2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Multi-swarm Genetic Gray Wolf Optimizer with Embedded Autoencoders for High-dimensional Expensive Problems

Jing Bi[1], Jiahui Zhai[1], Haitao Yuan[2], Ziqi Wang[1], Junfei Qiao[1], Jia Zhang[3] and MengChu Zhou[4]

Abstract— **High-dimensional expensive problems are often encountered in the design and optimization of complex robotic and automated systems and distributed computing systems, and they suffer from a time-consuming fitness evaluation process. It is extremely challenging and difficult to produce promising solutions in a high-dimensional search space. This work proposes an evolutionary optimization framework with embedded autoencoders that effectively solve optimization problems with high-dimensional search space. Autoencoders provide strong dimension reduction and feature extraction abilities that compress a high-dimensional space to an informative low-dimensional one. Search operations are performed in a low-dimensional space, thereby guiding whole population to converge to the optimal solution more efficiently. Multiple subpopulations coevolve iteratively in a distributed manner. One subpopulation is embedded by an autoencoder, and the other one is guided by a newly proposed Multi-swarm Gray-wolf-optimizer based on Genetic-learning (MGG). Thus, the proposed multi-swarm framework is named Autoencoder-based MGG (AMGG). AMGG consists of three proposed strategies that balance exploration and exploitation abilities, *i.e.*, a dynamic subgroup number strategy for reducing the number of subpopulations, a subpopulation reorganization strategy for sharing useful information about each subpopulation, and a purposeful detection strategy for escaping from local optima and improving exploration ability. AMGG is compared with several widely used algorithms by solving benchmark problems and a real-life optimization one. The results well verify that AMGG outperforms its peers in terms of search accuracy and convergence efficiency.**

I. INTRODUCTION

Evolutionary algorithms (EAs) are inspired by the evolutionary operations of living organisms in nature. EAs include basic operations such as genetic coding, population initialization, crossover variations, and operational retention mechanisms. Compared with traditional optimization algorithms, *e.g.*, calculus-based methods, and exhaustive search ones, EAs provide global optimization mechanisms with high robustness and wide applicability. They are self-organizing, adaptive and self-learning. They can effectively deal with complex problems, *e.g.*, NP-hard problems [1], [2], which are difficult to be solved by traditional optimization algorithms. Thus, EAs have been widely applied in robotics and automation [3], [4], industrial scheduling [5], resource allocation [6], and other fields [7]-[9].

With the advent of cloud computing, big data, and artificial intelligence, more and more optimization problems suffer from the high dimensionality issue. In addition, a large number of fitness evaluations (FEs) are often required before satisfactory solutions can be obtained for EAs [10]. However, each FE may consume a large amount of time or resources for many real-world high-dimensional expensive problems (HEPs), which is intractable and unacceptable [11], [12]. The efficiency of handling HEPs decreases rapidly as the dimensions of problems continue to increase due to the difficulty of constructing accurate surrogate models with limited data samples [13], [14].

Rather than using surrogate models, generating high-quality offsprings more efficiently provides an alternative option to HEPs with limited computing resources. In other words, if more promising solutions can be generated, the use of expensive FEs can better lead to significant improvement in the efficiency and quality of finally obtained solutions. Therefore, a core idea of solving HEPs can resort to accelerating the reproduction of promising solutions. Nevertheless, an extremely large search space associated with complex HEPs makes it a big challenge to produce high-quality solutions with limited computing resources. It brings a chance to adopt dimension reduction techniques to yield promising solutions in lower-dimensional spaces more efficiently [10]. It is worth noting that EAs typically degrade their performance when dealing with complex samples. As one of unsupervised artificial neural networks, autoencoders have been proven to be effective for realizing dimension reduction [15], [16].

Due to successful applications of autoencoders in neural network research communities [17]-[19], and, recently, EAs [20], this work proposes an Autoencoder-based Multi-swarm Gray wolf optimizer based on Genetic learning (AMGG) to efficiently solve HEPs. In AMGG, autoencoders are constructed by using data samples produced in previous generations of evolution. Since the data samples are getting closer to better solutions after several generations of evolution, the trained autoencoder can implicitly capture connections among decision variables, and efficiently learn the distribution and landscapes of solutions. At the end of a training process, AMGG can compress the original landscape

*This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 62173013 and 62073005, and the Fundamental Research Funds for the Central Universities under Grant YWF-22-L-1203.

[1]J. Bi, J. Zhai, Z. Wang and J. Qiao are with the Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China. Email: bijing@bjut.edu.cn, zhaijiahui@emails.bjut.edu.cn, wangziqi0312@163.com, junfeiq@bjut.edu.cn.

[2]H. Yuan is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China. Email: yuan@buaa.edu.cn.

[3]J. Zhang is with the Department of Computer Science in the Lyle School of Engineering at Southern Methodist University, Dallas, TX 75205, USA. Email: jiazhang@smu.edu.

[4]M. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA. Email: zhou@njit.edu.

information into a highly informative low-dimensional search space, where promising solutions are more likely to be generated than those in the original high-dimensional space.

AMGG includes a Multi-swarm Gray-wolf-optimizer based on Genetic-learning (MGG), which includes three proposed strategies to well balance exploration and exploitation abilities. The first strategy, called a Dynamic subgroup Number Strategy (DNS), divides the whole population into many small subpopulations at an early stage and periodically decreases the number of subpopulations, *i.e.*, increases the number of individuals in each subpopulation. With DNS, in early stages of evolution, more and smaller subpopulations motivate the whole population to enable more exploration. Besides, at later stages of evolution, fewer and larger subpopulations enable more exploitation. The second strategy is a Subpopulation Recombination Strategy (SRS), based on which subpopulations are recombined at each cycle of DNS by using stagnation information of global optima. The recombination in SRS provides each subpopulation access to superior information shared by other subpopulations, which is very beneficial for exploitation. The last strategy is a Purposeful Detection Strategy (PDS) in which the historical information of search processes is used to help subpopulations jump from current local optima, thereby improving the exploration ability. This work evaluates AMGG by using six benchmark HEPs and optimal task offloading problem in edge computing. Experimental results demonstrate that it outperforms its peers in terms of search accuracy and convergence efficiency.

