國立臺灣大學電機資訊學院電機工程學系
碩士論文
Department of Electrical Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Master Thesis

基於子空間映射與多臂吃角子老虎機技術之實數最佳化

Real-valued Optimization by Subspace Projection and Multi-armed Bandit Techniques

彭俊人
Chun-Jen Peng

指導教授：于天立博士
Advisor: Tian-Li Yu, Ph.D.

中華民國 106 年 7 月
July, 2017

摘要

本論文提出了一新的實數多模態 (multimodal) 最佳化方法。大多數的現實問題都可以由多個單峰問題合成。而對於實數最佳化，我們較感興趣的是多模態且可被拆解成多個階層化的單峰問題。然而，一旦面對多模態問題，我們就必須面對利用性 (exploitation) 與探索性 (exploration) 的問題。本論文提出一解決多模態問題的技巧。此技巧結合了階層式聚合 (hierarchically clustering) 與最小描述長度 (minimum description length)，來辨識搜尋空間中可能的單峰模型。接著，我們使用一加一演化策略 ((1+1)-Evolutionary Strategy) 來最佳化一齊次座標 (homogeneous coordinate) 上的線性轉換矩陣，並利用此矩陣將原搜尋空間投影到另一有良好邊界定義的子空間。此投影嘗試定義出只包含一單峰之感興趣區域 (region of interest)。最後，我們提出一新的多臂吃角子老虎 (multi-armed bandit) 技術，來分配給各個子空間的資源，以最大化獲得全域最佳解的機率，我們將此結果與自適應共變異數矩陣演化策略 (Covariance Matrix Adaptation Evolution Strategies, CMA-ES)、標準粒子群演算法 (Standard Particle Swarm Optimization 2011)、與實數域蟻群演算法 (Ant Colony Optimization for Continuous Domain, ACO_R) 結合，並利用 CEC 2005 實數最佳化測試問題來比較結果。

Abstract

This thesis presents a new technique for real-valued multimodal optimization. Most of the real-world problems can be described as a composition of uni-modal subproblems. For real-valued optimization, we are interested in problems that are composed of *multiple hierarchically decomposable* uni-modals. However, allocating resources to exploit different uni-modals leads to the common delimma between *exploration* and *exploitation*. This thesis proposed some new techniques that aim to solve multimodal problems more efficiently. A new technique combining hierarchical clustering and minimum description length (MDL) is proposed to help identify potential uni-modals in the search space. Then, a linear transform matrix in homogeneous coordinate, optimized by the (1+1)-Evolutionary Strategy, projects the search space to a subspace with well-defined boundary. This projection aims to define a region of interest (ROI) that isolates a uni-modal subproblem. Finally, a new multi-armed bandit technique, aiming to maximize the probability of aquiring the global optimium, is proposed to allocate resources for each uni-modal. We combined our new techniques with Covariance Matrix Adaptation Evolution Strategy (CMA-ES), Standard Particle Swarm Optimization (SPSO) 2011, and Ant Colony Optimization for Continuous Domain (ACO_R). The results are evaluated with the CEC2005 Special Session on Real-Parameter Optimization benchmark problems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Attention plays an important role in human vision. For example, when we look at an image, our eye movements comprise a succession of *fixations* (repetitive positioning of eyes to parts of the image) and *saccades* (rapid eye jump). Those parts of the image that cause eye fixations and capture primary attention are called *regions of interest* (ROIs). Studies in visual attention and eye movement have shown that humans generally only attend to a few ROIs. Detecting these visually attentive regions in images is challenging but useful in many multimedia applications, such as automatic thumbnail cropping, object recognition, content-based image retrieval, adaptive image compression and automatic browsing in small-screen devices.

Describe the common dilemma between exploration and exploitation for real-valued optimization algorithms. Describe our basic assumption that problems worth solving are hierarchical decomposable [**?**].

1.1 Thesis Objectives

We propose a technique that helps identify *regions of interest* (ROI) to explore and *allocate resources* according to remaining evaluations.

First, describe why is it important to identify ROIs. Describe how subspace projection creates a *well-defined boundary* that some algorithms need. Describe how subspace projection helps solve *inseparable problems* while enhance the ability to find optimium.

Second, describe how the proposed resources allocation benefits optimization. Describe different strategies one should take given different evaluations left. Describe how Multi-armed Bandit (MAB) algorithms are suitable for the scenario, instead of decision theory, reinforcement learning and Markov Decision Process. MAB learns models from outcomes while the actions do not change the state of the world.

After identifying the potential uni-modals in the search space, we also need to determine the resource to invest in each promising region in order to find the global optimum. However, for a fixed amount of total evaluations, spending more time searching on one hill implies less attention on exploring other possible uni-modals in the search space. This is the common delimma between *exploration* and *exploitation* for all real-valued optimization algorithms.

1.2 Roadmap

This thesis is composed of seven chapters.

Chapter 2 presents three optimization algorithms that are adopted for comparisons. These three algorithms each have different characteristics. The Covariance Matrix Adaptation Evolutionary Strategy. The Standard Particle Swarm Optimization. The Ant Colony Optimization for Continous Domain.

Chapter 3 presents some clustering techniques that guides the construction of ROIs and later becomes the initial points for algorithms in each arm.

Chapter 4 first describes four basic affine transformation: translation, rotation, scaling and shearing. Then the projective transformation and homogeneous coordinate are presented.

Chapter 5 briefly describes some common multi-armed bandit algorithms, including ... Then we present our new bandit techniques. Tranditional bandit algorithms focus on minimizing regret, while our new bandit focus on the probability of getting a rank 1 result.

Chapter 6 gives details of our new algorithms. First, the framework and pseudo code are given. Then a detailed process of initialization is given in ...

Chapter 8 summarizes this thesis. The conclusion and contributions are also given.

Some further improvements and future works are also discussed at the end.

Chapter 2

Real-valued Optimization Algorithms

Overview of real-valued optimization

2.1 Covariance Matrix Adaptation Evolution Strategy

Describe history of *Evolutionary Strategies* (ES).

The simplest algorithm is (1+1)-ES.

Here we describe the (1+1)-ES with one-fifth success rule with independent restarts.

The pseudo code of (1+1)-ES is given in Algorithm 1.

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is an extended version fo CSA-ES with de-randomized adaptation of covariance matrix.

Describe the underlying covariance matrix model.

Describe how to update *mean*.

Describe how to update *covariance matrix*.

Describe *step-size* control.

