PURPOSE

Basic GA, as the first learning experience of actually building the program based on a real problem, also of processing the data
FLOW

Setting

➢ Design Variable 32
➢ Constraint 22
➢ Population 50
➢ Number of Evaluations 9950
FIRST GENERATION

Randomized generation for the seed, between \([0, 1]\) as a standardized value of each variables

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.114849</td>
<td>0.808894</td>
<td>0.677372</td>
<td>...</td>
<td>0.28848</td>
<td>0.056884</td>
</tr>
</tbody>
</table>
**PARENT SELECTION**

**Tournament Method**

Randomly selected 2 individual

-Individual with lower objective value are chosen

- repeated enough times to get a full population
CROSSOVER

Simulated Binary Crossover – SBX (Deb and Goyal, 1996)

- Symmetric -> Avoid any bias towards particular parent
- When parents values are distant, distant children values possible
  When parents values are close, distant children values unlikely
  → Converging Search
CROSSOVER

Following the equation of:

\[
\tilde{\beta} = \begin{cases} 
\frac{1}{(2u)^{n+1}}, & \text{if } u \leq 0.5 \\
\frac{1}{(2(1-u))^{n+1}}, & \text{otherwise.} 
\end{cases}
\]

\[
x_i^{(1,t+1)} = 0.5 \left[ (1 + \tilde{\beta}) x_i^{(1,t)} + (1 - \tilde{\beta}) x_i^{(2,t)} \right], \\
x_i^{(2,t+1)} = 0.5 \left[ (1 - \tilde{\beta}) x_i^{(1,t)} + (1 + \tilde{\beta}) x_i^{(2,t)} \right],
\]

Parameter

\[ n = 15 \]

Steps:

➢ Randomize \( u \)
➢ Get \( \tilde{\beta} \)
➢ Calculate children value
MUTATION

The Mutated Value is calculated following the probability function defined by Deb and Goyal (1996) that depends on the perturbance factor $\delta$:

$$P(\delta) = 0.5(n + 1)(1 - |\delta|)^n$$

In accordance to the following equation,

$$\bar{\delta} = \begin{cases} (2u)^{\frac{1}{n+1}} - 1, & \text{if } u < 0.5 \\ 1 - [2(1 - u)]^{\frac{1}{n+1}}, & \text{if } u \geq 0.5. \end{cases}$$

$$c = p + \bar{\delta}\Delta_{max}$$

Parameters:

$$p_m = \frac{1}{32}, \quad \Delta_{max} = 1, \quad n = 15$$

Steps:

- Randomize $u$
- Get $\bar{\delta}$
- Calculate Mutated Value $c$
Penalty points to objective value
-> Harder to be selected

\[ f' = f + \alpha \times \Omega(x) \]

\[ \Omega(x) = \sum_{i=1}^{22} w_i, \]

\[ w_i = \begin{cases} 
0, & g_i(x) > 0 \\
|g_i(x)|, & g_i(x) \leq 0 
\end{cases} \]

Parameter: \( \alpha = 1 \)
From the solution that satisfied all the constraint conditions,
The best solution is of $f = 0.062543495$

Found on Generation 99

The median solution is of

- Only from satisfying trials: $f = 0.069217208$
- From all 21 Trials: $f = 0.078854218$

Satisfied Individual’s $f$ in the Best Trial
CONCLUSION

This project helped me build my foundation:
- GA and Technical Programming Skills

Further study:
- Try making adjustment and research the effects
  - Ideas: Change the n from small to big along the loop so that the search can be even more converging
- Try the multi-objective techniques
Infeasibility Driven Evolutionary Algorithm with Bump Hunting

Third Evolutionary Computation Competition 2019

Category: Single Objective

Group Number: S02

Kamrul Hasan Rahi
Research Masters Student

Supervisors: Dr. Hemant Kumar Singh and Professor Tapabrata Ray
Multidisciplinary Design Optimization (MDO) Group
School of Engineering and Information Technology (SEIT)
The University of New South Wales, Australia.
IDEA with Bump Hunting

Constrained single objective wind turbine design optimization problem.

- Population based stochastic optimization algorithms are preferred since the objective and constraint functions may be highly nonlinear with functional/slope discontinuity.
- To deal with constraints, strategies often prefer a feasible solution over infeasible ones. They are referred as Feasibility First constraint handling strategies e.g. NSGA-II.

However, preserving marginally infeasible solutions during the course of search and actively recombining them can result in faster rate of convergence over feasibility first strategies. **Infeasibility Driven Evolutionary Algorithm (IDEA)**[1] is one such scheme known for its superior performance on constrained optimization problems.

Smart reduction in variable space is yet another scheme that can offer significant benefits to the process of recombination. **Bump Hunting**[2] is an approach that can be used to identify potential regions of interest.

The proposed approach employs IDEA with original variable bounds until 50% of the computational budget is exhausted. Thereafter, it identifies reduced variable bounds using Bump Hunting and runs IDEA using these reduced bounds for the remaining computational budget.

