies such as Bull (1997); Wiegand et al. (2001). The results indicate that using the maximum to aggregate the fitness of individuals over multiple evaluations performs significantly better than using the minimum or the average. Also, using multiple collaborators might fare better in domains involving complex non-linear interactions among the components, but it also increases the computational requirements. Wiegand (2003) argues that the sampling of collaborators introduces a certain bias on the search: wider peaks (see Figure 1) are more likely to have collaborators sampled from them, and this may bias the search toward suboptimal solutions; this is termed *relative overgeneralization*. Relative overgeneralization may shift the focus of the search from optimality to robustness — depending on the problem, one criteria may be more important than the other. The basins of attraction for different equilibrium points are visualized in (Panait et al., 2004); this provides a graphical illustration of the impact of multiple collaborators onto the learning process.

In this paper, we focus on the number of evaluations used to assess the fitness of an individual. Typically, a coevolutionary learning process is allowed a fixed computational budget, which usually translates into a given total number of evaluations (computing the fitness of an individual may require multiple evaluations). We argue that a fixed number of collaborators may not be optimal in such a case: if too few collaborators are used, the populations may not identify the better areas of the joint space, while too many collaborators might waste evaluations in later stages of the search. As a consequence, we suggest that coevolutionary algorithms may benefit from a time-dependent allocation of evaluations: early generations might take advantage of diversities in the other populations to better sample the joint search space and identify promising areas, while fewer evaluations may be desirable as the populations start to converge. This is reminiscent of our previous research and recommendations for genetic programming (Luke et al., 2003).

Next, we illustrate the perspective of the search space as one population might get when in combination with various other populations. Based on these observations, we propose a simple scheme that employs a time-dependent allocation of evaluations; the preliminary results indicate that no fixed allocation of evaluations outperforms it.

Single-Agent Perspectives on the Search Space

Imagine a simple two-dimensional learning task as the one presented in Figure 1. Based on the values for the x and y components, the curve shows the performance of the overall solution ($f(x,y)$ is plotted along the z axis). The figure shows two peaks of different sizes. The lower peak represents a locally-optimal but globally-suboptimal solution; the wide coverage of the peak implies small changes in solution quality when any of the components is slightly altered. The higher peak represents the globally optimal solution; its smaller coverage implies that the solution quality degrades rapidly as any of the components is changed. Both peaks represent Nash equilibria points — modifying either the x or the y value (but not both) would lead to a decrease in the function value. This implies that no single population has a rational incentive to move away from the Nash equilibrium, once both populations have converged there.

In coevolutionary learning, each population is afforded only a partial glimpse at the search space, via the fitness assessment of each individual. Specifically, each population can only respond to the fitness difference among the individuals in that population, and such differences lie along the *projection* of the joint fitness space: one such projection is shown in Figure 2. Ideally, each of the populations perceives the two peaks in this manner, so that they both learn

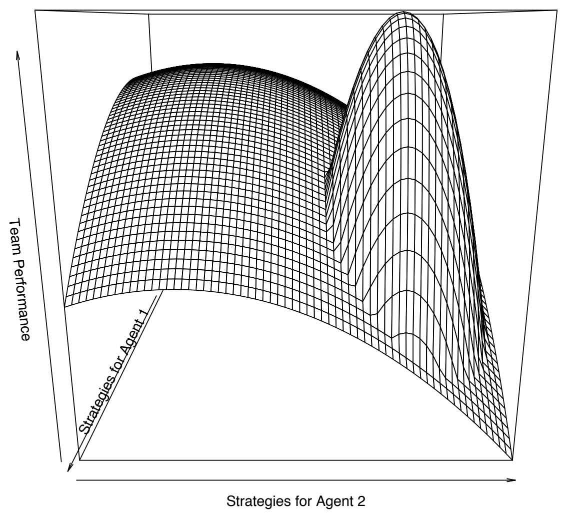

Figure 1: A bimodal search space for the possible rewards received by a two-agent team. Wider peaks may attract many search trajectories, although such peaks may be globally suboptimal.

