Simple optimization with EnergyPLAN (solar, wind, fuel share) With Domain Knowledge

March 30, 2014

1 Introduction

This report is about the chromosome representation of the optimization of energy systems. This report also presents a new mutation based on domain knowledge. The report reviews the quality indicators that are used for comparing multi-objective optimization evolutionary algorithms. The results of newly developed mutation is presented and compared with generic mutation.

2 Chromosome representation

Chromosome representation is one of the important steps in evolutionary algorithms. A good representation helps the algorithm to search the space more efficiently. In the next few sections, the problems of generic representation is discussed and a repair function is developed to deal with the problem.

2.1 Representation

In this optimization problem, we need to deal with two kinds of design variables. First four variables are the capacity of four different sources in MW (Off-shore wind, on-shore wind, PV and conventional power plant). And we have another three design variables those represent the share of three fuels (coal, oil and N. Gas) for conventional power plant. Each fuel share value represents a relative share with respect to the uses of other fuels. As an example, if both coal and oil share is 0.5, EnergyPLAN[1] tries to use same amount of coal and oil to meet the demand. But if 0.25 and 0.5 is given for coal and oil, EnergyPLAN tries to use double amount of oil than coal.

For the decision variables of the capacity of different sources, we use real representation where the maximum possible capacity of a region is given. On the other hand for fuel shares, we use real representation within a range of 0 to 1.

[1] It is assumed that EnergyPLAN is already introduced.

Table 1: Fuel Share Example

fuel	coal	Oil	N. gas
gene value	0.10	0.051	0.82
Absolute fuel share before repair	0.103	0.053	0.844
gene value after repair	0.097	0.097	0.78
Absolute fuel share after repair	0.1	0.1	0.8

2.2 Repair function

The concept of repair function [2] is vastly used in the field of evolutionary algorithm. The main concept of repair function is that repair the genes[2] so that the in-feasible gene is changed to a feasible gene. In the case of fuels mixing in a conventional power plant, most of the time, decision maker are not interested in exact percentage (e.g.: 31% coal, 23% oil and 44% natural gas). Because the exact percentage is difficult to implement. So, we want to employ the concept of step size, where decision maker can define the step size. In the case of previous example, if the step size is 5%, then the shares will be changed, 30% for coal, 25% for oil and 45% for natural gas. It is already mentioned that fuel shares in EnergyPLAN are treated as relative fuel shares with respect to other fuel shares. Each gene that represents a fuel share has a limit between 0 to 1. So, the repair function that we use here adjust the the gene values in a way so that relative shares of each fuel will be a multiple of the given step size. An Example is given in the table 1 where step size is 0.1 (i.e.: 10%). The 1$^{\text{st}}$ row of the tables is the gene values for fuel shares for a particular individual. The 2$^{\text{nd}}$ row is the absolute shares calculated using equation (1). 3 $^{\text{rd}}$ and 4$^{\text{th}}$ rows present the gene values what we expect after *repairing*.

$$fuelShare_i = \frac{geneValue_i}{\sum_{i=1}^{n} geneValue_i} \text{where } i \in Coal, Oil, N.gas \tag{1}$$

So, our proposed repair function takes a real number and repair the gene in such a way so that the gene maps with nearest valid gene value of the invalid value. The following equation describes the function.

$$rgv = \begin{cases} value + (sts - (value \bmod sts)) & \text{if } (value \bmod sts) >= sts/2, \\ value - (value \bmod sts) & \text{otherwise.} \end{cases}$$

Where rgv is the repaired gene value and sts is the step size, $value$ is the rounding gene value (e.g., rounded off two decimal place).

Next, we present an algorithm to repair the gene value according to the step size given by a decision maker algo. 1. The values represent in the fuel share genes do not represent the absolute shares of the fuels. It represent the relative

[2]Each individual may contains several genes

share of the fuels. So, a two step process can solve the problem. In the 1st step, actual fuel shares are calculated (step # 2 and 3 of algorithm 1) and adjust the absolute shares according to the given step size (Step # 4). The adjustment is done by using the equation (2.2). Then, in the 2nd step, the new relative shares are assign according to the fixed absolute share (Step # 7). Step #5 and #6 is necessary to make some minor change to the gene, if sum of all gene are not equal to 1. Sometime this may be happen because of rounding effect. For the same rounding effect, it may be possible to have a gene value greater than 1. In that case step #8 and #9 are require to scale down the gene values accordingly.

Algorithm 1 Repair Function for fuel genes

```
1: function REPAIR FUEL GENE(StepSize, fuelGenes)
2:     originalSum = sum of all values of fuelGenes
3:     normalize each gene value according to originalSum
4:     fixedGeneValues=adjust each gene value by stepSize
5:     if sum of all the scaled gene value is not equal to '1' then
6:         adjust a randomly gene value to make the sum equal to '1'
7:     newGeneValues = OrinialSum * fixedGeneValues
8:     if any new gene value is greater than 1 then
9:         Scale down each gene value by maximum gene value
10:    return newfuelGenes
```

A step-wise example is given in table.2. And positions of use of the repair function within the main evolutionary loop is shown in fig. 1.

