International Journal of Applied Operational Research
Vol. 12, No. 2, pp. 95-103, Spring 2024

Journal homepage: ijorlu.liau.ac.ir

Optimization of active microarchitecture design

A. Sedaghat*, J. Porkar, I. Nejati

Received: 4 December 2023 ; **Accepted:** 15 March 2024

Abstract A complementary follow-on tool is required to optimize the geometric parameters of the topology solutions once the flexure design topology of an active micro-architected material has been synthesized. This will enable the production of final designs that not only meet the desired DOFs for the constituent materials but also best meet the desired performance requirements. This paper introduces a computational tool to identify the boundaries of the performance capabilities achieved by general flexure system topologies, provided that their geometric parameters are allowed to vary from their smallest allowable feature sizes to their largest geometrically compatible feature sizes for given constituent materials. The boundaries fully define flexure systems' design spaces and allow designers to visually identify which geometric versions of their synthesized topologies best achieve desired combinations of performance capabilities.

Keyword: Optimization, Topologies, Micro Artificial Material, Metrical System, Fletcher System.

1 Introduction

A model that describes the performance of an active architecture design with a given set of design parameters must be established to set up the optimization problem. This model can be *1*-a real experiment, whose parameters can be arbitrarily changed or controlled, *2*- a finite element analysis (FEA)-based numerical model, *3*- a closed-form analytical model for the parameterized topologies, or *4*- a regression model based on data generated by finite element analysis or real experiments. This paper introduces a tool that can optimize the parameters of flexure system topologies and be applied to a host of other diverse applications.

System performance boundary identification is a relatively new field of study, but it shares many of the same objectives as the well-studied multi-objective optimization. A multi-objective optimization problem (MOOP) deals with more than one objective function, which aims to find a set of solutions to simultaneously optimize all the objective functions. Several methods, such as the weighting method as one of the most widely used methods, have been proposed to solve sets of local or global Pareto-optimal solutions [1]. The ε-constraint method

* **Corresponding Author. (✉)**
E-mail: sedaghat.alirezaa@gmail.com (A. Sedaghat)

A. Sedaghat
Faculty of Mechanical Engineering, Lahijan Branch, Islamic Azad University, Lahijan, Iran

J. Porkar
Faculty of Mechanical Engineering, Lahijan Branch, Islamic Azad University, Lahijan, Iran

I. Nejati
Faculty of Mechanical Engineering, Lahijan Branch, Islamic Azad University, Lahijan, Iran

[DOI: 10.71885/ijorlu-2024-1-658]

was first proposed by Haimes et al. [2], the method of the global criterion was first unveiled by Yu [3], and the achievement scalarizing function (ASF) approach was first introduced by Wierzbicki [4,5]. Other solving methods include normal boundary intersection (NBI) [6], evolutionary algorithms (EAs) [7], lexicographic ordering [8], and goal programming [9]. More recently, MOOP methods have focused on stochastic optimization algorithms, including various EAs [10,11]. Such algorithms generate more reliable global Pareto-optimal solution sets but require significantly more function evaluations than deterministic algorithms and are thus generally better suited for complicated black-box-model optimizations. The boundary identification approach proposed in this paper has, in part, been adapted from various deterministic MOOP methods such that the complete continuous boundary (including concave portions) that circumscribe the performance capabilities achieved by general flexure topologies can be identified and refined with a desired accuracy.

The utility of this computational tool combined with the current flexible assertive community treatment (FACT) approach results in a novel advantageous approach that sets itself apart from other existing design optimization approaches. Whereas other approaches, e.g., topology optimization [12–14] or module optimization [15], simultaneously combine the design typology optimization with geometry optimization tasks, the proposed approach decouples those tasks in such a way that the time-consuming computations are reserved solely for the simpler geometry optimization task only. This optimization occurs after the FACT approach has directly generated and finalized the most promising topologies without performing expensive iterative calculations. Thus, by decoupling the topology synthesis and geometry optimization tasks, the speed at which optimal designs can be generated from start to finish, as well as the likelihood of identifying the global optimum solutions, increases. The boundary identification approach proposed in this paper has in part been adapted from various deterministic multi-objective optimization methods such that the complete continuous boundary (including concave portions) that circumscribe the performance capabilities achieved by general flexure topologies can be identified and refined with a desired accuracy.

