TẠP CHÍ KHOA HỌC & CÔNG NGHỆ ĐẠI HỌC DUY TÂN
DTU Journal of Science and Technology 1(50) (2022) 3-7

TRƯỜNG ĐẠI HỌC DUY TÂN
DUYTAN UNIVERSITY

Constrained optimization using swarm intelligence integrated with Deb's feasibility rules developed in Python

Giải bài toán tối ưu hóa ràng buộc sử dụng trí tuệ bầy đàn kết hợp quy tắc khả thi của Deb được phát triển bằng Python

Hoang Nhat Duc[a,b*], Tran Xuan Linh[a,b], Tran Van Duc[b,c]
Hoàng Nhật Đức[a,b*], Trần Xuân Linh[a,b], Trần Văn Đức[b,c]

[a]*Institute of Research and Development, Duy Tan University, Da Nang, 550000, Vietnam*
[a]*Viện Nghiên cứu và Phát triển Công nghệ Cao, Trường Đại học Duy Tân, Đà Nẵng*
[b]*Faculty of Civil Engineering, Duy Tan University, Da Nang, 550000, Vietnam*
[b]*Khoa Xây dựng, Trường Đại học Duy Tân, Đà Nẵng*
[c]*International School, Duy Tan University, Da Nang, 550000, Vietnam*
[c]*Viện Đào tạo Quốc tế, Trường Đại học Duy Tân, Đà Nẵng*

(Ngày nhận bài: 02/11/2021, ngày phản biện xong: 11/12/2021, ngày chấp nhận đăng: 18/01/2022)

Abstract

This study utilizes Particle Swarm Optimization (PSO) and Deb's feasibility rules to construct a method for constrained optimization named as frPSO. PSO is a capable swarm intelligence based metaheuristic and Deb's feasibility rules is an effective constraint-handling method. This integrated tool is developed in Python. frPSO has been tested with three basic constrained optimization problems. Experimental results point out that frPSO is a powerful tool for solving complex design tasks.

Keywords: Swarm intelligence; Particle Swarm Optimization; Constrained optimization; Feasibility rules; Metaheuristic.

Tóm tắt

Nghiên cứu của chúng tôi sử dụng thuật toán tối ưu hóa bầy đàn (PSO) và các quy tắc khả thi của Deb để xây dựng một công cụ giải các bài toán tối ưu hóa có ràng buộc. Phương pháp này được đặt tên là frPSO. PSO là một thuật toán tối ưu hóa mạnh dựa trên trí tuệ bầy đàn và các quy tắc khả thi của Deb là một phương pháp xử lý ràng buộc có hiệu quả cao. Công cụ kết hợp này đã được chúng tôi xây dựng bằng Python. frPSO đã được thử nghiệm với 3 bài toán tối ưu hóa có ràng buộc cơ bản. Kết quả tính toán cho thấy frPSO là một công cụ mạnh để giải các bài toán thiết kế phức tạp.

Từ khóa: Trí tuệ bầy đàn; Tối ưu hóa bầy đàn; Tối ưu hóa có ràng buộc; Các quy tắc khả thi; Thuật toán tìm kiếm.

1. Introduction

Civil engineers are frequently encountered complex design problems in which an objective function is either minimized or maximized and a set of constraints must be satisfied [1-3]. These design problems are formulated as constrained optimization problem involves multiple decision variables and constraints. Due to the complexity of the tasks, researchers and practitioners have increasingly resorted to

**Corresponding Author:* Hoang Nhat Duc, Faculty of Civil Engineering, Duy Tan University, 55000, Danang, Vietnam; Institute of Research and Development, Duy Tan University, 55000, Danang, Vietnam
Email: hoangnhatduc@duytan.edu.vn

metaheuristic due to their versatility, general-purpose capability, and effective searching performance [4].

To handle constraints, conventional methods often rely on penalty based approaches for modifying objective functions of the optimization problem [5, 6]. Despite being simple to implement, the penalty based approaches suffer from the difficulty of selecting appropriate penalty coefficients and how to adapt these coefficient throughout the searching process. To correct such issue, Deb [7] proposes a set of feasibility rules used for comparing solutions found during the optimization process. The advantage of this method is that it is free from the challenge of selecting the penalty coefficients.