Table 2: An step (mentioned in the algorithm 1) wise example of repair function for fuel share

Description	coal	Oil	N. gas
Original Value	0.104173	0.051276	0.98
Step # 3	0.091746	0.045159	0.863095
Step # 4	0.1	0	0.9
Step # 7	0.113545	0	1.021904
Step #8, 9	0.111111	0	1

3 Problem Specific Mutation

Mutation, one of the key operators of an EA, basically mimic the idea of mutation from nature. The presented mutation operator, specially designed to deal with the optimization problem of an energy system, is a combination of three different mutations. In a single mutation process, only one mutation is applied among three mutations. First mutation is designed to favours renewable sources i.e, increase the capacity of RE sources and use cleaner fuel, other mutation is designed to do completely the opposite (i.e.: favours conventional sources and

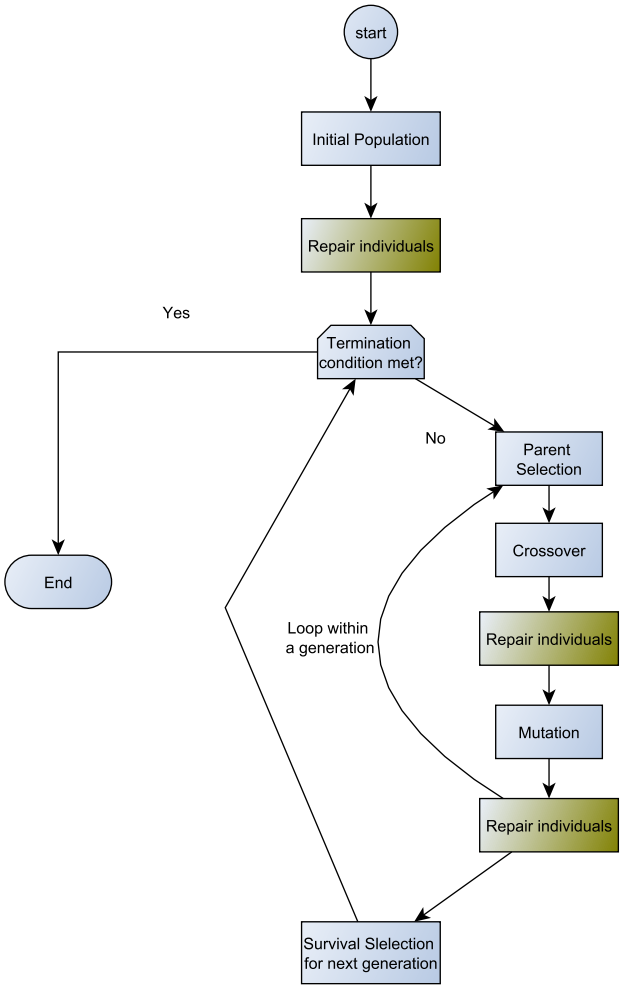


Figure 1: Position of applying repair function within the main loop of evolutionary algorithm

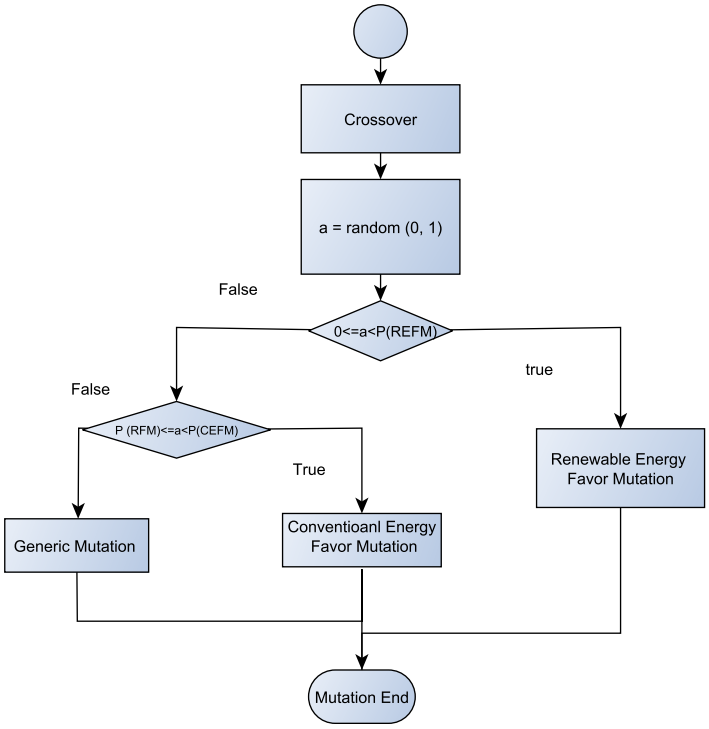


Figure 2: Mutation Flowchart.

use low cost fuel). The third mutation is a traditional/generic mutation (polynomial mutation or non-uniform mutation [4]). The reason behind the use of generic mutation is to keep the diversity of the population. Without the use of a traditional mutation, we may lead the search toward extreme solutions. In that case, we will lose interesting trade-off solutions for the decision makers. The selection of a mutation among three mutations depends on a random number drawn over uniformly distributed probability distribution. Fig. 2 illustrates the overall mutation process (P_{REFM} is the probability of applying the renewable energy favour mutation and P_{CEFM} is the probability of applying conventional energy favour mutation).

