# A Parallel Gannet Optimization Algorithm with Communication Strategies (PGOA)

Jingbo Su[1], Ruobin Wang[1,2*], Fangdong Geng[1], Qiang Wei[1], and Lin Xu[3*]

[1] School of Information Science and Technology, North China University of Technology, 100144 Beijing, China
[2] Beijing Urban Governance Research Center, North China University of Technology, 100144 Beijing, China
[3] STEM, University of South Australia, 5095 Adelaide, Australia

**Abstract.** Meta-heuristic algorithms have gained remarkable success in solving complex and large-scale problems. However, as the dimension of the problem increases, their elaborate implementations may lead to lower convergence speed and struggle with local optima easily. Therefore, it is time-consuming for people to tune for high-dimensional problems. In this paper, a parallel Gannet Optimization Algorithm (GOA) for two novel communication strategies is proposed, and comparisons with the original GOA on 13 100-dimension benchmark functions are committed. Comprehensive experimental results indicate that the improved algorithm outperforms the original algorithm in not only better escaping local optima, but shorter running time.

**Keywords:** Gannet Optimization Algorithm, Parallel, Communication Strategies

## 1 Introduction

Meta-heuristic algorithms, i.e., optimization methods designed according to the strategies laid out in the meta-heuristic framework are mostly designed based on the activities of natural organisms. There are generally three categories of meta-heuristics that are invited to solve various problems, for instance, local search meta-heuristics (e.g. Simulated Annealing, SA) can not only use strategies to evade local optimum but record search information and utilize them to find better solutions; constructive meta-heuristics (e.g. Ant Colony Optimization, ACO) are often adaptations of greedy algorithms to search for the optimal solution in a local phase; population-based meta-heuristics like Genetic Algorithms (GA) and Evolutionary Algorithms (EA) are designed to update the worse individuals by the recombination of some parts of the total population. Therefore, meta-heuristics can be a viable alternative to most exact methods such as branch-and-bound and dynamic programming when dealing with large-scale and complex problems by generating randomness. Currently, many algorithms use parallel techniques and refine the original algorithms to better performance, for instance, Parallel Genetic Algorithm (PGA), Parallel

Ant Colony Optimization (PACO), Parallel Compact Fish Migration Optimization[1] (PCFMO), etc. Furthermore, proposing novel meta-heuristic algorithms with superior performance and simpler implements has recently become one of the hottest research directions. There are many efficient algorithms, for example, Tumbleweed Algorithm[2] (TA), Bamboo Forest Growth Optimizer[3] (BFGO), and Gannet Optimization Algorithm[4] (GOA). Although GOA, inspired by the Gannets' behavior underwater, outstandingly solves continuous complex optimization problems, it still suffers from the drawback of easily falling into local optimum when solving high-dimensional problems. As a result, parallel communication strategies will be proposed in this paper and they are accustomed to advancing the property of GOA. We use 13 traditional mathematical test functions to verify the efficiency of the improved algorithm, and simulation outcomes show that it outperforms the original GOA in both convergence behavior and running time.

The rest of the paper is organized as follows. In Section 2, related work will be provided. Section 3 explains the proposed communication strategies of PGOA in detail. The data from simulation results are used to demonstrate and discuss the efficiency of our algorithm in Section 4 and the summary of this paper in Section 5.

## 2 Related Work

### 2.1 Gannet Optimization Algorithm

The Gannet Optimization Algorithm, as a novel nature-inspired meta-heuristic algorithm, mathematics the various unique behaviors of gannets during foraging and is used to enable exploration and exploitation[4]. And there are a host of similarities of GOA (including individual position matrix, local optima escaping methods, etc.) with other meta-heuristics algorithms. But differently, thanks to its U-shaped and V-shaped diving patterns (during the exploration phase), GOA is more likely to explore the optimal region within the search space, meanwhile, sudden turns and random walks can ensure finding better solutions. Experiments disclose that the GOA shows significantly superior results and additional efficiency over other algorithms as the dimension increases. Based on the competitive advantages of GOA in high dimensions, thus we decide to improve it to solve high-dimensional problems faster.

