Information Sharing and Conflict Resolution in Distributed Factored Evolutionary Algorithms

Stephyn G. W. Butcher
Johns Hopkins University
Baltimore, MD
steve.butcher@jhu.edu

John W. Sheppard
Montana State University
Bozeman, MT
john.sheppard@montana.edu

Shane Strasser
Montana State University
Bozeman, MT
shane.strasser@msu.montana.edu

ABSTRACT

Competition and cooperation are powerful metaphors that have informed improvements in multi-population algorithms such as the Cooperative Coevolutionary Genetic Algorithm, Cooperative Particle Swarm Optimization, and Factored Evolutionary Algorithms (FEA). However, we suggest a different perspective can give a finer grained understanding of how multi-population algorithms come together to avoid problems like hitchhiking and pseudo-minima. In this paper, we apply the concepts of information sharing and conflict resolution through Pareto improvements to analyze the distributed version of FEA (DFEA). As a result, we find the original DFEA failed to implement FEA with complete fidelity. We then revise DFEA and examine the differences between it and FEA and the new implications for relaxing consensus in the distributed algorithm.

KEYWORDS

multi-population algorithms, factored evolutionary algorithms, Pareto improvement

ACM Reference Format:
Stephyn G. W. Butcher, John W. Sheppard, and Shane Strasser. 2018. Information Sharing and Conflict Resolution in Distributed Factored Evolutionary Algorithms. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205455.3205512

1 INTRODUCTION

Competition and cooperation are among the big themes in many biologically inspired optimization algorithms. This is especially true for the Genetic Algorithm (GA) but often just as true for similar algorithms such as Particle Swarm Optimization (PSO). Algorithm improvements have often come from varying the degrees of cooperation and competition. For example, Potter and de Jong [9] developed the Cooperative Coevolutionary Genetic Algorithm (CCGA) to combat hitchhiking in the GA by using multiple populations or "subspecies" that cooperated by focusing on separate parts of an optimization problem. This line of research also includes Strasser *et al.*'s Factored Evolutionary Algorithms (FEA), which re-introduced some competition back at the multi-population level [10]. FEA was extended further by creating a distributed version, called DFEA, that does not rely on a global context [1].

While the trade off between competition and cooperation has been a major theme of this research, Butcher *et al.* [2] started to investigate these algorithms from a different perspective. Instead of viewing these algorithms on a continuum of cooperation and competition, we started to look at information sharing both within and between populations and how conflicting information was resolved. We believe this perspective can be generalized and applied to better understand the relationship between FEA and DFEA. Our analysis will show that DFEA, as originally designed, introduces different communication semantics than those in FEA. We start by analyzing FEA in terms of information sharing, applying this analysis to the original DFEA and comparing the results, and then developing a new version of DFEA that faithfully replicates the communication semantics of FEA.

For example, in FEA, there is a central context that acts like a blackboard architecture to which the populations communicate potentially better values and from which better values are communicated to the populations. When multiple better values are received, this creates a conflict. This conflict is resolved, variable by variable, in a Pareto improving fashion. Thus the context so constructed does not exhibit the hitchhiking problem, which often plagues single population algorithms. This perspective led Butcher *et al.* to develop the Pareto Improving PSO (PI-PSO), a single population *gbest*-like PSO that does not exhibit hitchhiking.

In this paper we re-evaluate DFEA in terms of information sharing and conflict resolution. We demonstrate that the distributed version differs from the undistributed version in important ways. We use this information to improve DFEA and show how it will converge to the same answer as FEA given the same random seed. This perspective also has implications for relaxing consensus in DFEA, which we explore in the final section. We begin with some background.

2 BACKGROUND

One of the main problems biologically inspired optimization algorithms face is the curse of dimensionality. Although not

GECCO '18, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205512

Table 1: Hitchhiking in PSO

$pbest_j$	x	f(x)
1	[1.53, 1.84, 5.29, 0.59]	34.06
2 ($gbest_{new}$)	[*0.42*, 2.01, *4.76*, 1.84]	30.26
3	[3.23, 0.72, 4.68, 0.47]	33.07
4	[2.83, 3.83, 2.71, 1.27]	31.64
$gbest_{old}$	[*2.39*, 1.24, *5.71*, 0.34]	39.97

unique to these algorithms or optimization problems in general, as the search spaces of problems grow larger, so do candidate solutions for those problems. As a result, in algorithms like GA and PSO, we start to see a phenomenon called *hitchhiking*. Hitchhiking occurs when the current best solution in the population is replaced by a successor solution that increases overall fitness but includes some individual variable values that are inferior to those they replaced. In Particle Swarm Optimization, this has been called "Two Steps Forward, One Step Back" [12].

Table 1 shows an example of hitchhiking in PSO during the update to the global best while optimizing the Sphere function. Although Individual 2 has the best overall fitness and will replace the existing global best, if we look at individual variables, only x_1 and x_3 are actual improvements over $gbest_{old}$ (blue). x_2 and x_4 are actually inferior values and are thus *hitchhikers* (red).

At least one strand of research aimed at mitigating hitchhiking has focused on the continuum between competition and cooperation by increasing cooperation through multi-population algorithms. For example, Potter and de Jong [9] developed the Cooperative Coevolutionary Genetic Algorithm (CCGA) to combat hitchhiking in the Genetic Algorithm (GA) by using cooperating "subspecies" to focus on separate parts of the problem. Van den Bergh and Engelbrecht [13] first applied a CCGA-like version of PSO to training neural networks and later generalized their algorithm, producing several versions including the Cooperative PSO (CPSO) and the Hybrid CPSO [14].

Strasser *et al.* [10] developed a more general *meta*-algorithm, Factored Evolutionary Algorithms (FEA), that continued the cooperative aspects of CCGA but permitted factors—variable partitions—to overlap and re-introduced a measure of competition at the multi-population level. Table 2 shows an example of FEA, without overlapping populations, using PSO as the underlying optimizer. In this case, a *context* C representing the composite solution is updated with information from the individual swarms. Conflicts between C and the swarms are resolved by inspecting variables one by one. As each variable is evaluated, the value that represents a Pareto improvement is the one that is accepted for inclusion in C_{new} (blue). This process is what prevents hitchhiking.

