# Bare Bones Particle Swarms with Jumps

Mohammad Majid al-Rifaie[1] and Tim Blackwell[2]

[1] Goldsmiths, University of London, New Cross, London SE14 6NW,
`m.majid@gold.ac.uk`
[2] Goldsmiths, University of London, New Cross, London SE14 6NW,
`tim.blackwell@gold.ac.uk`

**Abstract.** Bare Bones PSO was proposed by Kennedy as a model of PSO dynamics. Dependence on velocity is replaced by sampling from a Gaussian distribution. Although Kennedy's original formulation is not competitive to standard PSO, the addition of a component-wise jumping mechanism, and a tuning of the standard deviation, can produce a comparable optimisation algorithm. This algorithm, Bare Bones with Jumps, exists in a variety of formulations. Two particular models are empirically examined in this paper and comparisons are made to canonical PSO and standard Bare Bones.

**Keywords:** Particle swarm optimisation, Bare Bones swarms, random restart, global optimisation

## 1 Introduction

There has been many attempts to understand the behaviour of the swarms in Particle Swarm Optimisation algorithm (PSO). This proved to be difficult due the presence of many moving parts (e.g. the effects of various parameters on the trajectory of the particle, particles' oscillation around constantly changing centres, the effects of swarm topology on its performance, etc.). A number of theoretical studies have tried to understand the dynamics of PSO, mainly concentrating on particle trajectories, swarm equilibria and formal convergence to local optima proofs [1–3]. In 2003, in one such attempt, Kennedy [4] proposed a minimised version of PSO – Bare Bones (BB) swarm optimisation – where the velocity update is eliminated. In this paper, after briefly describing BB, the Bare Bones with Jumps (BBJ1) algorithm [5] is presented alongside a second model, BBJ2. The performance of the newly introduced algorithms are compared against a standard PSO (which is taken here to be the Clerc and Kennedy (CK) [1] formulation), as well as Bare Bones (BB) swarm optimisation.

## 2 Bare Bones Swarm

It is known that particles converge to a weighted average between their personal best and neighbourhood best positions [6], but in order to understand the behaviour of particles and identify the similarity it has with other stochastic population-based optimiser, Kennedy [4] proposed a modified algorithm without the velocity formula in the update equation. The standard Bare Bones swarm (BB) has the following update formula:

$$x_{id} = g + \sigma_{id} N(0,1) \tag{1}$$
$$g = \frac{1}{2}(g_{id} + p_{id}) \tag{2}$$
$$\sigma_{id} = |g_{id} - p_{id}| \tag{3}$$

where $N(0,1)$ is the Gaussian distribution between 0 and 1; $g_i$ is the best informer in the neighbourhood of particle $i$ and $p_{id}$ is the personal best position of particle $i$ in dimension $d$.

In the next section, two new variants of this minimised algorithm are presented. The main differences are: a component-wise jumping method, and the presence of an implicit scale parameter that multiplies the standard deviation of the sampling distribution.

## 3 Bare Bones with Jumps

Bare Bones swarm can be generalised [5] so that the search focus $g$ (centre of the search volume at stagnation) and the search spread $\sigma$ can each be chosen from local or global neighbourhoods. This idea is embodied in the following rules:

$$g_i = BEST(p_i \in N_i) \tag{4}$$
$$\delta_{id} = |p_{i-1\,d} - p_{i+1\,d}| \quad \text{–local neighbouthood} \tag{5}$$
$$\delta_{id} = |g_{id} - p_{id}| \quad \text{–global neighbourhood} \tag{6}$$
$$x_{id} = g_{id} + \alpha \delta_{id} N(0,1) \tag{7}$$

where $\alpha$ is an arbitrary number and $N_i$ denotes the search neighbourhood of particle $i$. $N_i$, the $\mu$-neighbourhood can be global, or any local structure. The separation factor $\delta_i$ which controls search concentration, can be taken from a local or a global informer neighbourhood (the $\sigma$-neighbourhood). Theoretically, it is shown that for the sphere function, there is a critical value, $\alpha_c = 0.65$, such that, for $\alpha > \alpha_c$ the swarm resists collapse. Fastest convergence occurs at the critical value, but larger values promote exploration [5]. The Bare Bones with Jumps algorithm, Algorithm 1, includes a probabilistic jumping mechanism: a particle may jump uniformly in any dimension with probability $p_J$. This can be viewed as a partial re-initialisation (since in general not every component undergoes a jump) or, alternatively, as a tail broadening mechanism, allowing further search in areas where the Gaussian distribution tails are thin.

