EasyChair Preprint

№ 12642

k-NNC: a Simple Classification Model for Reducing Computational Volumn in Metaheuristic Optimization

Hoang-Anh Pham and Manh-Hung Ha

March 20, 2024

HỘI NGHỊ CƠ HỌC TOÀN QUỐC
Kỷ niệm 45 năm thành lập Viện Cơ học, Hà Nội, 09/04/2024

k-NNC: Một mô hình phân loại đơn giản giúp giảm hiệu quả khối lượng tính toán trong tối ưu hóa bằng metaheuristic

Phạm Hoàng Anh[1,2*] và Hà Mạnh Hùng[1]
[1]*Bộ môn Cơ học kết cấu, Trường Đại học Xây dựng Hà Nội*
[2]*Nhóm nghiên cứu Cơ học Vật liệu và Kết cấu tiên tiến, Trường Đại học Xây dựng Hà Nội*
*Email: anhph2@huce.edu.vn

Tóm tắt: Những năm gần đây, các thuật toán tối ưu metaheuristic (MH) ngày càng được áp dụng phổ biến trong các tính toán thiết kế tối ưu kỹ thuật do khả năng tìm kiếm toàn cục và giải quyết được các bài toán với số lượng lớn biến thiết kế. Tuy nhiên, nhược điểm của các phương pháp tối ưu dùng MH là khối lượng tính toán lớn do MH thường yêu cầu hàng ngàn lần tính hàm mục tiêu và các ràng buộc. Mới đây, phương pháp so sánh k láng giềng gần nhất (k-nearest neighbor comparison, k-NNC) đã được đề xuất nhằm giảm chi phí tính toán khi thực hiện tối ưu bằng MH. k-NNC xem xét một giải pháp thiết kế mới thông qua so sánh k thiết kế gần nhất hiện có của nó (k-láng giềng gần nhất) với một thiết kế khác trong quần thể. Thiết kế mới sẽ bị loại bỏ mà không cần thực hiện đánh giá nếu phần lớn trong số k thiết kế lân cận gần nhất là kém hơn thiết kế được so sánh. k-NNC đã được kết hợp với các thuật toán Rao để tối ưu trọng lượng kết cấu dàn. Như được chỉ ra qua các ví dụ số, k-NNC giúp giảm đáng kể số lần phân tích kết cấu. Ở báo cáo này, khả năng của k-NNC được khẳng định khi kết hợp với một số thuật toán MH thông dụng khác như tiến hóa vi phân và Jaya. Kết quả khi áp dụng vào một số bài toán tối ưu kỹ thuật đã chứng minh k-NNC là mô hình đơn giản và hiệu quả để tiết kiệm chi phí tính toán cho MH.

Từ khóa: k-NNC, metaheuristic, thiết kế tối ưu, mô hình phân loại

1. Mở đầu

Trong những năm gần đây, các thuật toán tối ưu hóa metaheuristic (MH) đã được phát triển và ứng dụng thành công để giải quyết các bài toán tối ưu hóa trong kỹ thuật do có những ưu điểm vượt trội so với các thuật toán truyền thống dựa trên gradient. Đặc biệt, metaheuristics đã cho thấy khả năng tìm ra giải pháp tối ưu cho các bài toán có quy mô lớn. Một số thuật toán được sử dụng phổ biến nhất là Thuật toán di truyền (GA) [1], Tối ưu hóa bầy đàn (PSO) [2] và Tiến hóa vi phân (DE) [3]. Tuy nhiên, nhược điểm của thuật toán metaheuristic là chúng thường yêu cầu một lượng lớn phân tích hàm.