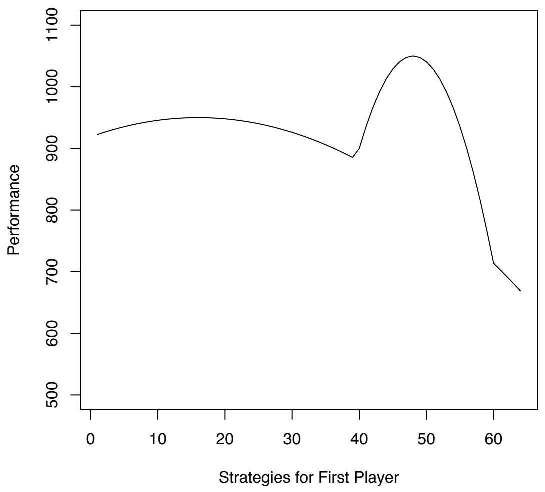

Figure 2: A desirable projection of solution quality for the first population provides enough information on the location of the optimal peak. Such a projection may be computed at each point x as $max_{y \in Y} f(x, y_i)$, where Y denotes the range of the second argument.

to choose actions that lead to globally optimal solution (the higher peak).

Figure 3 shows approximations of the search space obtained via projection when selecting random collaborators from the other population. In order to compute these projections, we proceeded as follows: for each possible individual a in population 1, we generated uniformly K random individuals $(y_i)_{i=1..K}$ for the second population (K was set to 3, 5, 10, and 20). The "fitness" of x (i.e. the projection value at x) was set to $max_{i=1..K} f(x, y_i)$, where $f(x, y_i)$ is the quality of the solution formed by x and y_i (f is the same bimodal function used in Figures 1-2). Due to stochasticity, the process is repeated three times (there are three curves on each plot in Figure 3). The graphs show that the projections are very noisy for small numbers of samples, but they become more and more accurate when increasing the number of randomly generated collaborators. When using 20 collaborators, the projections are almost perfect.

Upon random initialization of the coevolutionary search process, the collaborators are spread across the entire space. In this situation, the quality of the projection is very sensitive to the number of collaborators used. As the evolution progresses, the populations start to converge. This departure from uniformity alters the projections of the search space, and thus the ranking of individuals in each population.

Assume that the distributions of individuals in the subpopulations resemble a normal distribution[1]. Then suppose that the other population starts to converge towards the lower, wider peak. Figure 4 uses a normal distribution instead of a uniform distribution for choosing the collaborators from the other population. The normal distribution is centered on the lower peak, and collaborators generated outside the valid range are ignored. With a wide standard deviation (20) and a large number (5-20) of collaborators, the projection resembles the one in Figure 2. However, with a lower variation (suggesting the population of collaborators has converged), the projection is far less accurate. The individuals corresponding to the suboptimal peak are evaluated almost perfectly, but the individuals corresponding to the optimal peak are eventually eliminated. Importantly, the differences among the graphs become small (especially for those with standard deviations 2 and 5), which suggests that additional collaborators have diminishing returns.

On the other hand, suppose that the second population starts to converge towards the higher, narrower peak (Figure 5). As before, a wide variance and a large number of collaborators provides a good-quality projection of the space. As the population of collaborators converges (the standard deviation drops), the projection becomes less accurate, especially around the suboptimal peak (which is less important to global convergence). The minor differences among projections for low standard deviations hint again at the diminishing returns of additional collaborators.

What can we learn from Figures 2-5? First, we observe that the quality of projection increases almost insignificantly with more collaborators when the other population starts to converge. This is intuitive, given the fact that the samples are not uniformly distributed, but rather they're generated according to normal distributions. Second, we observe that the information regarding the higher peak fades away as the other population converges to the suboptimal peak. This means that a population might benefit from additional collaborators at the early stages of search, but the impact of more collaborators fades once the populations start to converge to suboptimal solutions.

