Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multicriteria Decision Making (MCDM 2007)

Playing in the Objective Space: Coupled Approximators for Multi-Objective Optimization

Harold Soh*, Ong Yew Soon†, Mohamed Salahuddin*, Terence Hung* and Lee Bu Sung†
*Institute of High Performance Computing
1 Science Park Road, 01-01 The Capricorn
Singapore Science Park II
Singapore 117528
Email: sohsh@ihpc.a-star.edu.sg, mohdsh@ihpc.a-star.edu.sg,
terence@ihpc.a-star.edu.sg
†School of Computer Engineering
Nanyang Technological University
Blk N4, 2b-39, Nanyang Avenue
Singapore 639798
Email: asysong@ntu.edu.sg, ebslee@ntu.edu.sg

***Abstract*— This paper presents a method of integrating computational intelligence with the operators used in evolutionary algorithms. We investigate approximation models of the objective function and its inverse and propose two simple algorithms that use these *coupled approximators* to optimize multi-objective functions. This method is a break from traditional approach used by standard cross-over and mutation operators, which only explore the objective space through "near-blind" manipulation of solutions in the parameter space. Fundamentally, our proposed *intelligent operators* use learned models of the coupling between the objective space and the parameter space to generate successively better solutions by extrapolating (or interpolating) from known solutions directly in the objective space. We term our implementation of the developed techniques as the Coupled Approximators Evolutionary Algorithm (CAEA). Promising empirical results with the DTLZ test suite prompt us to suggest several avenues for future research including combination with local search methods, incorporation of domain-knowledge and more efficient search algorithms.**

I. Introduction

The Evolutionary Algorithms (EA) arena has shown tremendous activity in the past few years, with advancements being made on both the theoretical and applied fronts. One particular area of interest has been the development of multi-objective evolutionary methods that are capable of optimizing expensive and difficult problems quickly and reliably. Previous work in this area has focussed mainly on using inexpensive models of the exact evaluation function. This work explores an alternative approach by merging computational intelligence methods with evolutionary algorithms to derive *intelligent operators* which generate candidate solutions based on models of the objective function and its inverse. We term the pairing of these models as *coupled approximators* and present two algorithms that use these models to explore the objective space.

This paper is organized as follows: Section II reviews some relevant work on multi-objective optimization with evolutionary algorithms and fitness approximation. Section III presents details of our proposed method, i.e., the Coupled Approximators Evolutionary Algorithm (CAEA). In Section IV, we present results and an analysis of simulations with scalable benchmark problems from the DTLZ [1] test suite. Section V conludes this paper with possible future work and conclusions derived from this study.

II. Background

In this section, we cover related work in the fields of multi-objective optimization, evolutionary algorithms and handling computationally expensive function evaluations.

A. Multi-objective Optimization with Evolutionary Algorithms

Intuitively, the problem of multi-objective optimization can be viewed as a search through a m-dimensional space for all minimum (or maximum) objective vectors that satisfy posed constraints. The search is usually performed indirectly by varying parameter vectors, also called decision vectors, in a d-dimensional space. From this point forward, we assume, without loss in generality, minimization problems.

We begin by defining m-dimensional fitness space of all feasible solutions. Given an objective function $f(\hat{x}) : \mathbb{R}^d \rightarrow \mathbb{R}^m$ and constraint functions $g(\hat{x}) : \mathbb{R}^d \rightarrow \mathbb{R}^k$ and $h(\hat{x}) : \mathbb{R}^d \rightarrow \mathbb{R}^l$, the feasible objective space is defined as the set of all vectors in the function's range that satisfy the given constraints, $F = \{\hat{f}_i \in \mathbb{R}^m | \hat{f}_i = f(\hat{x_i}) \wedge (g(\hat{x_i}) > \mathbf{0}) \wedge (h(\hat{x_i}) = \mathbf{0})\}$.

Since our goal is to find the best solutions in a given F, it is helpful to consider what it means for one solution to be better than or *dominates* another. Consider $\hat{f} = (f_1, f_2, ..., f_m), \hat{g} = (g_1, g_2, ..., g_m) \in F$. $\hat{f}$ is said to dominate $\hat{g}$, denoted as $\hat{f} \succ \hat{g}$ iff $\forall i \in \{1, 2, ..., m\} : f_i \leq g_i \wedge \exists j \in \{1, 2, ..., m\} : f_j < g_j$. For example, in fig. 1, solution A dominates solution C while solutions B and C are non-dominating.

A Pareto optimal vector (or point), denoted as $\hat{f}^*$, is a vector which is not dominated by any other vector in the objective space ,i.e., $\nexists \hat{f}_j \in F : \hat{f}_j \succ \hat{f}^*$. The Pareto optimal set or front,

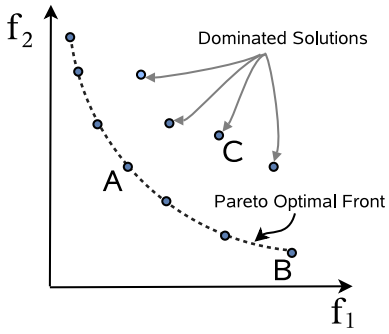


Fig. 1. A sample convex Pareto optimal front for a minimization problem.