Để tiết kiệm chi phí tính toán trong tối ưu hóa bằng MH, các mô hình dự đoán được xây dựng trên thuật toán học máy (ML), chẳng hạn như mạng nơ-ron (NN) [4], hệ thống suy luận mờ thần kinh thích ứng (ANFIS) [5], mạng nơ-ron sâu (DNN) [6], máy vectơ hỗ trợ (SVM) [7], thuật toán cây tăng cường (BA) [8]-[10], đã được sử dụng làm giải pháp thay thế cho việc đánh giá hàm trực tiếp. Hạn chế chính của các mô hình ML là yêu cầu một quy trình đào tạo phù hợp. Đào tạo ML bao gồm tìm cấu trúc mô hình phù hợp, chọn tập dữ liệu huấn luyện và xác định các giá trị tham số mô hình phù hợp. Tuy nhiên, không có quy tắc chung cho những nhiệm vụ này. Chúng thường phụ thuộc vào loại bài toán và kinh nghiệm của người thiết kế và đôi khi chúng được quyết định bằng cách thử và sai. Hơn nữa, việc đào tạo một mô hình ML có độ chính xác cao trong một số trường hợp thậm chí còn mất nhiều thời gian hơn so với phân tích trực tiếp, đặc biệt là trong các bài toán phức tạp đòi hỏi lượng lớn dữ liệu đào tạo.

Trong nghiên cứu gần đây, một mô hình phân loại dựa trên thuật toán k-NN đã được đề xuất [11]. Trong cách tiếp cận mới này, quá trình đào tạo là không cần thiết và độ chính xác của mô hình đề xuất được duy trì một cách thích ứng trong quá trình tối ưu hóa. Mô hình mới có tên *phương pháp so sánh k-láng giềng gần nhất* (k-NNC). Trong quá trình tối ưu hóa, k-NNC đánh giá một giải pháp thiết kế mới bằng cách so sánh k láng giềng gần nhất của nó với một giải pháp thiết kế hiện có. Giải pháp thiết kế

mới sẽ bị loại mà không thực hiện phân tích nếu phần lớn k láng giềng gần nhất kém hơn so với giải pháp được so sánh. Trong [11], hiệu quả của k-NNC đã được kiểm chứng với thuật toán Rao [12] thông qua các ví dụ tối ưu kích thước giàn phẳng và không gian với biến rời rạc.

Trong báo cáo này, khả năng của k-NNC được khẳng định khi kết hợp với một số thuật toán MH thông dụng khác như tiến hóa vi phân [3] và Jaya [13]. Kết quả khi áp dụng vào một số bài toán tối ưu kỹ thuật đã chứng minh k-NNC là mô hình đơn giản và hiệu quả để tiết kiệm chi phí tính toán cho MH.

2. Tóm tắt mô hình phân loại k-NNC

Phương pháp k-NNC nhằm đánh giá một giải pháp mới $\boldsymbol{x'}_p$ (không dùng phân tích) trước khi đem so sánh với một giải pháp hiện có $\boldsymbol{x}_p$. Việc đánh giá $\boldsymbol{x'}_p$ được dựa trên *k* giải pháp gần nhất của nó trong quần thể. Tóm tắt các bước của k-NNC như dưới đây.

Với mỗi giải pháp mới được tạo ra $\boldsymbol{x'}_p$:

Bước 1: Tìm *k* giải pháp gần nhất với $\boldsymbol{x'}_p$ (k láng giềng gần nhất, k-NN) dựa trên khoảng cách Euclidean theo biểu thức (1),

$$d(\boldsymbol{x'}_p, \boldsymbol{x}_q) = \sqrt{\sum_{i=1}^{n} \left(\frac{x'_{p,i} - x_{q,i}}{x_{iU} - x_{iL}} \right)^2} \tag{1}$$

Trong đó, $d(\boldsymbol{x'}_p, \boldsymbol{x}_q)$ là khoảng cách Euclidean giữa $\boldsymbol{x'}_p$ và $\boldsymbol{x}_q$, với $\boldsymbol{x}_q$ là giải pháp thứ *q* trong quần thể.

