Self-Adaptive Multimethod Search for Global Optimization in Real-Parameter Spaces

Jasper A. Vrugt, Bruce A. Robinson, and James M. Hyman

Abstract—Many different algorithms have been developed in the last few decades for solving complex real-world search and optimization problems. The main focus in this research has been on the development of a single universal genetic operator for population evolution that is always efficient for a diverse set of optimization problems. In this paper, we argue that significant advances to the field of evolutionary computation can be made if we embrace a concept of self-adaptive multimethod optimization in which multiple different search algorithms are run concurrently, learn from each other through information exchange using a common population of points. We present an evolutionary algorithm, entitled A Multialgorithm Genetically Adaptive Method for Single Objective Optimization (AMALGAM-SO), that implements this concept of self adaptive multimethod search. This method simultaneously merges the strengths of the covariance matrix adaptation (CMA) evolution strategy, genetic algorithm (GA), and particle swarm optimizer (PSO) for population evolution and implements a self-adaptive learning strategy to automatically tune the number of offspring these three individual algorithms are allowed to contribute during each generation. Benchmark results in 10, 30, and 50 dimensions using synthetic functions from the special session on real-parameter optimization of CEC 2005 show that AMALGAM-SO obtains similar efficiencies as existing algorithms on relatively simple unimodal problems, but is superior for more complex higher dimensional multimodal optimization problems. The new search method scales well with increasing number of dimensions, converges in the close proximity of the global minimum for functions with noise induced multimodality, and is designed to take full advantage of the power of distributed computer networks.

Index Terms—Adaptive estimation, elitism, genetic algorithms, nonlinear estimation, optimization.

I. INTRODUCTION

Many real-world search and optimization problems require the estimation of a set of model parameters or state variables that provide the best possible solution to a predefined cost or objective function, or a set of optimal tradeoff values in the case of two or more conflicting objectives. Locating global optimal solutions is often painstakingly tedious, especially in the presence of high dimensionality, nonlinear parameter interaction, insensitivity, and multimodality of the objective function. These conditions make it very difficult for any search algorithm to find high-quality solutions quickly without getting stuck in local basins of attraction when traversing the search space en route to the global optimum. Unfortunately, these difficulties are frequently encountered in real-world search and optimization problems. In this paper, we consider single-objective optimization problems with \( n \) decision variables (parameters), and in which the parameter search space \( X \), although perhaps quite large, is bounded. We denote \( \mathbf{x} = (x_1, \ldots, x_n) \) as the decision vector, and \( \mathbf{y} = f(\mathbf{x}) \) as the associated objective function for a given function or model \( f \). Throughout this paper, we focus on minimization problems

\[
\min_{\mathbf{x} \in \mathbb{X}} \mathbf{y} = f(\mathbf{x}), \quad \mathbf{x} = (x_1, \ldots, x_n),
\]

In the last few decades, many different algorithms have been solved to find the minimum of the objective function in (1). Of these, evolutionary algorithms have emerged as a revolutionary approach for solving complex search and optimization problems. These methods are heuristic search algorithms and implement analogies to physics and biology to evolve a population of potential solutions through the parameter space to the global minimum. Beyond their ability to search enormously large spaces, these algorithms possess the ability to maintain a diverse set of solutions and exploit similarities of solutions by recombination. In this context, four different approaches have found widespread use: i) self-adaptive evolution strategies [3], [19], [34]; ii) real-parameter genetic algorithms [10], [11], [24]; iii) differential evolution methods [37]; and iv) particle swarm optimization (PSO) algorithms [13], [26]. These algorithms share a number of common elements, and show similarities in search principles on certain fitness landscapes [6], [27].

Despite this progress made, the current generation of optimization algorithms usually implements a single genetic operator for population evolution. For example, the majority of papers published in the proceedings of the recent special session on real-parameter optimization at CEC-2005, Edinburgh, U.K., describe methods of optimization that utilize only a single operator for population evolution. Exceptions include contributions that use simple self-adaptive [43] or memetic algorithms combing global and local search in an iterative fashion [29], [31]. Reliance on a single biological model of natural selection and adaptation presumes that a single method exists that can efficiently evolve a population of potential solutions through the parameter space and work well for a diverse set of problems. However, existing theory and numerical benchmark experiments have demonstrated that it is impossible to develop a single, universal algorithm for population evolution that is always efficient for a diverse set of optimization problems [42].

This is because the nature of the fitness landscape (objective function mapped out as a function of \( \mathbf{x} \)) can vary considerably between different optimization problems, and perhaps more
importantly, can change shape en route to the global optimal solution.

In a recent paper [41], we have introduced a new concept of self-adaptive multimethod evolutionary search. This approach entitled, A Multialgorithm Genetically Adaptive Multiobjective (AMALGAM) method, employs a diverse set of optimization algorithms simultaneously for population evolution, adaptively favoring individual algorithms that exhibit the highest reproductive success during the search. By adaptively changing preference to individual algorithms during the course of the optimization, AMALGAM has the ability to quickly adapt to the specific difficulties and peculiarities of the optimization problem at hand. Synthetic multimodal benchmark studies covering a diverse set of problem features have demonstrated that AMALGAM significantly improves the efficiency of evolutionary search, approaching a factor of 10 improvement over other available methods.

In this paper, we extend the principles and ideas underlying AMALGAM to single objective real-parameter optimization. We present an evolutionary search algorithm, called A Multi-algorithm Genetically Adaptive Method for Single Objective Optimization (AMALGAM-SO), which simultaneously utilizes the strengths of various commonly used algorithms for population evolution, and implements a self-adaptive learning strategy to favor individual algorithms that demonstrate the highest reproductive success during the search. In this paper, we consider the Covariance Matrix Adaptation (CMA) evolutionary strategy, Genetic Algorithm (GA), Particle Swarm Optimizer (PSO), Differential Evolution (DE), and Parental-Centric Recombination Operator (PCX) for population evolution. To test the numerical performance of AMALGAM-SO, a wide range of experiments are conducted using a selected set of standard test functions from the special session on real-parameter optimization of the IEEE Congress on Evolutionary Computations, CEC 2005. A comparison analysis against state-of-the-art single search operator, memetic, and multimethod algorithms is also included.

The remainder of this paper is organized as follows. Section II provides the rationale for simultaneous multimethod search with genetically or self-adaptive updating, and presents a detailed algorithmic description of the AMALGAM-SO method. Section III presents a new species selection mechanism to maintain useful population diversity during the search. After discussing the stopping criteria of the AMALGAM-SO algorithm in Section IV, the experimental procedure used to test the performance of this new method is presented in Section V. In Section VI, simulation results of AMALGAM-SO are presented in dimensions 10, 30, and 50 using the test suite of benchmark functions from CEC 2005. Here we focus specifically on which combination of algorithms to include in AMALGAM-SO. A performance comparison with a large number of other evolutionary algorithms is also included in this section, including an analysis of the scaling behavior of AMALGAM-SO, and a case study illustrating the performance of the method on a test function with noise induced multimodality. Finally, a summary with conclusions is presented in Section VII.

II. TOWARDS A SELF-ADAPTIVE MULTIMETHOD SEARCH CONCEPT

Much research in the optimization literature has focused on the development of a universal operator for population diversity that is always efficient for a large range of problems. Unfortunately, the No Free Lunch Theorem (NFL) of [42] and the outcome of many different performance studies ([22] amongst many others) have unambiguously demonstrated that it is impossible to develop a single search operator that is always most efficient on a large range of problems. The reason for this is that different fitness landscapes require different search approaches. In this paper, we embrace a concept of self-adaptive multimethod optimization, in which the goal is to develop a combination of search algorithms that have complementary properties and learn from each other through a common population of points to efficiently handle a wide variety of response surfaces. We will show that NFL also holds for this approach, but that self-adaptation has clear advantages over other search approaches when confronted with complex multimodal and noisy optimization problems.

The use of multiple methods for population evolution has been studied before. For instance, memetic algorithms (also called hybrid genetic algorithms) have been proposed to increase the search efficiency of population based optimization algorithms [23]. These methods are inspired by models of adaptation in natural systems, and typically use a genetic algorithm for global exploration of the search space (although recently also PSO algorithms are used: [29], combined with a local search heuristic for exploitation. Memetic algorithms do implement multimethod search but do not genetically update preference to individual algorithms during the search, which is a potential source of inefficiency of the method. Closer to the framework we propose in this paper, is the self-adaptive DE (SaDE) algorithm developed by [43]. This method implements two different DE learning strategies and updates their weight in the search based on their previous success rate. Unfortunately, initial results of SaDE are not convincing, receiving a similar performance on the CEC-2005 test suite of functions as many other evolutionary approaches that do not implement a self-adaptive learning strategy.

