# Multi Agent Collaborative Search Algorithm with Adaptive Weights

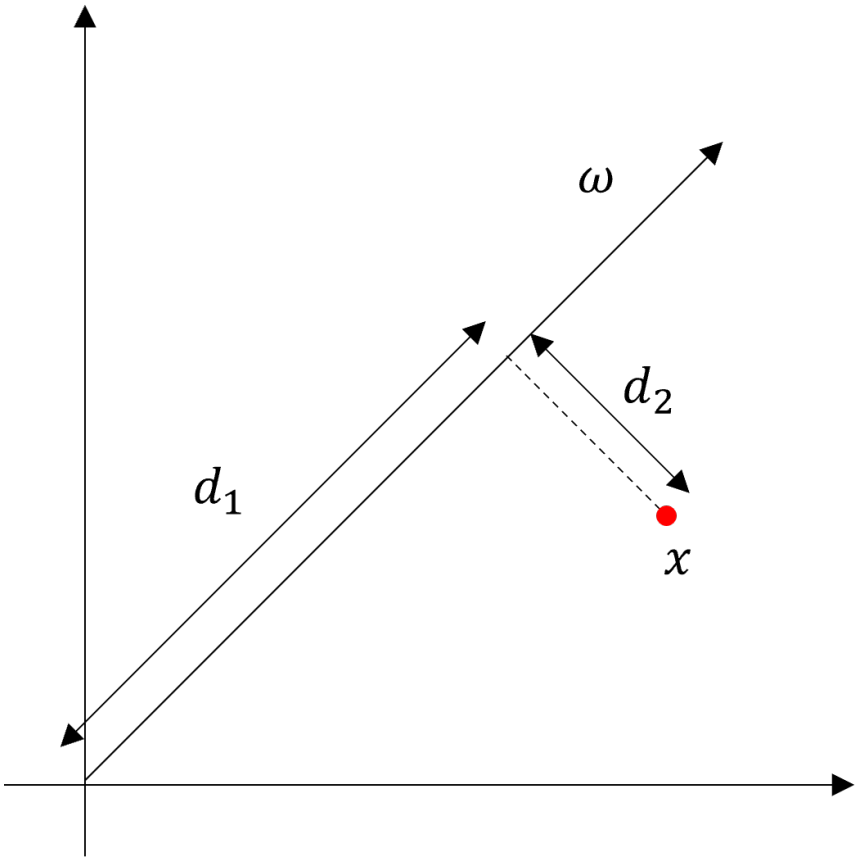
Li Cao[1], Maocai Wang[1], Guangming Dai[1], and Max Vassile[2]

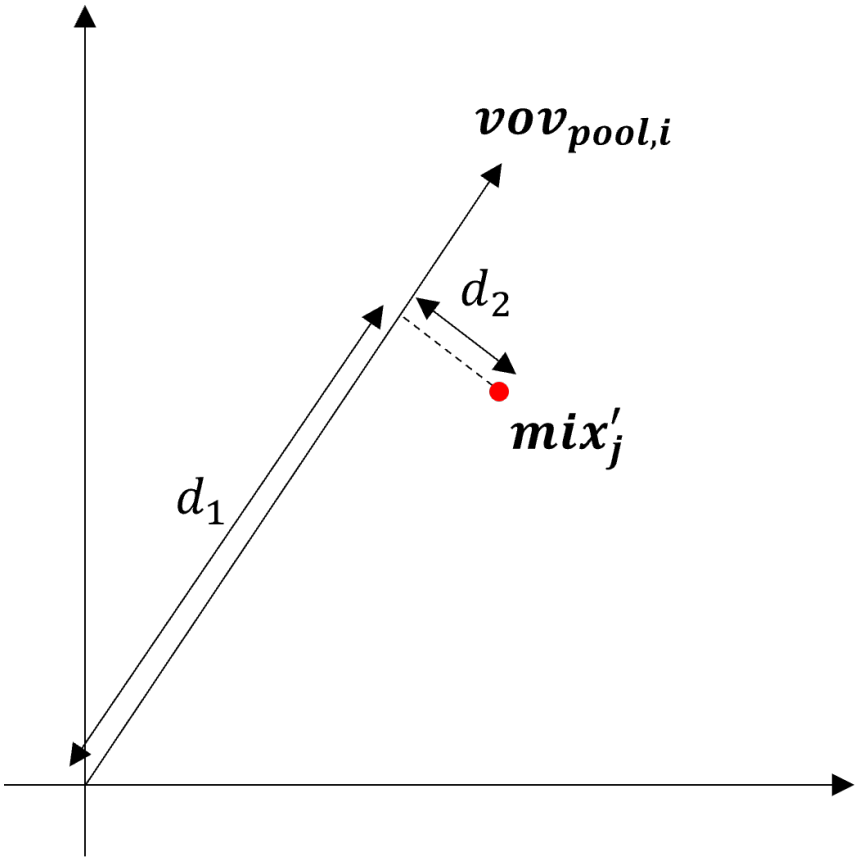
[1]China University of Geosciences College of Computer Science
[2]University of Strathclyde Department of Mechanical and Aerospace Engineering

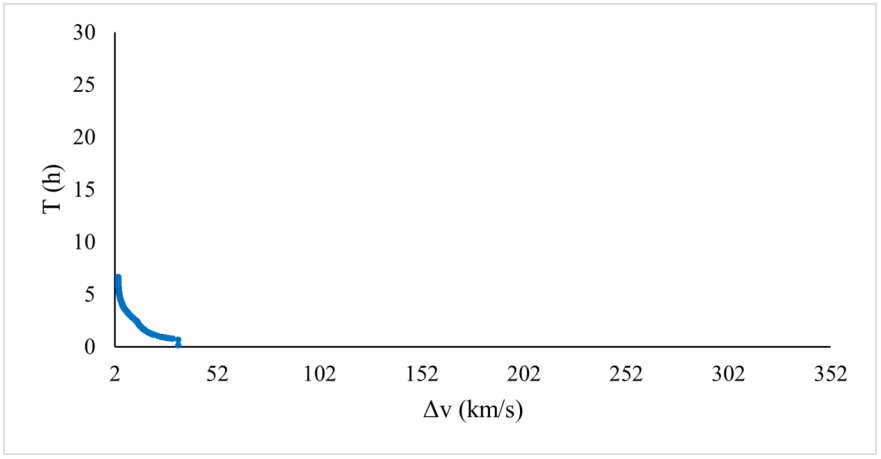
September 1, 2023

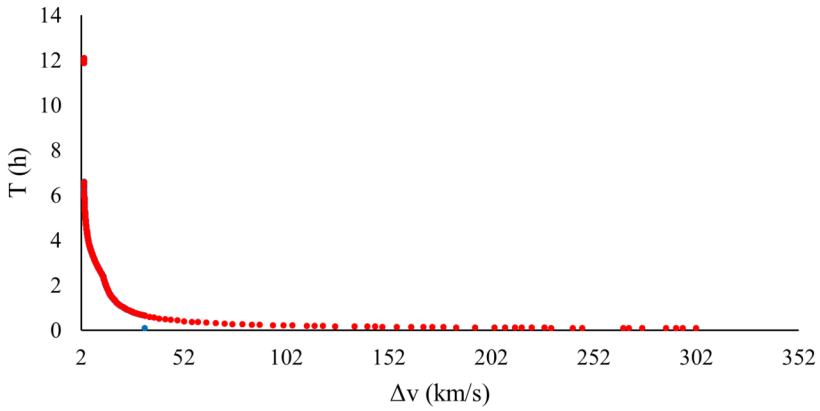
**Abstract**

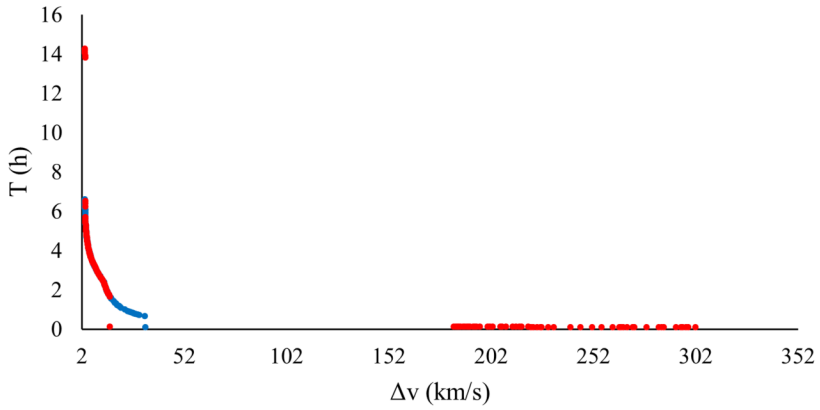
This paper presents a new Multi Agent Collaborative Search Algorithm with Adaptive Weights (named MACSAW). MACS is a memetic scheme for multi-objective optimization which contains two kind of actions, the local actions and social actions. The former explore the neighborhood of some virtual agents and the latter push the individual towards the Pareto front. On the base of the latest version of MACS, MACS2.1, we improve the old algorithm from three direction. First, a new kind of utility function is introduced to enhance the convergence. Next, a new social action process which contains more operators and adaptive parameters is embedded in MACSAW. Finally, MACS2.1 lacks the weight vectors adjustment process which leads to diversity losing in some real problems and MACSAW adds it. Further, MACSAW is compared with some state-of-art algorithms and MACS2.1 on some standard benchmarks. It gets competitive results. Two real optimization problems is tackled and the results are analyzed in details.