Assigning pieces of the problem to individual, cooperating populations did not unequivocally increase performance of these algorithms, however. It was found that the CCGA does not work as well with problems with high *epistasis*—variables whose values are highly correlated. Similarly, Van

Table 2: FEA-PSO Determination of C_{new}

$gbest$	x	f(x)
C	[2.39, 1.24, 5.71, 0.34]	39.97
s_1	[1.53, ----, ----, ----]	36.59
s_2	[----, 2.01, ----, ----]	42.47
s_3	[----, ----, 4.68, ----]	29.27
s_4	[----, ----, ----, 1.27]	41.47
C_{new}	[1.53, 1.24, 4.68, 0.34]	25.90

den Bergh and Engelbrecht discovered in their research on CPSO that the partitioning of the variables mattered because they were often highly correlated. They labeled this phenomenon *pseudo-minima*. It was for this reason that they introduced the Hybrid CPSO [14].

In contrast, Strasser et al. found overlapping factors in FEA were able to solve both hitchhiking and avoid pseudo-minima, if the factors overlapped appropriately. And this result applied to all of the evolutionary algorithms they used as optimizers [11].

3 INFORMATION SHARING AND CONFLICT RESOLUTION

FEA is a meta-algorithm composed of three steps: optimize, compete, and share. The optimize step consists of applying an algorithm such a GA, PSO, Hill Climbing, etc., to a subset of the problem. This subset is called a factor, $\mathbf{X}_i$ which generally includes only some values in $\mathbf{X}$. The problem to be optimized is $f(\mathbf{X})$. Because $\mathbf{X}_i \subset \mathbf{X}$, we have $\mathbf{X} \setminus \mathbf{X}_i = \mathbf{R}_i$ missing values needed to evaluate $f(\mathbf{X}_i)$. These are supplied by the *context*, $\mathbf{C}_i$, which can be thought of as a blackboard architecture as in [3] or, more generally, by Engelmore [6]. This is one form of information sharing.

In the Compete Step, the information flows from the swarms to the context. At minimum, at least one swarm will have been optimizing some variable x_j and this new, potentially better, value will need to be evaluated against the existing value in the context, c_j. This is where the conflict resolution comes in. If we have an existing context, $[c_1, c_2, c_3, c_4]$ and a new x_2, we will evaluate both $f([c_1, c_2, c_3, c_4])$ and $f([c_1, x_2, c_3, c_4])$ and either keep c_2 or select x_2 depending on which gives the better fitness. More generally, there may be many swarms optimizing x_j, the set of which we will designate $\mathbf{O}_j$, and there will be $|\mathbf{O}_j| + 1$ conflicting possibilities for x_j, including c_j.

The order of variable resolution can be arbitrarily set but this does not mean that every order will end up with the same result. Using X_i to denote either c_i or x_i, we might evaluate X_1, X_3, X_4, X_2 and this will end up with a different result than X_1, X_2, X_3, X_4 for C. Throughout the following discussion, We will use the order X_1, X_2, X_3, X_4 where $v_i = x_i$ or c_i but the results do not depend on this order. It just makes the bookkeeping and exposition clearer. Algorithm 1 shows the pseudocode for the FEA Compete Step using PSO as the optimizer.

Algorithm 1 FEA-PSO Compete

Input: Function $f(x)$, swarms $\boldsymbol{\mathcal{S}}$, optimizers $\boldsymbol{O}$, context $\mathbf{C}$
Output: New global context $\mathbf{C}$

1: **for** $j = 1$ **to** d **do**
2: $fitness \leftarrow f(\mathbf{C})$
3: $value \leftarrow \mathbf{C}.c_j$
4: **for** j in $\boldsymbol{O}_j$ **do**
5: $candidate \leftarrow \boldsymbol{\mathcal{S}}[j].p_{gbest}$
6: $\boldsymbol{C}.c_j \leftarrow candidate.x_j$
7: **if** $f(\boldsymbol{C})$ is better than $fitness$ **then**
8: $value \leftarrow candidate.x_j$
9: $fitness \leftarrow f(\boldsymbol{C})$
10: **end if**
11: **end for**
12: $\boldsymbol{C}.c_j \leftarrow value$
13: **end for**
14: **return** $\mathbf{C}$

3.1 Reconciliation

In Butcher *et al.* [2], they were concerned with comparing PSO and PI-PSO and how differences in information sharing and conflict resolution solved the problem of hitchhiking. In this paper we are concerned with how differences in the FEA and DFEA algorithms give rise to differences in information flows. Thus, we need a notation that is more abstract than the value-based notation of Tables 1 and 2. For this we will use a few symbols. First, we use the term *reconciliation* to cover both the sharing of new values of x_j from optimizers of x_j, O_j and the resolution of the conflicting values. A variable, v_j, that has not be reconciled yet will be denoted by $\bigodot$. For a variable that has been reconciled we will use $\bigoplus$. Note that this says nothing about whether c_j was kept or one of $x_j^k, \forall k = 1 \ldots |O_j|$ replaced it.

As an example consider a $d = 4$ problem. After the optimize step, the context appears as $[\bigodot\ \bigodot\ \bigodot\ \bigodot]$. When X_1 is reconciled, the context becomes $[\bigoplus\ \bigodot\ \bigodot\ \bigodot]$. The important thing to note here is that X_1, whether it is c_1 or x_1, is determined in the context of the other variables, X_2, X_3, X_4. If we continue with X_2, we have $[\bigoplus\ \bigoplus\ \bigodot\ \bigodot]$, followed by $[\bigoplus\ \bigoplus\ \bigoplus\ \bigodot]$, and finally followed by the reconciliation of X_4, $[\bigoplus\ \bigoplus\ \bigoplus\ \bigoplus]$. Keeping this pattern of reconciliation in mind, we now apply this same analysis to the original DFEA.

3.2 Reconciliation in the Original DFEA

In most respects, DFEA is the same as FEA. The main difference is that instead of a single context, $\mathbf{C}$, each swarm has its own context, $\mathbf{C}_k$. The challenges become keeping these separate blackboards synchronized and determining the implications when they are not synchronized. The implications of losing synchronization—relaxing consensus—were explored in Butcher *et al.* [1] and we will investigate them later in this paper. For now, we will maintain full consensus.

The first challenge arises because there are multiple contexts and possibly many optimizers of x_j, so some $\mathbf{C}_k$ must be designated as having *the* c_j. This challenge is solved by designating one of the swarms optimizing x_j to be the *arbiter* of x_j, which we denote $a(x_j)$. During the DFEA Compete Step, all optimizers of x_j communicate their values of x_j to the arbiter and the arbiter compares those values (including its own) with c_j found in its context. A simplified version of this DFEA Compete Step, utilizing PSO, is presented in Algorithm 2.