Infeasibility Driven Evolutionary Algorithm (IDEA)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population Size $N$</td>
<td>100</td>
</tr>
<tr>
<td>Crossover probability</td>
<td>1.0</td>
</tr>
<tr>
<td>Mutation probability</td>
<td>0.1</td>
</tr>
<tr>
<td>Distribution index: Crossover</td>
<td>20</td>
</tr>
<tr>
<td>Distribution index: Mutation</td>
<td>20</td>
</tr>
<tr>
<td>Infeasibility ration ($\alpha$)</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Steps:
1. Generate $N$ initial solutions using LHS sampling.
2. Evaluate these $N$ solutions.
3. Create $N$ offspring using SBX and PM.
4. Evaluate these $N$ offspring solutions.
5. Select $N$ solutions from these $N$ parent and $N$ offspring solutions using IDEA ranking.

Collection of $2N$ solutions

Feasibility First ranking

IDEA ranking

Marginally infeasible solutions

Feasible space

Infeasible space

Constraint
Bump Hunting (BH) for Space Reduction: Illustration

Lower bound = max(min(all variables from all boxes), global LB)
Upper bound = min(max(all variables from all boxes), global UB)

Promising hyper-rectangles identified using best 50% solutions (Minimization sense).
IDEA with BH for Wind Turbine Design Problem

Potential regions of interest. Bounds identified using results of 21 runs of NSGA-II, IDEA for 10,000 and 30,000 functions evaluations.

Volume=1.4429E-13

Reduced variable bounds identified after 5,000 function evaluations

Top 50% of solutions (with constraint violation less than 1e-3 or feasible solutions) were used to identify the reduced variable bounds.

Volume=2.9170E-11

Overshoots and Missed spaces

Low height of red and blue bar is preferred.
Thank you for your attention

http://www.mdolab.net/
Applying Differential Evolution to Wind Turbine Optimization in Considering Constraint Violations

Hori Takato  Uchitane Takeshi  (Aichi Institute of Technology)

Evolutionary Computing Symposium 2019 @ Minamiawaji sity
Wind turbine optimization constraints

There are many constraints on the problem of Wind turbine optimization.

The constraint violation rate is depending on the constraint conditions. Particular, 3, 13, 17, 18, 19, and 20th constraints have many constraint violations.
Three steps to applying Differential Evolution to eliminate constraint violations

1. Identify constraints with a higher constraint violation rate from random value vectors.
2. Add a penalty corresponding to the constraint violation rate and minimize the number of constraint violations by Differential Evolution with $m$ generations.
3. Perform Differential Evolution in $k$ generations with individuals with few constraint violations as initial individuals.

num of individual $\times (n+m+k)=10000$

i. If $n$ and $m$ are too large, fewer generations to minimize objective function.
ii. If $n$ and $m$ are too small, the objective function is optimized with constraint violations remaining.
## Result

<table>
<thead>
<tr>
<th>F</th>
<th>CR</th>
<th>num of individuals</th>
<th>n</th>
<th>m</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.2</td>
<td>50</td>
<td>5</td>
<td>25</td>
<td>170</td>
</tr>
<tr>
<td>0.5</td>
<td>0.2</td>
<td>25</td>
<td>10</td>
<td>20</td>
<td>370</td>
</tr>
<tr>
<td>0.3</td>
<td>0.2</td>
<td>25</td>
<td>10</td>
<td>20</td>
<td>370</td>
</tr>
<tr>
<td>0.3</td>
<td>0.5</td>
<td>25</td>
<td>10</td>
<td>20</td>
<td>370</td>
</tr>
</tbody>
</table>

Best: 0.055301  
Var.: 1.9632E-07  
Ave.: 0.056092571  
Med.: 0.05605
Mutation based on Variance of Individuals in IDE

P4-01  Ryukoku University
○ Yuta Furukawa  Keiko Ono  Kenta Matsuo
IDE
Differential Evolution With an Individual-Dependent Mechanism

◆ A kind of differential evolution method proposed by Lixin Tang et al.
◆ Efficient search is possible by IDP setting to set parameters based on individuals’ fitness and IDM strategy to set an appreciate search direction and its range.
**Mutation: IDP Setting**

◆ Sort population based on fitness

\[ F \rightarrow F_o = (\text{randn} \left( \frac{o}{NP} \right), 0.1) \quad \text{good} \quad (o = 1, 2, \ldots, NP) \]

\[ \text{bad} \]

◆ Individuals with good fitness are set to smaller F and each search range is reduced.

◆ Individuals with bad fitness are set to larger F and its search range is expanded.

\[ \text{CR}\rightarrow \text{CR}_i = (\text{randn} \left( \frac{i}{NP} \right), 0.1) \quad (i = 1, 2, \ldots, NP) \]

◆ Individuals with good fitness are set to smaller CR to inherit more information from parents.