Bước 2: So sánh k-NN với $\boldsymbol{x}_p$. $\boldsymbol{x'}_p$ được xem là kém hơn $\boldsymbol{x}_p$ nếu đa số thành viên trong k-NN kém hơn $\boldsymbol{x}_p$. Trong trường hợp này, $\boldsymbol{x'}_p$ sẽ được loại bỏ.

Bước 3: Nếu đa số thành viên k-NN tốt hơn $\boldsymbol{x}_p$, $\boldsymbol{x'}_p$ được xem là giải pháp có tiềm năng và sẽ được giữ lại và chuyển sang bước phân tích tiếp theo như thông thường.

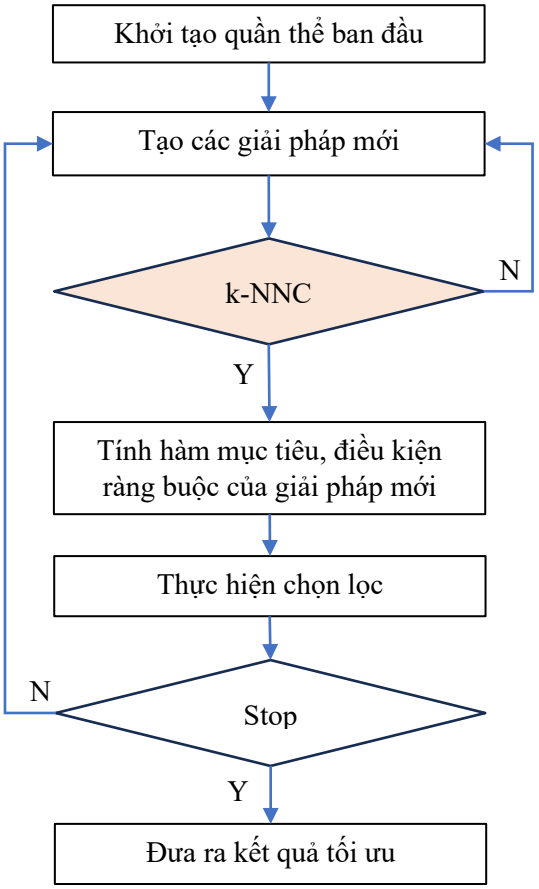
Sơ đồ thuật toán tối ưu hóa MH sử dụng k-NNC được mô tả trong Hình 1. Thông tin chi tiết về k-NNC có thể tham khảo trong [11].

3. Kiểm chứng k-NNC

Trong [11], k-NNC được kết hợp với thuật toán Rao để tối ưu kích thước giàn phẳng và không gian với biến rời rạc. Kết quả cho thấy, k-NNC có thể giảm từ 40% đến 60% số lần phân tích kết cấu. Trong mục này, k-NNC được kết hợp với hai thuật toán MH thông dụng khác là tiến hóa vi phân (DE) [3] và Jaya [13]. Các ví dụ kiểm chứng là một số bài toán tối ưu mẫu đã được nhiều nghiên cứu dùng để kiểm chứng khả năng của các thuật toán tối ưu, bao gồm: dầm hàn (Welded-Beam), lò xo (Spring), bình chịu áp (Vessel), bộ giảm tốc (Speed-Reducer), và dầm bê tông (Concrete-Beam). Chi tiết về các bài toán này có thể tham khảo trong [14].

Các tham số của DE và Jaya sử dụng trong tính toán như sau:

- Kích thước quần thể: 40
- Hệ số tỉ lệ và lai ghép của DE tương ứng là 0.8 và 0.9
- Số vòng lặp tối đa: 1000
- Điều kiện dừng: khi sai số tương đối ε nhỏ hơn 10^{-6}, với $\varepsilon = |\bar{f}/f^* - 1|$; $\bar{f}$ là giá trị trung bình của hàm mục tiêu trong quần thể; f^* là giá trị hàm mục tiêu nhỏ nhất trong quần thể.