Algorithm 2 DFEA-PSO Compete

Input: Function $f(x)$, arbitrator look up $a(x)$, swarms $\boldsymbol{\mathcal{S}}$, optimizers $\boldsymbol{O}$, contexts $\boldsymbol{C}$
Output: global contexts $\boldsymbol{C}$

1: **for** $j = 1$ **to** d **do**
2: $\mathbf{C} \leftarrow \boldsymbol{C}[a(x_j)]$
3: $fitness \leftarrow f(\mathbf{C})$
4: $value \leftarrow \mathbf{C}.c_j$
5: **for** k in $\boldsymbol{O}_j$ **do**
6: $candidate \leftarrow \boldsymbol{\mathcal{S}}[k].p_{gbest}$
7: $\mathbf{C}.c_j \leftarrow candidate.x_j$
8: **if** $f(\mathbf{C})$ is better than $fitness$ **then**
9: $value \leftarrow candidate.x_j$
10: $fitness \leftarrow f(\mathbf{C})$
11: **end if**
12: **end for**
13: $\mathbf{C}.c_j \leftarrow value$
14: **for** j in $\boldsymbol{O}[i]$ **do**
15: $\mathbf{C}[j].\mathbf{c}_i \leftarrow \mathbf{c}_i$
16: **end for**
17: **end for**
18: **return** $\boldsymbol{C}$

Table 3: Evolution of Context(s) in FEA and DFEA

t_i	FEA	C_i	DFEA-PSO
1	$[\bigoplus\ \bigodot\ \bigodot\ \bigodot]$	1	$[\bigoplus\ \bigodot\ \bigodot\ \bigodot]$
2	$[\bigoplus\ \bigoplus\ \bigodot\ \bigodot]$	2	$[\bigodot\ \bigoplus\ \bigodot\ \bigodot]$
3	$[\bigoplus\ \bigoplus\ \bigoplus\ \bigodot]$	3	$[\bigodot\ \bigodot\ \bigoplus\ \bigodot]$
4	$[\bigoplus\ \bigoplus\ \bigoplus\ \bigoplus]$	4	$[\bigodot\ \bigodot\ \bigodot\ \bigoplus]$

The DFEA Compete Step presented represents the reconciliation of all the contexts, $\mathbf{C}_k \subset \boldsymbol{C}$ and we can analyze the operation of the algorithm with the notation we have developed. We assume that the arbitrator for x_j is $\mathbf{C}_j$ ($\mathbf{C}$ in Line 2 of Algorithm 2), and as before we take the variables in order, X_1, X_2, X_3, X_4. Once again, we start at the end of the optimize step when all the contexts are identical.

In that case, when we reconcile X_1, we will have $\mathbf{C}_1 = [\bigoplus\ \bigodot\ \bigodot\ \bigodot]$, and when we reconcile X_2, we will have $\mathbf{C}_2 = [\bigodot\ \bigoplus\ \bigodot\ \bigodot]$. Similarly, $[\bigodot\ \bigodot\ \bigoplus\ \bigodot]$ and $[\bigodot\ \bigodot\ \bigodot\ \bigoplus]$ follow for X_3 and X_4.

We would now like to compare the state of the two algorithms after their respective Compete Steps. To do this we observe, that since our examples have used the same order of reconciliation, we can line up the single FEA context at time t_i with the corresponding DFEA context C_i. This is shown in Table 3.

Aligned this way, the difference between the algorithms becomes immediately apparent. For FEA, X_2 is reconciled at t_2 in the presence of the reconciled value of X_1 from t_1 ($\bigoplus$). For DFEA, X_2 is reconciled in $\mathbf{C}_2$ with the *unreconciled* value of X_1 ($\bigodot$). While it is possible that X_2 will be the same value in both cases, it is not guaranteed. Additionally, it seems unlikely that every v_j would reconcile exactly the same way for both algorithms over all variables and iterations.

This analysis so far has been only for the Compete Steps. Both FEA and DFEA have an additional step, the Share Step, which has a greater importance for DFEA. In FEA, the Share Step is mostly a bookkeeping step where the new context is communicated back to the individual populations and fitness values are re-evaluated. Additionally, a form of elitism introduces the context as an actual individual in the population. In DFEA, the Share Step includes this same bookkeeping but also simulates the movement of information in a network. We will now look at how this influences information sharing.

Consider two swarms that are regarded to be *neighbors*. Neighbors can be defined in a number of ways but Butcher *et al.* define it in the context of the set of optimizers, $\boldsymbol{O}$. If two swarms, s_j and s_k are both members of some set of optimizers $\boldsymbol{O}_i$, then they are considered to be neighbors. During the Share Step, those two swarms can compare how recent the values of all variables in their contexts, $\mathbf{C}_j$ and $\mathbf{C}_k$ are. If $\mathbf{C}_j$ has a newer value of c_i than $\mathbf{C}_k$, then $\mathbf{C}_k$ will take $\mathbf{C}_j$'s value. If the reverse is true, $\mathbf{C}_j$ will take $\mathbf{C}_k$'s value of c_i.

The neighbor relation sets up the possibility of swarms being indirect neighbors as well. If s_j and s_k are both members of $\boldsymbol{O}_i$ and s_k and s_m are both members of $\boldsymbol{O}_q$ then s_j and s_m are indirect neighbors. Information will flow from one to the other depending on the number of iterations of the Share Step. The neighbor relation induces a topology on the swarms through which information travels, and if the Share Step iterates a sufficient number of times, information will flow from $\mathbf{C}_1$ to $\mathbf{C}_d$. As a result, all $\mathbf{C}_i$ will be identical. This is called *consensus*.

Relative to FEA, however, this sharing takes place too late to affect how reconciliation plays out in the DFEA Compete Step. Consider reconciliation of c_1 and c_2. First c_1 is reconciled with the optimizers $\boldsymbol{O}_1$, and we have $\mathbf{C}_1 = [\bigoplus \bigodot \bigodot \bigodot]$. Next, c_2 is reconciled with optimizers $\boldsymbol{O}_2$, and we have $\mathbf{C}_2 = [\bigodot \bigoplus \bigodot \bigodot]$. During the Share Step, c_1 from $\mathbf{C}_1$ will be shared with $\mathbf{C}_2$, but c_1 will not have been determined in the context of $\mathbf{C}_2$. In fact, when c_1 is changed to c_1' in $\mathbf{C}_2$, $\mathbf{C}_2$ is not re-evaluated at all. We thus have no way of knowing if the change is Pareto improving or not. To signify this, we use the $\bigotimes$ symbol: $\mathbf{C}_2 = [\bigotimes \bigoplus \bigodot \bigodot]$. With full consensus, $\mathbf{C}_2$ will eventually look like $[\bigotimes \bigoplus \bigotimes \bigotimes]$. Although each of the values c_1, c_2, c_3, and c_4 will have been Pareto improvements when they were evaluated during reconciliation, at no point were they evaluated *collectively*. Even with full consensus, the contexts in DFEA do not collectively preserve the information semantics of the context in FEA.