◆ Individuals with bad fitness have a larger CR to inherit more information from mutant individuals.
Mutation: IDM Strategy

Solution Space
Population: NP
Superior: S \( \frac{ps}{NP} \)
Inferior: I \( \frac{1-ps}{NP} \)

\[ v_i = x_0 + \begin{cases} 
    F_o \cdot (x_{r1} - x_0) + F_o \cdot (x_{r2} - d_{r3}), & o \in S \\
    F_o \cdot (x_{\text{better}} - x_0) + F_o \cdot (x_{r2} - d_{r3}), & o \in I 
\end{cases} \]

S: Superior set
I: Inferior set
\( x_{\text{better}} \): Individuals randomly selected from the superior set

\[ d_{r3}^j = \begin{cases} 
    L^j + \text{rand}^j_i(0,1) \cdot (U^j - L^j), & \text{if} (\text{rand}^j_i(0,1) < P_d) \\
    x_{r3}^j, & \text{otherwise} 
\end{cases} \]

Generate new individuals randomly or selected from a current population

\textit{ps}: Determine the composition of \( d_{r3} \) and the proportion of individuals in the superior and inferior sets
When the target problem is high-dimensional problem

Introduce the Self-Organizing Map (SOM)

\[ NP = \{ x_1, x_2, \ldots, x_i \} \]
\[ x_i = \{ x_i^1, x_i^2, \ldots, x_i^j \} \rightarrow \{ x_i^1, x_i^2 \} \]

Normalize the Solution space

\[ 0 \leq x_i^1 \leq 1, \ 0 \leq x_i^2 \leq 1 \]
In the previous method, individual diversity is not considered, because ps calculates only based on the number of generations.

The proposed method adopts a population clustering method (Dirichlet process gaussian mixture model) to capture a landscape by using ps.

Ps utilizes a standard deviation for each cluster and determine according to the following formula:

\[ ps = 0.1 + 0.9 \times (1 - \bar{\sigma}(C_n)^{100}) \]

\( \bar{\sigma}(C_n) \): Standard deviation of each cluster
Crossover

◆ Insert an individual randomly

\[
FOR \ j = 1 \ to \ D \\
\]
\[
\quad u_{i,g}^j = \begin{cases} 
\quad v_{i,g}^j & \text{if } rand_i^j(0,1) \leq CR_i \ or \ j = j_{rand} \\
\quad x_{i,g}^j & \text{otherwise}
\end{cases}
\]

◆ Select crossover points at random

\[
IF(u_{i,g}^j < L \ or \ u_{i,g}^j > U) \\
\quad u_{i,g}^j = L + rand_i^j(0,1) \cdot (U - L)
\]

◆ If the value of the \( j \)-th dimension is out of solution space, a next individual generated at random in the solution space instead of pulling back
Optimize individuals in two stages

- Update an individual by comparing the number of elements that satisfy the constraints.
- If all constraints are satisfied and its fitness is better than before, an individual is updated.

![Diagram showing fitness and constraint relationship](image-url)
The proposed method doesn’t need a parameter fitting method
Algorithm Presentation (s05, m05)

Jernej Zupančič, Aljoša Vodopija, Tea Tušar, Erik Dovgan, Bogdan Filipič

Jožef Stefan Institute (JSI), Ljubljana, Slovenia
Single-objective optimization algorithm (s05)

- Algorithm: jDE (Python Package: pygmo, function: saDE)
- DoE method: Latin hypercube sampling
- Constraint handling technique (CHT): dynamic penalty function
  \[ \bar{f}(x) = f(x) + (ct)^\alpha \sum_i v_i(x) \]
- Parameters and configuration:
  - Population size: 20
  - No. of generations: 500
  - DE variant: rand/1/bin
  - CHT parameters: \( c = 1.0, \alpha = 1.0 \)
Evolution Computation Competition 2019
Single objective optimization

Jun-ichi Kushida (Hiroshima City University)

• **Method**: DE with $\varepsilon$ constraint method and pareto approach
• Optimize constraint violation $\Phi(x)$ and $f(x)$ separately

In the two-objective space ($f$, $\Phi$ space)

- **Individual with Pareto rank = 1**
  → Prioritize improvement of $\Phi(x)$
  (Search feasible solution)

- **Individuals with Pareto rank $\neq 1$**
  → Preserve diversity
ε epsilon level comparison

Relax constraints by ε and compare parent and child

If either constraint violation > ε, compare by constraint violation

If both violation ≤ ε, compare by function value

When comparing \( x_1 \) and \( x_2 \)
Control of $\epsilon$ level

$\epsilon$ value in generation $t$

$$\epsilon(t) = \begin{cases} 
\epsilon(0)(1 - \frac{t}{T_c})^{cp}, & 0 < t < T_c \\
0, & t \geq T_c 
\end{cases}$$

Initial $\epsilon$ value: $\epsilon(0) = \Phi(x_{\theta})$

$x_{\theta}$ is the $\theta$th individual among the initial individuals sorted in ascending order by constraint violation ($\theta = r \times NP$)