II. PROPOSED FRAMEWORK

A. Overall Framework

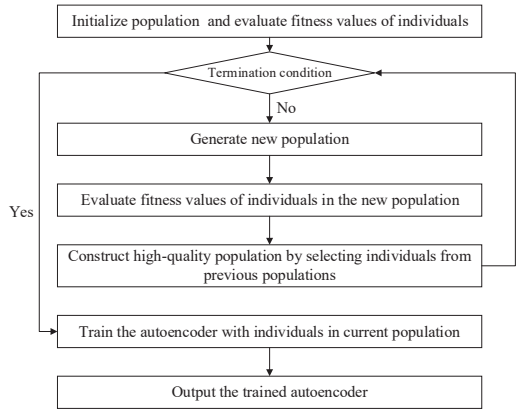


Fig. 1. Training process of the autoencoder in stage 1.

To improve the ability of EAs to deal with HEPs, this work proposes a framework of AMGG. As shown in Figs. 1 and 2, AMGG includes two stages, *i.e.*, initial population evolution and autoencoder training, and a co-evolution process of two subpopulations. In the first stage, the data samples (individuals) selected from previous populations are used to train an autoencoder. The data samples are accumulated during the first few generations of evolution. Once a predefined number of generations is reached, the autoencoder is trained by the chosen individuals and obtained, and the final population ($\mathbb{P}$) in the first stage enters the second stage. In the second stage, $\mathbb{P}$ is divided into two subpopulations according to fitness value, *i.e.*, $\mathbb{P}_1$ and $\mathbb{P}_2$. $\mathbb{P}_1$ is further encoded as $\check{\mathbb{P}}_1$ with the trained autoencoder. It is worth noting that $\mathbb{P}_1$ and $\check{\mathbb{P}}_1$ are subpopulations with original and lower dimensions, respectively. Then, $\check{\mathbb{P}}_1$ is adopted as an initial subpopulation, which evolves with MGG to yield $\check{\mathbb{P}}_1^{'}$ in the low-dimensional search space. Next, $\check{\mathbb{P}}_1^{'}$ is decoded to $\mathbb{P}_1^{'}$ in the original search space. Meanwhile, $\mathbb{P}_2$ directly evolves with in the original space. MGG mainly consists of three proposed strategies to balance exploration and exploitation abilities. In this way, MGG is derived based on a multi-swarm framework that cooperates with DNS, SRS, and PDS, respectively.

Based on fitness values of individuals in $\mathbb{P}_1$ and $\mathbb{P}_2$, elite individuals of both subpopulations are selected and formed as a new population $\mathbb{P}$ used in the next generation. If a termination condition is not met, $\mathbb{P}$ is divided again, and the information between $\mathbb{P}_1$ and $\mathbb{P}_2$ is exchanged dynamically. In this way, the whole population is updated with AMGG to avoid trapping into local optima and finding global ones.

B. Training of the Autoencoder

At the first stage, data samples are kept through several generations of evolution. Better solutions are obtained as individuals evolve, and therefore, the trained autoencoder has the increasing probability of obtaining the compressed representation of a region closer to global optima. The autoencoder is adopted to explore search spaces of HEPs and capture connections among decision variables. The pseudo codes for training the autoencoder are given in Algorithm 1.

Algorithm 1 Training of the Autoencoder

```
Initialize ℙ
Initialize d, a, A and C
Evaluate the fitness value f(x_i) of each x_i in ℙ
Ω ← ∅
Initialize x_α, x_β and x_δ
g_1 ← 1
repeat
    ℙ' ← MGG(ℙ)
    Evaluate the fitness value f(x_i') of each x_i' in ℙ'
    for each individual i in ℙ do
        if f(x_i') < f(x_i) then
            x_i ← x_i'
        end if
    end for
    Update ℙ
    Ω ← Ω ∪ ℙ
    g_1 ← g_1 + 1
until g_1 ≤ ĝ_1
ℸ ← trainAutoencoder(Ω, ĝ_1, d)
return ℸ
```

In Algorithm 1, Ω denotes a database for archiving

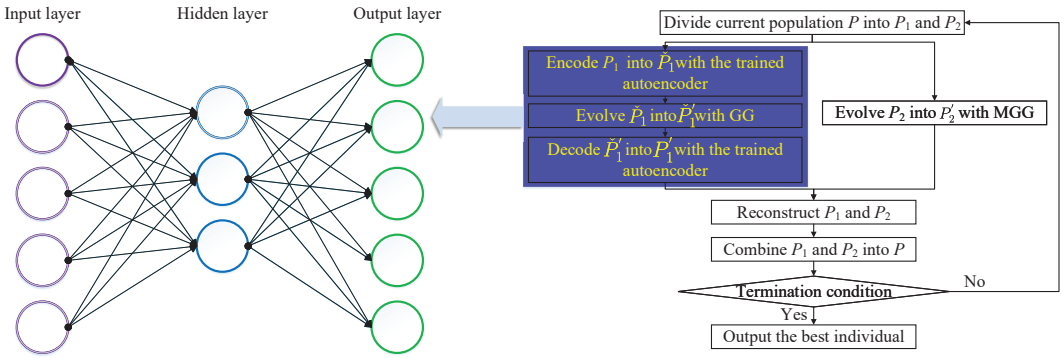


Fig. 2. Multi-swarm coevolution in stage 2.

selected data samples in previous populations. **trainAutoencoder** is a function in the Matlab Deep Learning Toolbox [21]. d is a user-defined parameter indicating the dimensionality of latent representation. A sigmoid function is adopted, and a mean square error is adopted as a loss function in the encoding and decoding operations of the autoencoder.