Now, we present the design of a mutation operator that can favour renewable energy sources. Traditional mutation operator makes a change to an individual so that the value of genes [3] may be decreased or increased. Most of the time, a probability distribution is used to generate a new value of a gene e.g., polynomial mutation, normal mutation [4]. In that case, the gene may fall both in the upper side (value increased) or lower side (value decreased) of the gene (anywhere within gLB to gUB, in the left figure of fig. 3, where gV, gLB and gUB represent

[3] An individual may consists of several genes.

gene value, lower bound and upper bound of the gene respectively). But to favour renewable sources, we want the mutation operator should make changes to some chosen genes so that the new gene value can only be increased (anywhere within gV to gUB in the left figure of fig. 3) or decreased. As an example, consider a gene value of an individual is **1250** and the lower and upper bound of the gene is 0 and 2000 respectively. If the gene represents the capacity of a wind source, the mutation should generate a new gene only within the range of 1250 to 2000. If we want to favour conventional sources, the mutation should produce a new gene within the range of 0 to 1250.

To design the renewable or conventional favour mutation, we use the *normal probability distribution*[9]. We will have a normal distribution centered on the gV of an individual (mean, μ is set to gV and standard deviation, σ is set to $d/3$, where $d = |gV - gUB$ or $gLB|$). The selection of σ value ensures that most of the time the new gene value will lie near to the old gene value (i.e., mean value). Then the probability density function (pdf) is truncated within an upper or lower range [11]. The red distribution in the left side of fig. 3 shows the truncated pdf distributed within the range of gLB *to* gUB. On the other hand, the blue distribution shows the truncated pdf within the range of gV *to* gUB. And the corresponding cumulative distribution function is shown in the right side of the figure. A random number is drawn from an uniform distribution within a range of 0 to 1. A new gene value is obtained using *inverse cdf or quantile function*[10]. The details of the mutation is given in the following algorithm 2.

Mutation	off-Shore Wind	On-shore Wind	PV	Conventional PP	coal	Oil	N. Gas
favour RE	✓	✓	✓	–	✗	–	✓
favour CE	✗	✗	✗	–	✓	–	✗

Table 3: Different sources that should be increased or decreased in two different mutations. ✓indicate an increase and ✗indicate a decrease, – indicate doing nothing.

The mutation that favours conventional energy is the same as the previous one except the choice of favouring decision variables. But process that is applied on the variables is identical. The table 3 shows the decision variables that are increased or decreased according to the corresponding mutation.

4 Quality Indicators for Multi-objective Optimization Problem

The way of comparing two algorithms in multi-objective optimization are completely different than single objective optimization. For a single objective optimization case, a algorithm is preferable than another, can be simple identified by measuring the value of objective function that need to be maximized or minimized. On the other hand, for a multi-objective optimization case, the

Algorithm 2 Mutation favours Renewable sources

function MUTATIONFAVOURSRES($individual, ListOfFoveredGene$)
  for each gene of an individual **do**
    if gene $\in ListOfFoveredGene$ **then**
      Define a normal distribution; $\mu = gV$, $\sigma = d_u/3$
      Truncate the distribution where distribution lower bound = gV and distribution upper bound = gUB
      Calculate the cdf of the distribution
      Draw a random number from an uniform distribution within the range of 0 to 1
      gV = inverse cdf (quantile function) for the random number
    else
      Define a normal distribution; $\mu = gV$, $\sigma = d_l/3$
      Truncate the distribution where distribution lower bound = gLB and distribution upper bound = gV
      Calculate the cdf of the distribution
      Draw a random number from an uniform distribution within the range of 0 to 1
      gV = inverse cdf (quantile function) for the random number
  return *individual*

Where gV is the gene value, gLB is the lower bound of the gene , gUB is the upper bound of the gene, $d_u = gUB - gV$, $d_l = gV - gLB$ and $ListOfFoveredGene$ is given in table 3.

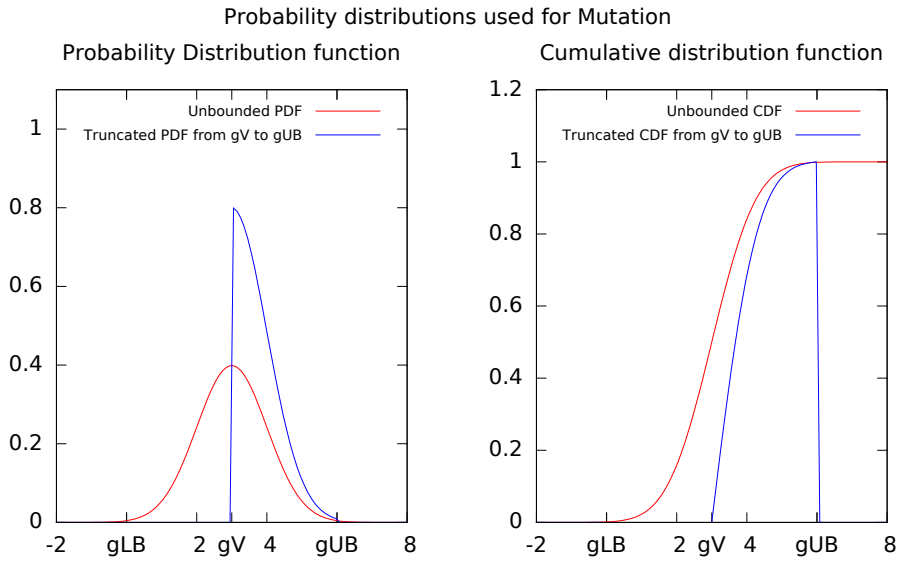


Figure 3: An example of probability density function and cumulative distribution function used in the mutation.

comparison is not so simple. Unlikely the single objective case, multi-objective optimization algorithm generates a set of non-dominated solutions called Pareto optimal set. Also the assessment criteria of an algorithm is depends the user demands. As an example, a user may be in interested on an algorithm that approximate global Pareto-front consistently, rather than an algorithm that converge to global Pareto-front but occasionally. The quality of a Pareto-Optimal set generated by a method/algorithm mainly depends on three factors [7].