Determine by the violation of the initial population

Gradually decreases

Set $\epsilon = 0$ at $T_c$ generation
Setup

<table>
<thead>
<tr>
<th>Parameter</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>NP</td>
</tr>
<tr>
<td></td>
<td>$50 \rightarrow 10$</td>
</tr>
<tr>
<td></td>
<td>(after $T_c$ gen.)</td>
</tr>
<tr>
<td>$T_c$</td>
<td>$140^{th}$ generation</td>
</tr>
<tr>
<td>$r$</td>
<td>0.1</td>
</tr>
<tr>
<td>$cp$</td>
<td>3</td>
</tr>
</tbody>
</table>

- Strategy of $\varepsilon$ DE: rand/1 /bin

Mutant vector $\boldsymbol{v} = \boldsymbol{x}_{r1} + F_t (\boldsymbol{x}_{r2} - \boldsymbol{x}_{r3})$

$\boldsymbol{x}_{r1}$ is randomly selected from individuals with pareto rank = 1

Parameter of $t$-th generation

- $F_t = 0.6 - 0.3 \cdot \frac{\varepsilon(t)}{\varepsilon(0)}$
- $CR_t = 0.1 + 0.9 \cdot \frac{\varepsilon(t)}{\varepsilon(0)}$

Gradually increase $0.3 \rightarrow 0.6$

Gradually decrease $0.9 \rightarrow 0.1$
Convergence of populations in $f, \phi$ space

Early stage of search

Later generations

At the end of the search
Experimental result

Average, maximum, median, and average values over 21 trials

<table>
<thead>
<tr>
<th>min</th>
<th>0.054334</th>
</tr>
</thead>
<tbody>
<tr>
<td>max</td>
<td>0.057439</td>
</tr>
<tr>
<td>median</td>
<td>0.055819</td>
</tr>
<tr>
<td>average</td>
<td>0.055805</td>
</tr>
</tbody>
</table>

Transition of the best solution for the trial of median
Complexity Reduction Fast Moving Natural Evolution Strategy

Number: s08
Takuya kato, Kazuki Kamata and Isao Ono
Tokyo Institute of Technology
Natural Evolution Strategy (NES) [Wierstra 08]

• Minimizes the expected objective function value:
  \[ J(\theta) = \int f(x)p(x|\theta)dx. \]
  - \(f(x)\) : objective function
  - \(p(x|\theta)\) : probability distribution
  - \(\theta\) : parameter of probability distribution

• Natural gradient descent method [Amari 85]:
  \[ \theta \leftarrow \theta - \eta F^{-1}(\theta)\nabla_{\theta} J(\theta). \]
  - \(\eta\) : learning rate
  - \(F(\theta)\) : Fisher's information matrix
    \[ F(\theta) = E_x[\nabla_{\theta} \ln p(x|\theta) (\nabla_{\theta} \ln p(x|\theta))^T] \]
Natural Evolution Strategy (NES) [Wierstra 08]

- Algorithm when \( p(x|\theta) \) is a normal distribution

1. Initialize the generation \( g = 0 \) and the probability distribution \( N(m^{(g)}, C^{(g)}) \).
2. Make \( \lambda \) individuals \( \{x_i\}_{i=1}^{\lambda} \) according to \( N(m^{(g)}, C^{(g)}) \).
3. Evaluate \( x_i \), and update \( m \) and \( C \) as follows.

\[
\begin{align*}
    m &\leftarrow m - \eta \sum_{i=1}^{\lambda} \frac{f(x_i)}{\lambda} (x_i - m) \\
    C &\leftarrow C - \eta \sum_{i=1}^{\lambda} \frac{f(x_i)}{\lambda} (x_i - m)(x_i - m)^T - C
\end{align*}
\]

4. If a stop condition is not met, \( g = g + 1 \) and go to step 2.
Natural Evolution Strategy (NES) [Wierstra 08]

- Fitness shaping [Wierstra 08]
  - Makes the algorithm invariant under monotonically increasing transformation.
  - Replaces \(-\frac{f(x_i)}{\lambda}\) with a normalized weight \(w_i\).

\[
m \leftarrow m + \eta \sum_{i=1}^{\lambda} w_i (x_i - m)
\]
\[
C \leftarrow C - \eta \sum_{i=1}^{\lambda} w_i \left( (x_i - m)(x_i - m)^T - C \right)
\]

◆ The better \(f(x_i)\) is, the larger \(w_i\) is.

\[
w_{i}^{\text{rank}} = \frac{\hat{w}_i}{\sum_{j=1}^{\lambda} \hat{w}_j} - \frac{1}{\lambda'}
\]
\[
\hat{w}_i = \max \left( 0, \ln \left( \frac{\lambda}{2} + 1 \right) - \ln(i_{\text{ord}}) \right), \text{where } i_{\text{ord}} \text{ is rank relating to } f(x).
\]
Uses a normal distribution with a restricted covariance matrix as $p(x|\theta)$.