2 Performance boundary identification

The optimization problem is set up where a flexure system topology's design parameters are the model inputs, x_i, and the performance capabilities achieved by the design instantiations defined by these corresponding input parameters are the model outputs, f_j. Constraint functions are also provided to define the combination of input values permissible.

A boundary search algorithm consists of two main processes: *directional maximization* and *gap reduction*. Both processes rely on an optimization approach that implements two numerical optimization methods, i.e., the Sequential Quadratic Programming (SQP) [16,17] algorithm and Augmented Lagrangian Pattern Search (ALPS) [18–20] algorithm, to achieve the local extremum of an objective function.

In each local optimization process, the SQP method is implemented first. The SQP method starts with a given initial guess and attempts to compute, or "step to," another " closer " point to the local extremum. At each point, the gradient (derivatives) and Hessian matrix (the symmetric matrix of second derivatives) of the objective function are approximated using adjacent points and then used to construct a Quadratic Programming (QP) subproblem [21]. The solution of this QP subproblem is used to compute the step towards the next point. The SQP process terminates when the "step" is smaller than a prescribed resolution of the input parameter in all directions x_i or when it fails to generate the next point. This typically occurs

[DOI: 10.71885/ijorlu-2024-1-658]

because the derivatives or second derivatives of the specific objective function cannot be correctly evaluated to set up the QP subproblem.

Once the SQP process terminates, the algorithm proceeds with the optimization process by implementing the ALPS method to solve for the local extremum of the same objective function, starting at the point corresponding to the best inputs (that generate the maximum or minimum value of the objective function) identified using the SQP method. The ALPS algorithm searches for a better value of the augmented Lagrangian function [22] among a set of points, called a *mesh*, located around the current point (the center point of the mesh) at a distance $\pm r_i$, along the direction of each input parameter x_i. The distance r_i is called the *mesh size* and is initially chosen to be between 10% and 30% of the total range of each x_i. If there is a point among the mesh points that increases the value of the augmented Lagrangian function over the current mesh center point, it becomes the new center point in the next step, and the mesh size r_i increases by a factor of k, which is typically set to be 2. On the other hand, if no improvements can be achieved from all the mesh points around the center point, the mesh size decreases by a factor of k. The objective function converges to its extremum value by taking these iterative steps. The ALPS process terminates when the mesh size is smaller than the prescribed resolution in every direction.

A theoretical example of a system with only two inputs, x_1 and x_2, is used to conceptually explain how the boundary search algorithm employs the local optimization approach to plot a concave boundary that circumscribes the full system design space for two of the system's achievable outputs, $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$. Figure 8A shows the constraint function for this theoretical example as the red spline boundary line.

The algorithm begins with the directional maximization process. Starting at an initial guess, this algorithm first identifies a set of other allowable input combinations that result from adding and subtracting the resolution increment of each input, Δx_i, to and from the first randomly selected combinations of inputs along each input's axis. In the example shown in Figure 8A, this first set of input combinations is shown as the four blue dots immediately surrounding the blue dot labeled $O_{1,1}$. Note that although Δx_1 is shown as equal to Δx_2, these resolution increments do not have to be equal for other scenarios. The system's model is then used to map all these input combinations to their corresponding output combinations, represented by the five blue dots shown in Figure 8B. The original input dot, $O_{1,1}$, maps to the output dot, $Z_{1,1}$. The SQP algorithm then approximates the gradient and Hessian matrix of the objective function defined by

$$J(x_1, x_2) = \cos(\theta)\, f_2(x_1, x_2) + \sin(\theta)\, f_1(x_1, x_2), \tag{1}$$

using the input and output combinations (i.e., the adjacent blue dots in Figure A and Figure 8B), where θ is initially set to 0, the largest f_2 output can be pursued first. This gradient and Hessian matrix are then used to construct a QP subproblem. In the example shown in Figure 8, the newly determined input combination is shown as the red dot labeled $O_{1,2}$ (Figure A). Note that the corresponding output combination of this dot, shown as the red dot labeled $Z_{1,2}$ (Figure B), possesses an f2 value larger than any of the previous blue dots. The SQP algorithm then repeats this process by finding a new set of allowable input combinations. In the example shown in Figure A, this new set of input combinations is shown as the four red dots immediately surrounding the red dot labeled $O_{1,2}$. Note that each of the five red dots shown in Figure A maps to a corresponding red dot in Figure B. Thus, the SQP algorithm rapidly finds an efficient path toward a local maximum of the objective function by iteratively stepping from one cluster of dots to the next.

