Pareto Improving Selection of the Global Best in Particle Swarm Optimization

Stephyn G. W. Butcher*, John W. Sheppard†, Shane Strasser†
*The Johns Hopkins University, Dept. of Computer Science, Baltimore, MD 21218
steve.butcher@jhu.edu
†Montana State University, Gianforte School of Computing, Bozeman, MT 59717
{john.sheppard, shane.strasser}@montana.edu

Abstract—**Particle Swarm Optimization is an effective stochastic optimization technique that simulates a swarm of particles that fly through a problem space. In the process of searching the problem space for a solution, the individual variables of a candidate solution will often take on inferior values characterized as "Two Steps Forward, One Step Back." Several approaches to solving this problem have introduced varying notions of cooperation and competition. Instead we characterize the success of these multi-swarm techniques as reconciling conflicting information through a mechanism that makes successive candidates Pareto improvements. We use this analysis to construct a variation of PSO that applies this mechanism to $gbest$ selection. Experiments show that this algorithm performs better than the standard $gbest$ PSO algorithm.**

I. Introduction

The Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) are just two of the many optimization algorithms that have been inspired by nature and society. And when these algorithms have run afoul of such problems as the Curse of Dimensionality and "hitchhiking," researchers have looked for deeper inspiration with some finding it on the spectrum of cooperation and competition. For example, Potter and de Jong [16] sought to improve the performance of the GA in the face of hitchhiking by introducing cooperation. The Cooperative Coevolutionary Genetic Algorithm (CCGA) uses "subspecies" that focus on separate parts of the problem. However, the CCGA does not work as well with problems with high *epistasis*–variables whose values are highly correlated.

Van den Bergh and Engelbrecht [18] first applied a CCGA-like version of PSO to training neural networks. In that research, much like Potter and de Jong, they discovered that the partitioning of the variables mattered because they were often highly correlated. They labeled this phenomenon *pseudo-optima*. They later generalized their algorithm creating the Cooperative PSO (CPSO) [19] as a solution to the "Two Steps Forward, One Step Back" problem, as they characterize hitchhiking in PSO. However, like its CCGA counterpart, the CPSO was prone to being mislead by pseudo-optima. They introduced the Hybrid CPSO, which alternated between a CPSO step and a full solution PSO step, to address this issue.

Strasser *et al.* [17] developed a more general meta-algorithm, Factored Evolutionary Algorithm (FEA), that continued the cooperative aspects of CCGA but permitted factors—variable partitions—to overlap. This enabled the FEA to solve both the hitchhiking and pseudo-minima problems while also being generally applicable to a wide range of evolutionary algorithms including the GA, PSO and others. Strasser also demonstrated that the CPSO was a special case of the FEA-PSO combination and that the FEA-PSO generally performed better than the CPSO or Hybrid CPSO.

In this paper we suggest that the success of these multi-population algorithms is not because of their locations on a spectrum of cooperation and competition but because of the way that the algorithms handle conflicting information in the construction of successor candidates. By implicitly using *Pareto efficiency* as a guide, these algorithms eliminate hitchhiking by resolving information conflicts with the shared solution on a variable by variable basis.

We begin analyzing successful algorithms like CPSO and FEA by using information sharing and Pareto efficiency more explicitly rather than deferring to cooperation. We then present a new PSO variant, Pareto Improving PSO (PI-PSO), that uses these mechanisms directly in the selection of the $gbest$ and thus does not need multiple populations.

In Section II we describe the PSO algorithm and give a concrete example of hitchhiking. We follow with a discussion of FEA combined with PSO (Section III). This discussion investigates the mechanism whereby FEA-PSO is able to avoid hitchhiking. We demonstrate that this mechanism involves using Pareto efficiency to resolve conflicting information in a multi-population algorithm and not necessarily cooperation. In Section IV we apply this same mechanism to the $gbest$ selection in a variant of PSO we call PI-PSO. We present experiments in Section V that show this new algorithm performs better than PSO and sometimes better than FEA-PSO. We conclude in Section VI with a discussion of potential future work opened up by this approach.

II. Particle Swarm Optimization

We begin by describing the $gbest$ variant of PSO [11] and a particular challenge it faces in the search for a candidate solution. This challenge has been called "Two Steps Forward, One Step Back" [19] but bears a striking resemblance to "hitchhiking" in the GA [12]. We will use the shorter term, *hitchhiking* hereafter.

The PSO algorithm operates on a population (swarm) of candidate solutions (particles). Each particle has a position, x;

TABLE I
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

velocity, v; fitness, $f(x)$; and the best position it has attained so far, $pbest$. The algorithm begins with particles initialized to random positions. Each iteration updates every particle's velocity and position and, if warranted, its $pbest$. After all particles are updated, the global best, $gbest$, is updated from the swarm's current set of personal bests.

Because all particles are updated before the $gbest$ is evaluated as opposed to after each particle is updated, this version simulates a *parallel* algorithm [15] instead of an *sequential* one [11]. Our analysis does not depend on the parallel version but it is more similar to the multi-population algorithms we will discuss later.