- Covariance matrix: $\sigma^2 D (I + vv^T) D$
  [Akimoto 14]
  - $D$ : diagonal matrix
  - $v$ : vector
  - $\sigma$ : scalar
- Mean vector: $m$
- Parameter of normal distribution: $\theta = (m, \sigma, v, D)$

- Reduces time and space complexity.

Algorithm:
1. Initialize each variables.
2. Generate $\lambda$ samples: $x_i \sim p(x|\theta)$.
3. Sort $x_i$ with a preference order operator $<_p$.
4. Switch learning rates of $\sigma, v$ and $D$ according to search situation.
5. Update $\theta = (m, \sigma, v, D)$ using natural gradient.
6. If the stopping condition is met, stop, otherwise $g \leftarrow g + 1$ then go to step 2.
CR-FM-NES for Wind Turbine Design Optimization Problem

• Parameters:
  - Sample size: 48
  - Others: default

• Preference order operator: $<_p$
  1. Upper and lower constraint violation
  2. Problem constraint violation
  3. Objective function

• Solutions which do not meet upper and lower constraint are not simulated.

<table>
<thead>
<tr>
<th></th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
<th>6th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective function</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Problem constraint violation</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Upper and lower constraint violation</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
The 3rd Evolutionary Computation Competition

University of Tsukuba, M1 Yuta Kobayashi
University of Tsukuba, Claus Aranha
Problem Exploration

![Histogram of NotSatisfyCount across Constraint Names]

- **Constraint 1**
- **Constraint 2**
- **Constraint 3**
- **Constraint 4**
- **Constraint 5**
- **Constraint 6**
- **Constraint 7**
- **Constraint 8**
- **Constraint 9**
- **Constraint 10**
- **Constraint 11**
- **Constraint 12**
- **Constraint 13**
- **Constraint 14**
- **Constraint 15**
- **Constraint 16**
- **Constraint 17**
- **Constraint 18**
- **Constraint 19**
- **Constraint 20**
- **Constraint 21**
- **Constraint 22**

Colors range from -0.8 (blue) to 0.8 (red), indicating the intensity of constraint violation.
Problem Exploration

NotSatisfyCount

Constraint_Name

3-13

NotSatisfyCount

Constraint_Name

3-13-20
2-step Search

- finding non-constrained initial population:
  \[ z_i = \frac{c_i - \mu_i}{\sigma_i} \]
  \[ f = \sum_{i=1}^{22} w_i z_i \]
  \( w_i \): weight of constraint
  \( c_i \): value of constraint
  \( \mu_i \): mean of constraint
  \( \sigma_i \): std of constraint

- Initialize by Latin Hyper Cube Sampling
- Search spaces that satisfy constraint by using DE
- Initialize populations by using solutions that satisfy all constraints
- Search better solutions by MOEA/D-DE

8000 evaluations
2000 evaluations
Pre-experiment - how to find constraint weight

<table>
<thead>
<tr>
<th>Constraints</th>
<th>not_satisfy_propotion</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Constraint1</td>
<td>0.54192</td>
</tr>
<tr>
<td>#Constraint2</td>
<td>0.31853</td>
</tr>
<tr>
<td>#Constraint3</td>
<td>0.94864</td>
</tr>
<tr>
<td>#Constraint4</td>
<td>0.26685</td>
</tr>
<tr>
<td>#Constraint5</td>
<td>0.20023</td>
</tr>
<tr>
<td>#Constraint6</td>
<td>0.7698</td>
</tr>
<tr>
<td>#Constraint7</td>
<td>0.24959</td>
</tr>
<tr>
<td>#Constraint8</td>
<td>0.13028</td>
</tr>
<tr>
<td>#Constraint9</td>
<td>0.31362</td>
</tr>
<tr>
<td>#Constraint10</td>
<td>0.19650</td>
</tr>
<tr>
<td>#Constraint11</td>
<td>0.52900</td>
</tr>
<tr>
<td>#Constraint12</td>
<td>0.0</td>
</tr>
<tr>
<td>#Constraint13</td>
<td>0.93847</td>
</tr>
<tr>
<td>#Constraint14</td>
<td>0.58237</td>
</tr>
<tr>
<td>#Constraint15</td>
<td>0.52680</td>
</tr>
<tr>
<td>#Constraint16</td>
<td>0.29401</td>
</tr>
<tr>
<td>#Constraint17</td>
<td>0.89310</td>
</tr>
<tr>
<td>#Constraint18</td>
<td>0.74274</td>
</tr>
<tr>
<td>#Constraint19</td>
<td>0.74027</td>
</tr>
<tr>
<td>#Constraint20</td>
<td>0.94039</td>
</tr>
<tr>
<td>#Constraint21</td>
<td>0.0</td>
</tr>
<tr>
<td>#Constraint22</td>
<td>0.40636</td>
</tr>
</tbody>
</table>

