Project: Evolutionary Algorithm for Multi-Objective Optimization

Team31: Zhicong Sun,Yuxin Zhou,Zikang Gan,Hao Lee,Pengjie Liu

December 2020

.

1 Introduction of this report

In this project, we have improved the SPEA2 algorithm and then propose a algorithm named SPEA2PRO, this algorithm defeats all the compared evolutionary algorithms using two main methods. This paper introduces the basic knowledge in Section 2, compares and analysis three classical algorithms in Section 3, then explains the improvement work in Section 4. Finally, it summarizes our work and attaches the code.

2 Basic

2.1 Multi objective optimization

In life, many problems are composed of multiple goals that conflict and influence each other. People often encounter the optimization problem of making multiple objectives as best as possible in a given region, that is, multi-objective optimization problem. If there is more than one optimization objective in the optimization problem and it needs to be dealt with at the same time, it becomes a multi-objective optimization problem.

Multi objective optimization problems are very common and important in real life, such as engineering applications. These practical problems are usually very complex and difficult, and they are one of the main research fields. Since the early 1960s, multi-objective optimization problems have attracted more and more attention of researchers with different backgrounds. Therefore, it is of great scientific research value and practical significance to solve the multi-objective optimization problem.

In mathematical expression, the optimization problem with only one objective function is called single objective optimization problem, and the optimization problem with more than one objective function and needs to be handled simultaneously is called multi-objective optimization problem. The mathematical problems can be expressed as follows: Given decision vector space $\hat{X} = (x_1, x_2, \cdots, x_n)$, It satisfies the following constraints:

$$g_i(X) \geq 0 (i = 1, 2, \cdots, k)$$

$$h_i(X) = 0 (i = 1, 2, \cdots, k)$$

There are R optimization objectives, which are in conflict with each other

$$\hat{F}(X) = (f_1(X), f_2(X), \cdots, f_r(X))$$

Seeking a set of Solutions $X^* = (x_1^*, x_2^*, \cdots, x_n^*)$, make $\hat{F}(X^*)$ satisfy two constraints and achieve the optimization at the same time.

In multi-objective optimization, there may be different optimization objectives for different sub objective functions, some may be maximum objective functions, some may be minimum objective functions, or both.

For example, the design variable $\hat{X}_1 = (x_1, x_2, \cdots, x_n)^T$ is a set of determined vectors corresponding to a point in the n-dimensional Euclidean design variable space, while the corresponding objective function $f(x)$ corresponds to a point in the r-dimensional Euclidean objective function R^r space. In other words, the objective function f corresponds to a mapping :$f : R^n \to R^r$ from n-dimensional design variable space to m-dimensional objective function space.

Therefore, design variables, objective functions and constraint functions are the three elements of multi-objective optimization problems. A set of design variables can usually be represented by vector $\hat{x} = \{x_1, x_2, \cdots, x_n\}$, which is called a solution of optimization problem.

Objective function can be regarded as a mathematical expression to evaluate the performance index of design system. In practical engineering design, designers (decision makers) hope to optimize these performance indexes at the same time. All the objective functions $f_1(x), f_2(X), \cdots, f_r(X)$ constitute the objective function vector of the multi-objective optimization problem.

Consider two decision vectors $a, b \in X$. A Pareto dominates B, denoted as $a>b$;

If and only if$\{\forall i \in \{1, 2, \cdots, n\}\, f_i(a) \leq f_i(b)\} \wedge \{\exists j \in \{1, 2, \cdots, n\}\, f_i(a) < f_i(b)\}$.

If there is no Pareto dominant decision vector in the whole parameter space, the decision vector is called Pareto optimal solution. All Pareto optimal solutions constitute a set of Pareto optimal solutions.

In the case of multiple Pareto optimal solutions, it is difficult to choose which solution is better without more information about the problem. Therefore, all Pareto optimal solutions can be considered as equally important. Therefore, for the multi-objective optimization problem, the most important task is to find as many Pareto optimal solutions as possible. Therefore, in the multi-objective optimization, the following two tasks are completed: 1) to find a set of solutions as close as possible to the Pareto optimal domain. 2) Find a set of solutions as different as possible

The first task must be done in any optimization work. It is not advisable to converge to a solution that is not close to the real Pareto optimal solution set. Only when a group of solutions converge to the true Pareto optimal solution, can we ensure that the group of solutions is approximately optimal.

