Extended Multi-Objective fast messy Genetic Algorithm Solving Deception Problems

Richard O. Day and Gary B. Lamont

*Air Force Institute of Technology,
Dept of Electrical and Computer Engineering
Graduate School of Engineering & Management,
Wright-Patterson AFB (Dayton) OH, 45433, USA
Email: {Richard.Day,Gary.Lamont}@afit.edu

Abstract. Deception problems are among the hardest problems to solve using ordinary genetic algorithms. Designed to simulate a high degree of epistasis, these deception problems imitate extremely difficult real world problems. [1]. Studies show that Bayesian optimization and explicit building block manipulation algorithms, like the fast messy genetic algorithm (fmGA), can help in solving these problems. This paper compares the results acquired from an *extended* multiobjective fast messy genetic algorithm (MOMGA-IIa), ordinary multiobjective fast messy genetic algorithm (MOMGA-II), multiobjective Bayesian optimization algorithm (mBOA), and the non-dominated sorting genetic algorithm-II (NSGA-II) when applied to three different deception problems. The *extended* MOMGA-II is enhanced with a new technique exploiting the fmGA's basis function to improve partitioned searching in both the genotype and phenotype domain. The three deceptive problems studied are: interleaved minimal deceptive problem, interleaved 5-bit trap function, and interleaved 6-bit bipolar function. The *unmodified* MOMGA-II, by design, explicitly learns building block linkages, a requirement if an algorithm is to solve these hard deception problems. Results using the MOMGA-IIa are excellent when compared to the non-explicit building block algorithm results of both the mBOA and NSGA-II.

1 Introduction

Algorithms that solve problems by realizing good building blocks (BBs) are useful in solving extremely difficult problems: Protein Structure Prediction [2], 0/1 Modified Knapsack [3], Multiple Objective Quadratic Assignment Problem [4, 5], Digital Amplitude-Phase Keying Signal Sets with M-ary Alphabets and many academic problems [6, 7]. MOMGA-IIa originated as a single objective messy GA (mGA). It evolved from being a single objective mGA into a multiobjective mGA called the MOMGA [8]. Many different MultiObjective Evolutionary Algorithms (MOEAs) were produced during this time period; however,

* The views expressed in this article are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

the MOMGA is the only MOEA explicitly using good BBs to solve problems – even the Bayesian optimization algorithm (BOA) uses a probabilistic model to find good building blocks. The MOMGA has a population size limitation: as the BB size increases so does the population size during the Partially Enumerative Initialization (PEI) phase. This renders the MOMGA less useful on large problems. To overcome this problem, the MOMGA-II, based on the single objective fmGA, is designed. The fmGA is similar to the mGA in that it specifically uses BBs to find solutions; however, it has a reasonable population size and lower run time complexity (See Table 1) when compared to the mGA. MOMGA-II includes many different repair, selection, and crowding mechanisms. Unfortunately, the MOMGA-II is found to be limited when solving large deception problems [6]. This called for the development of basis function diversity measures in the MOMGA-IIa which are designed for smart BB searching in both the geno- and pheno-type domains. Also discussed in this investigation is the mBOA and the NSGA-II [7] neither of which compares well to MOMGA-IIa results.

Table 1. Complexity Estimates for serial GAs

	Single Objective Algorithm				Multiple Objective Algorithm		
Phase	*sGA*[a]	*ssGA*[b]	*mGA*	*fmGA*	*NSGA-II*	*mo-BOA*	*MOMGA-IIa*
Initialization	$O(l^n)$	$O(l^n)$	$O(l^k)$	$O(l)$			
Recombination	$O(gnq)$	$O(g)$					
Primordial	$O(\emptyset)$		$O(\emptyset)$	$O(l^2)$[c]			
Juxtapositional			$O(l \log l)$	$O(l \log l)$			
Overall	$O(l^n)$	$O(l^n)$	$O(l^k)$	$O(l^2)$	$O(mn^3)$[d]	$O(n^{3.5})$[e]	$O(megn^2)$[f]

[a] l is the length of chromosome, n is the size of population, q is group size for tournament selection, g is the number of generations.
[b] l is the length of chromosome, n is the size of population, g is the number of generations of reproduction.
[c] Building Block Filtering
[d] m is the number of objectives
[e] This complexity is problem specific and in this case has been taken from the spin glass problem.[9]
[f] e = number of eras, g = max number of generations

The next section discusses in detail the MOMGA-II and MOMGA-IIa algorithm domains. In addition, a short description of the mBOA and the NSGA-II is provided. The mBOA and NSGA-II have been used to solve these three multiobjective problems (MOPs) in previous research [6, 10]. The three MOPs are then described in detail in Section 3. Next, experimental design, resources, parameter settings, and algorithm efficiency are discussed briefly in Section 4. Finally, in the results section, the mBOA, NSGA-II, MOMGA-II and MOMGA-IIa results are compared and analyzed.

