SNDL-MOEA: Stored Non-Domination Level MOEA

Matt D. Johnson
University of Missouri-Rolla
1870 Miner Circle
Rolla, Missouri 65409-0360
matt@umr.edu

Daniel R. Tauritz
University of Missouri-Rolla
1870 Miner Circle
Rolla, Missouri 65409-0360
tauritzd@umr.edu

Ralph W. Wilkerson
University of Missouri-Rolla
1870 Miner Circle
Rolla, Missouri 65409-0360
ralphw@umr.edu

ABSTRACT

There exist a number of high-performance Multi-Objective Evolutionary Algorithms (MOEAs) for solving Multi-Objective Optimization (MOO) problems; two of the best are NSGA-II and ϵ-MOEA. However, they lack an archive population sorted into levels of non-domination, making them unsuitable for construction problems where some type of backtracking to earlier intermediate solutions is required. In this paper we introduce our Stored Non-Domination Level (SNDL) MOEA for solving such construction problems. SNDL-MOEA combines some of the best features of NSGA-II and ϵ-MOEA with the ability to store and recall intermediate solutions necessary for construction problems. We present results for applying SNDL-MOEA to the Tight Single Change Covering Design (TSCCD) construction problem, demonstrating its applicability. Furthermore, we show with a detailed performance comparison between SNDL-MOEA, NSGA-II, and ϵ-MOEA on two standard test series that SNDL-MOEA is capable of outperforming NSGA-II and is competitive with ϵ-MOEA.

Categories and Subject Descriptors

I.2.M [**Computing Methodologies**]: Artificial Intelligence

General Terms

Algorithms, Design, Experimentation.

Keywords

Evolutionary Multiobjective Optimization, Pareto Optimality, Constructive Problem Solving.

1. INTRODUCTION

Multi-Objective Optimization (MOO) problems are characterized by having multiple objectives which need to be simultaneously optimized and which may conflict with one another [1]. Multi-Objective Evolutionary Algorithms (MOEAs) are an extension of the standard EA capable of solving MOO problems. A solution a to a MOO problem is said to *dominate* a solution b if, for every objective, a is either equal to or superior to b and for at least one objective a is superior to b. The set of solutions which is superior is referred to as the *non-dominated* set, while the other individuals are said to be *dominated* [1]. An arbitrary set of solutions can be sorted into levels of non-domination by iteratively removing the non-dominated solutions and placing those subsets in levels.

There exist a number of high-performance MOEAs; two of the best are NSGA-II [3] and ϵ-MOEA [2]. However, they lack an archive population sorted into levels of non-domination, making them unsuitable for construction problems where some type of backtracking to earlier intermediate solutions is required. The Tight Single Change Covering Design (TSCCD) problem [8] is such a construction problem. It requires assembling a sequence of blocks of integers. The problem uses two variables which dictate the size of the problem. The first variable, v, indicates the number of integers available for use. The second variable, k, indicates the number of integers in a block, which is just a set of integers. A TSCCD must have the following traits [8]:

1. Every possible pair of integers in the range 1 to v occurs at least one time.
2. Each block contains the same integers as the block which precedes it, except for the transfer. A transfer is an element of a block that differs from the values in the previous block.
3. Each block contains $k - 1$ pairs which can be constructed with the transfer and one other element of that block. These pairs do not occur in previous blocks.

TSCCD problems can be solved with extensive backtracking but at great computational expense [7]. In order to solve them with a MOEA, we construct solutions by starting with a single block and incrementally adding blocks one at a time. Storing the solutions in various stages of the construction process allows the same partial solution to be used in multiple construction attempts.

In this paper we introduce our Stored Non-Domination Level (SNDL) MOEA for solving construction problems such as TSCCD. SNDL-MOEA combines some of the best features of NSGA-II and ϵ-MOEA with the ability to store and recall intermediate solutions necessary for construction problems; some of its salient features are as follows:

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

- The main population is stored in a special data structure that maintains levels of non-domination. This makes it unnecessary to re-sort the population every generation like NSGA-II does.

- The number and size of the levels of non-domination are user configurable so that the user can fully customize the depth and breadth of the archive for problem specific parameter tuning.

- It incorporates a deterioration prevention mechanism in the form of negative efficiency preservation (see Section 2).

