Title Frame

Progress Seminar

Multi objective feature selection using Non dominated Sorting based evolutionary algorithm

Ved Prakash
Roll No. 1903016
Department of Computer Science & Engineering
Indian Institute of Information Technology Guwahati

January 30, 2024

Agenda

Introduction: Feature Subset Selection

Feature Subset Selection Problem

To get optimal subset of relevant features which represents original feature set with increased classification accuracy

- Feature Selection is one of the important and critical data pre-processing step for building the statistical or machine learning models
- Feature Selection Algorithms reduces the dimensionality of the data by removing the redundant and irrelevant features
- It also results into lesser learning time, improved model interpretability, reduced overfitting and improved classification performance

Introduction: Approaches for Feature Subset Selection

- Among all the feature subset selection methods, which have been proposed in the literature, broadly they can be categorized as filter, wrapper and embedded methods based on the way how method evaluate the feature–
  - In flter methods, feature are evaluated according to their information value or statistical measures
  - In wrapper methods, feature subset as a whole is evaluated using some learning algorithm
  - Embedded methods are part of the algorithm for example Lasso Regression

Feature Subset Selection as Multi Objective Optimization Problem

- Feature Subset Selection methods have goal of minimizing the classification error by selecting less number of features.
- less number of feature is important constraint as it helps to reduce the curse of dimensionality and hence reduces the chance of over fitting by removing the redundant and irrelevant features and improve generalization on new data, it also helps to reduce the cost of acquiring the data.
- Hence, feature subset selection problem can be formulated as multi-objective optimization problem with two objectives, *i.e.*, to reduce the classification error and to reduce the number of attributes in the feature subset.
- In general, these two goals are conflicting to each other, hence the the objective is to balance the trade-off between these two conflicting goals.

Steps of Basic Evolutionary Algorithm

Initial Population:

	Credit History	Purpose of Credit	Age	Average Balance in Saving Account	present resident since - years	Nature of job
Individual 1	1	1	1	0	0	0
Individual 2	0	0	0	0	0	1
Individual 3	1	1	0	0	0	0
Individual 4	0	1	0	1	1	0
Individual 5	0	0	0	1	1	1
Individual 6	0	0	1	1	1	0

Fitness Calculation, Selection and Cross-Over

	Credit History	Purpose of Credit	Age	Average Balance in Saving Account	present resident since - years	Nature of job
Individual 1	1	1	1	0	0	0
Individual 4	0	1	0	1	1	0

	Credit History	Purpose of Credit	Age	Average Balance in Saving Account	present resident since - years	Nature of job
Offspring 1	1	1	1	1	1	0
Offspring 2	0	1	0	0	0	0

Mutation

	Credit History	Purpose of Credit	Age	Average Balance in Saving Account	present resident since - years	Nature of job
Offspring 1	1	1	0	1	1	0

Multi-Objective Optimization and Multi-objective Evolutionary Algorithm

- Multi-Objective Optimization Problems (MOOP) refer to real-world problems that require simultaneous optimization of multiple objectives or conflicting performance measures.
- Non-dominated sorting is usually defined for points belonging to a M-dimensional space $\mathbb{R}^M$. In evolutionary computation, each such point is an objective vector associated with a certain solution of an optimization problem, or an individual is called a population. Let $\mathbb{P} = \{\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_N\}$ be a population of size N. A point $p \in \mathbb{P}$ contains a vector of M objectives and is represented as $p = \langle p_1, p_2, \ldots, p_M \rangle$.