**Initialization Phase.** The GOA starts with a random solution matrix X, as in Eq.1. The individual vector $x_i$ , one of particles with D dimensions, denotes the position of the i-th individual. And each individual can be calculated by $x_{i,j} = r_1 \cdot (ub_j - lb_j) + lb_j,\ i = 1, 2, \ldots, N,\ j = 1, 2, \ldots, D$ is equivalent to a candidate solution to the problem.

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,D} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,D} \end{bmatrix} \tag{1}$$

$r$ is a random number uniformly distributed between 0 and 1, thus the probability of exploration and exploitation phases is equal.

**Exploration Phase.** As gannets find prey, they adjust their dive pattern in terms of the depth of the prey diving. There are two types of diving are purposed: a long and deep U-shaped dive and a short and shallow V-shaped dive. At any moment, $t$ will be represented as $1 - Iter/T_{max\_iter}$. Eq.2-5 are the equations for the exploration phase of GOA as follows:

$$X_m(t) = \frac{1}{N} \sum_{i=1}^{N} X_i(t) \tag{2}$$

$$u_2 = 2 \cdot t \cdot (2r_4 - 1) \cdot \cos(2\pi r_2) \cdot (X_i(t) - X_r(t)) \tag{3}$$

$$v_2 = 2 \cdot t \cdot (2r_5 - 1) \cdot V(2\pi r_3) \cdot (X_i(t) - X_m(t)) \tag{4}$$

$$MX_i(t+1) = \begin{cases} X_i(t) + u_1 + u_2, \ q \geq 0.5 \\ \\ X_i(t) + v_1 + v_2, \ q < 0.5 \end{cases} \tag{5}$$

$q$ and all $r_i$ are random values ranging from 0 to 1; the gannet will behave in a U-shaped dive pattern if $q \geq 0.5$ , and use a V-shaped pattern otherwise. Here, V-shaped formula can be expressed as: $V(x) = \begin{cases} -x/\pi + 1, \ x \in (0, \pi) \\ x/\pi - 1, \ x \in (\pi, 2\pi) \end{cases}$ .

**Exploitation Phase.** Two actions are proposed when the gannet captures prey after rushing into the water-Levy and Turns. Here they define a variable called Capture Capacity (we use CC as its abbreviation in the expression), which is primarily affected by the gannet energy. If the gannets have sufficient energy, they will perform a Levy random walk, otherwise, in most cases, Turn behavior is common when catching prey. Eq.6-11 are the equations for GOA exploitation phase:

$$R = \frac{M \cdot velocity^2}{0.2 + (2 - 0.2) \cdot r_6} \tag{6}$$

$$CC = \frac{1}{R \cdot (1 + \frac{Iter}{T_{max\_iter}})} \tag{7}$$

$$\sigma = \left(\frac{\Gamma(1+\beta)\cdot \sin(\frac{\pi\beta}{2})}{\Gamma(\frac{1+\beta}{2}\cdot\beta\cdot 2^{\frac{\beta-1}{2}})}\right)^{\frac{1}{\beta}} \tag{8}$$

$$Levy(Dim) = 0.01\cdot\frac{\mu\cdot\sigma}{|v|^{\frac{1}{\beta}}} \tag{9}$$

$$delta = CC\cdot|X_i(t) - X_{best}(t)| \tag{10}$$

$$MX_i(t+1) = \begin{cases} t\cdot delta\cdot(X_i(t) - X_{best}(t)) + X_i(t)\ ,\ \mathrm{CC}\geq c \\ \\ X_{best} - (X_i(t) - X_{best})\cdot Levy(Dim)\cdot t\ ,\ \mathrm{CC} < c \end{cases} \tag{11}$$

$M = 2.5kg$, $velocity = 1.5m/s$, and $\beta = 1.5$ are constant values; $\mu$, $\sigma$, $c$ and $r_6$ are random values ranging from 0 to 1; the gannet will perform Levy walk if $CC \geq c$ $(c = 0.2)$ , and reveal Turn walk otherwise.