In Section 2 we provide some background on MOEAs, including the performance metrics we use for comparison and descriptions of NSGA-II and ϵ-MOEA, the two algorithms which inspired our algorithm and against which we compare it. Section 3 describes our SNDL-MOEA in detail, while in Section 4 we present a detailed performance comparison between SNDL-MOEA, NSGA-II, and ϵ-MOEA on two standard test series, showing that SNDL-MOEA is capable of outperforming NSGA-II and is competitive with ϵ-MOEA on general MOO problems. In the same section we also present results for applying SNDL-MOEA to the TSCCD construction problem, demonstrating its applicability. This is followed by a brief discussion of the results in Section 5, and conclusions and some ideas for future work in Section 6.

2. BACKGROUND

The goal of most MOEAs is to find the Pareto-optimal set of solutions. Given the set of all solutions to a given problem (including those which have not been discovered by the algorithm), the Pareto-optimal set will be the subset of solutions which are non-dominated [1]. This set will contain all the best solutions; within that set users can select a solution which fits their particular needs.

One of the problems associated with many MOEAs occurs during the removal of individuals from the population. While it is necessary to remove individuals to keep the population size in check, doing so can lead to a condition known as *deterioration*, which is the situation in which one or more individuals are dominated by an individual which previously existed in the solution set, but is no longer [6]. Deterioration can be prevented by employing an efficiency preservation method, which involves accepting a new solution only when the solution dominates one or more of the current solutions. Alternately, negative efficiency preservation (deleting an individual only when a superior individual is inserted) may be employed to the same end [6].

While our goal was to develop a MOEA with special storage properties appropriate for solving construction problems, we wanted our MOEA still to be competitive with the best existing general purpose MOEAs. In the following subsections we provide background on the metrics we used to evaluate our MOEA's performance and describe NSGA-II and ϵ-MOEA, two of the best MOEAs currently known and the inspiration for SNDL-MOEA.

2.1 Metrics

Two important metrics for comparing MOEAs based on the quality of the solution sets they generate are (1) convergence, which measures how close the generated solutions are to the Pareto-optimal front, and (2) diversity, which measures how well distributed the solution set is. Both measures and our use of them are described next.

Convergence Metric. The convergence metric used in this research is essentially the same as the metric presented in [2]. First, a Pareto-optimal front is generated for each test problem. Then, each generated individual is compared to the solution set and the distance from the individual to the closest Pareto-optimal solution is calculated. The average of these distances is the measure of convergence.

Diversity Metric. In MOEA applications, the user is typically interested in obtaining an evenly distributed set of solutions residing on the generated Pareto front, devoid of clusters of solutions.

The diversity metric we employed is based on the Voronoi Diagram. Steven S. Skiena discusses this diagram in *The Algorithm Design Manual* [9]. The problem description reads as follows: "Decompose space into regions around each point such that all the points in the region around p_i are closer to p_i than they are to any other point in S". Thus, a Voronoi Diagram of a solution set generated by a MOEA would select a region of space around each of the individuals. In a perfectly distributed solution, each region would have exactly the same area.

We employed the Qhull software package[1] to produce Voronoi Diagrams. The diversity metric is calculated as follows: First, the Qhull software is employed to draw a Voronoi Diagram of a solution set. Then, the standard deviation of the areas in each of these solutions is calculated (over all 30 runs). This value is the metric used to compare the diversity of solutions generated. Smaller values are preferable, because a smaller value reflects less variation in the areas.

2.2 NSGA-II

An algorithm called the "nondominated sorting-based multi-objective evolutionary algorithm II" (NSGA-II) is currently the benchmark against which other MOEAs are compared. This algorithm is efficient and produces good results, but suffers from deterioration. We have tried to maintain consistency in this paper with the language and symbology used in [3] which contains a detailed description and analysis of NSGA-II.

NSGA-II begins with a series of five steps which sets the stage for the primary loop. The first of these steps is random initialization of the population P_0. This population is then sorted on the basis of nondomination using a special non-dominated sorting algorithm. The best rank is level 1; members of this group are not dominated. Members of level 2 can be dominated only by members of level one. Thus, members of a particular level can be dominated by members of lower numbered levels, but not higher numbered levels. The fitness of every individual is set equal to that individual's nondomination level. This "fitness" value is then used in Binary Tournament Selection. Mutation and recombination operations are then performed, resulting in the offspring population Q_0. At this point, the initial five steps are complete, and the algorithm will enter its main evolutionary loop.

[1] `http://www.qhull.org`

The primary loop begins by defining R_t to be the combination of P_t and Q_t ($t = 0$ the first time). Thus, R_t includes all the individuals from the previous iteration, which makes it elitist. R_t is then sorted on the basis of nondomination, leading to the creation of P_{t+1}.