1. Minimize the distance between true Pareto-front and obtained non-dominating set.

2. Maximizing the Spread of the solution in the set.

3. Maximising Number of elements in the set

Many literatures are published to capture the all three criteria into a single value. In fact, no indicator really able to capture all three issues discussed above within a single value [7]. Actually interesting enough to mention that the issues discuss above are contradictory to each other and the design of a assessment indicator itself is a multi-objective optimization problem. Therefore, it is not worthy to make a decision for an algorithm based on single assessment indicator.

Here, some of the main quality indicators found in literature are described below.

4.1 Generational Distance (GD)

The measurement of generational distance (GD) is done by calculating the distance between the elements of achieved Pareto front and known true Pareto front (TPF) [8].

$$GD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \tag{2}$$

where n is the number of element found in a achieved non-dominated set and d_i is the Euclidean distance between the points in found non-dominated set and nearest points of TPF (measured in objective space). The closer the value of GD to 0 means the obtained Pareto-front is closer to known true Pareto-front. In the case of $GD = 0$, the achieved Pareto front exactly match with TPF. In the picture 4, the red lines represent the distance between the non-dominated points and true Pareto-front points. The main problem of this quality indicator is that it value of the indicator may be misleading, if there are two different non-dominated sets need to be compared, and one set has very few points close to the TPF points and another set has many non-dominated points including the points of earlier set. The problem also exists if the two sets have same number of points. Consider, a set has 50 points those are very compact and close to each other and also close to TPF, on the other hand, another set has same number of points but they are well distributed but not so close of the TPF. In the field of multi-objective optimization distribution of non-dominated points is an important issue to be considered.

4.2 Inverted Generational Distance (IGD)

To deal with the problem of GD, a very similar quality indicator is proposed, called inverted generational distance(IGD) [8]. The concept of the indicator is opposite to GD. Here, the distance from the points of TPF to the point of obtained non-dominated set is measured and averaged.

$$GD = \frac{\sqrt{\sum_{i=1}^{k} d_i^2}}{k} \tag{3}$$

where, k is the number of points in true Pareto-front and d_i is the distance between a point of TPF and the nearest point of obtained non-dominated Pareto front. The better the quality of the solution set, the less the value of IGD will found. When $IGD = 0$, that means the solution set not only lies on the TPF, but also it covers each points of know true Pareto-front. By comparing the points of TPF with non-dominated set, IGD gets free from the problems founded in GD.

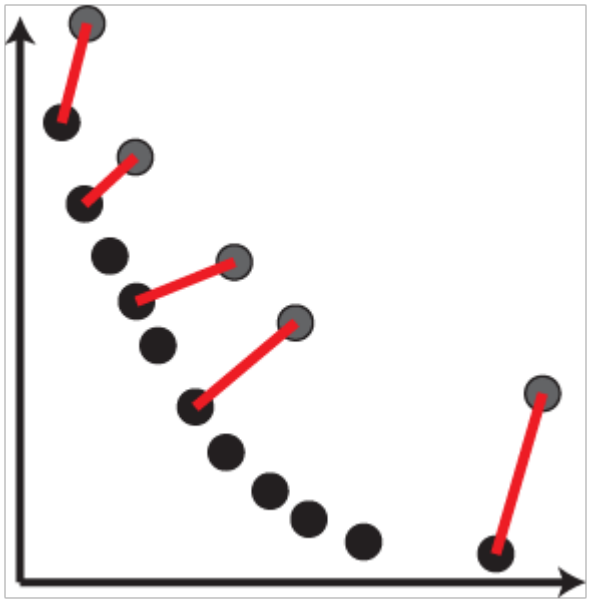


Figure 4: Generational Distance

4.3 Spread

It is highly desirable that obtained Pareto-front should be well distributed and well spread to provide all possible options to the decision maker. Deb at al. [3] introduced an indicator, called *spread*, to understand how well distributed the front is. The indicator measures the span of the achieved non-dominated solution set. The indicator is measured by taking Euclidean distance between consecutive solutions and make a average of those distances. Then another two Euclidean distances are measured from the extreme two points of TPF and boundary points of the obtained set. All the distances are measured in objective space. Finally using the following formula 4, the value of the Spread is measured the non-uniformity in the distribution.

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}} \tag{4}$$

Where, d_f and d_l are the Euclidean distances between the extreme solutions of TPF and boundary solutions of achieved set fig. 5. $\bar{d}$ is the average distances over all d_i, assuming that there are N number of solutions and $N-11$ number of consecutive distances in the obtained set. A good distribution will have d_i value close to $\bar{d}$ and d_f and d_l equal to zero, make the indicator value, Δ, as close to zero.