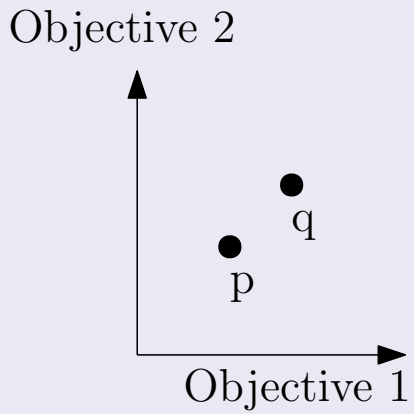
Dominance

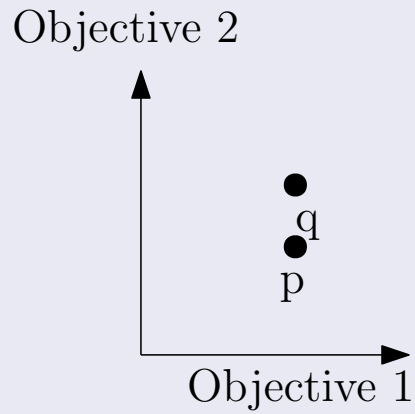
Definition (**Dominance**)

A point $p = \langle p_1, p_2, \ldots, p_M \rangle$ *dominates a point* $q = \langle q_1, q_2, \ldots, q_M \rangle$, *written as* $p \prec q$, *if both the following conditions are satisfied:*

- *Point p is better than or equal to point q for all the objectives*
  Mathematically, $\forall m \in \{1, 2, \ldots, M\}\ p_m \leq q_m$
- *Point p is better than point q for at-least one objective*
  Mathematically, $\exists m \in \{1, 2, \ldots, M\}\ p_m < q_m$

Example



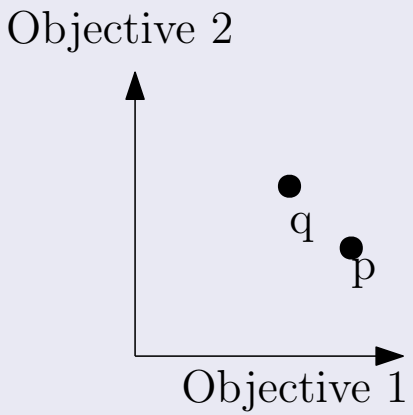


Non Dominance

Definition (**Non-Dominance**)

If neither $p \prec q$ nor $q \prec p$ holds, we say that p and q are non-dominating.

Example

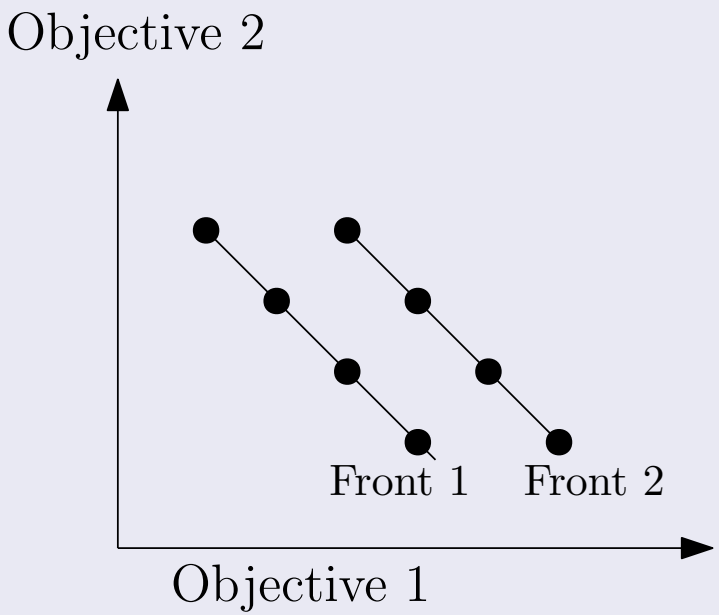


Front

Definition (**Front**)

A set of points which are non-dominated with each other constitutes a front.

Example



Problem Statement

Binary Representation of Solution

In this work, binary strings have been used to represent a solution. A solution p can be represented as,

$$p = (p_1, p_2, \ldots, p_m) \, ; p_j \in \{0, 1\} \tag{1}$$

In Equation (1), $s_j = 0$ represents that j^{th} feature is not included in the solution s and $s_j = 1$ indicates that it has been included.

Problem Statement

Multi Objective Feature Selection Problem Statement

A multiobjective feature selection problem can be formulated as,

$$\underset{(err(s),|s|)}{\text{minimize}}\ p = (p_1, p_2, \ldots, p_m)\,; p_j \in \{0,1\}; j = 1, 2, \ldots, m \quad (2)$$

$|s|$ is number of features in solution s.