[DOI: 10.71885/ijorlu-2024-1-658]

Once the SQP algorithm terminates, the boundary-plotting algorithm proceeds with the optimization process by supplying allowable input combinations mapped to output combinations that produce the largest objective-function value identified by the SQP algorithm to the ALPS optimization algorithm. In the example shown in Figure A, the SQP process could not step beyond the cluster of dots surrounding $O_{1,2}$ and thus terminated. The input combination supplied to the ALPS algorithm is illustrated by the red dot labeled $O_{1,3}$ in Figure A, which maps to the output combination (shown as the red dot labeled $Z_{1,3}$ in Figure B) that achieves the largest f_2 value found using the SQP algorithm. The ALPS algorithm initially identifies a set of other allowable input combinations that result from adding and subtracting an initial mesh size, r_i, to and from the input combination supplied to the algorithm along each input's axis. The initial mesh size, *r2*, in Figure A, is set to 20% of the range of its corresponding input parameter (i.e., $r_i=0.2|x_{i,max}-x_{i,min}|$). In the example shown in Figure A, the first set of ALPS-generated input combinations is depicted as the four purple dots surrounding the red dot labeled $O_{1,3}$. Although r_1 is shown as equal to r_2, the mesh sizes are not typically equal for other scenarios. The system's model is then employed to map these input combinations to their corresponding output combinations, represented by the four purple dots (Figure B). The ALPS algorithm then identifies if any of these input combinations map to an output combination that produces an objective-function value larger than any previously produced during the optimization process. Suppose, for instance, that the input combination of the example $O_{1,4}$ (Figure A) maps to the output combination $Z_{1,4}$ (Figure B) that achieves the largest f_2 value previously identified. The ALPS algorithm would then step to the dot representing that input combination (e.g., $O_{1,4}$). The algorithm would then identify a set of other allowable input combinations that result from adding and subtracting the previous mesh size (e.g., r_i in this case) multiplied by an expansion factor (i.e., 2) to and from this input combination along each input's axis. Thus, for the example shown in Figure A, the next set of ALPS-generated input combinations are shown as the three orange dots surrounding the purple dot labeled $O_{1,4}$. These orange input dots shown in Figure A map to the three orange output dots shown in Figure B. The algorithm then identifies if any of these output combinations produce an objective-function value larger than any previously produced during the optimization process. Since none of the orange dots in Figure B possess an f_2 value larger than $Z_{1,4}$, the ALPS algorithm would then identify a set of other allowable input combinations that result from adding and subtracting the previous mesh size (e.g., $2r_i$ in this case) divided by the same expansion factor to and from the input combination $O_{1,4}$ in Figure A along each input's axis. Therefore, for the example shown in Figure A, the next set of ALPS-generated input combinations are shown as the three light-green dots surrounding the same purple dot labeled $O_{1,4}$. These light-green input dots shown in Figure A map to the three light-green output dots shown in Figure B. Again, since none of the light green dots in Figure B possess an f_2 value larger than $Z_{1,4}$, the ALPS algorithm would then identify another set of other allowable input combinations that result from adding and subtracting the previous mesh size (e.g., r_i in this case) divided by the same expansion factor to and from the input combination $O_{1,4}$ in Figure A along each input's axis. Thus, for the example shown in Figure A, the next set of ALPS-generated input combinations are shown as the four dark-green dots surrounding the same purple dot labeled $O_{1,4}$. This process repeats until either *(i)* one of the new input combinations maps to an output combination with an objective-function value larger than any produced previously, or *(ii)* the mesh size becomes equal to or less than a specified input tolerance, which herein is set to the resolution of the input parameters, Δxi. If the first option *(i)* occurs, the ALPS algorithm will step to the improved input combination, and the ALPS process will continue to iterate. If the second option (ii) occurs, it will terminate. For the

[DOI: 10.71885/ijorlu-2024-1-658]

example shown in Figure A, the second option occurred as the mesh size of the four yellow dots shown immediately surrounding the purple dot $O_{1,4}$ is equal to Δx_2, and none of the new output dots generated ever surpassed the f_2 value of $Z_{1,4}$, as depicted in Figure B.

