Evaluating Factored Evolutionary Algorithm Performance on Binary Deceptive Functions

Shane Strasser and John W. Sheppard
Gianforte School of Computing
Montana State University
Bozeman, MT 59717-3880

***Abstract*—Factored Evolutionary Algorithms (FEA) have been shown to be an effective method to optimize objective functions by partitioning the search space into overlapping subpopulations, or factors. FEA is comprised of three main steps: Update, Compete, and Share. While there exists previous work exploring FEA's convergence properties, it is still unknown where FEA obtains most of its performance gains. In this paper, we examine FEA performance by evaluating FEA on a set of commonly used deceptive unitation functions as well as several versions of the Royal Road problem. These problems provide a complex landscape for the search algorithm to explore but are well understood and researched. In evaluating FEA on these problems, we discovered that FEA's Compete step contributes the most to its performance effectiveness. Additionally, we identify a class of problems that may degrade the performance of FEA.**

I. Introduction

Combinatorial and nonlinear optimization remain among the most difficult yet important classes of problems being addressed today by advanced, heuristic search methods [1]. One such approach to heuristic search often used to solve such problems is defined by the class of population-based algorithms, including Evolutionary Algorithms and swarm-based algorithms, because the randomness used in the search process helps to escape local optima.

Factored Evolutionary Algorithms (FEA) are a generalization of cooperative coevolutionary algorithms [2], [3] that allow for subpopulations to overlap with one another [4]. FEA decomposes the optimization problem into a set of factors and encourages the factors to overlap, which allows the factors to compete with one another for inclusion in a full solution. Another unique property of FEA is that it allows for any population based algorithm to be used as the underlying optimization algorithm [5]. Consequently, subpopulations or subswarms are associated with each factor, and the desired search algorithm is used to search the subspace covered by that factor. Furthermore, creating the factors is based upon grouping variables that are highly correlated [4], [6].

Previous work by Strasser *et al*. demonstrated that the best-performing factor architecture also maintained better diversity between the individuals [6]. However, it is unknown in this case how the increased diversity relates to the improved performance. Furthermore, the question still remains as to whether FEA's performance is influenced most by creating subpopulations through the factoring of the function or by competition and collaboration among factors. Specifically, if a factor architecture is poorly constructed, will FEA still be able to locate good solutions?

We explore three questions in this paper. The first is what types of problems individual factors are able to solve. Second, we examine how effective competition is in combining solutions from the overlapping factors. Finally, we investigate if there exist guidelines for creating factors for problems without intuitive methods for factoring the function.

To investigate these questions, we explore how FEA performs on two sets of functions with binary variables: *unitation* functions, which are functions whose fitness is based on the number of zeros and ones in the input vector, and the Royal Road functions, which were proposed by Mitchell *et al*. to explore the building block hypothesis in GAs [7]. Even though these functions themselves do not have complex definitions, they are still able to define complex search spaces [8], and the functions contain properties found in real-world problems.

Previous work by Strasser *et al*. analyzed the performance of FEA on performing abductive inference on Bayesian networks and maximizing NK landscapes [4]. Unfortunately, since these problems are NP-complete, it is difficult to determine performance of various algorithms relative to global optimality on reasonably-sized instances of these problems. By using unitation and Royal Road functions, we are able to analyze the number of evaluations FEA requires directly to find the optimal solution. This also allows us to observe how robust FEA is to deception.

The rest of the paper is organized as follows. We begin by first reviewing related work in Section II, followed by a review of FEA in Section III. Section IV presents the experimental setup followed by the results for unitation functions in Section V and the results for the Royal Road functions in Section VI. An analysis of the results is given in Section VII. We conclude the paper with a summary of the work presented in this paper and several areas of future work in Section VIII.

II. Related Work

The earliest version of FEA was named Particle-based Routing with Overlapping Swarms for Energy Efficiency (PROSE) and was developed by Haberman and Sheppard [9]. PROSE was first developed to create an energy-aware routing protocol for sensor networks that ensures reliable path selection while minimizing energy consumption during message transmission. In their work, the authors let each subswarm represent a sensor

978-1-5386-2726-6/17/$31.00 ©2017 IEEE

node and all of the sensor's neighbors in the network. They were able to show that PROSE extends the life of the sensor networks by performing significantly better than energy-aware routing protocols that were state-of-the-art at the time.

Later work by Ganesan Pillai and Sheppard extended PROSE to learn the weights of deep artificial neural networks [10]. In that work, the authors developed an algorithm called Overlapping Swarm Intelligence (OSI) where each swarm corresponds to a unique path from input to output in the network. A common vector of weights is maintained across the swarms to describe a global view of the network, which is created by combining the weights of the best particles in each of the swarms. The authors showed that OSI outperformed several other PSO-based algorithms, as well as standard back-propagation, on deep networks.

Subsequently, a distributed version of OSI was developed by Fortier *et al*. called Distributed Overlapping Swarm Intelligence (DOSI) [11]. With DOSI, a communication and sharing algorithm was defined so that swarms could share values while also competing with one another. The key distinction from OSI was that a single global solution was not maintained for fitness evaluation. Their results showed that DOSI's performance was close to that of OSI's on several different networks.

OSI and DOSI have also been applied to the full and partial abductive inference problems in Bayesian networks, where the task is to find the most probable set of states for a set of nodes in the network, given a set of observations [5], [12] The authors were able to show that both OSI and DOSI outperformed several other population-based and traditional inference algorithms, including PSO, GA, simulated annealing, stochastic local search, and mini-bucket elimination.

