An Efficient Evolutionary Algorithm to Optimize the Choquet Integral

Muhammad Aminul Islam[a], Derek T. Anderson[b], Fred Petry[c], Paul Elmore[c]
[a]Department of Electrical and Computer Engineering, Mississippi State University, Mississippi State, MS, USA
[b]Department of Electrical Engineering and Computer Science, University of Missouri, MO, USA
[b]Geospatial Sciences and Technology Branch, Naval Research Laboratory, Stennis Space Center, MS, USA

Information fusion is an essential part of nearly all systems whose goal is to derive decisions from multiple sources. Often, a fusion solution has parameters and the goal is to learn them from data. Herein, we propose efficient *evolutionary algorithm* (EA) operators to facilitate learning the *Choquet integral* (ChI). Whereas many EAs provide a way to solve complex, unconstrained optimization tasks, most tend to perform relative poorly in light of constraints. Recently, a few EA-based approaches to optimizing the ChI have appeared. Namely, these methods focus on fixing the values of variables so conditions are met or feasible candidate pairs are identified for steps such as crossover. Herein, we introduce a new set of transparent operators that are guaranteed to naturally preserve constraints, thus eliminating the need to resort to costly evaluations and fixing of constraint violations. In particular, our method scales well to large numbers of inequality constraints, something that prior work does not. The proposed algorithm, coined *efficient ChI genetic algorithm* (ECGA), is evaluated on several synthetic data sets and it is compared to state-of-the-art algorithms. In particular, we show benefits in terms of solutions found and the time it takes to find such an answer.

1 Introduction

Data and information fusion, hereafter referred to as data fusion unless there is a reason to differentiate, is an essential part of nearly all state-of-the-art and emerging technologies. For example, self driving cars work on the basis of exploiting sensors like lidar and radar to determine safe navigation in complex and dynamic environments. Similarly, computer vision operates on the basis of extracting and exploiting a diverse set of features for tasks such as object detection. In a geospatial context, remote sensing systems frequently require the intelligent combining of human, sensor (e.g., multi and hyperspectral, synthetic aperture radar, RGB, near infrared, etc.), and algorithm outputs for tasks like object detection, land classification and earth observations, to name a few. These are just a few examples that emphasize the necessity of studying fusion for problems demanding the transformation of data to decisions.

In general, the term "fusion" is too generic; meaning the word has too much variation across domains and users.

* Corresponding author e-mail: mi160@msstate.edu

Herein, we narrow our focus and discuss a piece of the fusion puzzle called aggregation. Let $X = \{x_1, x_2, ..., x_N\}$ be N sources of data, let $z(\{x_i\})$ be the input from source i, and let $\mathbf{z} = (z(\{x_1\}), ..., z(\{x_N\}))^t$ be a vector of inputs. An aggregation operator is a mapping of data-to-data, $f(\mathbf{z}) = \mathbf{y}$. Typically, $\mathbf{y}$ is not multi-dimensional but real-valued, making the goal one of summarization.

Herein, we focus on the *Choquet integral* (ChI), a powerful aggregation operator.[1–5] The ChI performs nonlinear integration of $\mathbf{z}$ with respect to a *fuzzy measure* (FM). In many applications the FM is learned from available training data via solving an underlying optimization problem. While classical algorithms like gradient descent,[6] sub-gradient descent, and interior point can solve convex problems (e.g., quadratic programming,[7] linear programming,[8,9] and regularization based ChI[10]), they are not a good fit for non-convex objective functions. To find a solution to challenging and non-analytically solvable optimization tasks, Holland introduced a heuristic search method, the *genetic algorithm* (GA).[11,12] Herein, we propose a new efficient GA to solve the task of learning the highly inequality constrained (due to the FM) ChI. However, without loss of generality, our proposed operators can be applied to other evolutionary algorithms (EAs) and problems (beyond the ChI) when/if desired. In particular, the FM is defined over every possible subset of sources, which results in 2^N variables. By definition, a FM must satisfy the monotonicity property, which gives rise to a massive number of constraints, $N(2^{N-1} - 1)$. For example, a problem with 10 inputs has $1,024$ variables and $5,110$ constraints.

Constraint-based optimization with a GA has been addressed in a few indirect ways to date. For example, in[13–16] any candidate solution that violates a constraint is discarded. Beyond the obvious inefficiency of this approach, a problem is that some of these "invalid" solutions may actually reside in close proximity to a local or global optimum. Second, this approach is not scalable, meaning it suffers from solution starvation as the number of inputs and thus constraints increase. Another constraint handling approach[17,18] is to penalize the violating solutions to some degree so that they play a lesser role in the search process. However, an obvious issue with this approach is how do we identify a suitable penalty function and since solutions returned by the process may violate constraints, how is a valid solution obtained? The point is, prior direct (relating to the ChI) and indirect (GA optimization in general) work exists.

Another way to address constrained optimization problems is via cultural algorithms (CAs). A CA can incorporate the knowledge, prior or acquired during the optimization process, to guide the search process. Jin and Reynolds proposed a CA to solve a nonlinearly constrained optimization problem.[19] The idea is to partition the problem domain into small regions or "cells", and to label the cells as feasible, non-feasible, and unknown, with the knowledge acquired in the search process. As the algorithm progresses, it narrows the search space by removing the undesirable parts and it directs its search toward the region where the solution is most likely to reside. Obviously, this algorithm is highly computational intensive and storage demanding.

Recently, a few have developed customized GAs to address the ChI. For example, the *multiple instance learning based ChI GA* (MICIGA) was put forth.[20] MICIGA identifies the admissible range for each FM variable that does not violate the constraints and it allows the FM to change within that range. The amount of change, either "small" or "large", is stochastically determined by comparing a randomly generated value from a uniform distribution with a user specified threshold. Due to the absence of one of the main GA operators, namely crossover, the quality of the outcome from MICIGA depends primarily on random exploration and sampling of the initial population from the search space. Furthermore, as random changes occur to each FM variable at each iteration, it is hard to determine when MICIGA converges to a solution. In a separate work, the ChI was used for ontology matching.[21] This GA identifies valid candidate pairs. However, as the number of inputs, and thus inequalities increase, few (if any) pairs are available and a condition known as starvation can occur. Furthermore, the authors explicitly search for violations.[22] If any violation is found, then a bottom-up correction is applied by searching for the closest valid solution. As a result, search is not entirely "smooth" and as the number of inputs increase, more-or-less every solution is constantly being subjected to correction. Last, Mago and Modave proposed a GA for the 2-additive ChI for MCDM.[23] In particular, a 2-additive ChI restricts interaction to at most two sources, and as a result there are drastically fewer constraints to handle. However, their savings come at the expense of a trade off in representation.

In summary, none of the discussed algorithms offer an efficient way to solve the problem that we are interested in: optimizing the ChI. As a result, herein we propose an efficient algorithm, which we refer to as *efficient ChI GA* (ECGA). The proposed method introduces two new evolutionary operators that naturally enforce the monotonicity and boundary conditions of the FM. These operators can be used to create an offspring from parents during crossover as well as in mutation. Aided by these monotonic and boundary preserving operations, we also provide an efficient representation of the FM. It can be easily shown that the set of FMs is a convex set, which allows expressing a FM as a linear convex sum of the vertices of the FM convex polyhedron. However, the actual number of vertices of this convex set is exponential to N, making it computationally expensive if we wish to generate new points by enumerating vertices. To combat this, we provide a new computationally feasible representation of a FM with only N vertices of the FM set, which we refer to as a minimal set. This representation enables us to generate random points in the feasible solution space as well as prove that any point in the solution space can be reached via recombination of points. We use three monotonic operators, linear sum, product and *ordered weighted average* (OWA), to create three offspring, which compete against siblings and parents to survive to the next generation. Unlike,[20] which uses small scale mutation in the feasible interval, our mutation can make changes beyond the interval bounds, thus providing more degrees of freedom in exploring the solution space.

The remainder of this article is organized as follows. Section II provides an overview of the FM/ChI. Section III details the operations that preserve the FM properties and we introduce the new FM representation. Section IV is the overall GA and experiments are discussed in Section V.

