Weight Vector Grid with New Archive Update Mechanism for Multi-Objective Optimization

Xizi Ni Hisao Ishibuchi Kanzhen Wan
Ke Shang Chukun Zhuang
Southern University of Science and Technology(SUSTech)
Shenzhen, China

Good Afternoon everyone. I think this is the last presentation of the GECCO Student workshop. Thanks for your time to listening to me.
My name is Ni Xizi, from SUSTech, Shenzhen, China
Today I will present a project about a weight vectors grid based archive for the Multi-Objective Optimization

Background

Current Problem

Proposed Idea

Experimental Results

Future Study

Conclusion

This is the outline of my presentation.
First I will talk about some background information.
And what is current problem, how we solve the problem.
Then some experimental result will be offered.
Finally we will discuss what can be done in the future and the conclusion

Background

So, let begin

More than one objective to optimize

$Object1$: Faster CPU
$Object2$: Larger Storage
…

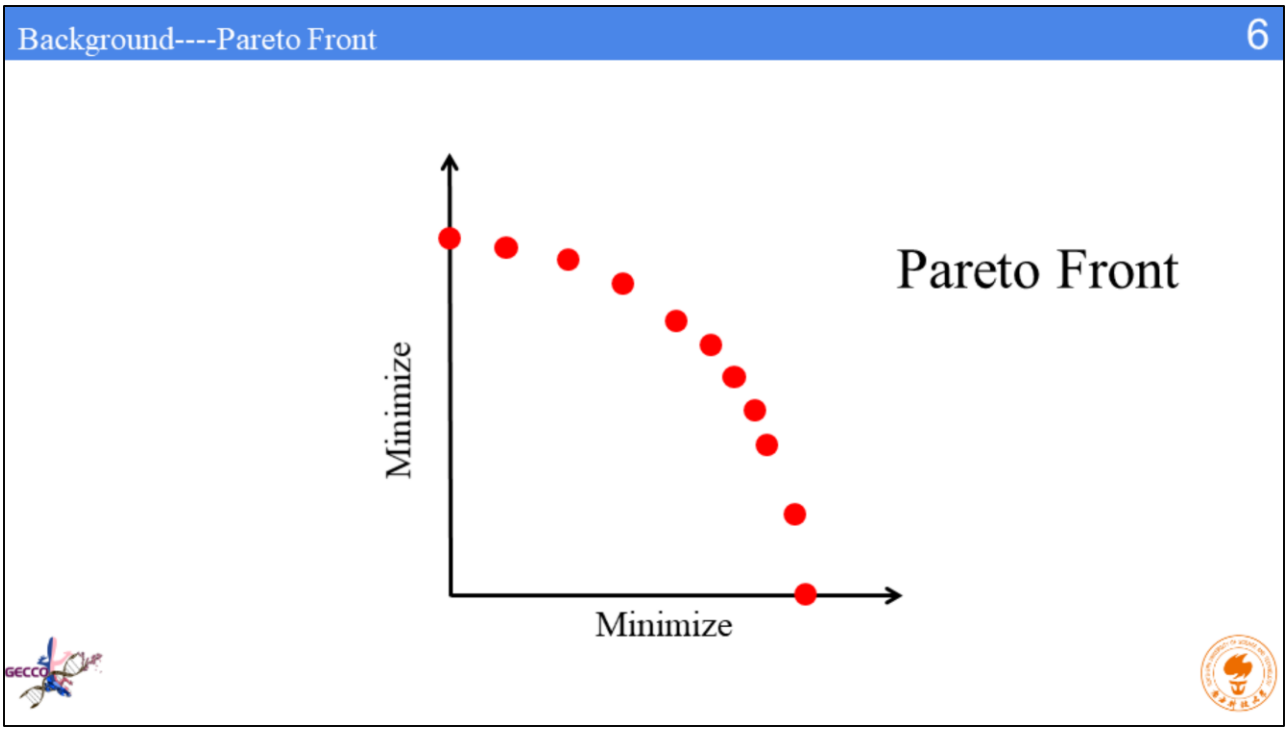
Multi-Objective Optimization is that we have more than one objective to optimize. For example, We want the computer to run faster and have Larger Storage.
We want to optimize all the components of the computer that is multi-objective optimization

Background----Pareto Dominance

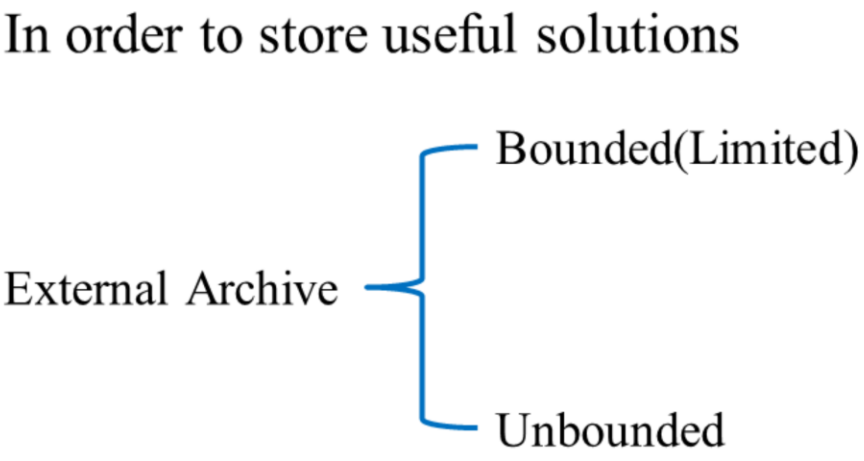
A dominate B: $\forall i\ A_i \geq B_i\ and\ \exists j\ A_j > B_j$

