**應用多種差分演化演算法於參數單目標最佳化問題**

|  |
| --- |
| 許廷豪 |
| 國立中山大學 |
| [a28615781a@gmail.com](mailto:a28615781a@gmail.com) |

# 摘要

本報告探索了差分進化演算法（DE）、多種自適應差分進化演算法（SaDE、JADE、SHADE、L-SHADE、jSO），對於解決參數單目標最佳化問題之應用及效能，並以6種測試函數Ackley、Rastrigin、HappyCat、Rosenbrock、Zakharov、Michalewicz等來進行實驗比較。此外，本報告在這些演算法的基礎下，於計算過中加入了正交基底變換，進一步探究了如此改變對於尋找解的影響。

**關鍵詞：啟發式演算法、差分進化、正交基底。**

## 簡介

在現實生活中有許多最佳化問題牽涉到多筆參數，且這些參數常常牽一髮而動全身，找到一組使問題最小化的完美參數非常困難。例如在手機CPU的設計中，續航力和程式運行速度通常無法兼顧，若想要高續航力，CPU的頻率就不能設過高，否則續航力不佳，然而過低的頻率又會造成運算速度降低，進一步導致使用者體驗不足。這類問題常涉及多種因素的取捨，且沒有明確的演算法能夠找到最佳解。在此狀況下，人們常利用超啟發式演算法（Metaheuristic Algorithm）搭配特定的測試函數來解決相關問題。

超啟發式演算法不依賴問題的定義，轉而使用隨機探索和啟發性原則來探索解空間，具有泛化的解決能力。超啟發式演算法泛用的特性使之常被用於現實生活中的各種問題，如工業排程問題[1]、無線網路之節點部署[2]、金融投資的最佳化[3]等。也有許多種類的超啟發式演算法被提出，如模擬退火(Simulated Annealing)[4]、基因演算法（Genetic Algorithm）[5]、差分進化演算法（Differential Evolution）[6]、蟻群演算法（Ant Colony Optimization）[7]及粒子群最佳化（Particle Sarm Optimization）[8]等演算法。

其中，差分進化演算法為本報告的觀察重點。差分進化演算法有著強大的搜尋和收斂能力，且能被應用在實數域的解空間中，是解決參數單目標最佳化問題的最佳演算法之一。此外，在差分進化演算法在被提出之後，經過了許多學者的多次改良，能夠更加適應各類問題的本質，甚至不需調整超參數[9]。

本報告旨在探討多種差分進化演算法於參數單目標最佳化問題中的表現，並比較其在不同測試函數上的收斂能力與解品質。為了更深入理解這些演算法的行為與差異，實驗部分選擇了六個常見的測試函數——Ackley、Rastrigin、HappyCat、Rosenbrock、Zakharov 與 Michalewicz 函數作為評估依據，涵蓋多種函數特性如多峰性、非分離性與狹窄極小值等。

此外，為了進一步改善演算法的搜尋能力，本報告在實驗過程中引入了正交基底的變換機制，試圖讓演算法能夠用不同的角度尋找解空間，以跳脫解空間中各種的局部最小值陷阱。本報告將進一步與原演算法比較，評估正交基底變換對於解空間的探索是否有益。

透過本報告，期望能更清楚了解各類差分演化演算法在不同問題上的優劣與特性，以及正交基底變換對於演算法的改良是否有益。

## 相關研究

* 1. **實數域超啟發式演算法**

在現實生活所遇到的各種最佳化問題中，常常遇到解空間並非離散，而是連續的。例如手機設計中，CPU的電壓、大小、放置位置等參數皆非離散。這導致了許多超啟發式演算法（如模擬退火、基因演算法）無法使用。

在此情況下，有許多學者改進了離散版本的演算法，使其也能利用於實數域中。例如Herrera等人提出的實數編碼基因演算法（Real-Coded Genetic Algorithm），修改了基因演算法中的交配和突變過程，加入了線性交配（Linear Crossover）、高斯突變（Gaussian Mutation）等實數域才能利用的方法，令基因演算法有了在實數域上尋找解的能力[10]。而Storn和Price則利用了實數空間的幾何及連續性，並結合了基因演算法的選擇、交配、突變三大步驟，提出了差分進化演算法（Differential Evolution）。差分進化演算法使用實數向量表示解，在每個世代中利用各個體的向量差分進行下一個世代個體運算，使其能利用向量相關運算進行解空間的搜尋[6]。