The investigations reported in [5] propose that a small jump probability $p_J = 0.01$ enhances performance over standard test set of 30D problems. This paper proposes a second Bare Bones with Jumps algorithm, model 2 (BBJ2), with an altered search spread component, and a smaller jump probability ($p_J = 0.001$):

$$x_{id} = g_i + \alpha \delta_{id} N(0,1) \tag{8}$$
$$\delta_{id} = |g_i - x_{id}| \tag{9}$$

**Algorithm 1** Bare Bones with Jumps Models 1 and 2

```
r ~ $U(0,1)$
if ( $r < p_J$ )
    $x_{id} = U(-X_d, X_d)$
else
    $x_{id} = g_i + \alpha\delta_{id}N(0,1)$
```

This algorithm utilises the difference between the neighbourhood best with the current position (in $|g_i - x_{id}|$, Equation 9) rather than the difference between either the left and right neighbours' bests (in local neighbourhood; see Equation 5) or the particle's personal best and the neighbourhood best (in global neighbourhood; see Equation 7). The reason behind proposing this alternation is to increase the influence of the current positions of the particles in the update equation on the assumption that this might offer a wider search capability.

In the next section, a set of experiments is designed to compare the performance of the algorithms referred to in this paper followed by some statistical analysis.

## 4 Experiments

The aim of this set of experiments is to compare the performance of the new BBJ variant, BBJ2, to BBJ1 and Bare Bones swarm (BB) and standard PSO (CK) [1]. The effect of the jumping mechanism is isolated by running experiments on BBJ2 without jumps (BBNJ), which is simply accomplished by setting $p_J$ to zero. In order to determine the quality of each algorithm, three performance measures are used (accuracy, efficiency and reliability which are presented next, in section 4.1).

### 4.1 Performance Measures

Three different performance measures [7] are used in the experiments conducted in this paper. These performance measures are accuracy, reliability and efficiency.

Accuracy of the swarms is defined by the quality of the best position in terms of its closeness to the optimum position. If knowledge about the optimum position is known *a priori* (which is the case here), the following would define the accuracy:

$$\text{Accuracy} = \left| f\left(p_g^t\right) - f\left(x_{opt}\right) \right| \tag{10}$$

where $p_g^t$ is the best position at time $t$ and $x_{opt}$ is the position of the known optimum solution.

If no information exists about the optimum solution, the fitness of the best position will be the accuracy of the swarm.

Another measure used is reliability which is the percentage of trials where swarms converge with a specified accuracy; this is defined by:

$$\text{Reliability} = \frac{n'}{n} \times 100 \tag{11}$$

where $n$ is the total number of trials in the experiment and $n^{'}$ is the number of successful trials.

Finally, efficiency is the number of iterations or objective function evaluations needed to converge with a specified accuracy (i.e. $10^{-8}$):

$$\text{Efficiency} = \frac{1}{n}\sum_{i=0}^{n} FEs \tag{12}$$

where $n$ is the total number of trials and *FEs* is the number of function evaluations before convergence.

## 4.2 Experiment Setup

The algorithms used are tested over a number of benchmarking functions from Jones et al. [8] and De Jong [9] test suite, preserving different dimensionality and modality (see Tables I and II in [10]). The first two functions (Sphere/Parabola and Schwefel 1.2) have a single minimum and are unimodal functions; Generalised Rosenbrock for dimension $D$, where $D > 3$, is multimodal; Generalised Schwefel 2.6, Generalized Rastrigin, Ackley, Generalized Griewank, Penalised Function P8 and Penalised Function P16 are complex high-dimensional multimodal problems with many local minima and a single global optimum; Six-hump Camel-back, Goldstein-Price, Shekel 5, 7 and 10 are lower-dimensional multimodal problems with fewer local minima. Goldstein-Price, Shekel 5, 7 and 10 have one global optimum and Six-hump Camel-back has two global optima symmetric about the origin. In order not to initialise the particles on or near a region in the search space known to have the global optimum, *region scaling* technique is used [11], which makes sure particles are initialised at a corner of the search space where there are no optimal solutions. The experiments are conducted with a population of 50 particles in global and local neighbourhoods independently. However, the halting criterion for this experiment is either to reach the optima (with function errors less than $10^{-8}$) or to exceed the $300,000$ function evaluations (FEs). There are 30 independent runs for each benchmarking function and results are averaged over these independent trials.

## 4.3 BB, PSO and BBJ Parameter values

Bare Bones enjoys the luxury of having no adjustable parameters. The parameters defined by Bratton [12] were used for the CK trials. $\alpha$ was set to 0.75 for both $BBJ$ models, and, following the recommendations in [5] $p_J$ was fixed at 0.01 for BBJ1. Preliminary experiments suggested that BBJ2 performs better with a smaller $p_J$ and a value of 0.001 was used in the following. A global $\mu$ neighbourhood is used for BBJ in every experiment.