2 Extended Multiobjective fast messy GA (MOMGA-II)

The MOMGA-II(a)[1] is a multiobjective version of the fmGA that has the ability to achieve a near-partitioned search in both the genotype and phenotype domains during execution. It is an algorithm that exploits "good" building blocks (BBs) in solving optimization problems. These BBs represent "good" information in the form of partial strings that can be combined to obtain even better solutions. The BB approach is used in the fmGA to increase the number of "good" BBs that are present in each subsequent generation of the algorithm. The fmGA algorithm executes in three phases: Initialization, Building Block Filtering, and Juxtapositional Phase. See Figure 1 for diagram of the program flow for MOMGA-IIa.

The algorithm begins with the Probabilistically Complete Initialization Phase. This phase randomly generates a user specified number of population members. These population members are of the *a priori* specified chromosome length and each is evaluated to determine its respective fitness values. Our implementation utilizes a binary scheme in which each bit is represented with either a 0 or 1.

The fitness functions used to calculate each string's merit are described in Section 3. In MOMGA-II each string has m fitness values, while in MOMGA-IIa each string has $f = (c * m + i + o) * m$ fitness values associated with it – corresponding to the m objective functions to optimize, c competitive templates, i inverse templates (equal to $c * m$), and o orthogonal templates.

The Building Block Filtering (BBF) Phase follows by randomly deleting locus points and their corresponding allele values in each of the population member's chromosomes. This process completes once the length of the population member's chromosomes have been reduced to a predetermined BB size. In order to evaluate these population members a competitive template (CT) is utilized to fill in the missing allele values. The competitive template is a fully specified chromosome and evolves, by allowing the best member found to replace the old competitive template(s), after each BB generation.

In the MOMGA-II, the next competitive template(s) is(are) randomly chosen from among the non-dominated points within the population. This is the difference between the MOMGA-II and MOMGA-IIa.

Where the MOMGA-II's competitive template selection mechanism selects the next competitive template randomly from the non-dominated points without regard to pareto front (PF) point placement, the MOMGA-IIa competitive template generation, replacement, and evolution is engaged in making sure competitive template selection partitions the search space in both the phenotype and genotype domain.

The innovative balance is achieved through the use of two mechanisms: Orthogonal competitive template generation and Target Vector (TV) guidance. Orthogonal competitive template generation is directed to partition the geno-

[1] When the reader sees MOMGA-II(a) in this report, the sentence is referring to both the MOMGA-II and the MOMGA-IIa.