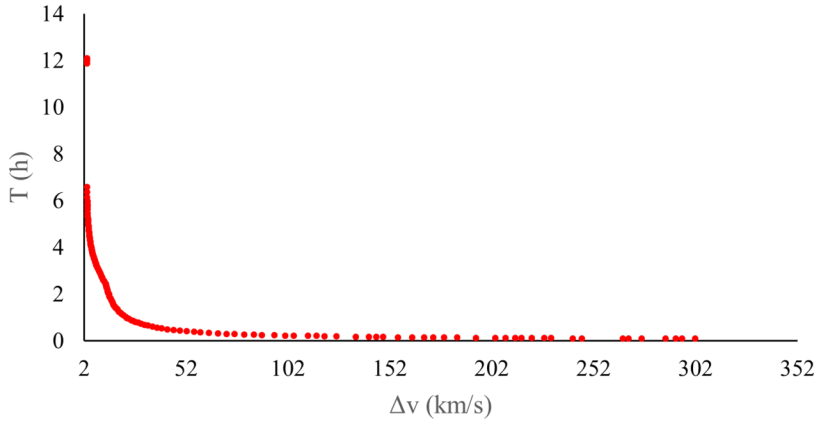


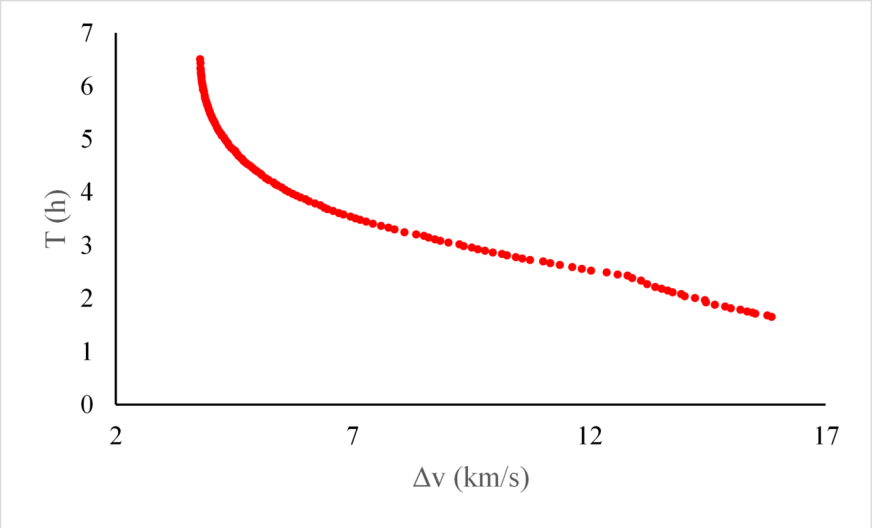


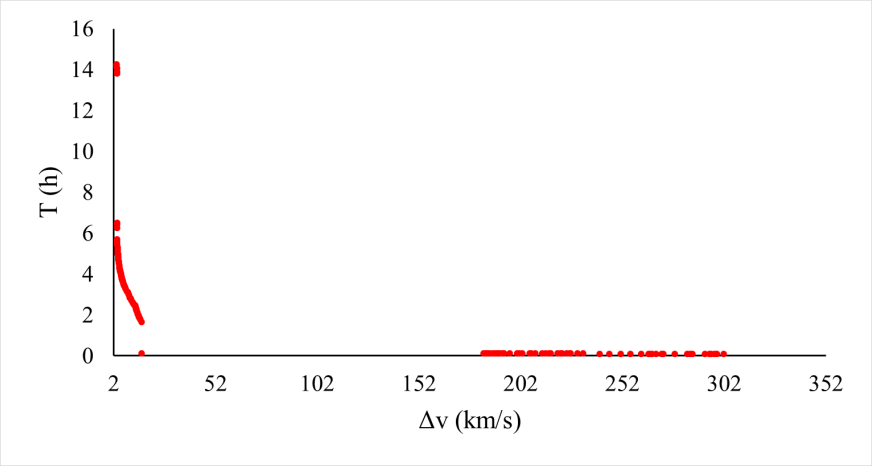



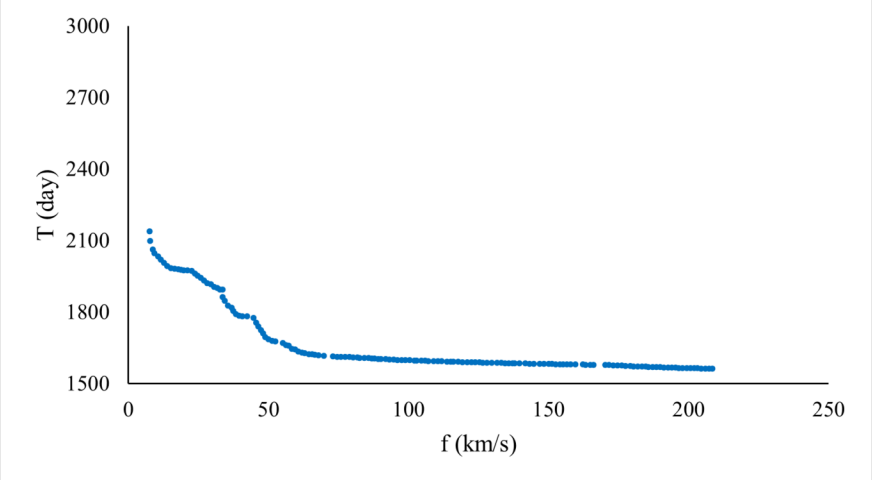


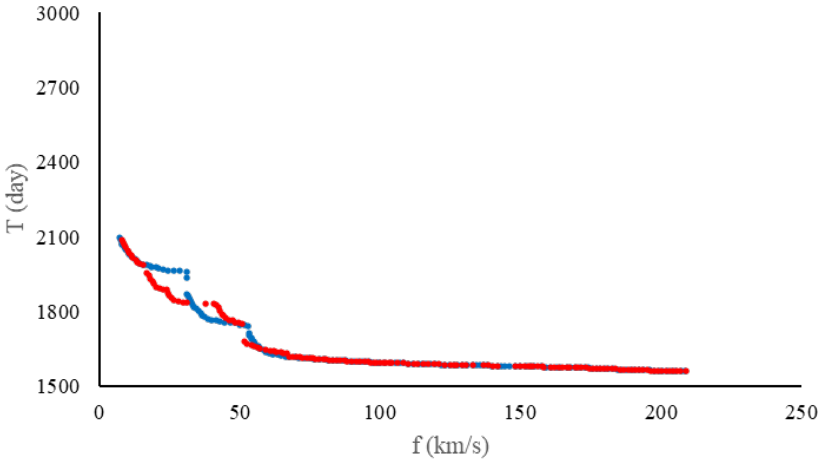



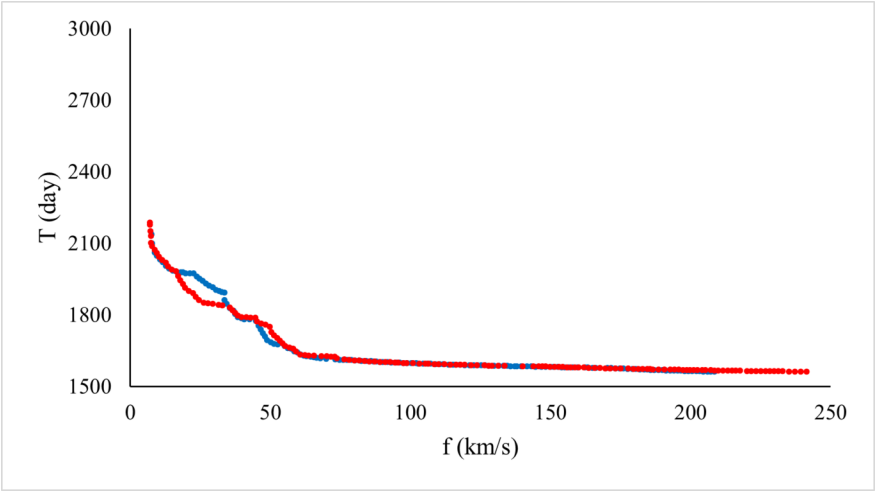



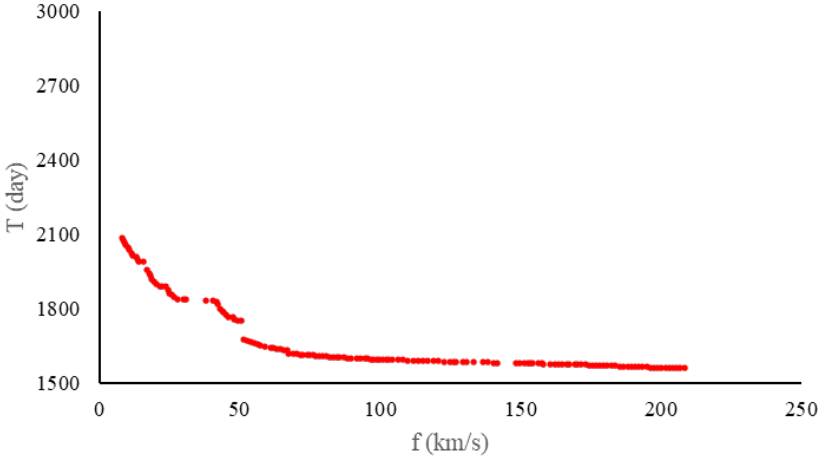


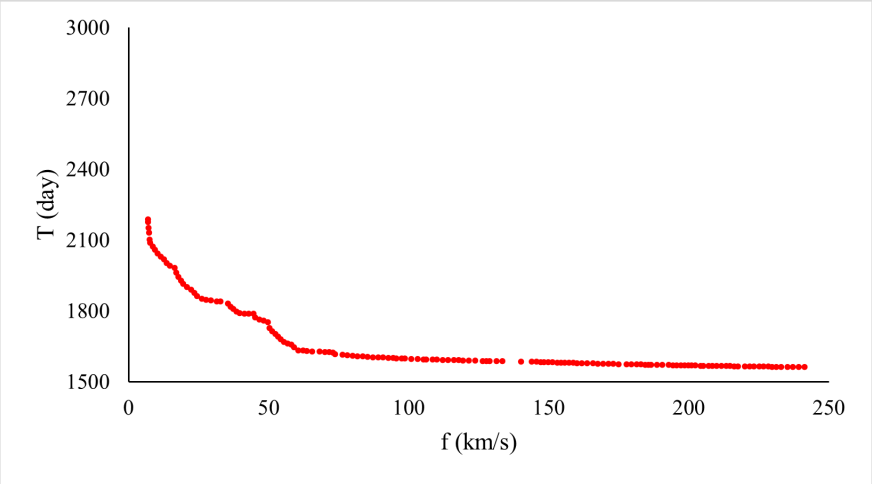



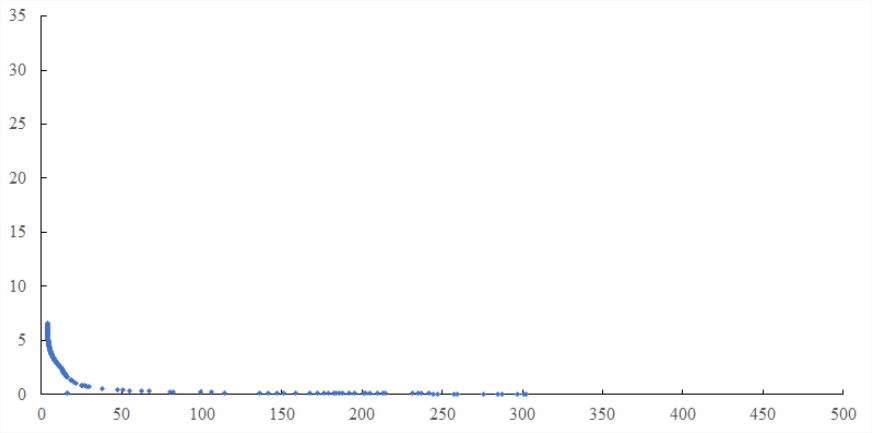





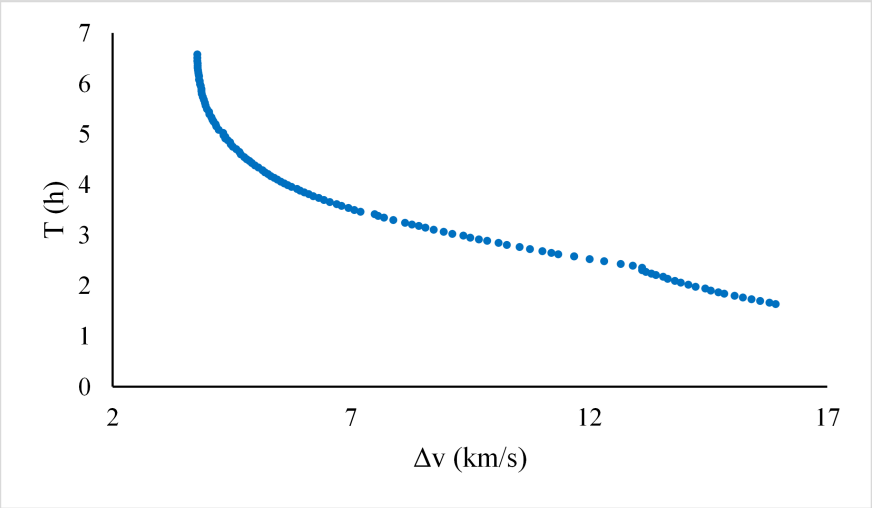



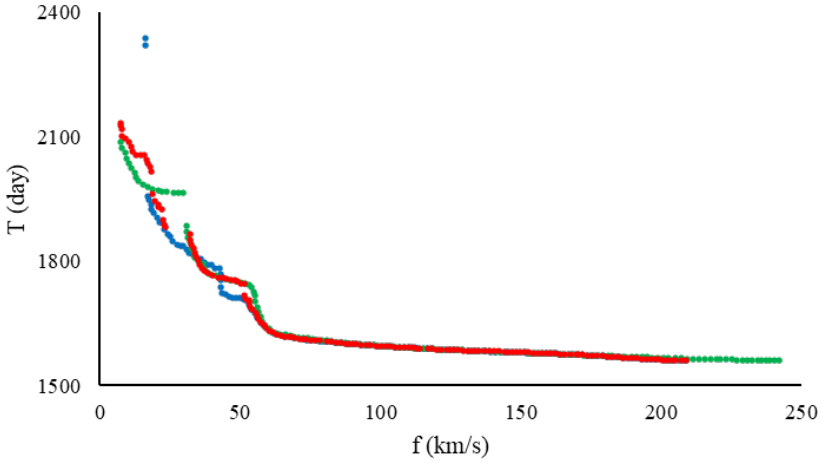



| | | |
|---|---|---|
| $\boldsymbol{off}_{i,k}$ | … | $\boldsymbol{off}_{i,k-max_{length}+1}$ |
| $rate_{i,k}$ | … | $rate_{i,k-max_{length}+1}$ |
| $op_{i,k}$ | … | $op_{i,k-max_{length}+1}$ |
| $Cr_{i,k}$ | … | $Cr_{i,k-max_{length}+1}$ |
| $F_{i,k}$ | … | $F_{i,k-max_{length}+1}$ |

| Index | 1 | 2 | ⋯ | $H-1$ | $H$ |
|---|---|---|---|---|---|
| $M_{CR}$ | $M_{CR,1}$ | $M_{CR,2}$ | ⋯ | $M_{CR,H-1}$ | $M_{CR,H}$ |
| $M_F$ | $M_{F,1}$ | $M_{F,2}$ | ⋯ | $M_{F,H-1}$ | $M_{F,H}$ |



ARTICLE TEMPLATE

# Multi Agent Collaborative Search Algorithm with Adaptive Weights

Li Cao[a,b,c], Maocai Wang[a,b], Guangming Dai[a,b], Massimiliano Vasile[c]

[a]School of Computer, China University of Geosciences,Wuhan 430074, China [b]Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan 430074, China [c]Department of Mechanical & Aerospace Engineering, University of Strathclyde, James Weir Building, 75 Montrose Street, Glasgow, G1 1XJ

**ARTICLE HISTORY**
Compiled June 13, 2023

**ABSTRACT**
This paper presents a new Multi Agent Collaborative Search Algorithm with Adaptive Weights (named MACSAW). MACS is a memetic scheme for multi-objective optimization which contains two kind of actions, the local actions and social actions. The former explore the neighborhood of some virtual agents and the latter push the individual towards the Pareto front. On the base of the latest version of MACS, MACS2.1, we improve the old algorithm from three direction. First, a new kind of utility function is introduced to enhance the convergence. Next, a new social action process which contains more operators and adaptive parameters is embedded in MACSAW. Finally, MACS2.1 lacks the weight vectors adjustment process which leads to diversity losing in some real problems and MACSAW adds it. Further, MACSAW is compared with some state-of-art algorithms and MACS2.1 on some standard benchmarks. It gets competitive results. Two real optimization problems is tackled and the results are analyzed in details.

**KEYWORDS**
evolutionary algorithm; multi-objective optimization; multi agent collaborative search; adaptive parameter; multiple operators; utility function; weight vector adjusting

## 1. Introduction

Multi Agent Collaborative Search [1] is a memetic multi-objective optimization algorithm. It has a long history of development and many gradually improved versions.

The oldest version of MACS is in 2005 [1], it has many agents which can take actions, communicate, sense and perceive the environment. Every agent performs a sequence of actions, and the algorithm takes those actions based on the local and global information. And a global archive is used to store the best achieved locations of each agent. The improved agents will be inserted in a list and communication is performed between the list and current population.

More improvement is made in 2011 [2]. In the new algorithm, every agent updates its position by a number of heuristics which are called collaborative actions. In order to avoid crowding, one restart mechanism is implemented. And the population is divided into two sub-populations, $\boldsymbol{P_k^u}$ and $\boldsymbol{P_k^l}$. The $\boldsymbol{P_k^l}$ implements a mix of actions which

CONTACT Maocai Wang Email: cugwangmc@126.com

either improve their location or explore the neighborhood. For $\boldsymbol{P_k^u}$, it implements heuristics mentioned before.

The next version is proposed in 2013 [3]. This version utilizes Pareto dominance and Tchebycheff scalarization function both as the criterion to select the solutions.

The latest version is proposed in 2019 [4] (called MACS2.1). A new archiving strategy which is named Energy Based Archiving (EBA) is embedded here to get solutions with better quality.

The MACS series display great performance on real optimization problems and standard benchmarks. However, there are still some shortcomings in MACS2.1.

First, in some real problem, MACS2.1 can't put the Pareto front to the same depth in all direction. Inappropriate allocation of computing resources may lead to this shortcoming.

Next, in social action, MACS2.1 utilizes only one operator ('DE/current-to-rand/1') to generate new individuals. In addition to lack of operators, the related parameter $CR$ and $F$ are fixed. Different kinds of operators and parameters have various characteristics. Using only one operator and fixed parameter setting may be inappropriate when we face different scenarios.

Finally, MACS2.1 initializes even vectors at the start. But if the problem has uneven Pareto front (PF), we can't get final solutions with good diversity [5].

A new algorithm is proposed in this paper to handle those drawbacks. Above all, we try to use a new kind of utility function in MACSAW to put the Pareto front to the same depth on all the directions. Secondly, a new social action which contains more operators and adaptive parameter setting is applied. Finally, a weight vector adjusting process is inserted to deal with the problems with uneven PF.

The paper has a structure as follows: In the first part, a brief description for multi-objective optimization problems is given. In the second part, the existing faults of MAC2.1, improvements for them and the new algorithm are discussed in details. Further, the new algorithm is compared with some state-of-art algorithms on the standard benchmarks, UF and ZDT. And the new algorithm is compared with original MACS2.1 on two real problems. Finally, some discussions and conclusions are made.

## 2. Problem formulation

This paper's target is to find the feasible set for solutions of the following problem [3]:

$$min_{\boldsymbol{x}\in\boldsymbol{D}} f(\boldsymbol{x}) \tag{1}$$

$\boldsymbol{D}$ is a hyperrectangle which is defined as:

$$\boldsymbol{D} = \left\{ x_j | x_j \in [b_j^l \; b_j^u] \subseteq \mathbb{R}, j = 1, ..., n \right\} \tag{2}$$

$f$ is the objective vector function:

$$f : \boldsymbol{D} \to \mathbb{R}^m, \boldsymbol{f(x)} = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_m(\boldsymbol{x})]^T \tag{3}$$

Optimality of one individual is defined by the concept of dominance: in problem (1), a individual $\boldsymbol{x} \in \boldsymbol{D}$ dominates $\boldsymbol{y} \in \boldsymbol{D}$ if $f_l(\boldsymbol{x}) \leq f_l(\boldsymbol{y})$ for all $l = 1, .., m$ and there exists one $k$ for $f_k(\boldsymbol{x}) \neq f_k(\boldsymbol{y})$. The symbol $\boldsymbol{x} \prec \boldsymbol{y}$ means that $\boldsymbol{x}$ dominates $\boldsymbol{y}$.

If a individual in $\boldsymbol{D}$ is not dominated by any other individuals in $\boldsymbol{D}$, the individual is said to be Pareto optimal. Further, all the non-dominated individuals in decision space form the Pareto set (PS) $D_p$ and its corresponding image in objective space is the Pareto Front (PF).

It is possible to calculate each individual's scalar dominance index from the concept of dominance:

$$I_d(\boldsymbol{x_i}) = |\{i^*|i, i^* \in N_p \wedge \boldsymbol{x_{i^*}} \prec \boldsymbol{x_i}\}| \tag{4}$$

where the $|\cdot|$ is the cardinality of a set and $N_p$ is the set of the indices of all the solutions. All non-dominated and feasible solutions $\boldsymbol{x_i} \in \boldsymbol{D}$ with $i \in N_p$ form the set:

$$X = \{\boldsymbol{x_i} \in \boldsymbol{D}|I_d(\boldsymbol{x_i}) = 0\} \tag{5}$$

$X$ is $D_p$'s subset, and the solution of problem (1) translates into finding the elements of $X$ by the above process.

## 3. Current defects of MAC2.1

This section is about the defects in the MAC2.1.

### *3.1. Uneven exploration in each part of PF*

If we pick the results of random three runs for Cassini problem by MACS2.1 in Figure 1, the defect of it is obviously shown.

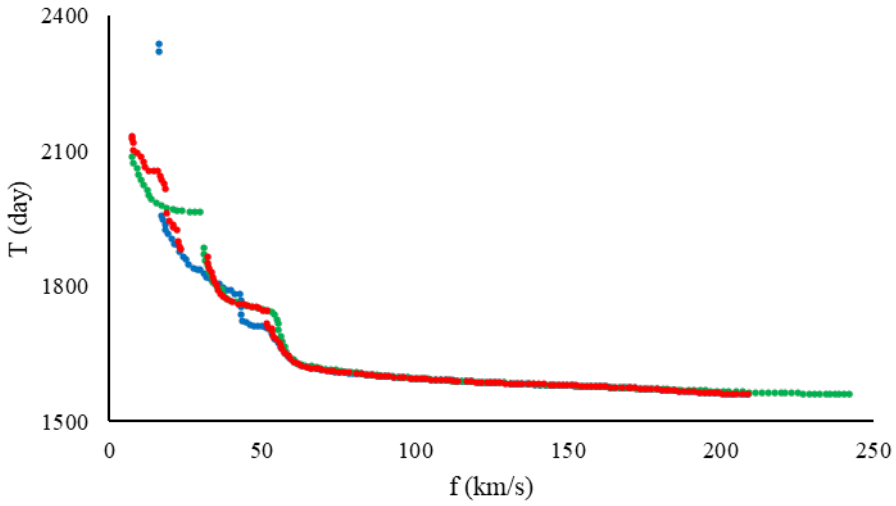



Figure 1.: MACS2.1's results of three random runs on Cassini.