Computer A		Computer B
Speed	≥	Speed
Storage	>	Storage
Network	≥	Network

Here is a basic idea of Multi-Objective optimization, Pareto Dominance.
We say A dominate B if all the value in A is better or equal than B and at least one value better than B.
For example, if the speed of computer A is faster than B, storage is larger than B, network is faster than B we say Computer A dominate Computer B.



However, it faster speed and larger storage is not usually exist. There is a balance between them.
Usually we optimize two or more objective we will finally get all the non-dominated solution. And it will become a pareto front.



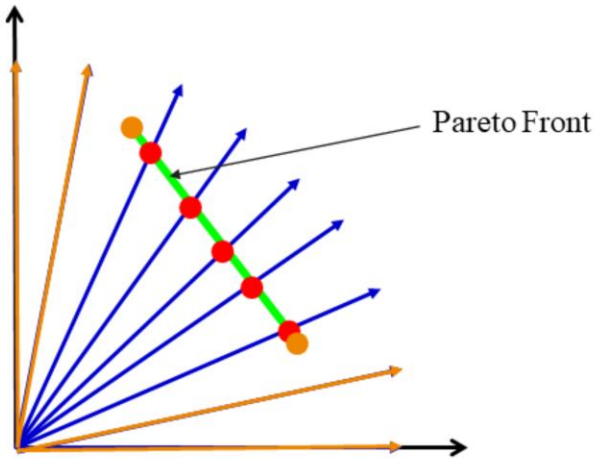
If we find the all the non-dominated solution, that size will be huge.
Therefore, we use an external archive to store all the non-dominated solution
The major classification is bounded external archive which means the size of archive is limited. And the other one is unbounded external archive, which size is unlimited.

Current Problem

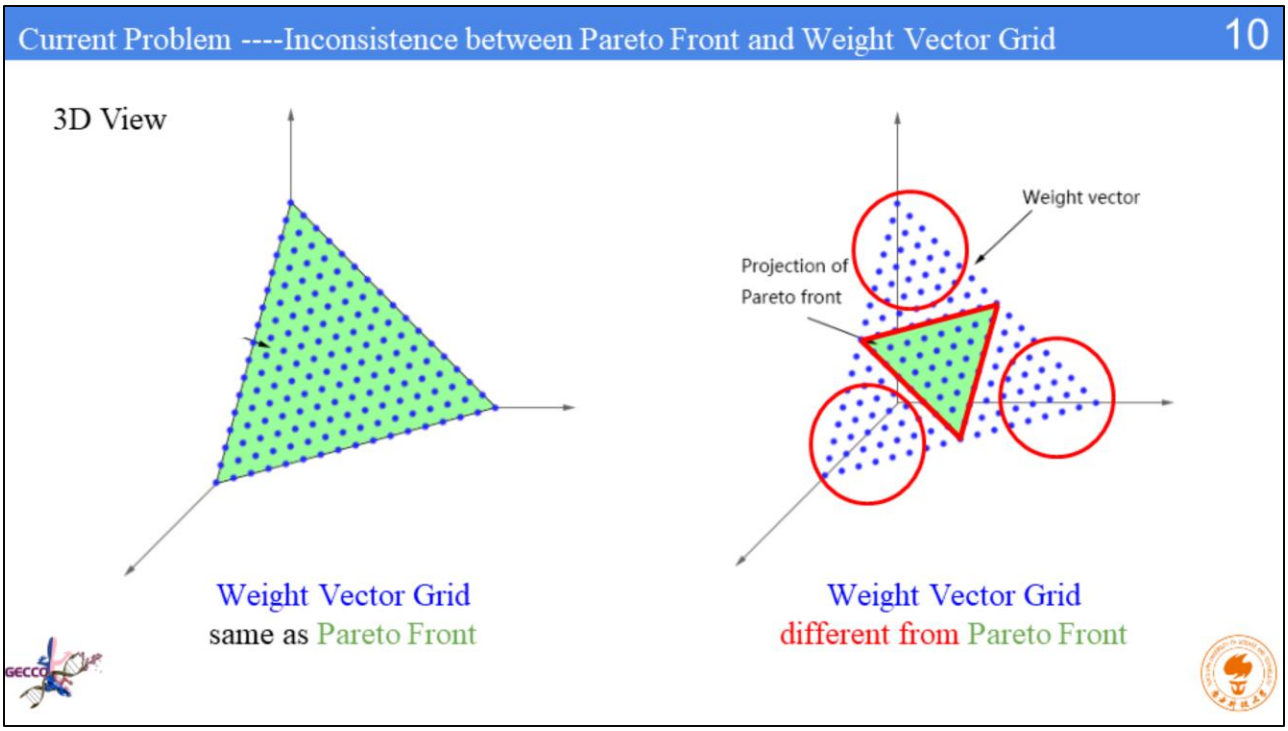
So, What is current problem?
Nowadays it is very popular to use decomposition based multi-objective evolutionary algorithm.