## 4.4 Results

In this experiment two types of $\sigma$-neighbourhoods (global and local) are tested. The results are shown in the following tables and figures:

- Global neighbourhood:
  - Table 1a reflects the accuracy of each algorithm over each function and the reliability of each algorithm averaged over all benchmarks in global neighbourhood. Table 1b highlights any significant difference in the accuracy of the algorithms over each function.
  - Table 2a shows the efficiency of each algorithm over each benchmark. Table 2b underlines any existing significant difference between any two algorithms over the benchmarks in the global neighbourhood.
  - Figure 1 shows the plots for the accuracy and efficiency measures.
- Local neighbourhood:
  - Table 3 displays the results using the same measures (accuracy and reliability) as Tables 1 but in the local neighbourhood topology.
  - Table 4 displays the results using the same measure (efficiency) as Table 2 but in a local neighbourhood topology.
  - Figure 2 shows the plots for the accuracy and efficiency measures.

Observing the reliability of the algorithms both in global and local neighbourhoods (see the last rows of Tables 1a and 3a), shows that on average BB is the least reliable algorithm. This finding does not come as a surprise as BB was proposed for understanding PSO rather than being deployed for optimisation purposes; the result of this experiment confirms this view empirically. Among other algorithms, BBJ2 shows the most reliable performance in both local and global neighbourhood. Additionally, BBJ2 shows better reliability in global vs. local neighbourhood, which is not always the expectation (as global neighbourhood is usually criticised for its premature convergence [13]. CK and BBJ1 show contradicting results in different neighbourhoods: BBJ1 is more reliable than CK in the global neighbourhood, but less reliable in the local neighbourhood.

In terms of the accuracy of the algorithms in the global neighbourhood (see Table 1b), BB shows significantly worse accuracy. When there exists convergence, in most cases, BBJ1 and BBJ2 outperform CK significantly. Over all benchmarks, BBJ1 and BBJ2 do not outperform each other significantly (except in $f_{11}$). As for the efficiency of the algorithms in the global neighbourhood (see Table 2), when there exists a significant difference BBJ2 outperform all algorithms over all benchmarks significantly. The second best algorithm is BBJ1.

In the local neighbourhood (see Table 3), compared to other algorithms, BB and BBJ1, are significantly worse in terms of accuracy. When functions with convergence are considered, BBJ2 outperform other algorithms. In terms of efficiency in the local neighbourhood (see Table 4b), CK is outperformed by BB in most significant cases. Observing functions with successful convergence, BBJ1 and BBJ2 are the least and the most efficient algorithms respectively.

In order to investigate the role of jumping in BBJ2, this mechanism is removed in a control algorithm – BBJ2 with No Jumps (BBNJ) – which uses the same parameters and update equations as BBJ2 but with $p_J = 0$. This algorithm, in terms of efficiency, outperforms BBJ2 in local neighbourhood in all 3 significant cases; however in global neighbourhood, BBNJ is outperformed by BBJ2 in all 4 significant cases. In terms of accuracy, both in global and local

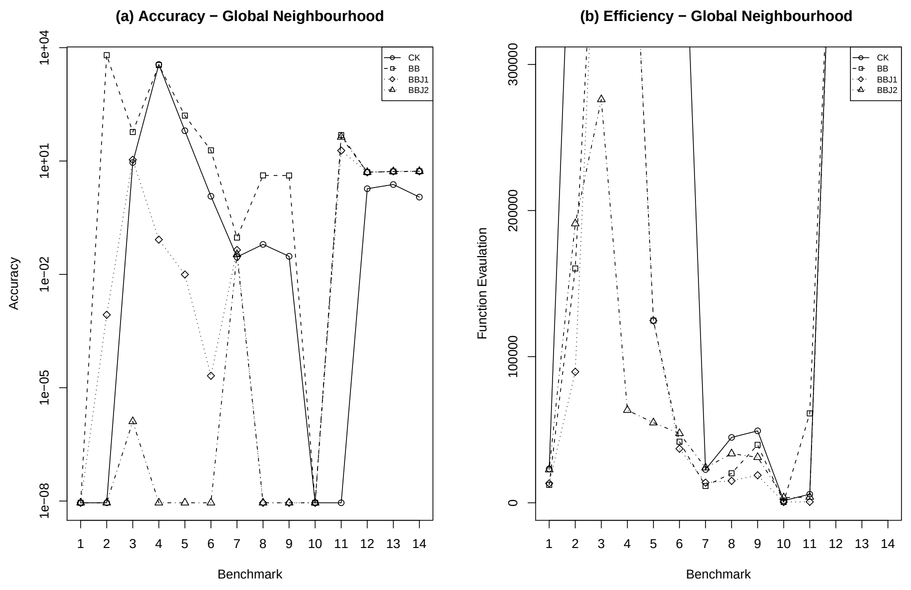

**Fig. 1.** Accuracy and efficiency in global neighbourhood

neighbourhood, whenever there is a difference, BBJ2 outperforms BBNJ in the entire cases, 12 of which are significantly better. Also in terms of reliability, BBNJ is the least reliable algorithm.