Other approaches that use ideas of multimethod search are the agent based memetic algorithm proposed by [5] and the ISSOP commercial software package developed by Dualis GmbH IT Solution. The work presented in [5] considers an agent based memetic algorithm. In a lattice-like environment, each of the agents represents a candidate solution of the problem. The agents cooperate and compete through neighborhood orthogonal crossover and different types of life span learning to solve a constrained optimization problem with a suitable constraint handling technique. This approach is quite different from what we will present in this paper. The ISSOP software package uses several optimization strategies in parallel within a neural learn and adaption system. This work has been patented and unfortunately, it is difficult to find detailed information about how ISSOP exactly works, and whether it performs self-adaptation at all. Moreover, we do not have access to the source code severely complicating comparison against our method.

In this paper, we propose a hybrid optimization framework that implements the ideas of simultaneous multimethod search with genetically adaptive (or self-adaptive) offspring creation to conduct an efficient search of the space of potential solutions. Our framework considers multiple different evolutionary search strategies simultaneously, implements a restart strategy with in-
**INITIALIZATION**

Set $t = 0$, and define $t_{\text{max}}$ and $N$

Select which $q$ search operators to use in multimethod search

Define their individual number of offspring points, $\{N_1, \ldots, N_q\}$

**DYNAMIC PART**

while $t < t_{\text{max}}$

$\text{CONVERGED} = \text{‘FALSE’}$

Generate initial population $P_0$ of size $N$ using LHS sampling

Compute fitness $f$ of each individual of $P_0$

while $\text{CONVERGED} = \text{‘FALSE’}$ do

Create offspring population $Q_t$ using $q$ search operators

Compute fitness of points $Q_t$

Combine parent and offspring population, $R_t = P_t \cup Q_t$

Sort $R_t$ in order of decreasing fitness value

Select $P_{t+1}$ from $R_t$ using new species selection approach

if STOPPING CRITERIA = ‘TRUE’ do

$\text{CONVERGED} = \text{‘TRUE’}$

else

Update generation, $t = t + 1$

end if

else

Update $\{N_1, \ldots, N_q\}$ from acquired information

Increase population size, $N = 2 \cdot N$ for restart run

Reset $t = 0$ and recompute $t_{\text{max}}$

end while

end while

Fig. 1. Flowchart of the AMALGAM-SO optimization method. The various symbols are explained in the text.

... how do we determine the weight of each individual algorithm in the search? Hence, if efficiency and robustness are the main goals then better performing search methods should be able to contribute more offspring during population evolution. The next few sections will address each of these issues.

**A. Multimethod Search: Basic Algorithm Description**

The multimethod algorithm developed in the present study employs a population-based elitism search strategy to find the global optimum of a predefined objective function. The method is entitled, A Multi Algorithm Genetically Adaptive Method for Single Objective Optimization or AMALGAM-SO to evoke the image of a procedure that blends the best attributes of individual search algorithms. The basic AMALGAM-SO algorithm is presented in Fig. 1, and is described below. The source code of AMALGAM-SO is written in MATLAB and can be obtained from the first author upon request.

The algorithm is initiated using a random initial population $P_0$ of size $N$ generated by implementing a maximum Euclidean distance Latin hypercube sampling (LHS) method. This sampling method is described in Fig. 2, and strategically places a sample of points within a predefined search space. Then, the fitness value (objective function) of each point is computed, and a population of offspring $Q_0$ of size $N$, is generated by using the multimethod search concept that lies at the heart of the AMALGAM-SO method. Instead of using a single operator for reproduction, we simultaneously utilize $q$ individual algorithms to generate the offspring $Q_0 = \{Q_0^1, \ldots, Q_0^q\}$. These algorithms each create a prespecified number of offspring points $N = \{N_1^1, \ldots, N_q^q\}$ from $P_0$ using different adaptive procedures, where $t$ denotes generation number. In practice, this means that each individual search operator has access to...
Fig. 3. Schematic overview of the adaptive learning strategy of \( \{N_1, \ldots, N_q\} \) in AMALGAM-SO for an illustrative example with three different search operators: the CMA evolutionary strategy, GA en PSO algorithms. The various panels depict (a) \( f_{\text{min}} \): the evolution of the minimum objective function value, (b) \( \Delta f_{\text{min}} \): function value improvement—the difference between two subsequent values of \( f_{\text{min}} \), and (c) which search operator in AMALGAM-SO has caused what respective fitness improvement. The information contained in panels (b) and (c) is post-processed after termination of each optimization run with AMALGAM-SO and used to update \( \{N_1, \ldots, N_q\} \) for the next restart run.

exactly the same genetic material contained in the individuals of the population, but likely produces different offspring because of different approaches for evolution. After creation of the offspring, the fitness value is evaluated, and a combined population \( R_0 = R_0 \cup R_0 \) of size \( 2N \) is created. Then, \( R_0 \) is sorted in order of decreasing fitness value. Finally, members for the next generation \( (t+1) \) are chosen using the new species selection mechanism described in Section III. Elitism is ensured because the best solution found so far will always be included in \( R \). This approach continues until one of the stopping criteria is satisfied, at which time the search with AMALGAM-SO is terminated and the method is restarted with an increased population size. Empirical investigations using a selected set of standard test functions from CEC 2005 demonstrated that a population doubling exhibits the best performance in terms of efficiency and robustness. Conditions for restarting the algorithm will be discussed in a later section. Note that AMALGAM-SO is setup in such a way that it can easily be parallelized on a distributed computer network, so that the method can be used to calibrate complex models that require significant time to run.

Before collecting any information about the response surface, AMALGAM-SO algorithm starts out in the first run with user-defined values of \( \{N_1, \ldots, N_q\} \). These values are subsequently updated to favor individual algorithms that exhibit the highest reproductive success. In principle, various approaches to updating \( N = \{N_1, \ldots, N_q\} \) could be employed based on different measures of success of individual algorithms. For example, one could think of criteria that directly measure fitness value improvement (exploitation), and diversity (exploration), or combinations thereof. Moreover, we could update \( N = \{N_1, \ldots, N_q\} \) after each generation within an optimization run, or only once before each restart run, post-processing the information collected previously. The approach developed here is to update \( N = \{N_1^t, \ldots, N_q^t\} \) using information from the previous run. The way this is done is summarized in Fig. 3, for an illustrative example with three different operators for population evolution in our multimethod optimization concept.

The first panel depicts the evolution of the best fitness value, \( f_{\text{min}} = f(x_{\text{best}}) \), as function of the number of generations with AMALGAM-SO. The curve shows a rapid initial decline, and stabilizes around 300 generations demonstrating approximate convergence to a region with highest probability. The middle panel depicts the absolute function value improvement (difference between two subsequent objective function values presented in the top panel), while the bottom panel illustrates which of the three algorithms in our multimethod search caused the various fitness improvements. From this example, it is obvious that the PSO and CMA algorithm have contributed most to minimization of the objective function, and hence the next restart run should be able to contribute more offspring during population evolution. The information contained in Fig. 3, is stored in memory, and postprocessed after termination of each run with AMALGAM-SO. A two-step procedure is used to update \( \{N_1, \ldots, N_q\} \). In the first step, we compute how much each individual algorithm contributed to the overall improvement in objective function. We denote this as \( \Psi_i, i = 1, \ldots, q \) with \( i = 3 \) in this illustrative example. Then, we update \( \{N_1, \ldots, N_q\} \) according to

\[
N_i = \left[ N \cdot \frac{\Psi_i}{\sum_{i=1}^{q} \Psi_i} \right].
\]
The learning mechanism in (2) ensures that the most productive search algorithms in AMALGAM-SO are rewarded in the next restart run by allowing them to generate more offspring. This method should result in the fastest possible decline of the objective function, as algorithms that contribute most in the reduction of the objective function will receive more emphasis during population evolution. Potentially, there are other metrics, however that could be used to determine the success and thus weighting of the individual search methods within AMALGAM-SO. However, our simulation results using the CEC-2005 testbed of problems, demonstrates that the suggested approach works well for a range of problems.

To avoid the possibility of inactivating algorithms that may contribute to convergence in future generations, minimum values for \( \{N_1, \ldots, N_q\} \) are assigned. These minimum values depend on the particular algorithm and population size used, and will be discussed later. To further increase computational efficiency, AMALGAM-SO explicitly includes the best member found in the previous run in the population of the next restart run, replacing one random member of the initial LHS sample. However, to avoid possible stagnation of the search, this approach is taken only if \( f(x_{\text{best}}) \) is at least \( 10^{-3} \) (in relative terms) better than the minimum function value obtained from the previous run.

**B. Multimethod Search: Which Search Algorithms to Include?**

In principle, the AMALGAM-SO method is very flexible and can accommodate any biological or physical model for population evolution. This implies that the number of search algorithms that potentially could be included in AMALGAM-SO is very large. For illustrative purposes, we here consider only the most popular and commonly used, general-purpose, evolutionary optimization algorithms. These methods are: 1) the covariance matrix adaptation (CMA) evolution strategy; 2) genetic algorithm (GA); 3) parental-centric recombination operator (PCX); 4) particle swarm optimization (PSO); and 5) differential evolution (DE). However, note that the AMALGAM-SO method is not limited to these search strategies. Search approaches that seek iterative improvement from a single starting point in the search domain, such as quasi-Newton and Simplex search can also be used. These methods are frequently utilized in mimetic algorithms, but have the disadvantage of requiring sequential evaluations of the objective function, complicating implementation and efficiency of AMALGAM-SO on distributed computer networks. Moreover, these search techniques are more likely to get stuck in a local solution, if the objective function contains a large number of local optima. In the following five sections, we briefly describe each of the individual evolutionary search strategies used as potential candidates in AMALGAM-SO. Detailed descriptions of these algorithms, including a discussion of their algorithmic parameters, can be found in the cited references. Note that we made no attempt to optimize the algorithmic parameters in each of these individual search methods: the values described in the cited references are used in AMALGAM-SO without adjustment.