MOFS-BDE Algorithm

- The MOFS-BDE algorithm was prposed by Zhang *et. al.* [3] in their work "Binary differential evolution with self-learning for multi-objective feature selection"
- The algorithm uses the mutation strategy such that, it selects the best one among the three random vectors as the base vector, and employs the difference between the remaining two vectors as a mutation probability to be used on the base vector to generate a mutation vector for the next crossover operator. It is illustrated in Equations (3), (4).

$$C_i = \begin{cases} \sigma & X_{best}(t) \prec X_i(t) \\ min(1, F.(X_{r1}(t) \oplus X_{r2}(t)) + \sigma) & \text{Otherwise} \end{cases} \quad (3)$$

$$v_{i,j}t = \begin{cases} X_{best}(t) & C_{i,j} < rand \\ 1 - X_{best}(t) & \text{Otherwise} \end{cases} \quad (4)$$

Approaches for Mutation

- The cross-over strategy is similar to standard differential evolution algorithm and it can be illustrated as follows:

$$u_{i,j}(t) = \begin{cases} v_{i,j}(t) & \text{if } U(0,1) \leq CR \text{ or } j = j_{rand} \\ x_{i,j}(t) & \text{Otherwise} \end{cases} \quad (5)$$

Area of enhancement in Existing Algorithms: Mutatiom Strategy

Out of three randomly selected vector, the two vectors other than best vector among three, have been used to calculate the mutation probability C_i as mentioned in Equation (6). This mutation probability C_i has been used to create the mutation vector v_i as illustrated in equation

$$C_i = \begin{cases} \sigma & \text{if } X_{best}(t) \prec X_i(t) \\ min(1, F.(X_{r1}(t) \oplus X_{r2}(t)) + \sigma) & \text{Otherwise} \end{cases} \tag{6}$$

$$v_{i,j}t = \begin{cases} X_{best}(t) & C_{i,j} < rand \\ 1 - X_{best}(t) & \text{Otherwise} \end{cases} \tag{7}$$

Analyzing the mutation strategy of MOFS-BDE algorithm on some of the broadly possible scenarios

- If both bits are same then $X_{r1}(t) \oplus X_{r2}(t)$ will become zero. then (6) will become

$$C_i = \begin{cases} \sigma & \text{if } X_{best}(t) \prec X_i(t) \\ min(1, F.(0) + \sigma) & \text{Otherwise} \end{cases}$$

i.e.,

$$C_i = \begin{cases} \sigma & \text{if } X_{best}(t) \prec X_i(t) \\ min(1, \sigma) & \text{Otherwise} \end{cases}$$

Analyzing the mutation strategy of MOFS-BDE algorithm on some of the broadly possible scenarios

Since, σ is a very small turbulance coefficinet and much much less than 1 so $min(1, \sigma) = \sigma$

$$C_i = \begin{cases} \sigma & \text{if } X_{best}(t) \prec X_i(t) \\ \sigma & \text{Otherwise} \end{cases}$$

So, if both bits are same then c_i will be σ. In this case, the condition $C_{i,j} < rand$ will most of the time be true. as random number generator theoretically generates values between 0 and 1 with equal distribution. and since σ is closer to zero then it is more chance that, rand will be greater than σ and hence more chance that $v_{i,j}t$ in Equation (7) will have same bit value as $X_{best}(t)$.

Approaches for Mutation

- **If both bits are different** If both bits are different then $X_{r1}(t) \oplus X_{r2}(t)$ will become 1. then (6) will become

$$C_i = \begin{cases} \sigma & \text{if } X_{best}(t) \prec X_i(t) \\ min(1, F.(1) + \sigma) & \text{Otherwise} \end{cases}$$

i.e.,

$$C_i = \begin{cases} \sigma & \text{if } X_{best}(t) \prec X_i(t) \\ min(1, F + \sigma) & \text{Otherwise} \end{cases}$$

So, if both bits are different then C_i depends on scaling factor F and turbulance co-efficient σ. Higher the value of F and σ (as σ is very small turbulance co-efficient so it mainly depends on F), higher will be the value of C_i. and hence more chance that $v_{i,j}t$ in Equation (7) will have bit value opposite to $X_{best}(t)$.