The creation of P_{t+1} consists of adding nondomination levels of R_t to P_{t+1} until adding the next level would cause the size of P_{t+1} to exceed the population size N (see Table 1 for the variables and constants used in our complexity analysis). Each level is subjected to the Crowding Distance operator before its individuals are added to the population. Then, if P_{t+1} is not large enough, the next nondomination level is sorted using the Crowded Comparison Operator $\prec_n$ (Algorithm 1). Individuals from that level are then added to P_{t+1} until it is of size N.

Algorithm 1 NSGA-II: Crowded Comparison Operator $\prec_n$ (Individual i, Individual j)

```
if $i_{rank} < j_{rank}$ then
    return true
else
    if $((i_{rank} = j_{rank})\ and\ (i_{distance} > j_{distance}))$ then
        return true
    else
        return false
    end if
end if
```

After P_{t+1} is created, Q_{t+1} will be defined to be the result of Binary Tournament Selection, Mutation, and Recombination applied to P_{t+1}. This is the last step in the primary loop. The loop will continue until the terminating condition is met.

The overall complexity per generation can be calculated by considering three major components of the algorithm. The nondominated sorting algorithm requires $O(M(2N)^2)$, the crowding distance assignment requires $O(M(2N)\ log(2N))$, and sorting requires $O(2N\ log(2N))$. This makes the overall complexity $O(MN^2)$.

2.3 ϵ-MOEA

ϵ-MOEA is a steady-state, elitist EA which seeks to obtain a diverse population quickly without suffering from the affects of deterioration. The following is a brief summary of the algorithm description available in [2].

ϵ - MOEA begins with the creation of an initial population $P(0)$. The ϵ non-dominated solutions from $P(0)$ are assigned to an archive population $E(0)$. ϵ dominance is a modified form of dominance and is presented in [6]. One individual from P and one individual from E are chosen. These individuals will mate and produce an offspring, c_i. This individual is assigned an identification array B of length M which is defined as follows:

$$B_j(f) = \begin{cases} \lfloor \frac{(f_j - f_j^{min})}{\epsilon_j} \rfloor, & \text{for minimizing } f_j \\ \lceil \frac{(f_j - f_j^{min})}{\epsilon_j} \rceil, & \text{for maximizing } f_j \end{cases} \quad (1)$$

where f_j represents the j^{th} objective and f_j^{min} its minimum possible value, and ϵ_j represents the tolerance allowed in the j^{th} objective [2]. The purpose of this is to reduce the significance of the objective values, allowing a given individual to dominate a greater number of other individuals.

After the creation of c_i, an attempt is made to insert c_i into the archive population. If the B array of any archive member dominates the B array of c_i, then c_i is said to be ϵ - dominated by that archive member and will not be added to the archive. If the B array of c_i dominates the B array of a particular archive member, that archive member will be deleted and replaced with c_i. If neither of the two previous cases occur, c_i is ϵ non - dominated within the archive, and may or may not be inserted into the population, based on a closer inspection (see [2] for details.) Additionally, the offspring c_i may be inserted into the population $P(t)$ if it dominates a member of $P(t)$. In this case, one of the dominated members will be replaced by c_i.

This process of selection, mating, and insertion executes for a specified number of iterations. The archive members make up the solution presented to the user. The complexity of ϵ-MOEA is not explicitly stated in [2], but appears to be $O(CMN)$. For comparison purposes, we will assume the number of children is equivalent to the population size, as done in NSGA-II, which results in $O(MN^2)$.

3. ALGORITHM DESCRIPTION

We now describe SNDL-MOEA. The user can set the number of domination levels, limit the size of the levels, and choose from three different selection approaches. Additionally, the user chooses the granularity of the objective comparison, the mutation and recombination methods, and the domination operation; these characteristics are incorporated into one problem specific object.

Parent Selection. The selection procedures utilize levels of non-domination, similar to the approach employed by NSGA-II [3]. The reader should note that SNDL-MOEA depends on the use of a container which stores the population in ranks of non-domination. This data structure is further explained later in this section. There are three different selection procedures from which the user can choose; those procedures are described next.

The first selection procedure creates a high selection pressure. The specified number of parents is selected at random from the top level. Random selection from the top level will continue until the desired number of parents has been reached. Duplicate parents are allowed.

The second selection process begins with the highest nondomination level. If the number of parents needed is less than the size of the level, parents will be selected randomly from that level. If the number of parents needed is greater than or equal to the size of the level, all individuals from the level are chosen. This approach will move from the current level to the next best as needed until enough parents are selected, or all levels have been visited. If all levels are visited and the number of parents has not been reached, parents are chosen at random from the population until the desired number is reached.