1) **CMA Evolution Strategy (ES):** The CMA is an evolution strategy that adapts a full covariance matrix of a normal search (mutation) distribution [18]–[20]. The CMA evolution strategy begins with a user-specified initial population of \( \lambda \) individuals \( x_{(0)}^{(0)} \). After evaluating the objective function, the best \( \mu_{\text{CMA}} \) individuals are selected as parental vectors, and their centers of mass are computed using a prespecified weighting scheme: 
\[
\mu_{(0)}^{(0)} = \sum_{k=1}^{\mu_{\text{CMA}}} w_k x_{(0)}^{(0)},
\]
where the weights \( w_k \in \mathbb{R} \) are positive, and sum to one. In the literature, several different weighting schemes have been proposed for recombination. For instance, one approach is to assign equal weighting to each of the CMA selected individuals, irrespective of their fitness. This is termed intermediate recombination. Other schemes implement a weighted recombination method to favor individuals with a higher fitness. After selection and recombination, an updated population is created according to
\[
x_{(t+1)}^{(0)} = \left( \mu_{(0)}^{(t)} + \sigma^{(t)} \mathbf{B}^{(t)} \mathbf{D}^{(t)} z_{k=1:} \right)
\]
where \( z_k \sim N(0, \mathbf{I}) \) are independent realizations of an \( n \)-dimensional standard normal distribution with zero-mean and a covariance matrix equal to the identity matrix \( \mathbf{I} \). These base points are rotated and scaled by the eigenvectors \( \mathbf{B}^{(t)} \) and the square root of the eigenvalues \( \mathbf{D}^{(t)} \) of the covariance matrix \( \mathbf{C}^{(t)} \). The covariance matrix, \( \mathbf{C}^{(t)} \) and the global step-size, \( \sigma^{(t)} \) are continuously updated after each generation. This approach results in a strategy that is invariant against any linear transformation of the search space. Equations for initializing and updating the strategy parameters are given in [21]. On convex-quadratic functions, the adaptation mechanism for \( \sigma^{(t)} \) and \( \mathbf{C}^{(t)} \) allow log-linear convergence to be achieved after an adaptation time which can scale between 0 and \( n^2 \).

The default strategy parameters are given in [2] and [21]. Only \( \mu_{(0)}^{(0)} \) and \( \sigma^{(0)} \) have to be derived independently of the problem. In all calculations presented herein, a weighted recombination scheme is used: \( u_k = \ln(\mu_{(t) + 1}) - \ln(k^1) \), where \( k^1 = 1, \ldots, \mu_{\text{CMA}} \) is the index of order of the best individuals per generation, sorted in ascending order with respect to their fitness. In addition, \( \mu_{(0)}^{(0)} \) is derived from the initial Latin hypercube sample, and the initial step size, \( \sigma^{(0)} \) is set to \( 1.5 \times 10^{-1} (x_{\text{max}} - x_{\text{min}}) \), where \( x_{\text{max}} \) and \( x_{\text{min}} \) represent the upper and lower boundaries of the feasible search space.

2) **Genetic Algorithm (GA):** Genetic algorithms (GAs) were formally introduced in the 1970s by John Holland at the University of Michigan. Since its introduction, the method has found widespread implementation and use, and has successfully solved optimization and search problems in many fields of study. The GA is an adaptive heuristic search algorithm that mimics the principles laid down by Charles Darwin in his evolution theory and successively applies genetic operators such as selection, crossover (recombination) and mutation to iteratively improve the fitness of a population of points and find the global optimum in the search space. While the crossover operation leads to a mixing of genetic material in the offspring, no new allelic material is introduced. This can lead to lack of population diversity and eventually “stagnation” in which the population converges to the same, non-optimal solution. The GA mutation operator helps to increase population diversity by introducing new genetic material.

In all GA experiments reported in this paper, a tournament selection scheme is used, with uniform crossover and polynomial mutation. Tournament selection involves picking a number of
strings at random from the population to form a “tournament” pool. The two strings of highest fitness are then selected as parents from this tournament pool. Uniform crossover creates a random binary vector and selects the genes where the vector is a 1 from the first parent, and the genes where the vector is a 0 from the second parent, to generate offspring. To increase population diversity, and to enable the search to escape from local optimal solutions, the method of polynomial mutation, described in detail in [9], is implemented, with default values for crossover \( p_c \), and mutation probability, \( p_m \), of 0.90 and 1/\( n \), respectively using a value of the mutation distribution index of \( n_m = 20 \) [9].

3) Parental-Centric Recombination Operator (PCX): The PCX operator has recently been proposed by [11] as an efficient recombination operator for creating offspring from an existing population of points within the context of GAs. Systematic studies have shown that this parent centric recombination is a good alternative to other commonly used crossover operators, and provides a meaningful and efficient way to solving real-parameter optimization problems.

The method starts by randomly initializing a population of points using Latin hypercube sampling. After this, the fitness of each individual is calculated and \( \mu_{PCX} \) Parents are randomly selected from the existing population of points. The mean vector \( g \) of these parents is computed, and one parent \( x^{(0)}_p \) from this set is chosen with equal probability. The direction vector of \( d^{(0)}_p = (x^{(0)}_p - g) \) is then computed to measure the distance and angle between the selected parent and \( g \). From the remaining \( \mu_{PCX} - 1 \) parents, perpendicular distances \( D^{(0)}_{PCX} \) to the line \( d^{(0)}_p \) are computed, and their average \( \bar{D}^{(0)}_{PCX} \) is calculated. The offspring is created as follows:

\[
x^{(t+1)}_k = x^{(t)}_p + \omega_k g^{(t)} + \sum_{i=1,i \neq p}^{\mu_{PCX}} \omega_i D^{(t)}_{PCX} e_i
\]

\( k = 1, \ldots, \mu_{PCX} \)  (4)

where \( e^{(t)}_i \) denote the \((\mu_{PCX} - 1)\) orthonormal bases that span the subspace perpendicular to \( d^{(t)}_p \). The parameters \( \omega_k \) and \( \omega_i \) are zero-mean, normally distributed variables with variance \( \sigma^2_{\omega_k} \) and \( \sigma^2_{\omega_i} \), respectively.

In contrast to other crossover operators, the PCX operator assigns a higher probability for an offspring to remain closer to the parents than away from the parents. This should result in an efficient and reliable search strategy, especially in the case of continuous optimization problems. In all our PCX runs, we have used the default values \( \sigma^2_{\omega_k} = \sigma^2_{\omega_i} = 0.01 \), and \( \mu_{PCX} = 3 \), as recommended in [11], and \( x^{(t)}_p \) is set to be the best parent in the population.

4) Particle Swarm Optimization (PSO): PSO is a population-based stochastic optimization method whose development was inspired from the flocking and swarm behavior of birds and insects. After its introduction in 1995 [13], [26], the method has gained rapid popularity in many fields of study. The method works with a group of potential solutions, called particles, and searches for optimal solutions by continuously modifying this population in subsequent generations. To start, the particles are assigned a random location and velocity in the \( n \)-dimensional search space. After initialization, each particle iteratively adjusts its position according to its own flying experience, and according to the flying experience of all other particles, making use of the best position encountered by itself, \( p^{\text{best}}_k \), and the entire population, \( x^{\text{best}} \). In contrast to the other search algorithms included in AMALGAM-SO, the PSO algorithm combines principles from local and global search to evolve a population of points toward the global optimal solution. The reproductive operator for creating offspring from an existing population is [13], [26]

\[
v^{(t+1)}_k = \varphi \cdot v^{(t)}_k + c_1 r_1 (x^{(t)}_{\text{best}} - x^{(t)}_k) + c_2 r_2 (p^{(t)}_k - x^{(t)}_k)
\]

\( x^{(t+1)}_k = x^{(t)}_k + v^{(t+1)}_k \)  (5)

where \( v^{(t)}_k \) and \( x^{(t)}_k \) represent the current velocity and location of a particle, \( \varphi \) is the inertia factor, \( c_1 \) and \( c_2 \) are weights reflecting the cognitive and social factors of the particle, respectively, and \( r_1 \) and \( r_2 \) are uniform random numbers between 0 and 1. Based on recommendations in various previous studies, the values for \( c_1 \) and \( c_2 \) were set to 2, and the inertia weight was linearly decreased between 0.9 at the first generation and 0.4 at the end of the simulation. To limit the jump size, the maximum allowable velocity was set to be 15 of the longest axis-parallel interval in the search space [40]. In addition, we implement a circle neighborhood topology approach and replace \( x^{\text{best}} \) in (5) with the best individual found within the closest 10% of each respective particle as measured in the Euclidean space.

