A Novel Evolution Optimization Algorithm Using a Multidimensional Geometric Method: Pivot Optimiser

Somsin Thongkrairat and Vanvisa Chutchavong

Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand.

somsin.to@hotmail.com

***Abstract* - Many optimisation techniques have recently been developed. Several mimic natural activity or another theory, such as the Grey Wolf Optimiser, which emulates wolf hunting mechanisms to find a global minimum, and Particle Swarm Optimisation, which uses birds' flocking behaviour to avoid each local minimum. Each developed algorithm uses guidelines to improve its mimicry and reach its goal. This work proposes the pivot optimiser, a new evolution optimisation algorithm inspired by the multidimensional geometric method to create a unique evolution in each generation. The goal of this imitation is to make an algorithm suitable for a multi-situation problem with a stable result. The results show that the pivot optimiser outperformed on competitive problems compared with other competitive optimisers.**

Keywords- optimisation, algorithm, swarm intelligence, geometric, GWO, PSO

I. INTRODUCTION

Developing an algorithm to solve optimisation problems has been a significant and challenging topic in computer science research [1]. Over the past decade, a massive number of optimisation algorithms have been developed and researched [2]–[4].

Many successful optimisers have been suggested as well-known optimisation algorithms based on their ability to produce satisfactory results in terms of performance benchmarking. Estimating and comparing algorithms' performance is especially important because every algorithm was invented for an individual purpose by an individual researcher. Accordingly, a standard benchmarking method is necessary for each category. Several common optimisation problems occur in test functions, such as the Rastrigin, Ackley and Rosenbrock functions. Each function contains different situations and characteristics to challenge the algorithm's performance. The goal in the continued search for a fitness value is to find a global minimum [5] [6].

Meta-heuristic optimisers have become an immensely popular method to solve these problems for two main reasons:

1) their highly flexible implementation. Most competitive optimisation problems present in black box form [2]. The researcher can easily apply the suggested algorithm to any problem by simply adjusting the most practical representive parameter and adapting input(s) and output(s) to the correct form [7].

2) their simple concept design. Because of derivation-free mechanisms in the implementation of meta-heuristic algorithms (unlike most deterministic algorithms, which use gradient-based calculations such as compass search [8]), meta-heuristic algorithms require fewer parameters for calculation and can apply some stochastic methods directly [9].

In addition, many successful algorithms are meta-heuristic, including Particle Swarm Optimisation (PSO), Grey Wolf Optimiser (GWO) and Differential Evolution (DE). It is well known that meta-heuristic algorithms are an excellent choice for recently developed optimisers.

Solving an optimisation problem does not involve finding the exact best answer. Rather, it involves finding a satisfying answer or the nearest best answer. In fact, any problem has a nearly infinite number of possible solutions. It is impossible to know where the best solution is located, and no computation theory to date can comprehensively search all solutions in a continuous space. Many algorithms are developed from another theory or concept – such as Darwinian evolution, social behaviours and astronautics theory – that is deemed 'reasonable' enough to provide a solution. As mentioned above, the algorithm is not designed to support the optimisation problem directly; rather, the researcher adapts the existing theory to solve the problem. In these efforts, representation is a serious subject to consider. While the theory may itself be reasonable, without fair representation, the reasonableness may not fully extend to the algorithm. Since the algorithm's mechanism is replicated from the theory, choosing the correct theory is essential.

In the optimisation research literature, we found some characteristics that increase our algorithm's all-around performance for the searching area. Before the success of meta-heuristic algorithms, the golden age of the 'direct search' searching algorithm family reigned [10]. In optimisation, the direct search family of algorithms utilises a non-population base with a deterministic algorithm that defines some actual searching methods [11]. These include golden section search, the Nelder–Mead method and the Luus–Jaakola (LJ) optimisation procedure [12]–[14]. Direct search is not the most competitive algorithm for finding the best solution, but such algorithms have consistently found the

DOI 10.5013/IJSSST.a.21.04.13 ISSN: 1473-804x online, 1473-8031 print

expected average result. No matter the searching method, each algorithm will cover almost the entire searching area or boundaries, meaning that the decision vector of the resulting solution will locate nearly the best solution [15]. The mechanism leading to this phenomenon is the comprehensiveness of the algorithm's searching area. Accordingly, one of the indispensable characteristics of a good optimiser is its comprehensiveness. At present, the direct search algorithm family is no longer popular in the research field because of its complexity and lack of coherent mathematical analysis [16].

We believe that the secret behind the success of direct search is the theory implemented in these algorithms. While many algorithms mentioned above were inspired by existing principles, we are interested in one particular category: geometric theory. Geometric theory is concerned with the shape, size and relative position of figures as well as the properties of space [17]. This theory's main goal is to solve real-world measurement problems by simplifying them into analysable components, such as lines, points, planes, distances, angles and surfaces [18]. Geometric theory has a long history that dates back to the sixth century [19]. Because of the theory's wide application and logical nature, it continues to be used to the present day. For the above reason, we selected the geometric method as the inspiration for the algorithm implemented in our optimiser.

In this paper, we address the issue of novel evolutionary optimisation, applying the geometric method to ensure that the algorithm is able to retain the population's uniqueness and evolve using a reasonable method to produce an outstanding generation as a result of the optimisation problem. The existing related evolutionary algorithm (EA) is discussed in Section II. The theory and implemented methods are described in Section III. The simulation methods and results are presented in Section IV. Finally, Section V concludes the paper and suggests directions for future work.

II. RELATED WORK

EAs are among the most successful optimisation algorithm families [20]. These algorithms take advantage of the population base concept that allows the computation process to keep several potential solutions, called 'populations', and evolve them to become better solutions for the next generation of populations [21].

The EA family was inspired by biological evolution in nature [22]. The mechanisms of the evolutionary solution process thus resemble natural evolutionary methods, such as selection, mutation and crossover. EA's mimickry of natural evolutionary methods reinforces its distinction as one of the most powerful and reasonable methods to solve optimisation problems.

The genetic algorithm was the first evolution-inspired algorithm to be applied to optimisation [23]. It introduces the implementation of natural evolutionary theory [24]. This algorithm can feasibly be applied to any problem due to the reasonable nature of the algorithm itself.