The three runs are in blue, red and green respectively. In each run, MACS2.1 can't push the each part of PF into same degree. This phenomenon may damage convergence.

MACS2.1 allocates computing resources equally on all sub-problems, but the problems on different directions may need unequal resources. The unreasonable allocation

may bring uneven exploration. Further, as evolutionary process has randomness, the distribution should be adaptive because it's impossible for one sub-problem to evolve on a fixed progress.

### *3.2. Insufficiency of mutation operators and defect of fixed parameters*

Since the proposing of Differential Evolution operator (DE) [6], many new mutation strategies on the base of it are still emerging for single-objective problem optimization. For example, 'DE/target-to-best/1'[7] , 'DE/rand-to-best/2' [8] and 'DE/current-to-rand/1' [9] are among them.

Diverse DE strategies may generate various offspring. Once we face problems with different characteristics, one strategy among them may be the useful one correspondingly. Many multi-operator methods have been proposed for solving different challenges. MSaDE [10] uses three forms of mutation strategies which provide dynamic rates of exploration and exploitation by the switching between them. NDE [11] proposes a novel triangular mutation rule to balance both the exploration capability and the exploitation tendency. MLCCDE [12] applies a flexible framework which combines multiple DEs. It efficiently and significantly improves the algorithm's performance. IMODE [13] is a multi-operator DE algorithm that updates several sub-populations with different strategy. LSAOS-DE [14] considers the problem landscape information and the performance histories of the operators for dynamically selecting the most suitable operator.

Further, the parameter setting in DE operator also influences its ability. Many previous researchers have tried different settings. In original DE [6], $CR = 0.1$ is thought to be a good choice and $CR = 1$ or $CR = 0.9$ can promote convergence. In [15], it says that the interval between 0.3 and 0.9 for $CR$ seems better. And in [16], it concludes that the number of trial solutions will be reduced dramatically with $CR = 1$. As for $F$, in [17], $[0.5, 1]$ is regarded as a appropriate value range. In [16], $F$'s typical value is $0.4-0.95$ and $F = 0.9$ is a good initial value for it. Many algorithms which have parameters that can be adjusted with the evolution process keep appearing. SHADE [18], SaDE [8] and JADE [19] are among them.

As MACS2.1 [4] divides the multi-objective problem into many single-objective problems. Multi-operator strategy and adaptive parameter method may be useful here. But it only uses 'DE/current-to-rand/1' strategy in social action, in view of rich optional strategies pool, it's not enough. Further, the $CR$ and $F$ parameter in MACS2.1 are fixed, it may damage the operator's ability when we face different problems.

### *3.3. Uneven PF leads to diversity losing*

MACS2.1 is also utilized to solve the 3imp problems [4], and the results are in Figure 2.

The PF in Figure 2 is not uniform and we get solutions with bad diversity.

If the PF is irregular and the initial weight vectors are even, a set of uneven solutions will be obtained. It is a phenomenon [5] which may decrease the diversity of final results. Adjusting the vectors is necessary here.

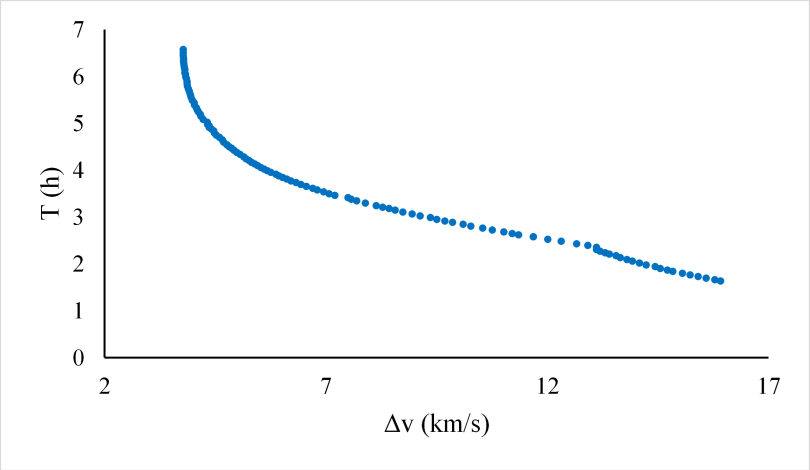



Figure 2.: MACS2.1's results of one random run on 3imp.

## 4. New algorithm: MACSAW

### *4.1. Framework of the New Algorithm*

The MACSAW's pseudo code is in Alg. 1. The details of the new algorithm are as follows.

At the start of the algorithm, some parameters are set. The maximum number of fitness evaluation is set as $n_{feval,max}$ and $max_{arch}$ is the upper limit of archive size. $n_{pop}$ is the size of population. $\rho_{ini}$, $\rho_{contr}$ and $\rho_{max,contr}$ are individual action related parameters which will be explained later. MACSAW initializes $n_{pop}$ individuals randomly in the search domain $\boldsymbol{D}$ by Latin Hypercube Sampling as initial population. The non-dominated individuals in the population are copied in the archive $\boldsymbol{A}$ to form its first version (line 4). Then a set of $n_\lambda$ $m$-dimensional unit vectors $\boldsymbol{\lambda_k}$ are generated. For bi and tri-objective space, they are sampled uniformly from a quarter of circle or eighth of a sphere. The first $m$ $\boldsymbol{\lambda_k}$ form a base in $\Re^m$. Each individual has a velocity value $V_i$ and it is initialized as zero.

The main loop starts (Line 7). The sub-problems on boundary [20] are updated at first and MACSAW guarantees their evolution in each round (Line 8). Afterwards, the sub-problems which will be solved by individual actions are decided by new utility function as Algorithm 2 (Line 11). Behind the individual action, new kind of social actions are performed on each individual gradually. Finally, two parameters that corresponds to convergence and diversity respectively are computed (line 15,16). If those parameters meet the preset conditions, the weight vectors are adjusted by Algorithm 5. It's worth noting that the $\boldsymbol{A}$ is updated when any new non-dominated solutions appears by Algorithm 8.

After individualistic and social actions, each individual that in the population is moved to the location of the element in the archive that best improves the corresponding sub-problem, unless that location is already occupied by another individual (line 18-32).

**Algorithm 1** Framework of MACSAW

Set $n_{feval,max}$, $max_{arch}$, $n_{pop}$, $\rho_{ini}$, $\rho_{contr}$, $\rho_{max,contr}$
Initialise neighbourhood size $\rho_i = \rho_{ini},\ \forall i \in 1, ..., n_{pop}$
Initialise population $P_o,\ n_{feval} = 0$
Copy the non-dominated elements of $P_0$ in the archive $A$
Initialise $n_\lambda$ vectors $\boldsymbol{\lambda_k}$ for $k \in 1, ..., n_\lambda$ such that $\|\lambda_k\| = 1$
Initialise the vectors of each individual's velocities $V_i = 0,\ \forall v \in 1, ..., n_{pop}$
**while** $n_{feval} < n_{max,feval}$ **do**
  Perform individualistic actions on boundary sub-problems through Algorithm 3
  Update archive $\boldsymbol{A}$ with non dominated elements through Algorithm 8
  Choose the sub-problems to be updated in next step by Algorithm 2
  Perform individualistic actions on chosen sub-problems through Algorithm 3
  Update archive $\boldsymbol{A}$ with non dominated elements through Algorithm 8
  Perform social actions through Algorithm 4
  Update archive $\boldsymbol{A}$ with non dominated elements through Algorithm 8
  Calculate the diversity degree of current population by Equation. 43
  Calculate the number no-dominated solutions in current population
  Adjust the weight vectors by Algorithm 5
  **if** there are at least as many individuals in the archive as objective functions **then**
    **if** the size of archive is same as objective functions number **then**
      set $n_{move}$ as number of objective functions
    **else**
      $n_{move}$ = min(size of the archive, size of current population)
    **end if**
    create a pool of $n_{move}$ individuals to be moved. Agents following exclusively one of the objectives are always chosen.
    **for** all agents in pool **do**
      find the individual in the archive which best matches to current sub-problems
      **if** the individual's position in the archive is better than current position **then**
        move the current individual to the position in the archive
        hide that position in archive for current run of social actions, to prevent multiple agents moving in the same position
      **end if**
    **end for**
  **end if**
**end while**

### *4.2. the new utility function*

The utility function was raised in MOEA/D-DRA [21] at first. It's a computing resources distribution approach.

As different sub-problems may have various computational difficulties, allocating different computing resources on them is reasonable. At first, the relative decrease for $i$th individual ($\Delta_i$) need to be calculated by (assuming the problem is a minimization problem):

$$relative\ decrease\ (\Delta_i) = \frac{old\ function\ value - new\ function\ value}{old\ function\ value} \tag{6}$$

A value $\pi_i$ is applied to decide which sub-problems will be handled next. If $\Delta_i$ is smaller than 0.001, the value of $\pi_i$ will be reduced. Specifically, the $\pi_i$ is computed by:

$$\pi_i = \begin{cases} 1 & \text{if } \Delta_i > 0.001 \\ (0.95 + 0.05\frac{\Delta_i}{0.001})\pi_i & \text{otherwise} \end{cases} \tag{7}$$

Finally, the sub-problems with the highest $\pi_i$ from randomly selected sub-problem set are chosen to be evolved.

This method focuses on the problems which have more improvement to enhance the quality of the final solutions. But in MOEA/D-DRA, it only considers scalarisation function values. In MACS2.1, the scalarisation function and dominance relationship need to be taken into account both. Thus, a new utility function is raised here to distribute resources. As individualistic actions tend to convergence [4] and improving on convergence seems can upgrade the poor performance on Cassini, we use new utility function on those actions.

The improvement on scalarisation function $Sca_i$ and dominance relationship $Dom_i$ of $i$th individual are calculated by Equation. 8 and 9 respectively.

$$Sca_i = \begin{cases} 0 & otherwise \\ \dfrac{Tch(\boldsymbol{w_i}, \boldsymbol{y_{i,t-1}}) - Tch(\boldsymbol{w_i}, \boldsymbol{y_{i,t}})}{Tch(\boldsymbol{w_i}, \boldsymbol{y_{i,t-1}})} & if\ Tch(\boldsymbol{w_i}, \boldsymbol{y_{i,t-1}}) > Tch(\boldsymbol{w_i}, \boldsymbol{y_{i,t}}) \end{cases} \tag{8}$$

$$Dom_i = \begin{cases} 0 & otherwise \\ \dfrac{\sum_{j=1}^{m}(y_{i,j,t-1} - y_{i,j,t})}{\sum_{j=1}^{m}(y_{i,j,t-1})} & if\ \boldsymbol{y_{i,t}}\ dominates\ \boldsymbol{y_{i,t-1}} \end{cases} \tag{9}$$

where $\boldsymbol{y_{i,t}}$ is the objective vector in $t$ generation of $i$th individual and $y_{i,j,t}$ is the $j$th objective value of this individual. $m$ is the objective space dimension. $\boldsymbol{w_i}$ is the related weight vector for the individual and $Tch(\boldsymbol{w_i}, \boldsymbol{y_{i,t-1}})$ is the Tchebycheff scalarisation function value. Once we get the rates, each sub-problem's total improvement rate is computed by Equation. 10.

$$Rate_i = Sca_i + Dom_i \tag{10}$$

All the $Rate_i$ are collected in a $1 \times N$ row vector $\boldsymbol{Rate}$ ($N$ is the number of sub-problems). Each column of it represents $i$th sub-problem's total improvement rate.

Then, the probability for evolving each individual needs to be decided by $\boldsymbol{Rate}$. And two kind of probabilities are defined, external probability and basic probability.

To guarantee that each sub-problem will be improved, The basic probability ($P_{i\_basic}$) is assigned to each individual equally by Equation. 11. The sum of $P_{i\_basic}$ is

$P$, and its a per-defined value in $[0, 1]$.

$$P_{i\_basic} = \frac{P}{N} \tag{11}$$

External probability for $i$th individual ($P_{i\_external}$) is calculated by Equation. 12. This part is employed to focus on the sub-problems which get better historical improvement. The $Rate(1, i)$ is the $1 \times i$ item for $\boldsymbol{Rate}$ and the $i$th individual's total improvement rate.

$$P_{i\_external} = (1 - P) \times \frac{Rate(1, i)}{\sum_{j=1}^{N}(Rate(1, i))} \tag{12}$$

Finally, $i$th individual's probability to be picked is Equation. 13.

$$P_i = P_{i\_basic} + P_{i\_external} \tag{13}$$

The sum of all the probabilities is 1 and whether the $i$th sub-problem will be evolved in next round is decided by $P_i$ (line 5 in Algorithm. 2).

**Algorithm 2** Computing resources allocating by new utility function

1: Calculate the improvement rate on scalarisation function $Sca_i$ and dominance relationship $Dom_i$ of each individual by Equation. 8 and 9
2: Sum the total rate of each individual $Rate_i$ by Equation. 10
3: Allocate the basic probability evenly to each sub-problems by Equation. 11
4: Distribute the external probability by the improvement values $\boldsymbol{Rate}$ as Equation. 12
5: Choose the sub-problems to be evolved in the next round individualistic action by each problems' probability

### *4.3. Individualistic actions*

The pseudo-code for individualistic actions is in Algorithm. 3. This part implements inertia, pattern search and differential evolution sequentially until an improvement happens (a new solution which satisfies Tchebycheff or dominant criterion is generated). This part is same as MACS2.1 [4].

#### *4.3.1. Inertia*

The previous moves define a direction ($\boldsymbol{V_i}$) in the decision space and inertia generates a new sample on this direction. The $i$th individual's trial position is:

$$\boldsymbol{x_{trial}} = \boldsymbol{x_i} + \alpha \boldsymbol{V_i} \tag{14}$$

**Algorithm 3** Individualistic actions with adaptive scalarisation method

**for** $i = 1 : n_{pop}$ **do**
  Set improved=FALSE
  **if** $\|\boldsymbol{V_i}\| \neq 0$ **then**
    Perform Inertia move
    Evaluate move
    **if** successful **then**
      set improved=TRUE
    **end if**
  **end if**
  **if** not improved **then**
    counter=0
    **while** $counter \leq max\_pat\_search\_dirs$ & not improved **do**
      counter=counter+1
      Pick random direction
      Perform Pattern Search
      Evaluate move
      **if** successful **then**
        set improved=TRUE
        $\boldsymbol{V_i} = x_{i,old} - x_i$
      **end if**
    **end while**
  **end if**
  **if** not improved **then**
    Perform Differential Evolution
    Evaluate move
    **if** successful **then**
      set improved=TRUE
    **end if**
  **end if**
  **if** not improved **then**
    Contract $\rho_i$
    **if** $\rho_i$ has contracted more than $\rho_{max,contr}$ times **then**
      $\rho_i = \rho_{ini}$
    **else**
      De-contract $\rho_i$ unless this would cause $\rho_i$ to be greater than $\rho_{ini}$
    **end if**
  **end if**
**end for**

where $\alpha$ is a random number between 0 and 1. Once $\boldsymbol{x_{trial}}$ exceeds the feasible area $D$, the $\alpha$ is contracted with a simple backtracking procedure to ensure that $\boldsymbol{x_{trial}}$ falls in $\boldsymbol{D}$. Moreover, if $\boldsymbol{x_i}$'s one component which is lower than $n$ is equal to either their lower or upper limit and $\boldsymbol{x_i} + \alpha \boldsymbol{V_i}$ is outside $\boldsymbol{D}$, those components of $\boldsymbol{V_i}$ are set as zero before the backtracking procedure. Target of this part is enhancing the exploration for the boundary in the search space.