In addition to requiring the solution of the optimization problem to converge to the approximate Pareto optimal region, the solution must also be evenly and sparsely distributed on the Pareto optimal domain. Multiple solutions are based on a set of good protocols. In multi-objective evolutionary algorithm, decision makers usually need to deal with two spaces decision variable space and objective space, so the diversity of solutions (individuals) can be defined in these two spaces. For example, if two individuals have large Euler distance in the decision variable space, then the two solutions are said to be different in the decision variable space; similarly, if two individuals have large Euler distance in the target space, they are said to be different in the target space. Although diversity in one space usually means diversity in another for most problems, this conclusion is not true for all problems. For such a complex nonlinear optimization problem, it is also a very important task to find a group of solutions with good diversity in the required space.

At present, there are two main algorithms to solve Pareto frontier solution, one is based on mathematical programming, the other is based on genetic algorithm. Multi objective genetic algorithm is an evolutionary algorithm used to analyze and solve multi-objective optimization problems. The core of multi-objective genetic algorithm is to coordinate the relationship between each objective function, and to find the optimal solution set that makes each objective function reach a larger (or smaller) function value as much as possible. Among many multi-objective optimization genetic algorithms, NSGA-II algorithm (NSGA-II) is the most influential and widely used multi-objective genetic algorithm. Since its emergence, it has become a basic algorithm in multi-objective optimization problems because of its simple, effective and obvious advantages.

2.2 Benchmark

DTLZ is one of the most popular multi-objective optimization problem sets. DTLZ contains 9 problems and the last 2 problems are with constrains. The number of dimensions of decision space and objective space are scalable. In this experiment, the first 4 and the seventh problems (i.e. DTLZ1-4 and DTLZ7) are used.

DTLZ1 is a relatively easy M-objective problem with linear Pareto optimal boundary. DTLZ2 is used to examinate the performance increment of a MOEA with the number of objectives increases. DTLZ3 changes a part of functions of DTLZ2, and it is used to examinate convergence ability. DTLZ4 is a problem based on DTLZ2 for examinating distribution of alogrithm. DTLZ7 contains a set of discontinuous Pareto optimal boundary.

In order to increase the difficulty of Pareto front, IDTLZ1, IDTLZ2 and so on are proposed which are based on problems of DTLZ, and these sets are used in this experiment.

2.3 Indicators

Because the solution of multi-objective optimization problem contains multiple target values, it is impossible to measure the performance of the algorithm by directly comparing the size of the target value of the solution obtained by the algorithm. Therefore, some performance indexes used to quantify the population obtained by the algorithm are proposed one after another. In this experiment, IGD and HV are available to measure

performance of algorithms.The formula of IGD is as follows and finally we decided to use IGD as it made a better result in this experiment.

$$IGD(P,R) = \frac{\sum_{r\in R} min_{p\in P}||p-r||}{|R|} \tag{1}$$

As we have mentioned above, there are some special values in the IGD formula where P stands for the population to be evaluated, R stands for a group of reference points on the Pareto front, $||p-r||$ represents the Euclidean distance between solution P and reference point R in the target space. In other words, smaller IGD value indicates higher degree of similarity between P and R, which means that P has better convergence and distribution. In addition, it is more important to calculate IGD to obtain a set of uniformly distributed reference points R on Pareto front of the current multi-objective optimization problem.

For HV, λ represents Lebesgue measure and its expansion is shown in the following formula. The set of reference points required in HV is not a sampling point at Pareto front, but one or more reference points far from the front.

$$HV(P,R) = \lambda(H(P,R)) \tag{2}$$

$$HV(P,R) = \{z \in Z | \exists p \in P, \exists r \in R : p \leq z \leq r\} \tag{3}$$

The selection of reference points in HV is also a key step to improve the prediction effect. In fact, HV computation is an NP problem, that is, its computational complexity is related to the M exponential of the target dimension. In order to improve the computational efficiency of HV in high-dimensional multi-objective optimization, Monte Carlo Simulation method is usually used to approximate the HV value at $M > 4$. In addition, in order to reduce the magnitude of the HV value, the target value of the solution in the population is usually normalized to $[0,1]^M$ before HV is calculated.

$$(HV(P,R)) = \int_{R^M} 1_{H(P,R)}(z)dz \tag{4}$$

Combined with the formula derivation process, method features and how to apply the two methods mentioned above to judge the advantages and disadvantages of the prediction results, in this experiment, we found that the effect by using IGD is better than that of HV. Therefore, the IDG Indicator value is mainly used as the experimental reference data in subsequent experiments.