Pietro et al. analysed the performance of the Simple EA (N+1-ES), which implements the simplest version of the population base algorithm by recombining the best and worst of the population as the candidate offspring and performing a decision process to replace members of the population pool [25].

Because biological evolution theory is neutral, some neutral evolution methods cannot be directly adapted for the optimisation context. R. Storn et al. developed an EA called DE whose combination method could adjust itself to fit the problem [26] [27]. This method adapts the neutral evolution method as the mathematical optimisation evolution method. The term 'mutation' refers to selecting one of the best individuals in the population and making some mathematical variance on its chromosome (decision vector) as a new candidate solution. The term 'crossover' refers to taking several populations and combining some of them, while the decision vector becomes the new candidate solution. Until recently, this method has been the prototype for EAs.

The above shows that DE has been highly sensible but demonstrates a lack of particles because the algorithm did not define the exact principle for creating offspring. Because users needed to define mathematical evolution methods to utilise in the algorithm, DE is overly complex in its application and unsuitable for some optimisation problems.

Farmani et al. [28] developed an algorithm called Self-Adaptive DE (SADE) that combines DE and mathematical evolution method and enables the algorithm to adapt the evolved method to suit the problem. It is a robust optimisation algorithm because it is suitable for most problem-solving styles without requiring the parameters to be adjusted. The benchmarking results are highly satisfactory compared to any other EA.

Another efficiency optimisation algorithm called swarm intelligence (SI) was inspired by nature, especially biological systems [29]. J. Kennedy et al. [30] developed an optimiser termed Paritcal swarm optimiser (PSO) by observing bird flock and fish school movement, then replicating these animals' formations and social interaction behaviours by representing particles as animals and representing the movement of an animal as the searching method in the system.

Schlüter et al. [31] introduced the Extended Ant Colony Optimisation (GACO). The actions of ants in their colonies inspired this algorithm. In it, artificial 'ants' are assigned to locate optimal solutions by moving through a parameter space representing all possible solutions and synchronising optimal solutions using the pheromone mechanism.

D. Karaboga et al. [32] introduced the Artificial bee colony algorithm (ABC). They analysed the foraging behaviour of honeybee swarms and determined that the bee population included three types: employed bees, who search among the best population; onlooker bees, who search across all populations; and scout bees, who locate new areas to search.

DOI 10.5013/IJSSST.a.21.04.13

ISSN: 1473-804x online, 1473-8031 print

S. Mirjalili et al. [33] developed GWO based on the concept of the leadership hierarchy and hunting mechanisms of grey wolves. GWO utilises encircling prey mechanisms to find the truly local minimum of a problem.

Because SI algorithms and EAs both use the population base concept, they may appear to be the same algorithm. However, their main ideas and implementation methods are completely different.

All mentioned algorithm above can be fairly recognised as the dominant algorithm because each offers the best performance in comparison with a successive algorithm. These algorithms are dominant due to the 'reasonableness' of their methods. GACO and ABC use the equilibrium of the population to distribute search possibilities across all space. GWO uses the leadership of the alpha wolf when hunting prey to obtain the global minimum. This reasonableness makes the optimiser efficient.

Another characteristic no less important than reasonableness is elitism. Elitism ensures that an algorithm produces a better solution for offspring in the next generation compared to the previous generation [34]. Elitist algorithms usually produce significantly better results and prevent the loss of satisfied offspring once produced [35] [36]. To ensure that an algorithm achieves elitism, the evolution and selection method must retain the 'uniqueness', or distinctive point, of each individual to guarantee that offspring will inherit only the good components in the next generation.

III. GEOMETRIC METHOD

This section first presents the geometric method implemented in the proposed optimiser. The details of the proposed pivot optimiser algorithm – in particular, the topic of mechanisms in algorithms and the justification for calling this algorithm 'sensible' – follow:

A. Theory

The standard form of a continuous optimisation problem is to minimise the objective function $F(x)$ [37]. The solution, or x, is a set that contains several values named the 'decision vector' according to the number of dimensions in the problem. In this paper, we will describe only single-objective optimisation; that is, the result of the objective function can only be a single real value. Finding the best solution is the goal of this optimisation.

As described above, the geometric method relates to points, lines and space. Many methods are standard methodologies to solve mathematical problems [38]. The main idea in applying this method to optimisation involves using the property in geometric terms to represent the property in the optimisation context as follows:

- Space in the Euclidean domain corresponds to the continued search space in the optimisation.
- Vector in space corresponds to the population by representing position in n-space as the decision vector in n-dimension.

Thus, all products of calculations can be represented directly in any direction. This section will rewrite theory in the form of multidimensional and computable expressions.

A1. Linear Distance and Angular Distance

The first and simplest method applied is distance. Any two points on the Euclidean domain (any n-dimension) can be measured with the ordinary straight-line distance between them using the Euclidean distance (ED), as in the following equation [39]:

$$D_{xy} = \sqrt{\sum_{i=1}^{n} \left(x_i - y_i\right)^2} \tag{1}$$

Where x and y are the points in the n-dimension, the distance between them can be calculated as shown in (1). This method also calculates the property named 'crowding distance', which, in some optimisation algorithms, defines the distance between solution x and solution y [40]. This property resembles linear distance but is used for the result solution, unlike the distance previously mentioned.

Another distance quantity is angular distance. Any vector in the same domain can be compared with another using the angle difference value according to the following equation:

$$\theta_{xy} = \cos^{-1}\left(\frac{x \cdot y}{|x||y|}\right) \tag{2}$$

where θ indicates the reference angle at the origin. In this paper, we intend to change the reference point to any position in space. Accordingly, we must bias the reference vector to x and y to move their reference point [Fig. 1]. This equation can be expressed as follows:

$$\theta_{rxy} = \cos^{-1}\frac{\sum_{i=1}^{n}\left(\left(x_i - ref_i\right)\times\left(y_i - ref_i\right)\right)}{\sum_{i=1}^{n}\left(x_i^2\right)\times\sum_{i=1}^{n}\left(y_i^2\right)} \tag{3}$$

The difference between any two points in multidimensional space can be calculated using Equation (3), where *ref* is the bias vector used to change the reference point from the origin to the *ref* point in n-dimensional space. This angle can represent the quality of betweenness for any point [41] and determine each individual's angular distance.