## 4.5 Discussion

More experiments are needed in order to form a concrete theoretical idea as to why BBJ2 outperforms the other algorithms. The initial thought behind this outperformance is the reliance on the difference between the particles' current positions and their neighbourhood best position. This effectively eliminates the direct influence of the particles' personal bests from the update equations. On the other hand, in the rest of the algorithms (used in this paper), each particle's personal best leaves a direct impact on the update equations. This presence of many influencing factors – which is one of the reasons why understanding PSO is complicated – in the update process might be counter-productive.

BB and BBJ, in contrast to CK, are distinguished by the absence of particle position information in the update rule. Search always begins at a point determined by particle informers $g$ or $g_i$ and the extent of the search is determined by informer separation, $|p_i - g_i|$ or $|p_{i-1} - p_{i-1}|$. A trial position $x_i \sim g_i + \sigma_i N(0, 1)$ is ignored if an informer $p_i$ is not bettered. The particle, figuratively speaking, returns to $p_i$ after a single trial at search centre $g_i$. On the other hand, BBJ2 retains information of an unsuccessful attempt since search spread is determined

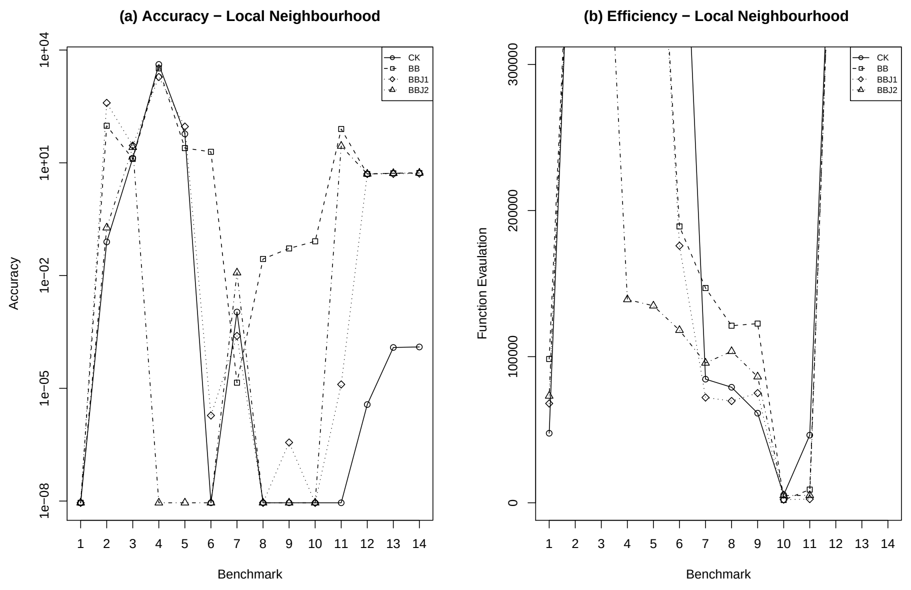

**Fig. 2.** Accuracy and efficiency in local neighbourhood

by the difference between $x_i$ and $g_i$. This provides a convergence inhibition mechanism: informers will crowd together as the swarm converges, with a consequent decrease, for BBJ1, in diversity. However in BBJ2, a trial position $x_i$ may lie beyond the informer group. This will lead to a broader search at the next iteration since $\delta^{BBJ2} = |g_i - x_i|$.

Finally, we note the significance of jumping: the probability of jumping in one or more dimensions is $1 - (1 - p_J)^D = 0.03$ (30 dimensions, $p_J = 0.001$). Even this small figure appears to be enough for enhanced performance. A law of diminishing returns applies since excessive jumping slows convergence. The fact that jumping appears to be less necessary in BBJ2 than in BBJ1 is perhaps attributable to the greater search diversity inherent in the formation of $\delta$. The efficacy of tail broadening for distribution based swarm optimisers has already been observed in a study of Lèvy bare bones [14]. We remark that tail broadening is a more subtle effect than re-initialisation. The latter is equivalent to jumping in each of the $D$ dimensions, occurring with only a very small probability ($prob = p_J^D$) in the BBJ models.