Hình 1. Sơ đồ thuật toán tối ưu metaheuristic sử dụng mô hình phân toại k-NNC

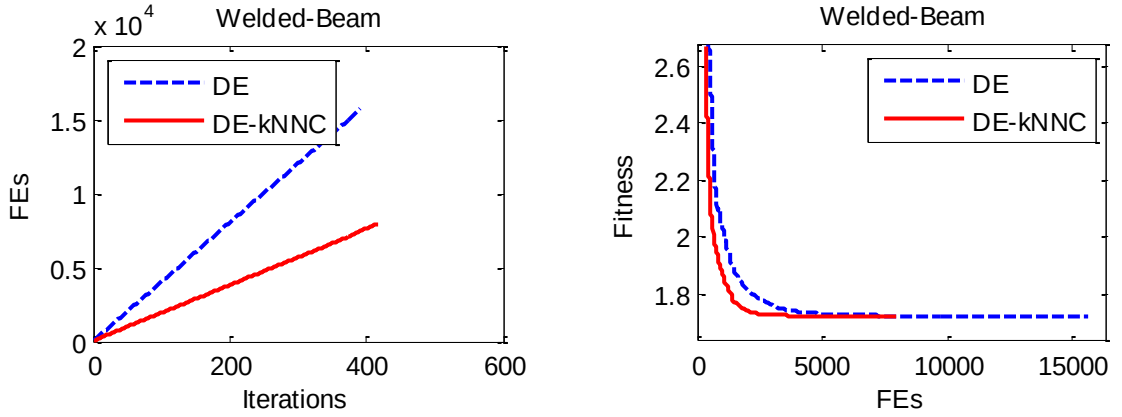
Với mỗi thuật toán và mỗi ví dụ, thực hiện 20 lần tối ưu độc lập nhau để có được kết quả thống kê. Kết quả tối ưu được trình bày trong Bảng 1, Bảng 2, và Hình 2-6. Các kết quả trong Bảng 1 và 2 bao gồm: giá trị tốt nhất (Best), giá trị trung bình (Mean), giá trị xấu nhất (Worst), và độ lêch chuẩn (SD). Tổng số lần phân tích hàm trung bình cũng được cho ở cột NFEs.

Bảng 1. So sánh kết quả tối ưu của DE và DE-kNNC

Ví dụ	**Thuật toán**	**Best**	**Mean**	**Worst**	**SD**	**NFEs**	**%**
Welded Beam	DE	1.7249	1.7249	1.7249	6.4129e-07	14466	
	DE-kNNC	1.7249	1.7249	1.7249	9.5137e-07	7098	49.07
Spring	DE	0.012666	0.012666	0.01267	1.09e-06	8516	
	DE-kNNC	0.01266	0.012681	0.012952	6.3776e-05	3705	43.51
Vessel	DE	6059.7149	6059.7162	6059.7204	0.0014051	7672	
	DE-kNNC	6059.7145	6059.7164	6059.7188	0.0010196	3639	47.43
Speed Reducer	DE	2994.4726	2994.4752	2994.4803	0.0017155	12846	
	DE-kNNC	2994.4738	2994.4772	2994.4822	0.0021473	6488	50.51
Concrete Beam	DE	359.208	359.208	359.208	1.1203e-05	2536	
	DE-kNNC	359.208	359.208	359.208	1.2546e-05	1169	46.10