Other applications of OSI and DOSI include learning the parameters or structure of Bayesian networks. For example, Fortier *et al.* adapted OSI to learn the structure of Bayesian classifiers by allowing subswarms to learn the links for each variable in the network [13]. The authors were able to show that in most cases OSI was able to significantly outperform competing structure learning approaches.

When learning Bayesian networks, latent or unobserved variables are often introduced, and Fortier *et al.* used OSI to learn the parameters of these latent variables [14]. A subswarm is created for each node with unknown parameters and all of the variables in that node's Markov blanket. The authors were able to show that OSI outperformed competing approaches, including the traditionally-applied expectation-maximization algorithm, and that the amount of overlap between the subswarms can impact the performance of OSI.

Strasser *et al*. were the first to define FEA as a general approach that can utilize any stochastic search algorithm [4]. Additionally, the authors were able to show that the best factor architecture is one that groups variables that have a high correlation. Later work by Strasser and Sheppard was able to prove that FEA converges to a single solution. Additionally, the authors were able to prove that FEA may become stuck in suboptimal solutions, called *pseudominima*, but that such behavior can be made rare with the right architecture [15].

III. Factored Evolutionary Algorithms

FEA is a class of optimization algorithm that functions by factoring the objective function. FEA is similar to the cooperative EAs like CCGA and CPSO; however, FEA encourages subpopulations to overlap with one another, allowing the subpopulations to compete and share information. Because FEA is able to work with any stochastic search algorithm, we were able to show that the FEA class includes CCPSO, CCGA, OSI, island EAs, and others. Here, we review a general definition of the FEA model presented by Strasser *et al*. [4]. For a more in-depth explanation, including pseudo-code, we refer the reader to the previously mentioned paper.

There are three major subfunctions in FEA: Update, Compete and Share. The Update function is the simplest and applies the base heuristic search algorithm to its local set of variables. The Compete function constructs a full solution from the factors that can be used by the factors to evaluate a partial solution, while the Share step uses this full solution to inject information back into the factors.

Given a function $f(\mathbf{X})$ to be optimized with $\mathbf{X} = \langle X_1, X_2, \ldots, X_n \rangle$, let $\mathbf{S}_i$ be some subset of $\mathbf{X}$. We call $\mathbf{S}_i$ a *factor* for which we will define a subpopulation. FEA then optimizes f over the variables in $\mathbf{S}_i$ by holding $\mathbf{R}_i = \mathbf{X} \setminus \mathbf{S}_i$ constant. In FEA we encourage the factors to be proper subsets of $\mathbf{X}$ where at least one factor overlaps with another factor. Without loss of generality, we assume every factor overlaps some other factor.

Because each subpopulation is optimizing over a subset of values in $\mathbf{X}$, the subpopulation defined for $\mathbf{S}_i$ needs to know the values of $\mathbf{R}_i$ for the local fitness evaluations. Given a factor $\mathbf{S}_i$ and its remaining values $\mathbf{R}_i$, fitness for a partial solution in $\mathbf{S}_i$ can be determined by calculating $f(\mathbf{S}_i \cup \mathbf{R}_i)$. The values for $\mathbf{R}_i$ are derived from the other subpopulations, which allows $\mathbf{S}_i$ to use values optimized by these other subpopulations.

The goal of the Compete step in FEA is to find the subpopulations with the state assignments that have the best fitness for each variable. FEA accomplishes this by constructing a global solution vector $\mathbf{G} = \langle X_1, X_2, \ldots, X_n \rangle$ that evaluates the optimized values from subpopulations. A variety of methods can be used to construct $\mathbf{G}$, but we found a greedy approach can be effective [4].

The Share step serves two purposes. First, it allows overlapping subpopulations to inject share knowledge with their neighbors. Previous work by Fortier *et al*. believed that this is one of the largest contributors to the FEA's performance [5]. Second, the Share step sets each factor's $\mathbf{R}_i$ values so that each $\mathbf{S}_i$ can evaluate its partial solution on f. Here, we use the method in Strasser *et al*. [4] where each $\mathbf{S}_i$ is seeded with values from $\mathbf{G}$.

The entire FEA algorithm works as follows. First, all of the subpopulations $\mathbf{S}_i$ are initialized. Next FEA searches by iterating over each subpopulation and having each optimize over its variables for a set number of iterations. Then $\mathbf{G}$ is updated by the Compete step. Finally, the Share step is performed by seeding each subpopulation with values from $\mathbf{G}$.

These three steps (Update, Compete, and Share) are repeated until some stopping criterion is satisfied, upon which the full global solution $\mathbf{G}$ is returned as the final solution.

IV. Experiments

To gain a better understanding of its performance, we evaluate FEA on a set of unitation and Royal Road functions. Unitation functions are functions in which the fitness is based upon counting the number of 1's. Royal Road functions are variations of unitation functions in that, in addition to the fitness being based upon the number of 1's, a specific structure is imposed upon the fitness landscape.

A. Unitation Functions

A unitation function is a fitness function whose output is based upon the number of ones in a binary array with no restriction on where the ones in the array occur. Let $u(\mathbf{X})$ be the count of the number of ones in a binary array $\mathbf{X}$

$$u(\mathbf{X}) = \sum_{i=1}^{N} X_i$$

where X_i is the ith binary variable in $\mathbf{X}$. The simplest unitation fitness function is referred to as *One Max* (OM), and is defined as

$$f(\mathbf{X}) = u(\mathbf{X})$$

The optimal solution consists of all 1's.