C. Multi-swarm Gray-wolf-optimizer based on Genetic-learning (MGG)

In MGG, the entire population is divided into many equal subpopulations during initial evolutionary generations. Each subpopulation adopts its own individuals to search for better regions in parallel with other subpopulations. This mechanism motivates more exploration. As the evolution process proceeds, DNS is adopted to decrease the number of subpopulations. SRS is adopted to share useful information about a subpopulation during the search process, thereby further improving the exploitation efficiency. Besides, when each subpopulation is trapped in a local optimum, PDS is used to help it jump out of the current local optimum, thereby improving MGG's exploration capability.

(1) Dynamic-subpopulation Number Strategy (DNS)

At the beginning of the evolutionary process, the entire population is divided into many smaller subpopulations. The number of subpopulations decreases accordingly, and the size of each subpopulation gradually increases as evolutions progress. At last, the number of subpopulations is reduced to one, which means all subpopulations are merged into one population. In DNS, many small subpopulations maintain the population diversity during the initial evolutionary generations because the information spreads slowly within a smaller-sized area, and many parallel evolving subpopulations facilitate exploration. Conversely, gradual reduction in the number of subpopulations is beneficial for exploitation in the later evolutionary generations. Thus, DNS pushes search from exploration to exploitation in the evolutionary process.

In DNS, we need to determine the number of subpopulations, and when to adjust them. An ordered set of integers $\mathbb{N}=\{n_1,n_2,\cdots,n_{k-1},n_k\}$, where $n_1>n_2>\cdots>n_{k-1}>n_k$ is defined. Each element in $\mathbb{N}$ denotes the number of individuals of a subpopulation. In this work, all subpopulations have the same size, which means that the number of subpopulations must be a factor in the population size. For example, if population size is 30, the number of subpopulations ($N^\diamond$) is selected from $\mathbb{N}=\{15,10,6,5,3,2,1\}$. Thus, the size ($\rho^\diamond$) of the first subpopulation is 2, *i.e.*, $\frac{\|\mathbb{N}\|}{N^\diamond}=2$. At the final generations of evolution, $N^\diamond=1$ and $\rho^\diamond=30$, respectively, *i.e.*, all subpopulations are merged into a single population. We thus adjust the number of subpopulations every ζ FEs where $\zeta=\hat{g}/\|\mathbb{N}\|$.

(2) Sub-population Reorganization Strategy (SRS)

In dynamic multi-swarm EAs, a stochastic recombination scheme allows individuals to have a varying neighborhood structure. Moreover, population of multi-swarm EAs is randomly regrouped after a certain number (T_*) of successive generations of stagnation in $\boldsymbol{x}_*$. In MGG, T_* is chosen as the subpopulation regrouping criterion. In this case, it is possible to regroup the entire population into multiple subpopulations when $\boldsymbol{x}_*$ keeps unchanged for T_* generations. Since the neighbor structure of each subpopulation is a ring model, each individual in a larger subpopulation, including the best one, may require more generations to fully extract useful information from other individuals. The extent of the information diffusion depends on the size of each subpopulation. Thus, we set $T_*=\rho^\diamond/2$ in this work.

(3) Purposeful Detection Strategy (PDS)

To further increase the global search ability for complex multi-modal problems, this work adopts some personal historical information to guide $\boldsymbol{x}_*$ to perform a purposeful detection operator that helps the population to avoid trapping into local optima. This work divides the search space of each dimension d ($1\le d\le D$) of the problem into S small segments. $\mathscr{M}_s^d$ denotes the number of visits for segment s ($1\le s\le S$) of dimension d. $\mathscr{M}_s^d$ is used to help $\boldsymbol{x}_*$ find promising positions in each individual. $\boldsymbol{x}_i^d$ denotes the value of dimension d of $\boldsymbol{x}_i$. ξ_s^d denotes segment s of dimension d. Then we have:

$$\mathscr{M}_s^d=\mathscr{M}_s^d+1, \text{if } \boldsymbol{x}_i^d \text{ lies within } \xi_s^d \tag{1}$$

To avoid that $\boldsymbol{x}_*^d$ falls into the same segment for different times, a tabu strategy is adopted in PDS. For example, if $\boldsymbol{x}_*^d$

has falled into ξ_s^d, a binary variable t_s^d for ξ_s^d is set to 1. In this way, $\boldsymbol{x}_*^d$ does not fall into ξ_s^d unless t_s^d is reset to 0. When $\boldsymbol{x}_*^d$ has falled into all segments, *i.e.*, all t_k^d are 1, they are all reset to 0. Then, for each dimension d, if $\boldsymbol{x}_*^d$ lies within a more visited segment ξ_s^d, *i.e.*, $\mathscr{M}_s^d$ is larger than $\mathscr{M}_k^d (1 \leq k \leq S,$ and $k \neq s)$, $\boldsymbol{x}_*^d$ is replaced by a random value within a less visited segment ξ_k^d where t_k^d is 0. Then, $\boldsymbol{x}_*^{'}$ is yielded as a new version of $\boldsymbol{x}_*$, and if $f(\boldsymbol{x}_*^{'}) \leq f(\boldsymbol{x}_*)$, $\boldsymbol{x}_*$ is replaced by $\boldsymbol{x}_*^{'}$. Then, t_s^d is set to 1. $\forall\ t_k^d = 1$, reset it to 0.