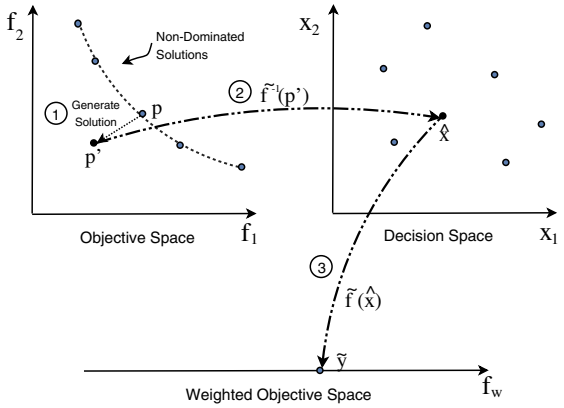


Fig. 2. Optimization using coupled approximators is a three-step process. First a solution is generated by extrapolation or interpolation in the objective space from a non-dominated solution, p. Next, the inverse model is used to obtain the estimated decision vector, $\hat{x}$. This decision vector is then verified by the approximated objective function. If the resultant estimated weighted objective function is less than the weighted objective values of the p, the solution is accepted for evaluation by the true objective function.

defined as the set of all Pareto optimal vectors, is denoted as $F^* = \{\hat{f}_i^* \in F | \nexists \hat{f}_j \in F : \hat{f}_j \succ \hat{f}_i^*\}$. An example of a Pareto-optimal front for a minimization problem is illustrated in fig. 1.

Multi-objective optimization is defined as a search for the Pareto optimal front, F^* and the corresponding $X = \{\hat{x} \in \mathbb{R}^p | f(\hat{x}) \in F^*\}$. Multi-objective optimization problems are abundant in the real-world and include tasks such as scheduling, routing and structural design.

In recent years, effective evolutionary algorithms have been developed to solve multi-objective problems. The more influential of these multi-objective evolutionary algorithms (MOEA) include the Non-Dominating Sorting Genetic Algorithm-II (NSGA-II) [2], the Strength Pareto Evolutionary Algorithm 2 (SPEA2) [3] and the Pareto Archived Evolutionary Strategy (PAES) [4]. Readers wanting a comprehensive survey and history of such methods are directed to [5] and [6].

B. Evolutionary Algorithms for Computationally Expensive Problems

A limitation of multi-objective evolutionary algorithms is that a relatively large number of evaluations are required to converge to the Pareto optimal front. This may prohibit the use of MOEAs when evaluations are expensive, for example simulating an engine to determine thermodynamic properties. Two solutions to resolve this issue has been proposed. The first is to use high performance computing methods to distribute the function evaluations across a cluster or the Grid. An example of this in the domain of materials engineering is the GPEM system [7]. The second method uses cheaper approximation models of the fitness function and uses the models to evaluate solutions instead of the true objective function, for example [8] [9] [10] [11] [12] [13]. For more details on some of the evolutionary computational frameworks that employ approximation models, the reader is referred to the survey papers in [14] and [15].

III. Coupled Approximators for Multi-Objective Optimization

The fundamental idea in this work is to model the objective function, its inverse and apply these models towards locating optimal solutions in the search space. We term these two models used in conjunction as *coupled approximators*. At each iteration of the optimization process, dominating or non-dominating solutions are constructed by extrapolating (or interpolating) from existing solutions in the objective space. The inverse model is used to obtain the estimated decision vector which is then verified by the approximated objective function. This three-step process is illustrated by fig. 2.

The Coupled Approximators Evolutionary Algorithm (CAEA) pseudocode (fig. 3) illustrates a conceptually simple algorithm based on the Non-dominated Sorting Algorithm (NSGA-II) [2]. The algorithm begins as a standard population based evolutionary algorithm by initializing a population of size N via random or latin hypercube sampling. Solutions are then evaluated and sorted according to non-domination ranks.

Next, the algorithm creates γN new individuals. The new operators are used first ,i.e., the forward and inverse approximation models are created and used to generate new solutions. It is possible that these operators fail to return valid solutions and if there is space remaining in the new population buffer, the remaining solutions are generated using standard crossover operators. The new solutions are evaluated and added to the current population. The merged population is then sorted into non-domination ranks and truncated based on rank and crowding distance. The following subsections detail the modeling methods and solution construction algorithms, the GenerateSolutionsCA function (fig. 3, line 6).

A. Generating New Solutions

The GenerateSolutionsCA method (fig. 4) first extracts the non-dominated front, P^* from the population and generates the coupled approximators by modeling fitness function with samples from the current population. As previously described, two models are generated:

Coupled Approximators Evolutionary Algorithm, CAEA

1) P_0 =Initialize(N)
2) Evaluate(P_0)
3) NonDominatedSort(P_0)
4) $t := 0$
5) **while** termination conditions not met **do**
6) S_{CA} := GenerateSolutionsCA(P_t, t):
7) S_{SO} := StandardOperators(P_t, $\gamma N - |S_{CA}|$)
8) $S := S_{CA} \cup S_{SO}$
9) Evaluate(S)
10) $P' = P_t \cup S$
11) NonDominatedSort(P')
12) P_{t+1} := Truncate(P_{t+1}, N)
13) $t := t + 1$
14) **done**

Fig. 3. The Coupled Approximators Evolutionary Algorithm.

GenerateSolutionsCA(P_t , t)

1) $P^* := \{p \in P_t |$ p is not dominated by any $p_i \in P_t\}$
2) $\tilde{f}(\hat{x})$:= CreateForwardModel(P_t).
3) $\tilde{f}^{-1}(\hat{y})$:= CreateInverseModel(P_t).
4) **if** t is even **or** $t = 0$ **then**
5) S_{CA} := GenDominating(P^*, $\tilde{f}(\hat{x})$, $\tilde{f}^{-1}(\hat{y}), \alpha$, ϵ_k)
6) **else**
7) S_{CA} := GenDiverse(P^*, $\tilde{f}(\hat{x})$, $\tilde{f}^{-1}(\hat{y})$, γN, β)
8) **return** S_{CA}

Fig. 4. Main method to generate dominating and non-dominated solutions relative to the current non-dominated front.