Algorithm 3 DFEA-PSO Reconcile

Input: Function $f(x)$, arbitrator look up $a(x)$, swarms $\boldsymbol{S}$, optimizers $\boldsymbol{O}$, global contexts $\boldsymbol{C}$
Output: global contexts $\boldsymbol{C}$

```
1: for j = 1 to d do
2:     C ← C[a(x_j)]
3:     fitness ← f(C)
4:     value ← C.c_j
5:     for k in O_j do
6:         candidate ← S[k].p_gbest
7:         C.c_j ← candidate.x_j
8:         if f(C) is better than fitness then
9:             value ← candidate.x_j
10:            fitness ← f(C)
11:        end if
12:    end for
13:    C.c_j ← value
14:    for k = 1 to d do
15:        C[k].c_j ← C.c_j
16:    end for
17: end for
18: return C
```

3.3 Revising Reconciliation in DFEA

In order to fix this discrepancy, we suggest a change to the DFEA algorithm. Based on Table 3 it would appear that the sharing step and Compete Step need to happen simultaneously. Additionally, if we wish the right-hand side of the table to match the left-hand side of the table, in the ideal case, we need to start by considering all swarms to be neighbors of all other swarms. Later we will investigate what relaxing consensus might mean in the revised algorithm. This new *reconcile* step that combines both information sharing and conflict resolution is described as Algorithm 3.

By adding a *broadcast* loop at 14 every time some c_i is reconciled, the reconciliation is communicated to all the other contexts. For any c_j to be reconciled, all $c_k, \forall k < j$ will be their reconciled values, just as in the FEA Compete Step. The revised DFEA relegates the Share Step to performing similar bookkeeping functions as it does in the FEA.

Using our previous notation, when X_1 is reconciled to $\mathbf{C}_1 = [\bigoplus \bigodot \bigodot \bigodot]$, all the other contexts $\mathbf{C}_j$ will have X_1 updated as well: $\mathbf{C}_2 = [\bigoplus \bigodot \bigodot \bigodot]$, $\mathbf{C}_3 = [\bigoplus \bigodot \bigodot \bigodot]$, and $\mathbf{C}_4 = [\bigoplus \bigodot \bigodot \bigodot]$. And when X_2 is reconciled it will be in the context of the reconciled value of X_1 just as in the FEA: $\mathbf{C}_2 = [\bigoplus \bigoplus \bigodot \bigodot]$. Table 4 shows the end result as compared to FEA. Now all four DFEA contexts, C_i, are the same as the FEA context at t_4.

This section has demonstrated how looking at these algorithms in terms of information flows and conflict resolution (reconciliation) can reveal a deeper structure and more interesting semantics than invoking cooperation versus competition. We examined how reconciliation works in FEA and original DFEA and showed that the semantics of the two were not identical as previously supposed. Using the same

Table 4: Final Context(s) in FEA and Revised DFEA

t_i	FEA	C_i	DFEA-PSO
1	[⊕ ⊙ ⊙ ⊙]	1	[⊕ ⊕ ⊕ ⊕]
2	[⊕ ⊕ ⊙ ⊙]	2	[⊕ ⊕ ⊕ ⊕]
3	[⊕ ⊕ ⊕ ⊙]	3	[⊕ ⊕ ⊕ ⊕]
4	[⊕ ⊕ ⊕ ⊕]	4	[⊕ ⊕ ⊕ ⊕]

framework, we devised a revised DFEA that does preserve the semantics of the FEA. Finally, we encountered something new. In the original DFEA, a value c_j that was a Pareto improvement in $\mathbf{C}_j$ was communicated to $\mathbf{C}_k$ without any evaluation ($\otimes$). What impact this might have on the operation of the algorithm and what it means for performance relative to the FEA is not entirely clear. For now, we refer to these values as *discordant* because they are injected into context without any conflict resolution.

4 EXPERIMENTS

Based on the previous discussion and analysis, our hypothesis is that the revised DFEA and FEA will perform the same. The revision that was made ensures that, other things being equal, FEA and the revised DFEA will end up with the same result. As for the original DFEA, it is difficult to say if the discordant values influence the performance of the algorithm. Because [1] showed that the original DFEA was sometimes better and sometimes worse than FEA, we also hypothesize that the results will be mixed. We will revisit this hypothesis and the results later in the discussion of future work.

In order to test our hypothesis, we ran a large number of experiments on standard benchmark functions from different categories for FEA-PSO, revised DFEA-PSO, and old DFEA-PSO. We include results for the single population *gbest* PSO as a baseline. The following sections describe the design, results and discussion of those experiments.

4.1 Design

We selected benchmark optimization problems from [7] and [14] that were scalable to multiple dimensions. The problems selected are shown in Table 5, arranged by categories inspired by [8]. All of the problems are minimization problems, and with the exception of the Exponential and Eggholder functions, they all have a minimizing solution and value of $f([0]^d) = 0$. The Exponential function has a minimum at $[-1]^d$, and the Eggholder function has a dimension-dependent minimum and minimizing vector. None of the functions except the Sphere function are separable in their current forms.

Experiments consisted of 50 runs of each algorithm on each benchmark function with a dimension of 32. Because we noticed that the results were not always normally distributed—hardly a surprise for optimization problems—the confidence intervals were 500 replications of the Bootstrap to estimate 95% confidence intervals [4]. Following [5], each algorithm used the same number of candidate solutions. In this case we chose 10 particles per dimension, that is $d \times 10 = 320$.