[1]Although Popovici and De Jong (2003) argue that distributions are usually not distributed normally, our argument holds for many distributions that indicate convergence of the population.

Most cooperative coevolutionary learning algorithms assume that each individual is evaluated when in combination with a fixed number of collaborators. However, our observations suggest that additional collaborators from a population with high diversity provide more information than when taken from populations that have started to converge. The next section experiments with a simple ad-hoc scheme that decreases the number of collaborators over time.

Experiments

There are many approaches to decreasing the number of collaborators over time, including methods that consider the diversity of the populations. Here, we will simply apply a trivial ad-hoc setting: start with 10 collaborators for the first 5 generations, followed by using only 2 collaborators for the rest of generations until exhausting the computational resources. In short notation, we will refer to this collaboration setting as *10*5+2*rest*.

All experiments have been performed using the ECJ library (Luke, 2003). The following settings are used in all the experiments: 32 individuals per populations, tournament selection of size 2 for the breeding process, domains discretized in 1024×1024 intervals, elitism of size 1 for each subpopulations, and mutation probability 1. Mutation worked as follows: a coin was repeatedly tossed, and the individual's integer gene was increased or decreased (the direction chosen at random beforehand) until the coin came up heads (making sure it does not go outside the allowed bounds). The coin is biased such that it comes up heads with probability 0.05. One of the collaborators is always chosen as the best individual from the other subpopulation at the previous generation; the others are chosen by a tournament selection of size 2. In all experiments, there are 250 runs for each of the methods. Performance in a run is computed as the average of the best performing individuals in the two populations at the last generation. Statistical significance is verified using t-tests assuming unequal variances at 95% confidence.

The coevolutionary algorithm is given a budget of 17600 evaluations per subpopulation; settings for both subpopulations are identical. With these settings, using 5 collaborators per fitness assessment allows around 110 generations for the coevolutionary algorithm. When choosing this budget, we took into consideration the fact that too small of a value would prevent differentiations among the algorithms because the search would be cut too short, whereas too large a budget might diminish the differences between those methods that waste evaluations and methods that use them effectively. The value we chose seemed to be a good compromise.

The experiments compared the performance of methods using a fixed number of collaborators against our technique employing a variable-sized number of collaborators. We tested 10 different numbers of collaborators (1 to 10). Using a single collaborator allowed for many generations, while more collaborators provided a more accurate fitness assessment mechanisms at the expense of fewer generations.

Number Collaborators	Mean Performance	St Dev Performance
1	960.308064	30.622387
2	989.164076	48.826596
3	1004.111361	49.314857
4	1020.230865	44.603223
5	1031.049270	37.868720
6	1035.819193	32.235383
7	1036.992860	30.671499
8	1042.224061	22.536767
9	1042.303012	22.076877
10	1041.991019	20.991886
10*5+2*rest	**1047.110667**	**16.520011**

Table 1: Performance of different collaboration schemes in the discretized two-peak problem domain. 10*5+2*rest significantly outperforms all other settings.

We must point out a few problems with our experiments. First, the same collaborators are used to evaluate an entire population at each generation, as opposed to drawing random collaborators for each individual. This reduces the evaluation noise in the population, but it also limits the exploration of the search space. We are not aware of much literature comparing these two settings, but we do not expect the difference to be significant. Second, the collaborators are chosen based on a tournament selection of size 2, as opposed to choosing them randomly. This setting might diminish the benefit of multiple collaborators later on in the search (the extra selection pressure might make the population seem more converged than it actually is). In our defense, we point out the relationship between this selection pressure and the diversity in the population: the results of the selection process might be similar (in terms of expectation) if using random selection from a less diverse population, or using tournament selection with size 2 from more diverse populations. Thus, we argue that other collaboration schemes could work better than the fixed collaboration schemes even if random selection were used instead. Third, we discretized the space of individuals in this preliminary study, but a real-valued representation might be appropriate for optimization problems such as the ones used in our experiments.