Two variations of OM are the *Needle* (N) and *Bi-Needle* (BN) problems. N consists of a search space with a large flat basin and a narrow optimal solution when the input contains all 1's, defined as

$$f(\mathbf{X}) = \begin{cases} 1+\alpha & u(\mathbf{X}) = N \\ 1 & \text{otherwise.} \end{cases}$$

where $\alpha > 0$ is some constant. BN, on the other hand, contains two narrow optimal solutions, one with all 0's and one with all 1's, and is defined as

$$f(\mathbf{X}) = \begin{cases} 1+\alpha & u(\mathbf{X}) = 0 \\ 1+\alpha & u(\mathbf{X}) = N \\ 1 & \text{otherwise.} \end{cases}$$

In this paper, we are interested in understanding the ability of FEA to handle deceptive problems. A deceptive function is one where multiple paths to local optimum solutions exist; however, one of more of those paths lead specifically to an inferior local optimum. Three deceptive versions of N and BN exists: *DecTrap* (DT), *TwoTrap* (TT), and *DecTwoTrap* (DTT).

DT contains one global optimum with all 1's and one suboptimal solution all 0's. Additionally, the function slope is defined such that the search will be biased towards the suboptimal solution. Formally, it is defined as

$$f(\mathbf{X}) = \begin{cases} N & u(\mathbf{X}) = N \\ N - 1 - u(\mathbf{X}) & \text{otherwise.} \end{cases}$$

On this function, search will be led in one of two directions: either toward all 0's or all 1's.

TT extends DT to contain two optimal solutions: all 0's and all 1's. However, a suboptimal solution is centered in the middle of the search space with $N/2$ 1's. Similar to DT, much of the search space guides the search algorithm toward the suboptimal solution. The function is defined as

$$f(\mathbf{X}) = \begin{cases} N - \frac{N}{2}u(\mathbf{X}) & u(\mathbf{X}) < \frac{N}{5} \\ 2u(\mathbf{X}) - \frac{2}{5}N & \frac{N}{5} \leq u(\mathbf{X}) \leq \frac{N}{2} \\ \frac{8}{5}N - 2u(\mathbf{X}) & \frac{N}{2} < u(\mathbf{X}) \leq \frac{4}{5}N \\ \frac{N}{2}u(\mathbf{X}) - \frac{2}{5}N^2 & \frac{4}{5}N < u(\mathbf{X}). \end{cases}$$

DTT is the opposite of TT; it has one optimal solution, one suboptimal solution, and is defined as

$$f(\mathbf{X}) = \begin{cases} N & u(\mathbf{X}) = \frac{N}{2} \\ -2u(\mathbf{X}) + N - 2 & u(\mathbf{X}) < \frac{N}{2} \\ 2u(\mathbf{X}) - N - 2 & \frac{N}{2} < u(\mathbf{X}). \end{cases}$$

B. Royal Road Functions

The Royal Road is an optimization problem that was original intended to be easy for GAs to solve [7], [8]. The function is defined as

$$f(\mathbf{X}) = \sum_{s \in \mathcal{S}} c_s \sigma_s(\mathbf{X})$$

where $\mathbf{X}$ is a bit string and $\mathcal{S}$ defines a set of schemata. A schema is a string where each element is from $\{0, 1, \#\}$ that denotes a pattern in a bit string [7]. For example, a schema of $\#\ 1$ is a pattern that matches any string where the second bit is 1 and the first bit is either 0 or 1. If the bit string $\mathbf{X}$ contains a schema s, then $\sigma_s(\mathbf{X})$ returns 1; otherwise it returns 0. c_s defines a cost or weight for schema s.

When first presented, Mitchell *et al*. proposed two different Royal Road functions, denoted R1 and R2 respectively. It was thought that R1 would be easy for a GA to solve by providing intermediate schemata to guide the search process as "building blocks." R2, on the other hand, eliminates the intermediate schemata, thus making search more difficult. Contrary to expectation, the authors found that intermediate schemata slowed down the GA.

Here, we extend R1 to combine characteristics of the deceptive unitation functions with characteristics in the Royal Road, which we denote R3. functions We do this by introducing deceptive schemata into the set $\mathcal{S}$. This is done by adding a schema that match all 0's in locations that are offset of the schema that match all 1's. Previous results on factor architectures suggest a factor should be generated for interrelated groups of variables. In R3, the negative schema represents groups of related variables; however, those groups of related variable are designed to lead to suboptimal solutions.

C. FEA on Unitation Functions

In this section we focus on 10-bit unitation functions. While unitation and Royal Roads functions are usually tested with a GA, for our FEA experiments, we use a discrete version of PSO known as Integer and Categorical PSO (ICPSO) [16].

TABLE I
FACTOR ARCHITECTURES VARYING FACTOR SIZE WITH NO OVERLAP.

Name	(Num Factors, Size)
F	(1,10)
S_9	(1,9), (1,1)
S_8	(1,8), (1,2)
S_7	(1,7), (1,3)
S_6	(1,6), (1,4)
S_5	(2,5)
S_4	(2,4), (1,2)
S_3	(3,3), (1,1)
S_2	(5,2)

TABLE II
FACTOR ARCHITECTURES WITH OVERLAP.