## 5 Conclusion

This paper briefly describes Bare Bones swarm optimisation which was proposed to provide better understanding of the behaviour of particle swarm algorithms. Although this algorithm does not intend to enhance the optimisation capability of standard PSO of Clerc-Kennedy (CK), the other variations (Bare Bones with Jumps Model 1 & 2) explained and introduced respectively in this paper offer promising results. The algorithms used in this paper are compared against each other using three performance measures (i.e. accuracy, efficiency and reliability). Using these measures, it is shown that in terms of accuracy, when benchmarks with successful convergence are considered, the accuracy of BBJ2 compared to all other algorithms is significantly better. Additionally, BBJ2 is empirically shown to be *both* the most efficient and the most reliable algorithm in both local and global $\sigma$ neighbourhoods. A brief discussion is also presented with the possible reasons which might boost the outperformance of BBJ2 compared to other algorithms, and an experiment is conducted to demonstrate that despite the very small jump probability of BBJ2, this mechanism plays a crucial role.

## References

1. Clerc, M., Kennedy, J.: The particle swarm-explosion, stability, and convergence in amultidimensional complex space. Evolutionary Computation, IEEE Transactions on 6(1) (2002) 58–73
2. Yang, Y., Kamel, M.: Clustering ensemble using swarm intelligence. In: Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE, IEEE (2003) 65–71
3. van den Bergh, F., P., E.A.: A study of particle swarm optimization particle trajectories. Information Sciences 176(8) (2006) 937–971
4. Kennedy, J.: Bare bones particle swarms. In: Proceedings of Swarm Intelligence Symposium, 2003 (SIS'03), IEEE (2003) 80–87
5. Blackwell, T.: A study of collapse in bare bones particle swarm optimisation. IEEE Transactions on Evolutionary Computing (99) (2012)
6. Trelea, I.C.: The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters 85(6) (2003) 317–325
7. Engelbrecht, A.P.: Fundamentals of Computational Swarm Intelligence. Wiley (2006)
8. Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the lipschitz constant. J. Optim. Theory Appl. 79(1) (1993) 157–181
9. Jong, K.A.D.: An analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan, Ann Arbor, MI, USA (1975)
10. al-Rifaie, M.M., Bishop, M., Blackwell, T.: Resource allocation and dispensation impact of stochastic diffusion search on differential evolution algorithm; in. In: Nature Inspired Cooperative Strategies for Optimisation (NICSO 2011), Springer. (2011)
11. Gehlhaar, D., Fogel, D.: Tuning evolutionary programming for conformationally flexible molecular docking. In: Evolutionary Programming V: Proc. of the Fifth Annual Conference on Evolutionary Programming. (1996) 419–429
12. Bratton, D., Kennedy, J.: Defining a standard for particle swarm optimization. In: Proc of the Swarm Intelligence Symposium, Honolulu, Hawaii, USA, IEEE (2007) 120–127
13. Clerc, M.: From theory to practice in particle swarm optimization. Handbook of Swarm Intelligence (2010) 3–36
14. Richer, T., Blackwell, T.: The lévy particle swarm. In: IEEE congress on evolutionary computation. (2006) 3150–3157

**Table 1.** Accuracy Details; Global Neighbourhood

(a) Accuracy± Standard Error is shown with two decimal places after 30 trials of 300,000 function evaluations. Total number of convergence of each algorithm over the benchmarks can be found in the last row.

| Fn | CK | BB | BBJ1 | BBJ2 | BBNJ |
|---|---|---|---|---|---|
| $f_1$ | 0.0 ±0.0 | 0.0 ±0.0 | 0.0 ±0.0 | 0.0 ±0.0 | 4.14E-05±4.13E-05 |
| $f_2$ | 0.0 ±0.0 | 6.34E+03±4.69E+02 | 8.51E-04±7.86E-04 | 0.0 ±0.0 | 2.72E+03±5.03E+02 |
| $f_3$ | 9.14E+00±3.18E+00 | 5.86E+01±1.80E+01 | 1.08E+01±4.47E+00 | 1.28E-06±6.09E-07 | 6.18E+00±2.80E+00 |
| $f_4$ | 3.60E+03±8.50E+01 | 3.46E+03±2.29E+01 | 8.32E-02±1.43E-02 | 0.0 ±0.0 | 4.61E+03±9.40E+01 |
| $f_5$ | 6.33E+01±2.57E+00 | 1.59E+02±4.93E+00 | 9.93E-03±3.37E-03 | 0.0 ±0.0 | 3.47E+02±9.56E+00 |
| $f_6$ | 1.17E+00±1.95E-01 | 1.92E+01±8.43E-02 | 2.07E-05±1.69E-05 | 0.0 ±0.0 | 1.98E+01±1.39E-02 |
| $f_7$ | 2.88E-02±6.13E-03 | 9.40E-02±3.39E-02 | 4.42E-02±7.18E-03 | 3.37E-02±6.43E-03 | 4.64E+00±2.18E+00 |
| $f_8$ | 6.22E-02±2.03E-02 | 4.16E+00±1.36E+00 | 0.0 ±0.0 | 0.0 ±0.0 | 6.44E+00±2.40E+00 |
| $f_9$ | 3.00E-02±1.44E-02 | 4.13E+00±3.23E+00 | 0.0 ±0.0 | 0.0 ±0.0 | 3.66E+01±1.92E+01 |
| $f_{10}$ | 0.0 ±0.0 | 0.0 ±0.0 | 0.0 ±0.0 | 0.0 ±0.0 | 2.72E-02±2.72E-02 |
| $f_{11}$ | 0.0 ±0.0 | 4.86E+01±7.37E+00 | 1.89E+01±6.36E+00 | 4.32E+01±7.50 | 5.67E+01±6.52E+00 |
| $f_{12}$ | 1.85E+00±4.97E-01 | 5.05E+00±0.00E+00 | 5.05E+00±7.38E-17 | 5.05E+00±1.13E-16 | 5.05E+00±9.99E-17 |
| $f_{13}$ | 2.39E+00±5.95E-01 | 5.27E+00±3.01E-17 | 5.35E+00±7.92E-02 | 5.27E+00±8.52E-17 | 5.27E+00±1.35E-16 |
| $f_{14}$ | 1.11E+00±4.68E-01 | 5.36E+00±6.02E-17 | 5.36E+00±9.03E-17 | 5.36E+00±9.52E-17 | 5.47E+00±1.11E-01 |
| $\sum$ | (180) | (99) | (198) | (268) | (93) |
| | 42.68% | 23.57% | 47.14% | 63.81% | 22.14% |