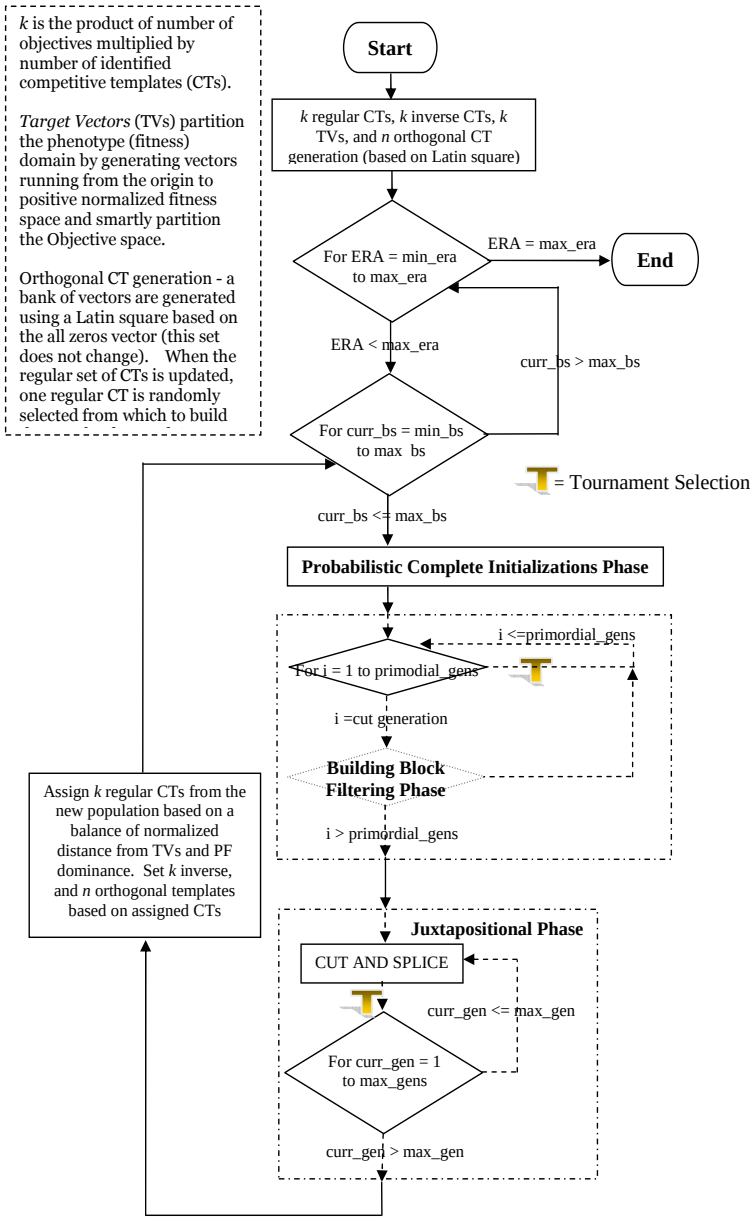


Fig. 1. Illustrated in this figure is the program flow of the MOMGA-IIa. Note the placement of each phase and where tournament selection is performed. Additionally, the MOMGA-IIa exploits and partitions in both the phenotype and genotype domains by updating and generating regular, inverse, and orthogonal competitive templates. See Section 2 for a detailed description of the algorithm.

type space while keeping a good partitioning of the phenotype space using TV guidance.

Through the BBF phase the length of the chromosome decreases, yet each chromosome continues to be evaluated for selection. During this phase these chromosomes are referred to as "underspecified" since each locus position does not have an associated allele value. To evaluate an underspecified population member, the member is overlayed upon the competitive template to fully specify the member. This process uses the allele values from the template to fill in any missing allele values in the population member to allow the fitness evaluation to take place and is repeated any time an underspecified population member needs to be evaluated. The BBF process is alternated with a selection mechanism to keep only the strings with the "best" BBs found, or those with the best number of fitness values. In the case of a tie, where two strings each have an equal number of better fitness values (ie. each have $\frac{m}{2}$ "best" fitness values), the string is randomly selected between the two.

The MOMGA-IIa has a more complex selection mechanism than MOMGA-II because it maintains more fitness values per solution; however, ultimately it is the same comparing only each string's best objective fitness value for all templates considered.

The juxtapositional phase follows and uses the BBs found through the BBF phase and recombination operators to create chromosomes that are fully specified. A chromosome is referred to as fully specified if it is not missing any locus positions, or in other words does not need to use the competitive template for evaluation.

Furthermore, the algorithms have an outer and inner loop and must completely iterate through each BB size a number of epochs before it terminates. For more information on the fmGA and BB theory, see [3, 11].

2.1 Non-dominated Sorting Algorithm-II

The non-dominated sorting algorithm-II (NSGA-II) is a generic non-explicit BB MOEA applied to multiobjective problems (MOPs) – based on the original design of NSGA. It builds a population of compete individuals, ranks and sorts each individual according to non-domination level, applies Evolutionary Operations (EVOPs) to create new pool of offspring, and then combines the parents and offspring before partitioning the new combined pool into fronts. The NSGA-II then conducts niching by adding a crowding distance to each member. It uses this crowding distance in its selection operator to keep a diverse front by making sure each member stays a crowding distance apart. This keeps the population diverse and helps the algorithm to explore the fitness landscape.

2.2 Multiobjective Bayesian Optimization Algorithm

The multiobjective Bayesian optimization algorithm (mBOA) was also used to solve these MOPs in previous research [12]. mBOA is identical to the single objective Bayesian Optimization Algorithm (BOA) [12] minus the selection proce-

dure. The mBOA's selection procedure is replaced by the non-dominated sorting and selection mechanism of NSGA-II. The BOA generates a child population of size n from a parent population. The child and parent population is then merged and the combined population is pareto ranked. Based on the pareto ranking and crowding distance function a new population is created from which BOA builds a new probabilistic model to generate children again.

3 Deception Problems

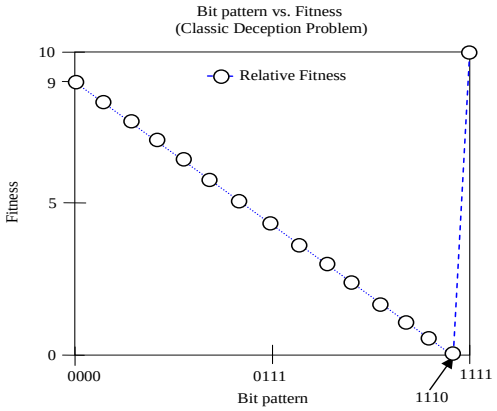


Fig. 2. This figure illustrates a classical deception problem.

In 1987, Goldberg's research group introduced deception to test the abilities of current genetic algorithms [13]. They designed problems having specific difficulties which genetic algorithms (GAs) might face in problem solving. These *deception* problems are often challenging to optimize and involve some degree of deception – resulting in conflicting objectives (e.g. the k-arm bandit competitions between hyperplanes [14]). Later, Whitley [14] proved deceptive attractors must have complementary bit patterns to the global optimum pattern in order to be either fully deceptive or consistently deceptive problems. He then defines a deceptive problem at least one more relevant lower order hyperplane competitor that guides a genetic search away from the global winner. Imagine a hill climbing search algorithm starting anywhere except with the bit configuration *111x*. The hill climbing algorithm always finds the suboptimal fitness of 9 as a solution. This example illustrates how a competitor hyperplane might guide a GA away from the optimal solution. Furthermore, it is every GA engineer's desire to build an algorithm that finds proper linkages within a problem, overcoming this type of deception.