![Graph showing NotSatisfyCount vs Constraint_Name](image)
## Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>DE</th>
<th>MOEAD-DE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population</td>
<td>100</td>
<td>Population</td>
</tr>
<tr>
<td>Crossover rate, C</td>
<td>1.0</td>
<td>Crossover rate, C</td>
</tr>
<tr>
<td>Scaling Factor, F</td>
<td>0.5</td>
<td>Scaling Factor</td>
</tr>
<tr>
<td>index parameter, $\eta_m$</td>
<td>20</td>
<td>index parameter, $\eta_m$</td>
</tr>
<tr>
<td>Mutation Rate</td>
<td>$1/32$</td>
<td>Mutation Rate</td>
</tr>
<tr>
<td>Decomposition method</td>
<td></td>
<td>SLD</td>
</tr>
<tr>
<td>Scalar aggregation function</td>
<td></td>
<td>Weighted Tchebycheff</td>
</tr>
</tbody>
</table>
## Result

<table>
<thead>
<tr>
<th>Trials</th>
<th>HyperVolume</th>
<th>satisfycount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.791</td>
<td>696</td>
</tr>
<tr>
<td>2</td>
<td>2.337</td>
<td>586</td>
</tr>
<tr>
<td>3</td>
<td>3.246</td>
<td>716</td>
</tr>
<tr>
<td>4</td>
<td>3.126</td>
<td>609</td>
</tr>
<tr>
<td>5</td>
<td>2.431</td>
<td>585</td>
</tr>
<tr>
<td>6</td>
<td>3.177</td>
<td>570</td>
</tr>
<tr>
<td>7</td>
<td>2.833</td>
<td>648</td>
</tr>
<tr>
<td>8</td>
<td>3.301</td>
<td>603</td>
</tr>
<tr>
<td>9</td>
<td>3.276</td>
<td>506</td>
</tr>
<tr>
<td>10</td>
<td>2.923</td>
<td>782</td>
</tr>
<tr>
<td>11</td>
<td>3.004</td>
<td>590</td>
</tr>
<tr>
<td>12</td>
<td>3.302</td>
<td>579</td>
</tr>
<tr>
<td>13</td>
<td>2.863</td>
<td>713</td>
</tr>
<tr>
<td>14</td>
<td>3.200</td>
<td>536</td>
</tr>
<tr>
<td>15</td>
<td>2.963</td>
<td>617</td>
</tr>
<tr>
<td>16</td>
<td>2.995</td>
<td>540</td>
</tr>
<tr>
<td>17</td>
<td>3.029</td>
<td>593</td>
</tr>
<tr>
<td>18</td>
<td>2.452</td>
<td>689</td>
</tr>
<tr>
<td>19</td>
<td>2.949</td>
<td>610</td>
</tr>
<tr>
<td>20</td>
<td>2.402</td>
<td>615</td>
</tr>
<tr>
<td>21</td>
<td>2.772</td>
<td>638</td>
</tr>
</tbody>
</table>

![Graph showing the comparison between satisfied and not satisfied objectives](image)
Evolutionary Computation Symposium Competition 2019
Application of CM2T

Yuto Fujii¹, Taiki Hanada¹, Yiping Liu¹, Naoki Masuyama¹, Yusuke Nojima¹, and Hisao Ishibuchi²
¹Osaka Prefecture University
²Southern University of Science and Technology
1. Wind Turbine Optimization Problem is a severe constrained problem.
   - Utilizing various infeasible solutions

2. The reference point for HV calculation is far from the true nadir point.
   - Modifying initial weight vectors to obtain solutions near the edges of the Pareto front
Introduction

Characteristics of Competition

1. Wind Turbine Optimization Problem is a severe constrained problem.
   - Utilizing various infeasible solutions

2. The reference point for HV calculation is far from the true nadir point.
   - Modifying initial weight vectors to obtain solutions near the edges of the Pareto front
CM2T (Constrained Multi-objective to Two-objective) is a constrained multi-objective evolutionary algorithm based on decomposition.

**Characteristic of CM2T**

- Solutions are evaluated and selected in each transformed two-objective optimization problem.
- Minimize Objective 1: Scalarizing function value
- Minimize Objective 2: Overall constraint violation value

CM2T proposed paper
Solutions corresponding to each vector are evaluated in the transformed two-objective (scalarizing function and overall constraint violation) space.

$w$ : Weight vector  \( g(w, x) \) : Scalarizing function  
\( CV \) : Overall constraint violation

Minimize $f_1$
Minimize $f_2$
Minimize $g(w_1, x)$
Minimize $g(w_6, x)$
Minimize $CV$
Solutions corresponding to each vector are evaluated in the transformed two-objective (scalarizing function and overall constraint violation) space.

- $w$: Weight vector
- $g(w, x)$: Scalarizing function
- $CV$: Overall constraint violation

**Minimize** $f_1$

**Minimize** $f_2$

**Minimize** $g(w_1, x)$

**Minimize** $g(w_6, x)$

**Minimize** $CV$
Solutions corresponding to each vector are evaluated in the transformed two-objective (scalarizing function and overall constraint violation) space.