Current Problem ----Inconsistence between Pareto Front and Weight Vector Grid 9

2D View

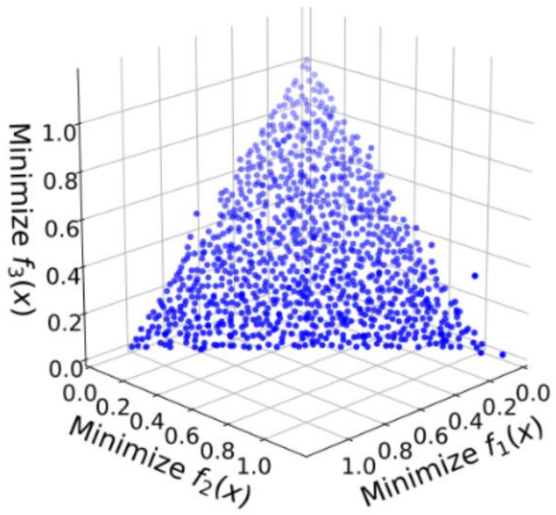


The general idea of decomposition based multi-objective evolutionary algorithm is that it use weight vector grid to divided objective space to many sub-problems and optimize them. However, it exist a problem.
Here is a set of uniformly distributed weight vector grid. The green line is the pareto front.
If the weight vectors pass through the pareto front, the intersections are the final solution.
But some weight vectors do not pass through the pareto front. The solution will be located on the end of the pareto front, which form a non-uniformly distributed solution set. In this example, there are more crowdy on the end of pareto front.

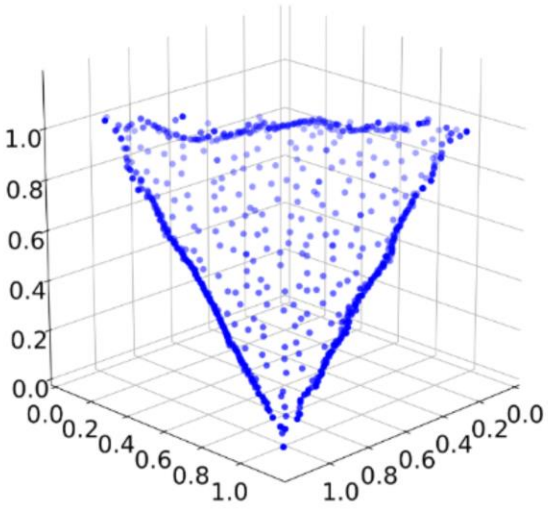


It is more clearly in the 3D figure. If the Weight Vector Grid is consistence with the pareto front which means that all the weight vectors pass through pareto front. The final solution will be uniformly distributed. However, if the weight vector grid is in-consistence with the pareto front. The solutions of the weight vectors which do not pass through the pareto front will located on the boundary.

Current Problem ----Inconsistence between Pareto Front and Weight Vector Grid



Weight Vector Grid
same as Pareto Front



Weight Vector Grid
different from Pareto Front

Let see the simple example. Weight Vector grid is consistence with pareto front in the left figure and inconsistence in the right.
It is clear that the left one is uniformly distributed and the right one have more points on the boundary.

Proposed Idea

So how to solve it?

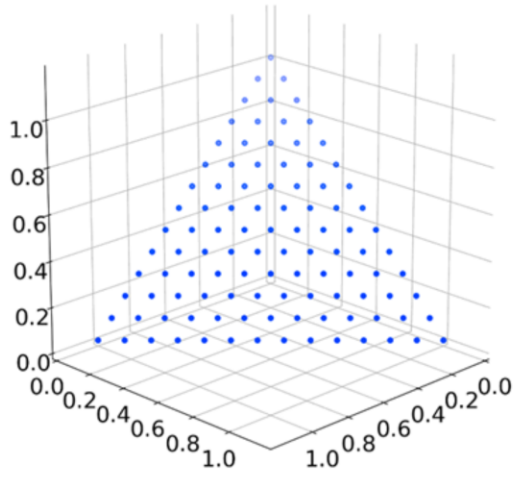
Use an uniformly distributed weight vector grid as an external archive

A new solution come, it will be assigned to the nearest weight vector

Use a scalarizing function if more than one solutions are assigned to the same weight vector

Our idea is build a uniformly distributed weight vector grid based external archive to store the solution.
When a new solution need to update to the external archive, it will first find the closest weight vector. And assign to it.
If a single weight vector have more than one solution assigned to it. It will use scalarizing function to find the better one.

