Proceedings of the IASS 2024 Symposium
Redefining the Art of Structural Design
August 26-30, 2024, Zurich Switzerland
Philippe Block, Giulia Boller, Catherine DeWolf,
Jacqueline Pauli, Walter Kaufmann (eds.)

Enhancing Spatial Truss Designs by Integrating Metaheuristic Optimization Techniques via Visual Programming in BIM-based Projects

Feyzullah YAVAN, Reza MAALEK*, Shahrokh MAALEK[a]

*Department of Digital Engineering and Construction, Karlsruhe Institute of Technology (KIT)
Bldg. 50.31, Am Fasanengarten, 76131, Karlsruhe, BW, Germany
Email: reza.maalek@kit.edu

[a] Digital Innovation in Construction Engineering (DICE) Technologies

Abstract

Creating sustainable, affordable, and nature-friendly designs is crucial to address many of the construction demands in contemporary societies. The advent of new digital technologies, including artificial intelligence (AI)-based metaheuristic optimization tools, have enabled the selection of optimal designs to achieve a fair balance between all factors that can predict project success. To this end, this study proposes a workflow for optimizing existing spatial trusses by integrating visual programming (VP), numerical structural analysis, and metaheuristic biologically/nature-inspired AI optimization methods. The methodology encompasses several steps including: (i) creating parametric trusses using VP from available digital designs in Dynamo; (ii) performing structural analysis using the finite element method (FEM) in Robot Structural Analysis (RSA) software; and (iii) generating new trusses using metaheuristic algorithms (MA). The effectiveness of the framework for truss optimization was validated on established benchmark problems, including a 2D 10-bar truss and a 3D 120-bar dome truss. The results for each benchmark problem were reported and compared using the following configurations: (i) two different MAs, namely, the Genetic Algorithm (GA) and Particle Swarm Optimization (PSO); and (ii) two different constraint handling techniques (CHT), namely the penalty function method and the separation technique. It was revealed that the GA combined with the penalty method yielded the best results. The results also showed that it was possible to achieve results (on average for both trusses) within 12% of the best reported in the considered literature with over an order of magnitude less function evaluations.

Keywords: Structural Optimization, AI in Construction, Generative Modeling, Parametric Design, Visual Programming

1. Introduction

Conventional engineering design workflows in the Architecture, Engineering, and Construction (AEC) industry, particularly in the context of the design-bid-build delivery strategy, is divided between architects and engineers, increasing the possibility of miscommunication, and hindering seamless wholistic optimal engineering design solutions. This is since the roles of architectural and engineering firms in the divided AEC sector are inherently separated. On the one hand, structural optimization (SO) is a natural step in the engineering process to minimize the use of resources/materials, reduce cost, and satisfy desired design constraints [1]. On the other hand, the traditional role of architects focuses on aesthetics, space management, and satisfying end-user (occupant) requirements. While parametric modeling is increasingly used by architectural firms to increase design flexibility, the models are typically not shared with the engineering team, and consequentially not used for engineering analysis and optimization [2]. This is in part due to the lack of interoperability between structural analysis and

parametric visual programming tools. Particularly in the absence of an interoperable platform for sharing data between different project proponents, such as the common data environments (CDE) in building information modeling (BIM) projects, the information modeled by the architects cannot be utilized in its fullest potential by the engineering team, which may result in rework during the design and modeling stages of the project. Developing an interoperable CDE to connect the engineering analysis software with parametric models, however, requires considerable technical knowledge and skill from the engineering team [3].

To allow for a seamless transition between parametric models generated by the architectural teams and structural analysis together with optimization by the engineering team, this study proposes a generic and user-friendly framework to not only link the information between the engineering and the architectural teams, but also incorporate advanced artificial intelligence (AI)-based metaheuristic algorithms (MA) to generate wholistic optimal solutions based on stakeholder requirements. The advancements in AI-based optimization, particularly in multi-object and many-objective optimization strategies -with conflicting interests-, have shown effective in solving complex engineering problems [4]. The integration of Visual Programming, finite element modeling (FEM), and AI-based optimization within one framework can support the simultaneous optimization of architectural and engineering requirements while minimizing design rework in BIM-based projects. This interoperability can not only reduce rework during the engineering design stage, but also provide designs that support local resource constraints to achieve a fair balance between various requirements, such as cost, time, and sustainability [5]. Given the inherent importance of reducing the embodied CO_2 and energy emissions as pillars to support the United Nation's (UN) Sustainable Development Goals (SDG) [6], the optimized structure can be configured to enable further analyses and simulations, such as life cycle analysis (LCA) and cost estimation through quantity surveying as illustrated in [7].