*4.3.2. Pattern Search*

If the improvement doesn't happen or the inertia is not executed ($\boldsymbol{V_i} = 0$), the pattern search is implemented. This part changes only one randomly component $j$ in $\boldsymbol{x_i}$ once. For the $i$th individual, the new position ($\boldsymbol{x_{trial}}$) is same as $\boldsymbol{x_i}$, expect for the $j$th component:

$$x_{trial,j} = x_{ij} + \alpha \Delta_j \rho_i \tag{15}$$

where $\alpha$ is a random number between $-1$ and 1, $\Delta_j$ is the difference between the upper and lower boundaries for $j$th variable and $\rho_i$ is the size of the hyperrectangle which has $\boldsymbol{x_i}$ as its center. If the exploration on direction $\Delta_j \rho_i$ fails, the new direction $-sign(\alpha)\beta\Delta_j\rho_i$ is utilized here with a randomly chosen $\beta$ between 0 and 1. Finally, if all the attempts fail, a new random direction which is different from all the previous directions is chosen.

This strategy is repeated until the predefined maximum number of directions have been explored or an improvement happens. The maximum number of directions will be explored is defined as:

$$max\_dirs = round\left(n - (n-1)\frac{curr\_arch\_size}{max\_arch\_size}\right) \tag{16}$$

where $max\_dirs$ is the maximum number of dimensions to scan, $n$ is the number of coordinates, $curr\_arch\_size$ is the size of current archive and $max\_arch\_size$ is the predefined maximum size of the archive. It is worth noting that $max\_dirs$ is changing with the evolution process. Once a better individual is acquired, the $\boldsymbol{V_i}$ is calculated by it.

*4.3.3. Differential Evolution*

Once pattern search also fails, individualistic action enters differential evolution stage. The displacement has the following format:

$$\boldsymbol{dx_i} = \alpha \boldsymbol{e}\left((\boldsymbol{x_i} - \boldsymbol{x_{i_1}}) + F(\boldsymbol{x_{i_2}} - \boldsymbol{x_{i_3}})\right) \tag{17}$$

where $\alpha$ is a random number between 0 and 1, $F$ is a predefined constant. $\boldsymbol{x_i}$ is the individual to be evolved and $\boldsymbol{x_{i_{1-3}}}$ are three individuals picked from current population randomly. $\boldsymbol{e}$ is a mask vector which is decided by:

$$e_j = \begin{cases} 1 & \text{if } \alpha_2 < CR \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

where $\alpha_2$ is a random number between 0 and 1, and $CR$ is another predefined constant. Then, the differential evolution move's trial position is defined as follows:

$$\boldsymbol{x_{trial}} = \boldsymbol{x_i} + \boldsymbol{dx_i} \tag{19}$$

If $\boldsymbol{x_{trial}}$ is out of boundary after Differential Evolution, the algorithm reduce the $\alpha$ or some components of $\boldsymbol{dx_i}$.

#### *4.3.4. Local neighborhood size management*

If all the previous actions fail, the local neighborhood size $\rho_i$ is reduced by a predefined constant $\rho_{contr}$. At extreme situation, if all the actions are still invalid, $\rho_i$ is reset to $\rho_{ini}$ after a predefined maximum number of contractions $\rho_{max,contr}$. Conversely, $\rho_i$ is increased by $\rho_{contr}$ if one action is successful and the upper boundary of $\rho_i$ is $\rho_{ini}$.

### *4.4. Social actions with multiple operators*

As MACS algorithms divide multi-objective problem into many single-objective problems and try to solve them respectively, some state-of-art methods in single-objective algorithms may be useful. As the DE part in individualistic action is not always executed, we only improve social action here.

Among the latest algorithms, IMODE [13] has relatively simple structure and excellent performance in operators choosing. And SHADE [18] is relatively useful and easy to understand in parameters adjusting. We want to transform their method and embed them into MACS.

The IMODE utilizes three mutation operators ('DE/current-to-$\phi$best with archive/1', 'DE/current-to-$\phi$best without archive/1' and 'DE weighted-rand-to-$\phi$best'). At the start, each operator generates same number of solutions. Then two parameters, diversity and quality, are computed for deciding the number of solutions that will be generated by each operator in the next round. The diversity of the individuals obtained by each operator are computed by:

$$D_i = \frac{1}{NP_i}(\sum_{i=1}^{NP_i} dis(\vec{x}_{i,j} - \vec{x}_i^{best})) \tag{20}$$

where $D_i$ is the diversity for $i$th operator, and $dis(\vec{x}_{i,j} - \vec{x}_i^{best})$ is the Euclidean distance between $j$th solution and the best one obtained by $i$th operator. $NP_i$ is the number of individual generated by $i$th operator.

Further, the diversity rate for the $i$th operator $DR_i$ can be formulated as follows:

$$DR_i = \frac{D_i}{\sum_{i=1}^{3} D_i} \tag{21}$$

As for quality rate of the $i$th operator, it has the following form:

$$QR_i = \frac{fit_i^{best}}{\sum_{i=1}^{3} fit_i^{best}} \tag{22}$$

where $fit_i^{best}$ is the best objective function value obtained by operator $i$th and $QR_i$ is its quality rate.

The improvement rate value for $i$th operator ($IRV_i$) is:

$$IRV_i = (1 - QR_i) + DR_i \tag{23}$$

Finally, the number of solutions that will be generated by $i$th operator ($NP_i$) next is defined as follows:

$$NP_i = max\left(0.1, min\left(0.9, \frac{IRV_i}{\sum_{i=1}^{3} IRV_i}\right)\right) \times NP \tag{24}$$

where $NP$ is the total number of solutions generated by all the operators.

The SHADE uses a historical memory for deciding DE control parameters. The memory has following structure:

| Index | 1 | 2 | ⋯ | $H-1$ | $H$ |
|---|---|---|---|---|---|
| $M_{CR}$ | $M_{CR,1}$ | $M_{CR,2}$ | ⋯ | $M_{CR,H-1}$ | $M_{CR,H}$ |
| $M_F$ | $M_{F,1}$ | $M_{F,2}$ | ⋯ | $M_{F,H-1}$ | $M_{F,H}$ |

Figure 3.: structure of historical memory

Before evolution, for the individual $\boldsymbol{x_i}$, a index $r_i$ is randomly picked from $[1, H]$. The $CR_i$ and $F_i$ are decided by the Equation. 25 and 26. $M_{CR,r_i}$ and $M_{F,r_i}$ are related value in $r_i$ column of memory.

$$CR_i = randn_i(M_{CR,r_i}, 0.1) \tag{25}$$

$$F_i = randc_i(M_{F,r_i}, 0.1) \tag{26}$$

where $randn_i(\mu, \sigma^2)$ and $randc_i(\mu, \sigma^2)$ are randomly sampling from normal and Cauchy distributions with mean $\mu$ and variance $\sigma^2$. After the generation of new individuals, the success attempts' related $CR_i$ and $F_i$ are recorded as $S_{CR}$ and $S_F$. They are utilized to update the memory as Equation. 27 and 28. $k$ has a initial value 1 and it increases with the generation of new offspring. $k$ will be set to 1 if $k$>$H$.

$$M_{CR,k,G+1} = \begin{cases} mean_{WA}(S_{CR}) & \text{if } S_{CR} \neq \phi \\ M_{CR,k,G} & \text{otherwise} \end{cases} \tag{27}$$

$$M_{F,k,G+1} = \begin{cases} mean_{WL}(S_F) & \text{if } S_F \neq \phi \\ M_{F,k,G} & \text{otherwise} \end{cases} \tag{28}$$

The index $k$ ($1 \leq k \leq H$) determines the position that need to be updated. The $mean_{WA}(S_{CR})$ and $mean_{WL}(S_F)$ formulated as follows:

$$mean_{WA}(S_{CR}) = \sum_{k=1}^{|S_{CR}|} \omega_k \cdot S_{CR,k} \tag{29}$$

$$mean_{WL}(S_F) = \frac{\sum_{k=1}^{|S_F|} \omega_k \cdot S_{F,k}^2}{\sum_{k=1}^{|S_F|} \omega_k \cdot S_{F,k}} \tag{30}$$

$$\omega_k = \frac{\Delta_{f_k}}{\sum_{k=1}^{|S_{CR}|} \Delta_{f_k}} \tag{31}$$

where $\Delta_{f_k} = |f(\boldsymbol{u_{k,G}}) - f(\boldsymbol{x_{k,G}})|$ is the the amount of improvement.

*4.4.1. mutation operators determining process*

In MACSAW, we introduce three mutation operators approach as IMODE and convert it to solve multi-objective problems. A detailed description for new social action is given in the following part.

For IMODE, it only focuses on one objective function and holds one archive. But MACSAW needs to handle many sub-problems and more objective functions. It also has a archive for all the problems. We made a revision here.

The MACSAW's three mutation operators have the following forms in details:

(1) DE/current-to-$\phi$best with archive/1 $\boldsymbol{v^i} = \boldsymbol{x^i} + F_{i,k} \times (\boldsymbol{x^{\phi,i}} - \boldsymbol{x^i} + \boldsymbol{x^{i,1}} - \boldsymbol{x^{i,2}})$
(2) DE/current-to-$\phi$best without archive/1 $\boldsymbol{v^i} = \boldsymbol{x^i} + F_{i,k} \times (\boldsymbol{x^{\phi,i}} - \boldsymbol{x^i} + \boldsymbol{x^{i,1}} - \boldsymbol{x^{i,3}})$
(3) DE weighted-rand-to-$\phi$best $\boldsymbol{v^i} = F_{i,k} \times \boldsymbol{x^1} + (\boldsymbol{x^{\phi,i}} - \boldsymbol{x^{i,3}})$

where $\boldsymbol{x^i}$ is the $i$th individual's value in decision space. To elevate the offspring's quality by picking parents from diverse sources, $\boldsymbol{x^1}$ and $\boldsymbol{x^3}$ are from the archive $\boldsymbol{A}$ or current population $\boldsymbol{P}$. The probability of choosing individuals from the archive or the population is formulated as:

$$p_{arch\_vs\_pop} = 1 - e^{-\frac{current\_size\_archive}{num\_individual}} \tag{32}$$

where $p_{arch\_vs\_pop}$ is the possibility of choosing individuals from archive in social actions, *current_size_archive* is the size of the archive. And *num_individual* is the size of current population.

Then $\boldsymbol{x^2}$ is randomly chosen from the mix consists of $\boldsymbol{A}$ and $\boldsymbol{P}$ (named it $\boldsymbol{M}$). $\boldsymbol{x^{\phi,i}}$ is randomly picked from the best 10% solutions in $\boldsymbol{M}$ on $i$th sub-problem with Tchebycheff scalarizing function. $F_{i,k}$ is mutation parameter at $k$th round and decided by historically obtained solutions on $i$th sub-problem.

IMODE has the same mutation operators structure as our new algorithm. To compare them, the meanings for the items in those operators in MACSAW and IMODE are listed in Table. 1.

Table 1.: The meanings for those items in MACSAW and IMODE

| | $\boldsymbol{x^i}$ | $\boldsymbol{x^{\phi,i}}$ | $\boldsymbol{x^1, x^3}$ | $\boldsymbol{x^2}$ |
|---|---|---|---|---|
| IMODE | the individual that will evolve | one individual from the best 10% solutions in the whole population | two random individuals in the whole population | one random individual from the union of the whole population and archive |
| MACSAW | the individual that will evolve | one random individual from the best 10% solutions in the union of the archive and population | two random individuals in archive or population | one random individual from the union of the archive and population |

$\boldsymbol{v^i}$ is the trial vector and enters the binomial crossover to get the offspring. It works as:

$$x_j^{'i} = \begin{cases} v_j^i, \ if \ \text{rand} \leq Cr_{i,k} \\ x_j^i, \ \text{otherwise} \end{cases} \tag{33}$$

where $j \in 1, ..., n$ and $rand$ is a random number from $[0, 1]$. $x_j^i$ is from the $i$th individual and $x_j^{'i}$ is $j$th component of offspring. $Cr_{i,k}$ is mutation parameter at $k$th round and got by historically obtained solutions on $i$th sub-problem, too.

Once offspring on $i$th sub-problem is better than old solution on scalarisation function or dominance relationship at $k$th round, its total improvement rate $rate_{i,k}$ will be calculated by Equation 8, 9 and 10. The offspring's value in decision space ($\boldsymbol{off_{i,k}}$), the operator number ($op_{i,k}$) that has been executed at this time, the $F_{i,k}$, the $Cr_{i,k}$ and the $rate_{i,k}$ are stored in the top of a sliding window $\boldsymbol{social\_archive_i}$. It has the following structure in Figure. 4 . The sliding window has a maximum length $max_{length}$, and it is set as $0.5N$ ($N$ is the population size). It's a queue. If the window size will bigger than $max_{length}$ after inserting, old record will be deleted from tail.

| | | |
|---|---|---|
| $\boldsymbol{off_{i,k}}$ | $\cdots$ | $\boldsymbol{off_{i,k-max_{length}+1}}$ |
| $rate_{i,k}$ | $\cdots$ | $rate_{i,k-max_{length}+1}$ |
| $op_{i,k}$ | $\cdots$ | $op_{i,k-max_{length}+1}$ |
| $Cr_{i,k}$ | $\cdots$ | $Cr_{i,k-max_{length}+1}$ |
| $F_{i,k}$ | $\cdots$ | $F_{i,k-max_{length}+1}$ |

Figure 4.: structure of $\boldsymbol{social\_archive_i}$

Before the $(k+1)$th round social action, the selected probability for each operator on $i$th sub-problem is computed by following process. The total $rate$ value for $j$th mutation operator ($sum_{i,j}$) and the sum of row 2 in $\boldsymbol{social\_archive_i}$ ($sum_i$) need to be computed at first. Then, the probability of selecting the $j$th mutation operators at the $(k+1)$th round, $p_{j,k+1}$, can be computed by:

$$\begin{cases} p_{j,k+1} = max\left(0.1, min\left(0.9, \dfrac{sum_{i,j}}{sum_i}\right)\right) & if \text{ all the operators appear} \\ p_{j,k+1} = \dfrac{1}{3} & otherwise \end{cases} \tag{34}$$

It is noteworthy that the basic value 0.1 is assigned to each operator. Before all the operators appearing in $\boldsymbol{social\_archive_i}$, $\frac{1}{3}$ is assigned to $p_{1,k+1} - p_{3,k+1}$. After attaining the probability, which operator will be picked at $(k+1)$th round is determined by the value $p_{1,k+1}$, $p_{2,k+1}$ and $p_{3,k+1}$ (line 2 in Algorithm. 4).

*4.4.2. related parameters determining process*

SHADE is modified for MACS2.1, the $w$ in SHADE is relative improvement in objective space. In MACSAW, $w_{i,t}$ is $t$th item's proportion in total relative improvement in $\boldsymbol{social\_archive_i}$ (Equation. 35).

On the one hand, if a better solution is got on $i$th sub-problem at $k$th round, the following procedure is applied to decide the mutation related parameters $CR_{i,k+1}$ and $F_{i,k+1}$ at the $(k+1)$th round. The auxiliary parameter $w$ is set here at first for calculating $MCR_{i,k}$ and $MF_{i,k}$ for next step as Equation. 35, 36 and 37.

$$w_{i,t} = \frac{rate_{i,t}}{\sum_{j=k-max_{length}+1}^{k} rate_{i,j}} \tag{35}$$

$$MCR_{i,k} = \sum_{t=k-max_{length}+1}^{k} (w_{i,t} \times Cr_{i,t}) \tag{36}$$

$$MF_{i,k} = \frac{\sum_{t=k-max_{length}+1}^{k}(w_{i,t} \times {F_{i,k}}^2)}{\sum_{t=k-max_{length}+1}^{k}(w_{i,t} \times F_{i,t})} \tag{37}$$

$Cr_{i,k}$ and $F_{i,k}$ are successful attempt's related parameters at the $k$th round. Afterward, the decision process for $Cr_{i,k}$ and $F_{i,k}$ is described in detail.