Proposed Ideas ---- Uniformly distributed weight vector grid



Uniformly distributed weight vector grid

Each solution assigned to only one weight vector

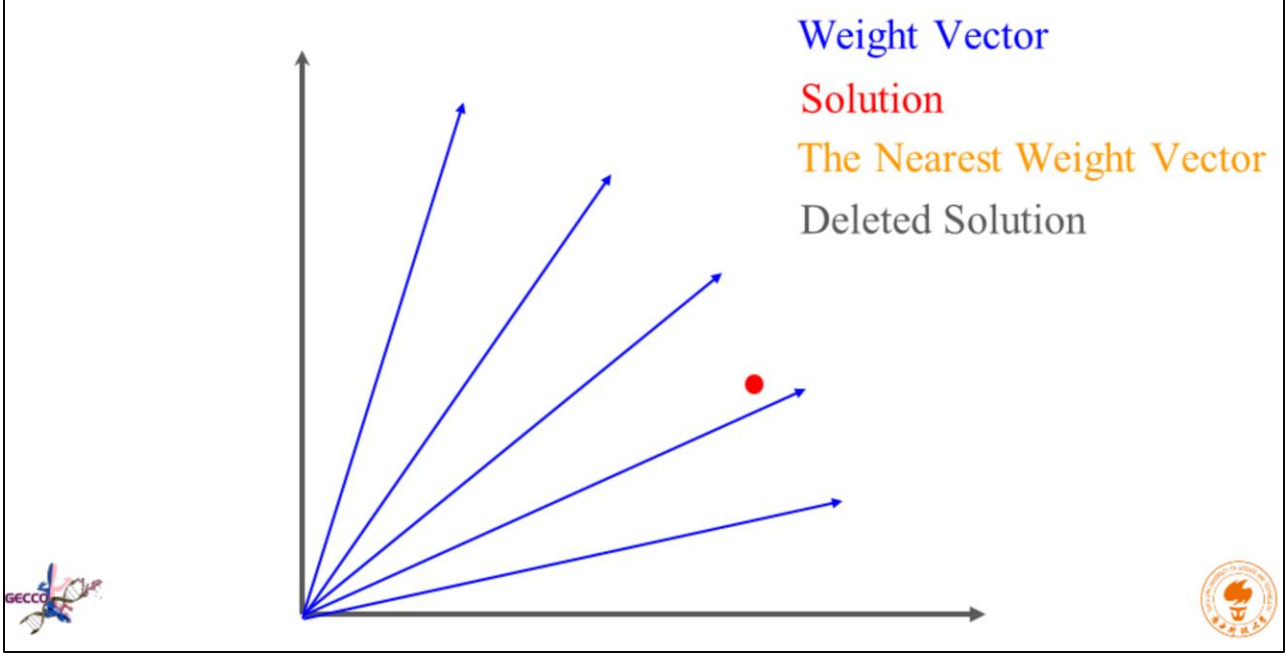
Each weight vector has only a single solution

Size of the weight vector grid ≥ The number of final solutions

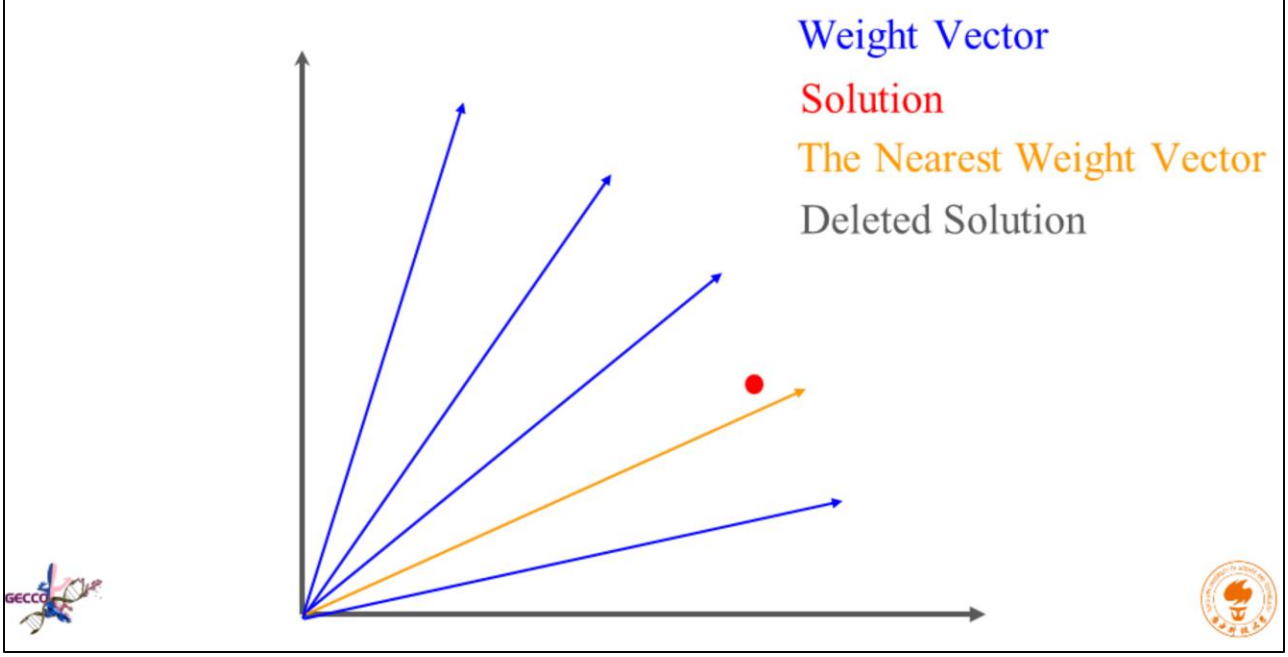
It is one-to-one map between weight vector and the solution. And some weight vector may not have any solution. Therefore the size of the weight vector grid which is the size of archive is larger or equal to the size of the final solution.