DOI 10.5013/IJSSST.a.21.04.13

ISSN: 1473-804x online, 1473-8031 print

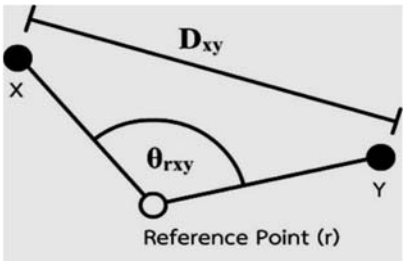

Figure 1. Linear distance and angular distance

Fig. 1 shows linear distance and angular distance in two-dimensional space. D_{xy} describes linear distance in the Euclidean domain for x and y, and θ_{xy} describes angular distance with bias from the reference point.

A2. Mahalanobis Distance

The ED is the absolute individual quantity of distance reference with the domain. However, in the case of population-based optimisation algorithms, there is another way to measure distance more effective.

The Mahalanobis distance (MD) is an alternative way to measure the distance between points in space using standard deviations for each dimension in the group of data [42]. It is thus a useful property that can explain the outlyingness of multivariate observations in the data set. MD can be calculated as follows:

$$MD_{xy} = \sqrt{\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{s_i^2}} \tag{4}$$

where S_i is the standard deviation of x and y for each dimension. However, MD is equal to ED if the dataset's bound is diagonal. In this paper, we use MD to eliminate unnecessary populations located too close to one another.

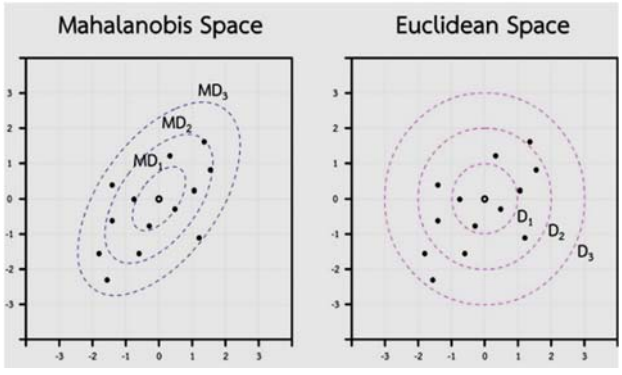

Figure 2. Mahalanobis space compared with Euclidean space

Fig. 2 shows the distinction between the same member in the Mahalanobis and Euclidean domains in two-dimensional space. The three contours on both sides represent the same distance in different domains. D_1, D_2 and D_3 are the one-, two- and three-unit contours in the Euclidean domain. MD_1, MD_2 and MD_3 represent the same distances as D_1, D_2 and D_3, respectively, in Mahalanobis space.

If the distribution of coordinates is not uniform, the ED and MD will not be equal. The MD domain's contour and distance trend will thus follow all members in space and can be used to classify the group of data [42].

A3. Uniform and Normal Distribution

Almost all meta-heuristic optimisers use a random method to create new solutions, unlike deterministic optimisers, which create variances to increase the possibility of finding a better solution. The simplest random method applied in optimisers is random uniform distribution, which equally distributes all possible output values [43]. The probability density function of random uniform distribution is presented in Equation (5) below:

$$P(i \mid a,b) = \frac{1}{b-a} \tag{5}$$

Equation (5) shows the parameters of the uniform random method. When a is the lower bound and b is the upper bound, the random output is equally likely to be any value between a and b. Many evolutionary optimisers calculate upper and lower bounds as a and b and use this kind of random method to evolve a new decision vector for each dimension. From our point of view, the use of this method sometimes creates a unreasonable population. In multidimensional optimisation, every new decision vector for each dimension should be created according to the same trend. However, in this random method, each dimension is not related to the others. To force the random values to follow the same trend, we suggest normal or Gaussian distribution with some control variables, as follows:

$$P(x \mid \mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{6}$$

In Equation (6), μ is the mean output value, and σ controls the standard deviation for control of variance. A normal distribution is usually applied in statistics and is often used in the natural and social sciences. We chose the normal random method as reasonable because it can control the trend of randomness using several parameters.

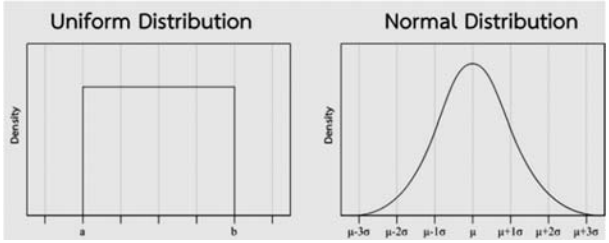

Figure 3. Difference between uniform and normal distributions

DOI 10.5013/IJSSST.a.21.04.13

ISSN: 1473-804x online, 1473-8031 print

Fig. 3 shows the output density of uniform distribution from (5). The range of output data is between a and b, and the possibility of all numbers is equal. In the normal distribution as calculated in (6), the highest density of distribution is μ. As such, the highest possible number is μ, and the possibility of distribution will decrease as the distance from μ increases. The σ variable controls the standard deviation of the random method.

B. Proposed Method: Pivot Optimiser

We developed the pivot optimiser based on the above-described theory. One pivot represents one member of the population pool. Two types of pivots exist: potential and control. While both use the same mechanism to search for the best solution, they have different purposes. Potential pivots are used to find the potential answer or best answer of the optimiser, while control pivots separate search ranges to cover all search space and prevent other pivots from repeatedly searching around unproductive areas. Next, the pivot optimiser algorithm is outlined.

B1. Pivot

In the proposed method, the pivot represents the individual population in the population pool. Like a typical optimiser, Each pivot contains a decision vector to search for the best solution, as well as several specific properties that relate to other pivots, including:

- Linear distance list;
- Angular distance list;
- Nearest pivot (NP); and
- Rival pivot (RP).

The linear distance list is the ED list from the pivot to other neighbourhood pivots in the system sorted in ascending order. The NP is the closet pivot measured by linear distance. The first pivot in the linear distance order is thus the NP, and the NP position is used to determine further properties.

From this point, other populations will be considered Candidate pivots (CPs) because every other pivot is used to determine the essential property of the RP.

Next, the angular distance list is the list of angles between the NP and other CPs using the considered pivot as the reference, following (3). Finally, the RP is selected by the nearest neighbours placed in opposite areas of the NP or the angle between the NP and CP > π.