The third selection procedure chooses parents at random from the entire population until the desired number of parents is reached.

Each selection procedure will add a few randomly generated individuals to the group of parents. This small addition is above and beyond the required number of parents and causes large jumps in a few of the individuals during recombination. This is intended to improve the robustness

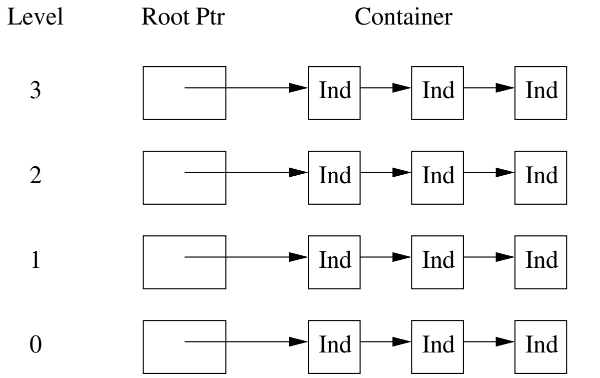

Figure 1: Population data structure with 4 levels

of the algorithm in problems with large search spaces which might not be adequately sampled by the initial population.

Recombination and Mutation. Following the selection process, the parents recombine to create children which are subjected to mutation. The recombination and mutation details are implemented in a problem specific class, which we will refer to as *Ind*. The problem specific class *Ind* is required to provide a function named *CrossAndMutate*, which will take another *Ind* as an argument. Both the crossover and mutation operations are implemented in this function. The programmer is free to implement any kind of crossover and mutation, provided the desired operations can be performed with only two parents.

Section 4 describes the use of a number of test problems used by the authors to test SNDL-MOEA against other algorithms. In these test problems, the genotype is typically implemented with a vector of floating point numbers. While each problem is implemented in its own special version of *Ind*, the *CrossAndMutate* operation typically proceeds as follows:

1. Choose a crossover point somewhere in the vector
2. Perform a typical one point crossover
3. Randomly choose one value v in the vector
4. Randomly add or subtract a small value in the range [0 - 1] to v
5. Ensure that the resulting gene is valid

Variations on this approach are employed as needed, but this sequence of steps represents the most common technique employed by the authors.

Insertion. In order to eliminate the computation of a fitness value and a non-dominated sort [3], the population is stored in ranks of non-domination. The container used to store the population consists of a vector of root pointers. Each root pointer refers to a container. The "top" level is considered the best. (The top level has the largest index.) Individuals in this level dominate individuals in lower levels. There are no duplicate individuals allowed in the population. Figure 1 illustrates this data structure.

The insertion procedure is shown in Algorithm 2. Children are checked against each level, starting with the top level and moving down as needed. If a child dominates every single individual in a given level, that child is said to strongly dominate the level. If the child dominates some, but not all individuals in a level, it weakly dominates that level [1]. A child which does not dominate and is not dominated by individuals in a level is said to be equal to that level.

If the child is found to be equal to a level, it is inserted into that level. If the child is a duplicate of an existing individual, the insertion operation has no impact. If the child strongly dominates a level, a new level above that one is created and the child is inserted there. If a child weakly dominates a level, the child is inserted into that level, and all the solutions it dominates are deleted from that level and inserted into a lower level. If after checking every level no place for the child is found, then it is an inferior solution, so it is deleted. If the number of levels exceeds the user specified limit, the worst level is deleted. This approach is similar to the update function for the ϵ - Pareto set as outlined in [6].

The current implementation for the level container is a binary search tree. Insertion into a binary search tree requires an object to have both a less than operator and an equivalency operator. The operators truncate the objectives based on a problem specific, programmer specified value. (For example, a comparison of floating point numbers in the range from zero to one might utilize only two digits.) This truncation limits the number of potential solutions, therefore limiting the size of the tree. (The truncation approach was inspired by the concept of ϵ - dominance as described in [6]. A standard dominance check is used by SNDL-MOEA.)

The size of the top non-domination level is limited by the objective truncation as described in the previous paragraph. The size of each lower level is limited by a user defined value. If an individual is inserted into one of the lower levels and the size exceeds the limit, the level is searched for the two individuals which are closest (Euclidean distance in objective space) to each other, and one of them is deleted. Since an individual in the top level is only deleted when it is replaced by a superior individual, the algorithm is guaranteed to converge to a solution [6]; this solution is not necessarily Pareto-optimal.

The result of this insertion approach is a data structure that maintains a number of different levels of non-domination. Every solution is unique. The best level of non-domination is always the top one. Additionally, a solution is not deleted unless it is certain that a better solution exists in the population, which prevents deterioration [6].