Input: Archive AP, New Solution NS
Output: AP
1: Find the closest weight vector wv
2: If wv has a solution, compare the new solution NS with the current one using a scalarizing function f. Assign the better solution to wv and update AP
3: Else assign NS to wv and update AP
4: Return AP

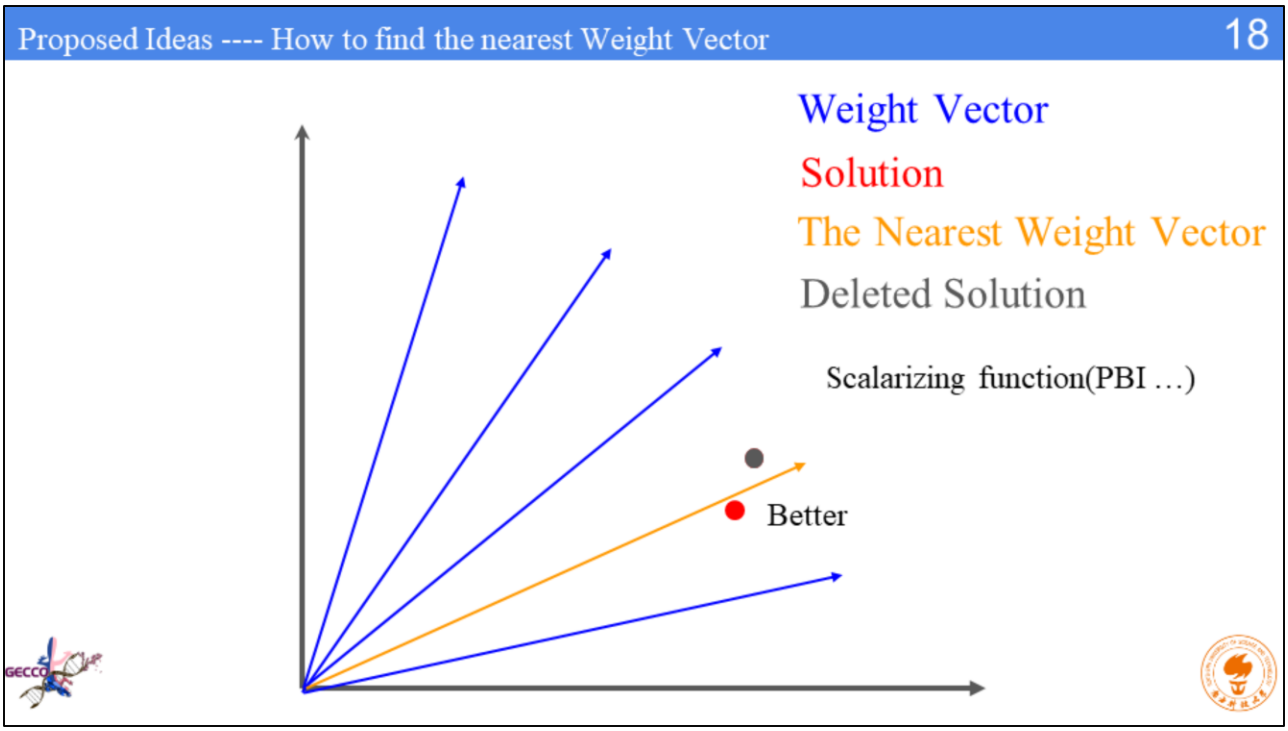
This is a simple algorithm framework



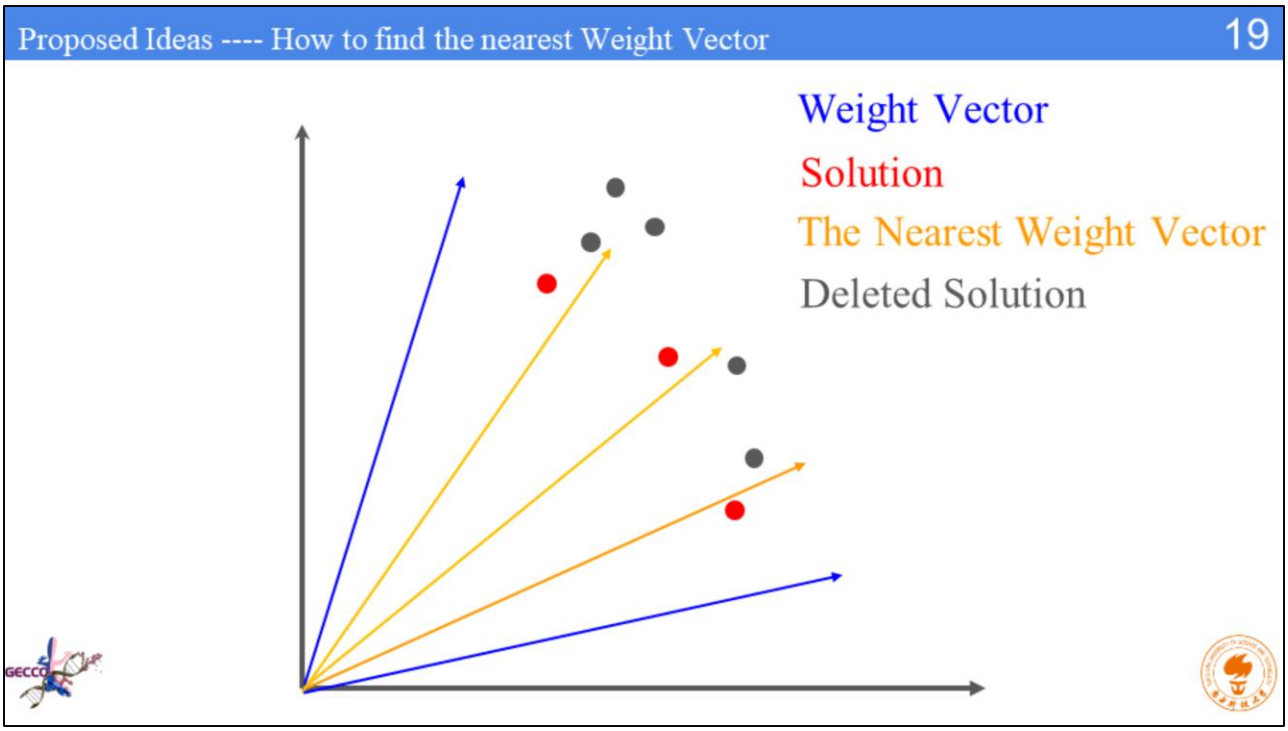