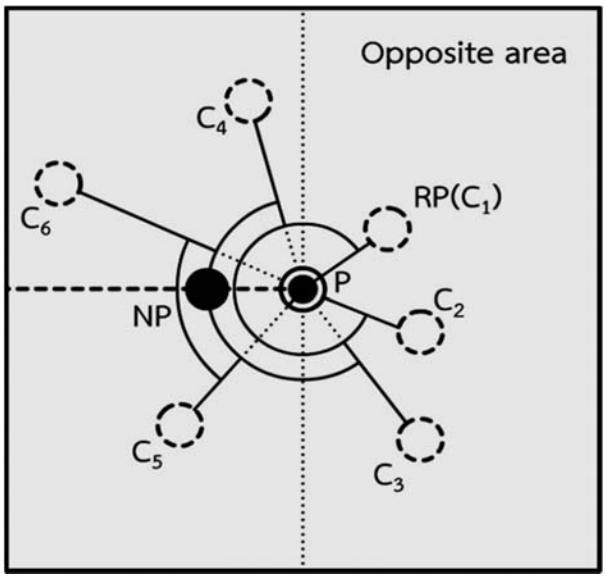

Figure 4. Properties of each pivot

Fig. 4 shows an example of the considered Pivot (P) with neighbourhood pivots and its NP and RP. Assume that there are eight pivots in the system and let P be the considered point. NP is the nearest pivot of P, and another pivot (C_1–C_7) is the CP, but RP (C_1) is the nearest pivot in the opposite site. Accordingly, C_1 is selected as the RP.

Nevertheless, the system sometimes cannot find an RP because, in large dimensional problems, the possibility of finding another pivot in the opposite area in all dimensions is decreased, dependent on the number of populations. In these cases, the last population member in the angular distance list will be selected as the RP.

Lastly, every pivot in the system will select its NP and RP to find the next result. The RP is the main path to determine the search range for the next solution because, in stochastic optimisation, it is difficult to conclude that the current position is the local minimum. As such, there are usually some locations around pivots that can offer a better solution. In the proposed method, RP and NP are used to calculate the search range around the pivot to find a better solution for the considered pivot.

B2. Offspring Creation

After each RP is selected, every pivot creates its offspring to serve as the next generation of the population. To classify pivots by their roles, all pivots will be sorted in ascending order by answer solution before being divided into two groups: lower and upper, controlled by a Lower factor (LF) and Upper factor (UF), respectively. LF is the ratio of the NP to the lower group, while UF is the ratio of NP to the upper group. LF and UF are adjustable depending on the problem but are set to 0.25 by default.

DOI 10.5013/IJSSST.a.21.04.13

ISSN: 1473-804x online, 1473-8031 print

The lower group is called a potential pivot. This group contains an answer and decision vector suitable for finding a better solution. However, because the members of the lower group are produced and sorted, a high chance exists that the population will be clustered rather than spread across the full search space. The offspring from this group is called a potential solution.

The upper group is called the control pivot. This group controls the search area to find new areas to search and push lower groups out of the local minimum. The new population created from the upper group is called a control solution.

Another pivot between two groups is called an intermediate pivot. It is used to memorise the search history and prevent searches of unproductive areas and overly close pivots. Intermediate pivots were previously part of the lower or upper group due to the sorting mechanism. Accordingly, almost all of them are analysed in the lower and upper group phase.

Usually, a competitive optimiser creates a population in the generation cycle [44]. One period of creation or one generation will produce a new population equal to the number of populations. In the proposed optimiser, we decide to produce offspring equal to the number of parents, as with the usual optimisers.

Only pivots in the upper group and lower group are used to create offspring. The Potential factor (PF) is used to decide the number of populations in each group. PF is the population ratio of the lower and upper groups selected to be the parent. The number of pivots is represented by *cn* for the lower group, calculated as (*PF x NP*), and *pn* for the upper group, calculated as (*(1-PF) x NP*). The population in both groups will be selected randomly by uniform distribution following *cn* and *pn*.

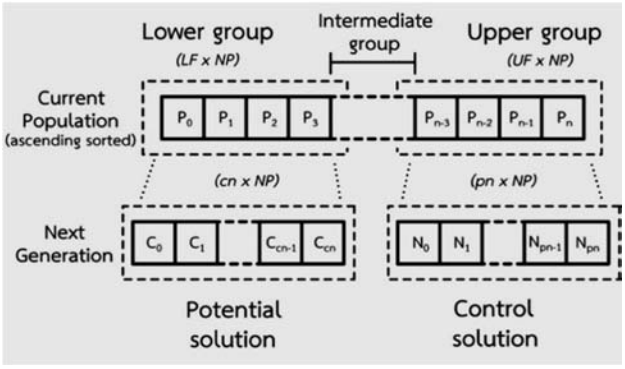

Figure 5. Population pool and next generation

Fig. 5 explains the population pool. In the current population pool, the lower group consists of the first pivot through the pivot number (*LF x NP*), and the upper group consists of the pivot number (*NP - UF x NP*) through the last pivot. For the offspring, potential and control solutions will be created, and the number of all populations will double.

For each pivot selected as a parent, the NP and RP's decision vectors are used to determine the offspring. First, the central search point, or CSP, is calculated by finding the centre point between the NP and RP for each dimension. In the proposed method, we suppose that the CSP offers the most favorable position for finding a better position around considered pivot because if the pivot did not reach the local minimum, there is always somewhere around the pivot that can offer a better answer. By taking a combination of positions for a pivot, the RP and NP are highly likely to produce a better new solution. The range from the centre point for finding new solutions is determined using the range from the NP to the RP. Offspring are probabilistically constructed in a normal distribution following (6), setting the μ parameter to the CSP and σ to the search range. The random output thus has a high chance of being near the CSP and spanning the search range. Fig. 6 describes the position of the CSP and range for the search area in two-dimensional space.