2 Background and Preliminaries

2.1 Fuzzy Measure and Choquet Integral

The ChI is a flexible way to fuse multiple inputs in a nonlinear manner. The FM, $g : 2^X \to \mathbb{R}^+$, is a function with the following two properties; (i) (boundary condition) $g(\emptyset) = 0$, and (ii) (monotonicity) if $A, B \subseteq X$, and $A \subset B$, then $g(A) \leq g(B)$. Without loss of generality, sometimes a normality condition is imposed upon the FM for simplicity and convenience such that $g(X) = 1$. Throughout this paper, we consider this condition, which results in $2^N - 2$ variables since two FM variables, $g(\emptyset)$ and $g(X)$ are constants due to boundary and normality conditions.

The ChI of an observation $\mathbf{z} = (z_1, z_2, \ldots, z_N)^t$, where $z_i = z(\{x_i\})$, on X with respect to FM g is

$$C_g(\mathbf{z}) = \sum_{j=1}^{N} (z_{\pi(j)} - z_{\pi(j-1)}) g(A_{\pi(j)}),$$

for $A_{\pi(j)} = \{x_{\pi(j)}, \ldots, x_{\pi(N)}\}$, permutation π such that $z_{\pi(1)} \leq z_{\pi(2)} \leq \ldots \leq z_{\pi(N)}$, and $z_{\pi(0)} = 0$.[10]

Next, we discuss the concept of a partially ordered set relative to the FM. This is critical to our representation and subsequent optimization of the ChI.

2.2 Definitions

Definition 2.1 (**Partially ordered set**) Let $A = \{a_i\}$, $i = 1, 2, ..., r$ be an arbitrary set and R be a reflexive, antisymmetric, and transitive binary relation on A. Then $\mathbf{P} = (A, R)$ is called a *partially ordered set* (poset), where A is called the ground set and R is the partial order.[24]

It is trivial to show that a FM is a poset (see Def. 2.3), where the monotonicity constraints characterize the inequality relations among its variables.

Definition 2.2 **(FM poset)** Let $F = \{g(\{x_1\}), g(\{x_2\}), \ldots, g(\{x_1, x_2\}), \ldots, g(X)\}$ be a lexicographically ordered FM. According to the definition of a FM and Def. 2.1, F is a poset.

Definition 2.3 **(Monotonic poset)** A monotonic poset is $g' : 2^X \to \mathbb{R}$ that preserves the FM monotonicity property and may or may not hold the boundary and normality conditions. As such, let $F' = \{g'(\{x_1\}), ..., g'(X)\}$ be a lexicograhically ordered poset, where g' lies on the real-valued line versus $[0, 1]$.

Let $\mathcal{F} = \{F\}$ be the set of all FM posets and $\mathcal{F}' = \{F'\}$ be the set of all monotonic posets. As $\mathcal{F}'$ encompasses all FM posets, clearly $\mathcal{F} \subset \mathcal{F}'$.

Definition 2.4 **(Anti-monotonic poset)** An anti-mononotic poset, $g'' : 2^X \to \mathbb{R}$ is a poset, where $g''(x)$ is monontonically non-increasing (versus mononotonically non-decreasing), i.e., if $A, B \subseteq X$, and $A \subset B$, then $g''(A) \geq g''(B)$.

An anti-monotonic poset can easily be converted to a monotoinicy poset and vice versa by simply multiplying the elements in the set by -1, which flips the inequality relations.

3 Foundation for Evolutionary Operators

In this section, the mathematics that facilitate the efficient optimization of the ChI are discussed. These are not the final mutation and crossover operations (provided in Section 4), rather they are the numeric operations used in the EA operators. This section focuses on i) unary and multi-ary monotonic operations, ii) boundary preservation and iii) an efficient representation of a FM.

3.1 FM Property Preserving Operations

Proposition 3.1 **(Unary monotonic)** A function $f : \mathcal{F}' \to \mathcal{F}'$ is a unary monotonic operation if it is monontonically non-decreasing, i.e., $\frac{df}{dt} \geq 0, t \in \Re$.

Proof. Let $\mathcal{F}' = \{\{s_1, s_2\} : \forall s_1, s_2 \in \Re \text{ and } s_1 \geq s_2\}$ be a set of posets with partial order relation $s_1 \geq s_2$ and $F' = \{a, b\}, F' \in \mathcal{F}'$. The function f operates elementwise on F' and maps it to $G = \{f(a), f(b)\}$. In order for G to be in $\mathcal{F}'$, $f(a)$ and $f(b)$ need to satisfy the partial order relations $f(a) \geq f(b)$, meaning that f has to be monotonicially non-decreasing, i.e., $f(a) \geq f(b)$ when $a \geq b$. This condition can be written using the basic definition of gradient at point b as $\left.\frac{df}{dt}\right|_{t=b} = \frac{f(a)-f(b)}{a-b} \geq 0$. This concludes our proof. □

Some examples of unary operations are: (i) multiplication by a constant: $f(t) = wt,\ w \in \mathbb{R}^+$ and (ii) addition or subtraction by constant: $f(t) = t \pm c$. This definition can easily be extended to multi-ary operations that combine a number of FM posets to produce a monotonic poset.

Proposition 3.2 **(Multi-ary monotonic)** A multi-ary monotonic operation $f : \mathcal{F}' \times \cdots \times \mathcal{F}' \times \cdots \times \mathcal{F}' \rightarrow \mathcal{F}'$, is a n-variate monontonically non-decreasing function $f(t_1, t_2, \ldots, t_i, \ldots, t_n)$, i.e.,$\frac{\partial f}{\partial t_i} \geq 0, t_i \in \Re$. Let $\mathcal{F}' = \{\{s_1, s_2\} : \forall s_1, s_2 \in \Re$ and $s_1 \geq s_2\}$ be a set of posets with partial order relation $s_1 \geq s_2$ and $F'_1 = \{a, b\}$, $F'_2 = \{c, d\}, F'_1, F'_2 \in \mathcal{F}'$. Then a 2-variate non-decreasing function f can combine F'_1 and F'_2 to create a new poset $F'_3 = \{f(a, c), f(b, d)\}, f(a, c) \geq f(b, d)$ that is also in $\mathcal{F}'$. This proposition can easily be proved for arbitrary N posets using the same procedure described in Proposition 3.1.

Examples of monotonic operators with two (or more) variables include the (i) linear conic sum ($f(t_1, t_2) = w_1 t_1 + w_2 t_2,\ w_1, w_2 \in \mathbb{R}^+$), (ii) monomial ($f(t_1, t_2) = t_1^{w_1} t_2^{w_2},\ w_1, w_2 \in \mathbb{R}^+$) (iii) numerous t-norms and t-conorms, and (v) the OWA.

Operations, such as the linear convex sum (a special case of linear conic sum where the sum of weights are equal to one), most t-norm and t-conorm's, and OWAs map the output into $[0, 1]$; thus they also satisfy the boundary condition and therefore directly map to a valid FM. However, many other operators do not, and as such we need to apply boundary preserving operations. Next, we propose an operation that enforces the boundary and normality conditions on F', and thus transforms the monotonic poset, F' to a valid FM poset, F.

Definition 3.3 (**Boundary fixing operation**) Let the function for the boundary fixing operation be defined as

$$f(t) = \begin{cases} 0 & \text{if } t = g'(\emptyset) \\ 1 & \text{if } t = g'(X) \\ \max(\min(1, t), 0) & \text{else} \end{cases}.$$

It can easily be shown that the above example operators preserve all the FM properties; however, we demonstrate it for one case, the linear convex sum in Section 3.2.

3.2 Efficient FM Representation

In order to facilitate a geometric interpretation and ultimately an efficient representation of F, we regard elements in F as a point $(\mathbf{p})$ in a multi-dimensional space. First, we show that F is a convex set, meaning any point in this set can be represented as a linear convex sum of its vertices. Second, we show an efficient way to represent a FM in terms of drastically fewer points (versus all of its verticies).

Proposition 3.4 **(FM as a convex set)** $\mathcal{F}$, the set of all FMs on X is a convex set. That is, the convex combination of any number of points in $\mathcal{F}$ also lies in $\mathcal{F}$, i.e., $\sum_i^r w_i \mathbf{p}_i \in F$ for $r \in \mathbb{N}, \mathbf{p}_i \in \mathcal{F},\ w_i \geq 0$ and $\sum_i^r w_i = 1$.