1) The *forward model*, $\tilde{f}(\hat{x}) : \mathbb{R}^d \rightarrow \mathbb{R}$, which is an approximation of a *weighted* objective function, $f_w(\hat{x}) = \sum_{i=1}^{m} w_i f_i(\hat{x})$. For simplicity, we consider equal weights in this study, i.e., $w_1 = w_2 = ... = w_m = \frac{1}{m}$. Note that we use the weighted objective function because the forward model is used by the following algorithms only as a means of verifying that the inverse model's prediction is accurate. If the forward model plays a more significant role, it may be useful to use a model of the actual objective function.
2) The *inverse model*, $\tilde{f}^{-1}(\hat{y}) : \mathbb{R}^m \rightarrow \mathbb{R}^d$, which is an approximation of the inverse objective function. In general, the inverse $f^{-1}(\hat{y})$ function may not exist so, the model approximates the inverse relation from the given samples.

Using P^* and the coupled approximators, dominating solutions (even iterations) or diverse, possibly non-dominating, solutions (odd iterations) are generated. The methods by which both types of solutions are constructed are as described in the following subsections.

B. Constructing Dominating Solutions

Recall that a given two solutions $\hat{f} = (f_1, f_2, ..., f_m), \hat{g} = (g_1, g_2, ..., g_m) \in F$, $\hat{f}$ dominates $\hat{g}$, $\hat{f} \succ \hat{g}$, iff $\forall i \in \{1, 2, ..., m\} : f_i \leq g_i \wedge \exists j \in \{1, 2, ..., m\} : f_j < g_j$. Hence, given a current population P, the algorithm (fig. 6) considers each non-dominated solution $\hat{p} \in P^*$ and attempts to generate a dominating solution $\hat{p}'$ by linearly extrapolating along each objective dimension, k.

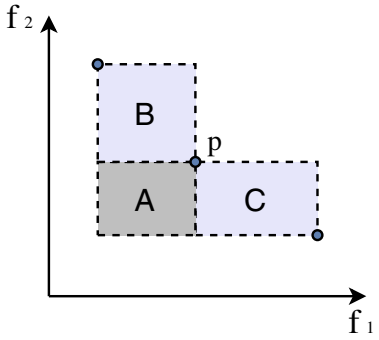


Fig. 5. Generated solution spaces for $\alpha = 1$ given some solution p. GenDominating generates solutions within space A. GenDiverse generates solutions in spaces A, B and C.

The algorithm considers each objective dimension in turn and extends the point by some *step distance*, $d_k := \alpha \cdot (p_k - q_k)$ where $\hat{q}$ is the next lowest value for the objective ,i.e., $\hat{q}$ s.t. $\nexists \hat{r} \in P^* : q_k < r_k < p_k$. The "aggression" parameter, α, determines how far along this objective to extend with $p_k - q_k$ as a unit. In fig. 5, the space A illustrates the neighbourhood region explored when $\alpha = 1$. A high α may cause a quick convergence to the front at the cost of making more errors. A smaller α is likely to yield a slower, more conservative search.

While not explicitly stated in the pseudocode, it is necessary to note that $\hat{q}$ does not exist when p_k is the minimum. If $|P^*| > 1$, we can choose not to extrapolate along any objective that p_k is the minimum. Otherwise, d_k can be set to some predefined value or the difference between p_k and some minimum reference objective value, r_k. If the objective space is in the positive real space $\mathbb{R}^+$, r_k can be defaulted to 0.

The inverse model is then used to estimate the decision variables, $\hat{x}$, that would yield the objective vector $\hat{p}'$. The decision variables are then validated with the forward model. If the estimated weighted objectives, $\tilde{y}$ is smaller than the weighted objective value of $\hat{p}$, the algorithm moves on to the next objective. Otherwise, the step distance is halved and the extension process repeated. The algorithm terminates is d_k falls below some predefined value, ϵ, for this objective. In this study, we found it convenient to set ϵ to be a percentage of the initial d_k value.

C. Constructing Diverse Solutions

This solution construction method (fig. 7) attempts to generate possibly non-dominating solutions to maintain a diverse pareto optimal front. The algorithm attempts to generate $\frac{\gamma N}{|P^*|}$ solutions from each non-dominated solution , $\hat{p} \in P^*$. A new solution p' is constructed by randomly perturbating the solution in the objective space along each objective, k, within the range of the next smaller objective value, l_k, and the

GenDominating(P^*, $\tilde{f}(\hat{x})$, $\tilde{f}^{-1}(\hat{y}), \alpha$, ϵ)