### 2.2 Parallel Mechanisms

There are many shortcomings of novel meta-heuristic algorithms found while solving high-dimensional problems because of their complicated exploration and exploitation phases. So the parallel strategy becomes a common improvement method for meta-heuristic algorithms.

There are two main forms of parallelism: one is parallelism on hardware[5][6], we call this type "true parallelism"; the other is grouping[7], which communicates current information every certain number of iterations[8].

Although multiple pieces of hardware do speed up the program execution compared to a single processor, numerous factors need to be considered first, like implementation platform, parallel model, etc., moreover, the costs of hardware, especially GPU costs, are too exorbitant.

Consequently, in recent decades, more and more researchers prefer to improve meta-heuristics with the other type, virtual parallelism, since it is easy to implement and efficient. For example, three communication strategies were proposed in Parallel Particle Swarm Optimization (PPSO) to improve the performance of PSO[7], PMPSO declared two parallel strategies which are used to solve MaOPs (Many-Objective Optimization Problems)[9], and a novel parallel heterogeneous meta-heuristic model with multiple communication strategies was generated to predict the wind power[10], etc.

The virtual parallel processing is conducted through the communications between groups (i.e. replacing poorer solutions from other groups), making it easier to implement multi-parallel searches in the space, which can not only accelerate the convergence speed but also avoid falling into the local optima. Therefore, most algorithms with parallel strategy can be widely used to solve complex engineering problems and do large-scale optimization such as wireless

sensor networks[11] and robot path planning[12], etc. since they outperform the fundamental algorithms in these aspects.

For the reasons above, virtual parallelism will be detailed and used in this paper because of its preeminent merits.

## 3 Parallel Gannet Optimization Algorithm

### 3.1 Initialization

Generate $N_p$ individuals $X_{d,i}^g$ for the g-th group, $d = 1, \ldots, D,\ i = 1, \ldots, N_p,\ g = 1, \ldots, G$. $G$ is the group number, which can be completely divided by $2^k$ (k is an integer); D is the dimension of the problem.

### 3.2 Update

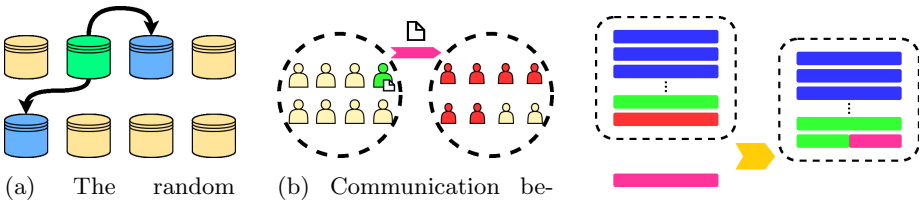
Four group arrays are proposed to record different fitness values: $pop\_fit_g^i$ records the solution of each individual in all groups; $group\_best_g$ and $group\_fmin$ note the best individual and its value in each group; $group\_worst_g$ and $group\_fmax$, in turn, indicate the worst one and its fitness solution in each group, and $global\_best$ and $global\_fmin$ represent the best solution for the population.

**Evaluation.** The value of $fitness\_func(X_{d,i}^g)$ of each D-dimensional individual is the evaluation of their execution solution.

### 3.3 Communication strategies

**Strategy 1.** The parallel communication strategy 1 is displayed in Fig.1a and Fig.1b. $X$ non-current groups will be randomly selected and sorted in descending order. The process is portrayed in Fig.1a. Then we migrate the best solution to the current group $group\_best_g$ to replace 75% worse individuals in $group_q$ every $T$ iterations, just like Fig.1b depicts. (Here, g, the current group, ranges in 0 to $(G-1)$ and $q = g \oplus 2^n$; $n = 0, \ldots, m-1$, where $G = 2^m$; the substitution rate is set before the iteration, and individuals in the random group whose fitness value is worse than the fitness value of the current group are replaced according to the substitution rate.)