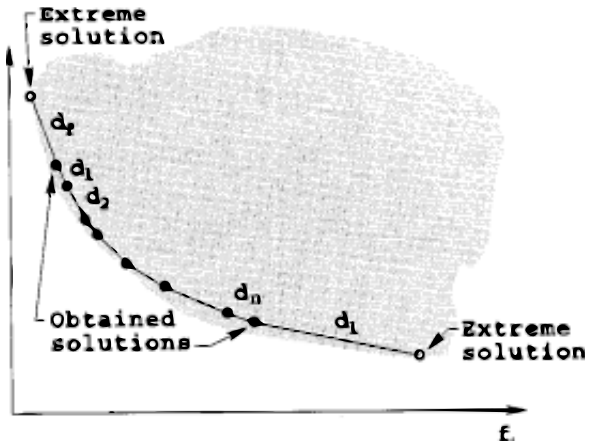


Figure 5: Calculation of Spread Indicator (taken from [3]

4.4 Generalized Spread

The indicator is introduced by Zhou A. [12]. The indicator is the extension of previous indicator of *spread* to work with more than 2 objectives.

$$\Delta(S, TPF) = \frac{\sum_{i=1}^{m} d(e_i, S) + \sum_{x \in TPF} |d(x, S) - \bar{d}|}{\sum_{i=1}^{m} d(e_i, S) + |TPF|\bar{d}} \tag{5}$$

Where S is the obtained non-dominating solution set and TPF is the true Pareto front, m is the number of objectives, $e_i \in \{e_1, \ldots, e_m\}$ are the m extreme solutions, $|TPF|$ is the number of solutions in TPF and

$$d(x, S) = \min_{y \in S, y \neq x} \parallel F(x) - F(y) \parallel^2 \tag{6}$$

$$\bar{d} = \frac{1}{|TPF|} \sum_{x \in S} d(x, S) \tag{7}$$

where $F(x)$ is the objective vector consists of all the m objective values for x.

4.5 Epsilon

This indicator is first proposed by Zitzler at al [14]. Here the concept of domination of one set to another set is considered in all objectives. Considering two sets, the concept is that how much minimum distance a bad set needs to be translated to dominate the better set. In the figure 6, the set coloured with black is better than the gray set for the minimization problem. The *red line* indicate the amount, that much the gray set needs to translate to be able to

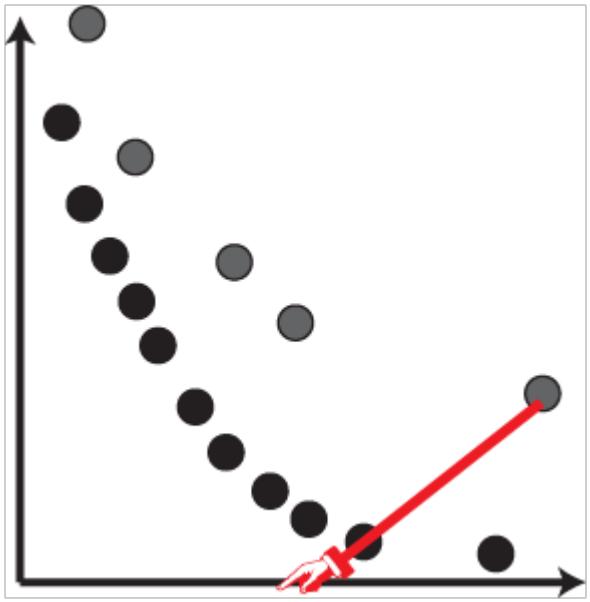


Figure 6: calculation of epsilon indicator

dominate black set. Mathematically,

$$I^1_{\epsilon_+}(S, TPF) = \inf_{\epsilon \in thbbR} \{\forall \vec{p} \in TPF \mid \exists \vec{s} \in S : \vec{s} \preceq \vec{p} + \epsilon\} \tag{8}$$

Where $\vec{s} \preceq \vec{p}$ means $\vec{s}$ dominates $\vec{p}$.

4.6 Hypervolume

This indicator/metric is also proposed by Zitzler [13] in 1999. The indicator measures the volume covered by the solutions of non-dominated set in objective space. For bi-objective optimization problem, each solution is covered by a rectangle from a reference point (for maximization problem, the reference point is generally origin, and for minimization problem reference point can be chosen from the worst solution with all population). The metric is measured by doing union of all rectangles founded from the solution of obtained non-dominated set. Mathematically,

$$HV = volume\left(\bigcup_{i=1}^{|Q|} v_i\right) \tag{9}$$

Where Q is the set of non-dominated solutions, for each solution in Q, a hypercube, v_i is created from the reference point. And the volume of the union of all hypercubes measures the value of the hypervolume. Figure 7 shows the union of all rectangles for a set.

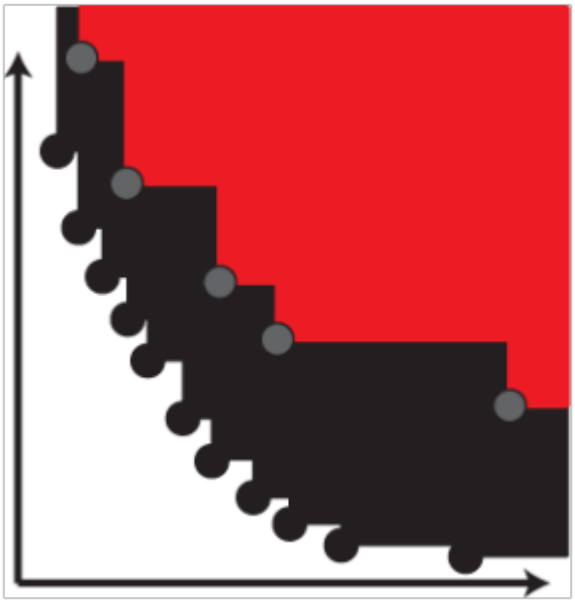


Figure 7: calculation of hyper-volume indicator

The main drawback of the indicator is, the value can be misleading if the shape of the set in objective space is concave [7]. The indicator may provide poor measurement if two algorithms may differ more than one issues discussed before sec. 4.