1) **for each** solution $\hat{p} \in P^*$ **do**
2) $\quad \hat{p}' := \hat{p}$
3) $\quad$ **for each** objective $k := 1, 2, ..., m$ **do**
4) $\quad\quad$ Let $\hat{q}$ s.t. $\nexists \hat{r} \in P^* : q_k < r_k < p_k$
5) $\quad\quad d_k := \alpha \cdot (p_k - q_k)$
6) $\quad\quad$ **do**
7) $\quad\quad\quad p'_k := p'_k - d_k$
8) $\quad\quad\quad \hat{x} := \tilde{f}^{-1}(\hat{p}')$
9) $\quad\quad\quad \tilde{y} := \tilde{f}(\hat{x})$
10) $\quad\quad\quad$ **if** $\tilde{y} < f_w(p)$
11) $\quad\quad\quad\quad$ **end while loop**
12) $\quad\quad\quad$ **else**
13) $\quad\quad\quad\quad d_k := d_k \div 2$
14) $\quad\quad\quad\quad p'_k := p_k$
15) $\quad\quad$ **while** $d_k > \epsilon$
16) $\quad$ **done**
17) $\quad S_{CA} := S_{CA} \cup \hat{x}$
18) **done**
19) **return** S_{CA}

Fig. 6. Algorithm to construct dominating solutions given a set of solutions, coupled approximation models, an "aggression" parameter, α and a termination condition, ϵ.

GenDiverse(P^*, $\tilde{f}(\hat{x})$, $\tilde{f}^{-1}(\hat{y})$, γN, β)

1) **for each** solution $\hat{p} \in P^*$ **do**
2) **for** $i := 1$ to $\gamma N / |P^*|$ **do**
3) $\quad\quad b := 0$
4) $\quad\quad$ **do**
5) $\quad\quad\quad \hat{p}' := \hat{p}$
6) $\quad\quad\quad$ **for each** objective $k := 1, 2, ..., m$ **do**
7) $\quad\quad\quad\quad$ Let $\hat{l}$ s.t. $\nexists \hat{r} \in P^* : l_k < r_k < p_k$
8) $\quad\quad\quad\quad$ Let $\hat{u}$ s.t. $\nexists \hat{v} \in P^* : u_k > v_k > p_k$
9) $\quad\quad\quad\quad p'_k =$random$(l_k, u_k)$
10) $\quad\quad\quad$ **done**
11) $\quad\quad\quad \hat{x} := \tilde{f}^{-1}(\hat{p}')$
12) $\quad\quad\quad \tilde{y} := \tilde{f}(\hat{x})$
13) $\quad\quad\quad b := b + 1$
14) $\quad\quad$ **while** $(\tilde{y} > f_w(p)) \wedge (b < \beta)$
15) $\quad\quad S_{CA} = S_{CA} \cup \hat{p}'$
16) $\quad$ **done**
17) **done**
18) **return** S_{CA}

Fig. 7. Algorithm to construct non-dominated solutions given a set of solutions, coupled approximation models, the number of solutions to generate, γN and a termination condition, β.

next larger objective value, u_k. Note if p_k is the minimum value, p_k is perturbated within some predefined range. As when constructing dominating solution, the inverse model is then used to estimate the corresponding decision variables, $\hat{x}$. Solutions are accepted for evaluation only if they are in the dominating or non-dominating space and are validated by the forward model. The spaces A, B and C in fig. 5 illustrate the neighbourhood region in the objective space that the GenDiverse algorithm explores.

IV. Empirical Results

In this section, we present results on simulations of our modeling operators on several real-valued continuous test problems and compare the solutions found to those obtained via standard cross-over and mutation operators only.

A. Implementation and Experimental Setup

CAEA was implemented in C++ and utilized the Fast Artificial Neural Network Library (FANN) [16]. To model the objective function, $f(\hat{x})$ and its inverse, $f^{-1}(\hat{y})$, we used simple radial basis function (RBF) networks with $5 \times p$ internal nodes for the forward model and $5 \times (m + p)$ internal nodes for the inverse model.

We used the PISA [17] system as an experimental environment running on a 16 CPU Intel XEON 2.6Ghz shared memory system with 64 gigabytes of memory. CAEA was tested on the DTLZ2, DTLZ3, DTLZ6, DTLZ7 test problems [1] with 2, 4 and 8 objectives. The population sizes were set to 200, 300 and 400 for 2, 4 and 8 objectives respectively. The number of variables was set to the recommended $m+9$ where m is the number of objectives. Each test was repeated 25 times with matched populations between algorithms. We compared CAEA against an algorithm using standard cross-over and mutation operators, NSGA-II, using the additive epsilon indicator, ϵ_+ and computed the Kruskal-Wallis statistical test [18]. The parameters for each algorithm is given in table I.

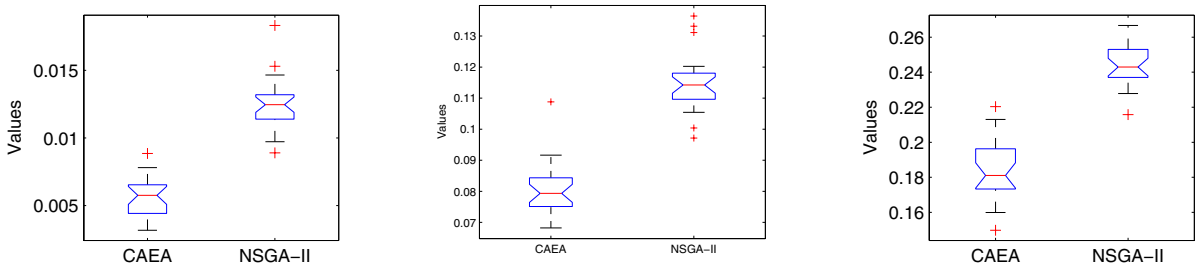
TABLE I
PARAMETERS FOR CAEA AND NSGA-II

Parameter	CAEA	NSGA-II
Crossover Probability, p_c	-	0.9
Crossover Spread Factor, n_c	15	15
Mutation Probability, p_m	0.01	0.01
Mutation Spread Factor, n_m	20	20
Aggression, α	10	-
GenDominating Stopping criteria, ϵ	$0.1 \times \text{initial} d_k$	-
GenDiverse Stopping criteria, β	20	-
Solution multiplier, γ	1	-