**Strategy 2.** The process of communication strategy 2 is disclosed in Fig.1c. The main idea of this strategy is to replace the worst solution of the current group ($group_{worst}^i$) with a combination of half of the global best individual ($global_{best}$) and half of the best individual in this group ($group_{best}^i$). After the substitution, update the individuals of $group_{worst}$, $group_{best}$, $global_{best}$ and their fitness value.



(a) The random group selection process for strategy 1.

(b) Communication between group $group_g$ and $group_q$ for strategy 1.

(c) The second parallel communication strategy.

Fig. 1: Two parallel communication strategies.

## 4 Complexity Analysis

In theory, the complexity of the PGOA is the same as GOA. Without considering the calculation of the fitness functions, the time complexity of the most two time-consuming components is the initialization process and the update of the gannet positions. First, we initialize $N$ individuals at the initialization phase and evaluate their fitness values, so the complexity of this phase is $O(N)$. In the next process, the complexity of PGOA for gannets' location updates is $O(T \times N_p \times G) + O(T \times N_p \times G \times D)$, where $T$ is the total iterations of the test; $G$ is the number of groups and $N_p$ is the individual number in each group, and $N_p \times G = N$ obviously. However, according to the experimental results, we find that the speed of PGOA is faster than the GOA when they are tested in the same function, and the more groups are divided, the faster the execution will be. In fact, PGOA and GOA have the same $N$ update times in total, but PGOA has $N_p$ individuals per update. And it is easier and faster for machines to compute small-scale matrices. Therefore, the running time of PGOA is shorter than that of GOA although PGOA has $G - 1$ more updates per iteration than GOA.

## 5 Experiments

A series of comprehensive experiments are conducted to test the performance of the PGOA in higher dimensional uni-modal and multi-modal CEC2013 functions. The pseudo-code of PGOA with 2 communication strategies is presented in section4.1; two types of functions of CEC2013 are shown in the next two sub-sections; In the discussion, we show the comparison of the best value and running time.

### 5.1 Pseudo-code of PGOA[4]

```
Parameters: N, D, lb, ub, G=4.
Ensure: global_best and global_fmin.
Initialize: G[i], Np = N / G, n = log(G), m = 0.
```

[4] Code of PGOA is accessible at `https://github.com/sujingbo0217/PGOA`

```
for i = 1 to max_iter
  for g = 1 to G
    if rand > 0.5
      do GOA Exploration Phase
    else
      do GOA Exploitation Phase
    for j = 1 to Np
      do Update
        G[g].pop_fit[j], global_best, global_fmin,
        G[g].best, G[g].fmin, G[g].worst, G[g].fmax
    if i % 5 = 0
      do Strategy.1
    if i % 3 == 0
      do Strategy.2
```

### 5.2 Benchmark Functions

The expression of CEC2013 test functions are displayed on Table.1. In addition, the lower bound and upper bound of each function are shown as well. **F1-F7** are uni-model functions and most of them have a single peak so that tested algorithms can converge easily, thus they are often used to test the convergence rate of algorithms; **F8-F13** are multi-model functions and have many local optimum points which are difficult to be optimized. Therefore, they are commonly used to verify the robustness of algorithms on escaping local optima. Moreover, the dimension of all benchmark functions is set to 100, which means that the dimension of all these functions is higher than before (30D) in order to test the adjustment of our algorithm on high dimensions.

### 5.3 Discussion

For fair, each experiment runs 30 times and averages the best value and running time. In addition, other parameters are represented in Table.2.