Name	(Num Factors, Size)
F	(1,10)
F_2	(2,10)
F_3	(3,10)
F_4	(4,10)
F_5	(5,10)
F_6	(6,10)

TABLE III
FACTOR ARCHITECTURES VARYING FACTOR SIZE AND OVERLAP

Name	(Num Factors, Size)	Overlap
F	(1,10)	—
T_9	(2,9)	8
T_8	(2,8)	6
T_7	(2,7)	4
T_6	(2,6)	2
T_5	(2,5)	0

TABLE IV
TRIALS TO OPTIMUM WITH VARYING FACTOR SIZE

	OM	N	BN	DT	TT	DTT
F	200	200	200	13	126	200
S_9	200	200	200	4	200	200
S_8	200	200	200	7	180	200
S_7	200	200	200	10	115	200
S_6	200	200	200	15	58	200
S_5	200	200	200	13	24	200
S_4	200	200	200	5	101	200
S_3	200	200	200	5	84	196
S_2	200	200	200	8	88	175

1) Varying Factor Size: First we examine how the factor size with no overlap affects the performance of FEA on the base unitation functions. To do so, we generate factors of decreasing size, starting with a full PSO and decreasing the factor size by one. Because the focus is strictly on size, we do not allow the factors to overlap. When possible, we generate factors of equal size, such as two factors of size $N/2$. In the case where factors consist of four bits, we create two factors of size four and one of size two. Following this process gives us the set of factor architectures shown in Table I.

2) Varying the Number of Overlapping Factors: Next we evaluate how the number of overlapping factors affect the performance of FEA. Here, we are able to examine how effective Compete is at determining optimal values for the full global solution. Starting with two factors optimizing over all ten variables, we increase the number of factors from two to five. By increasing the number of factors, we are able to evaluate how FEA is able to utilize multiple values for each variable during Compete. Table II presents the different factor architectures.

3) Varying Both Factor Overlap and Size: In these experiments, we vary both factor size and the amount of overlap. We use two factors of equal size and vary the number of bits each factor is optimizing and the amount of overlap between factors. Starting with two factors of size nine, we position each factor at both ends of the ordered variables. In this case, factor one optimizes bits 1–9 while factor two optimizes bits 2-10, which gives an overlap of eight. We then decrease the size of each factor, starting with nine and moving down to five, where there is no overlap between the factors. Table III presents the five different architectures.

D. FEA on Royal Road Functions

For the Royal Road functions, we used R1, R2, and R3 as defined above. Factor sizes were set to 2, 4, 8, and 16 with an overlap of 1, 2, 4, and 8, respectively. We also consider the case where there is no overlap, allowing us to replicate previous results by Ochoa *et al*. [17].

E. Experimental Setup

In all experiments, we use ICPSO as the underlying optimization algorithm as it has been shown to outperform other discrete PSO algorithms [16]. In all architectures, we use a budget of 50 individuals to distribute evenly between all the factors. We also consider a full, single-population ICPSO.

All of the unitation functions had ten bits. Each algorithm was run until the optimal solution was found; however, each algorithm was given a budget of 35,000 evaluations before terminating the algorithm on the unitation functions and 500,000 evaluations on the Royal Road functions. On the unitation functions, we performed 200 trials and on the Royal Road, we performed 30 trials. ICPSO completed one round of Updates before Compete and Share were performed. Each algorithm was initialized with randomly generated individuals. We note that this has the side affect of each of the factor architectures having individuals with different starting positions. However, because each architecture often contains individuals of different sizes, enforcing each architecture to have the individuals with same initial solutions is infeasible.

V. RESULTS: UNITATION FUNCTIONS

A. Varying Factor Size

Table IV shows the number of successful trials to find the optimal solutions, whereas Table V presents the average number of fitness evaluations excluding the failures required for each factor architecture to locate the optimal solution. The average number of fitness evaluations is calculated only from trials that were successful in locating the optimal solution. Additionally, the standard error is shown for the number of fitness evaluations.

TABLE V
AVERAGE FITNESS EVALUATIONS WITH VARYING FACTOR SIZE

	OM	N	BN	DT	TT	DTT
F	6.51E+2 (1.32E+1)	2.05E+3 (1.59E+2)	9.09E+2 (4.79E+1)	5.00E+2 (0.00E+0)	1.29E+3 (2.08E+2)	5.00E+2 (0.00E+0)
S_9	6.83E+2 (1.41E+1)	2.79E+3 (1.72E+2)	1.52E+3 (7.95E+1)	5.06E+2 (0.00E+0)	5.29E+3 (4.58E+2)	5.06E+2 (0.00E+0)
S_8	6.08E+2 (1.06E+1)	2.78E+3 (1.91E+2)	1.55E+3 (7.85E+1)	5.06E+2 (0.00E+0)	4.75E+3 (4.71E+2)	5.06E+2 (0.00E+0)
S_7	5.43E+2 (6.81E+0)	3.48E+3 (2.23E+2)	1.85E+3 (1.08E+2)	5.06E+2 (0.00E+0)	2.98E+3 (3.97E+2)	5.06E+2 (0.00E+0)
S_6	5.11E+2 (3.09E+0)	4.67E+3 (3.05E+2)	2.52E+3 (1.43E+2)	5.06E+2 (0.00E+0)	1.02E+3 (1.78E+2)	5.06E+2 (0.00E+0)
S_5	5.06E+2 (0.00E+0)	4.71E+3 (3.09E+2)	2.22E+3 (1.39E+2)	5.06E+2 (0.00E+0)	5.06E+2 (0.00E+0)	5.06E+2 (0.00E+0)
S_4	5.06E+2 (0.00E+0)	8.35E+3 (6.29E+2)	3.89E+3 (2.76E+2)	5.08E+2 (0.00E+0)	7.87E+2 (2.33E+1)	5.08E+2 (0.00E+0)
S_3	5.06E+2 (0.00E+0)	1.05E+4 (6.81E+2)	6.14E+3 (3.85E+2)	5.10E+2 (0.00E+0)	1.02E+3 (5.63E+1)	5.10E+2 (0.00E+0)
S_2	5.12E+2 (0.00E+0)	1.96E+4 (1.26E+3)	9.23E+3 (6.31E+2)	5.12E+2 (0.00E+0)	8.03E+2 (1.53E+2)	5.12E+2 (0.00E+0)