Proof. According to Proposition 3.2, linear sum preserves monotonicity. As such, the linear convex sum of FM in a multi-dimensional space, i.e., $\sum_i^r w_i \mathbf{p}_i$ is also monotonic. Furthermore, linear convex sum yields zero when all inputs are zeros, one when all inputs are ones and its output is bounded within $[0,1]$ if the inputs range is $[0,1]$. Therefore, linear convex sum preserves the boundary conditions. Since $\sum_i^r w_i \mathbf{p}_i$, where $w_i \geq 0$ and $\sum_i^r w_i = 1$, has all the properties of a FM, this concludes our proof. □

Since $\mathcal{F}$ is a convex set it forms a convex polyhedron. The vertices of this polyhedron can be obtained by solving the inequality and equality constraints. An example is provided next. However, we remark that Avis and Fukuda provided an efficient algorithm to enumerate the vertices from constraints.[25]

Example 1. In this example we illustrate the aforementioned FM representation for $N = 3$. Let the FM be $\mathbf{g} = \{g_1,\ g_2,\ g_3,\ g_{12},\ g_{13},\ g_{23},\ g_{123}\}$, which is shorthand notation for readability, i.e., g_{12} stands for $g(\{x_1, x_2\})$. The boundary, normality, and monotonicity properties are:

$$\begin{array}{rlrl}
g_{\emptyset} = 0 & & g_{12} \leq 1 & (10)\\
g_1 \geq 0 & (1) & g_{13} \leq 1 & (11)\\
g_2 \geq 0 & (2) & g_{23} \leq 1 & (12)\\
g_3 \geq 0 & (3) & g_{12} \geq g_1 & (13)\\
g_{12} \geq 0 & (4) & g_{12} \geq g_2 & (14)\\
g_{13} \geq 0 & (5) & g_{13} \geq g_1 & (15)\\
g_{23} \geq 0 & (6) & g_{13} \geq g_3 & (16)\\
g_1 \leq 1 & (7) & g_{23} \geq g_2 & (17)\\
g_2 \leq 1 & (8) & g_{23} \geq g_3 & (18)\\
g_3 \leq 1 & (9) & g_{123} = 1. &
\end{array}$$

Since the FM values for the empty set and X are constants, we can perform our analysis based on the remaining $2^N - 2$, or 6, variables. Equations (1)-(18) determine the feasible solution space of these 6 variables. Solving these equations analytically, we obtain the vertices of the solution space, a 6-dimensional convex polyhedron. Table 1 lists the vertices along with the corresponding intersecting equations. Furthermore, any point with the convex polyhedron can be represented as the linear convex sum of these vertices, i.e.,

$$\mathbf{p}_i = \sum_i w_i \mathbf{v}_i, w_i \geq 0, \sum_i w_i = 1.$$

Minimal FM Representation: Let the set of vertices of the FM convex polyhedron be $V = \{\mathbf{v}_j\}$. Building on Example 1, we now define a minimal set of vertices, $M = \{\mathbf{m}_i\}, i = 1, 2, \ldots, 2^N - 2$. Specifically, vertices in M can only be represented by themselves, i.e., they cannot be expressed as a linear convex sum of the remaining vertices in V. There exists exactly one $\mathbf{m}_i \in M$ corresponding to each variable, $g(A), A \in 2^X \setminus X$ such that

$$m_i(B) = \begin{cases} 1 & \text{if } B \supseteq A \\ 0 & \text{else} \end{cases}. \tag{19}$$

Table 1: Vertices for 3 sources

Vertex	g_1	g_2	g_3	g_{12}	g_{13}	g_{23}	Intersecting equations
$\mathbf{v}_1$	0	0	0	0	0	0	1∼6, 13∼18
$\mathbf{v}_2$	0	0	0	0	0	1	1∼5, 12
$\mathbf{v}_3$	0	0	0	0	1	0	1∼4, 6, 11
$\mathbf{v}_4$	0	0	0	1	0	0	1∼3, 5, 6, 10
$\mathbf{v}_5$	0	0	0	0	1	1	1∼4, 11, 12
$\mathbf{v}_6$	0	0	0	1	0	1	1∼3, 5, 10, 12
$\mathbf{v}_7$	0	0	0	1	1	0	1∼3, 6, 10, 11
$\mathbf{v}_8$	0	0	0	1	1	1	1∼3, 10∼12
$\mathbf{v}_9$	1	0	0	1	1	0	2, 3, 6, 7, 10, 11, 13, 15
$\mathbf{v}_{10}$	0	1	0	1	0	1	1, 3, 5, 8, 10, 12, 14, 17
$\mathbf{v}_{11}$	0	0	1	0	1	1	1, 2, 4, 9, 11, 12, 16, 18
$\mathbf{v}_{12}$	1	0	0	1	1	1	2, 3, 7, 10∼13, 15,
$\mathbf{v}_{13}$	0	1	0	1	1	1	1, 3, 8, 10∼12, 14, 17,
$\mathbf{v}_{14}$	0	0	1	1	1	1	1, 2, 9∼12, 16, 18,
$\mathbf{v}_{15}$	1	1	0	1	1	1	3, 7, 8, 10∼15, 17
$\mathbf{v}_{16}$	1	0	1	1	1	1	2, 7, 9∼13, 15, 16, 18
$\mathbf{v}_{17}$	0	1	1	1	1	1	1, 8∼12, 14, 16∼18
$\mathbf{v}_{18}$	1	1	1	1	1	1	7∼18

Any vertex $\mathbf{v} \in V$ in the polyhedron can be expressed in terms of the minimal set using the following equation

$$\mathbf{v} = \min(\mathbf{1}, \sum_{i}^{N} c_i \mathbf{m}_i), c_i \in \{0, 1\}. \tag{20}$$

The minimal set for Example 1 is $M = \{\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3, \mathbf{m}_4, \mathbf{m}_5, \mathbf{m}_6\}$, which correspond to vertices $\mathbf{v}_2$, $\mathbf{v}_3$, $\mathbf{v}_4$, $\mathbf{v}_9$, $\mathbf{v}_{10}$, and $\mathbf{v}_{11}$ respectively (from Table 1). Table 2 shows that all of the vertices in $\mathcal{F}$ can be expressed in terms of the minimal set, M, with respect to its corresponding coefficients (c's). Even though we show this for $N = 3$, it is easy to generalize to the case of any N.

In summary, the significance of Eq. 19 is it tells us that we can work with a small set of vertices and exploit it in EA operations like weighted recombination to produce new valid points (FMs). This efficient representation forms the basis of our EA search algorithm.

Table 2: Coefficients for the minimal set for generating vertices

c_1	c_2	c_3	c_4	c_5	c_6	Resultant vertex
0	0	0	0	0	0	$\mathbf{v}_1$
1	0	0	0	0	0	$\mathbf{v}_2 = \mathbf{m}_1$
0	1	0	0	0	0	$\mathbf{v}_3 = \mathbf{m}_2$
0	0	1	0	0	0	$\mathbf{v}_4 = \mathbf{m}_3$
1	1	0	0	0	0	$\mathbf{v}_5$
1	0	1	0	0	0	$\mathbf{v}_6$
0	1	1	0	0	0	$\mathbf{v}_7$
1	1	1	0	0	0	$\mathbf{v}_8$
0	0	0	1	0	0	$\mathbf{v}_9 = \mathbf{m}_4$
0	1	0	0	1	0	$\mathbf{v}_{10} = \mathbf{m}_5$
0	0	0	0	0	1	$\mathbf{v}_{11} = \mathbf{m}_6$
1	0	0	1	0	0	$\mathbf{v}_{12}$
0	1	0	0	1	0	$\mathbf{v}_{13}$
0	0	1	0	0	1	$\mathbf{v}_{14}$
0	0	0	1	1	0	$\mathbf{v}_{15}$
0	0	0	1	0	1	$\mathbf{v}_{16}$
0	0	0	0	1	1	$\mathbf{v}_{17}$
1	1	1	0	0	0	$\mathbf{v}_{18}$

4 Efficient GA for the ChI

The previous sections outlined the mechanics (operators and representation) of our evolutionary optimization approach. In this section, we focus on how to use these ideas for (i) initialization, (ii) selection, (iii) crossover and (iv) mutation in a GA. Again, while we focus on a GA, the operators discussed are equally applicable for other EA-based optimization approaches.