The first experiment compared the results of the settings mentioned above in the two-peaks domain illustrated in Figure 1. Specifically, the quality of a complete solution (x,y) is computed as

$$f(x,y) = \max \begin{cases} 950 - 500 * \left(\left(\frac{x-16}{64}\right)^2 + \left(\frac{y-16}{64}\right)^2 \right) \\ 1050 - 9600 * \left(\left(\frac{x-48}{64}\right)^2 + \left(\frac{y-48}{64}\right)^2 \right) \end{cases}$$

where x and y ranged from 0 to 64, discretized into 1024 intervals.

The results are summarized in Table 1. The results for 10*5+2*rest are significantly better than the ones for any fixed collaboration scheme. We believe this difference comes from a better allocation of computational resources that allowed the 10*5+2*rest method to find the optimal solution in 235 out of 250 runs.

Number Collaborators	Mean Performance	St Dev Performance
1	999.805368	0.702603
2	999.931824	0.165259
3	999.948679	0.151703
4	999.946774	0.133769
5	999.939639	0.142567
6	999.947729	0.110188
7	999.951085	0.082815
8	999.943306	0.107280
9	999.940341	0.117084
10	999.945124	0.088891
10*5+2*rest	**999.986614**	**0.054527**

Table 2: Performance of different collaboration schemes in the discretized Rosenbrock-like problem domain. 10*5+2*rest significantly outperforms all other settings.

Number Collaborators	Mean Performance	St Dev Performance
1	999.974879	0.087959
2	999.992303	0.017698
3	999.992725	0.014395
4	999.994459	0.003547
5	999.994457	0.003247
6	999.994630	0.003522
7	999.994800	0.003434
8	**999.995458**	**0.003786**
9	999.994743	0.004014
10	999.994870	0.004457
10*5+2*rest	**999.995700**	**0.003646**

Table 3: Performance of different collaboration schemes in the discretized Griewangk problem domain. 10*5+2*rest significantly outperforms all other settings, except for 8 collaborators.

The second experiment tested the methods using a Rosenbrock-like[2] two-dimensional search space with

$$f(x,y) = 1000 - \left(2 * \left(x^2 - y\right)^2 + (1-x)^2\right)$$

where x and y range between -5.12 and 5.12 discretized into 1024 intervals. The results are summarized in Table 2. The results for 10*5+2*rest are significantly better than the ones for all fixed collaboration schemes.

A third experiment used the Griewangk function

$$f(x,y) = 1000 - \left(1 + \frac{x^2}{4000} + \frac{y^2}{4000} - \cos(x) * \cos\left(\frac{y}{\sqrt{2}}\right)\right)$$

with x and y between -5.12 and 5.12, discretized again into 1024 intervals. This modified function has several subopti-

[2]This function resembles the two-dimensional Rosenbrock, but with a diminished influence of the non-linear component of the fitness function.

Number Collaborators	Mean Performance	St Dev Performance
1	999.944621	0.258064
2	999.945253	0.278818
3	999.896547	0.624545
4	999.927854	0.182885
5	999.940833	0.168899
6	999.911407	0.215897
7	999.912132	0.194326
8	999.893961	0.227951
9	999.903006	0.208813
10	999.877023	0.315515
10*5+2*rest	**999.996837**	**0.022781**

Table 4: Performance of different collaboration schemes in the discretized Booth problem domain. 10*5+2*rest significantly outperforms all other settings.

Number Collaborators	Mean Performance	St Dev Performance
1	999.295637	0.686536
2	999.282862	0.696519
3	999.306431	0.707764
4	999.255339	0.752893
5	999.253378	0.715312
6	999.286966	0.730213
7	999.270756	0.696554
8	999.251628	0.716294
9	999.266060	0.741226
10	999.215434	0.717056
10*5+2*rest	999.258291	0.747410

Table 5: Performance of different collaboration schemes in the discretized Rastrigin problem domain. Although 5*10+2*rest has lower mean performance than other settings, it is not statistically significantly dominated because of the large variance in performance.

mal peaks, and an optima of value 1000 at (0,0). The results of different collaboration methods on this problem domain are summarized in Table 3. The results indicate that 10*5+2*rest is significantly better than all other settings except for 8 collaborators; it is not possible to distinguish between these two settings with 95% confidence.