The issue with mutation approach of MOFS-BDE algorithm

- So, the issue with approach in Equation (6) are following
  - if $X_{best}(t) \prec X_i(t)$ hen it is very very high chance that the mutant vector will borrow bit from the $X_{best}t$.
  - otherwise
    - if X_{r1} and X_{r2} have same bits then it is very very high chance that the mutant vector will borrow bit from the $X_{best}t$.
    - if X_{r1} and X_{r2} have different bits and learning rate F is small, then also it is very high chance that, the bits of mutant vector will be borrowed from $X_{best}t$.
    - Hence the true spirit of differential evolution, i..e, to utilize the difference in population is not being utilized in these scenarios and hence it might impact the global exploration capability of the algorithm.

Proposed Algorithm

- We have proposed a 'Modified Mutation Strategy' to the algorithm

Modified Mutation Strategy

- we have proposed a weighted mutation strategy specially suitable for the **binary** differential evolution algorithm. The proposed mutation strategy is mentioned in Equation

$$\hat{X}_i = \underbrace{\mu(floor((X_{r1} + X_{r2} + X_{r3})/1.5)}_{initial} + \underbrace{(1-\mu)(X_{best\_among\_3}.int((X_{r2} \odot X_{r3})) + int(\neg X_{best\_among\_3}).int((X_{r2} \otimes X_{r3}))}_{later} \quad (8)$$

$$\hat{X}_i(t) = \begin{cases} 0 & \text{if } X_i(t) < 0.5 \\ 1 & \text{Otherwise} \end{cases} \quad (9)$$

- $\mu = \exp(-\sqrt{g})$ and μ is a monotonically decreasing function with increase in g, *i.e.*, generation.

Modified Mutation Strategy

- **During Initial Generation:** During initial generation, when μ will be closer to 1 then $(1 - \mu)$ will be closer to 0 and hence second part of the equation will have negligible impact on the outcome and hence it can be said that, during initial generation, outcome will be guided by first part of the equation, *i.e.*, $\mu(floor((X_{r1} + X_{r2} + X_{r3})/1.5)$.
  - If none of the feature subset X_{r1}, X_{r2}, X_{r3} consist this feature then chance is less that mutation vector consist this feature.
  - But, if any two of X_{r1}, X_{r2}, X_{r3} consists this feature, then chances will be more that, during initial generation, mutation vector will also have this feature.
  - If all of the feature subset X_{r1}, X_{r2}, X_{r3} consists this feature then chances are even stronger that mutation vector consist this feature.

Modified Mutation Strategy

- **During Later Generations:** During later generations, when μ will be closer to 0 then $(1-\mu)$ will be closer to 1 and hence first part of the equation will have negligible impact on the outcome and hence it can be said that, during later generations, outcome will be guided by second part of the equation, $(1-\mu)(X_{best\_among\_3}.int((X_{r2} \odot X_{r3})) + int(\neg X_{best\_among\_3}).int((X_{r2} \otimes X_{r3}))$

$$\hat{X}_i = \begin{cases} (1-\mu)(X_{best\_among\_3}.int((X_{r2} \odot X_{r3}))) & \text{if } X_{\text{best\_among\_3}} = 1 \\ (1-\mu)\left(int(\neg X_{best\_among\_3})\right).int(X_{r2} \otimes X_{r3})) & \text{Otherwise} \end{cases}$$

Modified Mutation Strategy

$$\hat{X}_i = \begin{cases} (1-\mu)(int((X_{r2} \odot X_{r3}))) & \text{if } X_{\text{best\_among\_3}} = 1 \\ (1-\mu)\,(int(X_{r2} \otimes X_{r3})) & \text{Otherwise} \end{cases}$$