4.1 Initialization

Our first action is to determine an initial set of candidate solutions. In order to ensure that subsequent operations have free range over the full feasible space, we begin by randomly picking points in a range between 1 to N from M (see Eq. 19). Next, we generate $n_p - N$ additional individuals using equation

$$\mathbf{p} = \min(\mathbf{1}, \sum_{j=1}^{n_r} w_j \mathbf{t}_j),\ \mathbf{t}_j \in M \text{ and } 1 \leq n_r \leq N \tag{21}$$

where n_p is the population size, $n_r \sim U(1, N)$ is an integer randomly selected from a uniform distribution, $\mathbf{t}_j$ is a point randomly selected from M and $w_j \in U(0, 1)$ is the corresponding weight sampled from a uniform distribution.

```
Algorithm 1: Initialization
1  Input initial population size, $n_P$
2  Initialize input population, $P$, to empty
3  Generate the minimal set, $M = \{\mathbf{m}_k\}, k = 1, 2, \ldots, N$ using Eq. 19 and add these to $P$
4  for $i = 1, \cdots, n_P - N$ do
5      Randomly select the number of vertices, $s \sim U(1, N)$
6      Randomly pick $n_r$ vertices, $\mathbf{t}_j \in M, j = 1, 2, \ldots, s$
7      Select $w_j \sim U(0, 1), j = 1, 2, \ldots, s$
8      Create a new valid FM, $\mathbf{p}$, using Eq. 21
9      Add $\mathbf{p}$ to $P$
10 Return $P$
```

4.1.1 Selection

Any GA selection method is applicable here. However, in this article we use non-linear ranking with stochastic selection and elitism (to ensure that that the best solution is not lost from one iteration to the next). Algorithm 15 summarizes our selection process.

```
Algorithm 2: Selection
1  Input population, $P$
2  Input number of offsprings to select, $n_o$
3  Calculate the rank of an individual, $\mathbf{p}_i \in P$ based on its position in the ascending sorting order of fitness scores,
     $f_r(\mathbf{p}_i)$
4  Square root the rank, $f_s(\mathbf{p}_i) = \sqrt{f_r(\mathbf{p}_i)}$
5  Calculate probability of an individual $\mathbf{p}_i$ as $\psi(\mathbf{p}_i) = \frac{f_s(\mathbf{p}_i)}{\sum_i f_s(\mathbf{p}_i)}$
6  for $i = 1, \ldots, n_o$ do
7      $\lambda_i = 0$
8  $r \sim U(0, \frac{1}{n_s})$
9  $sum = 0$
10 for $i = 1, \ldots, n_s$ do
11     $sum = sum + \psi(\mathbf{p}_i)$
12     while $sum \leq r$ do
13         $\lambda_i = \lambda_i + 1$
14         $r = r + \frac{1}{n_o}$
15 Return index of the selected individuals, $\lambda = (\lambda_1, \cdots, \lambda_{n_s})$
```

4.1.2 Crossover

While multiple parents can be recombined to spawn an offspring, we limit our discussion to the case of two herein. Our approach involves a monotonic operation followed by the boundary operation, discussed in Section III. Specifically, in

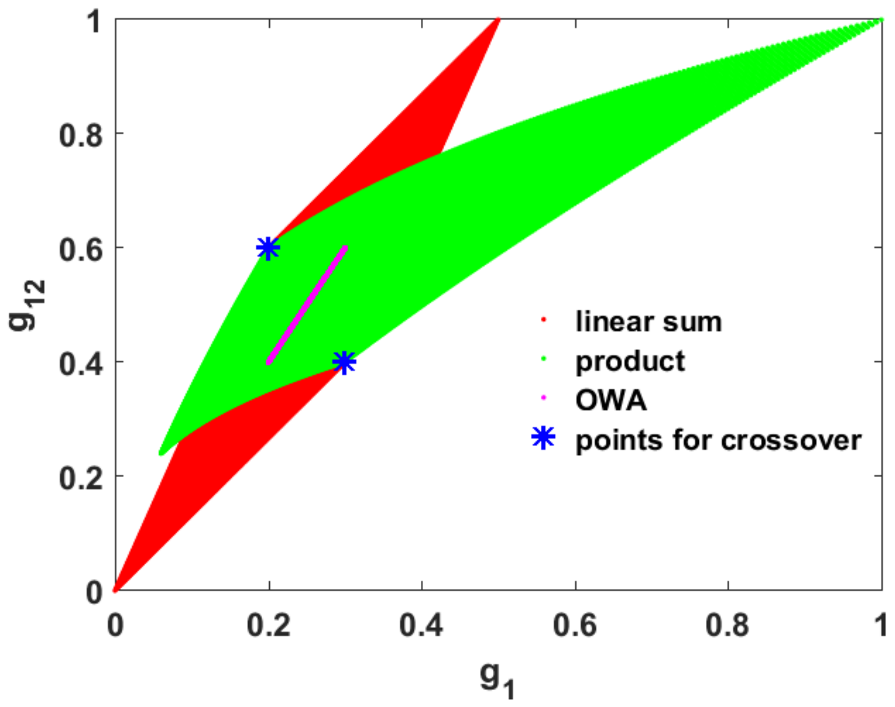


Fig 1: Example illustration of crossover for the case of two arbitrary points at $g_1 = (0.2, 0.6)$ and $g_{12} = (0.3, 0.4)$; a subset of the multi-dimensional optimization space. The different sets (represented by color coded points) show the range of the underlying linear sum, product and OWA operators on these two parents.

this section we focus on three operators that utilize sum, product and sorting operations. Let $\mathbf{p}_1$ and $\mathbf{p}_2$ be the parents. The three methods explored for generating a child are the following:

1. linear sum: $\tilde{\mathbf{p}}_1 = c_{11}\mathbf{p}_1 + c_{12}\mathbf{p}_2$,
2. monomial: $\tilde{\mathbf{p}}_2 = \mathbf{p}_1^{c_{21}}\mathbf{p}_2^{c_{22}}$,
3. OWA: $\tilde{\mathbf{p}}_3 = c_3 \max(\mathbf{p}_1, \mathbf{p}_2) + (1 - c_3)\min(\mathbf{p}_1, \mathbf{p}_2)$,

where the coefficients c_{11}, c_{12}, c_{21}, c_{22}, and c_3 are randomly generated from a uniform distribution, $U(0, 1)$. Figure 1 illustrates where the offspring reside in relation to the parents. Algorithm 6 is a formal description of our crossover algorithm.

4.1.3 Mutation

Mutation is what truly enables global search for solutions over the feasible space. Algorithm 14 is a formal description of our approach.

Algorithm 3: Crossover using recombination of two parents

```
1 Input crossover rate, $r_C$
2 Input the parents, $\mathbf{p}_1$, and $\mathbf{p}_2$
3 if $r \sim U(0,1) \leq r_C$ then
4     Randomly select the coefficients, $c_{11}$, $c_{12}$, $c_{21}$, $c_{22}$, and $c_3$ from a uniform distribution
5     Calculate the three offsprings, $\tilde{\mathbf{p}}_1 = c_{11}\mathbf{p}_1 + c_{12}\mathbf{p}_2$, $\tilde{\mathbf{p}}_2 = \mathbf{p}_1^{c_{21}}\mathbf{p}_2^{c_{22}}$, and
      $\tilde{\mathbf{p}}_3 = c_3 \max(\mathbf{p}_1, \mathbf{p}_2) + (1 - c_3) \min(\mathbf{p}_1, \mathbf{p}_2)$
6 Evaluate the fitness scores of $\mathbf{p}_1$, $\mathbf{p}_2$, $\tilde{\mathbf{p}}_1$, $\tilde{\mathbf{p}}_2$, and $\tilde{\mathbf{p}}_3$ and return the one with the best fitness score.
```