5) Differential Evolution (DE): While traditional evolutionary algorithms are well suited to solve many difficult optimization problems, interactions among decision variables (parameters) introduces another level of difficulty in the evolution. Previous work has demonstrated the poor performance of a number of evolutionary optimization algorithms in finding high quality solutions for rotated problems exhibiting strong interdependencies between parameters [37]. Rotated problems typically require correlated, self-adapting mutation step sizes in order to make timely progress in the optimization.

DE has been demonstrated to be effective in dealing with strong correlation among decision variables, and, like the CMA-ES, exhibits rotationally invariant behavior [37]. DE is a population-based search algorithm that uses fixed multiples of differences between solution vectors of the existing population to create children. In this study, the version known as DE/rand/1/bin or “classic DE” [37], [38] is used to generate offspring

\[
u^{(t+1)}_k = x^{(t)}_1 + F (x^{(t)}_{r2} - x^{(t)}_{r3}) \quad k = 1, \ldots, \lambda
\]

where \( F \) is a mutation scaling factor that controls the level of combination between individual solutions, and \( r_1, r_2, \) and \( r_3 \) are randomly selected indices from the existing population \( \{1, \ldots, \lambda\} \); \( r_1 \neq r_2 \neq r_3 \neq k \). Next, one or more parameter values of this mutant vector \( u^{(t+1)}_k \) are uniformly crossed with those belonging to \( x^{(t)}_k \) to generate the offspring \( x^{(t+1)}_{k,j} \)

\[
x^{(t+1)}_{k,j} = \begin{cases} u^{(t+1)}_{k,j} + x^{(t)}_{k,j}, & \text{if } U \leq CR \\ x^{(t)}_{k,j}, & \text{otherwise} \end{cases} \quad j = 1, \ldots, n
\]

where \( U \) is a uniform random label between \([0, 1]\) and \( CR \) denotes the crossover constant that controls the fraction of parame-
ters that the mutant vector contributes to the offspring. In all DE runs in this study, default values of \( F = 0.9 \) and \( CR = 0.9 \) are used [33]. Note that the value of \( CR \) is similar to the crossover probability used in the GA algorithm.

III. MULTIMETHOD SEARCH: PRESERVING POPULATION DIVERSITY

Preserving population diversity is crucial for successful and efficient search of complex multimodal response surfaces. A lack of diversity often results in premature convergence to a sub-optimal solution, whereas too much diversity often results in an inability to further refine solutions and more closely approximate the location of the global optimum. In multiobjective optimization, population diversity is easily maintained because the tradeoff between the various objectives naturally induces diversity in the population, enabling the algorithm to locate multiple, Pareto optimal solutions. In contrast, maintaining diversity in single-objective optimization is much more difficult because in the pursuit of a single, optimal solution, an algorithm may relinquish occupation of the regions of the search space with lower fitness. This genetic drift, in which the population is inclined to converge toward a single solution, is typical of many evolutionary search algorithms.

To prevent the collapse of evolutionary algorithms into a relatively small basin of highest attraction, numerous approaches have been devised to help preserve population diversity. Significant contributions in this direction are fitness sharing [16], crowding [25], local mating [8] and niching [30]. Many of these methods employ a distance measure on the solution space to ensure that members of the population do not become too similar to one another. In this study, a modified version of the speciation method presented in [28] is developed to improve diversity and search capabilities. The algorithm is presented in Fig. 4 and described below.

At each generation, the algorithm takes as input the combined population \( R \) with size \( 2N \), sorted in order of decreasing fitness. Initially, the best member of \( R \) (first element) is included as first member in \( P \). Then, in an iterative procedure, the next individual in \( R \) is compared to the species currently present in \( P \). If the Euclidean distance of this individual to all points present in \( P \) is larger than a user-defined distance, \( \alpha \), then this member of \( R \) will be added to \( P \). This process is repeated until all the slots in \( P \) are filled, and the resulting population is of size \( N \). In this approach, the parameter \( \alpha \) controls the level of diversity maintained in the solution.

It is impossible to identify a single generic value of \( \alpha \) that works well for a range of different optimization problems. Certain optimization problems require only a small diversity in the population to efficiently explore the space (and thus small values for \( \alpha \)), whereas other problems require significant variation in the population to prevent premature convergence to local optimal solutions in multimodal fitness landscapes. To improve the species selection mechanism, the value of \( \alpha \) is therefore sequentially increased from \( 10^{-10} \delta \) to \( 10^{-1} \delta \) with a factor of 10, where \( \delta = x_{\text{MAX}} - x_{\text{MIN}} \). The upper and lower boundaries of the search space are fixed at initialization. For each different value of \( \alpha \), an equal number of points is added to \( P_{t+1} \). This procedure preserves useful diversity in the population at both small and large distances, and improves the robustness and efficiency of AMALGAM-SO.

**INITIALIZATION**

Define \( \alpha \)

Set \( P_{t+1} = \emptyset \)

**DYNAMIC PART**

\[ |P_{t+1}| < N \] do

Get best unprocessed member \( r \) of \( R_t \)

Set \( \text{FOUND} = \text{‘FALSE’} \)

for all \( p \in P_{t+1} \) do

\[ D(p, r) = \sqrt{\sum_{i=1}^{n} (p_i - r_i)^2} \]

if \( D(p, r) \leq \alpha \) do

Set \( \text{FOUND} = \text{‘TRUE’} \)

end if

end for

if \( \text{FOUND} = \text{‘FALSE’} \) do

\( P_{t+1} = P_{t+1} \cup r \)

end for

end while

**Fig. 4.** Illustration of species selection approach to select \( P_{t+1} \) from \( R_t \). This procedure prevents the collapse of AMALGAM-SO into a relatively small basin of highest attraction. The selection of \( \alpha \) is explained in the text.

IV. MULTIMETHOD SEARCH: STOPPING CRITERIA AND BOUNDARY HANDLING

To implement the restart feature of the AMALGAM-SO method, the stopping criteria listed below are used. The first five stopping criteria only apply to the CMA-ES, and are quite similar to those described and used in [2]. The second list of stopping criteria applies to the GA/PCX/PSO/DE algorithms, and is different than those used for the CMA algorithm to be consistent with a difference in search approach.

**CMA-ES:**

1) Stop if the range of the best objective function values of the last \( 10 + \lceil 30n/\lambda \rceil \) generations is zero, or the ratio of the range of the current function values to the maximum current function value is below \( T_{\text{olFun}} \). In this study, \( T_{\text{olFun}} = 10^{-5} \) is used.

2) Stop if the standard deviation of the normal distribution is smaller than \( T_{\text{olX}} \) in all coordinates and the evolution path from (2) in [21] is smaller than \( T_{\text{olX}} \) in all components. In this study, \( T_{\text{olX}} = 10^{-9} \sigma(t) \) is used.

3) Stop if adding a 0.1-standard deviation vector in a principal axis direction of \( \mathbf{C}^{(t)} \) does not change \( \langle x \rangle_W^{(t)} \)

4) Stop if adding a 0.2-standard deviation in each coordinate does not change \( \langle x \rangle_W^{(t)} \)

5) Stop if the condition number of the covariance matrix exceeds \( 10^{14} \).
TABLE I
MATHEMATICAL TEST FUNCTIONS USED IN THIS STUDY, INCLUDING A SHORT DESCRIPTION OF THEIR PECULIARITIES. A DETAILED DESCRIPTION OF THE INDIVIDUAL FUNCTIONS AND THEIR MATHEMATICAL EXPRESSIONS CAN BE FOUND IN [39]

<table>
<thead>
<tr>
<th>UNIMODAL FUNCTIONS</th>
<th>MULTIMODAL FUNCTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Shifted Sphere Function</td>
<td>6. Shifted Rosenbrock’s Function</td>
</tr>
<tr>
<td>2. Shifted Schwefel’s Problem 1.2</td>
<td>7. Shifted Rotated Griewank’s Function without Bounds</td>
</tr>
<tr>
<td>3. Shifted Rotated High Conditioned Elliptic Function</td>
<td>8. Shifted Rotated Ackley’s Function</td>
</tr>
<tr>
<td>4. Shifted Schwefel’s Problem 1.2 with Noise in Fitness</td>
<td>9. Shifted Rastrigin’s Function</td>
</tr>
<tr>
<td>5. Schwefel’s Problem with Global Optimum on Bounds</td>
<td>10. Shifted Rotated Rastrigin’s Function</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EXPANDED FUNCTIONS</th>
<th>COMPOSITION FUNCTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>13. Shifted Expanded Griewank’s + Rosenbrock’s Function</td>
<td>15. Hybrid Composition Function</td>
</tr>
<tr>
<td></td>
<td>17. Function $f_{16}$ with Noise in Fitness</td>
</tr>
<tr>
<td></td>
<td>18. Rotated Hybrid Composition Function</td>
</tr>
<tr>
<td></td>
<td>19. Rotated Hybrid Composition Function</td>
</tr>
<tr>
<td></td>
<td>20. Rotated Hybrid Composition Function</td>
</tr>
<tr>
<td></td>
<td>21. Rotated Hybrid Composition Function</td>
</tr>
<tr>
<td></td>
<td>22. Rotated Hybrid Composition Function</td>
</tr>
<tr>
<td></td>
<td>23. Non-Continuous Rotated Hybrid Composition Function</td>
</tr>
<tr>
<td></td>
<td>24. Rotated Hybrid Composition Function</td>
</tr>
<tr>
<td></td>
<td>25. Rotated Hybrid Composition Function without bounds</td>
</tr>
</tbody>
</table>

GA/PCX/PSO/DE-algorithms:

1) Stop if the standard deviation of $x_{best}$ is smaller than $\text{ToDX}$ in all coordinates.
2) Stop if the ratio of the range of the best function values found so far (associated with $x_{best}$) to the maximum function value of $x_{best}$ is below $\text{ToFun}$. In this study, $\text{AcceptRate} = \frac{7.5 \times 10^{-2} \lambda}{2}$.
3) Stop if the acceptance rate of new points in $x_{best}$ is lower than AcceptRate. In this study, $\text{AcceptRate} = 7.5 \times 10^{-2} \lambda$.
4) Stop if the range of the best objective function values of the last $50 + [100m/\lambda]$ generations is zero.