Algorithm 2 MGG

1: $t_k^d \leftarrow 0$
2: $\mathscr{M}_s^d \leftarrow 0$
3: **for** $m = 1$ to $\|\mathbb{N}\|$ **do**
4: $N^{\diamond} = n_m$
5: Divide $\mathbb{P}$ into $N^{\diamond}$ subpopulations equally
6: **for** $\varpi = 1$ to $\rho^{\diamond}$ **do**
7: $\mathbb{P}_{\varpi}$ = GG($\mathbb{P}_{\varpi}$)
8: Update $\mathscr{M}_s^d$, $\boldsymbol{x}_*$, T_*
9: Perform DNS
10: Perform SRS
11: Perform PDS
12: **end for**
13: Combine all subpopulations into $\mathbb{P}$
14: **end for**

(4) Framework of MGG

By merging the above-mentioned components, MGG is realized in Algorithm 2. The details of Gray-wolf-optimizer based on Genetic-learning (GG) in Line 7 adopts genetic learning of GA to produce an additional exemplar for each individual i. GA and GWO are combined in a highly integrated way. GG includes two major processes: the first for generating exemplars with GA and the second for updating each individual in GWO. In GG, each individual is updated by $\boldsymbol{x}_{\boldsymbol{\alpha}}$, $\boldsymbol{x}_{\boldsymbol{\beta}}$ and $\boldsymbol{x}_{\boldsymbol{\delta}}$, and its corresponding exemplar.

D. AMGG

AMGG is realized in Algorithm 3, where **encode** and **decode** are Matlab functions[21].

(1) Population Splitting

Line 5 of AMGG performs population splitting to divide $\mathbb{P}$ into $\mathbb{P}_1$ and $\mathbb{P}_2$. To avoid being trapped into local optima, AMGG's current population is split into two subpopulations that evolve in a distributed and parallel manner. Individuals in $\mathbb{P}$ are ranked according to their fitness values in ascending order. The top $\|\mathbb{P}_1\|$ individuals are put into $\mathbb{P}_1$, and other $\|\mathbb{P}_2\|$ individuals are put into $\mathbb{P}_2$. The use of $\mathbb{P}_1$ aims to compress its solutions into a low-dimensional space by capturing the distribution characteristics of its solutions, thereby guiding its evolution more efficiently. The use of $\mathbb{P}_2$ aims to find optimal solutions around its current ones.

(2) Autoencoder-assisted Population Co-evolution

Line 6 in Algorithm 3 compresses each individual in $\mathbb{P}_1$ into its high-quality and low-dimensional one in $\check{\mathbb{P}}_1$ with the obtained autoencoder $\urcorner$. The lower-dimensional individuals

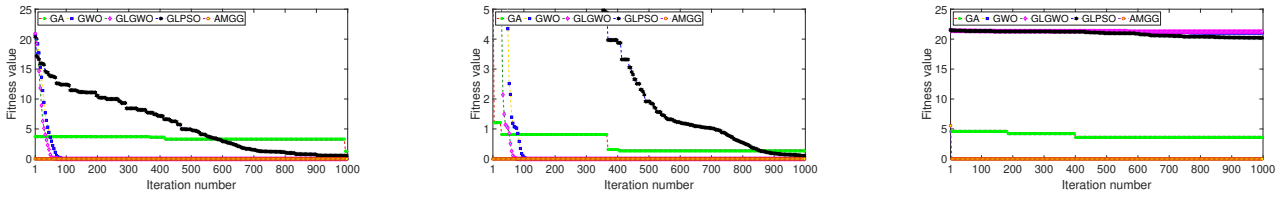
Algorithm 3 AMGG

1: Initialize $\boldsymbol{a}$, $\boldsymbol{A}$ and $\boldsymbol{C}$
2: Evaluate fitness value $f(\boldsymbol{x_i})$ of each $\boldsymbol{x_i}$ in $\mathbb{P}$
3: Determine $\boldsymbol{x_\alpha}$, $\boldsymbol{x_\beta}$ and $\boldsymbol{x_\delta}$
4: **for** $g \leftarrow 1$ to $0.9 * \hat{g}$ **do**
5: Perform population splitting
6: $\check{\mathbb{P}}_1 \leftarrow$ **encode**($\urcorner, \mathbb{P}_1$)
7: $\check{\mathbb{P}}_1^{'} \leftarrow$ **GG**($\check{\mathbb{P}}_1$)
8: $\mathbb{P}_1^{'} \leftarrow$ **decode**($\urcorner, \check{\mathbb{P}}_1^{'}$)
9: Update $f(\boldsymbol{x_i})$ of each $\boldsymbol{x_i}$ in $\mathbb{P}_1^{'}$
10: Select the best $\|\mathbb{P}_1\|$ individuals from $\mathbb{P}_1$ and $\mathbb{P}_1^{'}$ for constructing $\mathbb{P}_1$
11: $\mathbb{P}_2^{'} \leftarrow$ **MGG**($\mathbb{P}_2$)
12: Update $f(\boldsymbol{x_i})$ of each $\boldsymbol{x_i}$ in $\mathbb{P}_2^{'}$
13: Select the best $\|\mathbb{P}_2\|$ individuals from $\mathbb{P}_2$ and $\mathbb{P}_2^{'}$ for constructing $\mathbb{P}_2$
14: $\mathbb{P} \leftarrow \mathbb{P}_1 \cup \mathbb{P}_2$
15: **end for**
16: **for** $g \leftarrow 0.9 * \hat{g}$ to $\hat{g}$ **do**
17: $\mathbb{P} \leftarrow$ **MGG**($\mathbb{P}$)
18: **end for**
19: Select $\boldsymbol{x}_*$ from $\mathbb{P}$, and update its fitness value f_*
20: **return** $\boldsymbol{x}_*$

have more informative search space and help GWO to quickly generate promising offsprings, thereby improving the search performance of GWO. Then, $\check{\mathbb{P}}_1$ evolves with GG in Line 7 to avoid stagnation and yield $\check{\mathbb{P}}_1^{'}$. Due to reduced dimensions, original fitness functions cannot be directly used to evaluate $\check{\mathbb{P}}_1^{'}$. Thus, Line 8 decodes $\check{\mathbb{P}}_1^{'}$ into high-dimensional $\mathbb{P}_1^{'}$ with $\urcorner$. It is worth noting that there are errors in the construction of $\urcorner$. However, they can be viewed as the effect of uncertainty brought by $\urcorner$. Blessing of uncertainty benefits EAs and guides the population towards high-quality directions.