Table 5: Benchmark Optimization Functions by Category

Category	Benchmark Function
Bowl	Exponential, Sargan, Sphere
Many Local Optima	Ackley-1, Eggholder, Griewank, Rastrigin, Salomon, Stretched-V
Plate	Brown, Schwefel-2.23, Whitley, Zakharov
Ridge	Michalewicz, Schaffer-F6, Schwefel-2.22
Valley	Dixon-Price, Rosenbrock, Schwefel-1.2

PSO and the PSO portion of FEA and the various DFEA versions all used the same parameters: $\omega = 0.729$ and $\phi_1 = \phi_2 = 1.49618$. While PSO was run for 100 iterations, FEA versions were run for 20 FEA/DFEA iterations separated by 5 PSO iterations for a total of 100 PSO iterations. All FEA/DFEA variants used the "Simple Centered" factor of $i, i+1$, which followed the functional form of most of the benchmark functions—they are functions of adjacent x values—and shown by Strasser *et al.* to perform well [10]. With $d-1$ such factors, and $d = 32$, there were $\lfloor (320/31) \rfloor = 10$ particles per swarm for the FEA/DFEA-PSO variants.

4.2 Results

The results are shown in Table 6. Independent of which algorithm is best, what we are looking for, in general, is for FEA-PSO and the new DFEA-PSO to have similar performance. This appears to be true for Ackley, Brown, Exponential, Salomon, Schaffer-F6, Schwefel-1.2, Schwefel-2.22, Schwefel-2.23, Sphere, Stretched-V, and Zakharov. There were 19 benchmark functions overall, so our experiments show that the results were similar for FEA-PSO and the new DFEA-PSO for 11 of them (58%).

There were six cases where the original or old DFEA-PSO performed the same as the new DFEA-PSO, five where they performed better than FEA-PSO: Dixon-Price, Eggholder, Griewank, Michalewicz, Rosenbrock, and one where they were worse then FEA-PSO: Rastrigrin. In three cases, FEA-PSO was better than either version of DFEA-PSO (Rastrigrin, Sargan, Whitley). In three cases, PSO was better than any FEA variant: Salomon, Schwefel-1.2 and Zakharov.

4.3 Discussion

Given that FEA and the revised DFEA are demonstrably equivalent, it is surprising that there are eight cases out of 19 where they did not have the same results. All FEA/DFEA variants depend on PSO as the underlying optimizer. The variants all have the same numbers of factors for each problem and thus the same number of swarms. In the code tested, they are all initialized the same way and at the same time. Furthermore, the (D)FEA Compete/Share/Reconcile Steps as presented do not have any stochastic elements which might cause a purely *random* divergence.

Table 6: Comparison of PSO, FEA-PSO and both variants of DFEA-PSO

	PSO		FEA-PSO		New/Revised DFEA-PSO		Old DFEA-PSO	
Benchmark	Mean	Confidence Interval	Mean	Confidence Interval	Mean	Confidence Interval	Mean	Confidence Interval
ackley-1	1.84e+00	(1.77e+00, 1.90e+00)	**2.81e-03**	**(5.23e-06, 7.00e-03)**	**2.78e-03**	**(3.82e-08, 6.99e-03)**	**2.83e-03**	**(1.39e-08, 6.98e-03)**
brown	7.56e+00	(6.69e+00, 8.55e+00)	**1.22e-25**	**(3.72e-26, 2.42e-25)**	**1.21e-25**	**(5.11e-26, 1.98e-25)**	1.02e-23	(1.36e-24, 2.42e-23)
dixon-price	4.23e+01	(2.54e+01, 6.34e+01)	1.18e+02	(1.07e+02, 1.28e+02)	**3.85e+01**	**(2.93e+01, 4.70e+01)**	**3.87e+01**	**(2.98e+01, 4.92e+01)**
eggholder	-1.68e+04	(-1.71e+04, -1.64e+04)	-1.77e+04	(-1.79e+04, -1.74e+04)	**-2.10e+04**	**(-2.13e+04, -2.07e+04)**	**-2.05e+04**	**(-2.07e+04, -2.02e+04)**
exponential	-9.99e-01	(-1.00e+00, -9.99e-01)	-1.00e+00	(-1.00e+00, -1.00e+00)	-1.00e+00	(-1.00e+00, -1.00e+00)	-1.00e+00	(-1.00e+00, -1.00e+00)
griewank	4.96e-01	(4.47e-01, 5.45e-01)	9.49e-01	(8.99e-01, 9.91e-01)	**1.49e-01**	**(6.92e-02, 2.23e-01)**	**1.35e-01**	**(7.51e-02, 2.02e-01)**
michalewicz	-8.65e+00	(-9.02e+00, -8.33e+00)	-2.59e+01	(-2.63e+01, -2.56e+01)	**-3.06e+01**	**(-3.07e+01, -3.04e+01)**	**-3.06e+01**	**(-3.07e+01, -3.04e+01)**
rastrigin	1.03e+02	(9.59e+01, 1.11e+02)	**2.60e-02**	**(1.97e-05, 6.59e-02)**	7.28e-02	(1.36e-02, 1.51e-01)	1.01e-01	(3.98e-02, 1.99e-01)
rosenbrock	1.95e+02	(1.56e+02, 2.51e+02)	2.20e+02	(1.69e+02, 2.88e+02)	**4.60e+01**	**(1.96e+01, 7.77e+01)**	**5.09e+01**	**(1.31e+01, 1.04e+02)**
salomon	**1.41e+00**	**(1.33e+00, 1.48e+00)**	2.29e+00	(2.12e+00, 2.49e+00)	1.96e+00	(1.78e+00, 2.12e+00)	1.93e+00	(1.77e+00, 2.17e+00)
sargan	9.76e+00	(8.55e+00, 1.11e+01)	**2.04e-12**	**(1.37e-12, 2.91e-12)**	5.67e+02	(1.32e+02, 1.15e+03)	3.19e+03	(1.54e+03, 5.07e+03)
schaffer-f6	2.49e+00	(2.31e+00, 2.67e+00)	1.99e+00	(1.78e+00, 2.20e+00)	**9.86e-01**	**(8.60e-01, 1.09e+00)**	**9.34e-01**	**(8.23e-01, 1.04e+00)**
schwefel-1.2	**7.96e+03**	**(7.19e+03, 8.84e+03)**	6.59e+04	(5.72e+04, 7.60e+04)	6.53e+04	(4.59e+04, 8.85e+04)	6.88e+04	(5.55e+04, 8.44e+04)
schwefel-2.22	3.01e+02	(2.91e+02, 3.13e+02)	**1.22e-12**	**(7.73e-13, 1.75e-12)**	**1.35e-12**	**(5.74e-13, 2.40e-12)**	**1.54e-12**	**(8.63e-13, 2.41e-12)**
schwefel-2.23	2.44e-01	(1.18e-01, 4.15e-01)	**8.65e-102**	**(7.25e-116, 2.31e-101)**	**5.07e-102**	**(2.39e-111, 1.67e-101)**	**1.39e-101**	**(8.93e-103, 3.02e-101)**
sphere	0.00e+00	(0.00e+00, 0.00e+00)	0.00e+00	(0.00e+00, 0.00e+00)	0.00e+00	(0.00e+00, 0.00e+00)	0.00e+00	(0.00e+00, 0.00e+00)
stretched-v	1.20e+01	(1.14e+01, 1.28e+01)	4.37e+00	(3.78e+00, 5.02e+00)	4.24e+00	(3.91e+00, 4.54e+00)	**3.58e+00**	**(3.31e+00, 3.86e+00)**
whitley	9.82e+02	(9.67e+02, 9.99e+02)	**3.51e+02**	**(2.97e+02, 3.91e+02)**	5.35e+02	(4.84e+02, 5.80e+02)	5.62e+02	(5.20e+02, 6.02e+02)
zakharov	**1.39e+02**	**(1.28e+02, 1.51e+02)**	1.60e+03	(3.09e+02, 3.77e+03)	8.16e+02	(7.80e+02, 8.45e+02)	7.97e+02	(7.67e+02, 8.23e+02)