Here is a uniformly distributed weight vector grid
If a new solution wants to update to it.
It will first find the nearest weight vector.



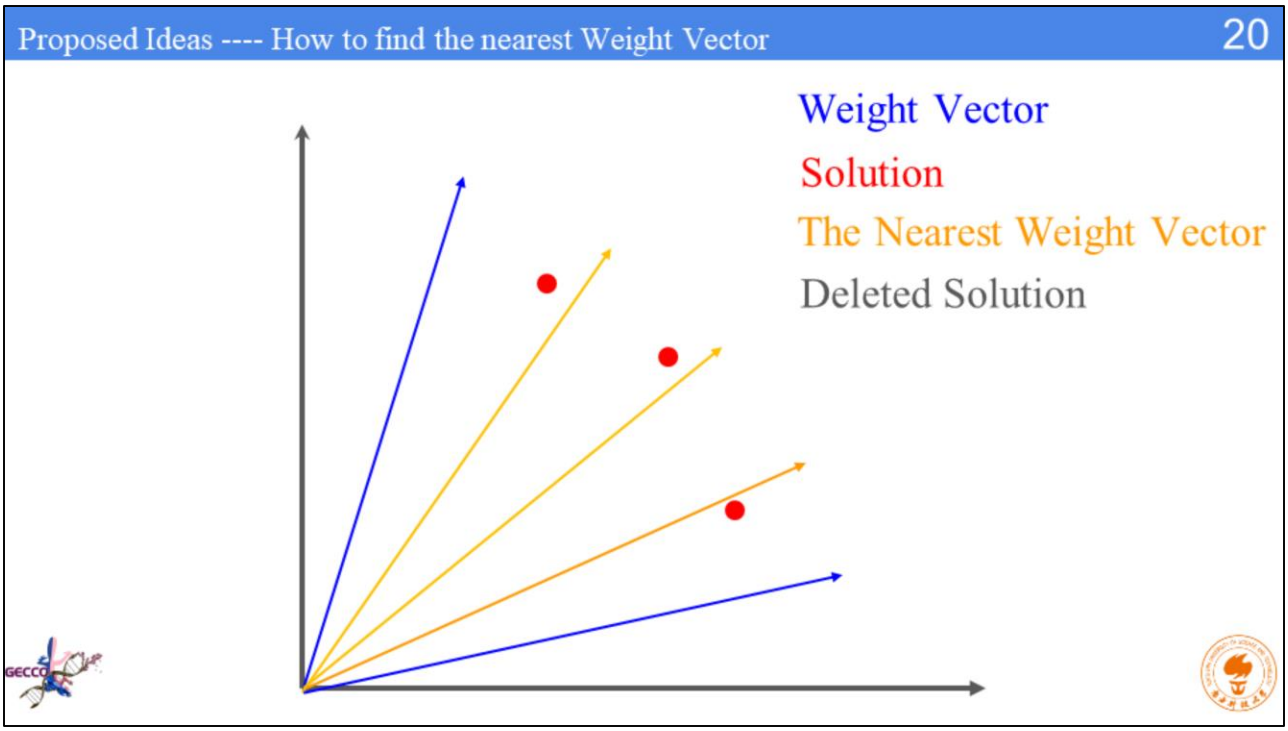
And assign to it



If another solution comes and it find that there is already have a solution in its closest weight vector.
It will use the scalarizing function to find the better one. The loser will be deleted.