The fourth experiment used with the Booth problem domain as described in (Schwefel, 1995). In this domain, the two-argument optimization function equals

$$f(x,y) = 1000 - \left((x+2*y-7)^2 + (2*x+y-5)^2\right)$$

(again transformed to a maximization problem), with x and y between -5.12 and 5.12, discretized into 1024 intervals. This is a challenging problem for coevolutionary search because of non-linearities among variables; the minimum of the function is at (1,3) and it has a value of 1000. The results of the methods in this domain are presented in Table 4. The 10*5+2*rest method has a significantly better performance than all fixed settings.

The fifth and last experiment used the Rastrigin function (converted for maximization) to assess the performance of the solution

$$f(x,y) = 1000 - \left(6 + \left(x^2 - 3\cos(2\pi x) + \left(y^2 - 3\cos(2\pi y)\right)\right)\right)$$

with x and y taking values between -5.12 and 5.12, discretized into 1024 intervals. A peculiarity of this function is the large number of suboptimal peaks surrounding the global optimum. Due to large variances of results in this problem domain, we performed 1000 runs for each of the settings. The results obtained with different methods are presented in Table 5. In this domain, the t-tests failed to indicate the superiority of any fixed collaboration scheme over the 10*5+2*rest setting.

Conclusions

The illustrations in Figures 2-5 suggest that early stages in the coevolutionary search can potentially take advantage of large numbers of collaborators to identify more promising areas of the joint space. This is the case because the populations at initial generations exhibit a high diversity in individuals. Later generations tend to converge to specific areas of the space, and thus large numbers of samples provide less information for the search process. Based on these observations, we argue that a variable number of collaborators may be a better approach.

We proposed a very simple ad-hoc technique for varying the number of collaborators: choosing 10 (maximum of all alternatives considered in this paper) collaborators for the first 5 generations, and choosing 2 collaborators thereafter. We felt this setting would provide a good tradeoff between initial exploration and later exploitation. This simple collaboration scheme was never dominated, and it was significantly better in several domains.

The paper pointed out that randomly selecting the same number of collaborators may be suboptimal due to variances in the population diversity. We preserved the use of random selection for collaborators, and we suggested varying the number of collaborators. Alternatively, one may select the same number of collaborators, but choose these collaborators based on how different they are from one another. Yet another approach may involve selecting varying numbers of collaborators based on the diversity of the populations that they are selected from. We are also currently experimenting with a method that selects collaborators partly based on how much information they provide to rank the individuals in the other population. Which of these approaches works better, and under what conditions, are possible directions for future research.

Acknowledgements

The authors would like to thank R. Paul Wiegand for comments and suggestions, and also for the help in plotting the figures.

References

Bull, L. (1997). Evolutionary computing in multi-agent environments: Partners. In Back, T., editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 370–377. Morgan Kaufmann.

Eriksson, R. and Olsson, B. (1997). Cooperative coevolution in inventory control optimisation. In Smith, G., Steele, N., and Albrecht, R., editors, *Proceedings of the Third International Conference on Artificial Neural Networks and Genetic Algorithms*, University of East Anglia, Norwich, UK. Springer.

Husbands, P. and Mill, F. (1991). Simulated coevolution as the mechanism for emergent planning and scheduling. In Belew, R. and Booker, L., editors, *Proceedings of the Fourch International Conference on Genetic Algorithms*, pages 264–270. Morgan Kaufmann.

Luke, S. (2003). ECJ 10: An Evolutionary Computation research system in Java. Available at http://www.cs.umd.edu/projects/plus/ec/ecj/.