Because of this we decided to run a second set of experiments on the same benchmark functions. This time each run, i, of FEA-PSO, old DFEA-PSO and revised/new DFEA-PSO used the same random seed, $seed_i$. As we can see in these results, shown in Table 7, the FEA-PSO and revised DFEA-PSO had exactly the same results as we would expect. Thus it appears that the random seed was the culprit in generating the differences in performance.

In this second set of experiments, what is perhaps more interesting is that for most of the problems, the differences in the DFEA versions did not seem to matter. For 14 of the 19 benchmark problems, all three algorithms performed about the same. For two benchmark problems, the old DFEA-PSO performed better than the FEA-PSO/revised DFEA-PSO (Rosenbrock, Stretched-V). For three of the benchmark problems, the FEA-PSO/revised DFEA-PSO performed better than the old DFEA-PSO (Salomon, Sargan, Zakharov).

5 RELAXING CONSENSUS

Not only did Butcher *et al.* introduce the distributed version of the FEA or DFEA, we also examined the implications of relaxing consensus. With our revised DFEA, we now examine the effects of relaxing consensus as well. In the previous work, relaxed consensus was achieved in the DFEA Share Step when newly reconciled values were not communicated throughout the network of contexts induced by the neighbor relation. In the revised DFEA, however, we have replaced the Compete and Share Steps with a Reconcile Step. The reconcile step replaces a network model using "hops" with a network model using broadcasted messages. In order to introduce an effect like relaxed consensus in the revised DFEA, we introduce the idea of dropped messages.

Referring back to line 14 in Algorithm 3, we see that after a new c_j is reconciled by the arbiter of x_j, it is broadcast to all other swarms. By broadcasting c_j, any $c_k, \forall k > j$ that is subsequently reconciled has the benefit of this new information. The main point of the previous discussion about reconciliation in the original DFEA was that this did not happen. In effect, the original DFEA never had full consensus in the sense that FEA does.

Now we introduce the idea that the message containing the new c_j may be dropped. This is accomplished through a success rate, r, which determines if a message is delivered from some arbiter of x_j to each of the remaining swarms. If $r = 0.8$, there is a 20% probability that the message from the arbiter of x_1, for example, to x_2 will go missing, in which case when c_2 is reconciled, $\mathbf{C}_2$ will not have the new value of c_1. This could happen for several iterations of the DFEA reconcile step, depending on r. But this also means that for any given iteration of the DFEA reconcile step, some arbiters will get c_1, and some will not with probability $1 - r$.

5.1 Experiment

In order to test the effects of dropped messages and their implications for relaxing consensus, we re-ran the benchmark experiments from above. All the parameters are the same. The only difference is that we simulated different success rates of $r = 1.0$ (the baseline), 0.8, 0.6, and 0.4. The results are presented in Table 8.

5.2 Discussion

Looking at the results, we can see how well the revised DFEA did on the various benchmarks with different success—or drop—rates. The first column in Table 8 is a baseline of 100% success. Comparing the results for a 100% and 80% success rate, we can see that the revised DFEA required a success rate of 100% on 10 of 19 benchmarks. These benchmarks were Ackley, Brown, Griewank, Rosenbrock, Schwefel-1.2, Schwefel-2.22, Schwefel-2.23, Sphere, Whitley, and Zakharov. Perhaps the most surprising appearance in this list is the Sphere function. The Sphere function is fairly simple and separable in its variables. One would think this would make missed messages less important relative to some of the other benchmark functions.