The velocity update equation is:

$$v' = \omega v + \phi_1(gbest - x) + \phi_2(pbest - x)$$

where v, v' are velocity, x is the current particle position, ω is a user defined *inertia* parameter and both ϕ_1 and ϕ_2 are vectors of random uniform variates from $(0, \phi_i)$, where each ϕ_i is another user defined parameter. The position update equation is:

$$x' = x + v'$$

As each new particle position is calculated, we compare $f(pbest)$ and $f(x')$ to determine if a new personal best has been achieved. After all particles' $pbest$s are updated, we pick the best as the new global best, $gbest_{new}$. The update process is repeated a fixed number of iterations or until some other stopping criterion is met. $gbest$ is returned as the solution. However, because of hitchhiking, the $gbest$ solution may not be as good as it could have been.

Hitchhiking is more easily explained with a concrete example. Suppose we are trying to minimize the four-dimensional Sphere function ($\sum_{i=1}^{4} x_i^2$) on the interval $[0, 10]^4$ and we find ourselves at some arbitrary iteration, ready to update the global best. Although hitchhiking can occur in all functions, we use the Sphere function for this example because it is separable. This permits the unambiguous attribution of changes in individual variables to overall fitness. If x_i increases, $f(x)$ increases and if x_i decreases, $f(x)$ decreases.

Table I shows the current positions and fitnesses of four particles' $pbest$s. Because particle 2's personal best has the lowest fitness, it will be the new global best. In order to show what hitchhiking is, we compare it with the $gbest_{old}$ it just replaced.

The old global best is shown at the bottom of Table I. As expected it has a worse fitness than its successor. However, if we compare each x_i pairwise, we can see that while Particle 2 was a global improvement, it was not a local improvement for all variables. While x_1 and x_3 (italics/blue) are better in $gbest_{new}$ than in $gbest_{old}$ because they have smaller values, x_2 and x_4 (bold/red) are actually larger than their corresponding values in $gbest_{old}$. The individually inferior values for x_2 and x_4 are *hitchhikers*.

III. FACTORED EVOLUTIONARY ALGORITHMS

One approach researchers have taken to solving the problem of hitchhiking in both GA and PSO is by introducing *cooperation* via multi-population versions of the algorithms. The technique started in GAs with CCGA [16] and was introduced into PSO as CPSO [19]. Another strand of research culminated in a multi-population technique called FEA [17]. Both CPSO and FEA-PSO were shown to be improvements over $gbest$ PSO. In this discussion, we focus on FEA-PSO because it has been demonstrated that CPSO is a special case of FEA-PSO [17].

In the following discussion we develop the argument that FEA-PSO and similar algorithms do not prevent hitchhiking through cooperation but through their conflict resolution mechanism. We start with a formal description of the FEA-PSO algorithm. In order to keep us from veering off into edge cases that are not relevant to the discussion, we make a few simplifying assumptions in our presentation.

Suppose we wish to minimize an explicit function $f(x)$ of d variables, X. We define a *factor* of X to be some proper subset of X, denoted X_i. The set of factors is F. For the moment, the only constraint we put on F is that each variable, x_j, must appear in at least one factor. This particular conception of factors is similar but not identical to the one underlying factor graphs.

If we try to evaluate $f(X_i)$, we have a problem in that it does not contain values for every variable in X. This difficulty is surmounted by the use of a *context*, C, that can be used to provide the missing values of X. We call the set of these *remaining* values R_i, where $R_i = C - X_i$.

For a concrete example, let us once again consider the Sphere function in four dimensions. In this case, $X = \{x_1, x_2, x_3, x_4\}$ and $d = 4$. We can define factors $X_1 = \{x_1\}$, $X_2 = \{x_2\}$, $X_3 = \{x_3\}$, and $X_4 = \{x_4\}$ with $F = \{X_1, X_2, X_3, X_4\}$. This particular factor architecture is called "non-overlapping" because no factor shares a variable. In the more general case, there could be sets of *optimizers* of x_i, O_i, equal to the set of factors that contain x_i.

The context C contains values for each variable in X, $\{c_1, c_2, c_3, c_4\}$. Thus $f(X_1, R_1)$ is $f(x_1, c_2, c_3, c_4)$. When factor X_1 is assigned to swarm s_1, only x_1 is optimized. The other values from C remain constant. Similarly, every x_i is being optimized by a swarm (and under the present assumptions, only one swarm).

A. FEA-PSO Initialization

At the beginning of the FEA-PSO algorithm, C is initialized at random, just like the position of any other particle. Each factor is assigned to its own swarm, which optimizes the

Algorithm 1 FEA-PSO Compete

```
Input: Function $f(x)$ to optimize, swarms $\boldsymbol{\mathcal{S}}$, optimizers $\boldsymbol{\mathcal{O}}$, full global context $\mathbf{C}$
Output: New global context $\mathbf{C}$
1:  for $i = 1$ to $d$ do
2:      $fitness \leftarrow f(\mathbf{C})$
3:      $value \leftarrow \mathbf{C}.c_i$
4:      for $j$ in $\boldsymbol{\mathcal{O}}_i$ do
5:          $candidate \leftarrow \boldsymbol{\mathcal{S}}[j].p_{gbest}$
6:          $\boldsymbol{C}.c_i \leftarrow candidate.x_i$
7:          if $f(\boldsymbol{C})$ is better than $fitness$ then
8:              $value \leftarrow candidate.x_i$
9:              $fitness \leftarrow f(\boldsymbol{C})$
10:         end if
11:     end for
12:     $\boldsymbol{C}.c_i \leftarrow value$
13: end for
14: return C
```

variables in X_i and treats the variables in R_i as constants. The Optimize Step executes the PSO algorithm for every swarm for some predetermined number of iterations, usually around 5-10. When the Optimize Step is done, each swarm has its own $gbest$, which may contain new values for X_i if a new $gbest$ was discovered. Because s_1's c_2 is s_2's x_2 these new values must be exchanged between the swarms. This is accomplished in the FEA Compete Step.