3.1 Complexity

The complexity of the insertion is described next, with all the relevant variables summarized in Table 1. First, the outer loop runs one iteration per child, which is $O(C)$, with C equal to the number of children. Next, the inner for loop runs once per domination level. Checking to see if an individual belongs to a domination level requires a domination check against all the individuals in that level. If the number of objectives is M and the maximum size of a level is Z, this requires $M \cdot Z$ operations, which brings the complexity to $O(CDMZ)$, where D is the number of domination levels. Removing one of the two closest individuals requires $M \cdot Z^2$

Table 1: Complexity analysis variables

Variable	Description
C	Number of children
D	Number of domination levels
M	Number of objectives
Z	Maximum level size
Z_t	Size of the top level
N	Population Size

operations, which gives us $O(CDMZ^2)$. This applies to all levels but the top level.

The top level is not bounded by Z. Individuals are not removed from this level, so the $M \cdot Z^2$ removal operation is not needed either. Insertion into the top level is linear in the level size; the size of the top level is bounded only by the objective truncation method used in the less than and equivalency operators as described previously. This approach allows the top level to grow beyond the Z size bound. However, as superior individuals are generated, the top level will be moved down quickly before it becomes large, which will cause its size to be pruned to Z. While pruning the level is a $M \cdot Z_t^2$ (Z_t is the size of the top level) operation, the Z_t will be reasonable in the early stages of the algorithm when a level shift occurs frequently. The Z_t will become large only when it contains members of the Pareto-optimal front, at which time it will never be moved down.

Thus, the overall complexity for the insertion algorithm per generation is $O(CDMZ^2)$). The term C indicates the number of children in a generation which will be set to N (population size) to match NSGA-II. This makes the average complexity of the algorithm $O(DMNZ^2)$.

Algorithm 2 Insert into the population

```
for all $x \in ChildPop$ do
  for $d = top\_level$ to $lowest\_level$ do
    Determine whether $x$ dominates level $d$
    if $x$ is equal to $d$ then
      Insert $x$ into $d$
      while $|d| > D\_SIZE$ AND $d \neq top\_level$ do
        Remove one of the two closest individuals
      end while
    else if $x$ strongly dominates $d$ then
      Create a new level
      Insert $x$ into the new level
      Insert the new level above $d$
    else if $x$ weakly dominates $d$ then
      Insert $x$ into $d$
      Remove dominated individuals from $d$
    end if
  end for
  while $numberLevelsUsed > numberLevelsAllowed$ do
    Delete the lowest level
  end while
end for
```

4. RESULTS

In this section SNDL-MOEA is compared to NSGA-II [3] and ϵ-MOEA [2]. NSGA-II converges quickly and produces a diverse population. ϵ-MOEA has the advantage of being not only fast, but also guaranteeing convergence. Code for both these algorithms was obtained from the Kanpur Genetic Algorithms Laboratory website[2].

Table 2: ZDT Algorithm Parameters

	ϵ-MOEA	NSGA-II	SNDL
Pop Size	100	100	100
Num Children	1	100	100
Num Generations	20000	200	200
NumEvals	20000	20000	20000
Eps 0	0.0075	N/A	N/A
Eps 1	0.0075	N/A	N/A
Crossover Prob	1	0.9	1
Mutation Prob	0.033	0.033	1
EtaC	15	15	N/A
EtaM	20	20	N/A
LevelSize	N/A	N/A	100
NumLevels	N/A	N/A	4
Selection	N/A	N/A	First

Five test problems used in this research are from Zitzler, Deb, and Thiele's ZDT test problem series [1]. They all seek to minimize the following two objectives:

$$f_1(\mathbf{x}) \qquad (2)$$

$$f_2(\mathbf{x}) = g(\mathbf{x})h(f_1(\mathbf{x}), g(\mathbf{x})) \qquad (3)$$

Each problem in this series has its own definition for $f_1(\mathbf{x})$, $g(x)$ and $h(f_1, g)$ [1].

Another five test problems utilized in this research come from Deb, Thiele, Laumanns, and Zitzler's DTLZ test problem series [4]. This problem set is considerably more difficult than ZDT. The user can specify the number of objectives as well as other problem features to customize the problems. The examples employed here all use three objectives. Additionally, they are implemented using the settings recommended in the paper describing the problems [4].

For all these test problems, NSGA-II was set to run 200 iterations with the number of children set to 100. ϵ-MOEA was set to run 20,000 generations with 1 child per generation. SNDL-MOEA was set to run 200 iterations with the number of children set to 100. These parameter settings ensure that each algorithm will use an equal number of fitness evaluations (20,000 each) for fair comparison. Each algorithm was executed 30 times per problem to facilitate statistical analysis.