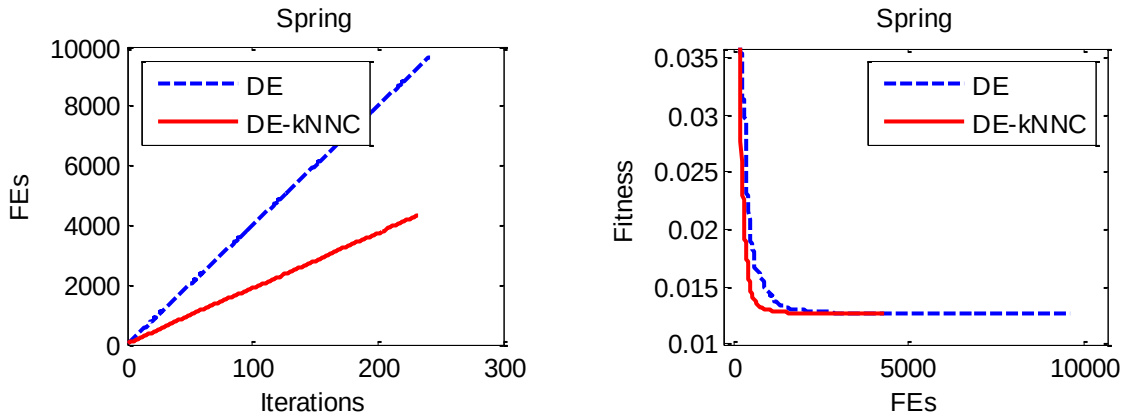
Bảng 2. So sánh kết quả tối ưu của Jaya và Jaya-kNNC

Ví dụ	Thuật toán	Best	Mean	Worst	SD	NFEs	%
Welded Beam	Jaya	1.7249	1.7249	1.7249	3.9571e-07	17892	
	Jaya-kNNC	1.7249	1.7249	1.7249	4.4494e-07	10259	57.34
Spring	Jaya	0.012679	0.012709	0.012727	1.5701e-05	40040	
	Jaya-kNNC	0.012679	0.012719	0.012781	2.258e-05	21781	54.40
Vessel	Jaya	6059.7147	6067.5701	6089.5269	8.8693	33148	
	Jaya-kNNC	6059.7148	6068.3109	6086.783	7.1033	10470	31.59
Speed Reducer	Jaya	2994.4717	2994.4726	2994.4735	0.00053409	7774	
	Jaya-kNNC	2994.4722	2994.4731	2994.4747	0.00074848	5700	73.32
Concrete Beam	Jaya	359.208	359.211	359.2358	0.0078537	20996	
	Jaya-kNNC	359.208	359.3631	362.251	0.67984	5033	23.97



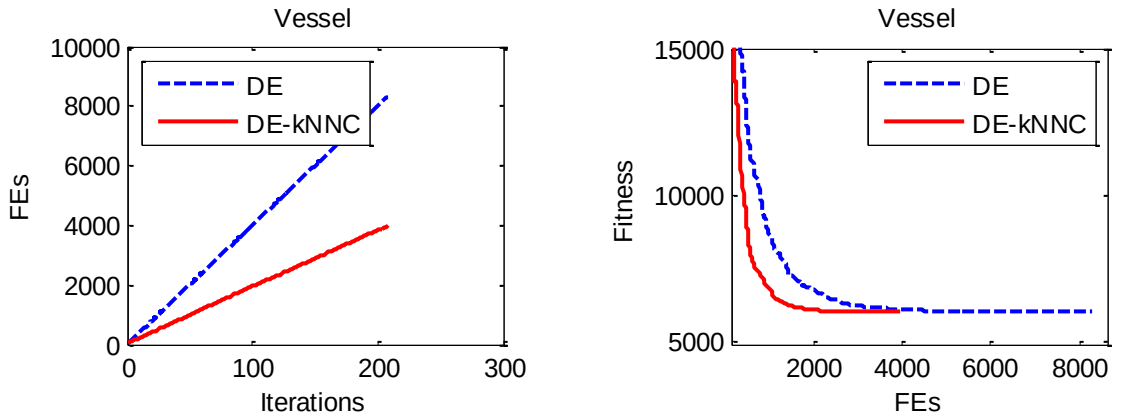
(a) (b)