Algorithm 4: Mutation

```
1  Input chromosome $\mathbf{p}$
2  Input mutation point, $i \sim U(1, n)$
3  Generate a random value, $w \sim U(-1, 1)$
4  if $w \geq 0$ then
5      Compute upper bound, $ub_{p_i}$ of $p_i$ from its superset variables
6      Calculate $p_i = min(ub_{p_i}, p_i + w)$
7      Find residual offset, $w_r = max(0, w - ub_{p_i})$
8      Combine $\mathbf{p}$ with $i$th vertex in minimal set as $\mathbf{p} = min(\mathbf{1}, \mathbf{p} + w_r\mathbf{m}_i)$
9  else
10     Compute lower bound, $lb_{p_i}$ of $p_i$ from its subset variables
11     Calculate $p_i = max(lb_{p_i}, p_i + w)$
12     Find residual offset, $w_r = min(0, w - lb_{p_i})$
13     Combine $\mathbf{p}$ with $i$th vertex in anti-monotonic set as $\mathbf{p} = min(\mathbf{1}, \mathbf{p} + w_r\mathbf{q}_i)$
14 Return $\mathbf{p}$
```

In Algorithm 14 we start by selecting mutation points, $\mathbf{p}_i$. Next, an offset is selected, $w \in [-1, 1]$, which determines the "direction" of alteration. If this value is positive then we compare it with the feasible upper bound, ub_{p_i}, computed from the superset variables of g_i and make the admissible change in p_i within this bound. Last, an residual offset is calculated, $w_r = max(0, w - ub_{p_i})$, and used to combine the individual with the ith vertex in M via

$$\mathbf{p} = min(\mathbf{1}, \mathbf{p} + w_r\mathbf{m}_i).$$

On the other hand, if this shift variable is negative we make an admissible change within the lower bound. and we use the residual offset, $w_r = min(0, w - lb_{p_i})$, to combine with an anti-monotonic point, $\mathbf{q}_i$, to preserve monotonicity. Like the minimal set, we generate an anti-monotonic point, $\mathbf{q}_i \in Q$, for each variable, $g(A)$, $A \subseteq 2^X \setminus X$, as follows

$$q_i(B) = \begin{cases} 1 & \text{if } B \subseteq A \\ 0 & \text{else} \end{cases}. \tag{22}$$

We can pre-compute minimal and anti-monotonic sets, Q, to increase computational efficiency. Table 3 shows the sets, subset and superset variables for the case of Example 1 ($N = 3$). As we can see, the subset and superset variables for each variable can easily be computed from the anti-monotonic set and minimal set respectively. For each variable, g_i, we look for variables that are 1s in the corresponding vertex, $\mathbf{m}_i$, which gives the superset (including g_i). In order to create strictly superset variables, g_i is excluded. In the same manner, subset variables can be obtained from anti-monotonic set Q.

Table 3: Minimal set, anti-monotonic set, and subset and superset variables for three inputs

	Minimal Set									Anti-monotonic Minimal Set								
Variables	Vertex	g_1	g_2	g_3	g_{12}	g_{13}	g_{23}	g_{123}	Superset variables	Point	g_1	g_2	g_3	g_{12}	g_{13}	g_{23}	g_{123}	Subset variables
g_1	$\mathbf{m}_1$	1	0	0	1	1	0	1	g_{12}, g_{13}, g_{123}	$\mathbf{q}_1$	1	0	0	0	0	0	0	$\emptyset$
g_2	$\mathbf{m}_2$	0	1	0	1	0	1	1	g_2, g_{12}, g_{123}	$\mathbf{q}_2$	0	1	0	0	0	0	0	$\emptyset$
g_3	$\mathbf{m}_3$	0	0	1	0	1	1	1	g_3, g_{13}, g_{123}	$\mathbf{q}_3$	0	0	1	0	0	0	0	$\emptyset$
g_{12}	$\mathbf{m}_4$	0	0	0	1	0	0	1	g_{123}	$\mathbf{q}_4$	1	1	0	1	0	0	0	g_1, g_2
g_{13}	$\mathbf{m}_5$	0	0	0	0	1	0	1	g_{123}	$\mathbf{q}_5$	1	0	1	0	1	0	0	g_1, g_3
g_{23}	$\mathbf{m}_6$	0	0	0	0	0	1	1	g_{123}	$\mathbf{q}_6$	0	1	1	0	0	1	0	g_2, g_3

5 Experiments

To demonstrate the quality of ECGA, we compare it against prior work on several different optimization functions. As there are no benchmark nonlinear ChI optimization problems, we propose experiments such that the location of the global optimum varies across cases. Specifically, we compare ECGA to the GA in[22] and,[20] referred to hereafter as FVFMGA and MICIGA respectively. We do not compare to,[21] as it suffers extremely from early termination due to starvation of crossover pair identification and it does not scale well at all.

The following experiments are performed. First, we demonstrate ECGA, FVFMGA and MICIGA on a small scale problem; three inputs, thus seven variables. Second, we investigate the impact of increasing the number of variables. Third, we demonstrate scalability versus execution time, which gives us insight into the computational complexity of these approaches. Last, we investigate the impact of the different equations for crossover in ECGA.

In each of our optimization tasks, we consider the following problem,

$$\min_{\mathbf{u}} \; h(C_{\mathbf{u}}) = e(\mathbf{u}),$$

subject to

$$u_i \leq u_j \text{ if } u_i = g(A), u_j = g(B) \text{ and } A \subset B,$$
$$\forall i, j \in \{1, 2, \ldots, 2^N - 1\},$$
$$u_i = 1, \text{ if } u_i = g(X),$$

where $\mathbf{u}$ is a vector representation of the FM variables, h is a function of ChI with respect to $\mathbf{u}$ and e is a function of $\mathbf{u}$. For simplicity and computational efficiency, we can chose to use a binary encoded index for variables in $\mathbf{u}$. This means that for a three input case, indices $1, 3,$ and 4 corresponds to binary representation of $001, 011,$ and 100 respectively and, consequently, $u_1 = g(\{x_1\}), u_3 = g(\{x_1, x_2\}), u_4 = g(\{x_3\})$ and so forth.

For the first three functions, we use the sum of squared residuals

$$e_i = \sum_{j=1}^{2^N-1} (u_j - c_j)^2, i = 1, 2, 3. \tag{23}$$

The idea is to compare the candidate solution, $\mathbf{u}$, to an underlying solution (ground truth) FM, $C = [c_1\ c_2\ \ldots\ c_{2^N-1}]$.

The first three functions are focused on the ChI. In the fourth data set we use a benchmark data set, the Rastrigin function. This is performed in order to demonstrate that our methods are equally applicable to any multi-variate function with inequality constraints. The conventional Rastrigin function spans $[-5.12, 5.12]$ and has a global minimum at zero. Without loss of generality, we modified the function by scaling and translating it so the function now spans $[0, 1]$ and the global minimum is now at 0.5. Thus, the function e_4, i.e., modified Rastrigin function, is

$$e_4(\mathbf{u}) = 10 * (2^N - 2) + \sum_{j=1}^{2^N-2} [u_j^2 - 10cos(2\pi u_j)], \tag{24}$$

Figure 2 shows the surface plot of Eq. 24.

5.1 Experiment 1: Small Scale Optimization Problem

This experiment has N=3, thus seven variables. The function e_1, e_2 and e_3 correspond to OWAs with weights $[0.1, 0.2, 0.7]$, (soft min), $[0.7, 0.2, 0.1]$ (soft max) and $[0.3, 0.4, 0.3]$ (mean like) respectively (see Table 4 for co-

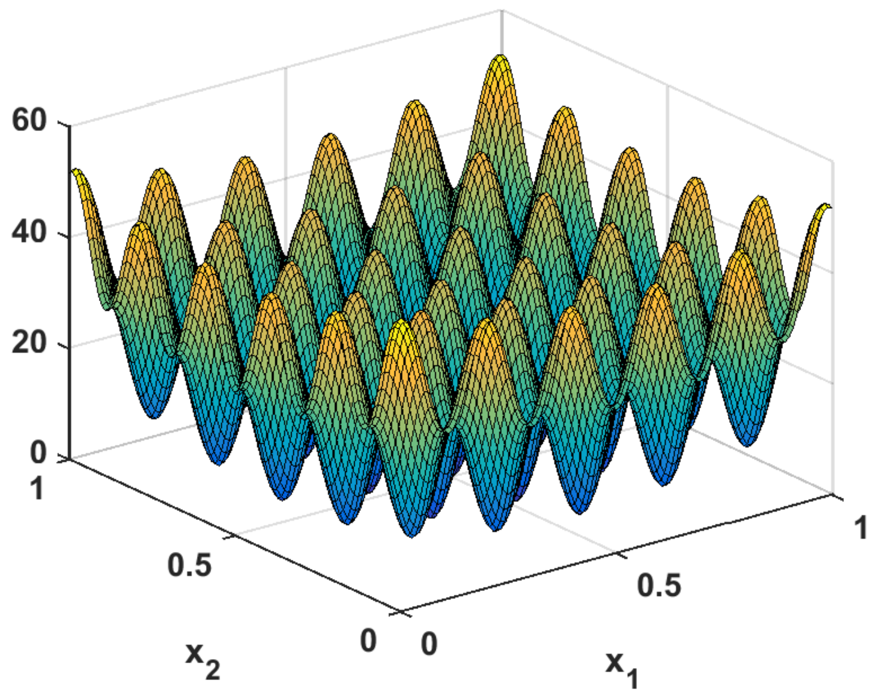


Fig 2: Surface plot of the modified Rastrigin function for two variables, which is the fourth data set explored herein. A non-ChI data set was selected to demonstrate the generalizability of our proposed ECGA.