B. FEA Compete Step

During the Optimize Step, each swarm, s_i, relies on the context, C, to provide values, R_i, of which it is not an optimizer. In this step, information flows from C to the swarms. When the Optimize Step is complete, each swarm, s_i, is ready to communicate new values for their respective factors, X_i, back to C. This step involves information flowing from the swarms back to C. Because information flows in both directions, we can think of C as a kind of "blackboard" architecture much like that described by Clearwater *et al.* [3] or, more generally, by Engelmore [7]. The pseudocode for the FEA Compete Step is shown in Algorithm 1.

We start our discussion of the algorithm by continuing our example above with factors, $F = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}\}$. We specifically want to contrast the generation of the new context, C_{new}, in FEA-PSO with the selection of the $gbest_{new}$ in PSO.

In Table II we find ourselves at the end of some arbitrary Optimize Step. We show the current context, C, in the first row. The next four rows show the new factor values for x_i that come from each swarm s_i's global best. The fitness value of each of the global bests is determined by both the variable, x_i, and missing values in R_i supplied by the current context, C. According to Algorithm 1, the new context is created by trying each value of x_i in the context to see whether or not it is an improvement. Thus if $f(x_1, c_2, c_3, c_4)$ is better than

TABLE II
FEA-PSO DETERMINATION OF C'

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

$f(c_1, c_2, c_3, c_4)$ then we will replace c_1 with x_1. If it is not, we will keep c_1. This process repeats for all variables.

In the example, we can see that we kept the values $c_2 = 1.24$ and $c_4 = 0.34$ from C but took the new values $x_1 = 1.53$ and $x_3 = 4.68$ to form C_{new}. We note that the fitness of C_{new} is 25.9 which is better than the fitness of any of the individual swarms' global bests. Additionally, C_{new} is an unambiguous improvement—on a variable by variable basis—over C. There is no hitchhiking. So while previous research has attributed the lack of hitchhiking to cooperation, we believe hitchhiking is eliminated by this conflict resolution.

Even though each individual swarm is optimizing a different variable, there is still a conflict between each swarm's value of x_i and the context's value c_i. This conflict is resolved by iterating over the variables and picking whichever value leads to the better fitness. In this way, no inferior value of variable i is chosen. This floor-like constraint on fitness with respect to the individual values of the variables is reminiscent of a concept in economics, Pareto efficiency.

C. Pareto Efficiency

Pareto efficiency, named for Italian economist Vilfredo Pareto, is usually applied to the allocation of resources for an individual or group of individuals subject to a preference function.[1] Consider the case where we want to redistribute apples and oranges between Jane and Sam who have started out with some allocation of each. If we can give Jane some of Sam's apples and Sam some of Jane's oranges and make both better off, then we can make a Pareto *improvement*. If we find ourselves at an allocation where no Pareto improvement can be made, then the allocation is Pareto *efficient*.

Applications of Pareto efficiency to PSO have generally focused on multi-objective optimization rather than single objective optimization. Pareto efficiency is a natural approach to resolving conflicts in problems with shared inputs and incommensurable outputs. There are many different possible combinations of minimizations of those outputs and one needs a way to evaluate them. In this sense, the outputs are like the apples and oranges of the example above. Applications to multi-objective optimization in PSO have included magnetostatics [2], job shop scheduling [13] [21], power dispatch [1] [20], and portfolio optimization [4] to name a few.

There has been research on transforming single-objective optimization problems into multi-objective optimization prob-

[1]Pareto efficiency is also known as Pareto optimality. We eschew the term *optimality* in this paper to avoid confusion.

lems via *helper objectives* and then applying Pareto efficiency [9], [14]. As we will demonstrate shortly, our approach is distinct from this research because we focus on what we view as the critical aspects of the multi-swarm PSO approaches (information sharing via a blackboard and conflict resolution) and apply them to a single-swarm PSO. We do not apply helper objectives and in no way transform our problems to be multi-objective. As far as we can determine, no one has applied this approach to identifying successor global bests in a single swarm PSO before.

We start by applying the concept of Pareto efficiency to *gbest* PSO. If we select a *new* global best, it always has a better fitness than the current global best. In this sense, it is a Pareto improvement. But given all the information in the swarm, this is a limited approach to finding a solution. To continue our economics example, it is like trying to improve Jane's situation by trading entire baskets of groceries. Jane might be better off but she could have improved even more if she had been able to fine tune the number of apples and oranges. Analogously, the successor global best will be better than the previous global best in an overall sense but it might have been better still if we could have examined the contributions of individual variables. This is why hitchhiking occurs; by focusing on candidate solutions as a whole, we cannot prevent suboptimal components being carried forward in $gbest$.