The convergence and diversity metrics described in Section 2 were applied to our experimental results. A two-sample two-tailed t-test with $\alpha = 0.05$ was used to ascertain whether the differences in convergence and diversity are statistically significant. Results from this test are included with the discussion of each sample problem.

Table 2 specifies the parameters used for each algorithm on the ZDT problems, while convergence and diversity metrics for the ZDT problems are presented in Table 4. Table 3 specifies the parameters used for each algorithm on the DTLZ problems, while convergence and diversity metrics for the DTLZ problems are presented in Table 5.

[2] `http://www.iitk.ac.in/kangal/codes.shtml`

Table 3: DTLZ Algorithm Parameters

	ϵ-MOEA	**NSGA-II**	**SNDL**
Pop Size	100	100	100
Num Children	1	100	100
Num Generations	20000	200	200
NumEvals	20000	20000	20000
Eps 0	0.02	N/A	N/A
Eps 1	0.02	N/A	N/A
Eps 2	0.05	N/A	N/A
Crossover Prob	1	0.9	1
Mutation Prob	0.033	0.1	1
EtaC	15	15	N/A
EtaM	20	20	N/A
LevelSize	N/A	N/A	100
NumLevels	N/A	N/A	4
Selection	N/A	N/A	First

Table 4: ZDT Results

	ϵ-MOEA	**NSGA-II**	**SNDL**
ZDT1			
Avg Con	0.00162059	0.00241427	**0.00039867**
Avg Div	**0.00082819**	0.00107833	**0.00089480**
ZDT2			
Avg Con	0.00238283	0.00241762	**0.00040317**
Avg Div	0.00101474	0.00098353	**0.00088526**
ZDT3			
Avg Con	0.00105527	0.00138903	**0.00061604**
Avg Div	0.00363732	**0.00129403**	0.00167855
ZDT4			
Avg Con	**0.00630488**	0.01467574	0.02256824
Avg Div	**0.00086732**	**0.00098623**	0.00438463
ZDT6			
Avg Con	0.00662071	0.01649615	**0.00291917**
Avg Div	0.00029392	**0.00022953**	0.00085335

4.1 ZDT Test Problem Series

ZDT1. T-Test analysis indicates that the difference in the average convergence between each of the algorithms is significant. SNDL-MOEA shows the best convergence, followed by ϵ-MOEA, with NSGA-II last. In terms of diversity, ϵ-MOEA and SNDL-MOEA are about the same; both are better than NSGA-II.

ZDT2. The t-test indicates no significant difference between ϵ-MOEA and NSGA-II. The convergence of SNDL-MOEA is better than the other two algorithms. There is no significant difference in diversity between NSGA-II and ϵ-MOEA. SNDL-MOEA displays better diversity than either of the other algorithms.

ZDT3. The only significant convergence difference is between SNDL-MOEA and NSGA-II, with SNDL-MOEA being slightly better. All of the differences between the diversity metric are significant, with NSGA-II being the best, followed by SNDL-MOEA, and ϵ-MOEA being worst.

Table 5: DTLZ1 Results

	ϵ-MOEA	**NSGA-II**	**SNDL**
DTLZ1			
Avg Con	0.31408666	0.31296544	0.34627738
Avg Div	**0.00016548**	**0.02627617**	0.00031916
DTLZ2			
Avg Con	**0.01889054**	0.02340261	0.02686478
Avg Div	0.00019262	0.00065760	**7.82032E-05**
DTLZ3			
Avg Con	**6.30348067**	42.64948	**5.84609866**
Avg Div	**0.32994313**	3269.401309	0.46402944
DTLZ4			
Avg Con	0.02931693	0.02833454	0.02699411
Avg Div	**0.00019657**	0.00072440	0.00184699
DTLZ5			
Avg Con	0.03070071	0.02654871	**0.02111537**
Avg Div	**7.4123E-07**	2.84239E-06	1.89804E-06

ZDT4. ZDT4 is a problem which involves a number of local Pareto-optimal fronts which parallel the global front. $\epsilon - MOEA$ achieves superior convergence on this problem. There is no significant difference between NSGA-II and SNDL-MOEA.