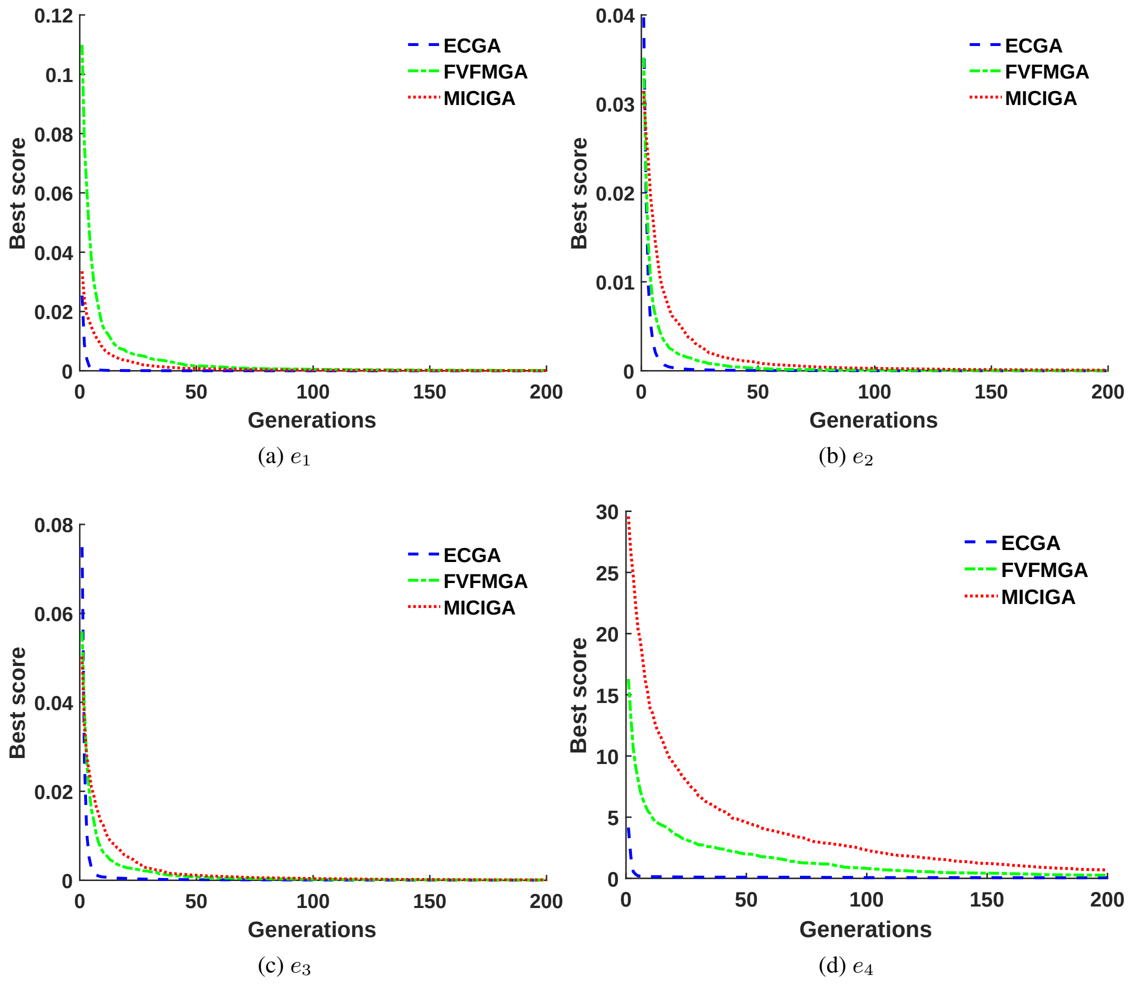
efficients). These three functions were selected in order to span the range of optimistic (union like), pessimistic (intersection like), and expected value (e.g., mean) like. The global optimum for each function is at zero. The fourth function is the Rastrigin function.

Table 4: Coefficients for e_1, e_2, and e_3 for three inputs

function	c_1	c_2	c_3	c_4	c_5	c_6	c_7
e_1	0.1	0.1	0.3	0.1	0.3	0.3	1
e_2	0.7	0.7	0.9	0.7	0.9	0.9	1
e_3	0.3	0.3	0.7	0.3	0.7	0.7	1

Binary encoded coefficient index is used.

The default parameter settings are as follows. The population size was 100 and 500 generations were used. For ECGA and FVFMGA, crossover rates vary from 0.6 to 0.9 in step increments of 0.1 and mutation rates of $0.05, 0.1, 0.2,$ and 0.3 were used. For MICIGA, we used the implementation provided by the authors, which can be found at `https://github.com/GatorSense/MICI`. While MICIGA was designed to solve multiple instance tasks, we adapted it to solve our four functions. MICIGA parameter settings are, small mutation rate from 0.6 to 0.9 in step sizes of 0.1, variance around sample mean of $0.05,\ 0.1,\ 0.2,$ and 0.3. Note MICIGA does not have



(a) e_1 (b) e_2 (c) e_3 (d) e_4

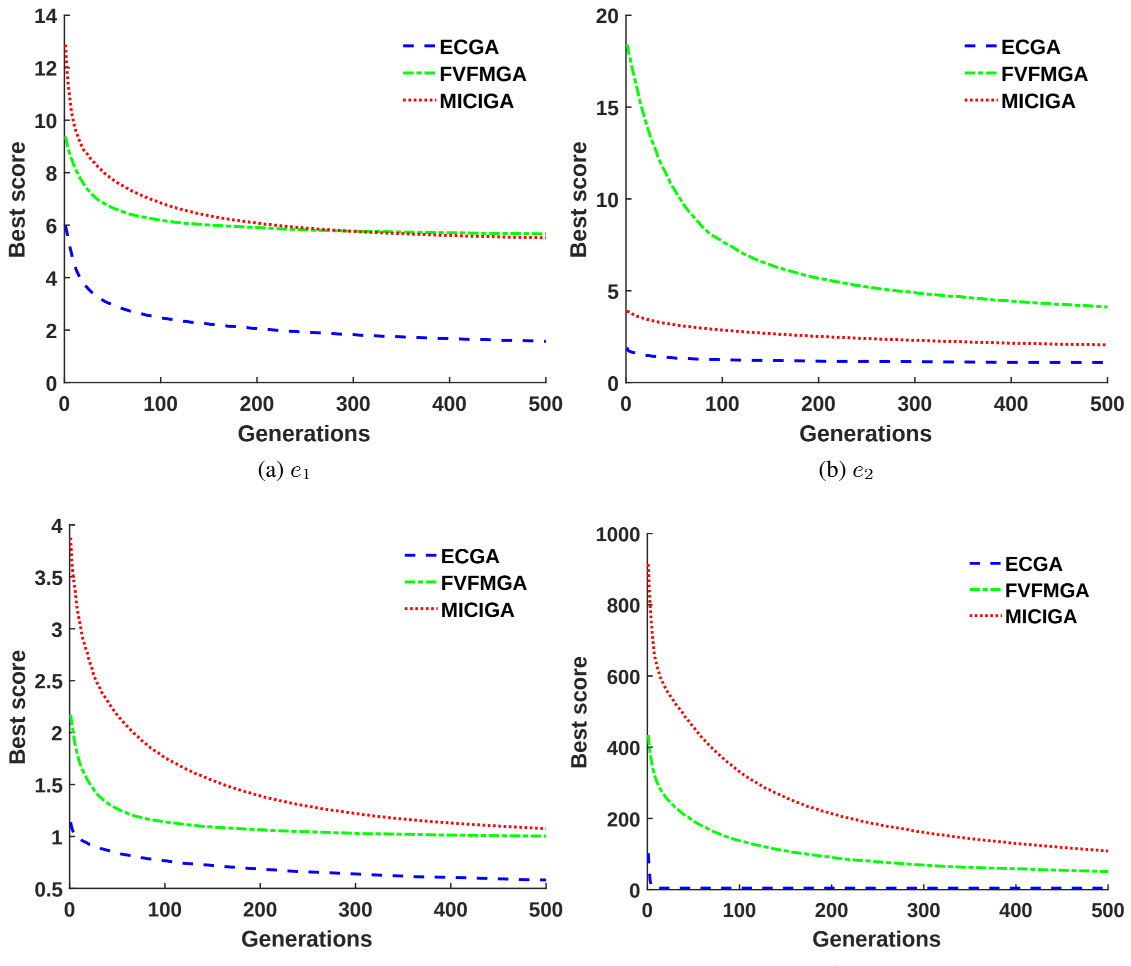
Fig 3: Experiment 1. Comparison of best fitness scores for $N = 3$ for e_1, e_2, e_3, and e_4.