2.2 Standard Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a swarm intelligence optimization algorithm. It was first proposed by J. Kennedy and R. C. Eberhart in 1995 [4] to simulate the foraging behavior of bird flocks. The swarm is composed of *particles* that move around in a given

Algorithm 1: (1+1)-ES with 1/5 success-rule

$\boldsymbol{X}_n$: solution of the n^{th} iteration, σ_n: step size of the n^{th} iteration,
$N(\mathbf{0}, \boldsymbol{I})$: multivariant normal distribution with mean vector $\mathbf{0}$
and identical covariance matrix $\boldsymbol{I}$.

input : f: evaluation function
output: X_{n+1}: best solution

Initialize $\boldsymbol{X}_0, \sigma_0$
while *termination criterion not met* **do**
  $\widetilde{\boldsymbol{X}}_n = \boldsymbol{X}_n + \sigma_n N(\mathbf{0}, \boldsymbol{I})$
  if $f(\widetilde{\boldsymbol{X}}_n) \leq f(\boldsymbol{X}_n)$ **then**
    $\boldsymbol{X}_{n+1} = \widetilde{\boldsymbol{X}}_n$
    $\sigma_{n+1} = 1.5\sigma_n$
  else
    $\boldsymbol{X}_{n+1} = \boldsymbol{X}_n$
    $\sigma_{n+1} = 1.5^{-1/4}\sigma_n$
return $\boldsymbol{X}_{n+1}$

multi-dimensional *search space* to find the best solution. Each particle updates its velocity according to its historical experience, as well as the information of the neighboring particles. The neighborhood of a particle is a set of information links defined by the *swarm topology*. PSO iteratively updates the swarm topology, the velocities, and the positions of each particle until the global optimum solution is found.

Throughout the years, numerous varients of PSO have been proposed to improve performance. As a result, a *standard* PSO, composed of clear principals, is needed as the baseline for comparison. Standard PSO (SPSO) provides a well defined version that follows the common principals of PSO design. It is intended to be a milestone with simple and clear implementation for future comparison. So far, there have been three successive versions of standard PSO: SPSO 2006, SPSO 2007, and SPSO 2011. The underlying principals of these three algorithms are generally the same as all PSO varients. The exact formula and implementation are slightly different due to latest theoretical progress. A detailed description of SPSO 2011 is given in the following paragrahs.

2.2.1 Initialization of the swarm

For a search space E with a given dimension D, E is confined by a set of minimum and maximum bounds in each dimension. The hyperparallelepid search space E can be formally defined as the Euclidean product of D real intervals [2].

$$E = \bigotimes_{d=1}^{D} [min_d, max_d]$$

For each position x within the multi-dimensional search space E, there exists a corresponding numerical value $f(x)$, i.e. *fitness*. A swarm is composed of particles, which explore different positions in the search space to find the best corresponding fitness value. At time t, each particle in the swarm possesses the following vectors with D coordinate:

- $x_i(t)$ is the **position** of the particle i at time t.
- $v_i(t)$ is the **velocity** of the particle i at time t.
- $p_i(t)$ is the **previous best position** the particle i had been to, at time t.
- $l_i(t)$ is best position of all the previous best positions in the **neighborhood** of particle i at time t.

Let $U(min_d, max_d)$ be a random number drawn from a uniform distribution within $[min_d, max_d]$, and $N_i(t)$ be a set of neighbours of particle i at time t defined by the swarm topology. In SPSO 2011 [2], each particle is initialized with a random position and velocity defined as following:

$$\begin{aligned}
x_i(0) &= U(min_d, max_d) \\
v_i(0) &= U(min_d - x_{i,d}(0), max_d - x_{i,d}(0)) \\
p_i(0) &= x_i(0) \\
l_i(0) &= argmin_{j \in N_i(0)}(f(p_j(0)))
\end{aligned}$$

The swarm size, denoted as S, differs in SPSO 2006 and SPSO 2011. In both SPSO

2006 and SPSO 2007, the initial number of particles S for dimension D is defined as:

$$S = 10 + \lfloor 2\sqrt{D} \rfloor,$$

However, emperically, the definition of swarm size in SPSO 2006 is far from optmial swarm size [2]. Therefore, in SPSO 2011, the swarm size is suggested as

$$S = 40,$$

yet it can also be defined by user [2].

2.2.2 Velocity update equations

Velocity update equations differs in different variations of PSO, since it is the core procedure of optimization that determines the performance of PSO. To prevent premature convergence [2], SPSO follows a random permutation order to update the positions of particles in the swarm. The position of particle i at time $t + 1$ depends on the previous position and the current velocity:

$$x_i(t + 1) = x_i(t) + v_i(t + 1).$$

The velocity of particle i at time $t + 1$ can be described as a combination of three vectors:

$$v_i(t + 1) = wv_i(t) + \alpha(p_i(t) - x_i(t)) + \beta(l_i(t) - x_i(t)),$$

where w mimics the momemtum of the particle, and α, β describes how successive actions are effected by the personal previous best position and the neighborhood previous best position.

In SPSO 2006 and SPSO 2007, the velocity update equation is applied dimension by

dimension as follows:

$$
\begin{aligned}
v_i(t+1) &= wv_i(t) + U(0,c)(p_i(t) - x_i(t)) + U(0,c)(l_i(t) - x_i(t)) \\
w &= \frac{1}{2\ln(2)} \simeq 0.721 \\
c &= \frac{1}{2} + \ln(2) \simeq 1.193
\end{aligned}
$$

Figure 2.1 visualize the three main vectors in a two dimensional problem. The first part, v_1, represents the influence of previous action on current velocity. w is the inertia weight that resembles the momentum, relating the next velocity to the latest velocity $v_i(t)$. It allows *exploration* by giving a tendency of expanding the search space. The second part, v_2 is a random vector drawn from the parallelepid $c(p_i(t) - x_i(t))$. c is the acceleration constant and the parallelepid $c(p_i(t) - x_i(t))$ represents the effect of personnal best memory $p_i(t)$ on current decision making. The third part, v_3 is a random vector drawn from the parallelepid $c(l_i(t) - x_i(t))$ The parallelepid $c(l_i(t) - x_i(t))$ mimics how the best social memory $l_i(t)$ contributes to current decision making. The last two parts allows *exploitation*, by attracting the particles near the possible optimum positions. Therefore, the new velocity depends on the latest action, personnal best memory, and social influence.

However, the velocity update equations of SPSO 2006 depends on the system of coordinates. The distribution of all possible next positions is more dense near the center of a D dimension rectangle. Moreover, the dimension-by-dimension updating method makes the algorithm dependent to system coordinates. It is extremely easy to find the optimum point lying on an axis, a diagonal or the center of the coordinate.