Table 7: FEA-PSO, Old and New DFEA-PSO with Same Random Seeds

	FEA-PSO		New/Revised DFEA-PSO		Old DFEA-PSO	
Benchmark	Mean	Confidence Interval	Mean	Confidence Interval	Mean	Confidence Interval
ackley-1	3.37e-03	(5.38e-07, 8.96e-03)	3.37e-03	(5.38e-07, 8.96e-03)	2.01e-03	(5.46e-08, 5.95e-03)
brown	2.90e-21	(4.32e-25, 9.17e-21)	2.90e-21	(4.32e-25, 9.17e-21)	2.96e-21	(1.51e-24, 9.30e-21)
dixon-price	3.73e+01	(2.82e+01, 4.72e+01)	3.73e+01	(2.82e+01, 4.72e+01)	3.20e+01	(2.22e+01, 4.14e+01)
eggholder	-2.12e+04	(-2.15e+04, -2.10e+04)	-2.12e+04	(-2.15e+04, -2.10e+04)	-2.07e+04	(-2.10e+04, -2.03e+04)
exponential	-1.00e+00	(-1.00e+00, -1.00e+00)	-1.00e+00	(-1.00e+00, -1.00e+00)	-1.00e+00	(-1.00e+00, -1.00e+00)
griewank	7.14e-02	(2.57e-02, 1.28e-01)	7.14e-02	(2.57e-02, 1.28e-01)	1.71e-01	(8.22e-02, 2.66e-01)
michalewicz	-3.09e+01	(-3.10e+01, -3.08e+01)	-3.09e+01	(-3.10e+01, -3.08e+01)	-3.09e+01	(-3.10e+01, -3.08e+01)
rastrigin	2.28e-01	(1.12e-01, 3.40e-01)	2.28e-01	(1.12e-01, 3.40e-01)	2.28e-01	(1.12e-01, 3.40e-01)
rosenbrock	1.58e+01	(6.38e+00, 2.70e+01)	1.58e+01	(6.38e+00, 2.70e+01)	**3.16e+00**	**(2.40e+00, 3.97e+00)**
salomon	**1.79e+00**	**(1.64e+00, 1.93e+00)**	**1.79e+00**	**(1.64e+00, 1.93e+00)**	2.82e+00	(2.63e+00, 3.05e+00)
sargan	**1.67e+03**	**(5.58e+02, 2.84e+03)**	**1.67e+03**	**(5.58e+02, 2.84e+03)**	1.35e+05	(1.04e+05, 1.77e+05)
schaffer-f6	9.26e-01	(8.24e-01, 1.02e+00)	9.26e-01	(8.24e-01, 1.02e+00)	8.85e-01	(7.79e-01, 9.71e-01)
schwefel-1.2	8.80e+04	(5.37e+04, 1.24e+05)	8.80e+04	(5.37e+04, 1.24e+05)	1.39e+07	(7.09e+06, 2.02e+07)
schwefel-2.22	4.63e-12	(1.06e-12, 1.11e-11)	4.63e-12	(1.06e-12, 1.11e-11)	4.63e-12	(1.06e-12, 1.11e-11)
schwefel-2.23	2.27e-94	(6.55e-100, 6.92e-94)	2.27e-94	(6.55e-100, 6.92e-94)	1.12e-99	(3.06e-101, 3.28e-99)
sphere	0.00e+00	(0.00e+00, 0.00e+00)	0.00e+00	(0.00e+00, 0.00e+00)	0.00e+00	(0.00e+00, 0.00e+00)
stretched-v	4.23e+00	(3.91e+00, 4.53e+00)	4.23e+00	(3.91e+00, 4.53e+00)	**2.83e+00**	**(2.54e+00, 3.14e+00)**
whitley	5.46e+02	(5.05e+02, 5.87e+02)	5.46e+02	(5.05e+02, 5.87e+02)	6.68e+02	(5.82e+02, 7.55e+02)
zakharov	**7.62e+02**	**(7.34e+02, 7.87e+02)**	**7.62e+02**	**(7.34e+02, 7.87e+02)**	3.46e+11	(2.01e+11, 4.81e+11)

Conversely, the revised DFEA continued to do well on nine of 19 benchmarks even with a success rate of 80% (drop rate of 20%). The benchmarks were Dixon-Price, Eggholder, Exponential, Michalewicz, Rastrigin, Salomon, Sargan, Schaffer-F6, and Stretched-V. Strangely, on Michalewicz, although the difference was not statistically significant, the lower success rate of 80% actually led to an *increase* in performance.

On a few benchmarks, the revised DFEA did fairly well even with a success rate of 60%, which is fairly low (a drop rate of 40% per variable, per iteration). These benchmarks were Eggholder, Exponential, Michalewicz, Rastrigrin, and Salomon. On a single benchmark, Exponential, the revised DFEA's performance was statistically indistinguishable whether the drop rate was 0%, 20%, 40% or 60%.

These results seem to suggest that some tolerance to noise is acceptable for some problems but that full consensus is best during the reconcile step.

6 CONCLUSIONS

In many biologically inspired algorithms such as the Genetic Algorithm and Particle Swarm Optimization, problems with hitchhiking are often tackled by introducing multiple populations and varying the degree of cooperation and competition. This line of research extends from Potter and de Jong's Cooperative Coevolutionary Genetic Algorithm [9], through Van den Bergh and Engelbrecht's Cooperative PSO [13], and Strasser *et al.*'s Factored Evolutionary Algorithm [10].

Following Butcher *et al.* [2], we focused instead on information sharing and conflict resolution or, taken together, reconciliation between the populations. Reconciliation relies on the shared context as a blackboard architecture through which the populations communicate and single variable Pareto improvement as the standard of conflict resolution. We applied this framework to DFEA [1] and demonstrated that DFEA does not preserve the information semantics of FEA as originally supposed. We then used the framework to develop a revised DFEA where the FEA information semantics are preserved.

In experiments we showed that, given the same initial conditions and random seed, the FEA-PSO and revised DFEA-PSO produce the exact same results as expected. However, the strong performance of the original DFEA-PSO suggests that there is more to conflict resolution than previously thought. While is seems clear why a set of interdependent Pareto improvements would lead to the elimination of hitchhikers, it is not equally clear why the concatenation of single value Pareto improvements would also be effective. This would appear to have research implications for the entire family of FEA algorithms.

We also demonstrated the effects of relaxing consensus in the revised DFEA. In these experiment we showed that some tolerance for noise exists in the revised algorithm but that full consensus during the reconcile step almost always leads to better performance.

In future work we anticipate investigating the semantics of reconciliation further and applying the new DFEA to other problems. We also wish to investigate the possibility of finding an optimal arbitration order. Additionally, the current DFEA algorithm is defined at a very high level. We would like to investigate concrete implementations of DFEA that are distributed, parallel, and possibly asynchronous.