crossover. Instead it performs small scale mutation. Each experiment was repeated 50 times and for each method, the average score for the best performer is reported. The best fitness score is used as the evaluation metric.

Figure 3 shows the results of Experiment 1. We can see that all three methods achieve the same best fitness score at some point (number of generations). This should be expected for a simple problem. However, ECGA outperforms the other algorithms in all four problems with respect to the number of generations needed to converge. Of the three methods, MICIGA has the worst results, which might be expected as it involves only mutation (taking more generations to get to an answer).

5.2 Experiment 2: Scalability

Experiment 2 has $N = 6$, 63 variables and 186 monotonicity constraints. The three error functions had coefficients generated randomly from a uniform distribution with ranges $[0, 1]$, $[0, 0.5]$ and $[0.5, 1]$ respectively. A population size



(a) e_1 (b) e_2 (c) e_3 (d) e_4

Fig 4: Experiment 2. Comparison of best fitness scores for $N = 6$ for e_1, e_2, e_3, and e_4.

of 200 and 500 generations was ran. The method specific parameter settings are the same as Experiment 1. Figure 4 shows the result.

Figure 4 shows the effectiveness of ECGA. Specifically, none of the other methods achieved the same quality of solution. On e_4, FVFMGA attained a solution as good as ECGA, but it took approximately 350 iterations versus 10. Also, if we look at the best score from the initial population, ECGA has the lowest score for all four error functions. This result is evidence that the initial population of ECGA is likely more distributed over the solution space. In summary, Experiment 2 demonstrates that our proposed method is suitable for larger scale problems, compared to prior work.

Table 5: Execution time in seconds

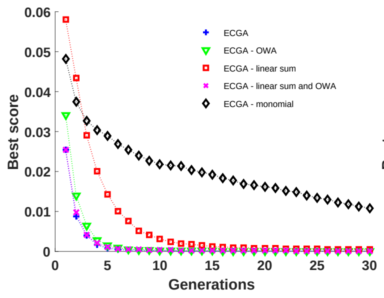
	Method	3 inputs					6 inputs				
		e_1	e_2	e_3	e_4	Average	e_1	e_2	e_3	e_4	Average
Minimum	ECGA	1.87	1.83	1.81	2.54	2.01	6.17	6.07	6.02	7.36	**6.41**
	FVFMGA	1.35	1.33	1.42	1.64	**1.44**	12.92	12.86	12.78	12.96	12.88
	MICIGA	5.47	5.49	5.53	5.50	5.50	52.17	44.35	48.60	51.26	49.09
Mean	ECGA	2.25	2.16	2.11	3.00	2.38	7.41	7.69	7.65	10.23	**8.25**
	FVFMGA	1.43	1.44	1.53	1.77	**1.54**	13.49	13.20	13.04	13.65	13.34
	MICIGA	6.44	6.52	6.45	6.92	6.58	82.45	71.88	74.18	85.00	78.38
Maximum	ECGA	3.38	2.58	2.59	3.67	3.05	10.09	10.25	10.32	12.75	**10.85**
	FVFMGA	1.71	2.16	2.19	2.10	**2.04**	14.97	14.64	15.35	15.14	15.02
	MICIGA	8.24	7.77	8.07	8.69	8.19	122.40	109.69	103.43	125.05	115.14
Best	ECGA	2.36	2.09	2.33	3.12	2.47	6.17	6.07	7.01	10.95	**7.55**
	FVFMGA	1.37	1.42	1.43	1.66	**1.47**	13.62	12.86	12.81	12.96	13.06
	MICIGA	7.14	7.31	7.31	8.18	7.48	89.76	61.22	66.40	119.00	84.09

5.3 Experiment 3: Computational Performance

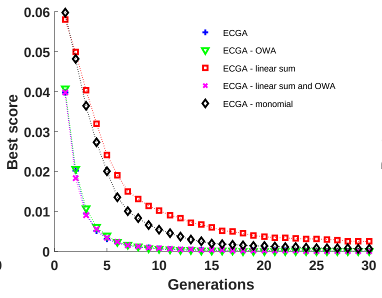
Our experiments were run on a Windows PC with an Intel Xeon CPU EP-2650 at 2GHz clock speed and 64GB of RAM. All parameters were kept the same for the $N = 3$ and $N = 6$ cases, except for population size. Each experiment was repeated for 50 times and the average running time for each combination of parameters (e.g., crossover rate = 0.6 and mutation rate=0.05 for ECGA and FVFMGA) was recorded. Table 5 lists the minimum, mean, maximum, and the best running time. Here, the best corresponds to the scenario that yields the best results. The number of variables increased by 9 times from $N = 3$ to 6 while running time for ECGA, FMFMGA, and MICIGA increased by 3.47, 8.65, and 11.91 respectively. While FVFMGA has the lowest execution time for $N = 3$, it spends on average 1.6 times more time than ECGA for $N = 6$. The reason is, FVFMGA checks each monotonicity relation for each variable, and there are $N(2^N - 1)$ such checks to make. This causes an extreme computational burden for FVFMGA as N grows. MICIGA has the maximum execution time due to the mutation operation of a large number of variables (60% to 80% of the total variables go through small scale mutation each iteration), which involves calculating the lower and upper bounds and assigning a random value. In summary, we conclude that our method scales well to problem size (N).

5.4 Experiment 4: ECGA Crossover Operators

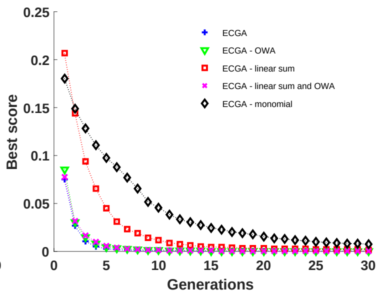
In Experiment 4, we explore the significance of the different operators involved in crossover; linear sum, monomial and OWA. Figure 6 reports the best fitness for the following scenarios; crossover with (i) all three operations, (ii) just linear sum, (iii) just monomial, (iv) just OWA, and (v) linear sum and OWA. As we can see, OWA, linear sum and OWA, and all three exhibit almost identical trends with best results for e_1, e_2, and e_3. However, OWA's performance appears to degrade considerably relative to the others for e_4. For relatively large problems, e.g., 63 variables, the best score trends