Hình 2. Welded-Beam: (a) Khối lượng tính toán; (b) Hội tụ của hàm mục tiêu



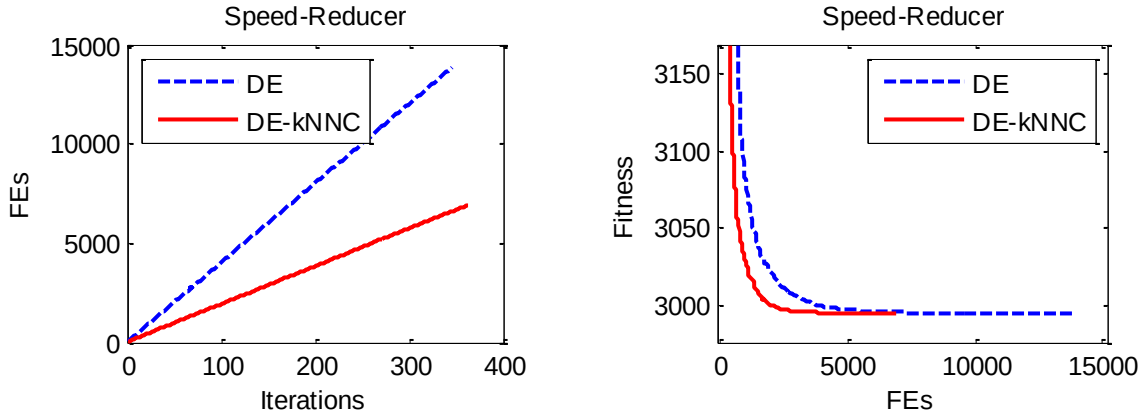
(a) (b)

Hình 3. Spring: (a) Khối lượng tính toán; (b) Hội tụ của hàm mục tiêu



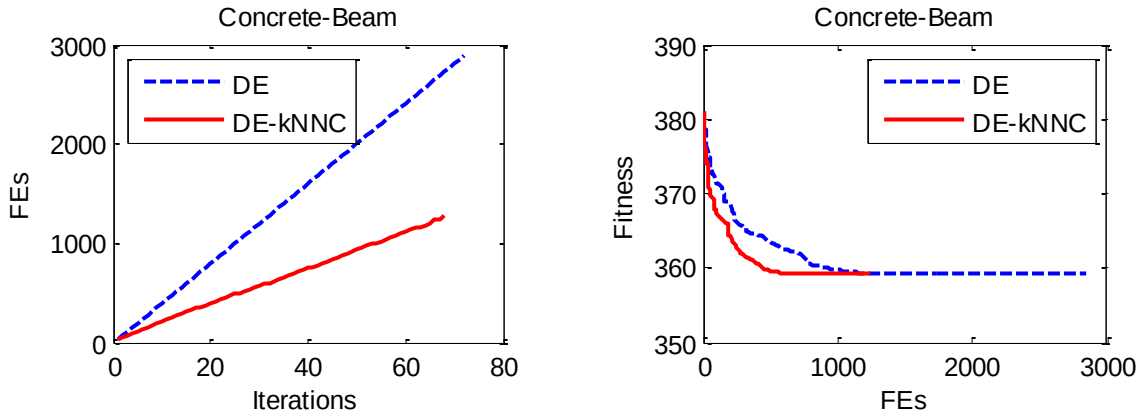
(a) (b)

Hình 4. Vessel: (a) Khối lượng tính toán; (b) Hội tụ của hàm mục tiêu



(a) (b)

Hình 5. Speed-Reducer: (a) Khối lượng tính toán; (b) Hội tụ của hàm mục tiêu



(a) (b)

Hình 6. Concrete-Beam: (a) Khối lượng tính toán; (b) Hội tụ của hàm mục tiêu

4. Thảo luận

Thông qua các kết quả có thể thấy:

- Khi kết hợp với k-NNC, khối lượng tính toán đã giảm đáng kể. Với DE sử dụng k-NNC, số lần phân tích hàm chỉ còn 44% đến 50% so với DE (Bảng 1). Với Jaya, hiệu quả đạt được của k-NNC là giảm từ 27% đến 76% số lần phân tích hàm (Bảng 2).
- Chất lượng kết quả tối ưu khi sử dụng k-NNC tương đương với kết quả tính theo thuật toán gốc.
- Mức độ giảm khối lượng tính toán của k-NNC khi kết hợp với DE là ổn định trong suốt quá trình tối ưu, thể hiện ở quan hệ tương đối tuyến tính giữa số lần tính hàm (FEs) và số vòng lặp (Iterations) trên Hình 2(a)-6(a).
- Sử dụng k-NNC cho tốc độ hội tụ nhanh hơn đáng kể so với khi không sử dụng k-NNC (Hình 2(b)-6(b).