Average best solution and running time are displayed in Table.3. It identifies that the average solution of PGOA outperforms the GOA in higher dimensions. Furthermore, the average running time of PGOA is significantly shorter than the original GOA. For a more intuitive comparison, the **difference rate** of Average Best Solution and the **difference** of Average Running Time are shown as bars at the following Fig.2

## 6 Conclusion

In this paper, GOA based on parallel communication strategies is proposed. Multiple experiments are conducted to validate the outstanding performance of the PGOA. In the higher dimension of the benchmark functions, both the optimal value and the execution speed of the PGOA are better than the original

GOA because of the randomness and substitution generated by the parallel group communication strategies. As a consequence, it will be a great attempt to apply the parallel method to solve a series of high-dimensional problems related to data mining, neural networks, and further fields in future work.

**Acknowledgement.** This work is supported by Beijing College Students Innovation and Entrepreneurship Training Project 2022.

## References

1. S.-C. Chu, X.-W. Xu, S.-Y. Yang; J.-S. Pan: Parallel fish migration optimization with compact technology based on memory principlefor wireless sensor networks. Knowl.-Based Syst. 2022, 241, 108124 (2022)
2. Q.-Y. Yang, S.-C. Chu, A.-H. Liang, J.-S. Pan: Tumbleweed Algorithm and Its Application for Solving Location Problem of Logistics Distribution Center. In: S.-C. Chu et al. (Eds.): ICGEC 2021, LNEE 833, pp. 641–652 (2022)
3. Q. Feng, S.-C. Chu, J.-S. Pan, J. Wu, T.-S. Pan: Energy-Efficient Clustering Mechanism of Routing Protocol for Heterogeneous Wireless Sensor Network Based on Bamboo Forest Growth Optimizer. in Entropy 2022, 24, 980. https://doi.org/10.3390/e24070980 (2022)
4. J.-S. Pan, R.-B. Wang, S.-C. Chu: Gannet optimization algorithm: A new metaheuristic algorithm for solving engineering optimization problems. Mathematics and Computers in Simulation (2022)
5. Venter, G., Sobieszczanski-Sobieski, J.: Parallel particle swarm optimization algorithm accelerated by asynchronous evaluations. Journal of Aerospace Computing, Information, and Communication, vol. 3, no. 3, pp. 123–137 (2006)
6. B. Cao, J. Zhao, Z. Lv, X. Liu, S. Yang, X. Kang, K. Kang: Distributed parallel particle swarm optimization for multi-objective and many-objective large-scale optimization. IEEE Access, vol. 5, pp. 8214–8221 (2017)
7. Roddick, J. F.: A parallel particle swarm optimization algorithm with communication strategies. Journal of Information Science and Engineering, vol. 21, no. 4, pp. 809–818 (2005)
8. Y. Sun, P. Hu, J.-S. Pan, S.-C. Chu: Overview of Parallel Computing for Meta-Heuristic Algorithms. In: Journal of Network Intelligence, Taiwan Ubiquitous Information, vol. 7, no. 3, pp. 656-684 (2022)
9. De Campos Jr, A., Pozo, A. T., Duarte Jr, E. P.: Parallel multi-swarm pso strategies for solving many objective optimization problems. Journal of Parallel and Distributed Computing, vol. 126, pp. 13–33 (2019)
10. J.-S. Pan, P. Hu, S.-C. Chu: Novel Parallel Heterogeneous Meta-Heuristic and Its Communication Strategies for the Prediction of Wind Power. Processes 2019, 7, 845. https://doi.org/10.3390/pr7110845 (2019)
11. R.-B. Wang, W.-F. Wang, L. Xu, J.-S. Pan, S.-C. Chu: Improved DV-Hop based on parallel and compact whale optimization algorithm for localization in wireless sensor networks. Wireless Networks. Springer (2022)
12. R.-B. Wang, W.-F. Wang, L. Xu, J.-S. Pan, S.-C. Chu: An Adaptive Parallel Arithmetic Optimization Algorithm for Robot Path Planning. https://www.researchgate.net/deref/https%3A%2F%2Fwww.hindawi.com%2Fjournals%2Fjat%2F2021%2F3606895%2F (2021)