In SPSO 2011, the velocity update equations eliminates the coordinate dependency by creating a hypersphere according to x_i, p_i and l_i, shown in Figure 2.1. The hypersphere is defined as:

$$H_i(G_i, ||G_i - x_i||),$$

with center G_i and radius $||G_i - x_i||$. When the personal best $p_i(t)$ is not the neighborhood

previous best $l_i(t)$, the center G_i is defined as:

$$G_i = \frac{1}{3}(x_i + (x_i + c(p_i - x_i)) + (x_i + c(l_i - x_i))) = x_i + \frac{c}{3}(p_i + l_i - 2x_i)$$

If the personal best is the best in the neighborhood, i.e. $p_i(t) = l_i(t)$, the center G_i is defined as:

$$G_i = \frac{1}{2}(x_i + (x_i + c(p_i - x_i))) = x_i + \frac{c}{2}(p_i - x_i)$$

In both cases, the acceleration constant c is:

$$c = \frac{1}{2} + \ln(2) \simeq 1.193$$

As shown in Figure 2.1, a random sample $x_i^{'}$ is drawn from the hypershpere with uniform random direction and uniform radius.

$$r = U(0, ||G_i - x_i||)$$

Therefore, the velocity update equation and new position of SPSO 2011 is defined as:

$$v_i(t+1) = wv_i(t) + x_i^{'}(t) - x_i(t)$$
$$x_i(t+1) = x_i(t) + v_i(t+1) = wv_i(t) + x_i^{'}(t)$$

where the inertia weight is also:

$$w = \frac{1}{2\ln(2)} \simeq 0.721$$

As mentioned before, the search space is defined as a hyperparallelepid with confinement $[min_d, max_d]$ in each dimension d. Therefore, after calculating the new velocities and positions of particles, we have to go through confinement check to make sure all particles stay inside the search space. There are mutliple options to handle confinement in SPSO 2011. The following method described in [2] treat boundary as "wall", so that all

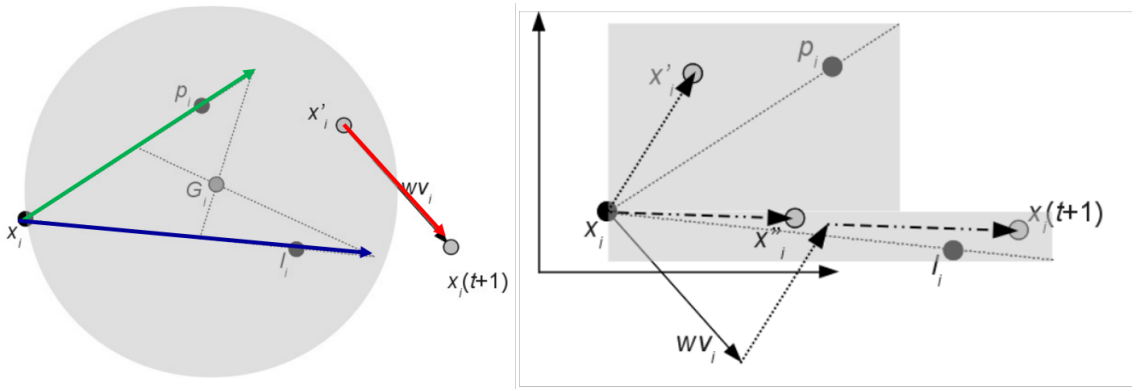

Figure 2.1: (a) SPSO 2011. (b) SPSO 2006.

particles bounce back with half of previous velocity after they reach the border. For the new position $x_{i,d}(t+1)$ of particle i in dimension d,

$$
\begin{aligned}
if(x_{i,d}(t+1) < min_d), x_{i,d}(t+1) &= min_d \\
v_{i,d}(t+1) &= -0.5 v_{i,d}(t+1) \\
if(x_{i,d}(t+1) > min_d), x_{i,d}(t+1) &= max_d \\
v_{i,d}(t+1) &= -0.5 v_{i,d}(t+1)
\end{aligned}
$$

2.2.3 The adaptive random topology

The swarm topology defines "who informs whom" to help distribute the information of optimum solution in the swarm. The random topology adopted in SPSO 2011 is updated under two circumstances [2]:

- at the very beginning
- after an iteration with no improvement of the best known fitness value

A good topology design should follow two rules [1]:

Describe random topology and when to update topology. The information links... The adaptive random topology described in [1] is formally equivalent to "Stocastic Star".

The pseudo code defined in [8]. The pseudo code is given in Algorithm 2.

Algorithm 2: Standard PSO 2011

$\boldsymbol{X}_n$: solution of the n^{th} iteration, σ_n: step size of the n^{th} iteration,
$N(\mathbf{0}, \boldsymbol{I})$: multivariant normal distribution with mean vector $\mathbf{0}$
and identical covariance matrix $\boldsymbol{I}$.

input : f: evaluation function
output: X_{n+1}: best solution

Initialize $\boldsymbol{X}_0, \sigma_0$
while *termination criterion not met* **do**
  $\widetilde{\boldsymbol{X}}_n = \boldsymbol{X}_n + \sigma_n N(\mathbf{0}, \boldsymbol{I})$
  if $f(\widetilde{\boldsymbol{X}}_n) \leq f(\boldsymbol{X}_n)$ **then**
    $\boldsymbol{X}_{n+1} = \widetilde{\boldsymbol{X}}_n$
    $\sigma_{n+1} = 1.5\sigma_n$
  else
    $\boldsymbol{X}_{n+1} = \boldsymbol{X}_n$
    $\sigma_{n+1} = 1.5^{-1/4}\sigma_n$
return $\boldsymbol{X}_{n+1}$

2.3 Ant Colony Optimization for Continuous Domain

Ant Colony optimization (ACO) is first proposed by Dorigo [3] to solve combinatorial optimization problems, including scheduling, routing, and timetabling. These problems aim to find optimal *combinations* or *permutations* of finit sets of available components. Inspired by the foraging behavior of natural ants, ACO mimics the pheromone deposition of ants along the trail to a food source. The deposited pheromone, which indicates the quantity and quality of the food, creates an indirect communication among ants and enables them to find the shortest paths. The pseudo code of ACO is given in Algorithm 3. Two major procedures: *solution construction* and *phermone update*, are detailed in the following paragraph.

Consider a search space $\boldsymbol{S}$ defined over a finit set of all possible *solution components*, denoted by $\boldsymbol{C}$. Each solution component, denoted by c_{ij}, is a decision variable X_i instantiated with value $v_i^j \in \boldsymbol{D}_i = \{v_i^1, ..., v_i^{|\boldsymbol{D}_i|}\}$. To construct a new solution, an artificial ants starts with an empty partial solution $s^p = \emptyset$. During each construction step, the partial solution s^p is extended with a feasible solution from the set $N(s^p) \in \boldsymbol{C} \setminus s^p$. The probabilistic pheromone model adopted for selecting a feasible solution from $N(s^p)$ can be defined as follows:

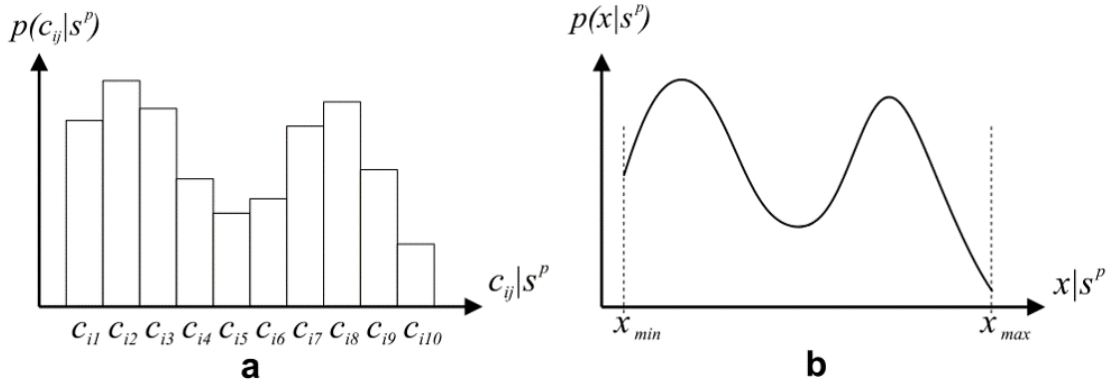

Figure 2.2: (a) Discrete probability distribution $p(c_{ij}|s^p)$. (b) Continuous probability density function $p(x|s^p)$

$$p(c_{ij}|s^p) = \frac{\tau_{ij}^{\alpha} \cdot \eta(c_{ij})^{\beta}}{\sum_{c_{i\ell} \in N(s^p)} \tau_{i\ell}^{\alpha} \cdot \eta(c_{i\ell})^{\beta}}, \forall c_{ij} \in N(s^p), \tag{2.1}$$

where τ_{ij} is the pheromone value associated with component c_{ij}, and $\eta(\cdot)$ is a weighting function. α and β are positive parameters which determine the relation between phermone and heuristic information.

The pheromone update

Over the years, multiple approaches of extending the ACO on continous domain have been given. One of the most successful version is ACO_R, proposed by Socha and Dorigo in 2008 [7]. It extends ACO to the continuous domain without making any major conceptual change to its structure. The fundamental idea underlying ACO_R is the shift from using a discrete probability distribution to using a continuous one, demonstrated in Figure 2.2.

A enhanced Gaussian kernel PDF as shown in Figure 2.3.

Algorithm 3: Ant Colony Optimization metaheuristic

while *termination criterion not met* **do**
    schedule activities
    solution contruction by ants();
    phermone update();
    daemon actions();

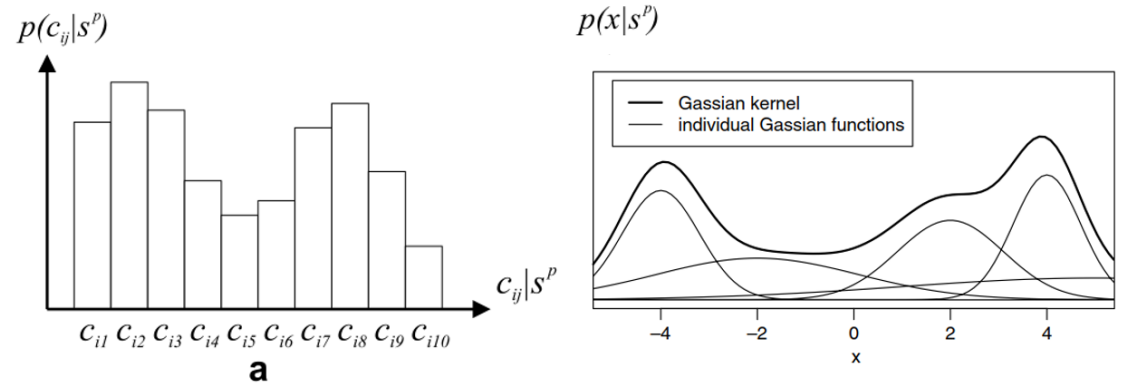

Figure 2.3: (a) Discrete probability distribution $p(c_{ij}|s^p)$. (b) Continuous probability density function $p(x|s^p)$

Chapter 3

Clustering Techniques

We are interested in solving real-valued multimodal problems composed of observable unimodals. Given the case, it is easier to solve an isolated unimodal within a subspace, than tackling the complete search space with the multimodal problem. Therefore, the first thing for solving a multimodal problem is to identify and isolate the potential uni-modals within the given search space.

We tried to isolate potential *fitness hills*, i.e. uni-modals, by clustering the initial samples points, and consider each cluster as a multi-dimension normal distribution. Different clustering techniques are often applied to identify different characteristics of clusters. Here we proposed a *hierarchical clustering* techniques to identify *fitness hills*, since we consider not only the density of the particles, but also the fitness values of different search points. Our basic assumption for the under lying uni-modal is a weighted normal distribution, since we need to take fitness into account instead of viewing each point with the same weight. We tend to focus on the particles with better fitness, than a dense cluster with less fitness. It is also easier to calculate the weighted mean vector and weighted covariance matrix in higher dimension.

Later, we applied the Minimum Description Length (MDL) to reduce the number of clusters. Although a complex Gaussian Mixture Modal is able to describe the sample distribution better, we believe that a more compact model, in terms of information entropy, is the better choice when multiple models can describe the same distribution. This also allows us to define a more stable subspace for further searching.

3.1 K-Means Clustering Techniques

In this section, we first describe the K-Means clustering techniques and its limitation for identifying the fitness hills. K-Means clustering aims to partition n observation data points into k clusters. There are two main procedures for K-Means clustering: the *assignment* step and the *update* step.

During the *assignment* step, an initial set of k means positions are given. Then, each data point is assigned to the cluster with the *nearest mean*. The assignment to the *nearest mean* can be formally described as creating clusters whose mean yields the least within-cluster sum of squares (WCSS), i.e. the sum of squared Euclidean distance. During the *update* step, the new centroids of each new clusters are calculated and assigned as the new means. This can be also be described as minimizing the within-cluster sum of squares (WCSS). The algorithm proceeds by alternating between these two steps until the means no longer change. Since there only exists a finite states of partitions, the algorithm must converge to a local optimum. However, different initial mean positions results in different clusters.

One of the main problem for using the K-Means clustering to identify *fitness hills* is that it considers only the spatial density and does not utilize the fitness value of each particle. This results in different clusters each centering at a point that does not necessarily possess the best fitness in the neighborhood. Therefore, the K-Means clustering often results in unstable clusters depending on initial conditions, and is highly sensitive to particle density. Another common phenomenon for being sensitive to spatial density is that the points on the border of clusters might belong to different clusters after each update. This makes the clusters unstable and costs unnecessary evaluations and computation for redefining the borders of ROI after each update.

The other problem for using the K-Means clustering is that one needs to decide the number of clusters. Determining the number of clusters is also a highly studied subjects.

Figure 3.1 shows how K-Means fails to identify a uni-modal and results in four clusters due to density. We would like our clustering method to be capable of identifying the uni-modal and be able to center around the particle with the best fitness.