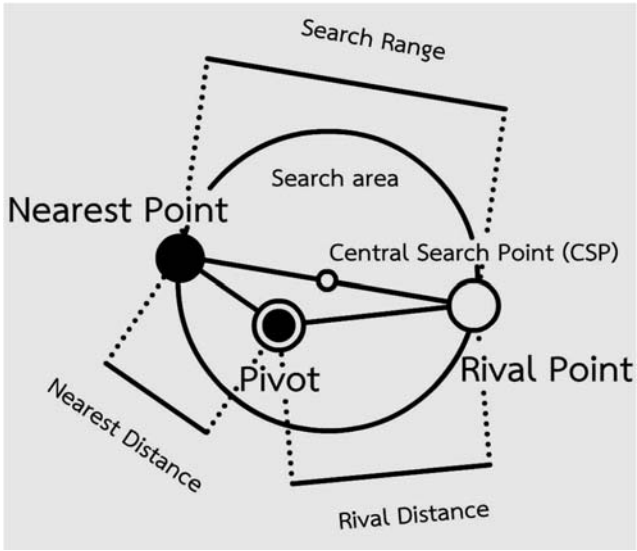

Figure 6. Search range calculation in two-dimensional space

B3. Selection (Remove Unnecessary Population)

When the new generation of the population is created, the proposed method selects the pivot that has proved useful for the future generation. Meanwhile, it removes some wasteful pivots. Two properties are used to indicate useful pivots and retain them for the next generation: border pivots and non-redundant pivots.

Border pivots are the pivots located at the border of the group measured by angular distance. Suppose there is no pivot in the opposite area from the considered pivot. The considered pivot is then the border pivot. The border pivot may not give a good result, but it can control the area for discovering a new local minimum.

A redundant pivot is a pivot placed around the group that contains the better result. This kind of pivot must be eliminated because it contains a bad decision vector and may affect neighbourhood pivots.

DOI 10.5013/IJSSST.a.21.04.13

ISSN: 1473-804x online, 1473-8031 print

To indicate the border and non-redundant pivots, the RP and NP are calculated by applying MD rather than ED, since MD is more closely related to the population. In this phase, the RP and NP in the MD domain are calculated for the new population pool. The border pivot is the pivot that can find the NP and RP in the opposite area. The redundant pivot is the pivot that achieved an answer less favorable than its NP and RP.

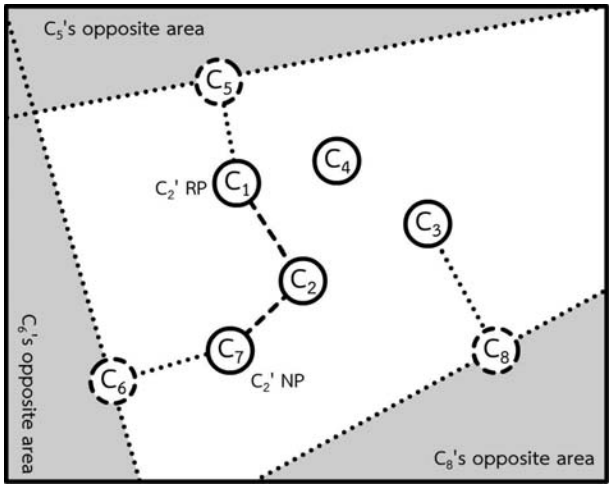

Figure 7. Population with border pivot and redundant pivot

Fig. 7 shows an example of a possible redundant pivot (C_2) and its border pivots in two-dimensional space. The pivots C_5, C_6 and C_8 are the border pivots because no other pivot appears in their opposite areas. For example, at C_2, there is NP (C_7) and RP (C_1). The pivot C_2 will be redundant if the results from C_1 and C_7 are better than C_2.

All populations in the previous and current phase will then be merged. This process will be applied to all populations except the lower group. We allow this group to nearly find each other and reach a local minimum if the considered pivot is a redundant pivot and not the border pivot. These results will update the list of pivots to be removed when the process is finished.

Sometimes, the total number of remaining populations may be less or more than the number of initial populations. The following process will compensate for the modified pivot.

B4. Population Compensation (Fill or Remove Population)

After the selection phase, two cases may occur in the population pool. The first is that the number of remaining populations is greater than the initial number, meaning that many individuals are useful. In this case, we must remove the extra population from the ascending sorted list. The other case is that the remaining population is less than the initial number. In this case, the population must be filled.

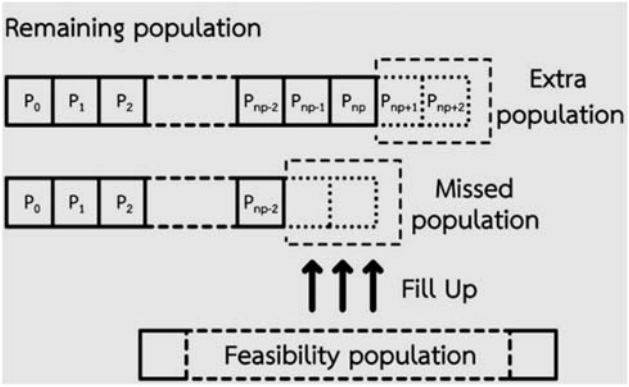

Figure 8. Remaining population management diagram

Fig. 8 shows an example of the remaining population in two cases. In the case of extra population, P_{np+1} and P_{np-+2} must be deleted, whereas in the case of missing population, P_{np} and P_{np-1} will be filled with a feasibility population.

We consider the new population to be filled based on two properties. The first is the boundary. We assume that pivots are located at the corner of all boundaries and select the farthest pivot from the considered pivot as the Boundary pivot (BP). Because the BP will provide an extreme solution for each boundary, it is used to locate an additional searching field for new local optimum areas. Accordingly, the area between the considered pivot and the BP may indicate the trend of results as well as a new optimising area. The BP centre point (BC) is measured using the centre point between the considering pivot and the BP.

The second property is the farthest pivot in the population pool. The Farthest pivot centre point (FC) is the centre point between the considered pivot and the farthest pivot measured by ED. The remaining population in this phase only includes useful members, as determined in the previous phase. A combination of decision vectors may thus discover omitted areas between pivots.

The BC and FC produce a new population group for each pivot. The possible boundaries area is produced using the BC as the centre point and the distance between the BC and the pivot as the search range with the normal distribution random method for all dimensions. The farthest possible population is produced using the FC and distance from the FC and the pivot, similar to the BC process.

Lastly, two groups emerge as feasibility populations and are randomly filled with the main population pool until the population reaches the initial population. This process will repeat until the number of the main population reaches the maximum.