Table 1: Benchmark Functions

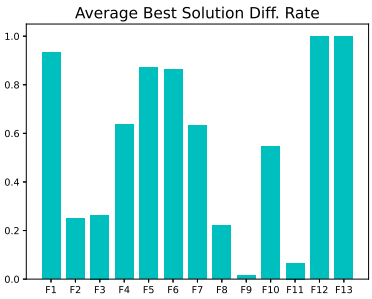
| Name | Function | LB | UB |
|---|---|---|---|
| F1 | $f(x) = \sum_{i=1}^{D} x_i^2$ | -100 | 100 |
| F2 | $f(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$ | -1 | 3 |
| F3 | $f(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)$ | -100 | 100 |
| F4 | $f(x) = \max\{\lvert x_i \rvert,\ 1 \le i \le n\}$ | -100 | 100 |
| F5 | $f(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$ | -5 | 10 |
| F6 | $f(x) = \sum_{i=1}^{D} (x_i + 0.5)^2$ | -100 | 100 |
| F7 | $f(x) = \sum_{i=0}^{D} i \cdot x_i^4 + rand(0,1)$ | -1.28 | 1.28 |
| F8 | $f(x) = \sum_{i=1}^{D} (-x \cdot \sin(\sqrt{\lvert x_i \rvert}))$ | -500 | 500 |
| F9 | $f(x) = 10D + \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i)]$ | 2.56 | 5.12 |
| F10 | $f(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$ | -32 | 32 |
| F11 | $f(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1$ | 300 | 600 |
| F12 | $f(x) = \frac{\pi}{D}\{10 \cdot \sin(\pi y_1)\} + \sum_{i=1}^{D-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_i + 1) + \sum_{i=1}^{D} u(x_i, 10, 100, 4)]$, where $y_i = 1 + \frac{x_i+1}{4}$ | -50 | 50 |
| F13 | $f(x) = 0.1(\sin^2(3\pi x_1) + \sum_{i=1}^{D}(x_i - 1)^2[1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2 + \sin^2(2\pi x_n)) + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$ | -50 | 50 |

Table 2: Parameter settings for PGOA

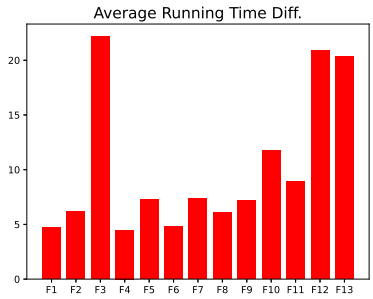
| Population (N) | Group (G) | Copy time (X) | Communication iteration (T) |
|---|---|---|---|
| 80 | 8 | 2 | 5 |

Table 3: Experimental Performance

| | Average Best Solution | | Average Running Time (s) | |
|---|---|---|---|---|
| Function | GOA | PGOA | GOA | PGOA |
| F1 | 559.26 | **37.67** | 5.63 | **0.90** |
| F2 | 54.37 | **40.81** | 7.27 | **1.10** |
| F3 | -687142.08 | **-868354.29** | 25.78 | **3.58** |
| F4 | 5.41 | **1.97** | 5.32 | **0.84** |
| F5 | 401622.14 | **51475.68** | 8.51 | **1.26** |
| F6 | 2635.95 | **364.50** | 5.69 | **0.89** |
| F7 | 533.88 | **195.87** | 8.66 | **1.29** |
| F8 | -14601.28 | **-17828.41** | 7.23 | **1.12** |
| F9 | 2161.49 | **2129.12** | 8.41 | **1.26** |
| F10 | 7.88 | **3.58** | 13.68 | **1.95** |
| F11 | 3442.19 | **3212.37** | 10.40 | **1.51** |
| F12 | 916459459.72 | **51.93** | 24.19 | **3.30** |
| F13 | 47749.57 | **10.05** | 23.58 | **3.25** |




(a) Difference rate




(b) Difference

Fig. 2: Comparison