$CR_{i,k+1}$ and $F_{i,k+1}$ are determined by the following Equation 38 and 39. In addition, if $F_{i,k+1}$ is small than 0, Equation 39 will be repeated until a positive value is got.

$$CR_{i,k+1} = max(0, min(1, randn_i(MCR_{i,k}, 0.1))) \tag{38}$$

$$F_{i,k+1} = randc_i(MF_{i,k}, 0.1) \tag{39}$$

On the other hand, if the algorithm fails on $i$th sub-problem at the $k$th round. $CR_{i,k+1}$ and $F_{i,k+1}$ are computed by the following Equation. 40 and 41. Similarly, Equation. 41 will be iterated until a positive $F_{i,k+1}$ is obtained.

$$CR_{i,k+1} = max(0, min(1, randn_i(0.5, 0.1))) \tag{40}$$

$$F_{i,k+1} = randc_i(0.5, 0.1) \tag{41}$$

In IMODE, as it has only one objective function, one memory is enough for operator picking and parameter deciding. For MACSAW, $N$ memories are needed because we have to choose operator and parameter respectively for each sub-problem. Further, if one attempt fails, we will choose $CR$ and $F$ by $randn_i(0.5, 0.1)$ and $randc_i(0.5, 0.1)$ to generate the parameters for next round. The distributions with fixed expectation are used as a restart process.

**Algorithm 4** Social actions with multiple operators

1: **for** $i = 1 : n_{pop}$ **do**
2: Determine the mutation operator that will be used applied by Equation. 34
3: Determine the needed parameter by Equation. 38, 39 or Equation. 40, 41
4: Get offspring by picked strategy
5: **if** get better offspring than $i$th individual **then**
6: Update the $i$th individual
7: Update $\boldsymbol{social\_archive_i}$
8: **end if**
9: **end for**
10: add candidate solutions in archive through Algorithm 8

### *4.5. Weight vector adjustment*

MACS2.1 initializes even weight vectors, but they can't ensure that the final solutions are uniform if the PF is not even [22] (Figure. 2 shows this conclusion clearly).

The concept of weight vector is put forward in MOEA/D [23] which has even initial vectors. After its appearing, many new algorithms combine it with vector adjustment process to handle problems with uneven PF. MOEA/D-AWA [22] is a pioneering algorithm which utilizes WS-transformation. It's a creative method that generates new weight vectors by solutions in archive with high quality. MOEA/D-URAW [24] is on the base of MOEA/D-AWA and uses a new weight initialization approach. MOEA/D-VOV [25] utilizes virtual objective vectors to supplement the PF estimation. MOEA/D-2ADV [26] has two types of adjustments for the direction vectors. Apart from MOEA/D-2ADV which directly updates vectors by current population, other algorithms adjust vectors by a external archive.

Among them, MOEA/D-VOV is the latest algorithm which arranges the weight vector set appropriately for the PF. It applies an arrangement method for weight vectors with virtual objective vectors supplementing the PF estimation. We try to embed the weight vectors arrangement method here.

But all the previous algorithms have one defect, the weight vector adjusting points are fixed by predefined parameter (the adjusting points for those algorithms are in Table. 2). It's unreasonable. On the one hand, assuming that the current archive is stable with wrong weight vectors but adjusting doesn't happen, more computing resources are wasted on the wrong search directions. On the other hand, if the adjusting starts when the current archive is immature, wrong search directions may be generated and reduce the quality of final solutions. So the adapting process should start when the archive is in a relatively steady state.

Table 2.: The adjusting points for those algorithms

| | the adjusting point |
|---|---|
| MOEA/D-AWA | Adjusting happens before $raat_evol \times G_{max}$ generation and at every $wag$ generation.<br>$G_{max}$ is the mix generation for evolution.<br>$raat_evol$ and $wag$ are predefined value. |
| MOEA/D-URAW | Adjusting happens before $90\% \times G_{max}$ generation<br>and at every $5\% \times G_{max}$. |
| MOEA/D-VOV | Adjusting happens at every $N \times G$ generation.<br>$N$ is the population size and $G$ is a predefined value. |
| MOEA/D-2ADV | Adjustment for the number of direction vectors may happen at every $\phi_1$ generation and adjustment for the positions of ineffective direction vectors may happen at every $\phi_2$ generation.<br>$\phi_1$ and $\phi_2$ are predefined value. |

How can we judge the state of current archive? Maybe the quality measurement parameters for solution or population can be utilized for deciding the state.

Many parameters have been proposed. MaOEA-DDFC [27] uses directional density on the project hyperplane for choosing the survive individuals. MaOEA/IGD [28] applies new $IGD$ parameter ($IGD^+$) to pick solutions. In PeEA [29], k-NN and dimensionality margin distance are used for diversity measuring. MSEA [30] divides the evolution into three stages by diversity and convergence parameters.

Here, we choose the approach in MSEA for judging the state as it is easier for un-

derstanding and more suitable for our propose. It is a creative optimization algorithm which picks solutions with different criteria in different stages of evolution. In stage 3 of it, the population is in good convergence and diversity. Then they only need to be fine-tuned. The parameters for the stage are the number of no-dominated solutions and diversity degree. The degree for individual $\boldsymbol{x}$ is calculated by Equation. 42 and 43. At first, normalized objective value of $\boldsymbol{x}$ need to be calculated by Equation. 42.

$$f'(x)_i = \frac{f(x)_i - z_{i,min}}{z_{i,max} - z_{i,min}} \tag{42}$$

where $z_{i,min}$ and $z_{i,max}$ represent the smallest and biggest value respectively in objective space on $i$th dimension of current population. $f(x)_i$ is the value in objective space on $i$th dimension of $\boldsymbol{x}$. $\boldsymbol{f(x)}$ and $\boldsymbol{f'(x)}$ is the objective vector and normalized objective vector for $\boldsymbol{x}$ separately.

$$Div(\boldsymbol{x}) = \left\|\boldsymbol{f'(x)} - \boldsymbol{f'(x_1)}\right\| + \epsilon \left\|\boldsymbol{f'(x)} - \boldsymbol{f'(x_2)}\right\| \tag{43}$$

where $\|\boldsymbol{f'(x)} - \boldsymbol{f'(x_1)}\|$ is the Euclidean distance between the normalized objective vector for $\boldsymbol{x}$ and its closest neighbor $\boldsymbol{x_1}$'s normalized objective vector in current population, and $\|\boldsymbol{f'(x)} - \boldsymbol{f'(x_2)}\|$ represents the Euclidean distance between the normalized objective vector for $\boldsymbol{x}$ and its second closest neighbor $\boldsymbol{x_2}$'s normalized objective vector in current population. $\epsilon$ is a predefined value ($\epsilon = 0.01$ is used here).

If the population $\boldsymbol{P}$ doesn't contain dominated individual and minimum diversity degree of all the individuals in $\boldsymbol{P}$ is smaller than half the existing maximum diversity degree, $\boldsymbol{P}$ is considered to have entered the stage 3. This criterion can be expressed as Equation. 44.

$$\begin{cases} \textit{There is no dominated individual in } \boldsymbol{P} \\ min_{\boldsymbol{x}\in\boldsymbol{P}} Div(\boldsymbol{x}) < \frac{1}{2} max_{\boldsymbol{x}\in\boldsymbol{P}} Div(\boldsymbol{x}) \end{cases} \tag{44}$$

As the population is directly relevant to the weight vectors instead of archive. We use same strategy to judge whether the population is stable for weight vector adjusting. Its pseudocode of weight vector adjusting process is in Algorithm. 5.

At first, the state of current population $\boldsymbol{P}$ is tested. For convergence, we calculate the number of dominated solutions. When there is no dominated solution in population, we consider that the population is relatively unchanging in convergence. For diversity, different from MOEA/D-VOV, the sum of diversity degree for each individual ($sum_{div,new}$ in line 8) is used. The population is regarded as stable in diversity when the relative changing of the sum in neighboring round is between 0.99 and 1.01. If it is steady in convergence (line 2-4 in Algorithm. 5) and in diversity (line 5-11 in Algorithm. 5). Then, the adjusting starts. The representative objective vector set is picked by Algorithm. 6 and weight vectors are updated by Algorithm. 7.

**Algorithm 5** weight vector adjusting process

1: Set two sign *cov* and *div* as 0
2: **if** there is no dominated individual in current population $\boldsymbol{P}$ **then**
3: Set *cov* as 1
4: **end if**
5: **if** this is the first time that enters Algorithm 5 **then**
6: Set $sum_{div,old}$ as $+\infty$
7: **end if**
8: Sum the diversity degree of each individual in $\boldsymbol{P}$ as $sum_{div,new}$ by Equation. 43 and 44
9: **if** $1.01 > \frac{sum_{div,new}}{sum_{div,old}} > 0.99$ **then**
10: State *div* as 1
11: **end if**
12: Assign $sum_{div,new}$ to $sum_{div,old}$
13: **if** $cov == 1$ and $div == 1$ **then**
14: Generate virtual objective vector set $\boldsymbol{vov}$ by Algorithm 6
15: Pick the representative objective vector set $\boldsymbol{vov_{rep}}$ and update the weight vectors for next generation by Algorithm 7
16: **end if**

In MOEA/D-VOV, a very huge archive is maintained for virtual weight vectors selecting. But this may consume a lot of computing resources. Here we use the mix of archive $\boldsymbol{A}$ and current population $\boldsymbol{P}$ for generating virtual objective vector set $\boldsymbol{vov}$ by Algorithm. 6. The size of the mix is much smaller than the predefined archive size in MOEA/D-VOV.

In Algorithm. 6, the archive $\boldsymbol{A}$ and current population $\boldsymbol{P}$ are combined as $\boldsymbol{mix}$. Then we get the normalized objective vectors set of it ($\boldsymbol{mix'}$) by Equation. 42 (line 2 in Algorithm. 6). A big virtual objective vector pool $\boldsymbol{vov_{pool}}$ is generated by incremental lattice design and the size of it is set as 20000 (it is same as the original paper). For each vector in $\boldsymbol{vov_{pool}}$, the individual in $\boldsymbol{mix'}$ which has the smallest $d_2$ value to it is picked and it's related $d_1$ is computed. Assuming the $i$ vector in $\boldsymbol{vov_{pool}}$ ($\boldsymbol{vov_{pool,i}}$) and $j$th individual in $\boldsymbol{mix'}$ ($\boldsymbol{mix'_j}$) are under processing, $d_1$ and $d_2$ are shown in Figure. 5 (line 5-10 in Algorithm. 6).

If the $d_1$ is lesser than $\theta$ (it is defined as 0.02 here), the vector in $\boldsymbol{vov_{pool}}$ is added to the virtual objective vector set $\boldsymbol{vov}$ after the transformation in Equation. 45. The vector after transformation is named as $\boldsymbol{new}$.

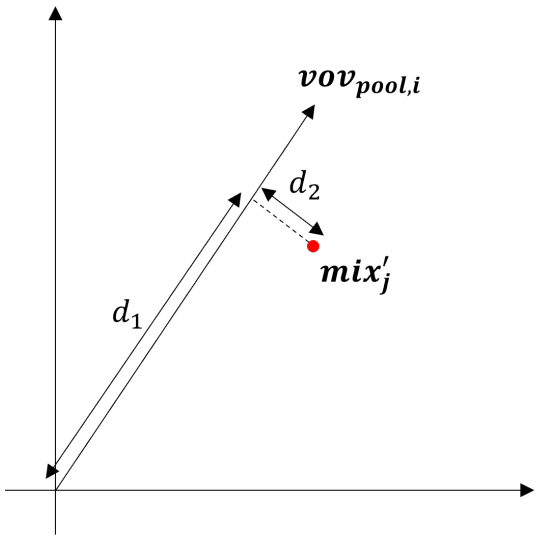



Figure 5.: $d_1$ and $d_2$ in weight vector adjusting process

$$\boldsymbol{new} = \boldsymbol{vov_{pool,i}} \times \frac{d_1}{\|\boldsymbol{vov_{pool,i}}\|} \tag{45}$$

**Algorithm 6** Generate virtual objective vector set $\boldsymbol{VOV}$

1: Combine the objective vectors for archive $\boldsymbol{A}$ and current population $\boldsymbol{P}$ as $\boldsymbol{mix}$
2: Calculate the normalized objective vectors set of $\boldsymbol{mix}$ as $\boldsymbol{mix'}$ by Equation. 42
3: Create a virtual objective vector pool $\boldsymbol{vov_{pool}}$ by incremental lattice design
4: **for** each vector in $\boldsymbol{vov_{pool}}$ **do**
5: Assuming the $i$th vector is chosen from $\boldsymbol{vov_{pool}}$ ($\boldsymbol{vov_{pool,i}}$), pick the $j$th individual in $\boldsymbol{MIX'}$ ($\boldsymbol{mix'_j}$) which has the smallest $d_2$ value on $\boldsymbol{vov_{pool,i}}$
6: Compute the $d_1$ value for $\boldsymbol{mix'_j}$ on $\boldsymbol{vov_{pool,i}}$
7: **if** $d_1 < \theta$ **then**
8: Calculate the vector $\boldsymbol{new}$ by Equation. 45
9: add the $\boldsymbol{new}$ in $\boldsymbol{vov}$
10: **end if**
11: **end for**

After obtaining the virtual objective vector set, it is integrated with $\boldsymbol{mix'}$ as the total weight vector pool $\boldsymbol{total\_pool}$. The vectors at the corner from $\boldsymbol{total\_pool}$ will be moved to the final weight vector set $\boldsymbol{W}$ firstly (line 2 in Algorithm. 7). Here we give an example. Assuming objective size is 3, the 3 vectors which are closest to $[1,0,0]$, $[0,1,0]$ and $[0,0,1]$ are moved to final weight vector set $\boldsymbol{W}$ at first. Then the Euclidean distances between any two vectors in $\boldsymbol{total\_pool}$ are computed for later screening process.

Afterward, the left vector from $\boldsymbol{total\_pool}$ which has the biggest distance to $\boldsymbol{W}$ by Equation. 46 is relocated to $\boldsymbol{W}$ gradually until it reaches the maximum size. This process assures that the $\boldsymbol{W}$ is in good diversity.

$$dis_i = \min_{j=1\sim N} \|\boldsymbol{total\_pool_i} - \boldsymbol{W_j}\| \tag{46}$$

where $\boldsymbol{total\_pool_i}$ is the $i$th vector in $\boldsymbol{total\_pool}$, $\boldsymbol{W_j}$ is the $j$th vector in $\boldsymbol{W}$ and $N$ is the size of $\boldsymbol{W}$. Finally, each vector in $\boldsymbol{W}$ is scaled into $[0, 1]$ by Equation 47 (line 7 in Algorithm. 7).

$$\boldsymbol{W_i} = \boldsymbol{W_i} \cdot (\boldsymbol{f_{max}} - \boldsymbol{f_{min}}) \tag{47}$$

where $\boldsymbol{W_i}$ is the $i$th vector in $\boldsymbol{W}$, $\boldsymbol{f_{max}}$ and $\boldsymbol{f_{min}}$ are $M \times 1$ vectors which contain the biggest and smallest values for each dimension of objective space respectively. Behind the creating of $\boldsymbol{W}$, each vector in it matches a individual in $\boldsymbol{mix}$ by Tchebycheff scalarization function and those individuals act as the population for next generation.

**Algorithm 7** Pick the representative objective vector set and update the weight vectors for next generation

1: Combine the $\boldsymbol{vov}$ and $\boldsymbol{mix'}$ in Algorithm. 6 as total pool $\boldsymbol{total\_pool}$
2: Move the vectors at the corner from $\boldsymbol{total\_pool}$ to the final weight vector set $\boldsymbol{W}$ firstly
3: Compute the Euclidean distances between any two vectors in $\boldsymbol{total\_pool}$
4: **if** $\boldsymbol{W}$ is not full **then**
5: Move the vector from $\boldsymbol{total\_pool}$ which has the biggest distance to $\boldsymbol{W}$ by Equation. 46
6: **end if**
7: Scale the $\boldsymbol{W}$ into $[0, 1]$ by Equation. 47
8: **for** each vector in $\boldsymbol{W}$ **do**
9: Pick the individual in $\boldsymbol{mix}$ which has the smallest scalarization function value on the vector and it will enter the population for next generation
10: **end for**

### *4.6. Archiving strategy*

This part is based on the physical concept of minimizing the total energy in a set of particles. Assume we have a set of equally charged particles in a sphere which can move towards its surface. Although their movements are not constrained here, the particles can only occupy specified positions for the electromagnetic force between them.

Assume there is a full archive with $r$ individuals and the iteration is $k$. Let $\boldsymbol{y_i}$ and $\boldsymbol{y_j}$ be the position of element $i$ and $j$ in objective space, the energy of the archive can be defined as:

$$E = \sum_{i=1}^{r} \sum_{j=i+1}^{r} \frac{1}{(\boldsymbol{y_i} - \boldsymbol{y_j})^T (\boldsymbol{y_i} - \boldsymbol{y_j})} \tag{48}$$

Suppose there are $q$ nominated candidates and they do not dominate any elements in the archive. The problem is choosing $r$ individuals with possible minimum energy $E$ from all the $q + r$ elements. It's impossible to complete the choosing by calculating the energy directly as there are $\binom{r+q}{r}$ possible combination.