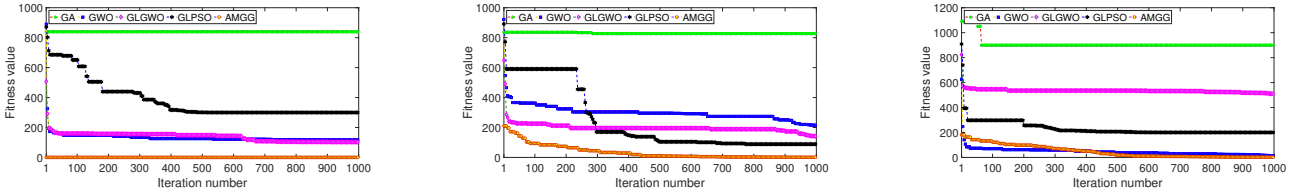
(3) Information Exchange

Line 9 in Algorithm 3 updates $f(\boldsymbol{x_i})$ of each $\boldsymbol{x_i}$ in $\mathbb{P}_1^{'}$. Line 10 selects the best $\|\mathbb{P}_1\|$ individuals from $\mathbb{P}_1$ and $\mathbb{P}_1^{'}$ for constructing $\mathbb{P}_1$. $\mathbb{P}_2$ evolves into $\mathbb{P}_2^{'}$ with MGG in Line 11. Line 12 updates $f(\boldsymbol{x_i})$ of each $\boldsymbol{x_i}$ in $\mathbb{P}_2^{'}$. Line 13 selects the best $\|\mathbb{P}_2\|$ individuals from $\mathbb{P}_2$ and $\mathbb{P}_2^{'}$ for constructing $\mathbb{P}_2$. Then, new population $\mathbb{P}$ is merged for the next generation by combing $\mathbb{P}_1$ and $\mathbb{P}_2$ in Line 14, and therefore, the information between them can be dynamically exchanged. $\mathbb{P}$ continues to evolve with MGG in Lines 16–18. Line 19 selects $\boldsymbol{x}_*$ from $\mathbb{P}$, and updates its fitness value f_*. Finally, Line 20 returns $\boldsymbol{x}_*$, which is a yielded final solution. In Algorithm 3, the execution time is mainly owning to the **for** loop, which stops after $\hat{g}$ iterations. As shown in Lines 5-–14, the complexity in each iteration of MGG is $\mathscr{O}(\|\mathrm{N}\| N^{\diamond} \hat{g}_2 D)$. Therefore, the complexity of Algorithm 3 is $\mathscr{O}(\hat{g} \|\mathrm{N}\| N^{\diamond} \hat{g}_2 D)$.



(a) F_1 (b) F_2 (c) F_3

Fig. 3. Fitness values of GA, GWO, GLGWO, GLPSO and AMGG in each iteration for F_1–F_3.



(a) F_4 (b) F_5 (c) F_6

Fig. 4. Fitness values of GA, GWO, GLGWO, GLPSO and AMGG in each iteration for F_4–F_6.

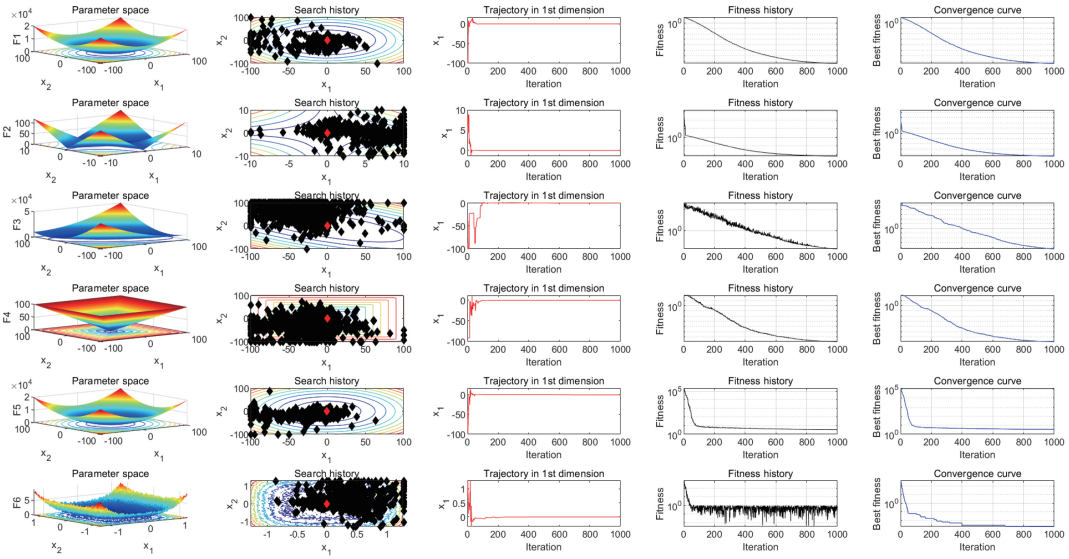
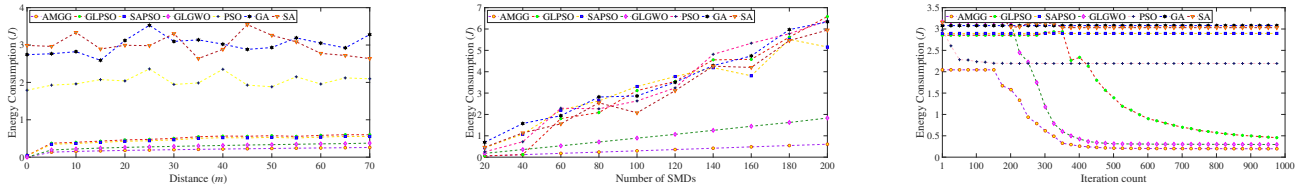


Fig. 5. 2D shapes, search histories, trajectories, fitness histories and convergence curves of F_1–F_6 with AMGG.