(b) Based on TukeyHSD Test, if the difference between each pair of algorithms is significant, the pairs are marked. X–o shows that the left algorithm is significantly better than the right one; and o–X shows that the right one is significantly better than the left algorithm.

| Fn | BBJ1-BB | BBJ2-BB | CK-BB | BBJ2-BBJ1 | CK-BBJ1 | CK-BBJ2 | BBNJ-BBJ2 |
|---|---|---|---|---|---|---|---|
| $f_1$ | – | – | – | – | – | – | – |
| $f_2$ | X – o | X – o | X – o | – | – | – | o – X |
| $f_3$ | X – o | X – o | X – o | – | – | – | o – X |
| $f_4$ | X – o | X – o | – | – | o – X | o – X | o – X |
| $f_5$ | X – o | X – o | X – o | – | o – X | o – X | o – X |
| $f_6$ | X – o | X – o | X – o | – | o – X | o – X | o – X |
| $f_7$ | – | – | – | – | – | – | o – X |
| $f_8$ | X – o | X – o | X – o | – | – | – | o – X |
| $f_9$ | – | – | – | – | – | – | – |
| $f_{10}$ | – | – | – | – | – | – | – |
| $f_{11}$ | X – o | – | X – o | o – X | – | X – o | – |
| $f_{12}$ | – | – | X – o | – | X – o | X – o | – |
| $f_{13}$ | – | – | X – o | – | X – o | X – o | – |
| $f_{14}$ | – | – | X – o | – | X – o | X – o | – |

**Table 2.** Efficiency Details; Global Neighbourhood

(a) Mean FEs (±standard error) is shown with two decimal places after 30 trials of 300,000 function evaluations.

| Fn | CK | BB | BBJ1 | BBJ2 | BBNJ |
|---|---|---|---|---|---|
| $f_1$ | 23224±194 | 12262±164 | 13270±148 | 22685±119 | 14454±244 |
| $f_2$ | – | 160358±2920 | 89637±575 | 191064±1290 | – |
| $f_3$ | – | – | – | 276020±7039 | 213310±7324 |
| $f_4$ | – | – | – | 63399±3805 | – |
| $f_5$ | – | 124701±12900 | 124701±12900 | 54825±3182 | – |
| $f_6$ | – | 41811±870 | 37004±318 | 47486±2226 | – |
| $f_7$ | 22786±259 | 11518±136 | 13807±335 | 24006±259 | 14036±833 |
| $f_8$ | 44735±567 | 20194±1701 | 15013±285 | 33627±744 | 21554±383 |
| $f_9$ | 49228±1309 | 39656±3719 | 18855±981 | 31147±720 | 26835±563 |
| $f_{10}$ | 1458±17 | 516±4 | 551±5 | 3515±37 | 534±8 |
| $f_{11}$ | 5876±397 | 61199±11951 | 663±10 | 3929±39 | 649±10 |
| $f_{12}$ | – | – | – | – | – |
| $f_{13}$ | – | – | – | – | – |
| $f_{14}$ | – | – | – | – | – |

(b) Based on TukeyHSD Test, if the difference between each pair of algorithms is significant, the pairs are marked. X–o shows that the left algorithm is significantly better than the right one; and o–X shows that the right one is significantly better than the left algorithm.