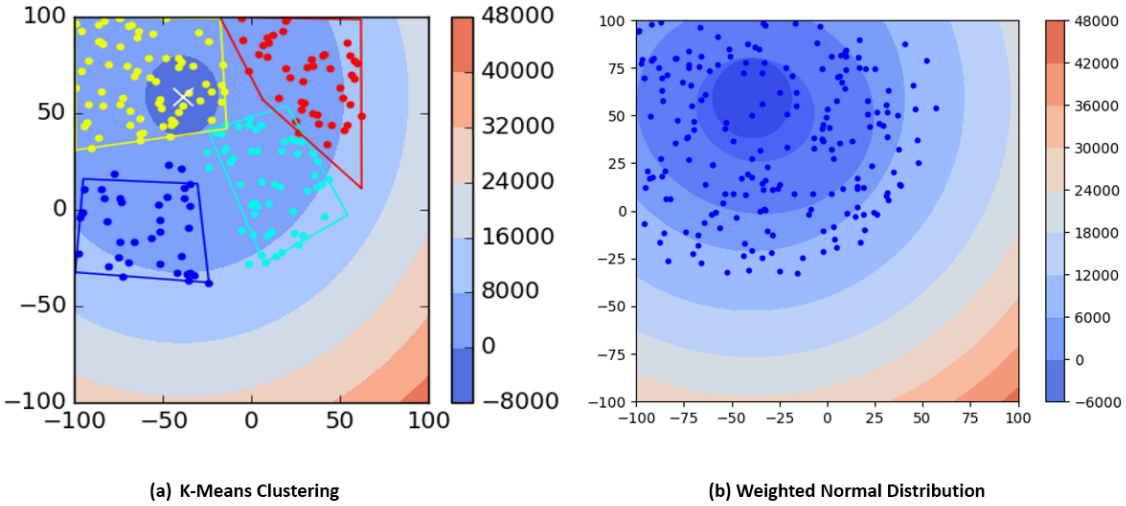

Figure 3.1: Comparing K-Means clustering with weighted Gaussian Distribution on CEC2005 F1 Problem

Describe how K-means clustering works and why it is popular Describe the limits for K-Means clustering, e.g. it cannot identify density nor unimodality.

3.2 Heirarchical Clustering

Describe the advantage of considering fitness instead of just density.

3.3 Determine number of clusters

3.3.1 Silhouette coefficient

3.3.2 Minimum description length and weighted multivariate normal distribution

MDL [6].

KMDL [5]

3.3.3 Silhouette coefficient

Describe how silhouette score decides number of clusters

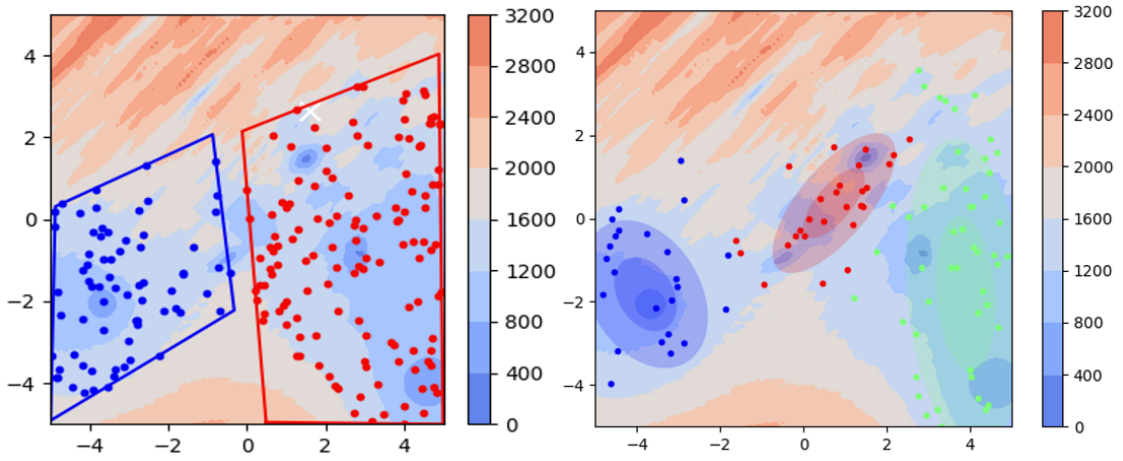

Figure 3.2: Comparing K-Means clustering with weighted Gaussian Distribution on CEC2005 F19 Problem

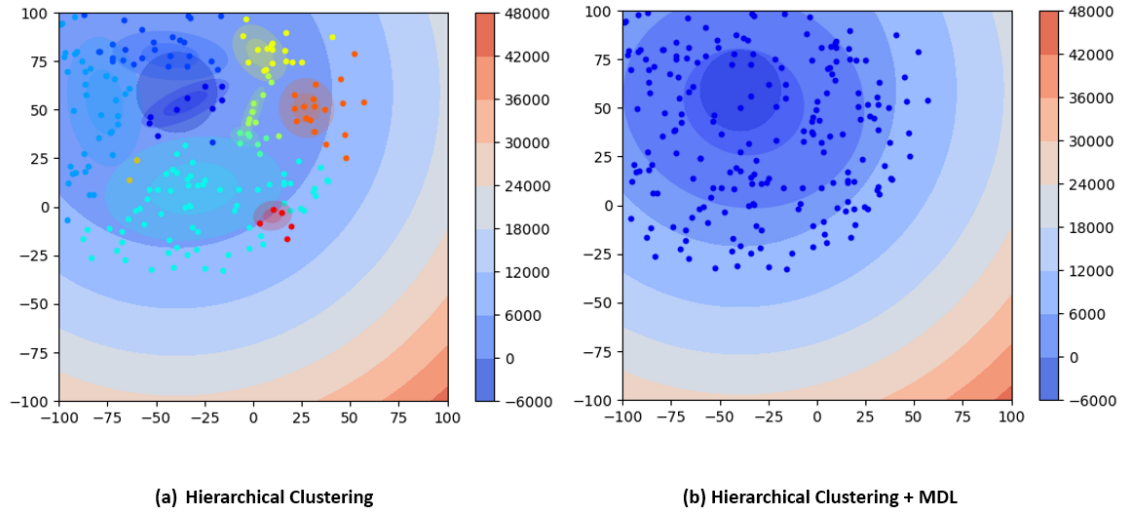

Figure 3.3: Before and after Minimal Description Length trimming on CEC2005 F1 Problem