The criteria above determine when to restart the algorithm. The overall stopping criteria of the AMALGAM-SO are: stop after $n \cdot 10^4$ function evaluations or stop if the best objective function value found so far is smaller than a predefined tolerance value.

V. NUMERICAL EXPERIMENTS AND PROCEDURE

The performance of the AMALGAM-SO algorithm is tested using a selected set of standard test functions from the special session on real-parameter optimization of the IEEE Congress on Evolutionary Computations, CEC 2005. These functions span a diverse set of problem features, including multimodality, ruggedness, noise in fitness, ill-conditioning, nonseparability, interdependence (rotation), and high-dimensionality, and are based on classical benchmark functions such as Rosenbrock’s, Rastrigin’s, Griewank’s and Ackley’s function. Table I presents a short description of these functions with their specific peculiarities. A detailed description of these functions appears in [39], and so will not be repeated here. In summary, functions 1–5 are unimodal, functions 6–12 are multimodal, and functions 13–25 are hybrid composition functions, implementing a combination of several well-known benchmark functions.

Each test function is solved in 10, 30, and 50 dimensions, using 25 independent trials to obtain statistically meaningful results. To prevent exploitation of symmetry of the search space, the global optimum is shifted to a value different from zero [15]. In all our experiments, the initial population size of AMALGAM-SO was set to 10, 15, and 20 in dimensions 10, 30, and 50, respectively. The minimum number of children an algorithm needs to contribute $\{N^\text{min}_1, \ldots, N^\text{min}_q\}$ was set to 5% of the population size for the GA/PCX/PSO/DE methods, and to 25% of the population size for the CMA-ES. These...
numbers were found most productive for a diverse set of problems. However, in the first trial with lowest population size, the value of $N_{\text{CMA}}$ is set to 80% of the population size, to improve efficiency of AMALGAM-SO on simple problems that do not require a restart run to find the global minimum within the desired tolerance limit.

For each function, a feasible search space of $[x_{\text{min}}, x_{\text{max}}]$ of $\mathbb{R}^n$ is prescribed at initialization. To make sure that the offspring remains within these bounds, two different procedures are implemented. For the GA/PCX/PSO/DE algorithms, points generated outside the feasible region are reflected back from the bound by the amount of the violation [33]. For the CMA algorithm, the boundary handling procedure implemented in the MATLAB code of the CMA-ES algorithm: (Version 2.35: http://www.bionik.tu-berlin.de/user/niko/formersoftwareversions.html) is used to penalize infeasible individuals.

To evaluate the numerical performance of AMALGAM-SO, we use two different measures. Both these measures are presented in [1]. The first criterion, hereafter referred to as $(p_b)$ measures the probability of success of AMALGAM-SO. This measure simply calculates as the ratio between the number of successful runs, and the total number of runs ($\approx 25$ in this particular case). The second criterion, entitled $SP_1$ measures the average number of function evaluations needed to solve a particular optimization problem. This is calculated as

$$SP_1 = E(T_n) \cdot p_b^{-1}$$  \(8\)

where $E(T_n)$ signifies the expected number of function evaluations of AMALGAM-SO during a successful run, which is identical to the ratio between the number of function evaluations in a successful run and the total number of successful runs. A successful optimization run is defined as one in which the AMALGAM-SO algorithm achieves the fixed accuracy level within $FE_{\text{max}}$ function evaluations. Consistent with previous work, $FE_{\text{max}}$ was set to $n \cdot 10^4$.

VI. RESULTS AND DISCUSSION

In this section, we present the results of our numerical experiments with AMALGAM-SO. In the first two sections, we present an analysis to determine which search algorithms to include in AMALGAM-SO, followed by simulation results in dimensions $n = 10, 30$ and 50 using the CEC-2005 test suite of functions. In this part of the paper, we are especially concerned with a comparison analysis of AMALGAM-SO against current state-of-the-art evolutionary algorithms. The final two sections focus on the empirical efficiency of AMALGAM-SO by providing plots that illustrate scaling behavior for various functions, and highlight the robustness of the method for a test function with noise induced multimodality.

A. Which Search Algorithms to Include in AMALGAM-SO?

The presented statistics in Table II highlight several important observations regarding algorithm selection. In the first place, notice that there is significant variation in performance of the various multimethod search algorithms. Some combinations of search operators within AMALGAM-SO have excellent search capabilities and consistently solve many of the considered test functions, whereas other combinations of algorithms exhibit a relatively poor performance. For instance, the DE-PSO algorithm has an overall success rate of 0, consistently failing to locate good quality solutions in all problems. On the contrary, the CMA-GA-DE algorithm demonstrates robust convergence properties across the entire range of test problems, with the exception of function 8. This benchmark problem is simply too difficult to be solved with current evolutionary search techniques.

Second, among the algorithms included in this study, the CMA evolution strategy seems to be a necessary ingredient in our self-adaptive multimethod optimization algorithm. The performance of AMALGAM-SO with CMA is significantly better in terms of $p_b$ and $SP_1$ than any other combination of algorithms not utilizing this method for population evolution. This finding is consistent with previous work demonstrating the superiority of a restart CMA-ES over other search approaches for the unimodal and multimodal functions considered here [22]. This highlights that the success of AMALGAM-SO essentially relies on the quality of individual algorithms, thus maintaining scope for continued effort to improve individual search operators.

Third, increasing the number of adaptive strategies for population evolution in AMALGAM-SO does not necessarily lead to better performance. For instance, the performance of the CMA-GA-PCX-DE-PSO variant of AMALGAM-SO is significantly worse on the unimodal (in terms of number of function evaluations) and multimodal test functions (success rate) than other, simpler strategies. It is evident, however that the CMA-GA-DE and CMA-GA-PSO search strategies generally exhibit the best performance. Both these methods, implement three individual algorithms for population evolution, and show nearly similar performance on the unimodal (1–6), and multimodal test functions (7–12). Their results have been highlighted in gray in Table II.

Fourth, it is interesting to observe that similar search operators as selected here for single objective optimization, were selected for population evolution in the multiobjective version of AMALGAM [41]. In that study, a combination of the NSGA-II [12], DE, PSO, and adaptive covariance sampling [17] methods was shown to have all the desirable properties to efficiently evolve a population of individuals through the search space and find a well distributed set of Pareto solutions. The similarity between search operators selected in both studies suggests that it is conceivable to develop a single universal multimethod optimization algorithm that efficiently evolves a population of individuals through single and multiobjective response surfaces.

There are two main reasons that explain the relative poor performance of AMALGAM-SO in the absence of CMA for population evolution. First, the initial population of $N = 10$ as used in AMALGAM-SO is too small for the PSO, PCX, DE, and GA search strategies to be productive and appropriately search the space. If $N \leq n$, then convergence of these methods essentially relies on the choice and efficiency of the mutation operator used to increase diversity of the population, because the $N$ points lie in an $N-1$ dimensional space. Even though the restart strategy
implemented in AMALGAM-SO successively increases population size, it will take quite a number of function evaluations to reach a population size, N, that is considered reasonable for the PSO, PCX, DE, and GA algorithms. For instance, a population size of at least N = 100 individuals is commonly used with population based global search methods. Starting out with a larger initial population might resolve at least some convergence problems, but at the expense of requiring many more function evaluations to find solutions within the tolerances of the global minimum, when compared to the CMA-ES algorithm. The covariance adaptation strategy used in this method is designed in such a way that it still performs well when and the population lies in a reduced space.