For the OM function, both the full ICPSO and every factor architecture were able to find the optimal solution every time. However, S_5, S_4, and S_3 were able to do so in the fewest iterations, requiring on average only one iteration. Conversely, on the N and BN functions, larger factors performed better. The full swarm performed the best, while the best FEA factor architecture was S_9. Notably, out of all base unitation functions, N required the most fitness evaluations.

On DT, the algorithms were able to locate the optimal solution only 5% of the time. In terms of locating the best solution, S_6 performed the best, finding the optimal solution 15 times. The worst architectures were S_3 and S_4, which found the optimal solution only five times. When the full ICPSO and FEA found the optimal solution, it was always in a single iteration. Additionally, the variety in the number of fitness evaluations for this function is caused by the extra fitness evaluations FEA performs during initialization.

The best architecture on TT was S_9, which found the optimal solution 100% of the time. However, it required the most fitness evaluations, roughly nine times the number of evaluations required needed by S_5, which had the lowest success rate out of all architectures. As the factor size decreased from S_9 to S_5, both the number of successful trials and number of evaluations steadily decreased. From S_5 to S_2, the number of successful trials and fitness evaluations increased.

Finally, on DTT, the larger factor architectures performed the best. The only two architectures that were unable to locate the optimal solution 100% of the time were S_3, and S_2. Similar to DT, FEA and the full swarm were always able to locate the optimal solution in one iteration with the differences in the number of fitness evaluations being caused by extra fitness evaluations during initialization.

B. Varying Factor Overlap

Table VI presents the number of successful trials required to find the optimal solution, whereas Table VII presents the average number of fitness evaluations required for each factor architecture to locate the optimal solution. The average number of fitness evaluations is calculated only from trials that were successful in locating the optimal solution. Additionally, the standard error is shown for the number of fitness evaluations.

We note that the best architecture on OM was F_4, which required the fewest number of fitness evaluations. On the other functions, FEA was often outperformed by the single-population ICPSO; however, this difference was not always significant. Only with DT and DTT is there a clear difference between F and F_2.

TABLE VI
TRIALS TO OPTIMUM WITH VARYING FACTOR OVERLAP

	OM	N	BN	DT	TT	DTT
F	200	200	200	15	200	200
F_2	200	200	200	9	200	200
F_3	200	200	200	9	200	200
F_4	200	200	200	16	200	200
F_5	200	200	200	15	200	200
F_6	200	200	200	15	200	200

Next there appears to be a correlation between the increase in both the number of factors and the number of fitness evaluations; however, there are a few exceptions, such as F_4 on N or F_3 and F_5 on BN. The only function that does not display this correlation is DT, where there is an increase from F to F_2, but then the number of evaluations levels out.

C. Varying Both Size and Overlap

Table VIII presents the number of successful trials for the various FEA factor architectures over each of the unitation functions and Table IX shows the average number of fitness evaluations for successful trials. The average number of fitness evaluations is calculated only from the trials that were successful in locating the optimal solution. Additionally, the standard error is shown for the number of fitness evaluations.

For OM, the factor architecture T_6 performed the best, requiring the fewest number of fitness evaluations. On N and BN, T_9 and T_8 tied with the full swarm (F); however, as the factor size and overlap decreased from T_8 to T_5, the number of fitness evaluations also increased.

DT was the most difficult function to optimize. On average, each factor architecture was able to locate the optimal solution only around 5% of the time. However, when FEA and the full ICPSO found the optimal solution, it was done in only one iteration. A similar result can be seen in DTT, where each architecture located the optimal solution within one iteration.

Finally, on TT, the best architectures were T_8 and T_9, which required the fewest number of fitness evaluations, taking only 11.5 iterations to find the optimal solution. As the amount of overlap decreases, the performance begins to decrease as well.

TABLE VII
AVERAGE FITNESS EVALUATIONS WITH VARYING FACTOR OVERLAP

	OM	N	BN	DT	TT	DTT
F	1.68E+3 (1.09E+2)	1.68E+3 (1.09E+2)	9.64E+2 (4.71E+1)	5.00E+2 (0.00E+0)	4.33E+3 (4.17E+2)	5.00E+2 (0.00E+0)
F_2	1.72E+3 (1.09E+2)	1.72E+3 (1.09E+2)	1.01E+3 (4.26E+1)	5.35E+2 (1.76E-1)	4.92E+3 (4.95E+2)	5.31E+2 (1.20E-1)
F_3	1.73E+3 (1.12E+2)	1.73E+3 (1.12E+2)	9.47E+2 (4.13E+1)	5.35E+2 (1.10E+0)	5.69E+3 (5.93E+2)	5.34E+2 (9.92E-2)
F_4	1.87E+3 (1.21E+2)	1.87E+3 (1.21E+2)	1.10E+3 (5.92E+1)	5.53E+2 (1.74E+1)	5.99E+3 (5.98E+2)	5.35E+2 (7.58E-2)
F_5	1.79E+3 (1.12E+2)	1.79E+3 (1.12E+2)	9.82E+2 (4.49E+1)	5.41E+2 (6.65E-1)	5.98E+3 (6.39E+2)	5.41E+2 (6.32E-2)
F_6	1.86E+3 (1.21E+2)	1.86E+3 (1.21E+2)	1.13E+3 (5.84E+1)	5.36E+2 (0.00E+0)	7.39E+3 (8.60E+2)	5.36E+2 (3.67E-2)