$w$: Weight vector  \quad g(w, x):$ Scalarizing function

$CV$: Overall constraint violation
Solutions in each subpopulation are evaluated and selected by the Pareto ranking and the crowding distance in the transformed objective space.
1. Wind Turbine Optimization Problem is a severe constrained problem. 
   - Utilizing various infeasible solutions

2. The reference point for HV calculation is far from the true nadir point. 
   - Modifying initial weight vectors to obtain solutions near the edges of the Pareto front
Modifying Initial Weight Vectors

Motivation

In the previous study [Ishibuchi et al. GECCO2017]

When the reference point is far from the nadir point, solutions at the edges of the Pareto front have larger HV contribution.

Circle size = HV Contribution

= Nadir point
Reference point
Far
Modifying Initial Weight Vectors

Our Method

To search for solutions near the edges of the Pareto front, we raised the initial weight vectors to the power of $\alpha (\alpha > 1)$.

Before: $w = (w_1, w_2, \ldots, w_n)$  \quad After: $w' = (w_1^\alpha, w_2^\alpha, \ldots, w_n^\alpha)$

$w$ : Weight vector  \quad $\alpha > 1$ : Pareto front  \quad $\circ$ : Solution

Initial weight vectors ($\alpha = 1$)
Parameters

Population size : 210
Subpopulation size : 21
Scalarizing function : Normalized Tchebycheff
Crossover : SBX (DI: 20)
Probability of crossover : 1.0
Mutation : Polynomial Mutation (DI: 20)
Probability of mutation : 1 / 32
The power of weight vectors : 4

The parameter tuning is not applied.
Wind Turbine Design Optimization using a Many-objective Evolutionary Algorithm with a Single Set of Reference Vectors

Ahsanul Habib
Research Associate
Multidisciplinary Design Optimization (MDO) Group
School of Engineering and Information Technology (SEIT)
University of New South Wales (UNSW), Canberra, Australia

14 December, 2019
1. Scope of the Problem
2. Proposed Algorithm
3. Numerical Experiments
A multi-objective wind turbine design optimization problem as part of the Evolutionary Computation Competition 2019. The problem involves 5 objectives, 32 continuous variables and 22 constraints, which are evaluated using WISDEM and OpenMDAO tools. The design optimization problem needs to be solved with a computational budget of 10,000 function evaluations.
Overview

Different steps of SRMEA.

- **Generate:** Reference vectors, $W$
- **Initialize:** Population, $P_{\text{init}}$ where $|P_{\text{init}}| = 11D - 1$
- **Update:** $FE$ and Archive
- **Update:** $Z_{\min}$, $Z_{\max}$ and adapt $W$
- **Generate:** $|W|$ offsprings, $C$ and evaluate only the unique ones
- **Combine:** $R = P + C$
- **Update:** $Z_{\min}$, $Z_{\max}$ and adapt $W$
- **Environmental Selection:** Select $P$ solutions as parents of next generation

**Output:** $ND$ solutions from Archive
Generation of Reference Vectors

$W$, Reference vectors originating from $z_{min}$. 

Direction vector
Unit Hyperplane
Reference point
The size of the initial population is predefined by the user ($N_{init}$).

The solutions are initialized within the variable bounds using

Latin hypercube sampling (LHS).
Adaptation of Reference Vectors

The update scheme for the $i^{th}$ reference vector is presented below:

$$W_i = \frac{W_{0,i} \odot (z_{\text{max}} - z_{\text{min}})}{||W_{0,i} \odot (z_{\text{max}} - z_{\text{min}})||}; \quad i = 1, \ldots N_W$$

Adaptation of $W$ for a 2 objective problem.
In each generation, $N_W$ offspring solutions are generated using simulated binary crossover (SBX) and differential evolution (DE) operator with an equal probability.

For DE, the first parent is from the sorted list of parents and the other two parents are randomly chosen.

Each offspring solution undergoes polynomial mutation (PM).
Assignment Operation

Assignment of solutions to $W$. 

Wind Turbine Design Optimization using SRMEA
Environmental Selection

Selecting solutions from active reference vectors.

Using inactive reference vectors later to select more solutions.
A ND-based constraint handling method is employed here to maintain the solution diversity during the search of feasible region(s). Two possible scenarios can occur:

- **All parent+offspring solutions are infeasible:** The solutions are normalized according to $z_{min}$ and $z_{max}$. Then, a non-domination (ND) sort is performed taking the CV of the solutions as an objective and the ED of the solutions (calculated with the normalized objective values) to the origin as the second objective. Finally, the population is sorted based on the ranks obtained from this ND sort algorithm.