Second, many of the test functions considered here are difficult to solve because they contain numerous local optimal solutions. The population evolution strategy used in the PSO, PCX, DE, and GA evolutionary algorithms will exhibit difficulty in the face of many local basins of attraction, because local optimal solutions tend to persist in the population from one generation to the next. This not only deteriorates the efficiency of the search, but also frequently results in premature convergence. Various contributions to the optimization literature have therefore proposed extensions such as fitness sharing, speciation, chaotic mutation, and use of simulated annealing for population evolution to increase the robustness of the PSO, PCX, DE, and GA evolutionary algorithms on multimodal problems. The CMA-ES algorithm, on the contrary implements a different procedure for population evolution that is less prone to stagnation in a local solution. This method approximates the orientation and scale of the response surface indirectly by continuously morphing and adjusting the covariance matrix of the parameters to be estimated. The population in CMA-ES is always sampled from this information, and it is therefore unlikely that exactly similar solutions will propagate from one generation to the next. This type of search strategy is more robust and efficient when confronted with difficult response surfaces that contain many local optima. The CMA method is therefore indispensable for population evolution in AMALGAM-SO.

To simplify further analysis in this paper, we carry the CMA-GA-PSO forward, as this version of AMALGAM-SO exhibits the highest success rate on composite function 15. The other multimethod search strategies discussed in Table II are therefore eliminated from further consideration. Table III summarizes the algorithmic parameters in the CMA-GA-PSO version of AMALGAM-SO, including their default values.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Problem</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMA-GA</td>
<td></td>
<td>1.00/1.618</td>
<td>1.00/2.498</td>
<td>1.00/7.736</td>
<td>1.00/2.751</td>
<td>1.00/7.759</td>
<td>1.00/1.2764</td>
<td>1.00/1.5535</td>
<td>0</td>
<td>0.80</td>
<td>0.74/0.64</td>
<td>0.74/0.64</td>
<td>1.00/2.864</td>
<td>1.00/1.084</td>
<td>0</td>
<td>(0.59)</td>
</tr>
<tr>
<td>CMA-PCX</td>
<td></td>
<td>1.00/2.020</td>
<td>1.00/2.251</td>
<td>1.00/7.736</td>
<td>1.00/2.758</td>
<td>1.00/6.674</td>
<td>1.00/1.4164</td>
<td>1.00/1.5585</td>
<td>0</td>
<td>0.72</td>
<td>0.76/0.71</td>
<td>0.76/0.71</td>
<td>1.00/2.864</td>
<td>1.00/1.614</td>
<td>0</td>
<td>(0.93)</td>
</tr>
<tr>
<td>CMA-DE</td>
<td></td>
<td>1.00/2.628</td>
<td>1.00/2.438</td>
<td>1.00/7.754</td>
<td>1.00/2.747</td>
<td>1.00/6.672</td>
<td>1.00/1.394</td>
<td>1.00/1.4282</td>
<td>0</td>
<td>0.84</td>
<td>0.71/0.62</td>
<td>0.71/0.62</td>
<td>1.00/2.504</td>
<td>1.00/1.3644</td>
<td>0</td>
<td>(0.64)</td>
</tr>
<tr>
<td>CMA-PSO</td>
<td></td>
<td>1.00/1.610</td>
<td>1.00/2.498</td>
<td>1.00/7.748</td>
<td>1.00/2.779</td>
<td>1.00/6.694</td>
<td>1.00/1.2846</td>
<td>1.00/1.5595</td>
<td>0</td>
<td>1.00</td>
<td>0.74/0.62</td>
<td>0.64/0.604</td>
<td>1.00/1.8764</td>
<td>1.00/1.694</td>
<td>0</td>
<td>(0.68)</td>
</tr>
<tr>
<td>GA-PCX</td>
<td></td>
<td>0.86/3.917</td>
<td>0.00/0.00</td>
<td>0.00/0.00</td>
<td>0</td>
<td>0.49</td>
<td>1.1565</td>
<td>0</td>
<td>0</td>
<td>0.89</td>
<td>3.68/0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(0.07)</td>
<td>0</td>
</tr>
<tr>
<td>GA-DE</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.99</td>
<td>3.68/0</td>
<td>0</td>
<td>0</td>
<td>0.04</td>
<td>(1.32)</td>
<td>0</td>
</tr>
<tr>
<td>GA-PSO</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.99</td>
<td>4.04/0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(0.23)</td>
<td>0</td>
</tr>
<tr>
<td>PCC-DE</td>
<td></td>
<td>1.00</td>
<td>2.751</td>
<td>1.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0</td>
<td>0.40</td>
<td>2.68/0</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.08</td>
<td>0.95</td>
<td>0.08</td>
</tr>
<tr>
<td>PCC-PSO</td>
<td></td>
<td>1.00</td>
<td>4.4764</td>
<td>0.04</td>
<td>1.7166</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.04</td>
<td>1.9164</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(0.27)</td>
<td>0</td>
</tr>
<tr>
<td>DE-PSO</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(1.18)</td>
<td>0</td>
<td>(0.93)</td>
</tr>
</tbody>
</table>

Table III

<table>
<thead>
<tr>
<th>CMA</th>
<th>(x)W(0)</th>
<th>Initial center of mass</th>
<th>Initial step-size</th>
<th>Recombination scheme</th>
<th>Initial LHS sample (0.15 \cdot (x_{max} - x_{min}) \ln(\mu_{CMA-1}) - \ln(k^1))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\sigma(0))</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(w_k)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\rho_c)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\rho_m)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\eta_m)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(c_1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(c_2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\varphi_{min})</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\varphi_{max})</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCC-DE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\rho_{PCX})</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\rho_{DE})</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\rho_{PSO})</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\rho_{AMALGAM-SO})</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\rho_{N_{1,2,3}})</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\rho_{N_{T_{1,2,3}})}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(\rho_{PDPincreased})</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
used in this study. As highlighted earlier, these default values are taken from standard implementations of the individual CMA, GA, and PSO algorithms in the literature. No attempt was made to tune them to further improve the performance of AMALGAM-SO on the test functions considered here.

B. Simulation Results in 10, 30, and 50 Dimensions

This section presents results of AMALGAM-SO for the considered benchmark problems in dimensions 10, 30, and 50. Results are tabulated in Tables IV–VI, and empirical distributions of normalized success performance are compiled in Figs. 5–7. Statistics are provided only for functions 1–17; the other functions are included here, because it receives the best performance on these problems. As highlighted earlier, these default values are taken from standard implementations of the individual CMA, GA, and PSO algorithms in the literature. No attempt was made to tune them to further improve the performance of AMALGAM-SO on the test functions considered here.

1) For unimodal problems 1–6 and multimodal problems 6 and 7, the performance of AMALGAM-SO and the G-CMA-ES are quite similar, although the G-CMA-ES is a bit more efficient in terms of the number of function evaluations to reach convergence. Note however, that the only strategy that will be successful for this problem is exhaustive uniform sampling of the search space. However, this approach is computationally very demanding and inefficient.

2) Function 8 remains unsolved in all dimensions: neither algorithm is able to locate the needle in the haystack. Perhaps the only strategy that will be successful for this problem is exhaustive uniform sampling of the search space. However, this approach is computationally very demanding and inefficient.

3) For all the other problems, the AMALGAM-SO method is generally superior to G-CMA-ES. This conclusion is true for all dimensions. For test functions in which both algorithms are unable to find solutions within the tolerance limit, the fitness values obtained with the AMALGAM-SO method are significantly smaller than those derived with the G-CMA-ES. This is most notable for test functions 15 and 17 in dimension 30 and 50, respectively.

The results presented in Tables IV–VI demonstrate that for relatively simple problems, single search operators will generally suffice and be efficient. In those cases, multimethod search is an overkill, and therefore about 10%–15% less efficient in terms of the number of function evaluations than the G-CMA-ES. For more complex optimization problems, however, self-adaptive multimethod search seems to have desirable advantages as it can switch between different operators to maintain efficiency and robustness of the search when confronted with multimodality, numerous local optima, significant nonlinearity, ruggedness, interdependence, etc. For these problems, a single operator is simply not powerful enough to efficiently and reliably handle all these difficulties.

So far, we have compared the performance of AMALGAM-SO against the G-CMA-ES only. To compare the performance of the AMALGAM-SO method against a variety of other search techniques, consider Figs. 5 and 6, which present empirical distribution functions of the success performance of individual algorithms divided by $SP_1$ of the best performing algorithm for the unimodal and multimodal test functions in dimension (A) $n = 10$, and (B) $n = 30$. The different color lines distinguish between the results of the BLX-GL50 [14], BLX-MA [31], CoEVO [32], DE [33], DMS-L-PSO [29], EDA [44], G-CMA-ES [2], K-PCX [36], L-CMA-ES [1], L-SaDE [43], and SPC-PNX [4] algorithms on the same functions. These various algorithms...
represents the current state-of-the-art, and are based on commonly used search methods, such as the CMA, PSO, DE, and GA algorithms. Moreover, they represent a broad array of optimization techniques, including single search operator (CoEVO, DE, EDA, G-CMA-ES, L-CMA-ES, K-PCX, SPC-PNX), and multithreaded memetic (BLX-GL50, BLX-MA, DMS-L-PSO), and self-adaptive (L-SaDE) evolutionary algorithms. The performance of these algorithms on the CEC-2005 test bed of functions considered here has been extracted from tables presented in [22]. The steeper the cumulative distribution function (CDF), the better the performance of an algorithm across the range of test functions examined. In other words, small values for the ratio of \( \frac{p}{n} \) and large associated cumulative frequency values are preferred. Similar plots have been used in [22] to compare performance of individual algorithms.