B. Results and Analysis

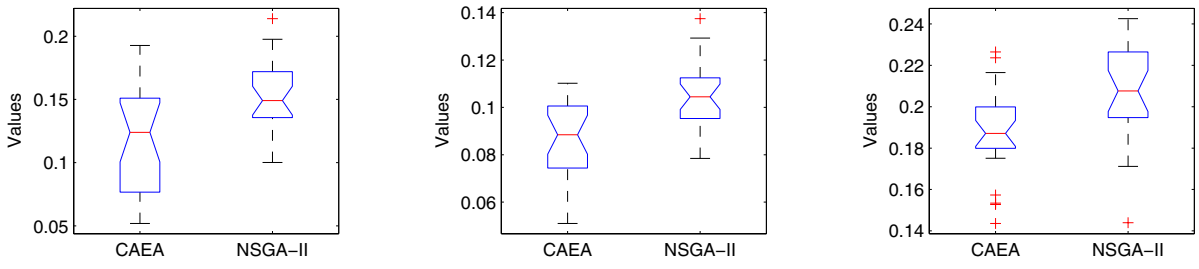
Figures 8 to 15 illustrate box plots of the additive epsilon indicator on the DTLZ test problems and tables II and III give the results of the Kruskal-Wallis statistical tests. We analyze snapshots of the runs at generations 25 and 50. At generation 25, the coupled approximators and construction algorithms generate significantly better solutions than those by standard evolutionary operators on DTLZ2, DTLZ3 and DTLZ7 with 2, 4 and 8 objectives.

The exception is DTLZ6, which is a deceptive problem and may have proved difficult for the RBFs to model accurately in higher dimensions. The models would then have made errorneous predictions and hence, generated poorer solutions. Evidence for this is suggested by fig. 16 and tbl. IV which illustrate the mean absolute errors made during a representative



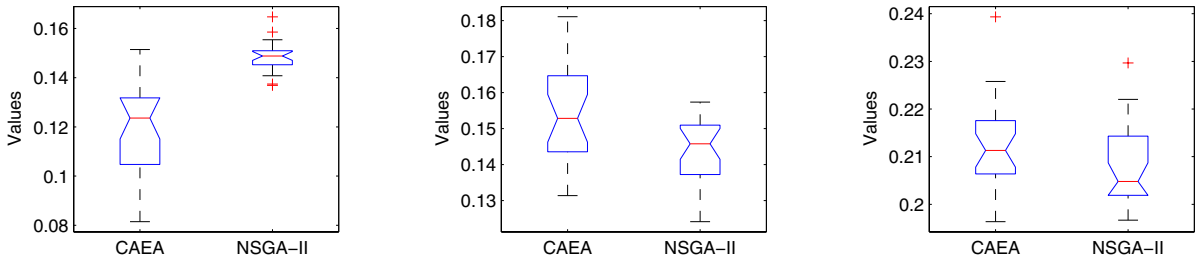
(a) 2 objectives, 11 variables (b) 4 objectives, 13 variables (c) 8 objectives, 17 variables

Fig. 8. ϵ_+ indicator values for CAEA and NSGA-II for DTLZ2 at Generation 25. Visually, CAEA outperforms NSGA-II for all three objective levels. This conclusion is confirmed by the Kruskal-Wallis statistical test.



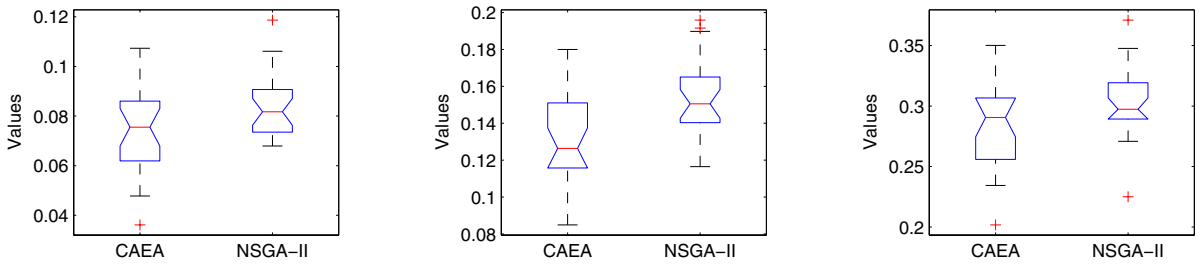
(a) 2 objectives, 11 variables (b) 4 objectives, 13 variables (c) 8 objectives, 17 variables

Fig. 9. ϵ_+ indicator values for CAEA and NSGA-II for DTLZ3 at Generation 25. As with DTLZ2, CAEA outperforms NSGA-II at all three objective values.