In contrast, FEA-PSO and similar algorithms focus on information sharing between individual swarms via a blackboard. These swarms contribute and receive information from the blackboard, C, on a variable by variable basis. Any contribution to the blackboard is viewed as potentially conflicting and is evaluated in the context of the current state of the blackboard. A new value for a variable is only accepted if it improves the fitness of the blackboard. This is equivalent to using Pareto efficiency to resolve information conflicts. As a result, there is no hitchhiking in FEA-PSO.

If we can summarize the differences succinctly, PSO treats candidate solutions as an entire unit and selects successors that improve fitness over their predecessors. FEA-PSO treats candidate solutions as collections of information that must be integrated into a global context—the blackboard—and resolves conflicts using Pareto efficiency.

Although the FEA-PSO and similar algorithms are implicitly using Pareto efficiency, they are not necessarily Pareto *efficient*. The algorithms simply guarantee that if some x_i in C is replaced, it will be a Pareto improvement under the function being optimized. Pareto efficiency requires that we be unable to make a Pareto improvement. A Pareto improvement in the FEA-PSO case involves allocating the information contained in all the swarms to get a *better* successor context, C_{new}. We obtain this result by considering C and a particular ordering of the variables, i. While any particular ordering does not matter, we cannot know if the ordering we picked is the one that will lead to a Pareto efficient outcome. Additionally, we only change a single variable at a time. In fact, determining a Pareto efficient outcome would be exponential in both the factor overlap and the dimension of the problem, $O(|O_i|^d)$, since all combinations would need to be tried to find the *best* successor(s).

TABLE III
FEA-PSO DETERMINATION OF C_{new} WITH OVERLAPPING FACTORS

gbest	x	f(x)
C	[2.39, 1.24, 5.71, 0.34]	39.97
s_1	[1.53, 1.84, ----, ----]	38.45
s_2	[----, 2.01, 4.76, ----]	32.53
s_3	[----, ----, 4.68, 0.47]	29.37
s_4	[----, ----, ----, 1.27]	41.47
C_{new}	[1.53, 1.24, 4.68, 0.34]	25.90

D. Pseudo Optima

There is a certain recursiveness involved when one speaks of PSO and FEA-PSO. Because FEA-PSO uses PSO as its optimizer, individual swarms are subject to the same problems as PSO. When we apply FEA-PSO to factors of a single variable, those individual PSO instantiations avoid hitchhiking because hitchhiking is impossible in a function of one variable. However, when we have factors of two or more variables, hitchhiking is unavoidable. And so while there is nothing that prevents us from having factors with more than one variable, the possibility of hitchhiking would seem to weigh strongly against doing so.

However, it turns out we are forced to consider not only larger factors but factors that overlap because of the possibility of *pseudo-optima*. As Strasser *et al.* found [17], optimizing individual factors as we have defined them can theoretically lead the individual swarms in FEA-PSO to get stuck in pseudo-optima. While a full treatment of pseudo-optima lies outside the scope of this paper, we can perhaps give an intuition for the phenomenon.

We start by noting that the global minimum we are searching for lies in $f(x)$'s problem space. However, each individual swarm is searching $f(X_i; R_i)$—it is unable to change any of the values in R_i. Because of this, the swarm may find minima of its parameterized function that are not minima of the overall function. The antidote involves increasing the sizes of factors—sometimes just to two variables but in other cases to more—and having those factors overlap. Thus $F = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}\}$ might become $F = \{\{x_1, x_2\}, \{x_2, x_3\}, \{x_3, x_4\}, \{x_4\}\}$.

Although selecting optimal factor architectures is an open research question, Strasser *et al.* found that, for many of the common benchmark functions used in optimization research, a "Simple Centered" factor architecture, $\{x_i, x_{i+1}\}$, like the one above, works well [17]. There appears to be a trade-off within FEA-PSO between hitchhiking and pseudo-optima in the individual swarms, depending on the factor architecture and function being optimized. Even so, the FEA-PSO Compete Step is still Pareto improving as we show in Table III.

Looking at both Algorithm 1 and Table III we see that in order to produce a new context C_{new}, every value of x_i is considered in addition to c_i. We have also increased competition. Although there are more values to consider, the

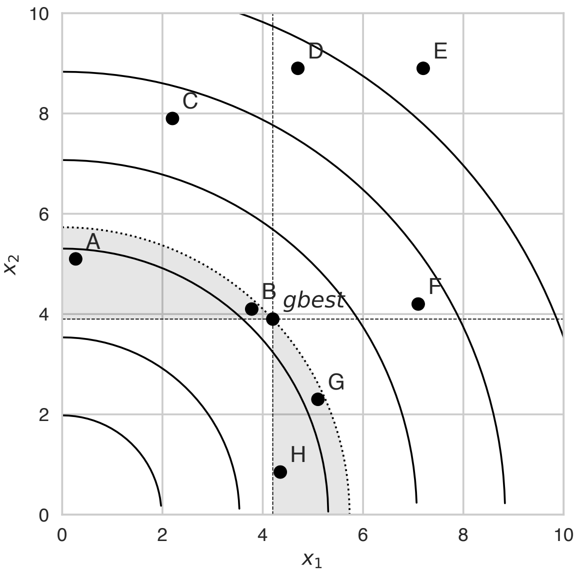

Fig. 1. Selecting $gbest$ in PSO (Sphere) and Hitchhiking

result is still a Pareto improvement. As before, however, we are still not guaranteed to find a Pareto efficient solution based on all the information available.