(a) Total energy consumed by all SMDs and edge servers with respect to different d.

(b) Total energy consumed by all SMDs and edge servers with respect to different M.

(c) Evolutionary curves of total energy consumption in each iteration.

Fig. 6. The real-life optimization problem in mobile edge computing systems.

III. PERFORMANCE EVALUATION

A. Experimental Setup

This section evaluates the proposed AMGG with typical benchmark functions [22]. Following [23]-[27], the parameters of AMGG are set as follows. In the training of Autoencoder in Algorithm 1, $\|\mathbb{P}\|$=35, $\hat{g}_1$=1000 and d=5. In addition, $\|\mathbb{P}_1\|$=5 and $\|\mathbb{P}_2\|$=30. In PDS, S=10. In MGG, $\mathbb{N}=\{15,10,6,5,3,2,1\}$. In GG, $\hat{g}_2$=1000, ω=0.04, and χ=7. In AMGG, $\hat{g}$=1000.

To comprehensively evaluate the performance of AMGG, we compare it with other algorithms with three types of

popular test suites including six benchmark functions [28]. They can be divided into multiple types including unimodal (F_1–F_2), multimodal (F_3–F_4), and composite ones (F_5–F_6). Their detailed descriptions are given in [24], [25].

AMGG is executed 30 times on each function. AMGG's efficiency and effectiveness are verified and compared with GA [26], GWO [29], genetic learning particle swarm optimization (GLPSO) [27]. We also compare it with an improved variant of GWO, *i.e.*, Genetic Learning GWO (GLGWO) [30].

B. Experimental Results

Results in Figs. 3-4 demonstrate the performance of AMGG in minimizing functions in comparison with its peers, *i.e.*, GA, GWO, GLGWO, GLPSO, and AMGG. It is shown that AMGG provides very competitive results. The reason is that it has autoencoder-based multi-swarm co-evolution with GWO, and it has three designed strategies, including DNS, SRS, and PDS, for improving global search ability in high-dimensional solution space. Thus, AMGG jointly provides avoidance of local optima, excellent exploitation, and strong exploration abilities.

C. Convergence Behavior Analysis

Convergence behavior analysis of AMGG is provided here. Following [31], significant changes are needed in individuals over the beginning iterations of optimization. It is helpful to guide a meta-heuristic algorithm to extensively explore solution space of HEPs. For example, it is shown in Fig. 5 that population of AMGG extensively finds high-quality areas of solution space and investigates the most promising one. Besides, the third subfigure for each function in Fig. 5 presents the trajectory of the first individual whose changes in the first dimension x_1 are given. It is observed that several significant changes exist in the beginning stage of iterations, and they are reduced gradually as iterations proceed. According to [31], this convergence behavior guarantees that AMGG finally converges to the optimal solution in the solution space of each HEP.

D. Case study of a real-life optimization problem

To further evaluate the actual performance of our proposed AMGG, a real-life optimization problem in mobile edge computing systems [32]-[35] is used to verify AMGG. This problem aims to minimize the total energy consumption of all smart mobile devices (SMDs) and edge servers used in a large-scale automated factory. The decision variables include task offloading ratio, computational speeds of SMDs, and data transmission power of SMDs [36].

Fig. 6(a) illustrates the total energy consumption comparison of AMGG, GLPSO, SAPSO [37], GLGWO, PSO [38], GA and SA [39] with respect to different settings of distance (d). It is shown that energy consumption increases as d increases. In addition, it is shown that the energy consumption of AMGG is the least among the seven algorithms when d varies from 0 to 70. It is obvious from Fig. 6(a) that the effect of increasing energy consumption of AMGG with d is the smallest, which indicates that the optimization efficiency and stability of AMGG are the best compared with other algorithms. Fig. 6(b) shows the energy consumption comparison of AMGG, GLPSO, SAPSO, GLGWO, PSO, GA, and SA with respect to different numbers of SMDs (M), respectively. It is shown that the energy consumption of AMGG is the least among the seven algorithms when M varies from 20 to 200. In addition, it is shown that the energy consumption of AMGG increases nearly linearly, and its reduction of energy consumption is the largest compared with other algorithms as M increases.

Fig. 6(c) shows the total energy consumption of AMGG, GLPSO, SAPSO, GLGWO, PSO, GA, and SA when they are adopted to solve the real-life optimization problem, respectively. It is clearly shown that the total energy consumption of AMGG is the least among the seven algorithms in each iteration. Specifically, the final total energy consumption of AMGG is 0.20 J, which is lower than that (0.30 J) of GLGWO and that (0.45 J) of GLPSO, respectively. Furthermore, AMGG only needs 952 iterations to converge to its final fitness value, while GLGWO and GLPSO need 971 and 973 iterations to converge to their final ones, respectively. Thus, AMGG achieves the least energy consumption in much fewer iterations than GLGWO and GLPSO, which verifies the performance of AMGG. The reason is that AMGG uses such strategies as DNS, SRS, and PDS, and autoencoder-assisted multi-swarm coevolution.