Repletely using this algorithm to update the solution to the archive.



The solutions store in the weight vector grid are the final solution set. And there will be uniformly distributed, since the weight vectors are uniformly distributed.

Experimental Results

We have do some experiment on this new archive mechanism.

- Initialization method: Random Initialization
- Mutation probability: 0.01 (Gaussian Mutation)
- Crossover probability: 1 (Blending Crossover)
- Scalarizing function: Tchebycheff
- Neighborhood size: 2% of the population size

This is the global experiment setting.

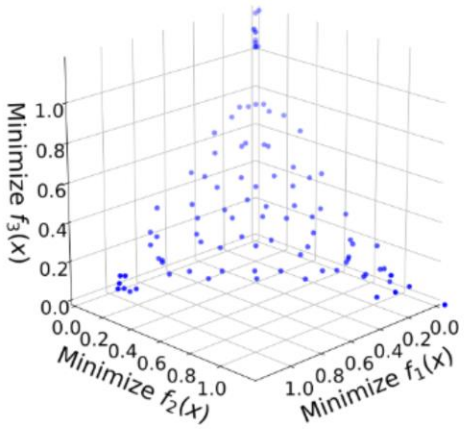
- MOEA/D-WV: MOEA/D with the weight vector based archive (Our Method)
- MOEA/D-NA: MOEA/D without the archive
- MOEA/D-DB: MOEA/D with the dominance-based archive
- MOEA/D-BA: MOEA/D with the bounded dominance-based archive

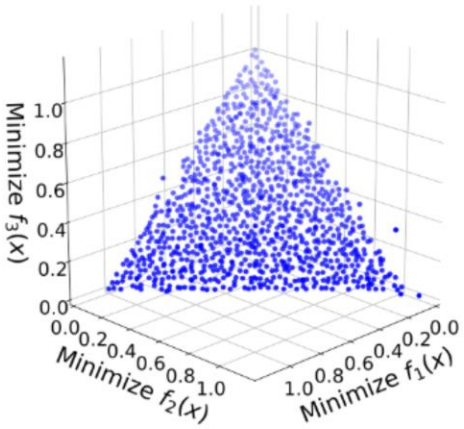
We use MOEA/D as our optimization algorithm and compare our method with three different kind of archive. MOEA/D without the archive, MOEA/D with dominance-based archive, and with bounded dominance-based archive.

- Dimension: 3 (i.e., 3 objectives)
- Population size: 100
- External Archive size: 1000
- Number of evaluated solutions: 100000
- Total Runs: 100 (Average results are reported)
- Test Problem: DTLZ1 (Triangle)
  I-DTLZ1 (Inverted Triangle)
  DTLZ2BZ (Concave PF)

First we compare the quality of the final solution set.

DTLZ1



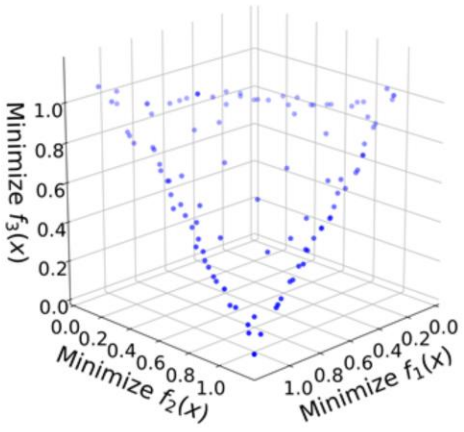


MOEA/D-NA

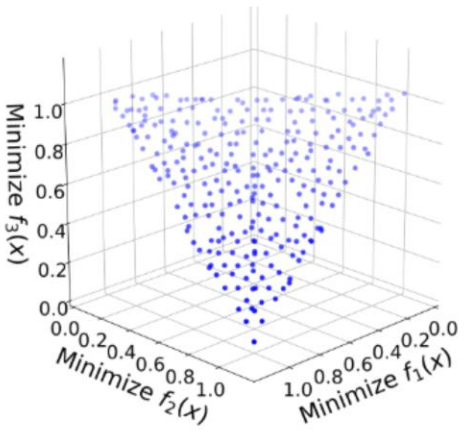
MOEA/D-WV

Here are some picture about the final solution set comparison.
The pareto front of DTLZ1 is triangle. The weight vector grid is consistence with pareto front.
We can visually see from the result that MOEA/D without archive and with weight vector grid archive are both uniformly distributed.