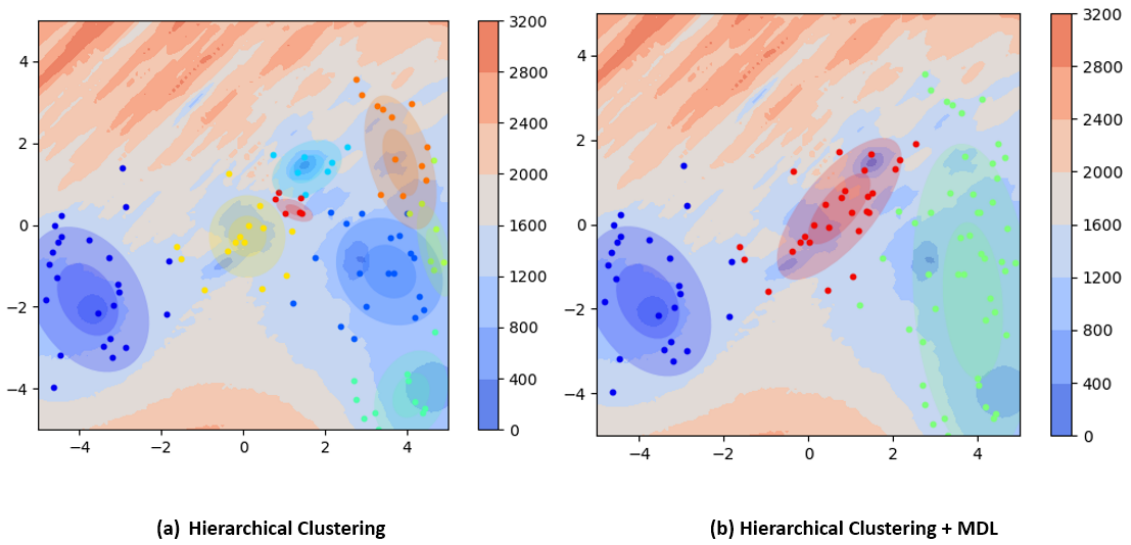

Figure 3.4: Before and after Minimal Description Length trimming on CEC2005 F19 Problem

3.3.4 Gap statistics

Describe how gap statistics estimates number of clusters.

3.3.5 Dip test

Describe how Dip-test checks unimodality Describe skynny-dip clustering

Chapter 4

Linear Projection

We would like to define a region of interest that contains only one uni-model for exploitation. Also, some algorithms, e.g. SPSO, requires a well-defined hypercube search space with fixed-value constraints in each dimension. Defining the projection onto subspace can be described as a model selection. In our case, this process does not require high accuracy, yet has a strict demand on time consumption. Combining the requirements, we use a linear projection matrix to project the original search space onto a subspace with feasible solutions bounded within $[0, 1]$ in each dimension.

One of the advantage for using projection matrix is that for a problem with ℓ variables, a linear projection matrix on homogeneous coordinate only requires $(\ell + 1)^2$ hyperparameters to be optimized. Therefore, less hyperparameters are assigned than directly define each hyperplane borders or vertices. This reduces the parameters that we need to optimize and results in less time consumption during model selection. We also use the (1+1)-ES, a fast and simple evolutionary strategy, to optimize the matrix. It allows us to rapidly approximate a high dimensional projection matrix within given number of iterations.

Furthermore, subspace projection also gives advantage for solving *inseparable problems*. For variabels that are not independent, projection allows the algorithm to solve an easier, rotated and sheared problem on subspace, as shown in Figure 4.1. The homogeneous projection matrix seperates the original ROI into less overlapped ROI, which reduces redundant search and sometimes enlarges the crutial regions.

In the following sections, we first describe the cannonical affine transformation. Then

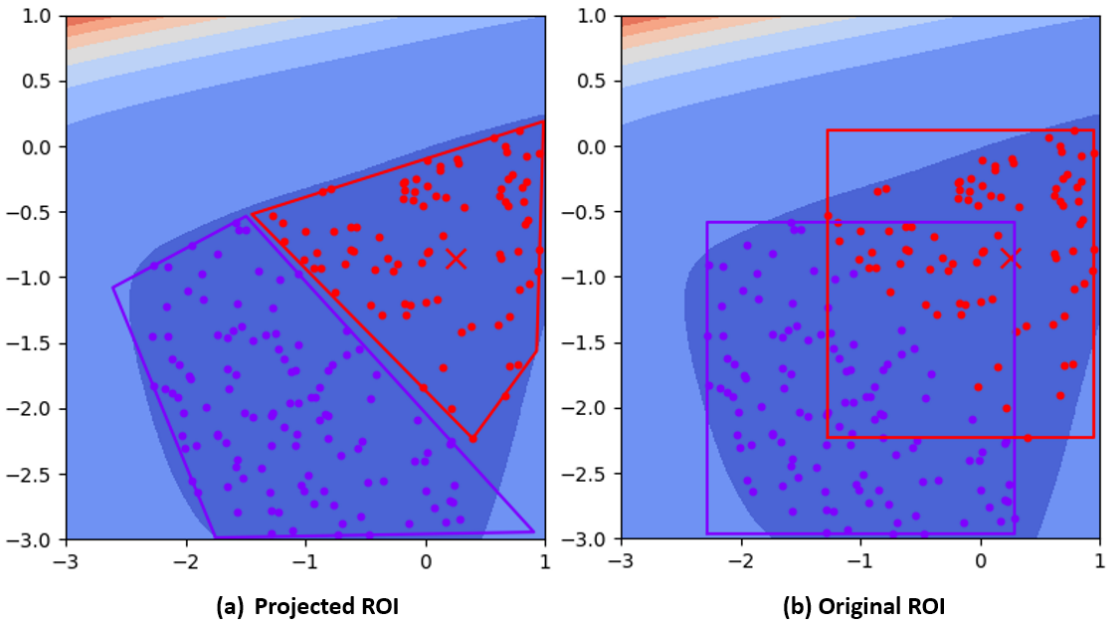

Figure 4.1: Projecting inseparable problems onto subspace.

we discuss how the projection matrix allows linear projection in a homogeneous coordinate. Finally, we give details of how we design the cost function and how we utilize the (1+1)-ES to optimize the projection matrix.

4.1 Affine Transformation

In geometry, an affine transformation preserves points, straight lines and planes. For two affine spaces A and B and any pair of points $P, Q \in A$, an affine transformation f determines a linear transformation φ that can formally defined as:

$$\overrightarrow{f(P)f(Q)} = \varphi(\overrightarrow{PQ})$$

In the following paragraph, we list some common affine transformation matrix in 2D.

1. Translation
2. Rotation
3. Scaling
4. Shear

4.1.1 Translation

4.1.2 Rotation

4.1.3 Scaling

4.1.4 Shear

4.1.5 Affine transformation matrix

4.2 Projection

4.2.1 Basic Projection

4.2.2 Homogeneous Coordinate

Augmented Matrix

4.2.3 Perspective Projection

4.3 Optimization for Projection Matrix

After identifying different clusters of samples, which represents a uni-modal, we would like to further define a reasonably good ROI. In order to obtain a nice ROI for a problem with D variables, we need to optimize the $(D+1)^2$ parameters in the projection matrix with a loss function. The loss function should cost little computational time and should suggest the right direction for optimization, in order to match the restrictions of a well-defined ROI. Also, the optimization only needs to define a reasonably well ROI for the alogirthms to search. In the later iterations, the matrix update procedure allows us to define a more accurate ROI with a better insight of the uni-modal subproblem. In our case, it is acceptable to sacrefice a bit of accuracy for computational time in hyperparameters tuning. We will describe the desgin of our loss function and the advantage of using a (1+1)-ES to optimize the matrix in the following sections.