It is worth to mentioned that all sets need to be normalized to get those matrices.

4.7 Discussion

It is clear from above discussion that no single metric or indicator is able to provide us a clear indication about an algorithm/method. The decision makers generally want a number of uniformly distributed solutions over a tread-off surface. So, a combination of some indicators is a good choice to compare the algorithms [7]. According to the discussion, the indicators can be categorizes into three different categories [1, 6]. Fig. 8 shows the relationship of indicators with the categories.

- Convergence: GD & Epsilon
- Diversity: Spread & Generalized Spread
- Convergence & Diversity: Hypervolume & IGD

In our experiment, we will use all the indicators to compare our mutation to generic mutation.

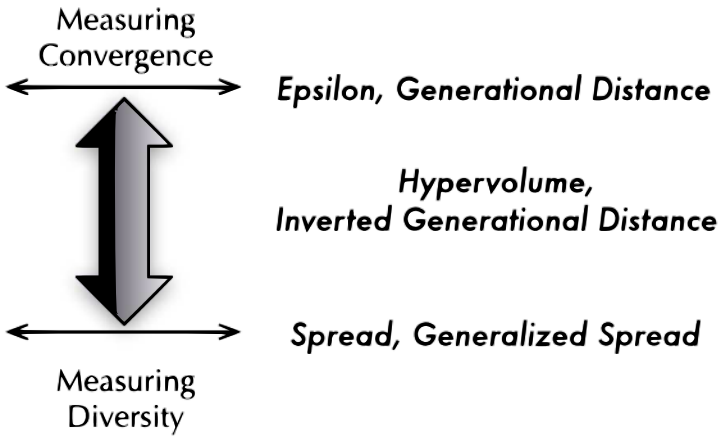


Figure 8: Indicators Category (adopted from [1])

5 Experimental Setup

To test the performance of our developed mutation, called DKMutation, we have done two kinds of experiments. The experiment is done on a dataset of Denmark. The first experiment is done using regular crossover and mutation operator (Simulated Binary Crossover and Polynomial mutation) with repair function. Next experiment is done using the same crossover but using DKMutation. To have a fair competition between two methods, we use same initial population. Therefore, two algorithms start from same initial individual. The generation of initial population is controlled by predefined seeds. Each experiment is carried out for 10 different random seeds. Therefore, we have carried out total 20 independent runs, 10 with the state-of-the-art operator and another 10 runs with our developed operator.

The following parameters are used for first experiment where we use the state-of-the-art operator.

- Population: 100
- Evolution: 5000
- crossover Probability: 0.9
- Mutation probability: 0.2
- Distribution index: 4
- Algorithm: NSGA-II
- Parent selection: Binary Tournament.

To measure the performance of the operator we need to measure the value of quality indicators describe in sec. 4. But expect hypervolume, it is require to know true Pareto-front for a problem earlier to measure the value of all other quality indicators. As the true Pareto-optimal set is not known, we construct the Pareto-front by combining the results from 20 different run. The approach is of combining the results of different runs is a well known practice in the field of evolutionary algorithm, when true Pareto-front is unknown [5].

After getting the true Pare-front, we have done the experiment again to get generational data for quality indicators. The reason behind to run the entire experiment again is that to capture the values of indicators in each generation. And to have values in each generation, we need the known Pareto-front constructed in previous experiment. Table. 5 summarize the whole experiments and purpose of the experiment.

Experiment Name	Operators	No of runs	Purpose
Exp-Regular 1	SBX & Polynomial	10	Generate true Pareto-front
Exp-DKMutation 1	SBX & DKMutation	10	
Exp-Regular 2	SBX & Polynomial	10	collect generational data for indicators
Exp-DKMutation 2	SBX & DKMutation	10	

6 Results

The results presented here divided into two parts. In the first part, the results are gather after completing of the first run. This results present the behavior of the algorithm (operator) at the end of the run. In the second phase, we try to capture the behaviors of the algorithm during the run. For this purpose, we collect the data about different indicators in each generation for different runs.

6.1 Results after completion of run

The table 4 below shows the results of 10 different run for Exp-regular 1 and Exp-DKMutation 1. The shadowed cells represents better performance for the corresponding indicator. It is noted that higher value of hypervolume represent better performance and lower value of IGD means better performance. When considering hypervolume, regular operators are manage to provide good performance 2 times more then the Domain-knowledge mutation. But in respect to IGD, DKMutation manage to outperform the regular operator 6 times.