| Fn | BBJ1-BB | BBJ2-BB | CK-BB | BBJ2-BBJ1 | CK-BBJ1 | CK-BBJ2 | BBNJ-BBJ2 |
|---|---|---|---|---|---|---|---|
| $f_1$ | X – o | X – o | – | – | o – X | o – X | o – X |
| $f_2$ | NP | NP | NP | X – o | o – X | o – X | NP |
| $f_3$ | NP | NP | NP | NP | NP | NP | o – X |
| $f_4$ | NP | NP | NP | NP | NP | NP | NP |
| $f_5$ | NP | NP | NP | X – o | NP | NP | NP |
| $f_6$ | NP | NP | NP | X – o | o – X | o – X | NP |
| $f_7$ | X – o | X – o | – | – | o – X | o – X | – |
| $f_8$ | X – o | X – o | – | – | o – X | o – X | o – X |
| $f_9$ | – | X – o | – | X – o | – | – | o – X |
| $f_{10}$ | X – o | X – o | o – X | – | o – X | o – X | – |
| $f_{11}$ | o – X | – | – | X – o | X – o | – | – |
| $f_{12}$ | NP | NP | NP | NP | NP | NP | NP |
| $f_{13}$ | NP | NP | NP | NP | NP | NP | NP |
| $f_{14}$ | NP | NP | NP | NP | NP | NP | NP |

**Table 3.** Accuracy Details; Local Neighbourhood

(a) Accuracy ± Standard Error is shown with two decimal places after 30 trials of 300,000 function evaluations. Total number of convergence of each algorithm over each benchmark is shown in brackets after the accuracy and standard error. Total number of convergence of each algorithm over the benchmarks can be found in the last row.

| Fn | CK | BB | BBJ1 | BBJ2 | BBNJ |
|---|---|---|---|---|---|
| $f_1$ | 0.0 ±0.0 | 0.0 ±0.0 | 0.0 ±0.0 | 0.0 ±0.0 | 9.57E-09±1.14E-10 |
| $f_2$ | 7.84E-02±1.09E-02 | 9.66E+01±8.68E+00 | 3.93E+02±4.38E+01 | 1.87E-01±3.02E-02 | 2.55E-01±2.02E-01 |
| $f_3$ | 1.33E+01±3.73E+00 | 1.27E+01±5.50E-01 | 2.88E+01±3.20E+00 | 2.59E+01±5.73E+00 | 2.99E+01±6.02E+00 |
| $f_4$ | 4.14E+03±7.11E+01 | 3.26E+03±3.10E+01 | 1.92E+03±6.89E+01 | 0.0 ±0.0 | 4.03E+03±4.77E+01 |
| $f_5$ | 5.87E+01±1.88E+00 | 2.46E+01±3.04E+00 | 9.22E+01±4.47E+00 | 0.0 ±0.0 | 2.85E+02±6.11E+00 |
| $f_6$ | 0.0 ±0.0 | 1.96E+01±2.24E-02 | 1.89E-06±1.55E-06 | 0.0 ±0.0 | 1.98E+01±1.27E-02 |
| $f_7$ | 1.07E-03±6.10E-04 | 1.41E-05±1.04E-05 | 2.48E-04±2.46E-04 | 1.19E-02±2.96E-03 | 1.95E-02±4.65E-03 |
| $f_8$ | 0.0 ±0.0 | 2.76E-02±1.92E-02 | 0.0 ±0.0 | 0.0 ±0.0 | 7.01E-01±2.69E-01 |
| $f_9$ | 0.0 ±0.0 | 5.27E-02±5.27E-02 | 3.62E-07±2.84E-07 | 0.0 ±0.0 | 2.05E-01±6.39E-02 |
| $f_{10}$ | 0.0 ±0.0 | 8.16E-02±4.55E-02 | 0.0 ±0.0 | 0.0 ±0.0 | 5.84E-09±5.17E-10 |
| $f_{11}$ | 0.0 ±0.0 | 7.92E+01±2.71E+01 | 1.27E-05±1.27E-05 | 2.79E+01±7.03E+00 | 4.86E+01±7.37E+00 |
| $f_{12}$ | 3.70E-06±1.27E-07 | 5.05E+00±0.00E+00 | 5.05E+00±0.00E+00 | 5.05E+00±4.26E-17 | 5.05E+00±1.13E-16 |
| $f_{13}$ | 1.22E-04±0.00E+00 | 5.27E+00±0.00E+00 | 5.10E+00±1.76E-01 | 5.27E+00±0.00E+00 | 5.27E+00±4.26E-17 |
| $f_{14}$ | 1.26E-04±1.12E-16 | 5.36E+00±5.22E-17 | 5.18E+00±1.79E-01 | 5.36E+00±1.09E-16 | 5.36E+00±6.02E-17 |
| $\sum$ | (208) | (145) | (199) | (241) | (108) |
| | 49.52% | 34.52% | 47.38% | 57.38% | 25.71% |