3 Comparison of three algorithm

3.1 Principle

3.1.1 NSGA-II

1. The construction method of non dominated set
Suppose the size of the population Pop is N, then divide it to m subsets $P_1, P_2, ..., P_m$ by some strategy.

```
sort(Pop){
  for each p∈Pop{
    for each q∈Pop{
      if (p dominated q) then
        sp = spU{q}
        else if (q dominated p) then
          np = np + 1;
    } end for q
      if (np = 0) then
        P1 = P1U{p}
  } end for p
  i = 1;
  while (Pi ≠ ∅){
    H = ∅;
    for each p∈Pi{
      for each q∈sp
      nq = nq - 1;
      if (nq = 0) then H = HU{q}
    } end for p
    i = i + 1;
    Pi = H;
  }end for while
}end for sort
```

Figure 1: Algorithm: Construct non dominated set

This algorithm consists of two parts. First one is to calculate n¡sub¿i¡/sub¿ and s¡sub¿i¡/sub¿, whose time complexity is $O(rN^2)$. And the second part is to get $P_1, P_2, ...P_m$ the worst situation is the size of the next population is N who gets N Pareto front, which means m = N. And its time complexity is $O(N^2)$. So the total time complexity of this algorithm is $O(rN^2;) + O(N^2) = O(rN^2)$.

2. The method to keep the distribution and diversity of solution population

In order to keep the distribution and diversity of solution groups, first step is to calculate every individual's distance to each other, then according to the distances, define a partial order set, which is the baisc of constructing new population.

```
crowding-distance-assignment(P){
  N = |P|;
  for each i, P[i]distance = 0;
    for each objective m{
      P = sort(P, m);
      for i = 2 to (N-1)
        P[i]distance += (P[i+1].m - P[i-1].m)
    } end for objective m
  P[0]distance = P[N]distance = ∞;
}
```

Figure 2: Algorithm: Calculate the crowding distance between each individual

The worst situation of algorithm 2.2 is to sort r targets, whose time complexity is $O(rNlogN)$. Otherwise the time complexity of calculating the crowding distance is $O(rN)$, so the total time complexity is $O(rNlogN)$.

3. NSGA-II of Deb

First initialize a population P¡sub¿0¡/sub¿, then create a new population Q¡sub¿0¡/sub¿ by crossing over and mutation. The size of them are both N. Then we should build partial order set, as is follow.

```
{
  Rt = PtUQt
  F = fast - nondominated - sort(Rt)
  Pt+1 = ∅ and i = 1
  until (|Pt+1| + |Fi| <= N)
    crowding-distance-assignment(Fi)
    Pt+1 = Pt+1UFi
    i = i + 1   (end of until)
  sort(Fi, >n)
  Pt+1 = Pt+1UFi[1:(N-|Pt+1|)]
  Qt+1 = make-new-Pop(Pt+1)
  t = t + 1
}
```

Figure 3: Algorithm: NSGA-II of Deb

In fact, based on F=fast-nondominated-sort(R_t), it could creat more than one set $F = (F_1, F_2, ...)$, but only some part of them will be selected to new population. And the time complexity of this algorithm consists of three parts (r is the number of targets):

1. Non-dominated sorting: $O(r(2N)_2)$.
2. Crowding distance assignment: $O(r(2N)log(2N))$.
3. Partial order set: $O(2Nlog(2N))$.

So, the total time complexity of this algorithm is $O(rN^2)$.

3.1.2 SPEA2

The main process of SPEA2 is shown as following:

```
{N is the size of population P, M is the size of Q, T is the number of generation.}
1.Inialization: Create a new population P0, and let Q0 be null, t=0.
2.Adaptation allocation: Calculate every individual's adaptation in Pt and Qt.
3.Environment selection: Select all non-dominated individuals to Qt+1 from Pt and Qt.
4.End condition: If t>=T, or other end conditions satify, then save all the non-dominated   individuals in Qt+1
to NDSet.
5.Pairing.
6.Evolution: Crossing over and mutation.
```

Figure 4: Algorithm: SPEA2

The time complexity of calculating individual adaptation is $O(rN^2logN)$. Then the time complexity of environment selection is $O(rN^3;)$, the average time complexity is $O(rN^2logN)$.