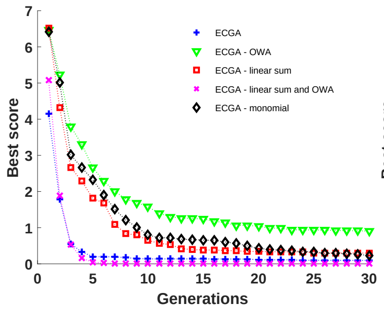
(a) e_1 for three inputs



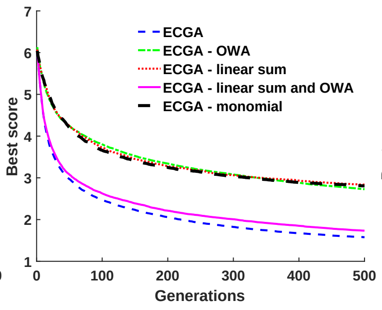
(b) e_2 for inputs



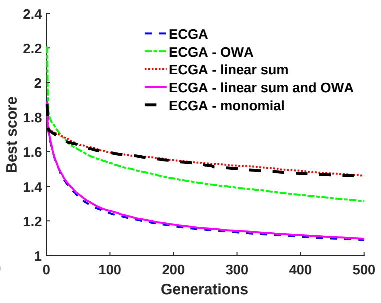
(c) e_3 for three inputs



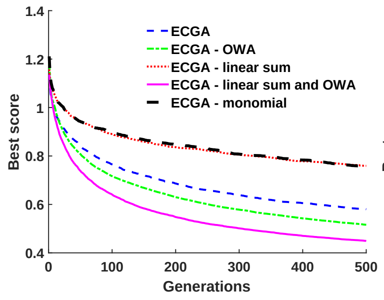
(d) e_4 for three inputs



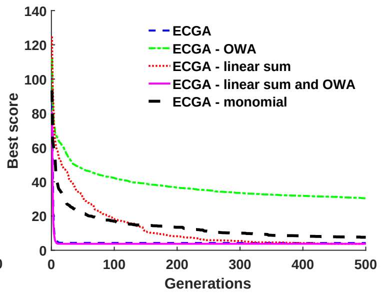
(e) e_1 for six inputs



(f) e_2 for six inputs



(g) e_3 for six inputs



(h) e_4 for six inputs

Fig 5: Experiment 4. Performance comparison of linear sum, OWA and monomial operators in ECGA crossover.

for linear sum and OWA are very close (if not identical) for all four functions. Based on the experimental findings, and guided by intuition, we are led to believe that individually none of these operators are universally best. The takeaway from this experiment is that the selection of appropriate operators is important and adding more operators may have diminishing results with increasing computational complexity.

6 Conclusion

Herein, we proposed new operators for an efficient evolutionary algorithm. Specifically, these operators are focused on learning the Choquet integral. As such, the underlying problem that we want to solve is minimization of some function error relative to a large number of inequality constraints (due to the monotonicity and boundary conditions of the fuzzy measure). To this end, we introduced new theory and algorithms with a solid geometric interpretation. Our method more-or-less allows us to "bypass" our constraints and run an evolutionary algorithm in a smooth and uninterrupted fashion. The method is applicable to a number of problems, but where it excels is in cases of larger numbers of inputs and on complicated (non-convex) error functions. Experiments demonstrated superiority in accuracy and computational performance relative to prior work on different ChI and a benchmark non-ChI function (the Rastrigin function).

In future work, we will extend our theory (operators) beyond the scope of optimizing the ChI. Whereas we demonstrated its operation on the Rastrigin function, we will go further and study the more abstract problem of efficient solutions to complex functions with large numbers of equality and inequality constraints. Also, we showed in[26] that most data-driven problems do not involve all variables in the fuzzy measure. As such, we will follow our ideas in[26] and look for a way to "compress" and further improve the efficiency of our evolutionary optimization in a data set/variable specific fashion. Last, many problems involve non-real-valued (e.g., intervals, type-1 sets, type-2 sets, etc.) integrands and/or measures. In the future we will explore the extension of our operators for uncertain inputs and measures.

Acknowledgments

Elmore and Petry would like to thank the Naval Research Laboratorys Base Program, Program Element No. 0602435N for sponsoring this research.

References

1. Tan, C. and Chen, X., "Induced choquet ordered averaging operator and its application to group decision making," *International Journal of Intelligent Systems* **25**(1), 59–82 (2010).
2. Tan, C. and Chen, X., "Induced intuitionistic fuzzy choquet integral operator for multicriteria decision making," *International Journal of Intelligent Systems* **26**(7), 659–686 (2011).
3. Kojadinovic, I., "Unsupervized aggregation of commensurate correlated attributes by means of the choquet integral and entropy functionals," *International Journal of Intelligent Systems* **23**(2), 128–154 (2008).
4. Torra, V. and Narukawa, Y., "On the meta-knowledge choquet integral and related models," *International journal of intelligent systems* **20**(10), 1017–1036 (2005).
5. Kroupa, T., "Application of the choquet integral to measures of information in possibility theory," *International journal of intelligent systems* **21**(3), 349–359 (2006).

6. Keller, J. M. and Osborn, J., "Training the fuzzy integral," *International Journal of Approximate Reasoning* **15**(1), 1–24 (1996).

7. Grabisch, M., Nguyen, H. T., and Walker, E. A., *Fundamentals of uncertainty calculi with applications to fuzzy inference*, vol. 30, Springer Science & Business Media (2013).

8. Beliakov, G., James, S., and Li, G., "Learning choquet-integral-based metrics for semisupervised clustering," *IEEE Transactions on Fuzzy Systems* **19**(3), 562–574 (2011).

9. Beliakov, G., "Construction of aggregation functions from data using linear programming," *Fuzzy Sets and Systems* **160**(1), 65–75 (2009).

10. Anderson, D. T., Price, S. R., and Havens, T. C., "Regularization-based learning of the choquet integral," in *Fuzzy Systems (FUZZ-IEEE), 2014 IEEE International Conference on*, 2519–2526, IEEE (2014).

11. Holland, J. H., "Adaptation in natural and artificial systems. an introductory analysis with application to biology, control, and artificial intelligence," *Ann Arbor, MI: University of Michigan Press* , 439–444 (1975).

12. Buckles, B. and Petry, F., "Genetic algorithms ieee computer society press," (1992).

13. Homaifar, A., Qi, C. X., and Lai, S. H., "Constrained optimization via genetic algorithms," *Simulation* **62**(4), 242–253 (1994).

14. Michalewicz, Z., "A survey of constraint handling techniques in evolutionary computation methods.," *Evolutionary Programming* **4**, 135–155 (1995).

15. Yeniay, Ö., "Penalty function methods for constrained optimization with genetic algorithms," *Mathematical and computational Applications* **10**(1), 45–56 (2005).

16. Joines, J. A. and Houck, C. R., "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, 579–584, IEEE (1994).

17. Deb, K., "An efficient constraint handling method for genetic algorithms," *Computer methods in applied mechanics and engineering* **186**(2-4), 311–338 (2000).

18. Coello, C. A. C., "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer methods in applied mechanics and engineering* **191**(11-12), 1245–1287 (2002).

19. Jin, X. and Reynolds, R. G., "Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problems: a cultural algorithm approach," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, **3**, IEEE (1999).

20. Du, X., Zare, A., Keller, J. M., and Anderson, D. T., "Multiple instance choquet integral for classifier fusion," in *Evolutionary Computation (CEC), 2016 IEEE Congress on*, 1054–1061, IEEE (2016).

21. Al Boni, M., Anderson, D. T., and King, R. L., "Constraints preserving genetic algorithm for learning fuzzy measures with an application to ontology matching," in *Advance Trends in Soft Computing*, 93–103, Springer (2014).

22. Anderson, D. T., Keller, J. M., and Havens, T. C., "Learning fuzzy-valued fuzzy measures for the fuzzy-valued sugeno fuzzy integral," in *Computational Intelligence for Knowledge-Based Systems Design*, 502–511, Springer (2010).

23. Magoč, T. and Modave, F., "Optimization of the choquet integral using genetic algorithm," in *Constraint Programming and Decision Making*, 97–109, Springer (2014).

24. Trotter, W. T., *Combinatorics and partially ordered sets: Dimension theory*, vol. 6, JHU Press (2001).

25. Avis, D. and Fukuda, K., "A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra," *Discrete & Computational Geometry* **8**(3), 295–313 (1992).

26. Islam, M., Anderson, D. T., Pinar, A. J., and Havens, T. C., "Data-driven compression and efficient learning of the choquet integral," *IEEE Transactions on Fuzzy Systems* **PP**(99), 1–1 (2017).