In this study, an optimization model based on the Deb's feasibility rules and the Particle Swarm Optimization (PSO) [8], named as frPSO, is developed in Python. We select the PSO because it is a highly effective nature-inspired metaheuristic and this search engine is appropriate for solving complex optimization tasks involving a large number of decision variables [9-11]. Python is selected because it is an interpreted high-level general-purpose programming language that has the advantages of effectiveness, simplicity, and code readability.

2. Methodology

Similar to other population based metaheuristic, PSO first creates a swarm of *S* individual within the boundary of the searched space. When an objective function and a set of constraints are specified, the algorithm computes the objective function and constraint functions' values. In Python, we can employ a function that returns two outputs to compute the objective function value and the constraints' values. Notably, to evaluate the feasibility of a solution, the constraint violation degree $\phi(x)$ is calculated as follows:

$$\phi(x) = \sum_j |\min_j(0, g_j(x))| + \sum_j \max_j |h_j(x)| \quad (1)$$

Herein, we rely on the following feasibility rules to compare the quality of solutions as follows [7]:

1. Among one feasible solution and one infeasible solution, the feasible solution wins.

2. Among two feasible solutions, the one having lower objective function value is selected.

3. Among two infeasible solutions, the one having smaller degree of constraint violation is considered to be the winner.

Based on such rules, the formulation of the cost function is stated as follows:

$$F(X) = \begin{cases} F(X) \ \ if \ g_j(x) \geq 0 \ \ \forall j \\ f_{\max} + \sum_{j=1}^{m} g_j(x) \end{cases} \quad (2)$$

where f_{max} denotes the objective function value of the worst solution in the set of feasible ones.

The basic operators of frPSO is demonstrated in **Fig. 1** that includes (i) Compute constraint violation, (ii) Revise objective function, (iii) Update particle velocity, and (iv) Update local and global best. It is noted that when a feasible solution is newly found, it is required to update the *fmax* parameter as shown in **Eq. 2**.

```
Fmax = 10**10
for i in range(PS):
    Pop_i = Pop[i, :]
    Fval[i], G_i = ObjFun(Pop_i)
    Nc = G_i.shape[0] # number of constraints
    Sum_Constr_Vio_i = 0
    for k in range(Nc):
        if G_i[k] < 0:
            Sum_Constr_Vio_i = Sum_Constr_Vio_i + abs(G_i[k])
    Phi[i] = Sum_Constr_Vio_i
```

(a)

```
if np.sum(Phi==0) > 0:
    Fmax = 0
    for i in range(PS):
        if Phi[i] == 0:
            if  Fval[i] > Fmax:
                Fmax = Fval[i]

for i in range(PS):
    if Phi[i] > 0:
        Fval[i] = Fmax + Phi[i]
```

(b)

```
for i in range(PS):
    # Update velocity
    for d in range(D):
        r1 = rn.random()
        r2 = rn.random()
        V[i, d] = K*(V[i, d] +
                     c1*r1*(Local_Best_Pop[i,d]-Pop[i,d]) +
                     c2*r2*(Global_Best_Sol[d]-Pop[i,d]))
```

(c)

```
# Update local best
if Fval[i] < Local_Best_Fval[i]:
    Local_Best_Fval[i] = Fval[i]
    Local_Best_Phi[i]  = Phi[i]
    Local_Best_Pop[i,:] = np.copy(Pop[i, :])

# Update global best
if Fval[i] < Global_Best_Fval:
    Global_Best_Fval = Fval[i]
    Global_Best_Phi  = Phi[i]
    Global_Best_Sol = np.copy(Pop[i, :])
```

(d)

Fig. 1 frPSO coded in Python: (a) Compute constraint violation, (b) Revise objective function, (c) Update particle velocity, and (d) Update local and global best

3. Experimental results

The first application of the frPSO method is to solve the following optimization problem [7]:

$$\begin{aligned} \text{Min. } & f(x) = (x_0^2 + x_1 - 11)^2 + (x_0 + x_1^2 - 7)^2 \\ \text{s.t } & g_0(x) = 4.84 - (x_0 - 0.05)^2 + (x_1^2 - 2.5)^2 \\ & g_1(x) = x_0^2 + (x_1^2 - 2.5)^2 - 4.84 \\ & 0 \le x_0, x_1 \le 6 \end{aligned} \quad (3)$$

The second problem is to design a bar of different cross-sections subjected to a tensile force F = 50kN (refer to **Fig. 2**). The design variables are lengths (L_1, L_2, and L_3) and diameters (D_1, D_2, and D_3) of three sections of the bar. The cost function is the total volume of the bar. In summary, this problem has three constraints involving the stress in each section and the total elongation of the whole system.