Since, μ will closing to zero during later generation, so $1-\mu$ will be closer to 1, hence,

$$\hat{X}_i = \begin{cases} X_{r2} \odot X_{r3} & \text{if } X_{\text{best\_among\_3}} = 1 \\ X_{r2} \otimes X_{r3} & \text{Otherwise} \end{cases}$$

Modified Mutation Strategy

- So, during later generations,
  - If $X_{\text{best\_among\_3}}$ consists this feature and any of the other two vectors selected for generating mutation vector consists this feature then chances are more that mutation vector will have this feature.
  - Also, if $X_{\text{best\_among\_3}}$ doesn't consists this feature but other two vectors selected for generating mutation vector consists this feature then there is chance that mutation vector will have this feature.
- Hence, it can be said that, the mutation operator purposed in Equation (9) is designed in such a way that, it tries to make balance between global exploration and local exploitation.

Application on Banking Dataset for credit risk prediction

- To assess the credit risk for each loans, banks calculates some of the credit risks kpi (key performance indicator) including probability of default, exposure at default, loss given default etc.
- These prediction are done using various modeling techniques. One of the very critical data pre-processing techniques to build the credit risk models are, feature subset selection

Application on Banking Dataset for credit risk prediction

- In our knowledge, the most of the research. which have been done for solving feature subset selection problem for banking and credit risk assessment using evolutionary computation are focused on one objective, *i.e,* model accuracy
- In this work, we have formulated feature subset selection for banking and credit risk assessment, as multi objective optimization problem. The two objectives which have been considered are, to minimize the error rate and to minimize the cardinality of the feature subset to be used for creating the model.

Dataset description

- We have used German credit data [2]
- The dataset includes 1,000 observations and 20 attributes
- There are 7 numerical and 13 categorical attributes
- Numerical attributes are age, credit amount, duration, installment rate, present residence since, number of existing credits at this bank, and number of people being liable to provide maintenance.
- Categorical attributes are gender, job, housing, savings account/bonds, checking account, credit history, purpose, personal status and sex, other debtors, property, other installment plans, telephone, and foreign worker.
- Each observation is labeled as either "good" or "bad" credit risk, based on whether the applicant paid back the loan in full or defaulted on it.