DOI 10.5013/IJSSST.a.21.04.13

ISSN: 1473-804x online, 1473-8031 print

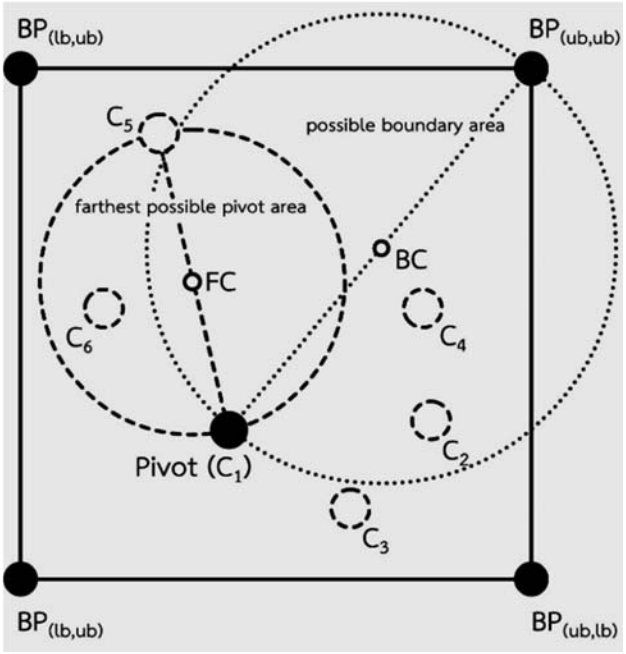

Figure 9. Area of feasibility population in two-dimensional space

Fig. 9 describes the searching area of the feasibility population for considering pivot C_1 in two-dimensional space. There are four BPs due to the combination of the upper bound (*ub*) and lower bound *(lb*) in two dimensions. In Fig. 9, the farthest BP is $BP_{(ub, ub)}$, so the BC is located between C_1 and $BP_{(ub, ub)}$. C_5 is the farthest pivot from C_1 and is used to indicate the FC and its search area, as shown in Fig. 9.

B5. Process Flow

Steps 1 through 4 will run repeatedly until the requirement is reached. Usually, the number of function evaluations is used as a requirement. The pseudocode for the process of the pivot optimisation algorithm is presented below in Fig. 10.

```
0.)  Prepare dataset from origin population

REPEAT UNTIL REQUIREMENT MET
    1.) Prepare geometric data
            a. Build angular distance list
            b. Sort geometric data by distance
    2.) Create offspring
            a. Select population
            b. Determine search range
            c. Merge new population
    3.) Remove unnecessary population
    4.) Adjust number of populations
```

Figure 10. Pseudocode for pivot optimisation algorithm

For useful implementation, computational complexity or big O notation must be considered. Acceptable complexity for a single-objective optimisation algorithm is suggested as $O(N^3)$ where N is the population's size [45]. Following the pseudocode in Fig. 10, which shows the algorithm's process flow when considering one generation, Step 1a takes time $O(N^2)$ but is dominated by Step 2a, which takes time $O(N^2 \log N)$ because the sorting algorithm takes time $O(N \log N)$ and N individuals to sort. Steps 2, 3 and 4 take time $O(N^2)$ for each process. Accordingly, the dominant complexity is $O(N^2 \log N)$, which is acceptable for the suggested complexity.

IV. EXPERIMENTAL RESULTS

The pivot optimiser algorithm was implemented and benchmarked with several well-known test functions. The implementation method and analysis of the results are presented in this section.

A. Implementation and Benchmarking

The proposed algorithm was implemented, and results were recorded using pagmo. Pagmo is a scientific library for massively parallel optimisation developed in C++ [46]. It contains a testing environment and algorithms that can be used to research the performance of optimisers. A custom algorithm can also be added to the environment to compare one optimiser with another and a different test suite.

This paper selects five standard test functions, as shown in Table 1: Rosenbrock, Rastrigin, Schwefel, Ackley and Griewank. Another four well-known optimisers (GWO, PSO, DE and ABC) were selected to compare performance.

TABLE I. BENCHMARK FUNCTION

Name	Function
Rosenbrock	$f_1 = \sum_{i=1}^{D-1}\left[100\left(x_i^2 - x_{i+1}\right)^2 + \left(x_i - 1\right)^2\right]$
Rastrigin	$f_2 = 10 \cdot D + \sum_{i=1}^{D} x_i^2 - 10 \cdot \cos\left(2\pi \cdot x_i\right)$
Schwefel	$f_3 = 418.9828872724338n - \sum_{i=1}^{D} x_i \sin\sqrt{\lvert x_i \rvert}$
Ackley	$f_4 = 20 + e - 20e^{-\frac{1}{5}\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}} - e^{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)}$
Griewank	$f_5 = \sum_{i=1}^{D} x_i^2 / 4000 - \prod_{i=1}^{D} \cos\frac{x_i}{\sqrt{i}}$
Katsuura (CEC2014 No.12)	$f_6 = \frac{10}{D^2}\prod_{i=1}^{D}\left(1 + i\sum_{j=1}^{32}\frac{\lvert 2^j - round(2^j x_i)\rvert}{2^j}\right)^{D^{1.2}} - \frac{10}{D^2}$
HappyCat (CEC2014 No.13)	$f_7 = \left\lvert \sum_{i=1}^{D} x_i^2 - D \right\rvert^{1/4} + \left(0.5\sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i\right)/D + 0.5$

DOI 10.5013/IJSSST.a.21.04.13

ISSN: 1473-804x online, 1473-8031 print

DE represents the outstanding performance of an EA in solving unconstrained optimisation problems. PSO, GWO and ABC represent SI optimisers. PSO is the first successful well-known SI algorithm, while ABC is a current successful SI algorithm inspired by bee colonies. GWO is another successful SI algorithm, which is frequently implemented in engineering problems since both engineering problems and part of the proposed algorithm use sorting population mechanisms.

We tested 10 and 20 dimensions for all test functions to represent low and high dimensions. To make a fair comparison, the same number of function evaluations were used for all optimisers and set to 10,000 * *D* (where *D* is a dimension of the problem) according to the suggested benchmark method [47].

Each function was run 25 times. The average best result and SD were recorded for each test function. The results of the benchmark function are shown in Table 2.