E. FEA Share Step

After the new context is determined, it must be communicated back to the swarms. This happens in the FEA Share Step.

The FEA Share Step mostly involves bookkeeping associated with the construction of C_{new}. Depending on the implementation, all of the particles in all of the swarms were evaluated with C and must be re-evaluated using the next context, C_{new}. Additionally, at least with the current version of FEA, the Share Step includes a form of *elitism*. The worst particle by fitness in each swarm is replaced by a particle based on the values in C_{new} This is where the communication loop between swarms is closed as each swarm sent its latest information about X_i and then receives the latest information about R_i in return. The algorithm completes when the indicated number of FEA steps have run and C is returned as the solution.

IV. PI-PSO

As noted previously, hitchhiking does not arise in a problem of one dimension. Either $f(x_1')$ is better than $f(x_1)$ or is it is not. If we consider a problem of two dimensions, when comparing any given particle's $pbest$ with the current $gbest$, we have two possibilities for each variable and four for the pairs of variables. We know that x_1 is either better or worse and that x_2 is either better or worse than their previous values. This leads to three zones for potential successor $gbest$ values (the current set of $pbest$s). These zones and the implications for Pareto improvements in the selection of a new $gbest$ are illustrated in Figure 1.

In this figure the arcs represent the contours of the Sphere function for two variables, x_1 and x_2. The current $gbest = (4.2, 3.9)$ also defines a contour (dotted) that is the dividing line between $pbest$s that have a better fitness (a lower contour) or a worse fitness (a higher contour). Additionally, the gray areas denote the set of points where the $pbest$ lies on a lower contour than the $gbest$ and thus has a better fitness but one or the other of the variables is larger than its value in $gbest$. All points in the gray zones include hitchhiking. We can thus see that $pbest$s C, D, E, F are all inferior to the current $gbest$. Points A, B and G involve hitchhikers. Only Point H has both a better fitness and no hitchhiking.

We have demonstrated both how hitchhiking arises in PSO and how the FEA-PSO mitigates it. However, we have also seen that algorithms like FEA-PSO involve trade-offs at the swarm level between preventing hitchhiking and becoming trapped in pseudo-minima. In order to overcome both problems, we propose a version of PSO that determines $gbest$ in a different way, more along the lines of FEA-PSO.

In the preceding discussion we developed the argument that FEA-PSO and similar algorithms mitigate hitchhiking not because of cooperation but because of how conflicting information is resolved in the multi-swarm setting. This conflicting information is reconciled in such a way that C_{new} is always a Pareto improvement. As a result we propose changing the role of the $gbest$ in PSO from being the best actual particle seen so far to being a blackboard communication mechanism between *particles* instead of *swarms*. We hypothesize that if the global best were constructed in a similar way, the performance would be on a par or better than FEA-PSO and certainly better than PSO. The reason for the first claim is that by not factoring the variables, we would avoid pseudo-optima. The reason for the second claim is that by using the global best as a blackboard and resolving information conflicts between particles in a Pareto improving way, we avoid hitchhiking. We call this algorithm Pareto Improving Particle Swarm Optimization (PI-PSO).

The difference between PSO and PI-PSO is fairly minimal, but we claim that the effect is significant. The basic PSO remains exactly the same except in how the global best is constructed. The pseudocode is shown in Algorithm 2. Basically, the algorithm takes the current global best, all the current personal bests and begins by taking x_1 out of each particle's personal best and trying it in the global best. At the end of the loop, x_1 is either the value we started with or it is the best one out of all the particles. This process repeats for all the variables, $x_2 ... x_d$. Unlike FEA-PSO, however, we do not practice *elitism*, thus the $gbest$ is never an actual particle in the swarm. The global best thus determined by the entire swarm need not be a single particle out of the swarm.

Unfortunately, as we have previously mentioned, we cannot be guaranteed of a Pareto efficient $gbest$. If we have a swarm of 320 particles with 32 dimensions, we would need to test 320^{32} combinations to find the best use of the information contained on each variable in the swarm. So while we admit the theoretical existence of a Pareto *Efficient* Particle Swarm

Algorithm 2 PI-PSO Select Global Best

Input: Function f to optimize, current global best p_{gbest}, current personal bests, $\mathbf{p_{pbest}}$.

Output: New p_{gbest}.

```
1:  for $i = 1$ to $d$ do
2:      $fitness \leftarrow f(p_{gbest})$
3:      $value \leftarrow p_{gbest}.x_i$
4:      for $j = 1$ to $n$ do
5:          $candidate \leftarrow p^j_{pbest}.x_i$
6:          $p_{gbest}.x_i \leftarrow candidate$
7:          $candidate\_fitness \leftarrow f(p_{gbest})$
8:          if $candidate\_fitness$is better than$fitness$ then
9:              $fitness \leftarrow candidate\_fitness$
10:             $value \leftarrow candidate$
11:         end if
12:         $p_{gbest}.x_i \leftarrow value$
13:     end for
14: end for
15: return $p_{gbest}$
```

TABLE IV
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, Schefel-1.2

Optimization (PE-PSO) algorithm, we hope to work towards better approximations of it. We will return to a discussion of this PE-PSO in Section VI.