3.1.3 PESA-II

The main process of PESA-II is shown as following:

```
1.Initialize and evaluate an inter population IP. And initialize a null external population.
2.Put non-dominated individual from IP to EP.
3.If satify the end condition, then return EP. Or erase IP and create new IP.
```

Figure 5: Algorithm: PESA-II

3.2 Advantages and disadvantages

The advantages of NSGA-II is mainly based on NSGA. First and the most important one is about time complexity. Compared with NSGA, NSGA-II sharply decrease the time complexity of creating partial order set. Second the crowding distance from NSGA-II is easier to select the suitable individual. Last the NSGA-II create the method to keep the distribution and diversity of solution population.

3.3 Metric comparison

In general, judging the metric of algorithm seems not comprehensive, if only consider the result of experiment, because of the data set. However, there are three metrics we should carefully consider when designing MOEA, they are convergence, diversity and efficiency.

3.3.1 Convergence

Based on the metric for convergence from Deb, all of those three algorithm have a good convergence and stability. However, NSGA-II gets bad convergence when obtaining 6 targets. In general, PESA-II gets better convergence, and it is better than SPEA2 in some cases.

3.3.2 Diversity

Based on the evaluation method from Deb, the result shows all of them have a good distribution. In detail, NSGA-II performs better, then is SPEA2, the last is PESA-II.

3.3.3 Efficiency

Considering target number is different in evolution. Larger the target number, lower the efficiency. At the same condition, the fastest is NSGA-II, then is SPEA2, the slowest is PESA-II.

4 Improvement

Although SPEA2 has certain advantages in reference to other multi-objective algorithms of the same type, its genetic evolution operation has strong randomness. Although this can expand the search range and not make the algorithm surround the local convergence, but at the same time, it must also search carefully. After reaching the vicinity of the Pareto optimal solution, the optimization efficiency of SPEA2 will often decrease significantly, and sometimes it is even impossible to search for the Pareto optimal solution. The resulting solution set is not the overall optimal solution. It often happens that the obtained solution set is near the optimal solution, and it can be obtained only by a little search in the nearby neighborhood. However, due to the large-scale randomness of genetic operations, it cannot guarantee that these points can be searched every time, which affects the convergence performance of the algorithm.

To solve these problems, we have made some optimizations to the SPEA2 algorithm and name this enhanced algorithm SPEA2PRO, the two main aspects of the optimizations are as following:

4.1 Enhance the capability of local search

The local search method has its unique local search ability, and its combination with other multi-objective algorithms becomes a feasible research direction. In order to ensure that the SPEA2 algorithm has global search capabilities while enhancing its local search performance, we separately set up an external population, which is used to store the non-dominated set generated in each iteration, and keep this population in the next total population Hope that by increasing the proportion of the non-dominated set, a better solution can be searched, and the Pareto optimal boundary of the multi-objective optimization problem can be quickly approached.

4.2 Eliminate some individuals regularly

In order to further enhance the diversity of the population in the iteration and reduce the risk of falling into the local optimum, we will 'reselect' some weak individuals in the population according to a certain proportion

every 10 generations, that is, delete them, and then reinitialize some individuals to join the next parent population. This method can make the population always keep new individuals joining, which is conducive to the evolution of the population, maintains the diversity of the population, improves the search performance of the evolution algorithm, and effectively prevents premature phenomena.

4.3 Analysis of experimental results

A platform, called PlatEMO, is used in our experiment, and our programming language is Matlab. Seven instances, including DTLZ1-4, DTLZ7, IDTLZ1, IDTLZ2, are chosen as benchmarks. The number of objective is set to three. The IGD is used as the performance metric on all the instances. Furthermore, the wilcoxon rank sum test [8] with a significance level of 0.05 is adopted to perform statistical analysis on the experimental results obtained by 30 runs independently. Evaluation is set to 50000. The size of population is set 100.

The experimental results show that our algorithm SPEA2PRO has a lower IGD value and beats these three algorithms on at least 4 of 7 instances as we listed. By the the wilcoxon rank sum test, we can find out that the SPEA2PRO algorithm has much better performance than the NSGN-II and PESA-II algorithms, and a little bit better than the SPEA2 algorithm.