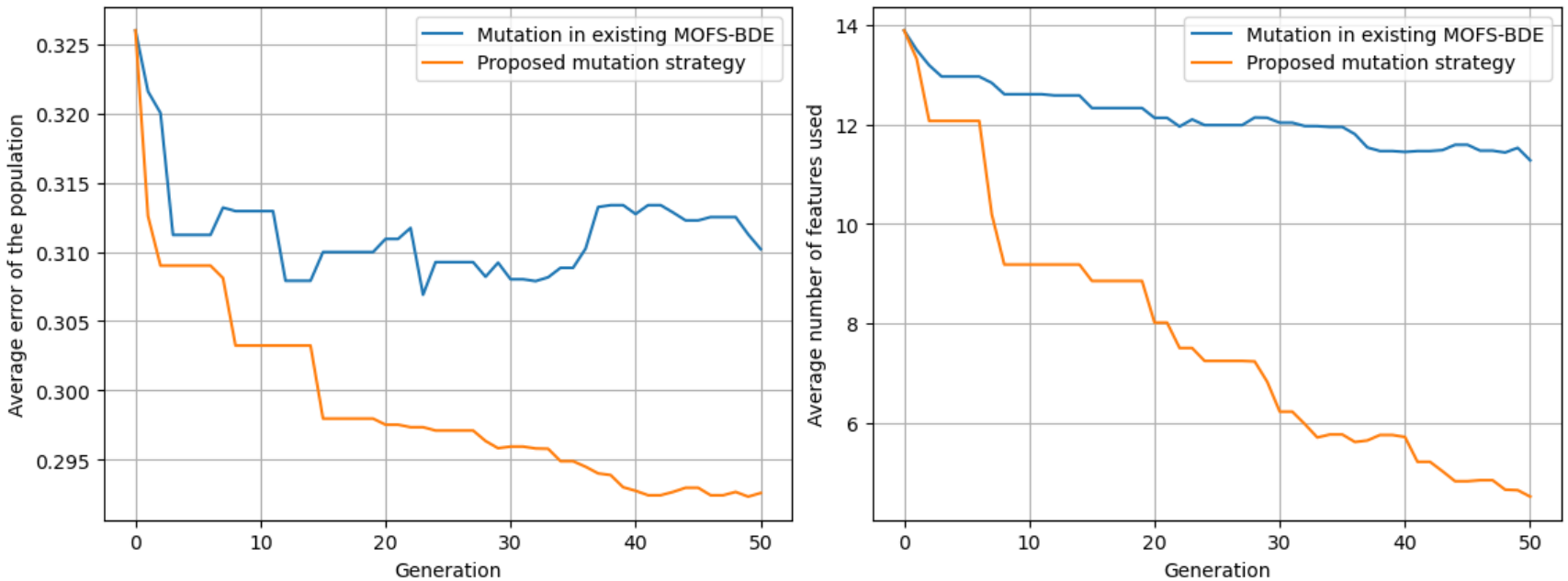
Experimental Setup

- We have performed one hot encoding on the categorical features of the German credit dataset [2].
- While doing so, to address the dummy variable trap, we have created one less dummy feature than the total number of categories available for that particular feature.
- The total number of features in the final dataset was 27. We split the dataset into two parts, we trained model on one and tested the performance on other.

Experimental Setup

- We have used multi objective feature selection binary differential evolution (MOFS-BDE) algorithm for the purpose of feature subset selection.
- The fitness function that we have used for the MOFS-BDE algorithm is, XGBoost algorithm, and the two objectives to the MOFS-BDE algorithms are "to minimize the errror rate" and "to minimize the cardinality" of the feature subset.
- We have compared performance of MOFS-BDE algorithm with existing mutation strategy versus modified mutation strategy.

Median and best accuracy of the population for each generation



(a) (b)

Figure: Average error and average number of features for each generation.

Conclusion

- The results shows that, modified MOFS-BDE algorithm with our proposed mutation strategy performs better than the existing MOFS-BDE algorithm.
- The modified algorithm is not only minizing the error but it is also fastly reducing the feature alnogwith geneation and hence this algorithm will be efficient in datasets with more number of features.

Limitation

- The limitation of any evolutionary based algorithm including MOFS-BDE algorithm is that, they take more time to converge and are computationally slow.

Future Work

- During the next progress seminar,
  - The comparison of old and proposed mutation strategy will be done using **Hypervolume**. Hypervolume is a metric used in multi-objective optimization problems to assess the quality of a set of solutions.
  - Also, the experiment will be performed on more datasets.

Thank You !

Questions ?

References I

[1] Manolis Georgioudakis and Vagelis Plevris.
A Comparative Study of Differential Evolution Variants in Constrained Structural Optimization.
Frontiers in Built Environment, 6, 07 2020.

[2] Hans Hofmann.
Statlog (German Credit Data).
UCI Machine Learning Repository, 1994.
DOI: 10.24432/C5NC77.

[3] Yong Zhang, Dun wei Gong, Xiao zhi Gao, Tian Tian, and Xiao yan Sun.
Binary Differential Evolution with Self-learning for Multi-objective Feature Selection.
Information Sciences, 507:67–85, 2020.

Appendix

Non-Dominated Sorting: Definition

Non-dominated sorting is defined as the procedure that produces a set of fronts $\mathbb{F} = \{F_1, F_2, \ldots\}$ such that the following conditions hold:

1. Union of the points in all the fronts is same as the population
   - $\bigcup_{k \geq 1} F_k = \mathbb{P}$
2. The fronts are dis-joint
   - $\forall i \neq j \; F_i \cap F_j = \emptyset$
3. All the points in a particular front are non-dominated with each other
   - $\forall k \geq 1 \, \forall p, q \in F_k : p \not\prec q, q \not\prec p$
4. The first front consists of points that are not dominated by any point
   - $\forall q \in F_1 \; \nexists p \in \mathbb{P} : p \prec q$;
5. All the points in a particular front are dominated by at-least on of the points in its preceding front
   - $\forall q \in F_k, k > 1 \; \exists p \in F_{k-1} : p \prec q$.