If archive $A$ is not full and there is enough space for the candidates, adding those candidates to the archive is enough (line 1-2). Then a symmetric matrix $\boldsymbol{M}$ which contains the reciprocals of the squared distances of all the elements in the archive is updated (line 4). And the total energy of the archive $E$ and a helper vector $\boldsymbol{E_2}$ can be computed from $\boldsymbol{M}$ (line 6-7). $\boldsymbol{E_2}$ is utilized to accelerate the calculation process which will be elaborate later. If the archive is not full, the following procedure will be

applied instead.

Given a archive $\boldsymbol{A}$ and a set of candidate elements $\boldsymbol{C}$, for each individual in $\boldsymbol{A}$, calculate the new $E$ assuming the individual is replaced by an elements in $\boldsymbol{C}$ (line 23). If the lowest variation of $E$ is negative, the related elements in $\boldsymbol{A}$ and $\boldsymbol{C}$ will be exchanged. This process repeats until the maximum specified number of iterations is reached or there is no more possible improvements (lines 16-29). The maximum number of iterations is defined as 100 here (named as $n_{it\_max}$). If the archive is not full and the left space is not enough for all the candidates, the algorithm adds, sequentially, the candidates which are related to the smallest increasing of the total energy of the archive (lines 11-12). In this case, the algorithm only adds candidates instead of exchanging individuals until the archive is full. Meanwhile, the $\boldsymbol{M}$ and $\boldsymbol{E_2}$ are updated.

The pseudo-code for the energy based archiving strategy is given in Algorithm 8.

**Algorithm 8** Energy Based Archiving

```
1:  if left room in the archive is enough for all candidates then
2:    Add all the candidates to the archive
3:    for all candidates do
4:      Update the matrix M which contains the reciprocal of the squared distance
        of each pair of candidates
5:    end for
6:    Update the total energy E of the archive
7:    Update the vector E_2
8:  else
9:    if only some candidates can be added then
10:     while archive is not full do
11:       Choose the candidate which gives the least possible addition of energy to
          the archive and add it
12:       Update M, E, and E_2
13:     end while
14:   else
15:     if the archive is full then
16:       Set improved as TRUE
17:       Set iterations as 0
18:       while improved and iterations < max_it do
19:         Set improved as FALSE
20:         iterations = iterations + 1
21:         Create a matrix containing the energy that the archive would have if each
            element of the archive were substituted with each candidate
22:         Locate the minimum entry E_new of this matrix
23:         if E_new < E then
24:           E_new is at position (i*, j*)
25:           Swap candidate j* with element i*
26:           Set improved as TRUE
27:           Update M, E, and E_2
28:         end if
29:       end while
30:     end if
31:   end if
32: end if
```

### *4.7. Computational complexity of MACSAW*

In this section we try to calculate the computational complexity of the new algorithm. The MACSAW is scattered into the following segments:

Table 3.: The parts and related computational complexity of MACSAW

| | Individualistic actions | Social actions | Weight adjust | Update population by archive | Update archive |
|---|---|---|---|---|---|
| position in Algorithm. 1 | line 8,10,11 | line 13 | line 15,16,17 | line 18-32 | line 9,12,14 |
| Computational complexity | $O(NMD)$ | $O(N(D+M+max_{length}))$ | $O(MSize_{vov\_pool}{}^2)$ | $O(MN^2)$ | $O(n_{it\_max}N^2+MN^2)$ |

#### *4.7.1. Individualistic actions*

As MACSAW divides the individualistic action into three parts (line 8,11 in Algorithm. 1 and Algorithm. 2).

For line 8,11 in Algorithm. 1, we only consider line 11 as it evolves more individuals. And the worst case for this part is that all the three predefined parts in Algorithm. 3 are executed. For those three parts, the biggest computing costs may be spent on pattern search (line 10-22 Algorithm. 3) as it may explore many random directions. And the biggest random direction amount which will be explored is the dimension of decision space (named $D$ here). Further, the new offspring need to be compared with the old one on both dominance relationship and scalarization function value. The objective space's dimension is named $M$ here. This process needs to traverse every dimension of objective space (line 17 in Algorithm 1). Overall, the total cost for line 11 is $O(NMD)$ ($N$ is the number of sub-problems, the population size and the individual number evolved in one round of individualistic actions).

For Algorithm. 2, line 1 expends most resource as it traverses all the objectives and individuals. It needs $O(NM)$.

The computational complexity for individualistic action is $O(NMD + NM) \approx O(NMD)$.

#### *4.7.2. Social actions*

For social action, line 2 and 3 in Algorithm. 4 need $O(max_{length})$ as they traverse the memory. Line 4 needs $O(D)$ as it considers all the dimension of decision space. Line 5 needs $O(M)$ for comparison. As social actions evolve $N$ individuals at one round, they consume $O(N(D+M+max_{length}))$.

#### *4.7.3. Weight adjust*

In weight vector adjusting, the state deciding needs $O(MN^2)$ in dominated-sorting [31] (line 2-4 in Algorithm. 5) and $O(MN^2)$ in diversity measuring (line 8-11 in Algorithm. 5).

In Algorithm. 6, line 4-11 is the most expensive part. Line 5 require $O(2NM)$ as distance calculating needs to traverse all the objectives and it's total cost is $O(2NMSize_{vov\_pool})$ for line 4-11. $Size_{vov\_pool}$ is the size of $\boldsymbol{vov_{pool}}$.

For Algorithm. 7, the biggest time consuming happens in line 3. It costs $O(MSize_{vov\_pool}{}^2)$.

Finally, the complexity for weight adjust is $O(M(N^2+Size_{vov\_pool}{}^2+NSize_{vov\_pool}))$. $Size_{vov\_pool}$ is a much larger number to $N$. So, the final result is $O(MSize_{vov\_pool}{}^2)$.

#### *4.7.4. Update population by archive*

In line 18-32 in Algorithm 1, the worst situation is traversing all the solutions in archive and population on all the objectives for matching the sub-problems which needs $O(MN^2)$.

#### *4.7.5. Update archive*

Cost for Algorithm 8 has been calculated in MACS2.1 [4]. Its value is $3Mrq + 2r^2 n_{it}$. $M$ is dimension of the objective space. $n_{it}$ is the times for energy minimization step. And the aim of this algorithm is pick $r$ individuals from $r + q$.

The Algorithm 8 is applied to screen solutions after individual and social action. And the mix of offspring and current archive is executed non-dominated sort before archiving (line 9,12,14 in Algorithm 1). Let's consider the sorting at first. Individualistic action may create more possible solutions than social action as it has three parts that can yield individuals. Therefore, worst case is that all the parts produce $3N$ solutions for the whole population in total and archive size is $max_{arch}$ which is always set as $N$ (if archive is full, the pattern will execute only once). The non-dominated sorting needs $M(3N + N)^2 \approx O(MN^2)$.

Further, let's calculate the cost for archiving process. As individualistic action may create more solutions than social action, line 11 in Algorithm 1 is picked for computing. Assuming the worst case, the archiving approach reaches biggest energy minimization step times $n_{it\_max}$ and $N$ solutions (same as $max_{arch}$) is already in the archive. The individualistic action create maximum possible non-dominated $3N$ solutions. The archiving strategy needs to select $N$ from total $3N+N$ individuals. Therefore, $r$ is equal to $N$ and $q$ is equal to $3N$. The cost is $9MN^2 + 2n_{it\_max}(N)^2 \approx O(n_{it\_max}N^2 + MN^2)$.

As $max_{length}$ is $0.5N$ here. $M$ and $D$ is always smaller than $N$ in the research field. Finally, the total computational complexity for MACSAW is $O(NMD) + O(N(D + M + max_{length})) + O(MSize_{vov\_pool}{}^2) + O(MN^2) + O(n_{it\_max}N^2 + MN^2) \approx O(MN^2 + n_{it\_max}N^2 + MSize_{vov\_pool}{}^2)$.

## 5. Experiment

The experiments have been carried out and their results are displayed in this part. We executed the comparative tests on MaOEA-DDFC [27], MOEA/D-DU [32], MOEA/D-PaS [33], MSEA [30], PeEA [29], RVEAa [34], RVEAiGNG [35], VaEA [36], MACS2.1 [4] and MACSAW. The details of other algorithms are as follows:

**MaOEA-DDFC** is a many-objective evolutionary algorithm which contains a favorable convergence indicator and a directional diversity indicator which are introduced to respectively measure the convergence and the diversity performances of an individual. Further, a new mechanism is employed to generate offspring with good convergence and a new mechanism in environmental selection is used to balance diversity and convergence.

**MOEA/D-DU** applies a new method for a better tradeoff between convergence and diversity in decomposition-based many-objective optimizers.

**MOEA/D-PaS** is a creative algorithm which can adaptively choose appropriate scalarizing function for a better tradeoff between search ability and its robustness on PF geometries.

**MSEA** is a multistage algorithm which can improve the diversity performance. It divides the optimization process into multiple stages according to the current popu-

lation's characteristic. And it updates the population by various selection criteria in different stages.

**PeEA** proposes an on-line PF shape estimation strategy to get the curvature of PF approximately. And it presents an adaptive scalarizing function and a new similarity measure parameter named dimensionality margin distance for diversity.

**RVEAa** is a inventive algorithm which does not require reference points or weight vectors and the searching in it is guided by current population.

**RVEAiGNG** applies improved growing neural gas to solve MOEAs with irregular PFs. The new method can balance exploration and exploitation appropriately.

**VaEA** adaptively adjusts the weight vectors based on the distribution of current population. Moreover, this approach is parameter-less and with lower time complexity.

### *5.1. Experiment Setting*

The standard benchmark sets UF and ZDT are tested here. And two real space mission design problems are handled. They are all multi-objective problems.

The UF is a intractable set which has complex Pareto set. ZDT suite is also a commonly used benchmark. The characteristics of UF can be found in [37] and the description for ZDT is in [38]. There are fifteen standard benchmarks and two real world problems. The numbers of fitness evaluations and corresponding variables numbers are listed in Table 4. And parameters of all the algorithms are in Table 5.

Table 4.: Objective numbers, variables numbers and maximum fitness evaluation for each problem.

| Problems | Population Size ($N$) | Objective No. ($M$) | Variables No. | Fitness Evaluation |
|---|---|---|---|---|
| UF1-7 | 100 | 2 | 30 | 300000 |
| UF8-10 | 91 | 3 | 30 | 273000 |
| ZDT1-3 | 100 | 2 | 30 | 300000 |
| ZDT4,6 | 100 | 2 | 10 | 300000 |
| Cassini | 150 | 2 | 6 | 600000 |
| 3imp | 150 | 2 | 5 | 600000 |

Table 5.: Parameters of all the algorithms.

| Algorithms | Parameters |
|---|---|
| MaOEA-DDFC | $K = 5$ $L = 3$ |
| MOEA/D-DU | $\delta = 0.9$ $K = 3$ |
| RVEAa | $\alpha = 2$ $fr = 0.1$ |
| RVEAiGNG | $\alpha = 2$ |
| MACS2.1/MACAW | $F = 1$ $CR = 1$ |
| | $\rho_{ini} = 1$ $\rho_{contr} = 0.5$ |
| | $\rho_{max,contr} = 5$ |

Meanwhile the two real space mission design problems are 3imp and Cassini [2]. The target of 3imp is to transfer a spacecraft from a circular Low Earth Orbit, with a radius of 7000km, to a circular Geostationary orbit, with a radius of 42000km by three impulsive manoeuvres. The spacecraft departs at time $t_0$ from the circular Low Earth Orbit and injects into the circular Geostationary orbit after a transfer time $T = t_1 + t_2$. The intermediate manoeuvre happens at $t_0 + t_1$ at a direction which has radius $r_1$ and angle $\theta_1$ in polar coordinates. The objective function for this problem

is the total transfer time $T$ and the sum of the three impulses $\Delta v_{tot}$. The decision vector is defined in a space $[t_0, t_1, r_1, \theta_1, t_2]^T \in D \subset \mathbb{R}^5$. And the decision variables have the following intervals: $t_0 \in [0 \quad 1.62]$, $t_1 \in [0.03 \quad 21.54]$, $r_1 \in [7010 \quad 105410]$, $\theta_1 \in [0.01 \quad 2\pi - 0.01]$ and $t_2 \in [0.03 \quad 21.54]$.

The target of Cassini is launching a spacecraft from earth to land Saturn through a sequence of swing-by's with the planets: Venus, Venus, Earth, and Jupiter. The total transfer arc consists with five small arcs. For each small arc, the departure time from planet $P_i$ and the arrival time at planet $P_{i+1}$ are given, then the required incoming and outgoing velocities at each swing-by planet $v_{in}$ and $v_{rout}$ can be calculated as the solution of a Lambert's problem. Every swing-by is modeled through a linked-conic approximation with powered manoeuvres. The mismatch between the required outgoing velocity $v_{rout}$ and the achievable outgoing velocity $v_{aout}$ is compensated through a $\Delta v$ manoeuvre at the pericentre of the gravity assist hyperbola. The departure time is defined as $t_0$ and the transfer time for each leg is $T_i$, $i = 1, .., 5$. Once each powered swing-by manoeuvre is computed, the normalized radius of the pericentre $r_{p,i}$ of each swing-by hyperbola can be got. So, in the searching process of the optimal solution a constraint on each pericentre radius should be considered and it is introduced into the objective by weight vector as following:

$$f(\boldsymbol{x}) = \Delta v_0 + \sum_{i=1}^{4} \Delta v_i + \Delta v_f + \sum_{i=1}^{4} \omega_i (r_{p,i} - r_{pmin,i})^2 \tag{49}$$

The decision vector is defined as $[t_0, T_1, T_2, T_3, T_4, T_5]^T$ and the objectives are $f(\boldsymbol{x})$ and $T = \sum_{i=1}^{5} T_i$. The minimum normalized pericentre radii are $r_{pmin,1} = 1.0496$, $r_{pmin,2} = 1.0496$, $r_{pmin,3} = 1.0627$ and $r_{pmin,4} = 9.3925$. The decision variables have the following intervals: $t_0 \in [-1000 \quad 0]MJD2000$, $T_1 \in [30 \quad 400]d$, $T_2 \in [100 \quad 470]d$, $T_3 \in [30 \quad 400]d$, $T_4 \in [400 \quad 2000]d$ and $T_5 \in [1000 \quad 6000]d$.

### *5.2. Results*

For the standard benchmark sets, HV and IGD are utilized as measurement parameters and the results are shown in Table 6-7.The HV/IGD values of each solution set are calculated with a set of reference points which are generated in a famous open source platform [39]. Every instance was run for 30 times independently, and in the following bracket we listed the standard deviation of HV and IGD. The Wilcoxon rank sum test with a significance level of 0.05 is adopted to do the analysis. The marks "+", "−" and "≈" mean that the result of another algorithm is significantly better, worse and similar to the result of MASCAWS. The best result for each test case is marked by '*'. MASCAWS demonstrates strong competitiveness in Table 6 and 7.

In general, two mixed (diversity and convergence) parameters HV and IGD are picked as measurement parameters. But the two real world problems have unknown true PFs. Therefore, the pure diversity PD parameter [40] which doesn't need reference are picked instead. We use it to quantify diversity. And HV is utilized to estimate convergence. The results of the 30 runs for MACS2.1 and MACSAW are gathered and the no-dominated solutions in them are applied as the reference for HV calculating. It is worth noting that the bigger PD is, the better diversity.

Those measurement parameter results for the real problems are shown in Table 8. Further, the run with middle HV is picked to plot the final results as Figure. 6-11.