Luke, S., Balan, G. C., and Panait, L. (2003). Population implosion in genetic programming. In *Genetic and Evolutionary Computation – GECCO-2003*, volume 2724 of *LNCS*, pages 1729–1739. Springer.

Panait, L., Wiegand, R. P., and Luke, S. (2004). A visual demonstration of convergence properties of cooperative coevolution. In *Parallel Problem Solving from Nature – PPSN-2004*. Springer.

Popovici, E. and De Jong, K. (2003). Understanding EA dynamics via population fitness distribution. In E. Cantu-Paz, *et al.*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, pages 1604–1605. Springer-Verlag.

Potter, M. (1997). *The Design and Analysis of a Computational Model of Cooperative CoEvolution*. PhD thesis, George Mason University, Fairfax, Virginia.

Potter, M. and De Jong, K. (1998). The coevolution of antibodies for concept learning. In Eiben, A. E., Baeck, T., Schoenauer, M., and Schwefel, H.-P., editors, *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature (PPSN V)*, pages 530–539. Springer-Verlag.

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

Potter, M. A., De Jong, K. A., and Grefenstette, J. J. (1995). A coevolutionary approach to learning sequential decision rules. In *Proceedings from the Sixth International Conference on Genetic Algorithms*, pages 366–372. Morgan Kaufmann.

Schwefel, H. (1995). *Evolution and Optimum Seeking*. John Wiley and Sons, New York.

Wiegand, R. P. (2003). *Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, Department of Computer Science, George Mason University.

Wiegand, R. P., Liles, W., and De Jong, K. (2001). An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In E. Cantu-Paz, *et al.*, editor, *Proceedings of the 2001 Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 1235–1242.

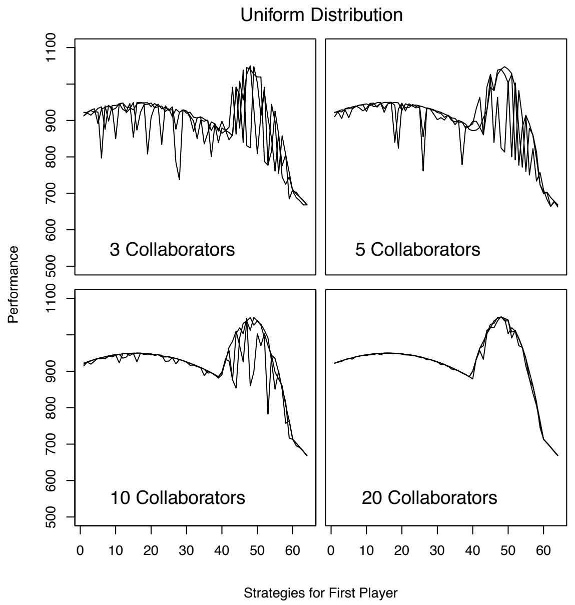

Figure 3: Projection of solution quality for the first population in the two-peaks domain, given a uniform distribution for the second population. The projection at point x is computed as $max_i f(x, y_i)$. Due to stochasticity, the process is repeated three times - there are three curves on each graph. Given more collaborators, the projection approximates