V. EXPERIMENTS

In order to test our hypothesis that the PI-PSO algorithm would be at least as good as the FEA-PSO and better than the basic PSO, we ran a large number of experiments on standard benchmark functions. The following sections describe the design, results and discussion of those experiments.

A. Design

We selected Benchmark optimization problems from [8] and [19] that were scalable to multiple dimensions. The problems selected are shown in Table IV, arranged by categories inspired by [10]. All of the problems are minimization problems, and almost all have a minimizing solution and value of $f([0]^d) = 0$. The notable exceptions are Exponential, which has a minimum at $[-1]^d$, and Eggholder, which has a dimension-dependent minimum and minimizing vector. Additionally, Sphere is *separable*, which means each dimension could be optimized individually. All of the other functions are non-separable in their current forms.

Each algorithm was run against each problem 50 times and the average minimum value discovered was recorded. Each problem was instantiated with 32 dimensions, $d = 32$. The results were then bootstrapped 500 times to estimate 95% confidence intervals/credible intervals [5]. Following [6] each algorithm used the same number of candidate solutions. In this case we chose 10 particles per dimension or $d \times 10 = 320$.

The PSO, PI-PSO, and PSO portion of the FEA-PSO all used the same parameters: $\omega = 0.729$ and $\phi_1 = \phi_2 = 1.49618$. Both the PSO and PI-PSO were run for 100 iterations. The FEA-PSO was run for 20 FEA iterations separated by 5 PSO iterations for a total of 100 PSO iterations. The FEA-PSO used a "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 [17]. With $d-1$ such factors, and $d = 32$, there were $\lfloor(320/31)\rfloor = 10$ particles per swarm for the FEA-PSO.

B. Results

The results of the experiments are reported in Tables V-IX for the mean minimum found by each algorithm during each experiment. The 95% confidence/credible intervals are show in parentheses. The algorithms with the best performance are shown in bold, if there is a clear winner, or italics for multiple winners. We now examine the results by Benchmark category.

It is no surprise that all the algorithms did well on the Bowl benchmarks (Exponential, Sargan and Sphere). All of these functions have a single global optimum without any trapping local optima. All three algorithms tied on the Sphere function, learning it perfectly without variance. The same cannot be said for Sargan where FEA-PSO did better than the others, or for Exponential where FEA-PSO and PI-PSO were tied for best performance. In all cases, PI-PSO did better than PSO.

In stark contrast to the Bowl benchmarks, the six benchmark functions in the Local Optima category are highly irregular (Ackley-1, Eggholder, Griewank, Rastrigin, Salomon, and Stretched-V). Consistent with previous research [17], FEA-PSO usually did better than PSO (four out of six). However, PI-PSO beat the other algorithms on five of six benchmarks.

The Plate benchmarks (Brown, Schwefel-2.23, Whitley, and Zakharov) have large, flat plateaus over their domains with spiky minima. FEA-PSO and PI-PSO each split these benchmarks with FEA-PSO performing better on Brown and PI-PSO performing better on Whitley. PI-PSO tied on each of the other functions. On Schwefel-2.23, PI-PSO tied with FEA-PSO. On Zakharov, PI-PSO tied with PSO. PI-PSO did better than PSO on all benchmarks except Zakharov.

Michalwicz, Schaffer-F6 and Schwefel-2.22 are all Ridge benchmarks characterized by sharp drop offs at various points in their domain. The PI-PSO did better on all of these benchmarks than the other algorithms.

Finally, we have the Valley benchmark functions that look like a tilted tube sawed in half (Dixon-Price, Rosenbrock, Schwefel-1.2). The PI-PSO again did better than all the other algorithms on these problems.

Overall, PI-PSO was the strongest performing algorithm. It was the best performing algorithm in 12 out of 19 benchmarks with three ties (Exponential, Sphere, Zakharov) and four losses (Sargan, Salomon, Brown, Schwefel-2.23). Comparing solely

TABLE V
"Bowl" Benchmark Optimization Problems

Benchmark	PSO	FEA-PSO	PI-PSO
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)
sargan	9.76e+00 (8.55e+00, 1.11e+01)	**2.04e-12** (1.37e-12, 2.91e-12)	1.55e-06 (1.16e-06, 2.00e-06)
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)

TABLE VI
"Local Optima" Benchmark Optimization Problems

Benchmark	PSO	FEA-PSO	PI-PSO
ackley-1	1.84e+00 (1.77e+00, 1.90e+00)	2.81e-03 (5.23e-06, 7.00e-03)	**4.44e-16** (4.44e-16, 4.44e-16)
eggholder	-1.68e+04 (-1.71e+04, -1.64e+04)	-1.77e+04 (-1.79e+04, -1.74e+04)	**-2.38e+04** (-2.40e+04, -2.35e+04)
griewank	4.96e-01 (4.47e-01, 5.45e-01)	9.49e-01 (8.99e-01, 9.91e-01)	**1.08e-01** (8.07e-02, 1.40e-01)
rastrigin	1.03e+02 (9.59e+01, 1.11e+02)	2.60e-02 (1.97e-05, 6.59e-02)	**0.00e+00** (0.00e+00, 0.00e+00)
salomon	**1.41e+00** (1.33e+00, 1.48e+00)	2.29e+00 (2.12e+00, 2.49e+00)	1.69e+00 (1.57e+00, 1.79e+00)
stretched-v	1.20e+01 (1.14e+01, 1.28e+01)	4.37e+00 (3.78e+00, 5.02e+00)	**2.84e+00** (2.59e+00, 3.04e+00)