實數最佳化問題由於解空間的性質，使其幾乎無法利用窮舉法等離散方法來求得最佳解，但也多了諸如線性、可微分等性質用於搜尋，也因此帶動了實數域超啟發式演算法的發展。

* 1. **自適應超啟發式演算法**

## 超啟發式演算法儘管具有強大的泛用能力，但是並非所有問題超啟發式演算法都能找到較佳解。這些問題常需要更多元的資訊才能讓演算法找到一個不錯的解，因此學者們提出了「自適應」策略。自適應超啟發式演算法能根據搜尋歷史、環境變化或當前表現來自動調整參數，使之更加契合問題本身，提升了探索效率並降低了落入區域最小值的機率。

## 由Kirkpatrick等人所提出的模擬退火[4]是最早具有自適應概念的演算法。模擬退火利用溫度控制解的接受程度，讓演算法能在搜尋初期接受較差的解，以跳出局部最小解，並在搜尋後期進一步收斂。對解的接受程度可被視為一種自適應的實踐，演算法利用了「時間」資訊來進一步的調整超參數──解的接受程度。而之後由Srinivas等人提出的自適應基因演算法（Adaptive Genetic Algorithm）正式提出了自適應的概念，自動的根據世代的表現修改交配率和突變率，讓演算法能更加適用各個問題。這種概念為超啟發式演算法帶來了巨大的突破，更進一步了的提高了各類演算法的表現和泛用程度。

## 關於自適應在差分演算法的應用上，Qin等人提出了自適應差分進化演算法，（Self-Adaptive Differential Evolution）紀錄並利用了上一個世代的經驗，自動調整交配率和差分距離，讓演算法視情況選擇探索解空間或進一步收斂[11]演算法不在受限於初始定義的超參數，而是能夠接受各個世代的回饋，進一步調整自己的尋找策略。自適應差分進化演算法為差分演算法帶來巨大的突破，也為後來的更進一步的研究奠定了基石。

* 1. **正交相關應用**

正交轉換改變了向量空間中點的「表達方式」，使演算法能夠用不同的角度的探索解空間。也因為正交轉換不改變長度和角度的相對比例，能夠在不失演算法的精度下進行不同視角的搜索，增加探索的多樣性以及收斂的穩定性。

在超發式演算法的研究中，Wang等人提出的正交實驗設計進化演算法（Orthogonal Design-Based Evolutionary Algorithm）利用了正交表（Orthogonal Array）系統性的從雙親的各種組合中產生新的子代[12]。透過正交實驗設計，該方法能夠在有限的試驗次數下保留最大的訊息量，使子代分布更具代表性，提升了搜尋效率與解的多樣性。

## 演算法介紹

本章節將依序介紹本報告中所實作的六種差分進化演算法，包含原本的差分進化，以及其他五種自適應差分進化演算法。每一個小節會介紹演算法流程和參數控制機制。

* 1. **差分進化演算法（DE）**

差分進化演算法[6]（Differential Evolution,下稱DE）是一種用來解決連續參數單目標最佳化問題的演算法。DE與基因演算法不同的是DE利用向量差分做為變異來源，再與原個體進行交配，而基因演算法是利用父代的資訊雜交再加上小部分變異。

DE的族群由實數向量組成，族群數量為，每一代會對個體進行突變操作。本實驗隨機採用以下變體規則：

* rand/1
* current-to-best/1

其中為變異向量，為該世代的第個體向量，為該世代中適應值最好的個體向量，則為該世代隨機三個不重複個體向量。每個向量皆有個元素，。

為差分距離。

計算出變異向量後會進行交配步驟，依下面公式計算出新的個體：

其中為變異率，為隨機值，，為一隨機索引值，，用以確保必不與上一世代個體相同。

接著是選擇步驟。DE採取貪婪策略，若新個體的適應值優於原個體，則新個體會加入下一個世代，否則會被拋棄，而加入新世代。

Initialize population of size ,,

Evaluate fitness of all individuals

while not termination\_condition do

for each individual in :

for each element in:

= mutate(,,)

= crossover(, , )

if fitness() < fitness():

=

end for

end while

return best\_fitness

**圖1. 差分進化演算法**

* 1. **自適應差分演算法（SaDE）**

自適應差分進化演算法[11]（Self-adaptive Differential Evolution,下稱SaDE）為一種基於DE的自適應演算法。SaDE在DE的基礎下加入了兩種機制：策略選擇和參數分布。