Many existing studies have demonstrated the effectiveness of MAs in SO problems; however, the studies commonly developed code from scratch, and not within design software packages used by architects and engineers in the industry. This approach may not be suited for the large-scale adoption of these optimization methods by the AEC industry, as it requires a high level of technology literacy. To support the AEC industry with an approach that can be easily integrated into their existing supply-chain processes, the referenced research [7] concentrated on devising a workflow by combining Revit-Dynamo with RSA to establish a reliable, streamlined, and intuitive setting for SO where MA methods can be readily implemented. In this article, the proposed workflow in [7] is enhanced by employing the RSA-API (Application Programming Interface) to speed up the process and leverage the features of RSA without relying on additional third-party packages. The suggested framework's applicability was validated on two truss benchmark problems, commonly used in open literature. Two distinct constraint handling techniques (CHT), namely the penalty function method and the separation technique, along with two different MAs, namely the genetic algorithm (GA) and particle swarm optimization (PSO), were also used during the validation process.

2. Background

The aim of SO is to find the best design among various possible choices while satisfying problem-specific constraints. Given the inherent complexity in structural design problems, AI-based MA strategies, such as GA, must be commonly employed to provide a close to optimal solution. Simulated annealing (SA), proposed by Kirkpatrick et al. [8], was one of the first examples of such algorithms that mimic physics by formulating temperature change of metals. In the early 90s, Holland [9], inspired from the theory of evolution, developed the GA. Further developments by Dorigo et al. [10] and Kennedy et al. [11] paved the path to the inception of well-known algorithms, including ant colony optimization (ACO), which is inspired by the foraging behavior of ants; and PSO, which mimics swarm behavior, respectively. The body of literature in MA is vast and the readers can refer to the referenced review articles [12], [13] for further information.

On the other hand, researchers focused on employing MA for SO with the aim of achieving optimum performance from a structure while lowering costs and material usage. In that direction, Saka and Ulker [14] implemented GA into the optimization methodology and created an algorithm to optimize space

truss structures while taking into consideration of nonlinear behavior of structures. Rajeev and Krishnamoorthy [15] presented a methodology to transform constrained problems into unconstrained ones and used discrete sets of solutions for problems to show the efficiency of GAs in handling discrete variables. Groenwold et al. [16] also used discrete optimization and added slenderness and buckling to the design problems and solved them with GA. In the article presented by Goodarzimehr et al. [17], scholars introduced a new smart algorithm known as the Bonobo Optimizer (BO) algorithm to optimize the sizing of truss structures, accommodating both discrete and continuous variables. The algorithm is mimicking primates' behavior, such as fission-fusion strategy and mating behaviors, to optimize the cross-sectional areas of truss elements. Their approach was formulated in MATLAB and the truss structures are analyzed using the direct stiffness method. Aydın [18] employed the Jaya algorithm with the objective to minimize the costs while increasing material efficiency on the prestressed structures. The optimization focused on the size of structural elements, including prestressed elements' size, based on different prestress loss conditions by considering both slenderness and displacement constraints. The study calculates the penalized objective function by using the equation proposed by Rajeev and Krishnamoorthy [15] which is one of the most successful formulations for the penalization process as stated by Aydın [18]. All steps of the design process and optimization were performed on the coded software. Singh et al. [19] provided an improved version of the follow the leader (iFTL) method, which mimics the behavioral movement of a sheep within the flock, to solve a variety of challenging optimization and truss design problems. The manuscript by [20] demonstrated the efficiency of artificial bee colony (ABC) algorithm by optimizing the layout and member sizes of truss structures, which was the first application of ABC to solve size and shape optimization simultaneously. The scholars have also created hybrid algorithms to increase MAs' performance. In this line, Jafari et al. [21] proposed a new combination of the culture algorithm (CA) together with the PSO, referred to as the particle swarm optimizer cultural (PSOC) method for SO problems with promising results. However, there was a need for the integration of AI-based optimization with visual programming (VP) in Building Information Modeling (BIM) projects to create a user-friendly, interoperable framework, suitable for low technology literacy and programming skills that enhances structural efficiency and sustainability. A workflow to bridge the latter gap by integrating RSA-API to accelerate the structural analysis and optimization processes was proposed in [7]. This study builds on the findings of [7] by comparing the impact of different AI-based optimizations, along with the choice of constraints on the results of the structural optimization. In the following, the two main MAs used in this study, namely the GA and PSO, were explained in more detail.

2.1. Genetic Algorithm

GA integrates evolutionary operators into the algorithm by mimicking the evolutionary theory to create a robust search mechanism [15]. GA is a mixture of Darwin's theory with a structured yet random technique of exchanging information that results in a systematic search strategy that can be used to find near optimal solutions to a variety of problems [22]. GA is simple to implement, due to their evolutionary basis are flexible to dynamic or changing environments by design, rendering them useful in a variety of disciplines, including engineering, economics, and biology [22]. GA can integrate various evolutionary operators, such as reproduction, deletion, dominance, crossover, duplication, mutation, inversion, migration, etc., depending on the optimization problem. Based on the specific problem at hand, GAs have the potential to be enhanced by increasing the number of operators [15]. In the present study, a simple GA is utilized that employs the reproduction, mutation, and crossover operators since these operators have been shown to be sufficient in solving the popular combinatorial knapsack problem. The knapsack problem resembles the combinatorial sizing optimization problem considered in this study. In terms of implementation, the PyMoo library [23] was utilized inside the Python environment.