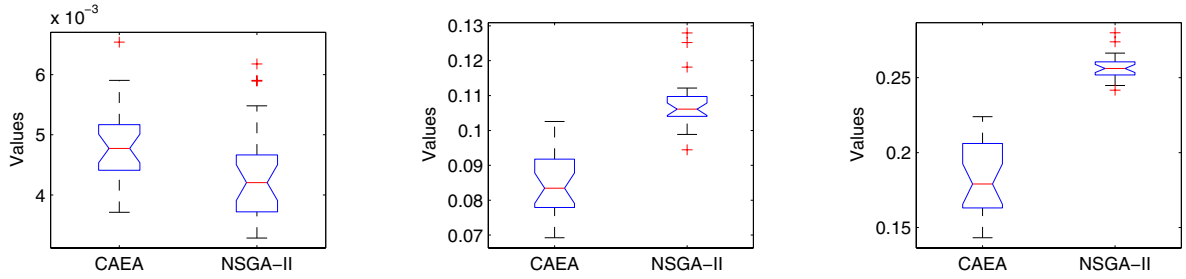
(a) 2 objectives, 11 variables (b) 4 objectives, 13 variables (c) 8 objectives, 17 variables

Fig. 10. ϵ_+ indicator values for CAEA and NSGA-II for DTLZ6 at Generation 25. CAEA is only able to outperform NSGA-II's standard operators on DTLZ6 with 2 objectives. No statistically significant results can be derived from the runs on higher objectives. However, a visual comparison indicates that CAEA performs poorer than NSGA-II on the higher objectives.



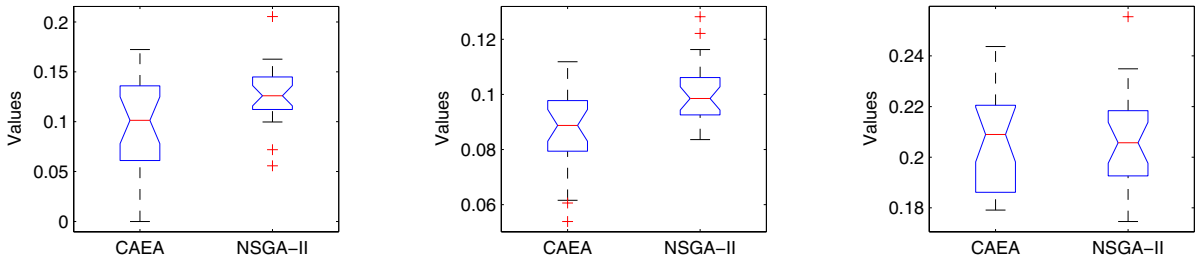
(a) 2 objectives, 11 variables (b) 4 objectives, 13 variables (c) 8 objectives, 17 variables

Fig. 11. ϵ_+ indicator values for CAEA and NSGA-II for DTLZ7 at Generation 25. CAEA is only able to outperform NSGA-II on DTLZ7 with 4 objectives. No statistically significant results can be derived from the runs on 2 and 8 objectives. However, a visual comparison suggests that CAEA performs better than NSGA-II even on higher objectives.



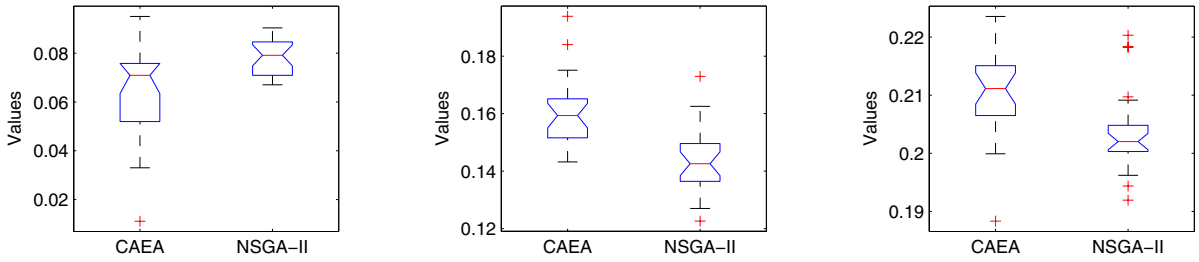
(a) 2 objectives, 11 variables (b) 4 objectives, 13 variables (c) 8 objectives, 17 variables

Fig. 12. ϵ_+ indicator values for CAEA and NSGA-II for DTLZ2 at Generation 50. CAEA performs better than NSGA-II on the problems with larger objectives. On DTLZ2 with 2 objectives, no statistically significant result can be concluded. However, a visual comparison suggests that NSGA-II has outperformed CAEA by the 50th generation.



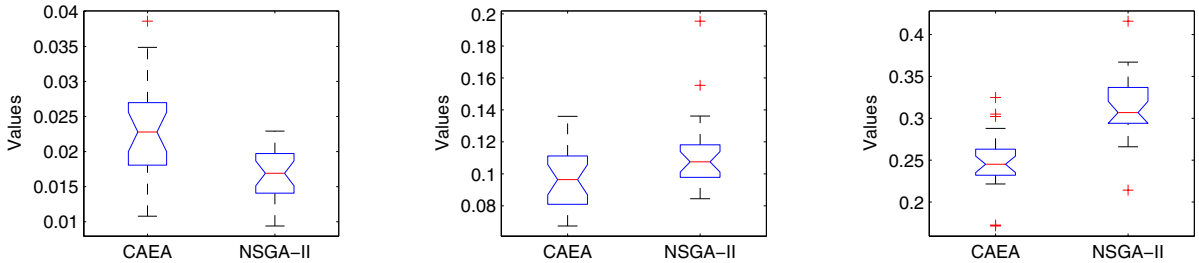
(a) 2 objectives, 11 variables (b) 4 objectives, 13 variables (c) 8 objectives, 17 variables

Fig. 13. ϵ_+ indicator values for CAEA and NSGA-II for DTLZ3 at Generation 50. No statistically significant result can be concluded for DTLZ3 with 2 and 8 objectives. With 4 objectives, CAEA outperforms NSGA-II.