在策略選擇中，SaDE引入了兩種變異策略，且其中一種善於探索解空間，另一種善於收斂。且SaDE會根據兩種策略的產生更優個體成功率，自動計算下一個世代兩種策略的採用機率。本報告中使用的探索策略為rand/1，收斂策略為current-to-best/1。

在參數分布中，SaDE將兩個重要的超參數和常態分布化，使這些參數不再依賴於固定常數。利用分布能夠增加解空間探索能力，進而讓演算法更容易跳出局部最小值。此外，演算法會紀錄成功的值，並在下一個世代根據紀錄的平均值更新的分布，使其更加容易進行解空間探索或收斂。

Initialize population of size

Initialize parameter sets for normal distribution and

Evaluate fitness of all individuals

while not termination\_condition do

Sample and from distrubtion

for each individual in :

for each element in:

= select\_strategy(,,)

= crossover(, , )

if fitness() < fitness():

=

Record

end for

Update distrubtion based on record

end while

return best\_fitness

**圖2. SaDE**

* 1. **引入檔案庫之自適應差分演算法**

引入檔案庫之自適應差分進化演算法[13]（Adaptive DE with Optional External Archive,下稱JADE）為另一種基於DE的自適應演算法。JADE與SaDE相同，使用分布來為和採樣，不同的是，JADE使用高斯分布為採用，且和都有記錄，並以、，更新分布的平均值，其中、為和成功紀錄列表、此外，JADE引入了新的變異策略：curret-to-best，來同時兼顧探索和收斂。另外，被淘汰的個體會被加入到檔案庫中，以增加突變策略的變異性。

* current-to-pbest/1

其中為世代中前的優良各體，為該世代中的隨機個體，為該世代中或檔案庫中的隨機個體，。儘管JADE在突變過程中只用了此策略，仍然能夠與SaDE相同，同時兼顧探索和收斂。

Initialize population of size

Initialize archive , memory and

Evaluate fitness of all individuals

while not termination\_condition do

for each individual in :

Sample

Sample

for each element in:

= mutate(,,)

= crossover(, , )

if fitness() < fitness():

=

Record ,

end for

Update , based on record

Resize Archive to bounded size

end while

return best\_fitness

**圖3. JADE**

* 1. **基於成功歷史自適應差分演算法**

基於成功歷史自適應差分演算法[14]（Success-History Based Adaptive DE，下稱SHADE），是JADE的延伸，進一步將參數的記錄機制推廣，將算出的參數放入歷史參數陣列，並引入加權更新。

* 1. **使用族群控制之基於歷史自適應差分演算法**

使用族群控制之基於成功歷史自適應差分演算法[15]（SHADE Using Linear Population Size Reduction，下稱L-SHADE），是SHADE的延伸，引入了線性族群大小控制，前期大族群著重全域搜尋，而後期加快了演算法的收斂效率。從此演算法開始，DE系列中最重要的三個參數、和皆自適應化。

* 1. **增強控制歷史自適應差分演算法**

增強控制歷史自適應差分演算法[16]（Success-History Based Adaptive DE with Enhanced Control，下稱jSO），是L-SHADE的延伸，並加入了一些功能。

* 和的分段限制
* 利用jF評估階段調整變異強度
* 避免選擇自己為。

* 1. **正交基底變換**

本報告在上述4個自適應差分進化演算法中加入了正交基底變換機制。若已經過0.75\*總世代數且最佳解尚未更新過久，演算法會自動以歷史紀錄最佳的個體做為新的正交基底，並將整個搜尋空間及個體以新的正交基底轉換，以便演算法能得到新的視野，並找到更優解。

Initialize population of size

Initialize parameter sets for normal distribution and

Evaluate fitness of all individuals

while not termination\_condition do

Sample and from distrubtion

for each individual in :

for each element in:

= select\_strategy(,,)

= crossover(, , )

if fitness() < fitness():

=

Record

end for

Update distrubtion based on record

if generation == total\_generation\*0.75:

Change basis based on best individuals

end for

end while

return best\_fitness

**圖4. SaDE-ortho**

## 實驗結果

本章節首先介紹實驗中所使用的測試函數，再介紹各項演算法的的實驗參數，並對實驗結果並對此加以分析。

* 1. **測試函數**

本次實驗使用了6種函數，分別是Ackley、Rastrigin、HappyCat、Rosenbrock、Zakharov和Michalewicz。本小節對各函數進行介紹和分析。