TABLE VII
"Plate" Benchmark Optimization Problems

Benchmark	PSO	FEA-PSO	PI-PSO
brown	7.56e+00 (6.69e+00, 8.55e+00)	**1.22e-25** (3.72e-26, 2.42e-25)	1.23e-09 (1.01e-09, 1.45e-09)
schwefel-2.23	2.44e-01 (1.18e-01, 4.15e-01)	*8.65e-102* (7.25e-116, 2.31e-101)	*5.35e-41* (0.00e+00, 1.63e-40)
whitley	9.82e+02 (9.67e+02, 9.99e+02)	3.51e+02 (2.97e+02, 3.91e+02)	**7.51e+00** (6.20e+00, 8.76e+00)
zakharov	*1.39e+02* (1.28e+02, 1.51e+02)	1.60e+03 (3.09e+02, 3.77e+03)	*1.41e+02* (1.28e+02, 1.57e+02)

TABLE VIII
"Ridge" Benchmark Optimization Problems

Benchmark	PSO	FEA-PSO	PI-PSO
michalewicz	-8.65e+00 (-9.02e+00, -8.33e+00)	-2.59e+01 (-2.63e+01, -2.56e+01)	**-3.19e+01** (-3.19e+01, -3.19e+01)
schaffer-f6	2.49e+00 (2.31e+00, 2.67e+00)	1.99e+00 (1.78e+00, 2.20e+00)	**4.85e-01** (4.20e-01, 5.40e-01)
schwefel-2.22	3.01e+02 (2.91e+02, 3.13e+02)	1.22e-12 (7.73e-13, 1.75e-12)	**0.00e+00** (0.00e+00, 0.00e+00)

TABLE IX
"Valley" Benchmark Optimization Problems

Benchmark	PSO	FEA-PSO	PI-PSO
dixon-price	4.23e+01 (2.54e+01, 6.34e+01)	1.18e+02 (1.07e+02, 1.28e+02)	**8.65e-05** (7.44e-05, 9.85e-05)
rosenbrock	1.95e+02 (1.56e+02, 2.51e+02)	2.20e+02 (1.69e+02, 2.88e+02)	**9.64e-01** (6.73e-01, 1.26e+00)
schwefel-1.2	7.96e+03 (7.19e+03, 8.84e+03)	6.59e+04 (5.72e+04, 7.60e+04)	**3.28e+02** (2.86e+02, 3.68e+02)

to PSO, however, PI-PSO was the best performer for 16 of 19 benchmarks with two ties (Sphere, Zakharov) and one loss (Salomon). PI-PSO performed better than FEA-PSO on 15 of 19 benchmarks, losing twice (Brown, Sargan) and tying twice (Sphere and Schwefel-2.23).

On the other hand, FEA-PSO was better than PSO on 12 of 19 benchmarks, tying twice (Sphere, Rosenbrock) and losing five times (Dixon-Price, Griewank, Salomon, Schwefel-1.2, Zakharov). PSO performed best only on Salomon, tied on three (Sphere, Zakharov, Rosenbrock), and lost the rest.

C. Discussion

As we hypothesized, the PI-PSO was much better than the PSO, beating the basic algorithm on 16 of 19 benchmarks. We attribute this both to the elimination of hitchhiking and the lack of pseudo-minima in the PI-PSO algorithm.

PI-PSO also did well when compared to FEA-PSO, beating it on 15 of 19 benchmarks. Given that FEA-PSO did so well against PSO (12 out of 19 benchmarks), it is difficult to attribute all of this success to pseudo-minima in FEA-PSO that were avoided in PI-PSO, though this may have played a part. What we hypothesize instead is that although both algorithms use the same basic algorithm to construct their context, C for FEA-PSO and p_{gbest} for PI-PSO, PI-PSO has significantly more information to evaluate because instead of working with relatively few factors representing only partial solutions, PI-PSO works with many particles representing entire solutions.

Although all the algorithms start with the same number of candidate solutions, PI-PSO definitely requires more fitness evaluations. If p is the number of particles, d the number of dimensions, s the number of swarms (factors), m the number of FEA iterations, and i the average width of a factor, then our estimates of fitness evaluations per PSO iteration are:

$$\begin{aligned}\text{PSO} &= p \\ \text{FEA-PSO} &= ps + \frac{di + ps + 1}{m} \\ \text{PI-PSO} &= dp + p + 1\end{aligned}$$

Given the difficulty of comparing algorithms with different structures, information and information processing, it is difficult to say that fitness evaluations are a fair means of comparison which is why we looked at candidate solutions. Different algorithms use the information differently. However, ultimately all factors should be presented so that users of these algorithms can make informed choices.