I-DTLZ1



MOEA/D-NA



MOEA/D-WV

However, when the shape of pareto front is different from the weight vector grid. The our method is still uniformly distributed but the MOEA/D without archive have more points on the boundary.

Experimental Results ---- Solution Set Comparison

Hypervolume The larger the better

	Algorithm	DTLZ1	I-DTLZ1	DTLZ2BZ	Average
Our Method	MOEA/D-WV	1.1460	0.3081	0.5773	0.6771
	MOEA/D-NA	1.0628	0.2717	0.5660	0.6335
	MOEA/D-BA	1.1509	0.3169	0.5827	0.6835
	MOEA/D-DB	1.1509	0.3169	0.5827	0.6835

Less than 1% difference in hypervolume between MOEA/D-WV and MOEA/D-DB

We did some experiment result by using Hypervolume as indicator.
We run those four algorithm on three different test problems for 100 time and the average result return.
We can see from the figure that there are less than 1% difference in hypervolume between the MOEA/D-DB and MOEA/D-WV

Experimental Results ---- Computational Time Comparison

- Dimension: 3, 4, 5 (i.e., 3, 4, 5 objectives)
- Population size: 100, 200, 300
- External Archive size: 1000, 2000, 3000
- Number of evaluated solutions: 100000, 200000, 300000
- Total Runs: 100 (Average results are reported)
- Test Problem: I-DTLZ1 (Inverted Triangle)

However they have huge difference in the computational time. Here we do some computational time comparison.

Experimental Results ---- Computational Time Comparison

Our Method

Computational Time (sec)				
Algorithm	3D	4D	5D	Average
MOEA/D-WV	397	1260	2572	1410
MOEA/D-BA	401	19505	92016	37307
MOEA/D-DB	1083	> 2 days	> 2 days	---

Decrease a lot of computational time

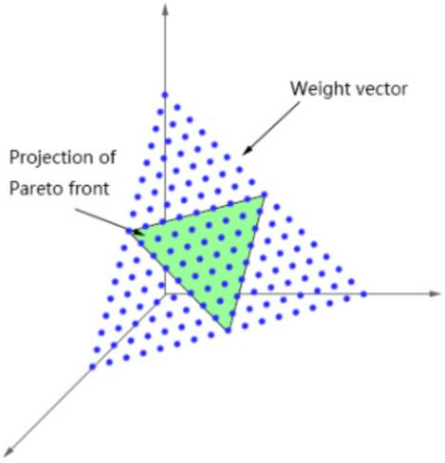
We run the three algorithm in 3D, 4D and 5D test problem for 100 time and average return. The number in red only run one time since they are time consuming.
We can clearly see the difference, our method run a lot faster than the other two algorithm.

Future Study

That's all the idea and the experiments. However there are many topic we can do about it.

Speed up the updating time

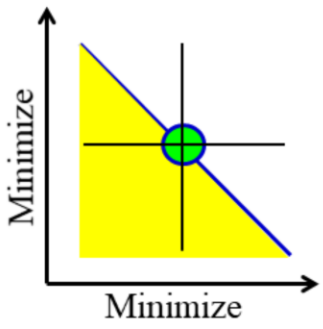
- Not all the solution needed to be updated to external archive (Usually early stage solutions are bad)
- Pruning the useless weight vectors (Out side of the Pareto Front)



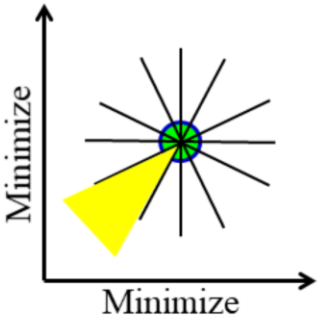
First, we need to speed up the updating time.
We have already do some experiments and we found that not all the solution are needed to be updated to external archive. Because in the early stage the solution are bad. This method can save a lot of time.
Another one is adaptive change the weight vector grid, deleted the weight vectors that are useless. We have not find a good way to adaptively change.

Using different scalarizing functions in MOEAs and external archive

- Weighted sum in MOEAs and PBI in external archive



Weighted Sum



PBI

Figures from: Presentation of Hisao Ishibuchi

Here is a interesting topic.
Since we have two weight vector grid, one is in the main loop the MOEA part, another one is in the external archive, we can use different scalarizing functions on different weight vector grid.
The different scalarizing function have different characteristic. In two objective optimization, the weighted sum have 50% solution that is better than it. However, the PBI only have less than 10%. So weighted sum will push the solution set faster than PBI. But weighted sum cannot solve the concave problem.
Because of difference advantage of different scalarizing function, we can use weighted sum in the MOEA part and the PBI in the external archive part.

Conclusion

In conclusion

The quality of the solution set stored by the weight vector grid-based external archive is almost the same as the unbounded domination-based external archive

The computational time is much shorter

The weight vector grid-based archive can be used in every multi-objective evolutionary algorithm

Thank you!

Q&A