Differential Evolution

- Differential Evolution algorithm starts with **initializing a random population** of candidate solutions and the search is done in a stochastic way by applying **mutation**, **crossover**, and **selection** operators to drive the population toward better solutions in the solution space.
- During each generation G, Differential Evolution produces the donor vector v_i for each individual x_i using the mutation operator in the current population. Below mentioned are the some of the most used mutation rules in Differential Evolution algorithm.

Approaches for Mutation in DE

- DE/rand/1

$$\hat{X}_i = X_{r1} + F(X_{r2} - X_{r3}) \tag{10}$$

- DE/best/1

$$\hat{X}_i = X_{\text{best}} + F(X_{r1} - X_{r2}) \tag{11}$$

- DE/rand-to-best/1

$$\hat{X}_i = X_{r1} + F(X_{\text{best}} - X_{r1}) + F(X_{r2} - X_{r3}) \tag{12}$$

- DE/current/1

$$\hat{X}_i = X_i + F(X_{r1} - X_{r2}) \tag{13}$$

- DE/current-to-best/1

$$\hat{X}_i = X_i + F(X_{\text{best}} - X_i) + F(X_{r1} - X_{r2}) \tag{14}$$

Pseudo code of Differential Evolution Algorithm

Algorithm PSEUDO CODE OF DE [1]

Input: NP: population size, F: mutation factor, CR: Crossover Probability, MAXFES: Maximum number of functions evaluation

Output: Solution set

Initialization:

1: $G \leftarrow 0$

2: Initialize all NP individuals with random positions in the search space

3: **while** $FES < MAXFES$ **do**

4: **for** $i = 1$ to NP **do**

Generate:

5: Select three individuals x_{r1}, x_{r2}, x_{r3} from the current population randomly. These must be distinct from each other and also distinct from individual x_i, *i.e.* $r_1 \neq r_2 \neq r_3 \neq i$

Mutation:

6: Form the donor vector using the formula $V_i = x_{r1} + F(x_{r2} - x_{r3})$

Pseudo code of Differential Evolution Algorithm

Algorithm PSEUDOCODE OF DE [1]

Crossover:

Trial vector u_i is developed either from the elements of the target vector x_i or the elements of the donor vector v_i as follows:

$$u_{i,j} = \begin{cases} v_{i,j} & if r_{i,j} \leq CR or j = j_{\text{rand}} \\ x_{i,j} & \text{otherwise} \end{cases} \tag{15}$$

where, $i = \{1, 2, \ldots, NP\}$, $j = \{1, 2, \ldots, D\}$, $r_{i,j} \in U(0, 1)$ is a uniformly distributed random number which is generated for each j and $j_{\text{rand}} \in \{1, 2, \ldots, D\}$ is a random integer used to ensure that $U_i \neq x_i$ in all class.

if $f(u_i) \leq f(x_i)$ **then**

    Replace the individual x_i in the population with the trial vector u_i

$FES \leftarrow FES + NP$

$G \leftarrow G + 1$

Binary Differential Evolution

- Binary Differential Evolution is an extension of the classical Differential Evolution (DE) algorithm, which operates on binary strings instead of continuous vectors.
- BDE has effectively been used to solve optimization problems including feature selections, classification and clustering.
- One critical component of BDE is the mutation strategy, which determines the degree of exploration and exploitation in the search space.

Binary Differential Evolution

- Several variants of BDE have been proposed to enhance its performance and scalability.
- Self-adaptive BDE (SABDE) and adaptive BDE (ABDE) adapt the control parameters of the mutation operator based on the search history.
- Multi-objective BDE (MOBDE) extends BDE to handle multi-objective optimization problems.
- Dynamic BDE (DBDE) adapts the mutation strategy and control parameters dynamically based on the current search status.
- Hybrid BDE (HBDE) combines BDE with other optimization algorithms to exploit their complementary strengths.