VI. Conclusion

In this paper we discussed PSO and how hitchhiking arises through the search for an ever better candidate solution in the $gbest$. The discussion followed with an analysis of a class of algorithms that have addressed the hitchhiking problem through cooperation. These algorithms include CPSO—descended from CCGA—and FEA-PSO, of which CPSO is a special case. Although experimental results have shown FEA-PSO to be generally better than PSO, our analysis does not support the conclusion that the elimination of hitchhiking actually comes from cooperation as opposed to competition. Instead we proposed that it is through Blackboard-based information sharing and the Pareto improving nature of conflict resolution that eliminates hitchhiking. In order to test this hypothesis, We extended these concepts to PSO itself, creating the PI-PSO, which in a sense actually maximizes competition not cooperation.

PI-PSO treats the p_{gbest} as a blackboard for inter-particle communication rather than simply a record of the best of the best particles. We hypothesized that if the $gbest$ was calculated in the same way that C is calculated in FEA-PSO, the PI-PSO would out-perform the conventional $gbest$ PSO. We ran multiple experiments on PSO, FEA-PSO and PI-PSO. PI-PSO outperformed PSO in 16 out of 19 functions.

We see a significant amount of future research to be done on both PI-PSO and the Pareto Efficiency approach to analyzing similar algorithms. We would like to see how PI-PSO performs with different numbers of particles and on problems of different dimensions. We also would like to investigate the performance of PI-PSO on discrete problems. Although PI-PSO outperformed FEA-PSO, we need to consider to some degree the number of fitness evaluations required. We hypothesize that there may be a way to harness the multi-swarm approach, both to decrease the computational requirements and potentially to bridge the gap between PI-PSO and the theoretical PE-PSO. Finally, we wish to apply Pareto efficiency analysis to Distributed Factored Evolutionary Algorithms (DFEA) to unravel puzzles in the theoretical and empirical behavior of the algorithm as compared to FEA.

References

[1] M. Abido. Multiobjective particle swarm optimization for environmental/economic dispatch problem. *Electric Power Systems Research*, 79(7):1105–1113, 2009.

[2] U. Baumgartner, C. Magele, and W. Renhart. Pareto optimality and particle swarm optimization. *IEEE Transactions on Magnetics*, 40(2):1172–1175, 2004.

[3] S. H. Clearwater, B. A. Huberman, and T. Hogg. Cooperative problem solving. In B. Huberman, editor, *Computation: The Micro and the Macro View*, pages 33–70. World Scientific, Singapore, 1992.

[4] T. Cura. Particle swarm optimization approach to portfolio optimization. *Nonlinear analysis: Real world applications*, 10(4):2396–2406, 2009.

[5] B. Efron. Bootstrap methods: Another look at the jackknife. *Ann. Statist.*, 7(1):1–26, 01 1979.

[6] A. Engelbrecht. Fitness function evaluations: A fair stopping condition? In *Proceedings of the IEEE Swarm Intelligence Symposium (SIS)*, pages 1–8, 2014.

[7] R. Engelmore. *Blackboard Systems*. Addison Wesley, Reading, MA, 1988.

[8] M. Jamil and X.-S. Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013.

[9] M. T. Jensen. Guiding Single-Objective Optimization Using Multi-objective Methods. In *Applications of Evolutionary Computing. Evoworkshops 2003: EvoBIO, EvoCOP, EvoIASP, EvoMUSART, EvoROB, and EvoSTIM*, pages 199–210, Essex, UK, April 2003. Springer. Lecture Notes in Computer Science Vol. 2611.

[10] S. Jurjanovic and D. Bingham. Optimization test problems. url-https://www.sfu.ca/ ssurjano/optimization.html, 2013.

[11] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[12] M. Mitchell, S. Forrest, and J. H. Holland. The royal road for genetic algorithms: Fitness landscapes and GA performance. In *Proceedings of the first European conference on artificial life*, pages 245–254. Cambridge: The MIT Press, 1992.

[13] G. Moslehi and M. Mahnam. A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics*, 129(1):14–22, 2011.

[14] F. Neumann and I. Wegener. Can Single-Objective Optimization Profit from Multiobjective Optimization? In *Multi-Objective Problem Solving from Nature: From Concepts to Applications*, pages 115–130. Springer, Berlin, 2008. ISBN 978-3-540-72963-1.

[15] K. E. Parsopoulos and M. N. Vrahatis. *Particle Swarm Optimization and Intelligence: Advances and Applications*. Information Science Reference - Imprint of: IGI Publishing, Hershey, PA, 2010.

[16] M. A. Potter and K. A. De Jong. A cooperative coevolutionary approach to function optimization. In *Parallel Problem Solving from Nature—PPSN III*, pages 249–257. Springer, 1994.

[17] S. Strasser, N. Fortier, J. Sheppard, and R. Goodman. Factored evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 21(3):281–293, 2017.

[18] F. Van Den Bergh and A. P. Engelbrecht. Training product unit networks using cooperative particle swarm optimisers. In *Proceedings of International Joint Conference on Neural Networks*, volume 1, pages 126–131, 2001.

[19] F. Van den Bergh and A. P. Engelbrecht. A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):225–239, 2004.

[20] L. Wang and C. Singh. Environmental/economic power dispatch using a fuzzified multi-objective particle swarm optimization algorithm. *Electric Power Systems Research*, 77(12):1654–1664, 2007.

[21] G. Zhang, X. Shao, P. Li, and L. Gao. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering*, 56(4):1309–1318, 2009.