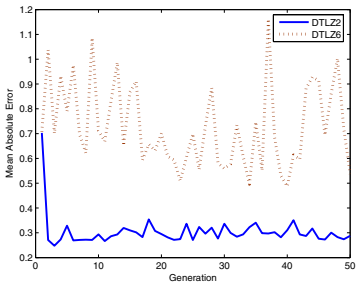
(a) 2 objectives, 11 variables (b) 4 objectives, 13 variables (c) 8 objectives, 17 variables

Fig. 14. ϵ_+ indicator values for CAEA and NSGA-II for DTLZ6 at Generation 50. CAEA outperforms NSGA-II on DTLZ6 with 2 objectives but performs poorly in comparison on the higher level objectives.

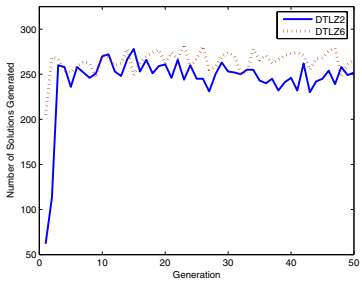


(a) 2 objectives, 11 variables (b) 4 objectives, 13 variables (c) 8 objectives, 17 variables

Fig. 15. ϵ_+ indicator values for CAEA and NSGA-II for DTLZ7 at Generation 50. CAEA finds significantly better solutions on DTLZ7 with 8 objectives. No statistically significant results can be obtained from the runs on 2 and 4 objectives. However, visual comparisons indicate that CAEA performs poorer compared to NSGA-II on the 2 objective problem and better on the 4 objective problem.



(a) Errors on DTLZ2 and DTLZ6 (4 objs)



(b) Number of Solutions Generated by Models

Fig. 16. While approximately the same number of solutions are generated on both problems, the coupled approximators makes significantly more errors on the DTLZ6 problem (dashed line) as compared to DTLZ2 (solid line).

TABLE II
KRUSKAL-WALLIS TEST FOR DTLZ PROBLEMS, GENERATION 25

	DTLZ2		
	$m = 2$	$m = 4$	$m = 8$
CAEA is better	$2.3e^{-16}$	$2.1e^{-15}$	$2.2e^{-16}$
NSGA-II is better	1.0	1.0	1.0
	DTLZ3		
	$m = 2$	$m = 4$	$m = 8$
CAEA is better	0.0030	$8.9e^{-05}$	0.0018
NSGA-II is better	0.99	0.99	0.99
	DTLZ6		
	$m = 2$	$m = 4$	$m = 8$
CAEA is better	$1.2e^{-10}$	H0	H0
NSGA-II is better	1.0	H0	H0
	DTLZ7		
	$m = 2$	$m = 4$	$m = 8$
CAEA is better	H0	0.00071	H0
NSGA-II is better	H0	0.99	H0

TABLE III
KRUSKAL-WALLIS TEST FOR DTLZ PROBLEMS, GENERATION 50

	DTLZ2		
	$m = 2$	$m = 4$	$m = 8$
CAEA is better	H0	$1.34e^{-15}$	$2.3e^{-16}$
NSGA-II is better	H0	1.0	1.0
	DTLZ3		
	$m = 2$	$m = 4$	$m = 8$
CAEA is better	H0	0.00040	H0
NSGA-II is better	H0	0.99	H0
	DTLZ6		
	$m = 2$	$m = 4$	$m = 8$
CAEA is better	0.0015	0.99	0.99
NSGA-II is better	0.99	$1.06e^{-06}$	0.00051
	DTLZ7		
	$m = 2$	$m = 4$	$m = 8$
CAEA is better	H0	H0	$1.61e-08$
NSGA-II is better	H0	H0	1.0

sample run on DTLZ6 as compared to DTLZ2.

At generation 50, CAEA outperforms NSGA-II on DTLZ2 (4 and 8 objectives), DTLZ3 (4 objectives), DTLZ6 (2 objectives) and DTLZ7 (8 objectives). For the rest of the problem instances, no statistical conclusion can be drawn except on the DTLZ6 problem (4 and 8 objectives) where NSGA-II outperforms CAEA.

The results indicate that for 3 of the 12 problems, i.e., DTLZ2 (2 Objectives), DTLZ3 (8 objectives) and DTLZ7 (2 objectives), CAEA had lost its early lead, allowing standard operators to "catch-up". Evidently, CAEA generated better solutions early in the optimization process and performance degraded later in the run. A plausible explanation for this phenomena is that as the population moved towards the Pareto optimal front, the samples became biased and the models were less able to generalize. The use of an archive to store previous solutions from which to base our models and/or local learning may circumvent this problem.

TABLE IV
MEAN ERRORS ON DTLZ2 AND DTLZ6

Problem	Mean Error	Standard Deviation
DTLZ2, 4 Objs.	0.3036	0.0624
DTLZ6, 4 Objs.	0.7283	0.1683
Problem	**Mean No. of Solutions Per Iteration**	**Standard Deviation**
DTLZ2, 4 Objs.	244.78	33.5356
DTLZ6, 4 Objs.	264.48	12.4904

C. Computational Time

The benefit obtained from using the coupled approximators is balanced by the extra computational time needed to create the models. Table V constrasts the computation time required by both algorithms on the DTLZ2 problem with 2, 4, and 8 averaged over 5 runs. The difference in computational time

TABLE V
AVERAGE COMPUTATION TIME ON DTLZ2 IN SECONDS

Algorithm	2 Objectives	4 Objectives	8 Objectives
CAEA	722.4	2018.2	7133.5
NSGA-II	1.4	2.8	7.2

is significant (approximately 3 orders larger), hence making coupled-approximators best suited to cases where the function evalutions are relatively expensive.