Initial experiments with this problem yielded very poor results for SNDL-MOEA. To improve convergence, mutation for the problem was modified such that the size of the jumps differs based on the generation number. This variation on mutation involves taking larger jumps in the early stages of the algorithm, and smaller jumps near the end. The size of the value added to v (v is a randomly chosen element from $\mathbf{x}$) is stored in *val*, and can be calculated in one of three ways:

1. In the first ten percent of generations: $val = rand + 1$, where rand is a real number in the range [0 - 1]
2. In the middle 80 percent of generations: $val = rand$
3. In the last 10 percent of generations: $val = generationsLeft/maxGenerations$

This tweak improved convergence, but not enough to overtake the other two algorithms. ϵ-MOEA and NSGA-II are to be preferred in terms of diversity on this problem, with no significant difference between them.

ZDT6. The differences in average convergence are significant, with SNDL-MOEA converging to a better solution than do the other two algorithms. The difference shown between their average diversities is also significant, with NSGA-II having the best diversity, followed closely by ϵ-MOEA.

4.2 DTLZ Test Problem Series

DTLZ1. With this problem, there is no significant difference in the convergence values; all three algorithms perform equally well. In terms of diversity, ϵ-MOEA is better than the other two. There is no significant difference in the diversity of NSGA-II and SNDL-MOEA.

DTLZ2. Statistical analysis indicates that the differences between average convergence and average diversity are all significant. ϵ-MOEA has the best convergence, while SNDL-MOEA has the best diversity.

DTLZ3. Statistical analysis indicates that the convergence of ϵ-MOEA and SNDL-MOEA is not significantly different. The other differences between convergence and diversity are significant. ϵ-MOEA is the best, SNDL-MOEA is second best, and NSGA-II is worst.

DTLZ4. The differences between the convergence values on all three algorithms is so small that none of them are statistically significant. However, the diversity metrics are significant, with ϵ-MOEA having a slight edge over the other two algorithms.

DTLZ5. The average differences shown for convergence are all statistically significant, with SNDL-MOEA having the best convergence for this problem. The diversity of ϵ-MOEA is the smallest for this problem; this is also statistically significant.

4.3 TSCCD

Solving a TSCCD is no easy task. Prior to the publication of [7] by Phillips, the largest known TSCCD was k = 4, v = 12. (Henceforth, we will use the notation TSCCD(v, k).)

Because the content of each block depends on the content of all the previous blocks, solutions are constructed one block at a time, rather than all at once. Tables 6 and 7 contain a solution to TSCCD(12,4) that was generated by SNDL-MOEA. Each column represents one block. Thus, the first block is $(1, 2, 3, 4)^T$, and the second block is $(1, 2, 9, 4)^T$. Note that the transfers are underlined in this solution.

Table 6: TSCCD(12,4) Part 1

1	1	1	1	1	1	1	1	1	3	3
2	2	5	6	11	7	7	7	12	12	12
3	9	9	9	9	9	9	9	9	9	6
4	4	4	4	4	4	10	8	8	8	8

Table 7: TSCCD(12,4) Part 2

3	4	2	2	2	2	2	11	11	11
12	12	12	11	5	5	5	5	5	5
10	10	10	10	10	10	7	7	7	7
8	8	8	8	8	6	6	6	3	12

Representation. The solution is represented as a vector of blocks, where each block is an object containing k integers in the range $1, 2, ...v$.

Dominance. Solutions to the TSCCD are generated one block at a time, and there will be solutions of varying lengths in the population. In order to determine whether one solution is better than another, the number of pairs that have occurred in the solution are counted and this number is compared to the total number of pairs required of a correct solution. This difference is to be minimized. A solution is not reached until this difference is zero.

Distance. The distance between two solutions is defined as the number of values which differ between solutions. Two solutions are compared one entry at a time. Every time a difference is encountered, the distance is incremented.

Crossover and Mutation. Since the correctness of a given block depends on the blocks which precede it, mutation is impractical, as one change could invalidate an entire solution. Crossover is not performed for the same reason. Instead, an attempt is made to generate a new correct block and append it to the solution.

Results. Initially in the construction phase, it is easy to select a block that is correct in relation to the blocks that precede it. However, when most of the blocks have been chosen, the last few blocks will typically be impossible to find, because of a mistake that was made early on, but was "correct" in terms of that block following the rules. The number of blocks generated will drop dramatically early in the evolutionary run, but will quickly level off as various routes through the search space are tried.

As the algorithm runs, the overall size of the population will increase until all the levels of nondomination are filled up. The lower levels will fill first, and gradually, the middle levels will fill. The upper levels will be more difficult to fill, and indeed may not fill up. However, the population will continue to evolve over time, and, theoretically, will always converge to a correct solution.