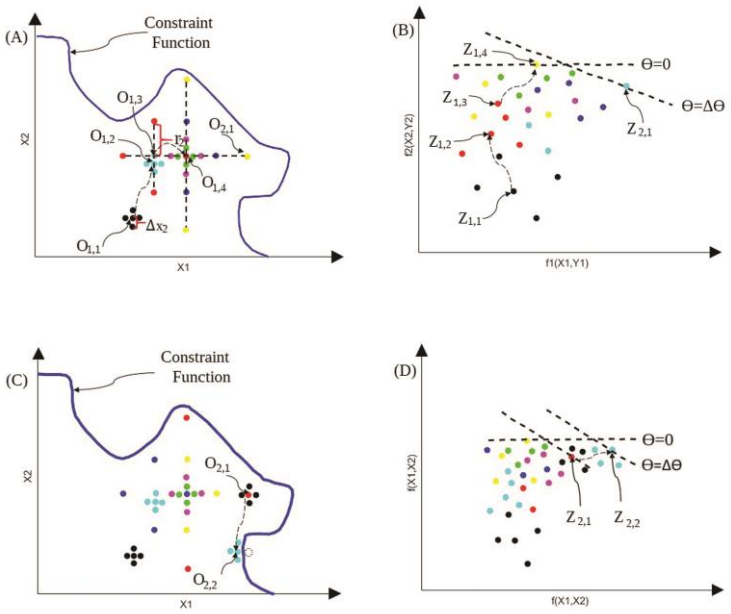

Fig. 1 Progression of the SQP and ALPS optimization algorithms for θ=0 initialized in the objective function in the input space (A) and the corresponding output space (B).

Continued progression of the same optimization algorithms for θ=$\Delta\theta$ incremented in the objective function in the same input space (C) and the corresponding output space (D).

Once the SQP and ALPS algorithms have both run their full course for determining the maximum value of $J(x_1, x_2)$ from Equation (1) for θ=0, θ is then incrementally increased by $\Delta\theta$, which is typically set to a value between $\pi/10$ and $\pi/20$. Using this new θ parameter, the algorithm then computes the value of the objective function in Equation (1) for all existing input combinations, which, for the example illustrated in this section, are shown as the dots in Figure A. From among these input combinations, the one that produces the maximum objective function value for θ=$\Delta\theta$ corresponds to the input combination represented by the dot labeled $O_{2,1}$ in Figure 8A, which maps to the output combination represented by the orange dot labeled $Z_{2,1}$ in Figure B. The algorithm would then supply the SQP algorithm with this input combination to generate more input combinations that produce larger objective function values as described previously. The four blue dots immediately surrounding the dot labeled $O_{2,1}$ in Figure C would be identified first for the example depicted in this section using this approach. These dots map to the four new blue dots shown in Figure D surrounding the dot labeled $Z_{2,1}$. The SQP algorithm would then identify the next input combination (e.g., $O_{2,2}$ shown in Figure C) that produces a larger objective function value. Note that the input combination dot $O_{2,2}$ maps to an output combination dot $Z_{2,2}$ (Figure D) farther away along the direction prescribed by the new θ value (i.e., $\Delta\theta$). As the SQP algorithm continues, four other input combinations will be identified immediately surrounding the input combination dot $O_{2,2}$ (colored red) in Figure C. The SQP algorithm would continue in this way until it terminates. The ALPS algorithm would then take over where the SQP algorithm left off, as described previously until the former terminates as well. Once the ALPS algorithm

[DOI: 10.71885/ijorlu-2024-1-658]

terminates, the algorithm will have found the input combination that achieves the maximum objective function value identified from among those previously tested for $\theta=\Delta\theta$.

The algorithm iterates this pattern of steps to identify the combinations of output values that lie farthest away along their prescribed directions defined by their corresponding θ value in a clockwise fashion until $\theta \geq 2\pi$ (i.e., all the directions have been swept). Any time before θ is incrementally advanced, the objective function of Equation (1) is rechecked for every combination of input values evaluated up to that point to ensure each of the previously identified output dots that used to be the farthest away along their prescribed directions are still the farthest away. If a new dot is ever identified to be farther away than a previous dot along a specific direction θ (i.e. if a new dot exceeds the dashed lines in Figure B), the iterative process is reset to that direction, and the process continues using that improved output dot.