* + 1. **Ackley**

Ackley函數具有多峰、對稱的特性，考驗演算法的全域探索能力。其全域最小值在原點，其值為0。在本實驗中，任一維度的元素。以下為Ackley函數的數學表達式。

* + 1. **Rastrigin**

Rastrigin函數是一個非線性非凸多鋒函數，在搜尋空間內有很多具有規律的極小值，非常容易落入極小值陷阱，要找到最小值非常困難。全域最小值落在原點，其值為0。在本實驗中，任一維度的元素。以下為Rastrigin函數的數學表達式。

* + 1. **HappyCat**

HappyCat函數是一個非線性函數，因其2維等高線圖看起來像一隻在微笑的貓因而得名。HappyCat的非線性特性再加上各個變數彼此關聯，使其無法單獨為特定維度的值進行最佳化，也導致收斂困難。全域最小值落在，其值為0。在本實驗中，任一維度的元素。以下為HappyCat函數的數學表達式。

* + 1. **Rosenbrock**

Rosenbrock函數是一個單鋒函數，在多維圖中有管狀構造，看起來像香蕉，因此又稱香蕉函數。Rosenbrock函數並沒有多個山谷，但最小值和極小值的變化很小，導致找到最小值非常困難，考驗演算法的收斂能力。全域最小值落在，其值為0。在本實驗中，任一維度的元素。以下為Rosenbrock函數的數學表達式。

* + 1. **Zakharov**

Zakharov函數是一個單鋒函數，沒有局部極小值。平方和四次方項使遠離原點的懲罰極高，但也使最小值所在的山谷極為平坦，非常考驗演算法的收斂能力。全域最小值落在原點，其值為0。在本實驗中，任一維度的元素。以下為Zakharov函數的數學表達式。

* + 1. **Michalewicz**

Michalewicz函數是一個多鋒函數，有大量的極小值，且極小值的數量隨著維度以指數成長，也因為引入了三角函數和高次方項，使其有極多不規則的波動全域最小值落在原點，非常考驗演算法的探索能力。全域最小值的位置視維度而定，值約為。在本實驗中，任一維度的元素。以下為Zakharov函數的數學表達式。

* 1. **實驗參數**

本小節將介紹本次實驗的五種演算法的實驗參數，並分析各個演算法的最終結果。

本次實驗分別測試了各個演算法在2維、10維及30維的表現，以上述6個函數測試。每個演算法在各測試函數有的測試次數，並會實驗30次，結果以30次的平均值和最小值呈現。

實驗參數中DE的差分距離設為，交配率

設為0.9，族群數量設為50。

SaDE的設為50；在每次個體變異時由平均值，變異量0.1常態分布採樣決定，其中的初始值設定為0.5，後演算法會自行決定的值；則由平均值0.5、變異量0.3的常態分布決定。

JADE、SHADE的設為50、L-SHADE的則是。三者的在每次個體變異時由平均值，變異量0.1常態分布採樣決定，其中的初始值設定為0.5，後演算法會自行決定的值；則由平均值、變異量0.1的高斯分布決定，其中的初始值設定為0.5。另外從前10%優秀的個體採樣得出。此外L-SHADE的族群數量不會低於4。

jSO的設為，其中為維度。在每次個體變異時由平均值，變異量0.1常態分布採樣決定，其中的初始值設定為0.8，後演算法會自行決定的值；則由平均值、變異量0.3的高斯分布決定，其中的初始值設定為0.3。另外從前40%優秀的個體採樣得出。此外jSO的族群數量不會低於4。

* 1. **實驗結果和分析**

實驗結果表見最後三頁。能夠發現自適應演算法的表現均比原本的DE好，甚至在30維的高維度下都能找到全域最小值，顯示出自適應的泛用性和強大性能。正交基底轉換機制則沒有一個穩定結果。如JADE和LSHADE在Rosenbrock的測試中，正交基底轉換過度擾亂了搜尋方向，導致演算法無法穩定收斂至0，甚至連局部最小值都無法收斂，表現明顯劣於不加入正交基底轉換的演算法。然而對於SaDE來說，加入正交基底轉換可以幫助收斂，如在30維的Rastrigin測試中，正交基底轉換能夠幫助SaDE每次皆穩定的找到最小值。總而言之，正交基底轉換是否能夠增加效能需要看演算法和測試函數而定