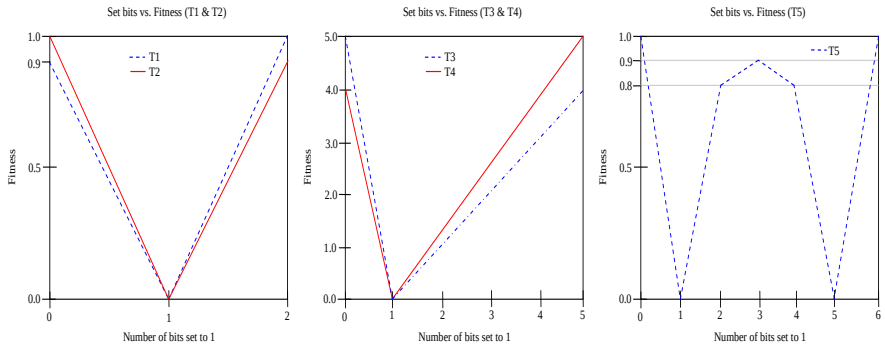
We evaluate the following five test functions in this investigation:

1. T1 - Interleaved minimal deception problem
2. T2 - Complement of T1
3. T3 - Interleaved 5-bit trap function
4. T4 - Complement of T3
5. T5 - Interleaved 6-bit bipolar deception function

These test functions are claimed to be difficult in four respects: deception, loose linkage, mutlimodality, and combinatorially - having a large search space [15–17]. Sections 3.1 through 3.3 included a detailed discussion of these deceptive problems.

In addition to solving these five test functions, difficulty is added by combining these functions together to make three multiobjective problems. By aggregating these test functions together, the order of deception is increased because the functions are paired in a manner that adds a relevant lower order hyperplane competitor to guide a genetic search away from the global winner. The following is the list of the MOPs investigated in this paper:

1. MOP 1: T1 and T2
2. MOP 2: T3 and T4
3. MOP 3: T2 and T5



a) T1 and T2 b) T3 and T4 c) T5 and T2

Fig. 3. These figures illustrate the fitness landscape for each function. Subfigure *a* illustrates deception problems T1 and T2, subfigure *b* illustrates deception problems T3 and T4, and subfigure *c* illustrates deception problems T5 and T2

3.1 Interleaved Minimal Deceptive Problem (T1 & T2)

The interleaved minimal deceptive problems are designed to test an algorithm's ability to discover loosely linked bits by dividing the string into two halves and coupling one bit from each half. Figure 4 illustrates how the bits are correlated. Bits having the same pattern are rewarded, while alternating couplets are not. Additionally, Figure 3.a illustrates the bit couplet fitness for T1 and T2.

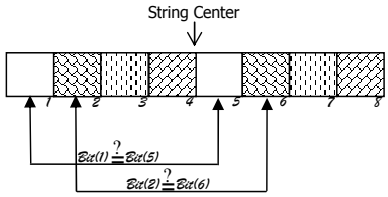


Fig. 4. This figure illustrates bit linkage in an eight bit solution. Notice that every i^{th} and $(\frac{l}{2}+i)^{th}$ bit is linked such that $i \leq \frac{l}{2}$, l=length of string, and the 1^{st} bit is bit 1.

3.2 Interleaved 5-bit trap function (T3 & T4)

The interleaved 5-bit trap function is devised to test an algorithm's ability to find loose linkages having non-consecutive bits. Bits in problems T3 and T4 both have correlated bits with a distance of $\frac{l}{5}$ from one another. Figure 5 illustrates how the bits in groups of 5 are coupled. Additionally, Figure 3.b graphically illustrates how the fitness behavior varies according to the number of bits that are set in the described 5-bit linkage pattern. Notice that T3 in Figure 3.b is similar to the classical deception problem example illustrated in Figure 2.

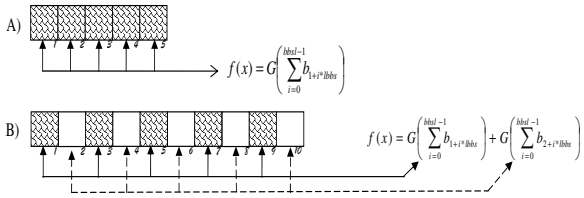


Fig. 5. This figure illustrates bit linkage in an 5 and 10 bit solution. In the figure, the fitness function for each set of bits is shown to the right with arrows indicating which bits contribute to each term of the fitness summation.