TABLE VIII
TRIALS TO OPTIMUM WITH VARYING BOTH FACTOR SIZE AND OVERLAP

	OM	N	BN	DT	TW	DTT
F	200	200	200	15	200	200
T_9	200	200	200	16	200	200
T_8	200	200	200	10	200	200
T_7	200	200	200	16	200	200
T_6	200	200	200	13	200	200
T_5	200	200	200	15	21	200

T_5 was the worst factor architecture, which was able to find the optimal solution only 21 times; however, when it did so, it required only one iteration.

VI. RESULTS: ROYAL ROAD FUNCTIONS

Table X presents the results for the different factor architectures on the Royal Road functions. "Success" refers to the number of times the algorithm was able to locate the optimal solution. "Evals" is the average number of evaluations required to find the optimal solution with the standard error shown in parentheses. A bold value indicates the factor architecture was significantly better than all other architectures using a Paired Student t-Test with $\alpha = 0.05$. If two or more values are bolded, there was no significant difference among those architectures, but they were significantly better than the rest. Note that the mean and standard error were calculated only for the trials that were able to locate the global optimums succesfully, which is why certain algorithms running on R3 have a standard erorr of "NA'.' In FEA, this also includes evaluations used during the competition phase. We note that each algorithm required 250 fitness evaluations during the initialization of the population.

On R1, full ICPSO was less effective than every factor architecture, only being able to locate the optimal solution 17 times. The best architectures had size = 8, overlap = 0; size = 16, overlap = 0;and size = 16, overlap = 8, which required the fewest number of fitness evaluations by a significant margin. On R2, full ICPSO had a much easier time locating the optimal solution, doing so 100% of the time. The best architecture had size = 16, overlap = 0, and the fewest number of evaluations.

R3, on the other hand, presented a much tougher challenge. Note that because several of the architectures were able to locate the optimal solution only once, hypothesis testing was not performed between the architectures. In terms of locating the optimal solution, both size =16, overlap = 8 and size =16, overlap = 0 had the best performance, as they successfully located the optimal solution four times. Additionally, when they did locate the optimal solution, they required the second-fewest number of fitness evaluations. The size = 8, overlap = 0 architecture performed the best, requiring only 8400 fitness evaluations; however, it did so only once.

To help interpret the results, particularly on R3, we also present the average fitness over all 30 trials of the Full and FEA versions of ICPSO in Table XI. From this table, we can see that the best architecture was size = 16, overlap = 8 followed by size = 8, overlap = 0. While most architectures achieved perfect performance on R1 and R2, these two architectures performed the best in terms of fitness for R3.

VII. ANALYSIS

From the experiments varying just the factor size, we observe the following. On OM, the smaller factors performed better than the larger factors. This supports our hypothesis that the variables in OM are independent. Conversely, the architectures with large factors outperformed those with smaller factors on all other functions. This suggests that although all functions are based upon the number of 0's and 1's, by requiring specific values, variable interactions result.

Another result is that on most functions, the best algorithm was the full ICPSO. Only on OM and TT did FEA outperform the single-population. FEA's superior performance on the OM is due to the Compete step, a Greedy algorithm that is able to join together the best solutions from individual factors quickly. The probability of a solution with ten 1's being generated is $\frac{1}{2^{10}} = 0.097\%$, whereas the probability of a subsolution with two 1's is $\frac{1}{4} = 25\%$. Given these odds, there is a high likelihood that each of the factors will contain the optimal subsolution, which Compete is then able to combine together into the optimal solution. Conversely, the full swarm can rely only on velocity updates to locate the optimal solution.

TT demonstrates the drawback of using a greedy method for Compete. As the population size decreases, a greater emphasis is placed on Compete being able to assemble good solutions. Because the majority of the search space leads to the suboptimal solution, the likelihood of Compete finding the optimal solution is small. Additionally, once Compete locates the suboptimal solution, the full global solution is unable to leave the suboptimal solution because finding a better solution requires changing more than one bit. Only when the full global solution is initialized in a region that leads to the optimal solution are the smaller factors able to find the optimum.