- **Some solutions are feasible:** If some solutions are feasible in the combined population (number of feasible solutions is less than $N_W$), they are automatically selected and the rest of the infeasible solutions are sorted according to the ND-based scheme mentioned in the previous step.
Parameter Settings

- Number of initial solutions, $N_{init} = 11D - 1$.
- Maximum number of function evaluation, $FE_{max} = 10,000$.
- Uniform spacing on unit hyperplane in NBI method, $H = 7$.
- Number of reference vectors, $N_W = 330$.
- Population size, $N = N_W$.
- Number of independent runs: 21.
- Crossover ($\rho_c$) and mutation probability ($\rho_m$): 1.0 and $1/D$.
- Distribution index of crossover ($\eta_c$) and mutation ($\eta_m$): 30 and 20.
- Crossover probability (CR) and differential weight (F) for DE: 1.0 and 0.5.
- Performance metrics: Hypervolume.
Hypervolume statistics for feasible non-dominated solutions obtained in 21 independent runs are as follows:

<table>
<thead>
<tr>
<th>Worst</th>
<th>Mean</th>
<th>Best</th>
<th>Median</th>
<th>Std</th>
<th>Success Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.9745</td>
<td>5.1195</td>
<td>5.2525</td>
<td>5.1378</td>
<td>0.0742</td>
<td>100</td>
</tr>
</tbody>
</table>

A success rate of 100% means that all 21 independent runs were able to obtain feasible solutions.
Median Hypervolume Convergence

Hypervolume convergence for the median run.
Thank you for listening!
Algorithm Presentation (s05, m05)

Jernej Zupančič, Aljoša Vodopija, Tea Tušar, Erik Dovgan, Bogdan Filipič

Jožef Stefan Institute (JSI), Ljubljana, Slovenia
Multi-objective optimization algorithm (m05)

- Algorithm: Modified version of NSGA-II capable of including various CHTs (our implementation in Python)
- DoE method: Latin hypercube sampling
- CHT: dynamic penalty function
  \[
  \tilde{f}(x) = f(x) + (ct)^\alpha \sum_i v_i(x)
  \]
- Parameters:
  - Population size: 48
  - No. of generations: 208
  - Crossover probability: 1.0
  - Mutation probability: 0.15
  - CHT parameters: \(c = 0.5, \alpha = 2.0\)
Algorithm overview for EC competition 2019

Hayato Noguchi (Ritsumeikan University)
Tomohiro Harada (Tokyo Metropolitan University)
Overview

- Use $I_{SDE} +$ as evaluation indicator
- Give the constraint processing to conventional $I_{SDE} +$
- Change Simulated Binary crossover (SBX) to Differential Evolution operator (DE) as crossover method
- Exclude similar solutions
\[ I_{SDE} + \]

1. Assign the total fitness of all \( m \) objectives to each individual
2. Give maximum \( I_{SDE} + \) value to the individual with the minimum fitness
3. Compare each \( I_{SDE} + \) values of remaining individuals
4. Take the top \( N \) individuals to the next generation

If individual \( p \) is feasible …

\[
I_{SDE} + (p) = \min_{q \in \mathbb{P}_{feasible}, p \neq q} \{ \text{dist}(p, q'_1), \text{dist}(p, q'_2), \ldots, \text{dist}(p, q'_{N_{feasible}-1}) \}
\]

\[
q'(j) = \begin{cases} p(j) & q(j) < p(j) \\ q(j) & \text{otherwise} \end{cases} \quad j \in (1, 2, \ldots, m)
\]

Convergence to the optimal Pareto front can be expected while keeping the diversity of the population.
Constraint Processing

- Calculate violations based on constraints

\[
violation(p) = \sum_{i=1}^{k} \max \left\{ 0, -\frac{g_k(p)}{g_{k,max}} \right\} \geq 0 \quad (g_k: \text{constraint function})
\]

\[
cI_{SDE} + (p) = \begin{cases} 
I_{SDE} + (p) & \text{(If } p \text{ is feasible)} \\
-violation(p) & \text{(If } p \text{ is infeasible)}
\end{cases}
\]

The higher \(violation(p)\) is, the less likely the individual \(p\) remains in the next generation.
Other Improvements

• Replace Simulated Binary crossover (SBX) to Differential Evolution operator (DE) as crossover method

\[
p_j^i = \begin{cases} 
  p_j^{r_1} + F \times (p_j^{r_2} - p_j^{r_3}) & \text{if } \text{rand}_j(0,1) \leq CR \lor j = j_{\text{rand}} \\
  p_j^i & \text{otherwise}
\end{cases}
\]

\[i \in (1,2,...,N)\]
\[j \in (1,2,...,\text{Individual Size})\]

• Exclude similar solutions

\[\min_{a \in A} \{\text{dist}(p, a)\} < \text{eps} \quad (A: \text{All evaluated solutions})\]

If many similar solutions exist, the threshold of exclusion \(\text{eps}\) is reduced gradually

\[\text{eps} = 0.5 \times \text{eps} \]
Parameters

Population size: 100
Number of generations: 100
Crossover probability (CR): 0.9
Scaling factor (F): 0.5
Mutation probability: 1.0
Threshold of excluding similar solutions: 0.01
Thank you for listening.