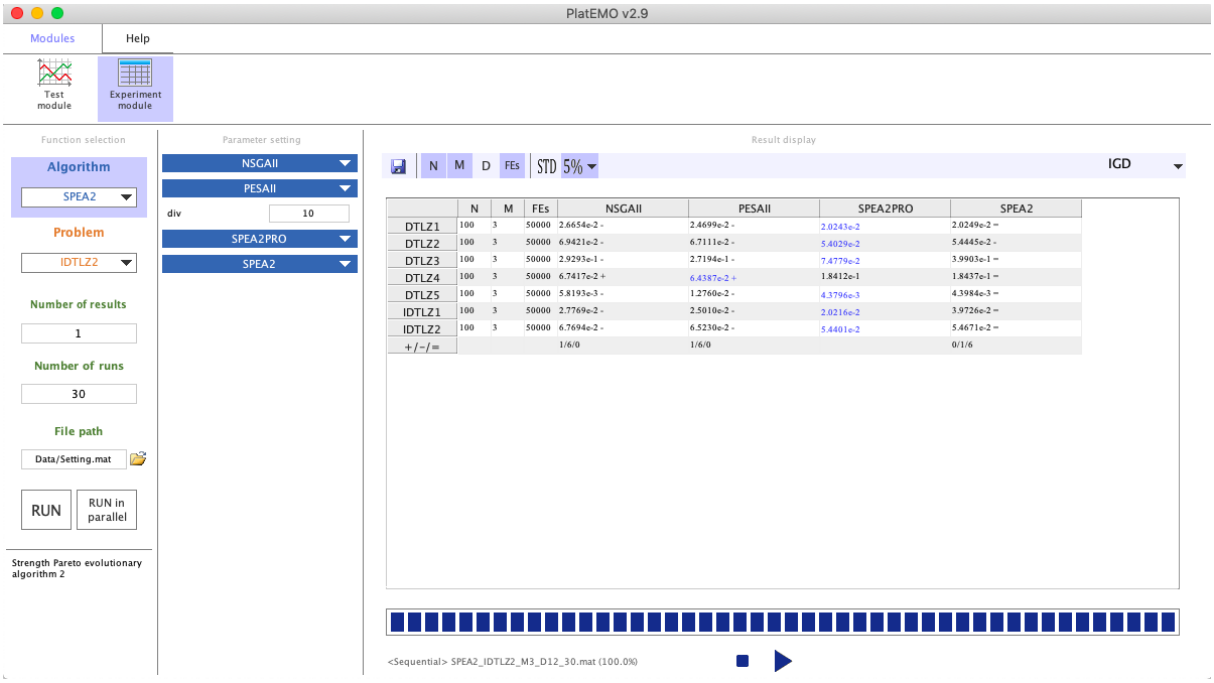

Figure 6: Compare our algorithm with classical algorithms

	N	M	FEs	NSGAII	PESAII	SPEA2PRO	SPEA2
DTLZ1	100	3	50000	2.6654e-2 -	2.4699e-2 -	2.0243e-2	2.0249e-2 =
DTLZ2	100	3	50000	6.9421e-2 -	6.7111e-2 -	5.4029e-2	5.4445e-2 -
DTLZ3	100	3	50000	2.9293e-1 -	2.7194e-1 -	7.4779e-2	3.9903e-1 =
DTLZ4	100	3	50000	6.7417e-2 +	6.4387e-2 +	1.8412e-1	1.8437e-1 =
DTLZ5	100	3	50000	5.8193e-3 -	1.2760e-2 -	4.3796e-3	4.3984e-3 =
IDTLZ1	100	3	50000	2.7769e-2 -	2.5010e-2 -	2.0216e-2	3.9726e-2 =
IDTLZ2	100	3	50000	6.7694e-2 -	6.5230e-2 -	5.4401e-2	5.4671e-2 =
+/-/=				1/6/0	1/6/0		0/1/6

Figure 7: Compare our algorithm with classical algorithms(Detail)

References

[1] WENG Liguo, WANG An, XIA Min and JI Zhuang—zhuang. Improved SPEA2 based on local search. Application Research of Computers, 2014.

[2] Abdullah Konaka, David W.Coitb, Alice E.Smithc,Multi-objective optimization using genetic algorithms: A tutorial, Reliability Engineering and System Safety 91 (2006) 992–1007