4.3.1 Loss function

The goal of the ROI is to seperate the search space into non-overlapping subspaces, each containing an approximately normalized fitness hill. In our loss function, we considered the following features:

1. Distances to the boundary for the **points within the cluster** yet excluded in the ROI.
2. Distances to the boundary for the **random samples within other ROIs** yet included in the ROI.
3. Distances to the boundary for the **random samples** in the subspace that are outside of the original search space boundary.
4. Distance of the weighted **mean** to the center of subspace $[0.5]^D$
5. The Mean Absolute Error (MAE) for each element in the weighted **covariance matrix** to a scaled identicle matrix
6. The sum of **reconstruction** error for each particle

First, we would like to keep all the particles that belong to this cluster in the boundary $[0, 1]^D$. It can easily be done by projecting all points within the cluster onto the subspace, and check whether if any of the projected position is outside of the boundary $[0, 1]^D$.

Second, we would like to avoid ROIs from overlapping by adopting the Monte Carlo method for region collision detection. When optimizing one ROI, we randomly generate a given amount of samples within the subspace for all the other ROIs. Then, we use the inverse projection matrix of each other ROIs to project these sample points back to the original search space. After that, we use the projection matrix of the ROI that we are optimizing to project all the exterior samples onto the subspace. This way, we can easily discover which exterior samples are within the $[0, 1]^D$ boundary. Figure 4.2 shows a screen shot of the red ROI before and after optimizing the projection matrix of the red ROI. We randomly generates 200 samples within the green and blue ROI and tries to exclude them in the red ROI.

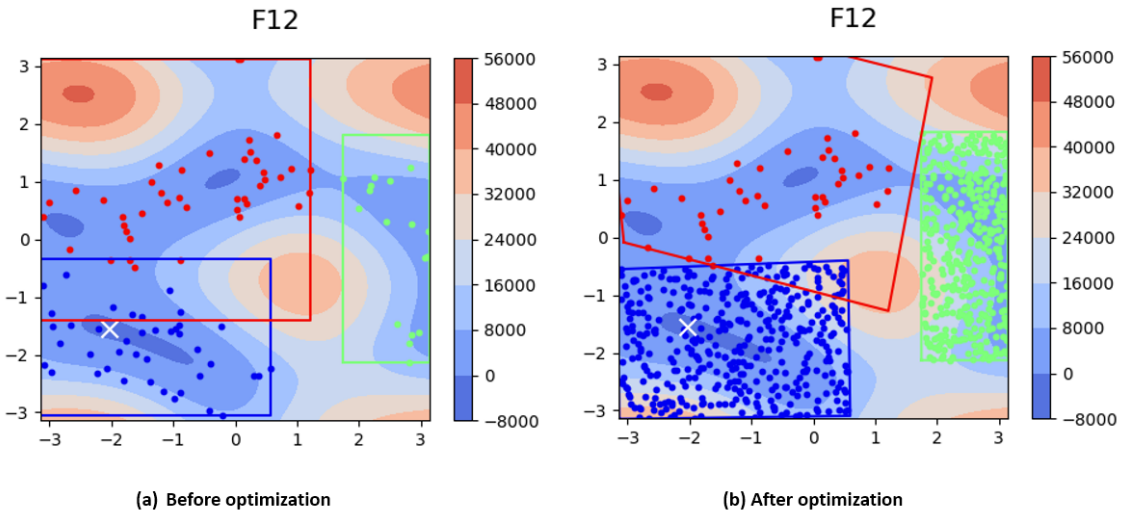

Figure 4.2: Optimization of projection matrix.

Third, we also use the Monte Carlo method described before to make sure that the ROI does not exceed the boundaries of the orignal subspace. We randomly generate samples from the subspace and inverse project it back to the search space to make sure that all points are within boundaries. If not, we would like to minimize the distance of sample points to the nearest border.

Forth, we would also like each underlying model for each subproblem to be a multivariate gaussian distribution with mean around the center $[0.5]^D$. If the algorithm searches near the boundary, it indicates that ROI should be updated since the optimum solution might be outside of the border. Placing the optimum solution at the center of the search space creates a more stable model for the algorithm to search. This allows a thorough search around the center, and decreases the frequency of ROI update. Therefore, we calculate the weighted mean of the points within the cluster on the subspace. Then we minimize the distance of weighted mean to the center.

Fifth, we would like the weighted covariance matrix to be approximately $0.2I$, where I is an identicle matrix. This enables the projection matrix to rotate, scale, and shear a fitness hill. The normalization preprocess is a common technique to make future optimization easier. Therefore, we calculate the MAE between the weighted covariance matrix and $0.2I$.

Finally, since the we project the particle positions in a homogeneous coordinate, the

correlations between reconstructed positions on the subspace might not be identical to the original ones. We would like to minimize the transformation error while keeping the ROI within the original boundaries. Therefore, for each possible matrix solutoin, we use it to project all points in the corresponding cluster onto a subsapce. Later, we use the inverse matrix to project all the points back to the original search space to get the reconstructed positions. Then, we calculate the MAE of all points in all dimensions between the original position and the reconstructed positions.

Given a possible solution of the projection matrix $X = [x_1, x_2, ..., x_{(d+1)^2}]$ for a problem with d variables. Let the corresponding projection matrix be $\mathbf{A}$ with shape $(d+1) \times (d+1)$. We define the linear projection mapping any position $\vec{x}$ to the subspace as T, then

$$T(\vec{x}) = \mathbf{A}\vec{x}$$

matrix A, we define the loss function f as:

$$f(X) = D_{include} + D_{exclude} + D_{boundary} + D_{center} + D_{covariance} + D_{reconstruction}$$

$$D_{include} =$$

$$D_{exclude} =$$

$$D_{boundary} =$$

$$D_{center} =$$

$$D_{covariance} =$$

$$D_{reconstruction} =$$

4.3.2 Optimization Algorithms

With the loss function defined in the previous section, we still need an optimization algorithm to optimize the projection matrix. The difficulty for *hyperparameters optimization* is to balance between speed and optimization ability. We need the hyperparameters to de-

fine a subspace that contains the possible optimum solution while costing few computation time.

We can simplify and stablelize the optimization process by giving a reasonable good initial solution. This allows the algorithm to start the search around a more prefered neighborhood. Here, the initial solution is a simple dot product of a translation matrix and a scaling matrix. The matrix defines a hypercube is bounded by the minimum and maximum value in each dimension of all positions in the cluster.