B. Discussion of Results

As shown in Table 2, the pivot optimiser provides suitable results compared to other algorithms. For test functions 1 to 5, our proposed method's results fall between the best and worst results. DE performed best, producing the greatest number of unimodal test functions. On test function 4 (the hard multimodal function), SI performed better than DE.

However, the important point regarding the proposed method is the low variance and stability of its results according to its purpose. Table 2 shows the outstanding performance of the pivot optimiser's SDs compared with other optimisers. DE generally has a good average result, but its variance is also high, as shown by the high value of its SDs. While SI may perform less favorably, it is more stable.

The results of test functions 6 and 7 show the competitiveness of the present test function results. All optimisers had outstanding results, and the proposed optimiser's SD means that it can contest with other well-known algorithms.

V. CONCLUSION

We have developed a new evolution optimisation algorithm with the aim of producing a unique evolution in each generation using the multidimensional geometric method as a sensible algorithm.

To validate the proposed algorithm's performance, several related and well-known algorithms were selected to benchmark and compare results. The multi-situation testing simulation results show that the proposed optimiser is suitable for optimising the general problem with a stable result.

The algorithm's complexity is acceptable but is also its most significant problem due to optimisation speed. It should be developed to improve feasibility of future real-world implementation. We plan to develop the multi-objective version of pivot optimisation and improve its complexity to handle various problems in future work

TABLE II. RESULTS OF BENCHMARK FUNCTION

Function		***Pivot Optimiser***		***GWO***		***PSO***		***DE***		***ABC***	
Name	**Dim**	***Avg.***	***SD***	***Avg.***	***SD***	***Avg.***	***SD***	***Avg.***	***SD***	***Avg.***	***SD***
F1 Rosenbrock	10	1.560E-2	6.198E-01	5.866E+00	6.101E-01	3.767E-02	1.652E-02	1.020E-12	4.382E-12	2.947E-01	1.682E-01
	20	5.642E+0	2.177E+00	1.592E+01	7.570E-01	4.753E+00	3.574E+00	3.386E-01	9.457E-01	7.570E-01	3.790E-01
F2 Rastrigin	10	1.362E-4	4.245E-01	0*	0*	3.184E+00	2.321E+00	1.983E-08	2.199E-08	3.755E-06	1.140E-05
	20	5.076E+0	2.030E+00	0*	0*	1.490E+01	4.458E+00	4.450E-08	2.483E-08	3.228E-01	4.242E-01
F3 Schwefel	10	6.145E+2	1.663E+00	1.322E+03	2.857E+02	3.121E+02	1.748E+02	1.557E-08	1.284E-08	2.872E+01	6.055E+01
	20	2.556E+3	5.833E+00	3.812E+03	4.953E+02	1.692E+03	3.191E+02	4.752E-08	2.919E-08	3.301E+02	1.163E+02
F4 Ackley	10	3.654E-8	1.842E-10	3.996E-16*	0*	3.144E-15	1.517E-15	1.396E-07	6.227E-08	6.752E-14	3.633E-14
	20	4.452E-5	2.720E-03	5.844E-15*	1.774E-15*	3.997E-05	0	3.103E-07	1.354E-07	1.290E+08	4.053E-08
F5 Griewank	10	1.563E-2	4.173E-03	8.259E-03	2.045E-02	2.619E-02	1.273E-02	3.416E-13	1.602E-12	7.288E-03	9.067E-03
	20	3.560E-3	6.242E-04	1.398E-03	3.809E-03	3.852E-03	5.950E-03	4.180E-08	4.375E-08	9.922E-04	3.180E-03
F6 Katsuura CEC2014 No.12	10	1.200E+3	4.802E-01	1.201E+03	4.794E-01	1.200E+03	4.908E-02	1.200E+03	3.072E-02	1.200E+03	9.208E-02
	20	1.200E+3	4.802E-01	1.201E+03	8.194E-01	1.200E+03	2.144E-02	1.200E+03	6.465E-03	1.200E+03	3.030E-02
F7 HappyCat CEC2014 No.13	10	1.300E+3	5.201E-01	1.300E+03	6.768E-02	1.300E+03	3.493E-02	1.300E+03	2.109E-02	1.300E+03	4.097E-02
	20	1.300E+3	5.201E-01	1.300E+03	7.569E-02	1.300E+03	4.828E-02	1.300E+03	4.585E-02	1.300E+03	4.216E-02

* The results of GWO in F2 and F4 were not used because, in this version of pagmo, GWO is overpowered for some origin-based answer.

DOI 10.5013/IJSSST.a.21.04.13

ISSN: 1473-804x online, 1473-8031 print

REFERENCES

[1] J. J. Liang and P. N. Suganthan, "Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constraint-Handling Mechanism," 2006 IEEE International Conference on Evolutionary Computation, Vancouver, BC, 2006, pp. 9-16, doi: 10.1109/CEC.2006.1688284. W.-K. Chen, Linear Networks and Systems. Belmont, CA, USA: Wadsworth, 1993, pp. 123–135.

[2] Wu, Guohua, R. Mallipeddi, and P. N. Suganthan. "Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization." National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report (2017).

[3] K. Deb, L. Thiele, M. Laumanns and E. Zitzler, "Scalable multi-objective optimization test problems," Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600), Honolulu, HI, USA, 2002, pp. 825-830 vol.1, doi: 10.1109/CEC.2002.1007032.

[4] [Bujok, Petr, and Aleš Zamuda. "Cooperative model of evolutionary algorithms applied to CEC 2019 single objective numerical optimization." 2019 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2019.

[5] Back, Thomas. Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford university press, 1996.

[6] Houck, Christopher R., Jeff Joines, and Michael G. Kay. "A genetic algorithm for function optimization: a Matlab implementation." Ncsu-ie tr 95.09 (1995): 1-10.

[7] Lee, Kang Seok, and Zong Woo Geem. "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice." Computer methods in applied mechanics and engineering 194.36-38 (2005): 3902-3933.

[8] Xiang Zhong, Wenhui Fan, Jinbiao Lin and Zuozhi Zhao, "A novel multi-objective compass search," 2010 IEEE International Conference on Progress in Informatics and Computing, Shanghai, 2010, pp. 24-29, doi: 10.1109/PIC.2010.5687962.

[9] Dorigo, Marco, and Gianni Di Caro. "Ant colony optimization: a new meta-heuristic." Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406). Vol. 2. IEEE, 1999.