## 結論

本研究探究了三十年來差分進化演算法的發展，並利用了正交基底轉換來進一步的改善演算法。

**參考文獻**

1. Jain, A. S., & Meeran, S. (1999). A state-of-the-art review of job-shop scheduling techniques. *European Journal of Operational Research, 113(2), 390–434.*
2. Ghosh, A., & Das, S. (2008). Coverage and connectivity issues in wireless sensor networks: A survey. *Pervasive and Mobile Computing, 4(3), 303–334.*
3. Anagnostopoulos, K. P., & Mamanis, G. (2011). The mean–variance cardinality constrained portfolio optimization problem: An experimental evaluation of five multiobjective evolutionary algorithms. *Expert Systems with Applications, 38(11), 14208–14217.*
4. Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220(4598), 671–680.
5. Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.
6. Storn, R., & Price, K. (1997). Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4), 341–359.
7. Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1), 29–41.
8. Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. In *Proceedings of ICNN’95 - International Conference on Neural Networks* (Vol. 4, pp. 1942–1948). IEEE.
9. Mohamad Faiz Ahmad, Nor Ashidi Mat Isa, Wei Hong Lim, Koon Meng Ang,Differential evolution: A recent review based on state-of-the-art works,Alexandria Engineering Journal,Volume 61, Issue 5,2022,Pages 3831-3872,ISSN 1110-0168
10. Herrera, F., Lozano, M., & Verdegay, J. L. (1998). **Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis**. *Artificial Intelligence Review*, 12(4), 265–319.
11. Qin, A. K., & Suganthan, P. N. (2005).  
    Self-adaptive differential evolution algorithm for numerical optimization.  
    *Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC)*, 1785–1791.
12. Wang, Y., Liu, H., Cai, Z., & Zhou, Y. (2007). An orthogonal design based constrained evolutionary optimization algorithm. *Engineering Optimization*, *39*(6), 715–736. <https://doi.org/10.1080/03052150701280541>
13. Zhang, J., & Sanderson, A. C. (2009).  
    JADE: Adaptive differential evolution with optional external archive.  
    *IEEE Transactions on Evolutionary Computation*, 13(5), 945–958.
14. Tanabe, R., & Fukunaga, A. (2013).  
    Success-history based parameter adaptation for differential evolution.  
    *IEEE Congress on Evolutionary Computation (CEC)*, 71–78.
15. Tanabe, R., & Fukunaga, A. (2014).  
    Improving the search performance of SHADE using linear population size reduction.  
    *IEEE Congress on Evolutionary Computation (CEC)*, 1658–1665.
16. Brest, J., Maučec, M. S., & Bošković, B. (2017).  
    Single objective real-parameter optimization: Algorithm jSO.  
    *Proceedings of the 2017 IEEE Congress on Evolutionary Computation (CEC)*, 1311–1318.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 演算法 | Ackley | Rastrigin | HappyCat | Rosenbrock | Zakharov | Michalewicz |
| DE | 4.4409e-16 | 0 | 2.1181e-05 | 0 | 9.599e-102 | -1.8013 |
| SADE | 4.4409e-16 | 0 | 2.1440e-05 | 0 | 9.375e-90 | -1.8013 |
| SADE\_O | 4.4409e-16 | 0 | 9.5457e-06 | 0 | 3.289e-90 | -1.8013 |
| JADE | 4.4409e-16 | 0 | 4.1180e-03 | 0 | 9.746e-73 | -1.8013 |
| JADE\_O | 4.4409e-16 | 0 | 5.6752e-03 | 0 | 2.392e-74 | -1.8013 |
| SHADE | 4.4409e-16 | 0 | 2.9052e-04 | 0 | 7.203e-71 | -1.8013 |
| SHADE\_O | 4.4409e-16 | 0 | 9.4926e-04 | 0 | 2.804e-68 | -1.8013 |
| LSHADE | 4.4409e-16 | 0 | 6.7953e-07 | 0 | 4.295e-197 | -1.8013 |
| LSHADE\_O | 4.4409e-16 | 0 | 3.6910e-05 | 0 | 5.319e-197 | -1.8013 |
| JSO | 4.4409e-16 | 0 | 3.2413e-06 | 0 | 2.592e-262 | -1.8013 |
| JSO\_O | 4.4409e-16 | 0 | 3.9232e-04 | 0 | 7.301e-269 | -1.8013 |