Table 8: Relaxing Consensus by Success Rates

	1.0		0.8		0.6		0.4	
Benchmark	Mean	Confidence Interval	Mean	Confidence Interval	Mean	Confidence Interval	Mean	Confidence Interval
ackley-1	**2.78e-03**	**(3.82e-08, 6.99e-03)**	3.34e-02	(2.08e-02, 4.52e-02)	1.10e-01	(9.15e-02, 1.28e-01)	4.61e-01	(3.58e-01, 5.86e-01)
brown	**1.21e-25**	**(5.11e-26, 1.98e-25)**	3.53e-20	(3.02e-21, 8.93e-20)	7.02e-15	(9.67e-17, 2.31e-14)	1.05e+00	(1.09e-10, 3.29e+00)
dixon-price	**3.85e+01**	**(2.93e+01, 4.70e+01)**	**1.62e+02**	**(2.81e+01, 3.57e+02)**	1.05e+03	(5.62e+02, 1.54e+03)	1.78e+04	(9.39e+03, 2.88e+04)
eggholder	**-2.10e+04**	**(-2.13e+04, -2.07e+04)**	**-2.05e+04**	**(-2.09e+04, -2.02e+04)**	**-1.98e+04**	**(-2.03e+04, -1.94e+04)**	-1.71e+04	(-1.77e+04, -1.66e+04)
exponential	**-1.00e+00**	**(-1.00e+00, -1.00e+00)**	**-1.00e+00**	**(-1.00e+00, -1.00e+00)**	**-1.00e+00**	**(-1.00e+00, -1.00e+00)**	**-9.98e-01**	**(-1.00e+00, -9.93e-01)**
griewank	**1.49e-01**	**(6.92e-02, 2.23e-01)**	7.52e-01	(6.98e-01, 8.03e-01)	5.78e-01	(5.23e-01, 6.21e-01)	4.90e-01	(4.45e-01, 5.42e-01)
michalewicz	**-3.06e+01**	**(-3.07e+01, -3.04e+01)**	**-3.08e+01**	**(-3.10e+01, -3.07e+01)**	**-3.06e+01**	**(-3.08e+01, -3.04e+01)**	-2.85e+01	(-2.87e+01, -2.82e+01)
rastrigin	**7.28e-02**	**(1.36e-02, 1.51e-01)**	**1.89e-01**	**(5.97e-02, 3.45e-01)**	**2.77e-01**	**(1.10e-01, 4.70e-01)**	4.14e+00	(2.07e+00, 6.33e+00)
rosenbrock	**4.60e+01**	**(1.96e+01, 7.77e+01)**	1.14e+03	(2.70e+02, 2.29e+03)	4.67e+03	(2.83e+03, 6.75e+03)	1.04e+05	(6.43e+04, 1.44e+05)
salomon	**1.96e+00**	**(1.78e+00, 2.12e+00)**	**1.74e+00**	**(1.69e+00, 1.79e+00)**	**1.90e+00**	**(1.85e+00, 1.96e+00)**	2.44e+00	(2.37e+00, 2.52e+00)
sargan	**5.67e+02**	**(1.32e+02, 1.15e+03)**	**2.80e+03**	**(9.57e+02, 4.94e+03)**	8.00e+03	(4.44e+03, 1.22e+04)	1.23e+04	(9.67e+03, 1.50e+04)
schaffer-f6	**9.86e-01**	**(8.60e-01, 1.09e+00)**	**9.52e-01**	**(8.56e-01, 1.06e+00)**	1.38e+00	(1.27e+00, 1.48e+00)	2.32e+00	(2.09e+00, 2.56e+00)
schwefel-1.2	**6.53e+04**	**(4.59e+04, 8.85e+04)**	3.22e+05	(2.86e+05, 3.55e+05)	2.97e+05	(2.74e+05, 3.24e+05)	2.75e+05	(2.56e+05, 2.95e+05)
schwefel-2.22	**1.35e-12**	**(5.74e-13, 2.40e-12)**	2.94e-06	(9.51e-08, 7.99e-06)	5.25e+00	(1.90e+00, 9.10e+00)	5.72e+01	(4.40e+01, 7.15e+01)
schwefel-2.23	**5.07e-102**	**(2.39e-111, 1.67e-101)**	5.22e+06	(1.27e+06, 1.00e+07)	1.02e+09	(6.05e+08, 1.56e+09)	4.07e+09	(2.89e+09, 5.48e+09)
sphere	**0.00e+00**	**(0.00e+00, 0.00e+00)**	4.46e+01	(3.64e+01, 5.39e+01)	1.54e+02	(1.36e+02, 1.71e+02)	3.75e+02	(3.50e+02, 4.02e+02)
stretched-v	**4.24e+00**	**(3.91e+00, 4.54e+00)**	**4.16e+00**	**(3.82e+00, 4.46e+00)**	5.40e+00	(5.05e+00, 5.76e+00)	9.26e+00	(8.70e+00, 9.82e+00)
whitley	**5.35e+02**	**(4.84e+02, 5.80e+02)**	9.54e+02	(9.02e+02, 1.00e+03)	5.52e+03	(4.28e+03, 7.94e+03)	4.38e+05	(3.31e+05, 5.62e+05)
zakharov	**8.16e+02**	**(7.80e+02, 8.45e+02)**	4.66e+03	(1.29e+03, 1.35e+04)	1.19e+04	(1.47e+03, 2.48e+04)	5.14e+03	(9.58e+02, 9.91e+03)

REFERENCES

[1] Stephyn Butcher, Shane Strasser, Jenna Hoole, Benjamin Demeo, and John Sheppard. 2016. Relaxing Consensus in Distributed Factored Evolutionary Algorithms. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*. ACM, 5–12.

[2] Stephyn G. W. Butcher, John W. Sheppard, and Shane Strasser. 2018. Pareto Improving Selection of the Global Best in Particle Swarm Optimization. In *2018 IEEE Congress on Evolutionary Computation (CEC)*. forthcoming.

[3] Scott H. Clearwater, Bernardo A. Huberman, and Tad Hogg. 1992. Cooperative Problem Solving. In *Computation: The Micro and the Macro View*, B. Huberman (Ed.). World Scientific, Singapore, 33–70.

[4] B. Efron. 1979. Bootstrap Methods: Another Look at the Jackknife. *Ann. Statist.* 7, 1 (01 1979), 1–26. https://doi.org/10.1214/aos/1176344552

[5] AP Engelbrecht. 2014. Fitness Function Evaluations: A Fair Stopping Condition?. In *Proceedings of the IEEE Swarm Intelligence Symposium (SIS)*. 1–8.

[6] Robert Engelmore. 1988. *Blackboard Systems*. Addison Wesley, Reading, MA.

[7] Momin Jamil and Xin-She Yang. 2013. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation* 4, 2 (2013), 150–194.

[8] Sonja Jurjanovic and Derek Bingham. 2013. Optimization Test Problems. https://www.sfu.ca/~ssurjano/optimization.html. (2013).

[9] Mitchell A Potter and Kenneth A De Jong. 1994. A cooperative coevolutionary approach to function optimization. In *Parallel Problem Solving from Nature–PPSN III*. Springer, 249–257.

[10] Shane Strasser, Nathan Fortier, John Sheppard, and Rollie Goodman. 2017. Factored Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation* 21, 3 (2017), 281–293.

[11] Shane Strasser, John Sheppard, and Stephyn Butcher. 2017. A Formal Approach to Deriving Factored Evolutionary Algorithm Architectures. In *Proceedings of the IEEE Swarm Intelligence Symposium*. 556–563.

[12] Frans Van den Bergh and Andries Petrus Engelbrecht. 2000. Cooperative learning in neural networks using particle swarm optimizers. *South African Computer Journal* 26 (2000), p–84.

[13] Frans Van Den Bergh and Andries P Engelbrecht. 2001. Training product unit networks using cooperative particle swarm optimisers. In *Proceedings of International Joint Conference on Neural Networks*, Vol. 1. IEEE, 126–131.

[14] Frans Van den Bergh and Andries Petrus Engelbrecht. 2004. A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation* 8, 3 (2004), 225–239.