[10] Lewis, Robert Michael, Virginia Torczon, and Michael W. Trosset. "Direct search methods: then and now." Journal of computational and Applied Mathematics 124.1-2 (2000): 191-207.

[11] Powell, Michael JD. "A direct search optimization method that models the objective and constraint functions by linear interpolation." Advances in optimization and numerical analysis. Springer, Dordrecht, 1994. 51-67.

[12] Tsai, C. H., Joseph Kolibal, and Ming Li. "The golden section search algorithm for finding a good shape parameter for meshless collocation methods." Engineering Analysis with Boundary Elements 34.8 (2010): 738-746.

[13] Lagarias, Jeffrey C., et al. "Convergence properties of the Nelder--Mead simplex method in low dimensions." SIAM Journal on optimization 9.1 (1998): 112-147.

[14] Liao, Bo, and Rein Luus. "Comparison of the Luus–Jaakola optimization procedure and the genetic algorithm." Engineering optimization 37.4 (2005): 381-396.

[15] Luus, Rein, and T. H. I. Jaakola. "Optimization by direct search and systematic reduction of the size of search region." AIChE Journal 19.4 (1973): 760-766.

[16] Kolda, Tamara G., Robert Michael Lewis, and Virginia Torczon. "Optimization by direct search: New perspectives on some classical and modern methods." SIAM review 45.3 (2003): 385-482.

[17] De Risi, Vincenzo, ed. Mathematizing space: The objects of geometry from antiquity to the early modern age. Birkhäuser, 2015.

[18] Tabak, John. Geometry: the language of space and form. Infobase Publishing, 2014.

[19] Kahn, Charles H. Pythagoras and the Pythagoreans. Hackett Publishing, 2001.

[20] Karaboga, Dervis, and Bahriye Basturk. "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm." Journal of global optimization 39.3 (2007): 459-471.

[21] R. Sarker, M. Mohammadian, X. Yao. Evolutionary Optimization, Kluwer Academic Publishers, Norwell, MA, USA,2002.

[22] Alba, Enrique, and Marco Tomassini. "Parallelism and evolutionary algorithms." IEEE transactions on evolutionary computation 6.5 (2002): 443-462.

[23] Oliveto, Pietro S., Jun He, and Xin Yao. "Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results." International Journal of Automation and Computing 4.3 (2007): 281-293.

[24] D. E. Goldberg. Genetic Algorithms for Search, Optimization, and Machine Learning, Addison-Wesley, USA, 1989.

[25] Oliveto, Pietro S., Jun He, and Xin Yao. "Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results." International Journal of Automation and Computing 4.3 (2007): 281-293.

[26] R. Storn and K. V. Price, "Differential evolution–A simple and efficient adaptive scheme for global optimization over continuous spaces," Technical Report, Computer Science Institute, University of California at Berkeley, USA, TR-95-012, 1995.

[27] Storn, Rainer, and Kenneth Price. "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces." Journal of global optimization 11.4 (1997): 341-359.

[28] Farmani, Raziyeh, and Jonathan A. Wright. "Self-adaptive fitness formulation for constrained optimization." IEEE transactions on evolutionary computation 7.5 (2003): 445-455.

[29] Beni, Gerardo, and Jing Wang. "Swarm intelligence in cellular robotic systems." Robots and biological systems: towards a new bionics?. Springer, Berlin, Heidelberg, 1993. 703-712.

[30] Kennedy, James, and Russell Eberhart. "Particle swarm optimization." Proceedings of ICNN'95-International Conference on Neural Networks. Vol. 4. IEEE, 1995.

[31] Schlüter, Martin, Jose A. Egea, and Julio R. Banga. "Extended ant colony optimization for non-convex mixed integer nonlinear programming." Computers & Operations Research 36.7 (2009): 2217-2229.

[32] Karaboga, Dervis, and Bahriye Basturk. "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm." Journal of global optimization 39.3 (2007): 459-471.

[33] Mirjalili, Seyedali, Seyed Mohammad Mirjalili, and Andrew Lewis. "Grey wolf optimizer." Advances in engineering software 69 (2014): 46-61.

[34] Simon, Dan. Evolutionary optimization algorithms. John Wiley & Sons, 2013.

[35] Rudolph, G. "Evolutionary search under partially ordered sets." Dept. Comput. Sci./LS11, Univ. Dortmund, Dortmund, Germany, Tech. Rep. CI-67/99 (1999).

[36] Deb, Kalyanmoy, et al. "A fast and elitist multi-objective genetic algorithm: NSGA-II." IEEE transactions on evolutionary computation 6.2 (2002): 182-197.

[37] Boyd, Stephen, Stephen P. Boyd, and Lieven Vandenberghe. Convex optimization. Cambridge university press, 2004.

[38] Manning, Henry. Geometry of four dimensions. Applewood Books, 2011.

[39] Danielsson, Per-Erik. "Euclidean distance mapping." Computer Graphics and image processing 14.3 (1980): 227-248.

[40] Bechikh, Slim, et al. "Preference incorporation in evolutionary multi-objective optimization: a survey of the state-of-the-art." Advances in Computers. Vol. 98. Elsevier, 2015. 141-207.

[41] Andoni, Alexandr, et al. "Practical and optimal LSH for angular distance." Advances in neural information processing systems. 2015.

[42] De Maesschalck, Roy, Delphine Jouan-Rimbaud, and Désiré L. Massart. "The mahalanobis distance." Chemometrics and intelligent laboratory systems 50.1 (2000): 1-18.

[43] Kuipers, Lauwerens, and Harald Niederreiter. Uniform distribution of sequences. Courier Corporation, 2012.

[44] Fogel, David B. "An introduction to simulated evolutionary optimization." IEEE transactions on neural networks 5.1 (1994): 3-14.
[45] Zitzler, Eckart, and Lothar Thiele. "Multi-objective optimization using evolutionary algorithms—a comparative case study." International conference on parallel problem solving from nature. Springer, Berlin, Heidelberg, 1998.
[46] Biscani et al., (2020). A parallel global multi-objective framework for optimization: pagmo. Journal of Open Source Software, 5(53), 2338, https://doi.org/10.21105/joss.02338
[47] Liang, Jing J., Bo Y. Qu, and Ponnuthurai N. Suganthan. "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization." Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore 635 (2013).