V. FUTURE WORK AND CONCLUSIONS

The experimental results obtained suggest that the use of models as operators, in particular the coupled approximators method, are promising alternatives or supplements to standard operators. On three of the four test problems (DTLZ2, DTLZ3 and DTLZ7), CAEA generated a better set of solutions early in the optimization process. However, these operators are not without bias and possible improvements include the following:

A. Local Models and Archives

Prior research [19] [20] [13] suggests that the coupled approximators method may be improved by clustering the objective space and generating local models instead of a global model. The use of an archive to store previously evaluated individuals may also assist in the modeling process.

B. Incorporation of Domain Knowledge

The No Free Lunch (NFL) theorem [21] justifies the use of prior knowledge in the search and optimization of a particular problem. CAEA can be modified in two ways to utilize available prior information. First, a problem specific approximation or modeling technique can be used in place of the RBFNs used in this work. The choice of any one particular technique can be based on prior knowledge, required accuracy and computational time. Secondly, the solution construction algorithms presented in this paper are simple and were developed to illustrate the viability of using coupled approximators. It remains future research to develop more efficient general and/or problem specific construction algorithms.

ACKNOWLEDGMENT

This work was funded by the A*STAR SERC Grant (No. 052 015 0024) for the Grid-based PSE for Engineering of Materials (GPEM) project.

REFERENCES

[1] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multiobjective optimization test problems," in *Congress on Evolutionary Computation (CEC'2002)*, vol. 1, 2002, pp. 825–830. [Online]. Available: citeseer.ist.psu.edu/deb02scalable.html

[2] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II," in *Proceedings of the Parallel Problem Solving from Nature VI Conference*, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds. Paris, France: Springer. Lecture Notes in Computer Science No. 1917, 2000, pp. 849–858. [Online]. Available: citeseer.ist.psu.edu/deb00fast.html

[3] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," Gloriastrasse 35, CH-8092 Zurich, Switzerland, Tech. Rep. 103, 2001. [Online]. Available: citeseer.ist.psu.edu/zitzler02spea.html

[4] J. D. Knowles and D. Corne, "Approximating the nondominated front using the pareto archived evolution strategy," *Evolutionary Computation*, vol. 8, no. 2, pp. 149–172, 2000. [Online]. Available: citeseer.ist.psu.edu/article/knowles00approximating.html

[5] C. A. C. Coello, D. A. V. Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, 2002.

[6] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley and Sons, Chichester, UK, 2001.

[7] M. Salahuddin, Y. S. Ong, T. Hung, B. S. Lee, H. Soh, Y. X. Ren, and E. Sulaiman, "Grid-based PSE for engineering of materials (GPEM)," in *GCA 2007 - 3rd International Workshop on Grid Computing Applications*, 2007.

[8] J. Knowles, "ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 50–66, 2006.

[9] Y. S. Ong, P. B. Nair, and A. J. Keane, "Evolutionary optimization of computationally expensive problems via surrogate modeling," *American Institute of Aeronautics and Astronautics Journal*, vol. 41, pp. 689–696, 2003.

[10] M. Emmerich, K. Giannakoglou, and B. Naujoks, "Single- and multi-objective evolutionary optimization assisted by gaussian random field metamodels." [Online]. Available: citeseer.ist.psu.edu/739286.html

[11] A. Ratle, *Parallel Problem Solving from Nature - PPSN V*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1998, vol. 1498/1998, ch. Accelerating the Convergence of Evolutionary Algorithms by Fitness Landscape Approximation.

[12] Y. S. Ong, P. B. Nair, and K. Y. Lum, "Max-Min surrogate-Assisted evolutionary algorithm for robust aerodynamic design," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 392–404, 2006.

[13] Z. Z. Zhou, Y. S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum, "Combining global and local surrogate models to accelerate evolutionary optimization," *IEEE Transactions On Systems, Man and Cybernetics - Part C*, vol. 37, 2007. [Online]. Available: http://ntu-cg.ntu.edu.sg/ysong/journal/GLSurrogate.pdf

[14] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing Journal*, 2003. [Online]. Available: citeseer.ist.psu.edu/jin03comprehensive.html

[15] Y. S. Ong, P. B. Nair, A. J. Keane, and K. W. Wong, *Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems*. Springer Verlag, 2004, vol. 10, pp. 307–332.

[16] S. Nissen. Fast artificial neural network library v2.0.0. [Online]. Available: http://leenissen.dk/fann/

[17] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler, "PISA — a platform and programming language independent interface for search algorithms," in *Evolutionary Multi-Criterion Optimization (EMO 2003)*, ser. Lecture Notes in Computer Science, C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, Eds. Berlin: Springer, 2003, pp. 494 – 508.

[18] J. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich," 214, July 2005.

[19] H. Soh and M. Kirley, "moPGA: Towards a new generation of multiobjective genetic algorithms," in *Congress on Evolutionary Computation (CEC'2006)*, 2006, pp. 1702–1709.

[20] R. Regis and C. Shoemaker, "Local function approximation in evolutionary algorithms for the optimization of costly functions," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 490–505, 2004.

[21] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, 1997.