We first tried CMA-ES.

Later, we utilize the (1+1)-ES, described in Algorithm 1.

Chapter 5

Multi-armed Bandit Algorithms

After identifying the unimodals in Chapter 3 and defining a ROI for exploitation in Chapter 4, we still need to decide how to allocate our resources. Describe the exploration vs. exploitation dilemma. One should take different strategies according to evaluations left. Instead of letting the algorithms handling both eploration and exploitation, we propose a bandit technique to help manage exploitation.

Multi-armed Bandit Algorithms are suitable for this scenario. It learns model from outcomes and the actions do not change the state of the world.

Unlike canonical MAB algorithms that minimize regrets, we wish to maximize the probability of gaining the maximum rank.

5.1 The Multi-armed Bandit Problem

Multi-aremed Bandit (MAB) Problem describes an agent needs to decide in K arms to pull at time t and receives a reward.

Describe *policy* and *regret*.

The goal is to minimize regret.

5.2 Some common MAB Algorithm

5.2.1 UCB

5.2.2 POKER

The Price of Knowledge and Estimated Reward (POKER) strategy considers three ideas: pricing uncertainty, exploiting the lever distribution and taking into account the horizon

5.3 The New Bandit Technique

Chapter 6

The New Bandit Technique

In this chapter, we illustrate the proposed algorithm step-by-step.

Overview of real-valued optimization

6.1 Framework of the New Bandit Algorithm

The goal is to identify the ROI for exploitation and maintain exploration through resource allocation.

Each algorithm needs to be modified to satisfy the following conditions:

1. Update one individual at a time
2. The algorithm can be projected onto a subspace and continue iterating
3. Replace one individual with a given position and fitness

The pseudocode of our new algorithm is given in Algorithm **??**

6.2 Initialization and Unimodal Identification

For a problem with D dimension, initialize with $100D$ points.

Keep only top 50% of points for unimodal identification. Use clustering techniques mentioned in Chapter 3.

Iteratively add $10D$ points until estimated cluster number is identical.

Algorithm 4: Framework of the new Bandit Algorithm

$\boldsymbol{X}_n$: solution of the n^{th} iteration, σ_n: step size of the n^{th} iteration,
$N(\mathbf{0}, \boldsymbol{I})$: multivariant normal distribution with mean vector $\mathbf{0}$
and identical covariance matrix $\boldsymbol{I}$.

input : f: evaluation function
output: X_{n+1}: best solution

Initialize $\boldsymbol{X}_0, \sigma_0$
while *termination criterion not met* **do**
  $\widetilde{\boldsymbol{X}}_n = \boldsymbol{X}_n + \sigma_n N(\mathbf{0}, \boldsymbol{I})$
  if $f(\widetilde{\boldsymbol{X}}_n) \leq f(\boldsymbol{X}_n)$ **then**
    $\boldsymbol{X}_{n+1} = \widetilde{\boldsymbol{X}}_n$
    $\sigma_{n+1} = 1.5\sigma_n$
  else
    $\boldsymbol{X}_{n+1} = \boldsymbol{X}_n$
    $\sigma_{n+1} = 1.5^{-1/4}\sigma_n$
return $\boldsymbol{X}_{n+1}$

With a given cluster number k, do K-Means clustering.

6.3 Define Region of Interest

Each arm is composed of an algorithm and a projection matrix. Initial matrix is set as a tight hyperbox that contains all points in the given cluster. Optimize matrix one-by-one. Resize each cluster to match the required population for each algorithm. Start algorithm with given initial positions and fitnesses.

6.4 Remain Evaluations Allocation

Calcualte remain evaluations allocatoin and normalize the value to 0 1. Add newly calculated allocatoin to record. Choose arm the argmax allocation to pull, i.e. update one individual. Max arm allocation -1.

6.5 Recluster

Chapter 7

Experiments

Overview of real-valued optimization

7.1 Test Problems

7.1.1 CEC2005 25 benchmark problems

Describe termination criterion and evaluation method for CEC2005 Describe reason for 25 repeat runs. Describe how initial setting effect real-valued optimization, so that all algorithms should start with identicle initial status.

7.2 Experiment Settings

Describe the parameters setting for CMA-ES, SPSO and ACOR.

We use the *Covariance Matrix Adaptation Evolution Strategy for non-linear numerical optimization in Python* package [1] provided by Hensen, the author of CMA-ES. CMA-ES initial mean and std, population settings.

SPSO2011 parameters (c, w) settings, and population settings.

For ACOR, we set our parameters according to the original paper [7]. The parameters are shown in Table 7.2.

[1] https://pypi.python.org/pypi/cma

Parameter	Symbol	Value
No. of ants used in an iteration	m	2
Speed of convergence	ξ	0.85
Locality of the search process	q	10^{-4}
Archive size	k	50

Table 7.1: Summary of the parameters used by ACO_R

Describe our bandit parameters setting, including the initial population, maximum number of arms, and (1+1)-ES step size.

Chapter 8

Conclusion

Contribution: find potential region to search allocate resources

Weakness: Potentially more NFE on unimodals and moving regions. Requires a better clustering techniques to identify underlying unimodals.

Future work: Non-linear transformation

Attention plays an important role in human vision. For example, when we look at an image, our eye movements comprise a succession of *fixations* (repetitive positioning of eyes to parts of the image) and *saccades* (rapid eye jump). Those parts of the image that cause eye fixations and capture primary attention are called *regions of interest* (ROIs). Studies in visual attention and eye movement have shown that humans generally only attend to a few ROIs. Detecting these visually attentive regions in images is challenging but useful in many multimedia applications, such as automatic thumbnail cropping, object recognition, content-based image retrieval, adaptive image compression and automatic browsing in small-screen devices.

Bibliography

[1] M. Clerc. Back to random topology. *Relatório Técnico, mar*, 2007.

[2] M. Clerc. Standard particle swarm optimisation. 2012.

[3] M. Dorigo and G. Di Caro. Ant colony optimization: a new meta-heuristic. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 2, pages 1470–1477. IEEE, 1999.

[4] J. Kennedy and R. E. P. S. Optimization. Ieee int. 4, 1995.

[5] I. O. Kyrgyzov, O. O. Kyrgyzov, H. Maître, and M. Campedel. Kernel mdl to determine the number of clusters. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 203–217. Springer, 2007.

[6] J. Rissanen. Universal coding, information, prediction, and estimation. *IEEE Transactions on Information theory*, 30(4):629–636, 1984.

[7] K. Socha and M. Dorigo. Ant colony optimization for continuous domains. *European journal of operational research*, 185(3):1155–1173, 2008.

[8] M. Zambrano-Bigiarini, M. Clerc, and R. Rojas. Standard particle swarm optimisation 2011 at cec-2013: A baseline for future pso improvements. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 2337–2344. IEEE, 2013.