Table 6.: HV values obtained by MaOEA-DDFC, MOEA/D-DU, MOEA/D-PaS, MSEA, PeEA, RVEAa, RVEAiGNG, VaEA, MACS2.1 and MACSAW on UF and ZDT

| Problem | MaOEA-DDFC | MOEA/D-DU | MOEA/D-PaS | MSEA | PeEA | RVEAa | RVEAiGNG | VaEA | MACS2.1 | MACSAW |
|---|---|---|---|---|---|---|---|---|---|---|
| UF1 | 4.0025e-1 (1.03e-1) - | 6.3866e-1 (2.06e-2) - | 7.1491e-1 (1.51e-3) +* | 6.1464e-1 (2.10e-2) - | 6.0004e-1 (2.60e-2) - | 5.9361e-1 (2.35e-2) - | 5.8129e-1 (4.22e-2) - | 5.9445e-1 (2.57e-2) - | 6.9693e-1 (1.61e-2) = | 7.0281e-1 (5.80e-3) |
| UF2 | 6.7618e-1 (5.72e-3) - | 6.9474e-1 (6.11e-3) - | 7.0803e-1 (2.28e-3) - | 6.8545e-1 (8.55e-3) - | 6.8154e-1 (4.99e-3) - | 6.7938e-1 (9.87e-3) - | 6.7856e-1 (1.05e-2) - | 6.8296e-1 (9.71e-3) - | 7.1245e-1 (1.74e-3) - | 7.1368e-1 (1.64e-3)* |
| UF3 | 4.9075e-1 (3.29e-2) - | 5.7530e-1 (2.54e-2) - | 6.8043e-1 (1.25e-1) +* | 4.3273e-1 (4.09e-2) - | 4.5165e-1 (3.21e-2) - | 4.1694e-1 (5.47e-2) - | 3.7839e-1 (3.55e-2) - | 4.3967e-1 (4.14e-2) - | 5.2611e-1 (4.74e-2) - | 6.4036e-1 (1.52e-2) |
| UF4 | 3.7648e-1 (6.35e-3) - | 3.9074e-1 (1.30e-3) + | 3.4938e-1 (7.21e-3) - | 3.8980e-1 (2.62e-3) + | 3.8353e-1 (9.79e-4) - | 3.8952e-1 (1.96e-3) + | 3.9140e-1 (9.37e-4) + | 3.8727e-1 (1.25e-3) + | 3.9956e-1 (1.01e-3) +* | 3.8399e-1 (3.13e-3) |
| UF5 | 2.5184e-1 (4.89e-2) - | 2.5153e-1 (6.08e-2) - | 1.8984e-1 (6.85e-2) - | 2.3504e-1 (7.75e-2) - | 2.5988e-1 (6.33e-2) - | 2.5234e-1 (6.13e-2) - | 2.4087e-1 (7.32e-2) - | 2.4814e-1 (4.85e-2) - | 3.6053e-1 (3.51e-2) - | 4.7083e-1 (2.00e-2)* |
| UF6 | 3.0555e-1 (6.46e-2) - | 3.2822e-1 (6.84e-2) - | 3.1019e-1 (8.01e-2) - | 3.0240e-1 (6.81e-2) - | 2.9271e-1 (7.33e-2) - | 2.7942e-1 (6.70e-2) - | 2.7244e-1 (7.93e-2) - | 3.2646e-1 (5.25e-2) - | 3.8826e-1 (2.58e-2) - | 4.4507e-1 (1.58e-2)* |
| UF7 | 3.8778e-1 (1.59e-1) - | 4.9843e-1 (9.40e-2) - | 5.7497e-1 (5.17e-3) +* | 4.3250e-1 (1.10e-1) - | 5.2882e-1 (6.12e-2) - | 4.0691e-1 (1.26e-1) - | 3.8401e-1 (1.23e-1) - | 4.8456e-1 (1.02e-1) - | 5.6481e-1 (3.17e-2) = | 5.7105e-1 (5.26e-3) |
| UF8 | 2.5631e-1 (5.52e-2) - | 4.0699e-1 (4.43e-2) =* | 3.9895e-1 (5.15e-2) = | 3.3978e-1 (3.33e-2) - | 2.5593e-1 (1.04e-1) - | 3.5135e-1 (3.32e-2) - | 3.5107e-1 (4.82e-2) - | 3.4131e-1 (3.57e-2) - | 4.0428e-1 (4.71e-4) = | 4.0439e-1 (5.03e-2) |
| UF9 | 4.7587e-1 (1.79e-1) - | 6.0619e-1 (8.10e-2) - | 5.8699e-1 (3.12e-2) - | 5.5903e-1 (7.11e-2) - | 5.3193e-1 (7.62e-2) - | 5.2995e-1 (4.29e-2) - | 6.3707e-1 (5.69e-2) - | 5.9338e-1 (4.77e-2) - | 7.3887e-1 (4.32e-4) +* | 6.7839e-1 (5.33e-2) |
| UF10 | 1.1979e-1 (3.83e-2) - | 1.6290e-1 (7.50e-2) - | 7.1090e-2 (2.51e-2) - | 1.2044e-1 (4.62e-2) - | 1.5552e-1 (8.17e-2) - | 1.3417e-1 (5.10e-2) - | 1.6092e-1 (8.08e-2) - | 1.8973e-1 (6.28e-2) - | 1.7085e-1 (2.35e-2) - | 2.8688e-1 (4.67e-2)* |
| ZDT1 | 7.1967e-1 (1.87e-4) - | 7.1852e-1 (5.12e-4) - | 7.1979e-1 (1.88e-4) - | 7.1970e-1 (4.57e-3) - | 7.1804e-1 (6.95e-4) - | 7.1982e-1 (2.22e-4) - | 7.2029e-1 (5.81e-5) - | 7.1871e-1 (4.56e-4) - | 7.2055e-1 (4.73e-5) - | 7.2059e-1 (3.50e-5)* |
| ZDT2 | 4.4410e-1 (1.60e-4) - | 4.4392e-1 (4.51e-4) - | 2.2230e-1 (2.26e-1) - | 2.6712e-1 (2.22e-1) = | 4.4379e-1 (2.65e-4) - | 4.4502e-1 (3.71e-5) +* | 4.4488e-1 (6.06e-5) + | 4.4469e-1 (9.73e-5) - | 4.4484e-1 (5.74e-5) = | 4.4482e-1 (5.13e-5) |
| ZDT3 | 5.9935e-1 (2.19e-4) - | 6.0123e-1 (6.48e-2) + | 5.9864e-1 (2.82e-4) - | 6.0589e-1 (2.25e-2) =* | 5.9616e-1 (4.74e-4) - | 5.9775e-1 (1.76e-3) - | 5.9971e-1 (4.32e-5) - | 5.9956e-1 (3.93e-4) - | 5.9991e-1 (1.12e-5) - | 5.9992e-1 (1.18e-5) |
| ZDT4 | 7.2014e-1 (8.30e-5) + | 2.3905e-1 (3.44e-1) - | 4.7971e-2 (1.83e-1) - | 4.5586e-1 (3.53e-1) = | 7.1855e-1 (5.30e-4) - | 7.1973e-1 (2.60e-4) + | 7.2038e-1 (4.29e-5) +* | 7.1369e-1 (5.48e-3) - | 7.1513e-1 (2.26e-3) - | 7.1867e-1 (2.60e-3) |
| ZDT6 | 3.8795e-1 (8.13e-4) - | 3.8747e-1 (9.14e-4) - | 3.8880e-1 (3.77e-5) - | 3.8897e-1 (7.18e-5) - | 3.8763e-1 (2.05e-4) - | 3.8889e-1 (1.36e-5) - | 3.8872e-1 (2.42e-5) - | 3.8866e-1 (7.31e-5) - | 3.9865e-1 (7.08e-6) +* | 3.9862e-1 (1.67e-5) |
| +/ − / ≈ | 1/14/0 | 2/12/1 | 3/11/1 | 1/11/3 | 0/15/0 | 3/12/0 | 3/12/0 | 1/14/0 | 3/8/4 | |

Table 7.: IGD values obtained by MaOEA-DDFC, MOEA/D-DU, MOEA/D-PaS, MSEA, PeEA, RVEAa, RVEAiGNG, VaEA, MACS2.1 and MACSAW on UF and ZDT

| Problem | MaOEA-DDFC | MOEA/D-DU | MOEA/D-PaS | MSEA | PeEA | RVEAa | RVEAiGNG | VaEA | MACS2.1 | MACSAW |
|---|---|---|---|---|---|---|---|---|---|---|
| UF1 | 2.5612e-1 (7.51e-2) - | 5.3339e-2 (1.06e-2) - | 6.2387e-3 (1.15e-3) +* | 8.9112e-2 (1.74e-2) - | 8.2346e-2 (2.32e-2) - | 1.0609e-1 (2.08e-2) - | 1.1813e-1 (4.10e-2) - | 1.0306e-1 (1.84e-2) - | 2.1335e-2 (1.31e-2) = | 1.6568e-2 (3.86e-3) |
| UF2 | 4.2118e-2 (8.74e-3) - | 2.2085e-2 (4.63e-3) - | 1.2828e-2 (1.78e-3) - | 3.7457e-2 (1.25e-2) - | 3.4647e-2 (6.50e-3) - | 3.9333e-2 (1.30e-2) - | 4.8686e-2 (1.97e-2) - | 3.7878e-2 (1.79e-2) - | 1.1370e-2 (1.80e-3) - | 9.7820e-3 (1.68e-3)* |
| UF3 | 1.8658e-1 (2.44e-2) - | 9.0282e-2 (1.16e-2) - | 3.9872e-2 (1.37e-1) +* | 2.6788e-1 (5.08e-2) - | 2.1287e-1 (3.94e-2) - | 2.6483e-1 (4.70e-2) - | 3.1974e-1 (2.22e-2) - | 2.6514e-1 (4.65e-2) - | 1.5774e-1 (3.10e-2) - | 6.5732e-2 (2.10e-2) |
| UF4 | 5.2509e-2 (4.72e-3) + | 4.0410e-2 (1.20e-3) +* | 6.8164e-2 (5.02e-3) + | 4.2360e-2 (2.39e-3) + | 4.5143e-2 (5.53e-4) + | 4.2317e-2 (1.76e-3) + | 4.0748e-2 (9.03e-4) + | 4.3133e-2 (8.18e-4) + | 2.5859e-1 (8.96e-4) + | 2.6910e-1 (1.71e-3) |
| UF5 | 2.4707e-1 (3.52e-2) - | 2.5724e-1 (5.88e-2) - | 2.9298e-1 (1.17e-1) - | 3.1813e-1 (1.28e-1) - | 2.5985e-1 (6.48e-2) - | 3.0559e-1 (9.31e-2) - | 3.4155e-1 (1.35e-1) - | 2.8035e-1 (8.20e-2) - | 1.6666e-1 (2.94e-2) - | 7.7012e-2 (1.40e-2)* |
| UF6 | 1.9604e-1 (9.49e-2) - | 1.4421e-1 (8.00e-2) - | 1.7590e-1 (1.08e-1) - | 2.3195e-1 (1.20e-1) - | 2.1200e-1 (1.20e-1) - | 2.7375e-1 (1.31e-1) - | 2.8902e-1 (1.77e-1) - | 1.8563e-1 (1.02e-1) - | 9.5888e-2 (2.09e-2) - | 6.0225e-2 (1.02e-2)* |
| UF7 | 2.3008e-1 (2.34e-1) = | 7.4842e-2 (9.78e-2) + | 7.4370e-3 (2.78e-3) +* | 1.7523e-1 (1.59e-1) = | 4.6155e-2 (6.33e-2) + | 2.1294e-1 (1.79e-1) = | 2.4798e-1 (1.76e-1) - | 1.1400e-1 (1.40e-1) + | 1.2572e-1 (1.76e-2) = | 1.2289e-1 (1.81e-3) |
| UF8 | 3.5738e-1 (7.27e-2) - | 1.4950e-1 (6.61e-2) +* | 2.4306e-1 (1.07e-1) = | 2.6696e-1 (5.27e-2) = | 3.3341e-1 (1.42e-1) - | 2.4247e-1 (4.66e-2) = | 2.3581e-1 (5.99e-2) = | 2.5004e-1 (6.77e-2) = | 3.4930e-1 (3.59e-3) - | 2.3307e-1 (1.12e-1) |
| UF9 | 3.0994e-1 (1.90e-1) - | 1.5622e-1 (7.90e-2) = | 3.0733e-1 (2.16e-2) - | 2.4470e-1 (7.66e-2) - | 2.3467e-1 (8.36e-2) - | 2.5608e-1 (4.86e-2) - | 2.1240e-1 (8.42e-2) = | 1.8703e-1 (6.17e-2) = | 9.9205e-2 (1.33e-3) =* | 1.6145e-1 (9.02e-2) |
| UF10 | 5.2247e-1 (1.08e-1) - | 4.1503e-1 (1.19e-1) - | 5.7877e-1 (5.41e-2) - | 4.9295e-1 (1.17e-1) - | 4.4810e-1 (1.31e-1) - | 4.8522e-1 (1.44e-1) - | 5.2246e-1 (1.94e-1) - | 3.7898e-1 (8.47e-2) - | 3.7580e-1 (4.83e-2) - | 2.7395e-1 (5.27e-2)* |
| ZDT1 | 4.2877e-3 (1.13e-4) - | 4.3794e-3 (2.35e-4) - | 4.3378e-3 (1.15e-4) - | 5.0116e-3 (6.57e-3) - | 5.8811e-3 (5.53e-4) - | 4.1951e-3 (1.20e-4) - | 4.0732e-3 (5.02e-5) - | 4.8845e-3 (2.55e-4) - | 3.6065e-3 (1.63e-5) = | 3.6034e-3 (1.24e-5)* |
| ZDT2 | 4.4366e-3 (9.03e-5) + | 3.8164e-3 (2.23e-5) +* | 1.4260e+0 (1.49e+0) = | 6.7034e-1 (8.56e-1) = | 5.0875e-3 (2.30e-4) + | 3.8979e-3 (3.11e-5) + | 4.1615e-3 (4.94e-5) + | 4.2407e-3 (1.15e-4) + | 2.3013e-1 (1.51e-5) - | 2.3012e-1 (1.16e-5) |
| ZDT3 | 4.7544e-3 (7.47e-5) - | 5.7081e-2 (7.23e-3) - | 8.4754e-3 (9.72e-4) - | 6.2775e-3 (7.51e-3) - | 1.2435e-2 (9.64e-4) - | 6.7194e-3 (6.87e-4) - | 4.9365e-3 (1.45e-4) - | 5.7919e-3 (3.94e-4) - | 4.3500e-3 (1.86e-5) - | 4.3339e-3 (2.88e-5)* |
| ZDT4 | 4.0757e-3 (5.78e-5) + | 2.8770e+1 (2.35e+1) - | 3.0309e+1 (1.83e+1) - | 3.3841e+0 (5.49e+0) = | 5.5423e-3 (4.10e-4) - | 4.2434e-3 (1.44e-4) + | 4.0238e-3 (6.19e-5) +* | 1.0682e-2 (6.51e-3) - | 6.0516e-3 (1.49e-3) - | 4.4955e-3 (2.01e-3) |
| ZDT6 | 3.8348e-3 (6.87e-4) + | 3.9613e-3 (6.79e-5) + | 3.1565e-3 (4.03e-5) + | 2.9862e-3 (1.15e-5) +* | 4.4175e-3 (2.49e-4) + | 3.0860e-3 (1.98e-5) + | 3.2881e-3 (2.78e-5) + | 3.3126e-3 (6.63e-5) + | 2.4583e-1 (2.64e-6) = | 2.4583e-1 (4.05e-6) |
| +/ − / ≈ | 4/10/1 | 5/9/1 | 5/8/2 | 2/9/4 | 4/11/0 | 4/9/2 | 4/9/2 | 4/9/2 | 1/9/5 | |