The next table 5 shows the results of *Spread and generalized Spread indicators*. It should be noted that each of the indicators need to be minimized to present the better result. It is already mentioned earlier that *generalized spread* is a multi-dimensional version of *spread* where spread works only for two objective case. Our problem is a bi-objective problem and still we observed that the values of two indicators are different. There are mainly two reasons behind getting different values. The first reason is that *spread* indicator do

Table 4: Results for HV and IGD

	HyperVolume		IGD	
Seed	Exp. Regular 1	Exp. DKMutation 1	Exp. Regular 1	Exp. DKMutation 1
365652	0.835233	0.850046	0.003717	0.002392
455875	0.889545	0.871887	0.006704	0.001743
456545	0.889545	0.837827	0.001859	0.002354
458478	0.827764	0.840518	0.007353	0.00231
545782	0.83592	0.839544	0.002697	0.002235
547945	0.883954	0.838292	0.006959	0.002266
549235	0.841302	0.833172	0.003796	0.004549
562366	0.890897	0.840934	0.006725	0.002228
652262	0.828183	0.873807	0.007784	0.002266
981354	0.888231	0.83999	0.003411	0.002288

not consider duplicate solutions, if duplicate solutions exist in obtained Pareto-front. As spread measures the distance of successive solutions, the distance between successive solutions for duplicated solutions are considered as zero. On the other hand, as *generalized spread* considers the distance of nearest solutions whose distance is greater than zero, each duplicate solution is considered as an independent solution. The second reason is that *spread* considers gaps between all the successive solutions. On the contrary, *generalized spread* consider the smallest gap in any direction. Consider an one dimensional case, where four solutions have values of $10, 12$, 16 and 18. If we consider *spread*, there are 3 distances of $2, 4, 2$. On the other hand, if we consider *generalized spread*, the three distances are $2, 2, 2$. We do not able to specify which on the better, but with DKMutation, we have managed to outperform regular mutation 15 times within 20 times.

Table 5: Results for Spread and generalized Spread

	Spread		Generalized Spread	
Seed	Exp. Regular 1	Exp. DKMutation 1	Exp. Regular 1	Exp. DKMutation 1
365652	0.657685	0.773247	0.573008755	0.558847165
455875	0.675538	0.785794	0.677625004	0.569152299
456545	0.675538	0.641181	0.689310642	0.385132845
458478	0.685045	0.648291	0.715928387	0.433595939
545782	0.619087	0.586206	0.544114878	0.419384069
547945	0.827126	0.544189	0.816414887	0.433969669
549235	0.644144	0.648021	0.552251426	0.636379722
562366	0.719552	0.515242	0.704807006	0.45772625
652262	0.70084	0.964505	0.697327808	0.587781616
981354	0.967737	0.618592	0.606944707	0.412998768

The following table 6 shows the results of *generational distance* and *epsilon* indicators. The results show that it is a tie for epsilon, but in generational

distance regular operator outperform DKMutation. Most probably the reason it that DKMutation is able to produce more diverse solution along the tread-off surface but the state-of-the-art operator produces results near to true Pareto front.

Table 6: Results for GD and Epsilon

	Generational distance		Epsilon	
Seed	Exp. Regular 1	Exp. DKMutation 1	Exp. Regular 1	Exp. DKMutation 1
365652	0.005615	0.00485	1573	754
455875	0.000641	0.002132	631	5.696
456545	0.000811	0.006319	596	694
458478	0.002539	0.005513	1842	689
545782	0.00578	0.004896	1432	797
547945	0.001582	0.005812	604	1008
549235	0.005617	0.005303	1051	1688
562366	0.000489	0.006442	629	652
652262	0.002267	0.003865	1841	689
981354	0.002626	0.005827	683	702

6.2 Results during run

In this section we are going to present generational results for two most important quality indicators, *hypervolume and inverse generational distance(IGD)*. To understand the overall behavior, we make an average and normalized the data for each generation using (10), (11) .

$$HV_{gen_i} = \frac{\sum_{j=1}^{k} HV_j^i}{k} * 100 \tag{10}$$

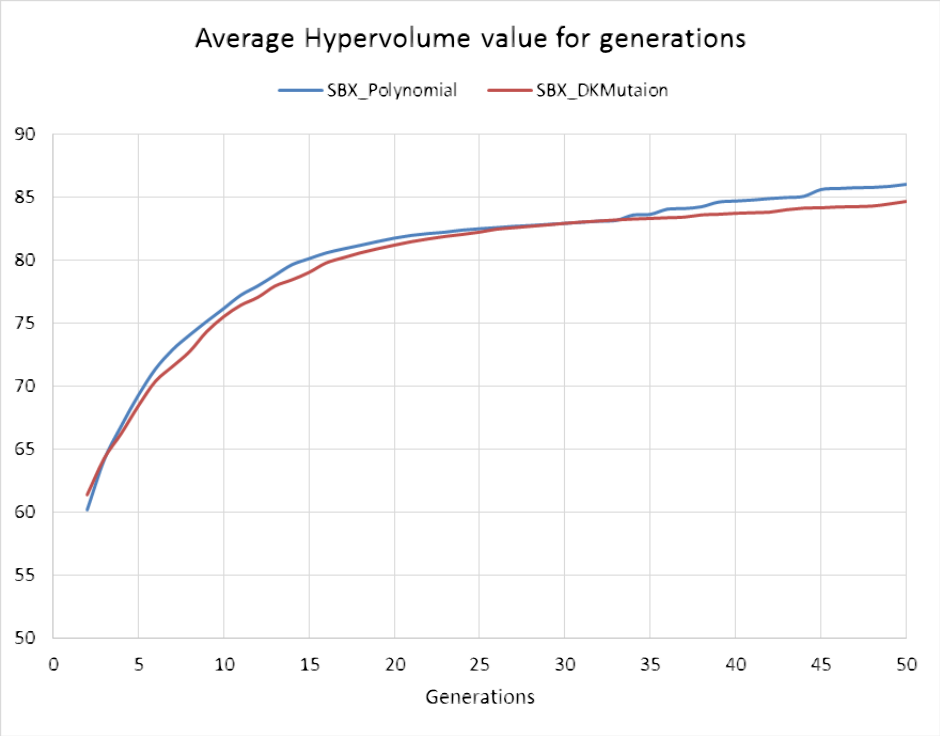
Where i is the generation number, k is the total number of runs with different seeds. In our case, k is equal to 10. HV_j^i means hypervolume of i^{th} generation of j^{th} run. Using the equation we measure the average value of hyperpervolume in a scale of 100 for regular operator and DKMutation.