Once the directional maximization process is complete, the α-shape of all the combinations of output value points is identified and plotted as the boundary. Many systems produce a cloud of output dots that form a concave—not convex—region like the one shown in Figure A. If the boundary of such a cloud of output dots is identified, the result would be the red boundary shown in Figure B. This boundary would grossly overestimate the achievable performance space of the actual system since it is convex rather than concave, like the cloud of output dots. Thus, the algorithm adopts the gap reduction process to address this issue to reduce the gap size along the boundary curve.

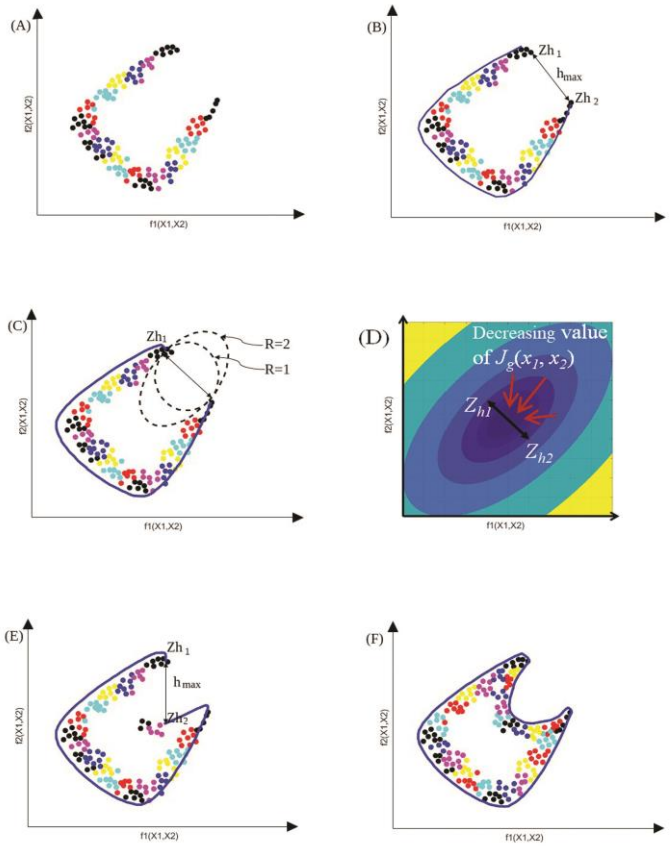

Fig. 2 Generated output dots (A); initially identified convex boundary (B); new objective function minimized to identify other output dots within circular or elliptical regions (C); elliptical contour plot of the new objective function with R=2 (D); boundary updated with a new h_{max} value (E); the process successfully identifies concave boundaries (F)

[DOI: 10.71885/ijorlu-2024-1-658]

This gap reduction process starts with identifying all the vectors that point from each output dot along the existing boundary to their neighboring dots on the same boundary. The magnitude, h_{max}, of the longest vector is identified because this vector points between the two output dots (e.g., Z_{h1} and Z_{h2}, shown in Figure B) that usually correspond to the opening of a previously unknown concave boundary. The algorithm then computes a new objective function, $J_g(x_1, x_2)$, defined by

$$\begin{aligned} J_g(x_1,x_2) &= \left(\frac{cos^2\alpha}{R^2} + sin^2\alpha\right)\left(f_1(x_1,x_2) - \frac{f_{1,Zh1}+f_{1,Zh2}}{2}\right)^2 \\ &+ \left(\frac{sin^2\alpha}{R^2} + cos^2\alpha\right)\left(f_2(x_1,x_2) - \frac{f_{2,Zh1}+f_{2,Zh2}}{2}\right)^2 \\ &+ 2sin\,\alpha\, cos\alpha\left(\frac{1}{R^2} - 1\right)\left(f_1(x_1,x_2) - \frac{f_{1,Zh1}+f_{1,Zh2}}{2}\right)\left(f_2(x_1,x_2) - \frac{f_{2,Zh1}+f_{2,Zh2}}{2}\right) \end{aligned} \quad (2)$$

for all the input combinations previously evaluated with R=1. This objective function is minimized to identify output combinations that lie within circular or elliptical regions like those shown in Figure C. The region is circular if R=1 in Equation (2). However, if R increases, it becomes an increasingly elongated ellipse, as shown in Figure C. The angle, α, in Equation (2) is defined by