Meanwhile, FEA with the S_9 architecture is able to locate the optimal solution more often than full ICPSO because FEA

TABLE IX
AVERAGE FITNESS EVALUATIONS WITH VARYING BOTH FACTOR SIZE AND OVERLAP

	OM	N	BN	DT	TW	DTT
F	6.71E+2 (1.20E+1)	1.68E+3 (1.09E+2)	9.64E+2 (4.71E+1)	5.00E+2 (0.00E+0)	4.33E+3 (4.17E+2)	5.00E+2 (0.00E+0)
T_9	5.65E+2 (7.64E+0)	2.03E+3 (1.27E+2)	9.82E+2 (4.40E+1)	5.29E+2 (4.87E-1)	3.48E+3 (3.15E+2)	5.26E+2 (1.05E-1)
T_8	5.69E+2 (7.60E+0)	1.86E+3 (9.79E+1)	9.63E+2 (4.02E+1)	5.23E+2 (5.77E-1)	3.34E+3 (3.09E+2)	5.21E+2 (7.83E-2)
T_7	5.36E+2 (5.16E+0)	2.12E+3 (1.19E+2)	1.15E+3 (6.47E+1)	5.18E+2 (2.56E-1)	5.24E+3 (4.75E+2)	5.16E+2 (7.73E-2)
T_6	5.15E+2 (2.55E+0)	3.48E+3 (2.35E+2)	1.59E+3 (9.17E+1)	5.12E+2 (1.54E-1)	1.99E+4 (1.64E+3)	5.11E+2 (5.02E-2)
T_5	5.06E+2 (0.00E+0)	4.46E+3 (2.96E+2)	2.27E+3 (1.28E+2)	5.06E+2 (0.00E+0)	5.06E+2 (0.00E+0)	5.06E+2 (0.00E+0)

TABLE X
AVERAGE NUMBER OF EVALUATIONS FOR FEA TO FIND OPTIMAL SOLUTION ON ROYAL ROAD FUNCTIONS.

	R1		R2		R3	
	Success	Evals	Success	Evals	Success	Evals
Full	17	4.28E+4(7.99E+3)	30	3.24E+4(3.64E+3)	2	4.50E+4(3.34E+4)
Size = 2 Overlap = 0	29	2.54E+5(1.65E+4)	28	2.36E+5(2.09E+4)	1	5.36E+4(NA)
Size = 4 Overlap = 0	30	9.77E+4(7.70E+3)	30	1.08E+5(9.69E+3)	4	9.41E+4(4.24E+4)
Size = 8 Overlap = 0	30	**3.39E+4(3.41E+3)**	30	4.26E+4(3.99E+3)	1	8.34E+3(NA)
Size = 16 Overlap = 0	30	**2.28E+4(2.56E+3)**	30	**2.26E+4(2.12E+3)**	4	2.12E+4(6.05E+3)
Size = 2 Overlap = 1	27	3.27E+5(2.81E+4)	27	3.64E+5(2.16E+4)	1	2.48E+5(NA)
Size = 4 Overlap = 2	30	1.59E+5(1.48E+4)	30	1.36E+5(1.47E+4)	2	1.58E+5(1.77E+3)
Size = 8 Overlap = 4	30	7.27E+4(8.87E+3)	30	5.88E+4(5.19E+3)	1	1.05E+5(NA)
Size = 16 Overlap = 8	30	**4.56E+4(5.22E+3)**	30	3.57E+4(4.15E+3)	4	1.92E+4(5.73E+3)

TABLE XI
AVERAGE FITNESS OF FEA ON ROYAL ROAD FUNCTIONS.

	Full	Size = 2 Overlap = 0	Size = 4 Overlap = 0	Size = 8 Overlap = 0	Size = 16 Overlap = 0	Size = 2 Overlap = 1	Size = 4 Overlap = 2	Size = 8 Overlap = 4	Size = 16 Overlap = 8
R1	204.00	252.00	256.00	256.00	256.00	244.00	256.00	256.00	256.00
R2	128.00	123.20	128.00	128.00	128.00	120.80	128.00	128.00	128.00
R3	139.47	122.13	127.47	130.67	125.33	118.40	126.67	109.33	137.60

has a smaller search space to explore. Conversely, S_8 has less chance of success because it requires both of the bits in the smaller factor to be 0s or 1s, instead of just one variable being 0 or 1, as is the case in S_9.

There are two ways for FEA to locate the optimal solution: through an individual factor locating the solution independent of all factors or by Compete piecing together good values from individual factors. This leads us to two possible explanations for FEA's poor performance on the unitation functions. First, by distributing 50 individuals between the factors, the individual factors are not as efficient as the full ICPSO at finding the optimal solutions. Ideally, we would expect FEA to be able to overcome the difference between the factor sizes and the full global solution by having factors collaborating with one another. FEA performs this collaboration through Compete.

Second, the greedy search in Compete makes it impossible for the full global solution to escape a local basin when more than one variable must be changed simultaneously. Therefore, if the full global solution becomes located at a suboptimal point, FEA can only locate an optimal solution if the factors locate the optimal solutions on their own. But because each factor has fewer individuals than a single-population ICPSO, the likelihood of finding the optimal solution decreases.

When varying the amount of overlap factor size, we discovered the best factor architectures has two large factors with a large amount of overlap. In the majority of the functions, T_9 required the fewest fitness evaluations. Only on OM was T_9 outperformed by a significant margin. While T_9 required the most evaluations on DT and DTT, the architecture was still able to find the optimal solution within one iteration. Additionally, the number of evaluations was relatively small.

We believe that this is because the larger factors are able to balance the benefits of having a global view of the entire function while decreasing the number of bits to optimize. For example, in N for T_9, the last factor only has to find a solution that has a 1 for the last bit in order for the first factor to able locate the optimal solution. This is because the second factor is the only factor optimizing over the last bit. Likewise, the

first factor only has to find a solution with a 1 for the first bit for the second factor to be able to locate the optimal solution.

From these results, we conclude that Compete is not effective at piecing together good solutions in unitation functions that have a high degree of variable interaction. This is because Compete is able to change only one bit at a time; therefore, if multiple bits need to be changed simultaneously for the full global solution to move from a suboptimal solution, Compete is unable to do so. Finally, we found that smaller factors are effective only when high interaction occurs between variables. Then the bits are unable to interact efficiently with one another to locate good solutions.