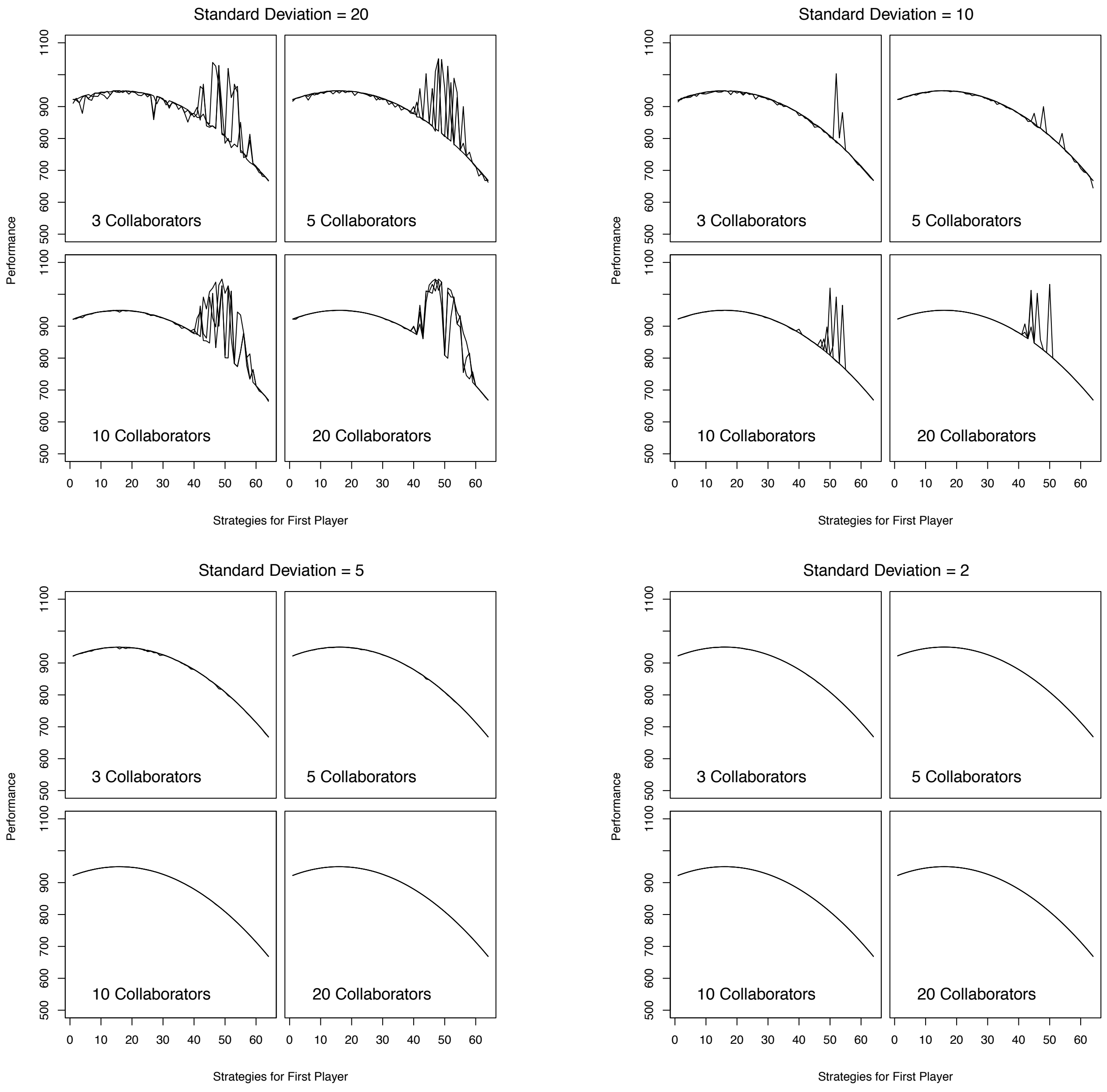

Figure 4: Projection of solution quality for the first population in the two-peaks domain, assuming the second population is normally distributed around the lower peak. The projection at point x is computed as $max_i f(x, y_i)$. $(y_i)_i$ are randomly generated according to the normal distribution with mean 16 (centered on the lower peak) and standard deviations equal to 20, 10, 5, and respectively 2. The process is repeated three times.