表1：實驗結果（2維、最小值）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 演算法 | Ackley | Rastrigin | HappyCat | Rosenbrock | Zakharov | Michalewicz |
| DE | 4.4409e-16 | 3.3165e-02 | 4.2076e-03 | 5.9814e-10 | 3.8074e-97 | -1.8005 |
| SADE | 4.4409e-16 | 0 | 8.8798e-03 | 0 | 4.3513e-74 | -1.8013 |
| SADE\_O | 4.4409e-16 | 0 | 8.8419e-03 | 0 | 2.2267e-76 | -1.8013 |
| JADE | 4.4409e-16 | 0 | 1.3797e-02 | 0 | 4.3490e-68 | -1.8013 |
| JADE\_O | 4.4409e-16 | 0 | 1.2833e-02 | 8.1762e-32 | 1.7312e-69 | -1.8013 |
| SHADE | 4.4409e-16 | 0 | 6.8820e-03 | 0 | 1.4116e-62 | -1.8013 |
| SHADE\_O | 4.4409e-16 | 0 | 6.4095e-03 | 0 | 9.8897e-63 | -1.8013 |
| LSHADE | 4.4409e-16 | 0 | 3.7855e-03 | 0 | 1.4281e-176 | -1.8013 |
| LSHADE\_O | 4.4409e-16 | 0 | 3.1907e-03 | 0 | 4.7173e-182 | -1.8013 |
| JSO | 4.4409e-16 | 0 | 6.7159e-03 | 0 | 4.1934e-242 | -1.8013 |
| JSO\_O | 4.4409e-16 | 0 | 8.6492e-03 | 0 | 1.2813e-237 | -1.8013 |

表2：實驗結果（2維、平均值）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 演算法 | Ackley | Rastrigin | HappyCat | Rosenbrock | Zakharov | Michalewicz |
| DE | 4.4409e-16 | 1.0990e+00 | 4.1491e-02 | 5.3544e+00 | 2.5550e-01 | -9.3510 |
| SADE | 3.9968e-15 | 0 | 2.8995e-02 | 0 | 3.7472e-138 | -9.6601 |
| SADE\_O | 4.4409e-16 | 0 | 4.6415e-02 | 0 | 1.4289e-130 | -9.6601 |
| JADE | 4.4409e-16 | 0 | 3.5281e-02 | 3.6297e-20 | 8.4531e-58 | -9.6601 |
| JADE\_O | 4.4409e-16 | 0 | 2.3923e-02 | 1.1688e-17 | 1.6839e-64 | -9.6601 |
| SHADE | 4.4409e-16 | 0 | 3.0568e-02 | 0 | 5.5102e-154 | -9.6601 |
| SHADE\_O | 4.4409e-16 | 0 | 4.2606e-02 | 0 | 3.0863e-156 | -9.6601 |
| LSHADE | 3.9968e-15 | 0 | 4.1997e-02 | 0 | 1.1000e-94 | -9.6601 |
| LSHADE\_O | 4.4409e-16 | 0 | 2.1129e-02 | 2.8991e-29 | 5.5437e-98 | -9.6601 |
| JSO | 4.4409e-16 | 0 | 3.6184e-02 | 0 | 1.1244e-78 | -9.6601 |
| JSO\_O | 4.4409e-16 | 0 | 2.5102e-02 | 0 | 2.7849e-80 | -9.6601 |

表3：實驗結果（10維、最小值）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 演算法 | Ackley | Rastrigin | HappyCat | Rosenbrock | Zakharov | Michalewicz |
| DE | 3.8784e-15 | 5.1201e+00 | 8.8952e-02 | 5.5926e+01 | 2.2104e+00 | -8.7760 |
| SADE | 3.9968e-15 | 0 | 7.3892e-02 | 3.9866e-01 | 4.6825e-121 | -9.6586 |
| SADE\_O | 3.8784e-15 | 0 | 9.8301e-02 | 9.9059e-07 | 1.1368e-114 | -9.6601 |
| JADE | 3.5231e-15 | 0 | 6.2265e-02 | 7.8715e-16 | 3.5576e-48 | -9.6601 |
| JADE\_O | 3.6415e-15 | 0 | 6.0615e-02 | 2.1385e-14 | 7.4012e-51 | -9.6546 |
| SHADE | 3.7600e-15 | 0 | 1.0187e-01 | 0 | 9.1222e-124 | -9.6589 |
| SHADE\_O | 3.8784e-15 | 0 | 8.4861e-02 | 0 | 6.1012e-123 | -9.6600 |
| LSHADE | 3.9968e-15 | 0 | 8.0644e-02 | 0 | 9.0283e-89 | -9.6601 |
| LSHADE\_O | 3.7600e-15 | 0 | 5.7520e-02 | 8.7623e-28 | 3.1572e-90 | -9.6601 |
| JSO | 3.7600e-15 | 0 | 7.7162e-02 | 0 | 4.4551e-71 | -9.6601 |
| JSO\_O | 3.2863e-15 | 0 | 5.6561e-02 | 0 | 4.5922e-71 | -9.6601 |