(b) Based on TukeyHSD Test, if the difference between each pair of algorithms is significant, the pairs are marked. X–o shows that the left algorithm is significantly better than the right one; and o–X shows that the right one is significantly better than the left algorithm.

| Fn | BBJ1-BB | BBJ2-BB | CK-BB | BBJ2-BBJ1 | CK-BBJ1 | CK-BBJ2 | BBNJ-BBJ2 |
|---|---|---|---|---|---|---|---|
| $f_1$ | – | – | – | – | – | – | – |
| $f_2$ | o – X | X – o | X – o | X – o | X – o | – | – |
| $f_3$ | o – X | – | – | – | X – o | – | – |
| $f_4$ | X – o | X – o | o – X | X – o | o – X | o – X | o – X |
| $f_5$ | o – X | X – o | o – X | X – o | X – o | o – X | o – X |
| $f_6$ | X – o | X – o | X – o | – | – | – | o – X |
| $f_7$ | – | o – X | – | o – X | – | X – o | – |
| $f_8$ | – | – | – | – | – | – | o – X |
| $f_9$ | – | – | – | – | – | – | o – X |
| $f_{10}$ | – | – | – | – | – | – | – |
| $f_{11}$ | X – o | – | X – o | – | – | – | o – X |
| $f_{12}$ | – | – | X – o | – | X – o | X – o | – |
| $f_{13}$ | – | – | X – o | – | X – o | X – o | – |
| $f_{14}$ | – | – | X – o | – | X – o | X – o | – |

**Table 4.** Efficiency Details; Local Neighbourhood

(a) Mean FEs ±Standard Error is shown with two decimal places after 30 trials of 300,000 function evaluations.

| **Fn** | **CK** | **BB** | **BBJ1** | **BBJ2** | **BBNJ** |
|---|---|---|---|---|---|
| $f_1$ | 47589±97 | 98383±327 | 67968±213 | 73090±196 | 49574±260 |
| $f_2$ | – | – | – | – | – |
| $f_3$ | – | – | – | – | – |
| $f_4$ | – | – | – | 139118±3975 | – |
| $f_5$ | – | – | – | 134816±2801 | – |
| $f_6$ | – | 189139±4687 | 175902±944 | 118098±389 | – |
| $f_7$ | 84612±4962 | 146979±4494 | 72048±332 | 95680±4051 | 49970±396 |
| $f_8$ | 79067±765 | 121186±1035 | 69658±489 | 103658±1287 | 68597±1434 |
| $f_9$ | 61328±374 | 122631±853 | 75080±392 | 86281±480 | 71144±1217 |
| $f_{10}$ | 5389±100 | 1891±31 | 2161±161 | 4935±53 | 1716±31 |
| $f_{11}$ | 46300±2012 | 9030±2367 | 2536±75 | 5063±51 | 2891±184 |
| $f_{12}$ | – | – | – | 8895±0 | – |
| $f_{13}$ | – | – | – | – | – |
| $f_{14}$ | – | – | – | – | – |

(b) Based on TukeyHSD Test, if the difference between each pair of algorithms is significant, the pairs are marked. X–o shows that the left algorithm is significantly better than the right one; and o–X shows that the right one is significantly better than the left algorithm.

| **Fn** | **BBJ1-BB** | **BBJ2-BB** | **CK-BB** | **BBJ2-BBJ1** | **CK-BBJ1** | **CK-BBJ2** | **BBNJ-BBJ2** |
|---|---|---|---|---|---|---|---|
| $f_1$ | o – X | o – X | o – X | X – o | X – o | o – X | X – o |
| $f_2$ | NP | NP | NP | NP | NP | NP | NP |
| $f_3$ | NP | NP | NP | NP | NP | NP | NP |
| $f_4$ | NP | NP | NP | NP | NP | NP | NP |
| $f_5$ | NP | NP | NP | NP | NP | NP | NP |
| $f_6$ | NP | NP | NP | X – o | X – o | X – o | NP |
| $f_7$ | o – X | – | – | X – o | X – o | – | X – o |
| $f_8$ | o – X | X – o | o – X | X – o | X – o | o – X | – |
| $f_9$ | o – X | o – X | o – X | X – o | X – o | o – X | X – o |
| $f_{10}$ | X – o | X – o | – | – | o – X | o – X | – |
| $f_{11}$ | X – o | X – o | X – o | – | – | – | – |
| $f_{12}$ | NP | NP | NP | NP | NP | NP | NP |
| $f_{13}$ | NP | NP | NP | NP | NP | NP | NP |
| $f_{14}$ | NP | NP | NP | NP | NP | NP | NP |