Algorithm MOFS-BDE

Parameters: The maximal iteration times T_{max} , the population size N, the frequency of implementing OPS T_{loc} , the scale F, and the crossover probability CR.

Input: The dataset for classification.

Output: The Pareto-optimal solutions with each corresponding to a feature subset.

1: **Initialize** a number of individuals, $P_0 = \{X_1, X_2, \ldots, X_N\}$
2: Let $t = 0$ ▷ Iteration steps
3: Iteration: Set the set $P_{t+1} = \phi$
4: **for** $i = 1, 2, \ldots N$ **do**
5: Randomly select three vectors from the population P_t , denoted as $X_{r1}(t)$, $X_{r2}(t)$ and $X_{r3}(t)$, $r_1 \neq r_2 \neq r_3 \neq i$
6: Select the best one from the three vectors as the base vector, $X_{best}(t)$
7: Generate a new mutation vector $V_{i(t)}$ for the i_{th} individual according to Eq. (3) and (4)
8: Generate a trial vector $U_{i(t)}$ for the i_{th} individual according to Eq. (5) :
9: Evaluate the fitness of the trial vector $U_i(t)$
10: Compare the i_{th} individual $X_{i(t)}$ with $U_{i(t)}$.If $X_{i(t)}$ dominates $U_{i(t)}$, save $X_i(t)$ into P_{t+1} ;if $U_{i(t)}$ dominates $X_{i(t)}$, save $U_{i(t)}$ into P_{t+1} ;otherwise, save both $U_{i(t)}$ and $X_{i(t)}$ into P_{t+1}

Algorithm MOFS-BDE

11: If the size of P_{t+1} is larger than N, remove $|P_{t+1}| - N$ individuals with higher ranks and shorter crowding distances from P_{t+1} using the method of non-dominated sorting and crowding distance ;

12: Step If $t/T_{loc} = t/T_{loc}$, run the problem-specific local search (refer to Algorithm 6 for details);

13: If $t < T_{max}$, let t ++ , and return back to Step 3; otherwise, terminate the algorithm, and output the Pareto-optimal solutions.

Algorithm ONE-BIT PURIFYING SEARCH (OPS)

Input: The population P_t and the set of non-dominated solution S_t.
Output: The new population P_t

1: Randomly select a solution from S_t , and set it as the reference solution, $X_{ref} = (x_{ref_1}, x_{ref_2}, \ldots, x_{ref_D})$
2: Randomly select two feature bits, u_1 and u_2, from X_{ref}, satisfying $x_{ref_{u1}} = 1$ and $x_{ref_{u2}} = 0$
3: Judge the relative importance between the two feature bits u_1 and u_2
4: **for** an optimal solution $X_h \in S_t$ **do**
5: Generate a new individual X_h by checking the following four cases: (Without loss of generality, we suppose $u_1 \prec u_2$)
6: **Initialize** $X_h\prime = X_h$
7: **if** $X_{h,u_1} = X_{h,u_2} = 1$ **then**
8: Set $x_{h,u_2}\prime = 0$
9: **else if** $X_{h,u_1} = X_{h,u_2} = 0$ **then**
10: Set $x_{h,u_1}\prime = 1$
11: **else if** $X_{h,u_1} = 1$ **AND** $X_{h,u_2} = 0$ **then**
12: Set $x_{h,u_1}\prime = 0$
13: **else**$x_{h,u_1}\prime = 1$ and $x_{h,u_2}\prime = 0$

Algorithm ONE-BIT PURIFYING SEARCH (OPS)

14: If $X_h\prime$ dominates X_h , population P_t saves $X_h\prime$ to replace X_h ; if $X_h\prime$ is dominated by X_h , the population keeps X_h unchanged; otherwise, it saves both $X_h\prime$ and X_h into P_t.

15: **if** If the size of P_t is larger than N **then**, remove $|P_t| - N$ individuals with high ranks, and reduce the crowding distances from P_t using the **Non-Dominated Sorting Algorithm** and Crowding Distance Methods.

16: Output population P_t