In the Royal Road results, we observe that the larger factor architectures performed the best on all functions. We believe this is because these larger factors map more closely to the basic building blocks. For example, in size = 2, overlap = 0, an individual factor must rely on six other factors locating all 1's at the same time in order to satisfy the smallest building block. Conversely, larger factors are able to find the good schema without requiring that other factors be in specific positions.

We also note that the results in Table X differ from those of Ochoa *et al*. [17]. That work showed that the optimal architecture was size = 4, overlap = 0. However, we found that the best non-overlapping architecture was size = 16. As stated previously, we believe that the larger factor sizes allow the individual factors to have a better global picture of the fitness landscape.

On comparing the overlapping factors versus the non-overlapping factors, we discovered that overlap was almost always a detriment to the performance of FEA on the Royal Road functions. The only time overlap did not hurt performance was for size = 16, overlap = 8. This is because the overlapping factors are attempting to locate the two halves of the different schemata.

R3 further demonstrates that having factors optimize over two halves negatively impacts the performance. This function contained groups of "negative" schemata that caused all of the architectures to be misled. In size = 8, overlap = 4, this means that there are factors optimizing over the variables within the deceptive schema. A factor optimizing over those variables is more likely to locate a schema of 0's instead of one with all 1's, because the factor is able to see an increase in fitness independent of the values in the full global solution.

VIII. Conclusions and Future Work

In this paper, we evaluated the peformance of FEA on two sets of binary problems: unitation and Royal Road functions. From the unitation functions, we discovered that there are several instances where the Compete step in FEA struggles to build good solutions. In the Royal Road functions, we demonstrated for the regular Royal Road functions, the optimal factor architecture is one that maps factors directly to schemata. However, we also discovered scenarios where mapping factors directly to schemata decreased the performance FEA.

For future work, we first want to explore hierarchical factors. Based on the results presented here, we see that the size of the factor can influence the performance of the different architectures. We want to explore creating a hierarchical FEA for problems where factor sizes become too large to help avoid hitchhiking. In those cases, subfactors could be created, which would result in a FEA being composed of several FEAs.

We also plan to investigate the scalability of FEA. Previous work by Strasser *et al*. demonstrated that FEA can require a large number of fitness evaluations due to the competition step [4]. We will explore how to reduce complexity by using different competition schemes. Additionally, we plan to evaluate FEA on a wider range of functions to further determine FEA's performance characteristics.

References

[1] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. John Wiley & Sons, 2005.

[2] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Proceedings of the International Conference on Parallel Problem Solving from Nature*. Springer, 1994, pp. 249–257.

[3] F. Van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[4] S. Strasser, J. Sheppard, N. Fortier, and R. Goodman, "Factored evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 2, pp. 281–293, 2017.

[5] N. Fortier, J. Sheppard, and S. Strasser, "Abductive inference in bayesian networks using distributed overlapping swarm intelligence," *Soft Computing*, vol. 19, no. 4, pp. 981–1001, 2015.

[6] S. Strasser and J. Sheppard, "A formal approach to deriving factored evolutionary algorithm architectures," in *submitted to Swarm Intelligence Symposium*, 2017.

[7] 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*. Cambridge: The MIT Press, 1992, pp. 245–254.

[8] J. N. Richter, "On mutation and crossover in the theory of evolutionary algorithms," Ph.D. dissertation, Montana State University, 2010.

[9] B. K. Haberman and J. W. Sheppard, "Overlapping particle swarms for energy-efficient routing in sensor networks," *Wireless Networks*, vol. 18, no. 4, pp. 351–363, 2012.

[10] K. G. Pillai and J. Sheppard, "Overlapping swarm intelligence for training artificial neural networks," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS)*, 2011, pp. 1–8.

[11] N. Fortier, J. W. Sheppard, and K. Pillai, "DOSI: training artificial neural networks using overlapping swarm intelligence with local credit assignment," in *Proceedings of the Joint 6th International Conference on Soft Computing and Intelligent Systems (SCIS) and 13th International Symposium on Advanced Intelligent Systems (ISIS)*, 2012, pp. 1420–1425.

[12] N. Fortier, J. Sheppard, and K. G. Pillai, "Bayesian abductive inference using overlapping swarm intelligence," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS)*, 2013, pp. 263–270.

[13] N. Fortier, J. Sheppard, and S. Strasser, "Learning Bayesian classifiers using overlapping swarm intelligence," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS)*, 2014, pp. 1–8.

[14] ——, "Parameter estimation in Bayesian networks using overlapping swarm intelligence," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2015, pp. 9–16.

[15] S. Strasser and J. W. Sheppard, "Convergence of factored evolutionary algorithms," in *Proceedings of the Conference on Foundations of Genetic Algorithms (FOGA) XIV*. ACM, 2017, pp. 81–94.

[16] S. Strasser, R. Goodman, S. G. W. Butcher, and J. W. Sheppard, "A new discrete particle swarm optimization algorithm," in *Proceedings of the ACM Genetic and Evolutionary Computation Conference (GECCO)*, July 2016, pp. 53–60.

[17] G. Ochoa, E. Lutton, and E. Burke, "The cooperative royal road: avoiding hitchhiking," in *Proceedings of the International Conference on Artificial Evolution (Evolution Artificielle)*. Springer, 2007, pp. 184–195.