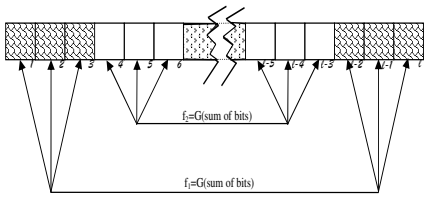


Fig. 6. This figure illustrates 6-bit bipolar linkage in an l bit solution. The bit string is broken in the middle to enhance the idea that string can be of any size as long it is a multiple of 6.

3.3 Interleaved 6-bit Bipolar Function (T5)

The interleaved 6-bit Bipolar function is constructed as a loose linkage problem having correlated bits in variable placement in the string. The first three bits and the last three bits are correlated, then the 4^{th}, 5^{th}, 6^{th}, $(l-4)^{th}$, $(l-5)^{th}$, and $(l-6)^{th}$ and so on until the middle 6 bits of the string are left. Graphically, the fitness function for T5 is illustrated in Figure 3.c.

4 Experiments

The experiment for all MOMGA-II MOPs were run simultaneously on the two computational clusters (ASPEN and Polywells) listed in Table 2. The MOMGA-IIa ran in serial on one computational cluster (TAHOE). The MOMGA-II is given 30 to 50 experiments to solve the three MOPs while the MOMGA-IIa is run for 10 experiments or less. Statistically, 10 runs are required to be able to compare MOEAs mathematically; however, the MOMGA-II is given more experiments in attempt to allow it to find all pareto front solutions for the larger deception problems. Unfortunately, even with the extra experiments, the MOMGA-II still did not find all pareto front solutions. See Section 5 for results.

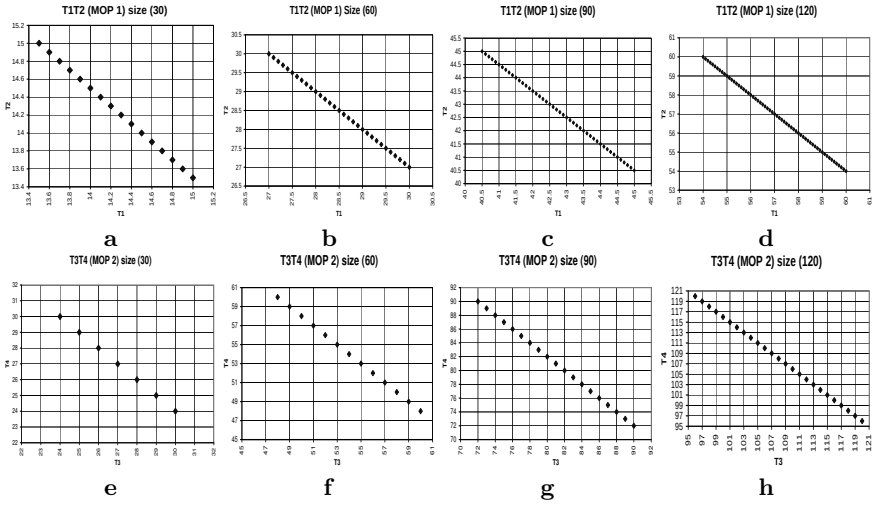


Fig. 7. This figure illustrates the pareto front findings for the T1 vs. T2 and T3 vs. T4 experiment using a string length of 30, 60, 90, and 120 bits.

4.1 Resources

Table 2 lists the resources used for these experiments. Each MOMGA-II experiment took approximately 1 week to complete including the time to process all

data for presentation. Each MOMGA-IIa experiment took approximately 2 days. This includes the time jobs sat idle in the scheduler queue and the post mortem collection of data for analysis. NSGA-II and mBOA run times were not recorded in this investigation and are unknown [10].

Table 2. System Configuration

Cluster 1 (TAHOE)	Cluster 2 (ASPEN)	Cluster 3 (Polywells)
Fedora Core 2	Redhat Linux 9.0	Redhat Linux 7.3
Dual Opteron 2.2 ghz	Ath XP 3000+ 2.1ghz	Ath XP 2800+ 2.0ghz
Cache(L1 I 64,D 64/L2 1024)KB	(64,64/512)KB	(64,64/512)KB
Gb Ethernet	Fast Ethernet	Gb Ethernet
RAM 4 GByte	1 GByte	1 GByte
Crossbar Switch	Crossbar Switch	Crossbar Switch
RAID 5	RAID 5	RAID 5
48 node,2 CPUS/node	48 node,2 CPUS/node	16 node,1 CPU/node

4.2 Parameter Settings

The MOMGA-II(a) has many parameters for proper program execution. BB sizes must be determined, elitism percentages, cut probability, splice probability, mutate probability, population sizing variable (n_a), era generations, and BB filtering schedule. The program is run x times and pareto front members are collected from each solution set. Table 4 indicates the number of times the program is run on each MOP. In some cases the experiments were terminated early if all pareto front solutions were found. All MOMGA-II(a) experiments are executed using a multiple objective fmGA. Tables 3 and 4 list all the parameters used for the experiments conducted in this study. n_a values for the MOMGA-II can be found in [6] while n_a values were set to 500 for the MOMGA-IIa. Table 3 lists the constant parameter settings while Table 4 identifies parameters that were varied for each MOP. In the cases where the programs were run for less than 30 times, this is because the MOMGA-II(a) found the optimal pareto front before each run completed.