The solution shown in Tables 6 and 7 was generated with SNDL-MOEA; the algorithm ran for a period of several days to generate this solution. Work is currently underway to solve TSCCD(20,5), but to date, we have not yet found a solution.

5. DISCUSSION

NSGA-II uses the concept of levels of non-domination, in which individuals are separated into groups; individuals in each group do not dominate each other. This idea of using levels of non-domination is also employed by SNDL-MOEA. The distinction between the two uses of this concept is that NSGA-II does a sort every generation [3], while SNDL-MOEA maintains a sorted list across the generations. This difference allows SNDL-MOEA to benefit from knowing which individuals are the best, and reduces the number of computations involved to get that information.

SNDL-MOEA, NSGA-II, and ϵ-MOEA all employ a diversity preservation method. NSGA-II uses a crowding distance computation and a crowded comparison operator, which are utilized during population creation every iteration. ϵ-MOEA does not explicitly calculate the distances between individuals. Instead, it utilizes a special identification array (B) which is built from abbreviated versions of the objective values. Only individuals with a unique B array are allowed into the archive. This approach divides the search space into a number of hyper-boxes, and maintains diversity by allowing only one solution into each box. SNDL-MOEA utilizes a concept similar to the B array; truncation of objective values occurs in the comparison operators of the individual. Only unique individuals are allowed in SNDL-MOEA containers.

NSGA-II suffers from deterioration. However, the solutions it generates will still be very close to the Pareto-optimal set. SNDL-MOEA and ϵ-MOEA do not suffer from deterioration, so they are able to get closer to the Pareto-optimal front.

6. CONCLUSIONS AND FUTURE WORK

NSGA-II, ϵ-MOEA, and SNDL-MOEA all produced good solutions to the ZDT and DTLZ test problems. To quantify the differences, points are awarded to each algorithm based on how it compares to the other algorithms on each problem. Every time an algorithm performs significantly better than the other two algorithms in either convergence or diversity, it is awarded one point. If two algorithms tie, both algorithms are awarded points. If all three algorithms are the same, no points are awarded. ϵ-MOEA and SNDL-MOEA are tied with nine points each. NSGA-II runs a distant second with three points. Based on the results presented in this paper, SNDL-MOEA is shown to be competitive with ϵ-MOEA and superior to NSGA-II.

SNDL-MOEA produces populations which are both diverse and very close to Pareto-optimal with a reasonable amount of work. Efficiency preservation is employed to prevent deterioration, which allows the overall quality of the solution to steadily and consistently increase over time. The user can choose one of three different selection procedures, each of which has a different impact on selection pressure. Additionally, the user can specify the number of non-domination levels and the size of each level, which also has a direct impact on selection pressure. Furthermore, because the levels of non-domination are stored, the algorithm has the ability to go back to a very early solution and using the same foundation to solve a problem again and again, which makes it very useful for construction problems.

While we have shown SNDL-MOEA to be competitive with ϵ-MOEA based on convergence and diversity, our algorithm can suffer from a higher complexity when the size or number of levels are set high (to be exact, when $DZ^2 > N$). We are currently working on an improved version of SNDL-MOEA which exhibits a time complexity of $O(MND \cdot log(Z))$ per generation, while still maintaining the archive population in levels of nondominance keeping it suitable for construction problems. Furthermore, we plan to extend our algorithmic comparison to other test problem series such as the rotated test problem series as described in [5].

7. REFERENCES

[1] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley & Sons LTD, 2001.

[2] K. Deb, M. Mohan, and S. Mishra. A fast multi-objective evolutionary algorithm for finding well-spread pareto-optimal solutions. Technical Report 2003002, Kanpur Genetic Algorithms Laboratory, February 2003.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transaction on Evolutionary Computation*, 6(2):182–197, 2002.

[4] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable test problems for evolutionary multi-objective optimization. Technical Report 2001001, Kanpur Genetic Algorithms Laboratory, 2001.

[5] A. W. Iorio and X. Li. Rotated test problems for assessing the performance of multi-objective optimization algorithms. In M. Keijzer, editor, *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 683–690. ACM Press, 2006.

[6] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10(3), 2002.

[7] N. Phillips. Finding tight single-change covering designs with v = 20, k = 5. In *Discrete Mathematics 231*, pages 403–409. Elsevier Science, 2001.

[8] D. Preece, R. Constable, G. Zhang, J. Yucas, W. Wallis, J. McSorley, and N. Phillips. Tight single-change covering designs. *Utilitas Mathematica*, 47:55–84, 1995.

[9] S. S. Skiena. *The Algorithm Design Manual.* Springer-Verlag New York, Inc., 1998.