The results presented in Figs. 5 and 6 demonstrate that AMALGAM-SO receives consistently good performance over a large range of problems and dimensions. As demonstrated earlier, the G-CMA-ES restart method is slightly more efficient than AMALGAM-SO for the unimodal problems of CEC-2005 in dimension 10 and 30. The CDF of G-CMA-ES reaches its maximum of one at a smaller ratio of \( \frac{p}{n} \) than AMALGAM-SO, demonstrating a higher efficiency in solving these unimodal problems. This difference in efficiency is however, generally small and approximates about 10%–15% in terms of function evaluations as determined previously from the numbers listed in Tables IV and V. On the contrary, for the multimodal test functions presented in Fig. 6(a) and (b), the empirical CDF of AMALGAM-SO is significantly sharper than the CDF of any of the other algorithms. Not only sharper, but AMALGAM-SO also successfully solves all problems consid-

### TABLE V

FUNCTIONS 1–17 IN DIMENSION \( n = 30 \): NUMBER OF FUNCTION EVALUATIONS (MIN, 7TH, MEDIAN, 19TH, MAXIMUM, MEAN AND STANDARD DEVIATION) NEEDED TO FIND A FUNCTION VALUE THAT IS WITHIN \( TOL \) FROM THE GLOBAL OPTIMUM. BECAUSE NONE OF THE OPTIMIZATION RUNS HAVE CONVERGED FOR FUNCTIONS 8, 13, 14, 16, AND 17 (INDICATED WITH AN ASTERISK), WE REPORT THE FINAL OBJECTIVE FUNCTION VALUES (IN PARENTHESES) FOR THESE FUNCTIONS. THE PERFORMANCE OF THE RESTART CMA-ES (FROM [2]) IS ALSO INCLUDED.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Tol</th>
<th>7th</th>
<th>Median</th>
<th>19th</th>
<th>max</th>
<th>mean</th>
<th>std</th>
<th>( p_n )</th>
<th>SP1</th>
<th>mean</th>
<th>std</th>
<th>( p_n )</th>
<th>SP1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1e-6</td>
<td>4500</td>
<td>4620</td>
<td>4740</td>
<td>4815</td>
<td>4875</td>
<td></td>
<td>4714</td>
<td>118</td>
<td>1.00</td>
<td>4714</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1e-6</td>
<td>1.38e4</td>
<td>1.41e4</td>
<td>1.45e4</td>
<td>1.47e4</td>
<td>1.52e4</td>
<td></td>
<td>1.44e4</td>
<td>377</td>
<td>1.00</td>
<td>1.44e4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1e-6</td>
<td>4.92e4</td>
<td>5.04e4</td>
<td>5.09e4</td>
<td>5.13e4</td>
<td>5.24e4</td>
<td></td>
<td>5.09e4</td>
<td>793</td>
<td>1.00</td>
<td>5.09e4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1e-6</td>
<td>2.25e4</td>
<td>4.38e4</td>
<td>5.04e4</td>
<td>1.62e5</td>
<td></td>
<td></td>
<td>9.16e4</td>
<td>8.10e4</td>
<td>0.96</td>
<td>9.64e4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1e-6</td>
<td>1.51e5</td>
<td>1.85e5</td>
<td>1.91e5</td>
<td>2.00e5</td>
<td>2.30e5</td>
<td></td>
<td>1.91e5</td>
<td>1.87e4</td>
<td>0.98</td>
<td>1.96e5</td>
<td>2.13e4</td>
<td>1.86e4</td>
</tr>
</tbody>
</table>

The results presented in Figs. 5 and 6 demonstrate that AMALGAM-SO receives consistently good performance over a large range of problems and dimensions. As demonstrated earlier, the G-CMA-ES restart method is slightly more efficient than AMALGAM-SO for the unimodal problems of CEC-2005 in dimension 10 and 30. The CDF of G-CMA-ES reaches its maximum of one at a smaller ratio of \( \frac{SP1}{SP1_{1st}} \) than AMALGAM-SO, demonstrating a higher efficiency in solving these unimodal problems. This difference in efficiency is however, generally small and approximates about 10%–15% in terms of function evaluations as determined previously from the numbers listed in Tables IV and V. On the contrary, for the multimodal test functions presented in Fig. 6(a) and (b), the empirical CDF of AMALGAM-SO is significantly sharper than the CDF of any of the other algorithms. Not only sharper, but AMALGAM-SO also successfully solves all problems consid-
The advantages of AMALGAM-SO are further illustrated in Fig. 7, that presents a graphical interpretation of the results presented in Table VI by plotting the cumulative distribution functions of the G-CMA-ES and AMALGAM-SO methods for functions 1–15 for \( n = 50 \). This figure only compares AMALGAM to the G-CMA-ES, as the best results in this dimension have been obtained with this latter algorithm. The CDF derived for AMALGAM-SO is significantly steeper than its counterpart for the restart G-CMA-ES, indicating both an improved efficiency and robustness of AMALGAM-SO over previous methods. This is a significant result, and further highlights the advantages of multimethod search for solving complex multimodal and high-dimensional optimization problems.

To explore why the AMALGAM-SO method achieves a consistently good performance, Fig. 8 presents diagnostic information on the behavior of the multimethod approach of genetically adaptive (or self-adaptive) offspring creation for test functions [Fig. 8(a)] 1 (unimodal), [Fig. 8(b)] 9 (multimodal), [Fig. 8(c)] 13 (composite), and [Fig. 8(d)] 15 (composite) in dimension \( n = 10 \). The boxplots were derived by computing the average fraction of children, \( \left(1/N\right) \cdot \left\{N_{\text{GEN}} + N_{\text{FNSO}}\right\} \) each algorithm is allowed to contribute during offspring creation in AMALGAM-SO for each optimization run, using information from all restarts runs, after which the boxplot is derived from the outcome of the 25 individual optimization trials. Hence, for each separate optimization trial the fractions sum up to 1.

For unimodal function 1, the boxplots fall on a single point without any variability between the 25 different optimization

---

Fig. 5. Unimodal functions: Empirical distribution functions of \( SP^1 \) divided by \( SP^1 \) of the best performing algorithm for test functions 1–6 in dimensions (a) \( n = 10 \) and (b) \( n = 30 \). Equation (8) describes how to derive \( SP^1 \) from the numbers listed in Tables IV–VI. Small values of \( SP^1 / SP^1_{\text{best}} \) and large values for the associated CDF are preferred.

Fig. 6. Multimodal functions: Empirical distribution functions of \( SP^1 \) divided by \( SP^1 \) of the best performing algorithm for test functions 7–12 in dimensions (a) \( n = 10 \) and (b) \( n = 30 \). Small values of \( SP^1 / SP^1_{\text{best}} \) and large values for the associated CDF are preferred.

Fig. 7. Empirical distribution functions of G-CMA-ES and AMALGAM-SO on test functions 1–15 in dimension \( n = 50 \). Values of \( SP \) and \( SP_{\text{best}} \) are derived from Table VI.
For this function, a single run is required to find solutions that are within the specified tolerance limit of the global optimum. The mean of the boxplots simply represents the number of children each algorithm is allowed to create in the first run, prior to any information about the response surface is collected.

For multimodal function 9, the boxplots clearly illustrate the importance of the GA algorithm for population evolution. For this particular problem, the CMA evolutionary search strategy is not very efficient, and needs to be augmented with other search operators to find the global minimum within the predefined number of function evaluations. This result illuminates AMALGAM-SO’s superior performance over the G-CMA-ES, achieving a 100% success rate on this optimization problem. Note also that there is relative small dispersion around the mean of the boxplots, reflecting that the 25 different trials depict similar information about the search space. Thus, the inferred peculiarities of the problem are very similar, irrespective of the properties of the initial population, its size, and the optimization trial.

For composite function 13, a large variation appears in derived child fractions between the different optimization runs: small variations in initial population between restart runs and optimization trials cause significant variations in the fraction of children each of the individual algorithms is allowed to contribute to the offspring population. This finding is characteristic for complex fitness landscapes whose properties vary considerably within the considered search domain. In the presence of so many distinct difficulties, it is unlikely that a single genetic operator will be able to conduct an efficient search. Efficient search results for this optimization problem are obtained by the simultaneous use of multiple search methods, with a selection method to enable adaptation to the specific peculiarities at hand.

Finally, for composite test function 15, both the CMA-ES and GA receive significant preference for population evolution. This explains the relatively high success rate of a multimethod search algorithm for this particular problem (\( p_n = 80\% \)), and provides an explanation why the restart G-CMA-ES alone has been unable to locate solutions close to the global optimum. In this case, AMALGAM-SO uses the GA algorithm to “boost” the performance of CMA.