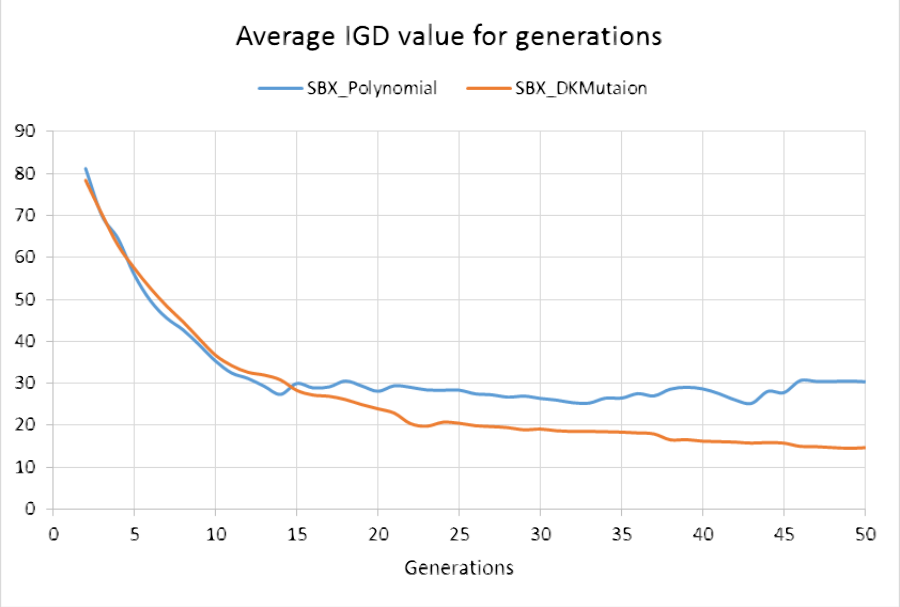
For IGD, it is expected that the value will be close to zero.

$$IGD_{gen_i} = \frac{\sum_{j=1}^{k} IGD_j^i}{k * IGD_{max}} * 100 \tag{11}$$

Where i is the generation number, k is the total number of runs with different seeds. IGD_{max} is the maximum IGD found in all different runs.

From the chart of hypervolume, it is clear that our developed mutation closely follow the regular mutation from the beginning until end.


Average Hypervolume value for generations
SBX_Polynomial
SBX_DKMutaion
90
85
80
75
70
65
60
55
50
0
5
10
15
20
25
30
35
40
45
50
Generations


Average IGD value for generations
SBX_Polynomial
SBX_DKMutaion
90
80
70
60
50
40
30
20
10
0
0
5
10
15
20
25
30
35
40
45
50
Generations

On the other hand, the chart of IGD is similar until 15 generation. After 15 generations, the curve represents DKMutation continue goes downward, but the other curve that represents polynomial mutation stabilizes.

7 Future work

The concept of favouring renewable energy sources, clean energy sources and favouring the conventional energy sources, low cost fuel can be applied to crossover operator. In the next experiment we may use the same distribution that is used in polynomial mutation instead of normal distribution. Another area we want to explore is smart initialization. State-of-the-art algorithm initializes individuals randomly from a uniformly distributed probability distribution. But we know that a renewable sources is better than another if the distribution of the source (distribution of wind or solar on every hour) match the distribution of electricity demand of the region. Therefor, we can design an initialization process that prioritize the better renewable sources.

References

[1] Guillermo Molina Arribere. Tcnicas de optimizacin para redes de sensores (optimization techniques for wireless sensor networks).

[2] P.P. Bonissone, R. Subbu, N. Eklund, and T.R. Kiehl. Evolutionary algorithms + domain knowledge = real-world evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 10(3):256–280, Jun 2006.

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

[4] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001.

[5] J.J. Durillo, A.J. Nebro, and E. Alba. The jmetal framework for multiobjective optimization: Design and architecture. *2010 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, Jul 2010.

[6] Siwei JIang and Jie Zhang. Consistencies or contradictions of performance metrics in multiobjective optimization? *IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews)*, 2014.

[7] Ruhul Sarker and Carlos A. Coello Coello. Assessment methodologies for multiobjective evolutionary algorithms. In *Evolutionary Optimization*, number 48 in International Series in Operations Research & Management Science, pages 177–195. Springer US, January 2002.

[8] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective evolutionary algorithm research: A history and analysis. Technical report, 1998.

[9] Wikipedia. Normal distribution — wikipedia, the free encyclopedia, 2013.

[10] Wikipedia. Quantile function — wikipedia, the free encyclopedia, 2013.

[11] Wikipedia. Truncated distribution — wikipedia, the free encyclopedia, 2013.

[12] Aimin Zhou, Yaochu Jin, Qingfu Zhang, B. Sendhoff, and E. Tsang. Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. In *IEEE Congress on Evolutionary Computation, 2006. CEC 2006*, pages 892–899, 2006.

[13] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[14] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, and V.G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.