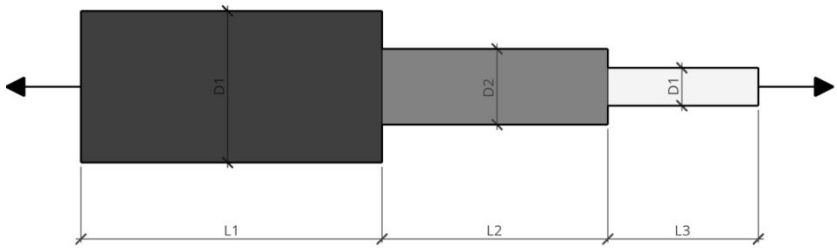


Fig. 2 Illustration of optimization second problem

This problem is mathematically formulated as follows:

$$\text{Min } f = \sum_{i=1}^{3} L_i \times D_i$$

$$\begin{aligned} \text{s.t. } & G_1(x) = 18 - F/A_1 \ge 0 \\ & G_2(x) = 100 - F/A_2 \ge 0 \\ & G_3(x) = 500 - F/A_3 \ge 0 \\ & G_4(x) = 0.1 - \left(\frac{FL_1}{A_1 E} + \frac{FL_2}{A_2 E} + \frac{FL_3}{A_3 E}\right) \ge 0 \end{aligned} \quad (4)$$

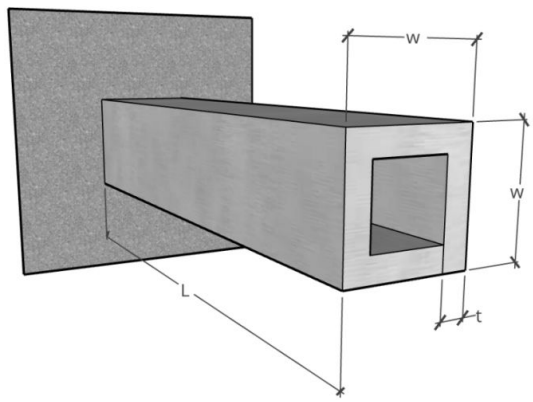


Fig. 3 Illustration of optimization third problem

In the third application, a cantilever beam with hollow cross section under the constraints on bending stress, shear stress, and vertical deflection of the free end [3, 12] is consider (refer to **Fig. 3**). This optimization task is formulated as follows:

$$\text{Min. } f = A = 4t(w-t)\,(\text{mm}^2)$$

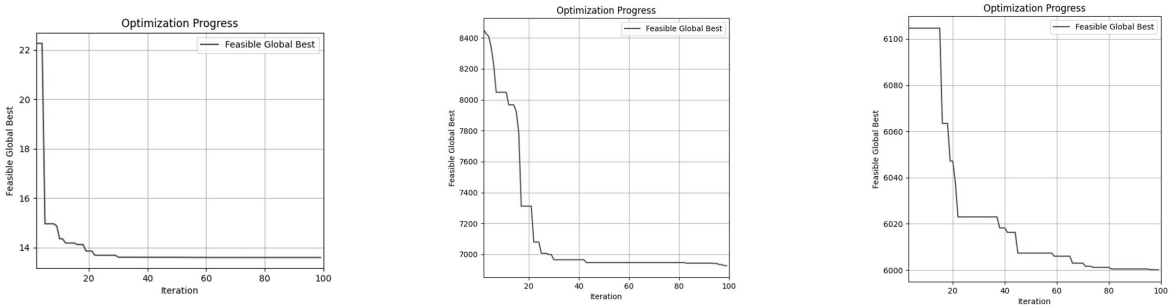
$$\text{s.t.} \quad \begin{aligned} &\sigma = Mw/(2I) \le \sigma_a \\ &\tau = VQ/(2It) \le \tau_a \\ &q = PL^3/(3EI) \le q_a \\ &8 - w/t \ge 0 \end{aligned} \tag{5}$$