表4：實驗結果（10維、平均值）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 演算法 | Ackley | Rastrigin | HappyCat | Rosenbrock | Zakharov | Michalewicz |
| DE | 7.5495e-15 | 2.2974e+01 | 5.1048e-01 | 1.3598e+03 | 1.4733e+01 | -26.1695 |
| SADE | 7.5495e-15 | 0 | 1.5548e-01 | 2.3814e-26 | 8.0082e-40 | -29.6309 |
| SADE\_O | 3.9968e-15 | 0 | 2.2800e-01 | 3.3361e-27 | 4.3819e-38 | -29.5972 |
| JADE | 3.9968e-15 | 0 | 1.4624e-01 | 2.4685e-22 | 1.2362e-30 | -29.5410 |
| JADE\_O | 3.9968e-15 | 0 | 1.3379e-01 | 2.6041e-17 | 1.6096e-31 | -29.1490 |
| SHADE | 7.5495e-15 | 0 | 1.3636e-01 | 0 | 8.2500e-62 | -29.6309 |
| SHADE\_O | 7.5495e-15 | 0 | 1.7564e-01 | 0 | 6.0872e-63 | -29.6309 |
| LSHADE | 3.9968e-15 | 0 | 1.1612e-01 | 0 | 7.0989e-47 | -29.6309 |
| LSHADE\_O | 3.9968e-15 | 5.5907e-04 | 9.7544e-02 | 1.6500e-17 | 1.7534e-51 | -27.7421 |
| JSO | 3.9968e-15 | 0 | 1.3400e-01 | 1.3849e-20 | 1.0819e-35 | -29.4745 |
| JSO\_O | 3.9968e-15 | 0 | 1.0887e-01 | 4.1290e-21 | 1.5171e-36 | -29.4997 |

表5：實驗結果（30維、最小值）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 演算法 | Ackley | Rastrigin | HappyCat | Rosenbrock | Zakharov | Michalewicz |
| DE | 6.9310e-01 | 4.5648e+01 | 2.2875e+00 | 4.4270e+03 | 4.3755e+01 | -22.8867 |
| SADE | 6.2853e-01 | 3.9790e-14 | 3.1581e-01 | 1.8091e+00 | 1.2993e-33 | -29.6220 |
| SADE\_O | 1.0557e-01 | 0 | 3.3963e-01 | 7.9732e-01 | 2.0369e-30 | -29.4405 |
| JADE | 4.1152e-15 | 4.3115e-01 | 1.9299e-01 | 5.3155e-01 | 2.7466e-25 | -29.3457 |
| JADE\_O | 4.2336e-15 | 1.1077e+00 | 1.9947e-01 | 7.9732e-01 | 1.6455e-24 | -28.7325 |
| SHADE | 3.1044e-02 | 0 | 2.7583e-01 | 5.3155e-01 | 3.8353e-56 | -29.6275 |
| SHADE\_O | 3.1044e-02 | 0 | 2.6661e-01 | 9.3021e-01 | 3.6485e-57 | -29.6289 |
| LSHADE | 3.9968e-15 | 0 | 1.6975e-01 | 1.1997e-28 | 9.4465e-33 | -29.6309 |
| LSHADE\_O | 3.9968e-15 | 5.9428e-01 | 1.4271e-01 | 2.6615e-11 | 7.2198e-45 | -27.1653 |
| JSO | 3.9968e-15 | 5.6843e-15 | 1.7623e-01 | 5.6999e-18 | 9.1690e-32 | -29.3425 |
| JSO\_O | 3.9968e-15 | 7.5791e-15 | 1.7204e-01 | 1.5288e-17 | 3.5075e-32 | -29.3604 |

表6：實驗結果（30維、平均值）