Note that for all problems considered in Fig. 8, the PSO algorithm seems to receive very little emphasis. This might indicate that the method is not very important for population evolution, and thus could be removed from AMALGAM-SO. However, a detailed comparison of the performance of CMA-GA with CMA-GA-PSO using the listed statistics presented previously in Table I, reveals that the PSO strategy is indispensable for population evolution for composite function 15. Thus, it is not only the weight of individual algorithms, but also the synergy between various search procedures that improves the efficiency and robustness of evolutionary optimization.

C. Scaling Analysis

The previous section considered only three different dimensions (\( n = 10 \)) for the benchmark functions. To better understand the scaling behavior of AMALGAM-SO, we follow the investigations of [20] and derive the relationship between dimensionality of the test problem, and the average number of function evaluations needed to find solutions with the tolerance limit. Fig. 9 presents the results of this analysis for test problems 1, 6, 9, and 11. These graphs were generated using 100 different optimization trials for each individual function and allow a rough determination of the measured efficiency of AMALGAM-SO. The presented results are representative for the other functions as well, and therefore we only present a subset that covers the range of difficulties involved. These results were obtained using \( \bar{N} = n_i \), i.e., the size of the initial population used in AMALGAM-SO is similar to the dimensionality of the problem. Note that the depicted lines illustrate an approximate linear scaling of AMALGAM-SO between \( n = 5 \) and \( n = 60 \) for each of the individual test functions. This is an
encouraging result, demonstrating that the algorithm is efficient and scales well with increasing dimension. In addition, the presented lines appear quite smooth, suggesting that our multimethod search strategy is well designed.

D. Test Function With Noise Induced Multimodality

The test functions considered so far have been restricted to deterministic problems whose objective function returns the same value each time the same \( \mathbf{x} \) is evaluated. Functions 4 and 17 explicitly include a noise term, but this noise vanishes when approaching the global minimum. Many real-world problems, however exhibit some form of noise in the estimation of the objective function that is persistent and can significantly alter the shape of the response surface. That is, given a fixed parameter vector \( \mathbf{x} \), the resulting objective function, \( f(\mathbf{x}) \) becomes a random variate. Such problems can confuse optimization algorithms considerably. Some functions can even change their topology from unimodal to multimodal depending on the noise strength, \( \varepsilon \). In the optimization literature, these functions have been called functions with noise induced multimodality (FNIM). In this final section, we test the robustness and efficiency of AMALGAM-SO on this class of problems.

We consider test function \( f_3 \) that is described in detail in [35]

\[
f_3(\mathbf{x}) = a - \frac{r_2^2 + \sum_{i=M_1}^{M_2} (x_i + z_i)^2}{b + r_3^2} - r_3^2
\]  

(9)

where

\[
r_1 = \sqrt{\sum_{i=1}^{M_1-1} x_i^2}, \quad r_3 = \sqrt{\sum_{i=M_2}^{n} x_i^2}, \quad 1 \leq M_1 < M_2 \leq n
\]  

(10)

with values of \( a = 5, b = 1, n = 40, M_1 = 23, M_2 = 30, \) and \( \varepsilon = 3 \) to result in significant multimodality. The global optimizer is found at

\[
\mathbf{x} = (0, \ldots, 0, \hat{x}_{M_2}, \ldots, \hat{x}_n)
\]  

(12)

where \( \hat{x}_{M_1}, \ldots, \hat{x}_n \) are located on a hypersphere of radius \( \sqrt{\sum_{i=M_1}^{M_2} x_i^2} = b \) and origin at \( (0, \ldots, 0) \) in the \( \mathbb{R}^{n-M_2+1} \) subspace. Results presented in [7] demonstrate that this function cannot be adequately solved with CMA-ES. This study therefore provides an excellent test case for AMALGAM-SO.

Fig. 10 shows the sampled \( (x_{n-1}, x_n) \) parameter space with AMALGAM-SO for test function \( f_3 \) using \( n = 5, b = 1, n = 40, M_1 = 23, M_2 = 30, \) and \( \varepsilon = 3 \). The black circle depicts the true optimum, whereas the gray circle illustrates the average radial distance estimated with CMA-ES.

The presence of noise in \( f_3 \) avoids AMALGAM-SO to converge to a single best solution, but instead to find multiple different solutions. These solutions center closely around the global minimum, and occupy the entire circle quite evenly. The average radial distance of the AMALGAM-SO sampled solutions to the origin is about 3.29, which is in very close agreement with the optimum value radius of \( \sqrt{11} \approx 3.32 \) much better than the average distance of approximately 4.3 derived with the CMA-ES. Our numerical results demonstrate that the GA generates about 50% of the offspring during the search for this particular function, explaining why AMALGAM-SO finds solutions that more closely approximate the true optimum. These findings provide further support for the claim that self-adaptive multimethod search has important advantages over search methods that use a single operator for population evolution.
VII. CONCLUSION

In the last few decades, the field of evolutionary computation has produced a large number of algorithms to solve complex, real world search and optimization problems. Most of these algorithms implement a single universal operator for population evolution. In this paper, we have shown that improvements to the efficiency of evolutionary search can be made by adopting a concept of self-adaptive, multimethod optimization. A hybrid multimethod optimization framework has been presented that can handle a wide variety of response surfaces efficiently. This approach is called A MultiAlgorithm Genetically Adaptive Method for Single Objective Optimization (AMALGAM-SO), to evoke the image of a search algorithm that blends the attributes of the best available individual optimization methods. A comparative study of different combinations of commonly used evolutionary algorithms has demonstrated that the best performance of AMALGAM-SO is obtained when the covariance matrix adaptation (CMA) evolution strategy, genetic algorithm (GA), and particle swarm optimizer (PSO) are simultaneously used for population evolution. Benchmark results in 10, 30, and 50 dimensions using synthetic functions from the special session on real-parameter optimization of CEC 2005 show that AMALGAM-SO obtains similar efficiencies as existing algorithms on relatively simple problems, but is increasingly superior for more complex higher dimensional multimodal optimization problems. In addition, AMALGAM-SO scales well with increasing number of dimensions, and converges in the close proximity of the global minimum for functions with noise induced multimodality.

The results presented herein demonstrate that self-adaptive multimethod optimization uncovers important new ways to further improve the robustness and efficiency of evolutionary search. Of course, the NFL theorem will still hold for AMALGAM-SO (as was demonstrated on the simpler problems), but initial results in this paper are very encouraging, demonstrating a significant speed up in efficiency for more complex (multimodal) higher dimensional optimization problems. The success of a multimethod search concept, however essentially relies on the quality and efficiency of individual algorithms. Thus, there is sufficient scope to further improve individual search operators.

In this study, we have not made any attempt to tune the algorithmic parameters in the AMALGAM-SO search algorithm. Instead, the various evolutionary methods were implemented using default values for their algorithmic parameters. It seems likely that further efficiency improvements of AMALGAM-SO can be made if we optimize the values of the algorithmic parameters in AMALGAM-SO using the outcome of the test suite of functions considered in this study. Moreover, theoretical analysis is needed to increase understanding why some combinations of search algorithms exhibit good convergence properties, and others show poor compatibility. We hope that the analysis presented in this paper stimulates theoreticians to develop new theories for self-adaptive multimethod search algorithms. The source code of AMALGAM-SO is written in MATLAB and can be obtained from the first author upon request.

ACKNOWLEDGMENT

The authors highly appreciate the computer support provided by the SARA Center For Parallel Computing at the University of Amsterdam, The Netherlands. The constructive comments of three reviewers have greatly improved the current manuscript.

REFERENCES


Jasper A. Vrugt received the M.S. degree in 1999 and the Ph.D. degree in 2004 (both cum laude) from the University of Amsterdam, Amsterdam, The Netherlands.

He joined the Los Alamos National Laboratory (LANL), Los Alamos, NM, in March 2005 as a Director’s Postdoctoral Fellow. Currently, he is a J. Robert Oppenheimer Fellow, holding a joint appointment between the Earth and Environmental Sciences Division (EES-6) and the Theoretical Division (T-7). His research focuses on advanced optimization, data assimilation and uncertainty estimation methods applied to computationally intensive, high-dimensional models, with applications in a wide variety of fields, including weather forecasting, hydrology, transport in porous media, and ecology.

Bruce A. Robinson is a Program Manager at the Los Alamos National Laboratory, Los Alamos, NM, and former staff member and Deputy Group Leader of the Hydrology, Geochemistry, and Geology Group. His research interests include numerical modeling of complex flow and transport systems, transport of heat, fluids, and chemicals in porous media, and application and development of optimization, data assimilation, and uncertainty estimation methods in environmental modeling.

James M. Hyman is Group Leader of the Mathematical Modeling and Analysis Group (T-7) in the Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM, and former President of the Society for Industrial and Applied Mathematics (SIAM). His research interests include building a solid mathematical foundation for difference approximations to partial differential equations and using mathematical models to better understand and predict the spread of epidemics. Most of his publications are in mathematical modeling and he has an active passion for writing quality software for numerical differentiation, interface tracking, adaptive grid generation, and the iterative solutions of nonlinear systems.