IV. CONCLUSIONS

There are many high-dimensional expensive problems in complex robotic systems. They often have a complicated time-consuming process of fitness evaluation, which brings a big challenge to yield promising solutions in a high-dimensional search space. To handle this problem, this work proposes an evolutionary optimization framework with an embedded autoencoder that significantly advances a field of high-dimensional expensive problems (HEPs). To generate promising offsprings of HEPs, this work takes advantage of the autoencoder for dimension reduction and embeds it into a gray wolf optimizer (GWO) for the first time. Our newly proposed strategies, *i.e.*, Dynamic subgroup Number Strategy (DNS), Subpopulation Reorganization Strategy (SRS) for sharing useful information of different subpopulations, and Purposeful Detection Strategy (PDS), are effective in improving the comprehensive performance of Multi-swarm GWO based on Genetic-learning (MGG). A multi-swarm framework named Autoencoder-based MGG (AMGG) is evaluated by using three types of popular test suites including typical benchmark functions and a real-life optimization problem. Experimental results demonstrate that AMGG outperforms such peers as genetic algorithm, GWO, genetic learning GWO, and genetic learning particle swarm optimization in terms of global search capability and accuracy, local optima avoidance, and robustness. Its parameter tuning and selection need future work by using Taguchi's experimental design method, grid search and other methods [40].

REFERENCES

[1] H. Yuan, J. Bi, J. Zhang and M. Zhou, "Energy Consumption and Performance Optimized Task Scheduling in Distributed Data Centers," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 9, pp. 5506–5517, Sept. 2022.

[2] H. Yuan, J. Bi and M. Zhou, "Energy-Efficient and QoS-Optimized Adaptive Task Scheduling and Management in Clouds," *IEEE Trans. on Automation Science and Engineering*, vol. 19, no. 2, pp. 1233–1244, Apr. 2022.

[3] J. Bi, H. Yuan, K. Xu, H. Ma and M. Zhou, "Large-scale Network Traffic Prediction With LSTM and Temporal Convolutional Networks," *2022 International Conference on Robotics and Automation*, Philadelphia, PA, USA, 2022, pp. 3865–3870.

[4] S. Tuli, S. S. Gill, P. Garraghan, R. Buyya, G. Casale and N. R. Jennings, "START: Straggler Prediction and Mitigation for Cloud Computing Environments Using Encoder LSTM Networks," *IEEE Trans. on Services Computing*, vol. 16, no. 1, pp. 615–627, Feb. 2023.

[5] H. Yuan, J. Bi, M. Zhou, Q. Liu and A. C. Ammari, "Biobjective Task Scheduling for Distributed Green Data Centers," *IEEE Trans. on Automation Science and Engineering*, vol. 18, no. 2, pp. 731–742, Apr. 2021.

[6] H. Yuan, J. Bi and M. Zhou, "Multiqueue Scheduling of Heterogeneous Tasks With Bounded Response Time in Hybrid Green IaaS Clouds," *IEEE Trans. on Industrial Informatics*, vol. 15, no. 10, pp. 5404–5412, Oct. 2019.

[7] A. Favaro, *etc.*, "An Evolutionary-Optimized Surgical Path Planner for a Programmable Bevel-Tip Needle," *IEEE Trans. on Robotics*, vol. 37, no. 4, pp. 1039–1050, Aug. 2021.

[8] X. Guo, M. Zhou, S. Liu and L. Qi, "Multiresource-Constrained Selective Disassembly With Maximal Profit and Minimal Energy Consumption," *IEEE Trans. on Automation Science and Engineering*, vol. 18, no. 2, pp. 804–816, Apr. 2021.

[9] R. Patel, E. Rudnick-Cohen, S. Azarm, M. Otte, H. Xu and J. W. Herrmann, "Decentralized Task Allocation in Multi-Agent Systems Using a Decentralized Genetic Algorithm," *2020 IEEE International Conference on Robotics and Automation*, 2020, pp. 3770–3776.

[10] S. Shan and G. G. Wang, "Survey of Modeling and Optimization Strategies to Solve High-Dimensional Design Problems with Computationally Expensive Black-Box Functions," *Structural and Multidisciplinary Optimization*, vol. 41, no. 2, pp. 219–241, Mar. 2010.

[11] T. W. Simpson, A. J. Booker, D. Ghosh, A. A. Giunta, P. N. Koch,and R. J. Yang, "Approximation Methods in Multidisciplinary Analysis and Optimization: A Panel Discussion," *Structural and Multidisciplinary Optimization*, vol. 27, no. 5, pp. 302–313, Jun. 2004.

[12] Y. Jin, H. Wang, T. Chugh, D. Guo, and K. Miettinen, "Data-Driven Evolutionary Optimization: An Overview and Case Studies," *IEEE Trans. on Evolutionary Computation*, vol. 23, no. 3, pp. 442–458, Jun. 2019.

[13] T. Sonoda and M. Nakata, "Multiple Classifiers-Assisted Evolutionary Algorithm Based on Decomposition for High-Dimensional Multiobjective Problems," *IEEE Trans. on Evolutionary Computation*, vol. 26, no. 6, pp. 1581–1595, Dec. 2022.

[14] R. G. Regis, "Evolutionary Programming for High-Dimensional Constrained Expensive Black-Box Optimization Using Radial Basis Functions," *IEEE Trans. on Evolutionary Computation*, vol. 18, no. 3, pp. 326–347, Jun. 2014.

[15] G. E. Hinton and R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[16] Y. Yoo, C. Y. Lee and B. T. Zhang, "Multimodal Anomaly Detection based on Deep Auto-Encoder for Object Slip Perception of Mobile Manipulation Robots," *2021 IEEE International Conference on Robotics and Automation*, 2021, pp. 11443–11449.

[17] C. Rolinat, M. Grossard, S. Aloui and C. Godin, "Human Initiated Grasp Space Exploration Algorithm for an Underactuated Robot Gripper Using Variational Autoencoder," *2021 IEEE International Conference on Robotics and Automation*, 2021, pp. 2598–2604.

[18] S. Ilager, K. Ramamohanarao and R. Buyya, "Thermal Prediction for Efficient Energy Management of Clouds Using Machine Learning," *IEEE Trans. on Parallel and Distributed Systems*, vol. 32, no. 5, pp. 1044–1056, May 2021.