Kết quả trên một lần nữa chứng minh k-NNC là mô hình hiệu quả để tiết kiệm chi phí tính toán khi thực hiện tối ưu hóa theo các thuật toán metaheuristic. Ưu điểm của k-NNC so với các mô hình học máy khác là sự đơn giản và không yêu cầu đào tạo mô hình.

Lời cảm ơn

Nghiên cứu này được tài trợ bởi Quỹ Phát triển khoa học và công nghệ Quốc gia (NAFOSTED) trong đề tài mã số 107.02-2021.03.

Tài liệu tham khảo

[1] J. H. Holland. Genetic algorithms. Scientific american, ***267***, (1), (1992), pp. 66-73.

[2] J. Kennedy, R. Eberhart. Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks (Vol. 4, (1995, November), pp. 1942-1948). IEEE.

[3] R. Storn, K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, ***11***, (1997), pp. 341-359.

[4] T. H. Nguyen, A. T. Vu. Speeding up Composite Differential Evolution for structural optimization using neural networks. Journal of Information and Telecommunication, ***6***, (2), (2022), pp. 101-120.

[5] N. Hosseini, M. R. Ghasemi, B. Dizangian. ANFIS-based optimum design of real power transmission towers with size, shape and panel design variables using BBO algorithm. IEEE Transactions on Power Delivery, ***37***, (1), (2021), pp.29-39.

[6] J. Liu, Y. Xia. A hybrid intelligent genetic algorithm for truss optimization based on deep neutral network. Swarm and Evolutionary Computation, ***73***, (2022), pp. 101120.

[7] H. Cao, H. Li, W. Sun, Y. Xie, B. Huang. A boundary identification approach for the feasible space of structural optimization using a virtual sampling technique-based support vector machine. Computers & Structures, ***287***, (2023), pp. 107118.

[8] N. T. Hieu, N. Q. Cuong, V. A. Tuan. Optimization of steel roof trusses using machine learning-assisted differential evolution. Journal of Science and Technology in Civil Engineering (STCE)-HUCE, ***15***, (4), (2021), pp. 99-110.

[9] T. H. Nguyen, A. T. Vu. An efficient differential evolution for truss sizing optimization using AdaBoost classifier. Computer Modeling in Engineering & Sciences, ***134***, (1), (2023), pp. 429-458.

[10] V. H. Truong, S. Tangaramvong, G. Papazafeiropoulos. An efficient LightGBM-based differential evolution method for nonlinear inelastic truss optimization. Expert Systems with Applications, ***237***, (2023), pp. 121530.

[11] H. A. Pham, V. H. Dang, T. C. Vu, B. D. Nguyen. An Efficient k-NN-based Rao Optimization Method for Optimal Discrete Sizing of Truss Structures. Applied Soft Computing, ***154***, (2024), pp. 111373.

[12] R. V. Rao. Rao algorithms: Three metaphor-less simple algorithms for solving optimization problems. International Journal of Industrial Engineering Computations, ***11***, (1), (2020), pp. 107-130.

[13] R. V. Rao. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. International Journal of Industrial Engineering Computations, ***7,*** (1), (2016), pp. 19-34.

[14] A. H. Gandomi, X. S. Yang. Benchmark problems in structural optimization. In Computational optimization, methods and algorithms, (2011), pp. 259-281. Berlin, Heidelberg: Springer Berlin Heidelberg.