where σ, τ, and q_a are bending stress, shear stress, and vertical deflection of the free end, respectively. The variables with subscript 'a' denote the allowable values. σ_a = 165N/mm^2. τ_a = 90N/mm^2. q_a = 10mm. E is the modulus of elasticity of steel = 21x10^{4}N/mm^2. $M = PL$ denotes the bending moment (N.mm). I and Q denote the moment of inertia and moment about the neural axis (NA) of the area above the NA, respectively [3]:

$$I = \frac{w^4}{12} - \frac{(w-2t)^4}{12} \tag{6}$$

$$Q = \frac{w^3}{8} - \frac{(w-2t)^3}{8} \tag{7}$$

For the 1st problem, the frPSO with a swarm size = 20 and 100 searching iterations has found the global best X = [2.24, 2.38] and the cost function $f(X)$ = 13.59. For the 2nd problem, the frPSO with a similar parameter setting has found the global best X = [80, 61, 41, 60, 26, 15.] and the cost function $f(X)$ = 6926.95. The global best and its cost function of the 3rd problem are X = [117.18 14.64] and 6000.14, respectively. It is noted that all of the problem constraints have been satisfied. The convergence rate of the frPSO for the three optimization problems is demonstrated in **Fig. 4**.



(a) (b) (c)

Fig. 4 Searching progress of the three problems: (a) Problem 1, (b) Problem 2, and (c) Problem 3

4. Conclusion

This study has developed and tested a metaheuristic based approach based on the PSO metaheuristic and the Deb's feasibility rules for constraint handling. This integrated approach has been coded in Python to facilitate its implementation and construction of optimization models. The model, named as frPSO, has been applied to solve three basic constrained optimization tasks including two structure design cases. Experimental results indicate that the frPSO is capable to find good candidate solution featuring desirable cost function value and satisfaction of constraints.

References

[1] S. M. Nigdeli, G. Bekdaş, and X.-S. Yang, "Metaheuristic Optimization of Reinforced Concrete Footings," *KSCE Journal of Civil Engineering,* vol. 22, pp. 4555-4563, November 01 2018.

[2] N. D. Hoang, "FR-DE Excel Solver: Differential Evolution with Deb's feasibility rules for solving constrained optimization problems in civil engineering," *DTU Journal of Science and Technology 04 (35),* 2019.

[3] J. S. Arora, *Introduction to Optimum Design, Fourth Edition*: Academic Press, 2016.

[4] A. Q. H. Badar, "Evolutionary Optimization Algorithms," *CRC Press,* 2021.

[5] C. A. Coello Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering,* vol. 191, pp. 1245-1287, 2002/01/04/ 2002.

[6] O. Kramer, "A Review of Constraint-Handling Techniques for Evolution Strategies," *Applied Computational Intelligence and Soft Computing,* vol. 2010, 2010.

[7] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering,* vol. 186, pp. 311-338, 2000/06/09/ 2000.

[8] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, 1995, pp. 1942-1948 vol.4.

[9] A. P. Piotrowski, J. J. Napiorkowski, and A. E. Piotrowska, "Population size in Particle Swarm Optimization," *Swarm and Evolutionary Computation,* vol. 58, p. 100718, 2020/11/01/ 2020.

[10] P.-T. Ngo, N.-D. Hoang, B. Pradhan, Q. Nguyen, X. Tran, Q. Nguyen*, et al.*, "A Novel Hybrid Swarm Optimized Multilayer Neural Network for Spatial Prediction of Flash Floods in Tropical Areas Using Sentinel-1 SAR Imagery and Geospatial Data," *Sensors,* vol. 18, p. 3704, 2018.

[11] D. Tien Bui, V.-H. Nhu, and N.-D. Hoang, "Prediction of soil compression coefficient for urban housing project using novel integration machine learning approach of swarm intelligence and Multi-layer Perceptron Neural Network," *Advanced Engineering Informatics,* vol. 38, pp. 593-604, 2018/10/01/ 2018.

[12] J. M. Gere and B. J. Goodno, *Mechanics of Materials, SI Edition*: Cengage Learning, 2013.