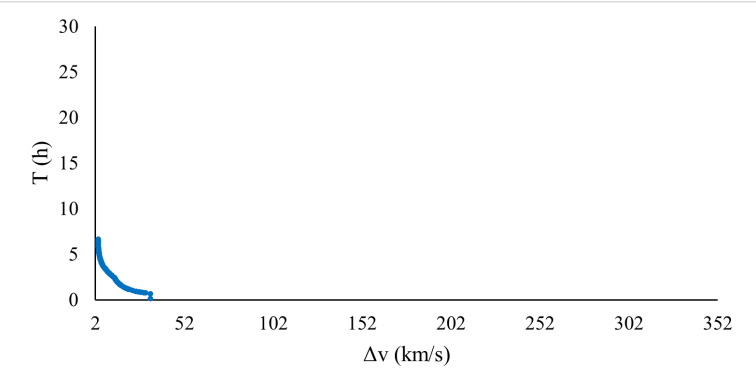



Figure 6.: Final results on 3imp for the test with middle HV of MACS2.1

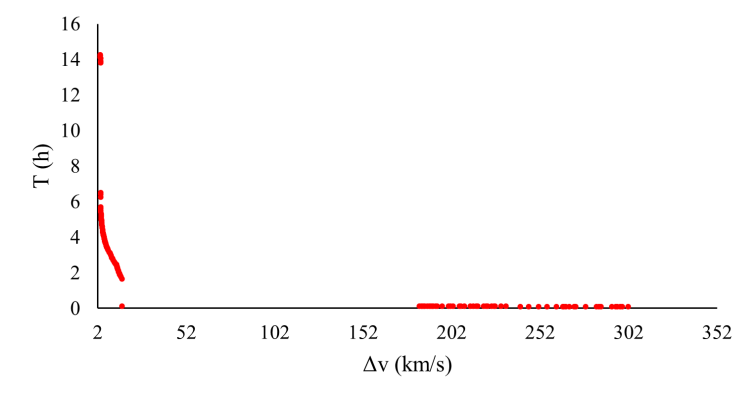



Figure 7.: Final results on 3imp for the test with middle HV of MACSAW

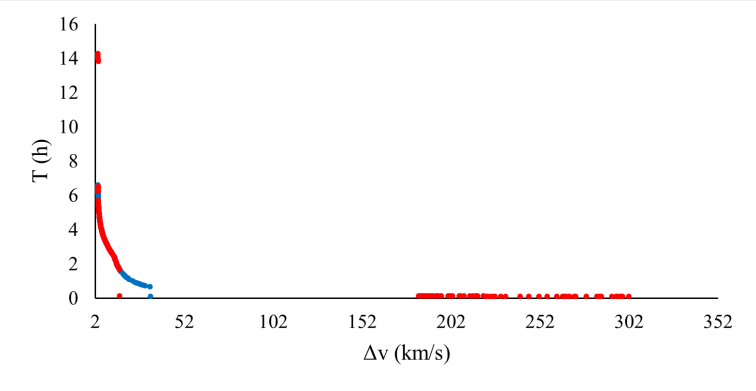



Figure 8.: Final results on 3imp for the test with middle HV of MACS2.1 and MACSAW (MACSAW in red and MACS2.1 in blue)

The 3imp problem's results are in Figure. 6-8. For convergence, it seems that the two algorithms put the PF into same depth in Figure. 8, and MACSAW obtains tiny advantage in Table. 8. For diversity, in Figure. 8, MACSAW gets the solutions on whole PF while MACS2.1 only can obtain results in a small area, but it also lose some

solutions in the center of PF. The PD value for MACSAW in Table 8 shows better diversity. The best result for each test case is marked by '*'.

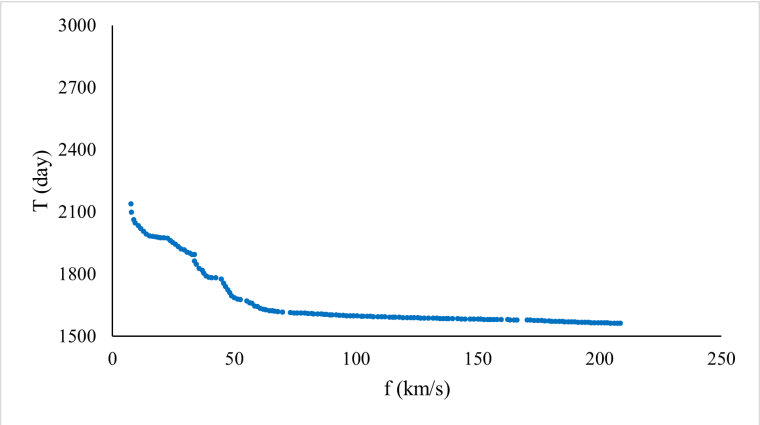



Figure 9.: Final results on Cassini for the test with middle HV of MACS2.1

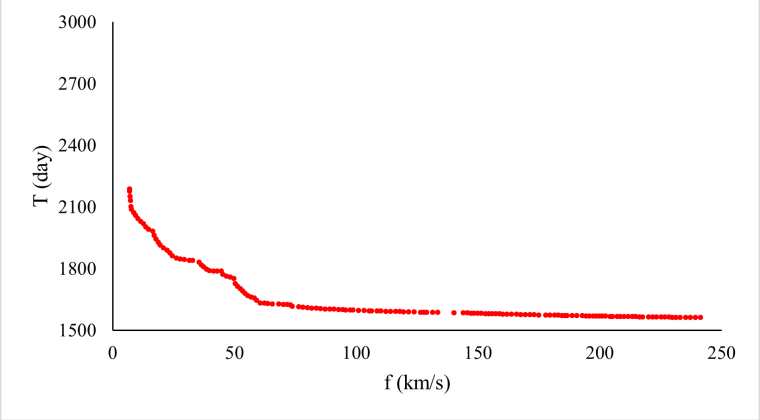



Figure 10.: Final results on Cassini for the test with middle HV of MACSAW

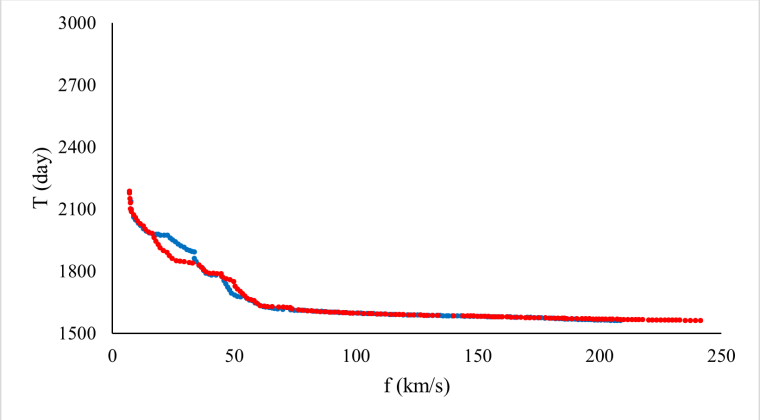



Figure 11.: Final results on Cassini for the test with middle HV of MACS2.1 and MACSAW (MACSAW in red and MACS2.1 in blue)

For Cassini, in Figure. 11, MACSAW nearly puts all the part of PF deeper than MACS2.1, and it gets more solutions in the rightmost area. In Table 8, they are nearly same in HV and PD.

Table 8.: The results on HV and IGD for the two real world problems.

| problem | MACS2.1 | MACSAW |
|---|---|---|
| 3imp/HV | 6.9214e-1 (2.89e-3) = | 6.9232e-1 (2.92e-3)* |
| Cassini/HV | 9.8347e-1 (2.26e-5) =* | 9.8346e-1 (3.52e-5) |
| 3imp/PD | 2.9146e4 (1.36e4) = | 3.0595e4 (1.71e4)* |
| Cassini/PD | 3.0595e4 (1.71e4) = | 3.0623e4 (1.62e4)* |
| $+/-/\approx$ | 0/0/4 | |

## 6. Discussion

In 3imp problems, we can get more solutions in the extreme regions on both sides. As 3imp has uneven PF, the weight vector process may be effective here. But MACSAW still can't obtain even solutions on the whole PF. The reason needs to be explored later.

In Cassini, we can't put the PF deeper in all parts and the measurement parameters are almost unchanged. The new utility function may need more improvements.

We introduced weight vector adjusting process, but it still has great computational complexity after we use a smaller archive. We need to make some improvements to reduce the computing resources required.

As DE pool has massive optional strategies. More possible operator combinations need to be explored.

## 7. Conclusions

In this paper, we presents a new Multi Agent Collaborative Search Algorithm with Adaptive Weights (MACSAW). The improvement can be divided into three parts from its previous version MACS2.1. First, a new kind of utility function is utilized. Secondly, a new social action process which contains more operators and adaptive parameters is used. Finally, a new weight vectors adjustment process with population state trigger is applied. The new algorithm shows competitiveness on some standard benchmarks and two real optimization problems.

There are still some flaws need to be tackled. The weight vector adjusting process may help us get more solutions in the extreme regions in 3imp, but they are still not even enough. And it has huge computational complexity. More works are needed for its improvement. In Cassini, MACSAW can't put all the part of PF deeper than MACS2.1. Maybe a better utility function is useful here. In MACSAW we directly use the IMODE's operators, but the potential DE strategy pool is huge as many researchers constantly propose new operators. More operator combinations need to be explored. Our future work will focus on these issues.

## Disclosure statement

No potential conflict of interest was reported by the authors.


## Funding

This work is supported by Natural Science Foundation of China under Grant No.42271391&No. 62006214, Joint Funds of Equipment Pre-Research and Ministry of Education of China Grant No. 8091B022148, the 13th Five-year Pre-research Project of Civil Aerospace in China, Hubei excellent young and middle-aged science and technology innovation team plan project under Grant No. T2021031, Hubei Natural Science Foundation of China under Grant No. 2019CFA023, and the Opening Fund of Key Laboratory of Geological Survey and Evaluation of Ministry of Education under Grant No. KLIGIP-2019B07.


## References


[1] Vasile M. Robust mission design through evidence theory and multiagent collaborative search. Wiley. 2005;(1).

[2] Vasile M, Zuiani F. Multi-agent collaborative search: an agent-based memetic multi-objective optimization algorithm applied to space trajectory design. Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering. 2011; 225(11):1211–1227. Available from: https://doi.org/10.1177/0954410011410274.

[3] Zuiani F, Vasile M. Multi agent collaborative search based on tchebycheff decomposition. Computational Optimization and Applications. 2013;56:189–208.

[4] Ricciardi LA, Vasile M. Improved archiving and search strategies for multi agent collaborative search. Cham: Springer International Publishing; 2019. p. 435–455. Available from: https://doi.org/10.1007/978-3-319-89988-$6_2$6.

[5] Qi Y, Ma X, Liu F, et al. Moea/d with adaptive weight adjustment. Evolutionary computation. 2014;22(2):231–264.

[6] Storn R, Price KV. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization. 1997;11:341–359.

[7] Price K, Storn RM, Lampinen JA. Differential evolution: a practical approach to global optimization. Springer Science & Business Media; 2006.

[8] Qin AK, Huang VL, Suganthan PN. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation. 2009;13(2):398–417.

[9] Iorio AW, Li X. Solving rotated multi-objective optimization problems using differential evolution. In: AI 2004: Advances in Artificial Intelligence: 17th Australian Joint Conference on Artificial Intelligence, Cairns, Australia, December 4-6, 2004. Proceedings 17; Springer; 2005. p. 861–872.

[10] Attia MM, Arafa M, Sallam EA, et al. An enhanced differential evolution algorithm with multi-mutation strategies and self-adapting control parameters. International Journal of Intelligent Systems and Applications. 2019;.

[11] Mohamed AW. A novel differential evolution algorithm for solving constrained engineering optimization problems. Journal of Intelligent Manufacturing. 2018;29:659–692.

[12] Zhang SX, Zheng LM, Tang KS, et al. Multi-layer competitive-cooperative framework for performance enhancement of differential evolution. Information Sciences. 2019;482:86–104.

[13] Sallam KM, Elsayed SM, Chakrabortty RK, et al. Improved multi-operator differential

evolution algorithm for solving unconstrained problems. In: 2020 IEEE Congress on Evolutionary Computation (CEC); 2020. p. 1–8.
[14] Sallam KM, Elsayed SM, Sarker RA, et al. Landscape-based adaptive operator selection mechanism for differential evolution. Information Sciences. 2017;418-419:383–404. Available from: https://www.sciencedirect.com/science/article/pii/S0020025516314931.
[15] Gaemperle R, Mueller SD, Koumoutsakos P. A parameter study for differential evolution. In: Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation; WSEAS. WSEAS; 2002. p. 293–298. Proceedings of the 3rd WSEAS International Conference on Neural Networks and Applications (NNA '02), Fuzzy Sets and Fuzzy Systems (FSFS '02), Evolutionary Computation (EC '02); Available from: http://www.cse-lab.ethz.ch/wp-content/papercite-data/pdf/gaemperle2002a.pdf.
[16] Ronkkonen J, Kukkonen S, Price K. Real-parameter optimization with differential evolution. In: 2005 IEEE Congress on Evolutionary Computation; Vol. 1; 2005. p. 506–513 Vol.1.
[17] Storn R. On the usage of differential evolution for function optimization. In: Proceedings of North American Fuzzy Information Processing; 1996. p. 519–523.
[18] Tanabe R, Fukunaga A. Success-history based parameter adaptation for differential evolution. In: 2013 IEEE Congress on Evolutionary Computation; 2013. p. 71–78.
[19] Zhang J, Sanderson AC. Jade: Adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation. 2009;13(5):945–958.
[20] Cai X, Mei Z, Fan Z. A decomposition-based many-objective evolutionary algorithm with two types of adjustments for direction vectors. IEEE Transactions on Cybernetics. 2017; 48(8):2335–2348.
[21] Zhang Q, Liu W, Li H. The performance of a new version of moea/d on cec09 unconstrained mop test instances. In: 2009 IEEE congress on evolutionary computation; IEEE; 2009. p. 203–208.
[22] Qi Y, Ma X, Liu F, et al. MOEA/D with Adaptive Weight Adjustment. Evolutionary Computation. 2014 06;22(2):231–264. Available from: https://doi.org/10.1162/EVCO_a_00109.
[23] Zhang Q, Li H. Moea/d: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation. 2007;11(6):712–731.
[24] Farias LRC, Araújol AFR. Many-objective evolutionary algorithm based on decomposition with random and adaptive weights. In: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC); 2019. p. 3746–3751.
[25] Takagi T, Takadama K, Sato H. Weight vector arrangement using virtual objective vectors in decomposition-based moea. In: 2021 IEEE Congress on Evolutionary Computation (CEC); 2021. p. 1462–1469.
[26] Cai X, Mei Z, Fan Z. A decomposition-based many-objective evolutionary algorithm with two types of adjustments for direction vectors. IEEE Transactions on Cybernetics. 2018; 48(8):2335–2348.
[27] Cheng J, Yen GG, Zhang G. A many-objective evolutionary algorithm with enhanced mating and environmental selections. IEEE Transactions on Evolutionary Computation. 2015;19(4):592–605.
[28] Sun Y, Yen GG, Yi Z. Igd indicator-based evolutionary algorithm for many-objective optimization problems. IEEE Transactions on Evolutionary Computation. 2019;23(2):173–187.
[29] Li L, Yen GG, Sahoo A, et al. On the estimation of pareto front and dimensional similarity in many-objective evolutionary algorithm. Information Sciences. 2021;563:375–400.
[30] Tian Y, He C, Cheng R, et al. A multistage evolutionary algorithm for better diversity preservation in multiobjective optimization. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2021;51(9):5880–5894.
[31] Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation. 2002;6(2):182–197.
[32] Yuan Y, Xu H, Wang B, et al. Balancing convergence and diversity in decomposition-

based many-objective optimizers. IEEE Transactions on Evolutionary Computation. 2015; 20(2):180–198.
[33] Wang R, Zhang Q, Zhang T. Decomposition-based algorithms using pareto adaptive scalarizing methods. IEEE Transactions on Evolutionary Computation. 2016;20(6):821–837.
[34] Cheng R, Jin Y, Olhofer M, et al. A reference vector guided evolutionary algorithm for many-objective optimization. IEEE Transactions on Evolutionary Computation. 2016; 20(5):773–791.
[35] Liu Q, Jin Y, Heiderich M, et al. An adaptive reference vector-guided evolutionary algorithm using growing neural gas for many-objective optimization of irregular problems. IEEE Transactions on Cybernetics. 2020;52(5):2698–2711.
[36] Xiang Y, Zhou Y, Li M, et al. A vector angle-based evolutionary algorithm for unconstrained many-objective optimization. IEEE Transactions on Evolutionary Computation. 2016;21(1):131–152.
[37] Zhang Q, Zhou A, Zhao SZ, et al. Multiobjective optimization test instances for the cec 2009 special session and competition; 2009.
[38] Huband S, Hingston P, Barone L, et al. A review of multiobjective test problems and a scalable test problem toolkit. IEEE Transactions on Evolutionary Computation. 2006; 10(5):477–506.
[39] Ye T, Ran C, Zhang X, et al. Platemo: A matlab platform for evolutionary multi-objective optimization. IEEE Computational Intelligence Magazine. 2017;12(4):73–87.
[40] Wang H, Jin Y, Yao X. Diversity assessment in many-objective optimization. IEEE Transactions on Cybernetics. 2017;47(6):1510–1522.