$$\alpha = arctan\left(\frac{f_{1,Zh2}+f_{1,Zh1}}{f_{2,Zh1}+f_{2,Zh2}}\right), \quad (3)$$

where $f_{1,\,Zh1}$ and $f_{2,\,Zh1}$ are the horizontal and vertical components of Z_{h1}, while $f_{1,\,Zh2}$ and $f_{2,\,Zh2}$ are those of Z_{h2} (Figure C). After computing the objective function of Equation (2) for all the previously evaluated input combinations for R=1, it identifies the existing input combination that produces the minimum objective function value. This input combination maps to the output combination closest to the center of the circle shown in Figure C. If no output dots are found within the circle, the input combination corresponding to either the output dot Z_{h1} or Z_{h2} will be chosen as the closest to the center of the circle and is thus supplied to the previously described SQP and ALPS algorithms to evaluate new input combinations. For the example shown in Figure, a new group of blue output dots (Figure C) is generated after optimization. Since none of these output dots lies within the center of the circle, the previous R-value in the objective function of Equation (2) is multiplied by a factor of 2, and the search region is expanded to an ellipse shown in Figure C. Figure E displays an elliptical contour plot of $J_g(x_1, x_2)$ for this R-value (i.e. R=2). If no output dots are found within the new ellipse, the process continues to iterate by multiplying the previous R-value by the same factor of 2 to further increase the elliptical search region. For the example shown in Figure C, however, output dots lie within the elliptical search region corresponding to an R-value of 2. Therefore, the input combination that maps to the output dot that lies within this region and possesses the minimum objective function value for R=2 is supplied to the SQP-ALPS optimization algorithm to identify an even better output dot that achieves an even smaller objective function value. This process will produce new output dots (e.g., the new set of orange dots shown in Figure 9). Whether or not these new output dots achieve a smaller objective function value, the output dot that obtains the minimum objective function value is identified and considered part of the system's performance boundary. It is thus redefined as either Z_{h1} or Z_{h2}. In the example shown in Figure E, the Z_{h2} output dot is redefined. Note also that h_{max} is updated as well. This boundary learning process is repeated until both *(i)* the horizontal component of the boundary vector with the largest magnitude (i.e., h_{max}) is less than a set

[DOI: 10.71885/ijorlu-2024-1-658]

percentage of the horizontal distance across the full cloud of output values, and *(ii)* the vertical component of the same vector is also less than the same percentage of the vertical distance across the same cloud. This percentage threshold is typically set between 5% and 10%.

Additionally, before the largest boundary vector is updated with a new magnitude (i.e., h_{max}), the entire optimization process in this section is repeated using the previous objective function in Equation (2) for all θ values. This is to ensure that any new dots evaluated using that objective function are allowed to improve the accuracy of the boundary if possible. Accordingly, both convex and concave boundaries, like the concave boundary shown in Figure F, can be identified that accurately define the achievable performance space of the system.

3 Conclusions

To set up the optimization problem, a model needs to be established that describes the performance of an active architecture design with a given set of design parameters. This model can be a real experiment, of which the parameters can be arbitrarily changed and controlled, a numerical model based on finite element analysis, a closed-form analytical model of the parameterized topologies, or a regression model based on data generated by finite element analysis or real experiments. Note also that although the tool is introduced in this paper as a tool for optimizing the parameters of flexure system topologies, it could also be applied to a host of other diverse applications.

Although few researchers have directly attempted system-performance boundary identification, its goal is similar to the goal of multi-objective optimization, which has been studied extensively. A multi-objective optimization problem (MOOP) deals with more than one objective function and aims at finding a set of solutions that optimizes all the objective functions simultaneously. Several methods have been proposed to solve the local or global Pareto-optimal solution set. The boundary identification approach proposed in this paper has in part been adapted from various deterministic multi-objective optimization methods such that the complete continuous boundary (including concave portions) that circumscribe the performance capabilities achieved by general flexure topologies can be identified and refined with a desired accuracy.