Table 3. Summary of Static parameters set for each experiment regardless of MOP and problem size. MOMGA-II(MOMGA-IIa)

Static parameter settings for each MOP			
Parameter	Setting	Parameter	Setting
Maximization	1(n/a)	Thresholding	0(n/a)
mGA(0)/fmGA(1)	1(n/a)	Shuffle Number	2(2)
		Tiebreaking	0(n/a)
Overflow	2.0(2.0)	Reduced initial pop	0(n/a)
Elitism %	25(0)	Extra pop members	0(n/a)
Prob cut	0.02(0.02)	Stop criteria factor	1.00(n/a)
Prob splice	1.0(1.0)	Partition file	0(n/a)
Prob mutation		Plotting file	0(n/a)
allelic	0.0(n/a)	Pop record file	0(n/a)
genic	0.0(n/a)	Copies	5 1 1(n/a)
Inverse Template	n/a(Y)	CT Guesses	n/a(1)

BB size selection for these problems is tricky because the identification of the length of one particular linkage (say 5 bits long) may not be enough to transform

solutions out of the search basis provided by the resident competitive templates. This is prominent for MOMGA-II when solving the MOP 2 of size 90 where the competitive templates used constrict the search into a space where the small BB sizes cannot overcome the competitive template basis. Normally, a BB size can be selected upon knowing the length of these linkages; however, in MOP 2 there are multiple linkages of five bits long each magnifying the difficulty and requiring a larger BB size to allow the MOMGA-II to find more PF_{true} points. It should be noted here that when the BB sizes increase, population size is also increased as well as run time for the algorithm to complete. MOMGA-II's limitation was found in MOP 2 with a size greater than or equal to 90. This limitation is overcome by MOMGA-IIa by using specially chosen competitive templates for search after each BB generation.

Table 4. Summary of Era parameters settings for each experiment using MOMGA-II(MOMGA-IIa)

Experiment	Start ERA	End ERA	Runs	CTs	Inverse CTs	Orthogonal CTs
MOP 1(30)	1	10(4)	30(3)	4(18)	0(18)	0(41)
MOP 1(60)	1	10(4)	30(9)	4(18)	0(18)	0(45)
MOP 1(90)	1	10(4)	30(10)	4(18)	0(14)	0(9)
MOP 1(120)	1	10(4)	30(10)	4(18)	0(14)	0(9)
MOP 2(30)	1	10(4)	30(10)	4(18)	0(18)	0(41)
MOP 2(60)	1	8(4)	50(10)	4(18)	0(18)	0(45)
MOP 2(90)	1	6(4)	50(10)	4(18)	0(18)	0(10)
MOP 2(120)	1	4(4)	30(8)	4(14)	0(14)	0(49)
MOP 3(30)	1	10(4)	10(1)	4(14)	0(14)	0(41)
MOP 3(60)	1	12(4)	10(1)	4(14)	0(14)	0(41)

4.3 Efficiency Finding Pareto-front points

Reduction of relative execution time for MOMGA-IIa is achieved by keeping the pareto front points (including duplicates) in memory. The MOMGA-IIa benefits from a creatively designed structure maintained as a dynamic linked list object which holds all pareto front members. Dominated solutions are deleted from the structure and from memory including all duplicates for that particular point. As the program runs, the pareto front point listings can become long. This is a disadvantage; however, elitism selection is $\mathrm{O}(p)$ run time as all solutions, p, are readily listed. In addition to this enhancement, epsilon-difference dominance, crowding techniques and dominance linkage can be instantiated. The MOMGA-IIa also has the ability to trace evolutionary solutions throughout the search process. This enhanced feature comes at a cost in space (memory or disk) and efficiency. All experiments in this paper are run with an active trace feature to allow for post mortem analysis of pareto front point conception.

5 Results

The MOMGA-IIa results are superior. In every case the MOMGA-IIa has either found more pareto front solutions or more unique solutions than all other algorithms tested on these three MOPs. Along with the enhanced algorithm, deception problems of larger sizes are added into this investigation to allow for even

Table 5. Summary of Results for all experiments. Included in this table are the number of optimal pareto front points, number of unique strings making up these Optimal points, and the number of points each algorithm (MOMGA-IIa, MOMGA-II, mBOA, and NSGA-II) have found in each category.