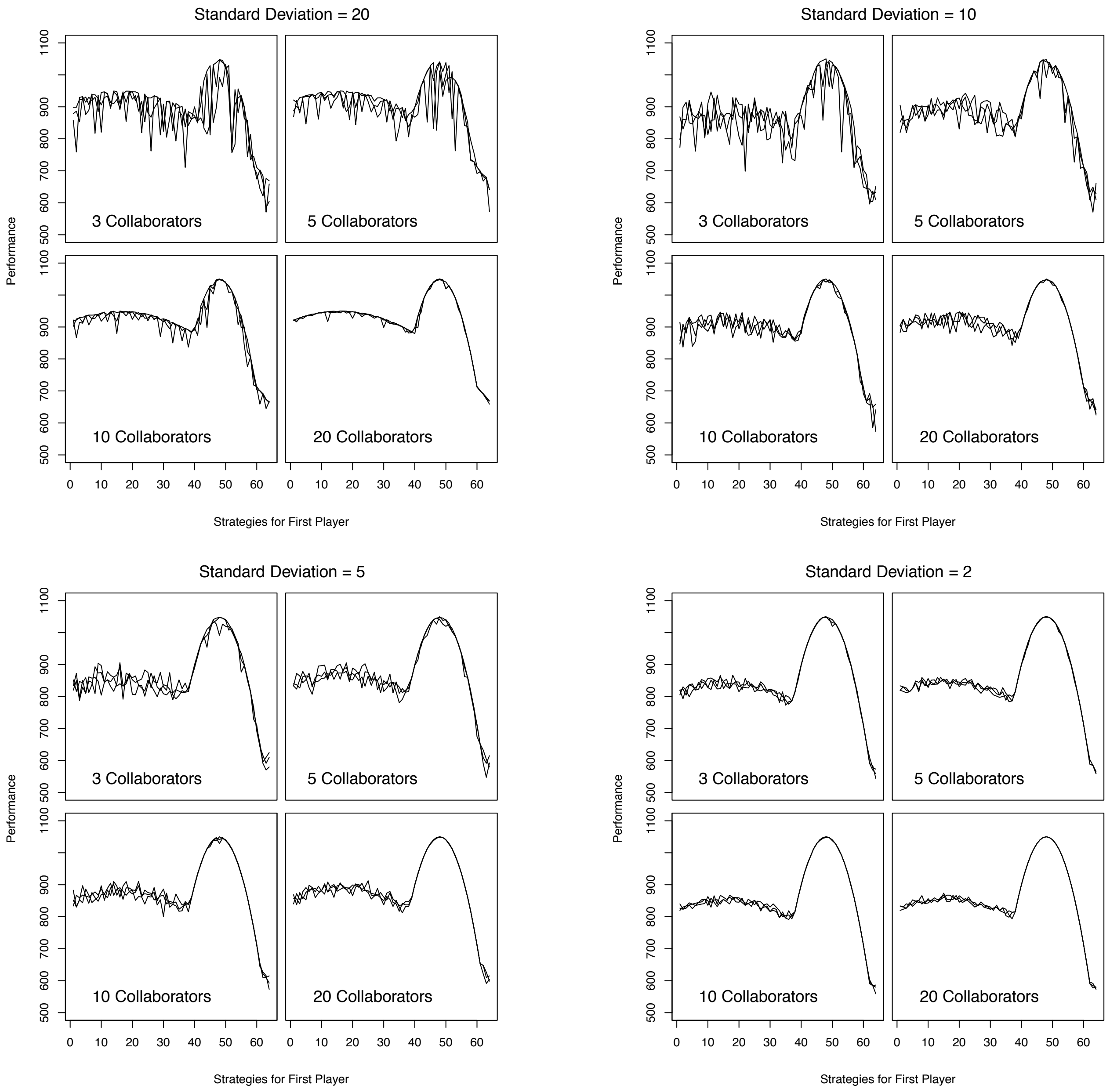

Figure 5: Projection of solution quality for the first population in the two-peaks domain, assuming the second population is normally distributed around the higher peak. The projection at point x is computed as $max_i f(x, y_i)$. $(y_i)_i$ are randomly generated according to the normal distribution with mean 48 (centered on the higher peak) and standard deviations equal to $20, 10, 5$, and respectively 2. The process is repeated three times.