[19] D. Saxena, A. K. Singh and R. Buyya, "OP-MLB: An Online VM Prediction-Based Multi-Objective Load Balancing Framework for Resource Management at Cloud Data Center," *IEEE Trans. on Cloud Computing*, vol. 10, no. 4, pp. 2804–2816, Dec. 2022.

[20] M. Cui, L. Li, M. Zhou, J. Li, A. Abusorrah and K. Sedraoui, "A Bi-population Cooperative Optimization Algorithm Assisted by an Autoencoder for Medium-scale Expensive Problems," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 11, pp. 1952–1966, November 2022.

[21] M. Beale, M. Hagan, and H. Demuth, "Matlab Deep Learning *toolbox*^{TM} User's Guide: PDF Documentation for Release r2019a," *The MathWorks, Inc, 2019*.

[22] I. U. Rahman, Z. Wang, W. Liu, B. Ye, M. Zakarya and X. Liu, "An N-State Markovian Jumping Particle Swarm Optimization Algorithm," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 11, pp. 6626–6638, Nov. 2021.

[23] M. Cui, L. Li, M. Zhou and A. Abusorrah, "Surrogate-Assisted Autoencoder-Embedded Evolutionary Optimization Algorithm to Solve High-Dimensional Expensive Problems," *IEEE Trans. on Evolutionary Computation*, vol. 26, no. 4, pp. 676–689, Aug. 2022.

[24] J. Bi, H. Yuan, J. Zhai, M. Zhou and H. V. Poor, "Self-adaptive Bat Algorithm With Genetic Operations," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 7, pp. 1284–1294, Jul. 2022.

[25] A. S. Nandan, S. Singh, R. Kumar and N. Kumar, "An Optimized Genetic Algorithm for Cluster Head Election Based on Movable Sinks and Adjustable Sensing Ranges in IoT-Based HWSNs," *IEEE Internet of Things Journal*, vol. 9, no. 7, pp. 5027–5039, Apr. 2022.

[26] J. Wang, M. Zhou, X. Guo and L. Qi, "Multiperiod Asset Allocation Considering Dynamic Loss Aversion Behavior of Investors," *IEEE Trans. on Computational Social Systems*, vol. 6, no. 1, pp. 73–81, Feb. 2019.

[27] Y. J. Gong, J. J. Li, Y. Zhou, Y. Li, H. S. H. Chung, Y. H. Shi and J. Zhang, "Genetic Learning Particle Swarm Optimization," *IEEE Trans. on Cybernetics*, vol. 46, no. 10, pp. 2277–2290, Oct. 2016.

[28] S. Mirjalili, and A. Lewis, "S-shaped Versus V-Shaped Transfer Functions for Binary Particle Swarm Optimization," *Swarm and Evolutionary Computation*, vol. 9, pp. 1–14, Apr. 2013.

[29] S. Mirjalili, S. M. Mirjalili and A. Lewis, "Grey Wolf Optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, Mar. 2014.

[30] E. Daniel, "Optimum Wavelet-Based Homomorphic Medical Image Fusion Using Hybrid Genetic–Grey Wolf Optimization Algorithm," *IEEE Sensors Journal*, vol. 18, no. 16, pp. 6804–6811, Aug. 2018.

[31] F. Bergh and A. P. Engelbrecht, "A Study of Particle Swarm Optimization Particle Trajectories," *Information sciences*, vol. 176, no. 8, pp. 937–971, Apr. 2006.

[32] Y. Wang, M. Sheng, X. Wang, L. Wang and J. Li, "Mobile-Edge Computing: Partial Computation Offloading Using Dynamic Voltage Scaling," *IEEE Trans. on Communications*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[33] L. Ale, S. A. King, N. Zhang, A. R. Sattar and J. Skandaraniyam, "D3PG: Dirichlet DDPG for Task Partitioning and Offloading With Constrained Hybrid Action Space in Mobile-Edge Computing," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 19260–19272, Oct. 2022.

[34] H. Yuan and M. Zhou, "Profit-Maximized Collaborative Computation Offloading and Resource Allocation in Distributed Cloud and Edge Computing Systems," *IEEE Trans. on Automation Science and Engineering*, vol. 18, no. 3, pp. 1277–1287, Jul. 2021.

[35] Y. Sahni, J. Cao, L. Yang and Y. Ji, "Multihop Offloading of Multiple DAG Tasks in Collaborative Edge Computing," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4893–4905, Mar. 2021.

[36] T. K. Rodrigues, J. Liu and N. Kato, "Application of Cybertwin for Offloading in Mobile Multiaccess Edge Computing for 6G Networks," *IEEE Internet of Things Journal*, vol. 8, no. 22, pp. 16231–16242, Nov. 2021.

[37] H. Yuan, J. Bi,W. Tan, M. Zhou, B. H. Li and J. Li, "TTSA: An Effective Scheduling Approach for Delay Bounded Tasks in Hybrid Clouds," *IEEE Trans. on Cybernetics*, vol. 47, no. 11, pp. 3658–3668, Nov. 2017.

[38] Y. Cao, H. Zhang, W. Li, M. Zhou, Y. Zhang and W. A. Chaovalitwongse, "Comprehensive Learning Particle Swarm Optimization Algorithm With Local Search for Multimodal Functions," *IEEE Trans. on Evolutionary Computation*, vol. 23, no. 4, pp. 718–731, Aug. 2019.

[39] S. Lyden and M. E. Haque, "A Simulated Annealing Global Maximum Power Point Tracking Approach for PV Modules under Partial Shading Conditions," *IEEE Trans. on Power Electronics*, vol. 31, no. 6, pp. 4171–4181, Jun. 2016.

[40] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi and J. Wang, "Dendritic Neuron Model With Effective Learning Algorithms for Classification, Approximation, and Prediction," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 30, no. 2, pp. 601-614, Feb. 2019.