MOP 1								
T1 vs. T2 - Sizes: (30/60/90/120) Unique Strings: $\{(2^{15}/2^{30}/2^{45}/2^{60})\}$ Pareto Front Strings: (16/31/46/61)								
Algorithm	Unique Strings Found				Pareto Front pts Found			
	30	*60*	*90*	*120*	*30*	*60*	*90*	*120*
MOMGA-IIa	**32768**	**300776**	**57661**	**32876**	**16**	**31**	**46**	**61**
MOMGA-II	596	21364	28	138	**16**	**31**	16	56
mBOA	224		591		**16**		**46**	
NSGA-II	7		5		6		3	
MOP 2								
T3 vs. T4 - Sizes: (30/60/90/120) Unique strings: $\{(2^{6}/2^{12}/2^{18}/2^{24})\}$ Pareto Front Strings: (7/13/19/25)								
Algorithm	Unique Strings Found				Pareto Front pts Found			
	30	*60*	*90*	*120*	*30*	*60*	*90*	*120*
MOMGA-IIa	**64**	**565**	**1280**	**3594**	**7**	**13**	**19**	**25**
MOMGA-II	32	98	54	31	**7**	12	6	14
mBOA	30	102	327		**7**	**13**	**19**	
NSGA-II	0	1	0		0	1	0	
MOP 3								
T5 vs. T2 - Sizes: (30/90) Unique strings: (1/1) Pareto Front Strings: (1/1)								
Algorithm	Unique Strings Found				Pareto Front pts Found			
	30		*90*		*30*		*90*	
MOMGA-IIa	**1**		**1**		**1**		**1**	
MOMGA-II	**1**		**1**		**1**		**1**	
mBOA	**1**		**1**		**1**		**1**	
NSGA-II	0		0		0		0	

more difficult problem resolution. As expected from the last investigation [6], this investigation found MOP 2 to be the most difficult to solve. Difficulty comes in the form of time to find all pareto front members.

The following sections describe, in detail, the experiment results for each MOP. Notice that the pareto front for MOP 1 and MOP 2 is linear. This is due to the linear slopes of the individual objective functions making up each of these MOPs. It is also worth noting that each point on the MOP 1 and MOP 2 pareto front may have many unique strings (solutions) equating to that particular pareto front point. This phenomena is illustrated in Table 5 where both the number of pareto front points and unique strings are listed.

5.1 T1 vs. T2 (MOP 1)

Figures 7a∼d reflect the results of the MOP 1 of sizes 30, 60, 90, and 120. Diamonds indicate the pareto front solutions found. Notice that these points are linear and discrete. For problem sizes 30, 60, 90, and 120 there are 16, 31, 46, and 61 total pareto front points. Not only does MOMGA-IIa find the same or more true pareto front points for MOP 1 than every other algorithm tested, it also finds more unique strings making up each of these pareto front points. Table 5 numerically shows in bold that MOMGA-IIa is the front runner in this experiment. In the problem size 30, the MOMGA-IIa has actually found all unique strings corresponding to each of the 16 pareto front solutions.

5.2 T3 vs. T4 (MOP 2)

Figures 7e∼h illustrate the results of the MOP 2 experiment. Similarly, diamonds indicate the pareto front solutions found. Notice that these points are linear and discrete. For the problem sizes 30, 60, 90, and 120 there are 7, 13, 19, and 25 total pareto front points in the entire search space. Not only does MOMGA-IIa find the same or more true pareto front points for MOP 1 than every other algorithm tested, it also finds more unique strings making up these pareto front points. Specifically, MOMGA-IIa overcomes the problem size issue and finds more pareto front points than its predecessor, MOMGA-II. Table 5 numerically shows in bold that MOMGA-IIa is the front runner in this experiment. In addition to finding all the pareto front solutions for MOP 2 size 30, the MOMGA-IIa finds all corresponding unique strings.

5.3 T5 vs. T2 (MOP 3)

No figures in this paper reflects the results of MOP 3 for it is rather uninteresting. See [6] for an example. There is only one point on the pareto front. All algorithms but the NSGA-II found the one and only pareto front point and the only unique string representing this solution.

5.4 Comparison

MOMGA-IIa results are excellent. The MOMGA-IIa finds all pareto front members in every MOP and at every size. Additionally, the MOMGA-IIa also finds more unique strings than any other algorithm tested in this investigation. Clearly, the MOMGA-IIa outperforms the MOMGA-II and is able to scale up to solve larger deception problems by having a pool of better competitive templates.

6 Conclusion

In conclusion, this experiment illustrates an explicit building block genetic algorithm's ability to solve deception problems. MOMGA-IIa's capabilities to explicitly find and use good multiobjective Building Blocks (MOBBs) is illustrated.