References

1. Song, G., Long, Q., Luo, Y., Wang, Y., & Jin, Y. (2020). Deep convolutional neural network based medical concept normalization. IEEE Transactions on Big Data, 8(5), 1195-1208.
2. Yan, Q., Wang, B., Gong, D., Luo, C., Zhao, W., Shen, J., ... & You, Z. (2021). COVID-19 chest CT image segmentation network by multi-scale fusion and enhancement operations. IEEE transactions on big data, 7(1), 13-24.
3. Babiker, M. A., Elawad, M. A., & Ahmed, A. H. (2019, September). Convolutional neural network for a self-driving car in a virtual environment. In 2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE) (pp. 1-6). IEEE.
4. Zhang, C., Li, R., Kim, W., Yoon, D., & Patras, P. (2020). Driver behavior recognition via interwoven deep convolutional neural nets with multi-stream inputs. Ieee Access, 8, 191138-191151.
5. Zimmer, L., Lindauer, M., & Hutter, F. (2021). Auto-pytorch: Multi-fidelity metalearning for efficient and robust autodl. IEEE transactions on pattern analysis and machine intelligence, 43(9), 3079-3090.
6. Wang, Q., Wu, B., Zhu, P., Li, P., Zuo, W., & Hu, Q. (2020). ECA-Net: Efficient channel attention for deep convolutional neural networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 11534-11542).

[DOI: 10.71885/ijorlu-2024-1-658]

7. Kajkamhaeng, S., & Chantrapornchai, C. (2021). SE-SqueezeNet: SqueezeNet extension with squeeze-and-excitation block. International Journal of Computational Science and Engineering, 24(2), 185-199.
8. Fishburn, P. C. (1974). Exceptional paper—Lexicographic orders, utilities and decision rules: A survey. Management science, 20(11), 1442-1471.
9. Charnes, A., & Cooper, W. W. (1977). Goal programming and multiple objective optimizations: Part 1. European journal of operational research, 1(1), 39-54.
10. Jong, K. A. D., (2006), Evolutionary Computation: A Unified Approach, MIT Press.
11. Padhye, N., Bhardawaj, P., & Deb, K. (2013). Improving differential evolution through a unified approach. Journal of Global Optimization, 55, 771-799.
12. Luo, Z., Chen, L., Yang, J., Zhang, Y., & Abdel-Malek, K. (2005). Compliant mechanism design using multi-objective topology optimization scheme of continuum structures. Structural and Multidisciplinary Optimization, 30, 142-154.
13. Bendsoe, M. P., & Sigmund, O. (2004). Topology optimization: theory, methods, and applications. Springer Science & Business Media.
14. Sigmund, O. (1997). On the design of compliant mechanisms using topology optimization. Journal of Structural Mechanics, 25(4), 493-524.
15. Cao, L., Dolovich, A. T., Schwab, A. L., Herder, J. L., & Zhang, W. (2015). Toward a unified design approach for both compliant mechanisms and rigid-body mechanisms: Module optimization. Journal of Mechanical Design, 137(12), 122301.
16. Fletcher, R. (2000). Practical methods of optimization. John Wiley & Sons.
17. Ronneberger, O., Fischer, P., Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W., Frangi, A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science, vol 9351. Springer, Cham. https://doi.org/10.1007/978-3-319-24574-4_28
18. Lewis, R. M., Torczon, V. J., & Kolda, T. G. (2006). A generating set direct search augmented Lagrangian algorithm for optimization with a combination of general and linear constraints (No. SAND2006-5315). Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA (United States).
19. Conn, A. R., Gould, N. I., & Toint, P. (1991). A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. SIAM Journal on Numerical Analysis, 28(2), 545-572.
20. Conn, A., Gould, N., & Toint, P. (1997). A globally convergent Lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds. Mathematics of computation, 66(217), 261-288.
21. Fang, S.-C. and Puthenpura, S. (1994), Numerical linear algebra and optimization–volume 1, by Philip E. Gill. Walter Murray, and Margaret H. Wright, Addison-Wesley, Redwood City, CA, 1991, 448 pp. Price: $46.25. Networks, 24: 128-129. Fang, S.-C. and Puthenpura, S. (1994), Numerical linear algebra and optimization–volume 1, by Philip E. Gill. Walter Murray, and Margaret H. Wright, Addison-Wesley, Redwood City, CA, 1991, 448 pp. Price: $46.25. Networks, 24: 128-129.
22. Débarre, D., Botcherby, E. J., Watanabe, T., Srinivas, S., Booth, M. J., & Wilson, T. (2009). Image-based adaptive optics for two-photon microscopy. Optics letters, 34(16), 2495-2497.

[DOI: 10.71885/ijorlu-2024-1-658]