MOMGA-IIa can find the loosely linked bits in deceptive problems using its manipulation of BBs and it has been shown to scale when "good" competitive templates are selected. The pedagogical problems solved in this report are limited in testing the capabilities of MOMGA-IIa; however, they do show that the MOMGA-IIa can solve difficult problems with ease. The MOMGA-IIa implicitly solves problems by identifying good MOBBs when iteratively selecting "good" competitive templates that partition both the geno- and pheno-type domains. Important aspects of this algorithm are the BB size, BB schedule and, competitive template numbers (regular, inverse, and orthogonal). An important conjecture learned from this investigation is the fact that as the search space increases it is important to increase the number of competitive templates - thus limiting the largest required BB size and ultimately limiting the population size generated by the MOMGA-IIa. In addition, a second long standing conjecture that finding larger BBs allows for more pareto front points along the extremes of the front to be found [18]. The new innovative tracing technique has identified interesting results concerning this second conjecture.

7 Future Work

Future work includes the results of algorithm runs on larger deception problems found in Table 5. Additionally, a detailed description of design on the MOMGA-IIa is in order along with the correction of the count of unique strings found. Finally, an examination of the evolutionary trace function that is embedded within the MOMGA-IIa is needed to illustrate where solutions are drawn from with respect to operators and BB sizes.

References

1. Colin Reeves and Christine Wright. An experimental design perspective on genetic algorithms. In L. Darrell Whitley and Michael D. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 7–22. Morgan Kaufmann, San Francisco, CA, 1995.
2. Richard O. Day. A multiobjective approach applied to the protein structure prediction problem. Ms thesis, Air Force Institute of Technology, March 2002. Sponsor: AFRL/Material Directorate.
3. Jesse B. Zydallis. *Explicit Building-Block Multiobjective Genetic Algorithms: Theory, Analysis, and Development.* Dissertation, Air Force Institute of Technology, AFIT/ENG, BLDG 642, 2950 HOBSON WAY, WPAFB (Dayton) OH 45433-7765, Feb 2002.
4. Richard O. Day, Mark P. Kleeman, and Gary B. Lamont. Solving the Multiobjective Quadratic Assignment Problem Using a fast messy Genetic Algorithm. In *Congress on Evolutionary Computation (CEC'2003)*, volume 1, pages 2277–2283, Piscataway, New Jersey, December 2003. IEEE Service Center.
5. Mark P. Kleeman. Optimization of heterogeneous uav communications using the multiobjective quadratic assignment problem. Ms thesis, Air Force Institute of Technology, March 2004. Sponsor AFRL.

6. Richard O. Day and Gary B. Lamont. Multi-objective fast messy genetic algorithm solving deception problems. *Congress on Evolutionary Computation; Portland, Oregon*, 4:1502–1509, June 19 - 23 2004.
7. Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002.
8. Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.
9. Martin Pelikan. *Bayesian Optimization Algorithm: From Single Level to Hierarchy*. Thesis, University of Illinois at Urbana-Champaign, 117 Transportation Building, 104 S. Mathews Avenue Urbana, IL 61801, July 2002. IlliGAL Report No. 2002023.
10. Nazan Khan, David E. Goldberg, and Martin Pelikan. Multi-objective byesian optimization algorithm. ILLiGAL Report 2002009, University of Illinois at Urbana-Champaign, 117 Transportation Building, 104 S. Mathews Avenue Urbana, IL 61801, March 2002.
11. David E. Goldberg, Kalyanmoy Deb, Hillol Kargupta, and Georges Harik. Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 56–64, July 1993.
12. Nazan Khan. Bayesian optimization algorithms for multiobjective and heierarchically difficult problems. Master thesis, University of Illinois at Urbana-Champaign, 117 Transportation Building, July 2003.
13. David Goldberg. *Simple Genetic Algofithms and the Minimal, Decptive Problem*, chapter Genetic Algorithms and Simulated Annealing, pages pp: 74–88. Morgan Kaufmann, Pubs, 1987.
14. L. Darrell Whitley. Fundamental principles of deception in genetic search. In Gregory J. Rawlins, editor, *Foundations of genetic algorithms*, pages 221–241. Morgan Kaufmann, San Mateo, CA, 1991.
15. Kalyanmoy Deb, Jeffrey Horn, and David E. Goldberg. Multimodal deceptive functions. Technical Report IlliGAL Report No 92003, University of Illinois, Urbana, 1992.
16. Kalyanmoy Deb and David E. Goldberg. Analyzing deception in trap functions. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 93–108. Morgan Kaufmann, San Mateo, CA, 1993.
17. David E Goldberg, Kalyanmoy Deb, and Jeffrey Horn. Massive multimodality, deception and genetic algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature 2*, Amsterdam, 1992. Elsevier Science Publishers B.V.
18. Jesse B. Zydallis and Gary B. Lamont. Explicit Building-Block Multiobjective Evolutionary Algorithms for NPC Problems. In *Congress on Evolutionary Computation (CEC'2003)*, volume 4, pages 2685–2695, Piscataway, New Jersey, December 2003. IEEE Service Center.