2.2. Particle Swarm Algorithm

As an alternative for GA, another nature-inspired stochastic optimization technique that mimics the social behavior of birds or fish groups is called the PSO algorithm. In PSO, individuals with the potential best solution share their information with other individuals and that information are constantly updated by the group members to reach the best result [24],[25]. The algorithm is known for its simplicity, user-friendly deployment, and efficient convergence; thus, it has been implemented in various fields, such as

engineering, economics, and bioinformatics [26]. In terms of implementation, this study used the PSO through the PyMoo library to provide alternative results to GA and compare them on benchmark SO problems. Further documentation about utilized library and operators can be found in [27].

2.3. Formulation of the truss optimization problem

In SO problems, the commonly used objective function is to minimize structural weight, while satisfying desired constraints [28],[29] as shown as follows:

$$min\, f(x) = \sum_{n=1}^{m} \left(\rho_j A_j l_j\right), \tag{1}$$

where, ρ, A, and l are accordingly the density of the material, cross-sectional area of the member, and the length of the member. The constraints, including the allowable member stress as well as displacement, adopted from [15], are as follows:

$$g_i(x) = \frac{\boldsymbol{\sigma_j}}{\boldsymbol{\sigma_a}} - 1 \leq 0, \qquad g_i(x) = \frac{\boldsymbol{u_j}}{\boldsymbol{u_a}} - 1 \leq 0, \tag{2}$$

where σ_j and σ_a are represent the stress value on the element and the allowable stress respectively. Similarly, u_j and u_a denote the displacement at the nodes and the allowable displacement. Given the non-linearity of the problem and the relationship between the constraints and the objective function, many CHTs have been developed over the past several decades [30] to solve the SO problem. Four categories have been created to group the most common CHTs [31] as follows:

1. Penalty-based approaches [30], [32], [33]; include strategies, such as the death penalty, static penalty, dynamic penalty, adaptive penalty, exact penalty, and the self-adaptive fitness formulation.
2. Separation of objective and constraints [30], [34], [35]; include the Adaptive constraint-handling technique (ACT), ε-constrained method, Constraint violation with interval arithmetic (CVI), etc.
3. Retaining the infeasible solutions in the population [31].
4. Hybrid techniques [30], [31]; Push and pull search, two stage framework, etc.

It should be noted that there are also alternative classification methodologies for CHTs. Among them, the fundamental and most popular CHT is the objective function penalization [30]. In this study, the first two CHTs, namely penalty-based through objective function penalization, and separation-based through the ε-constrained method were used.

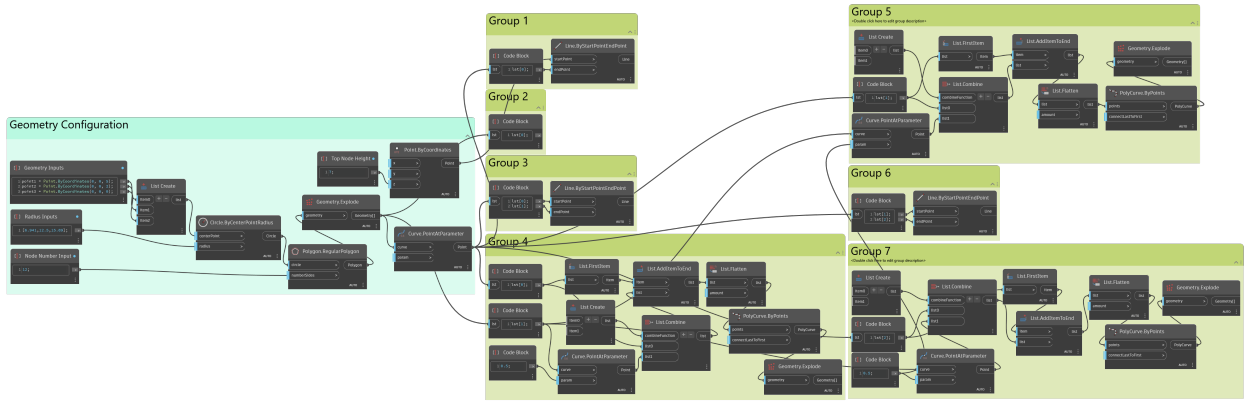
3. Methodology

The methodology is structured around the following steps:

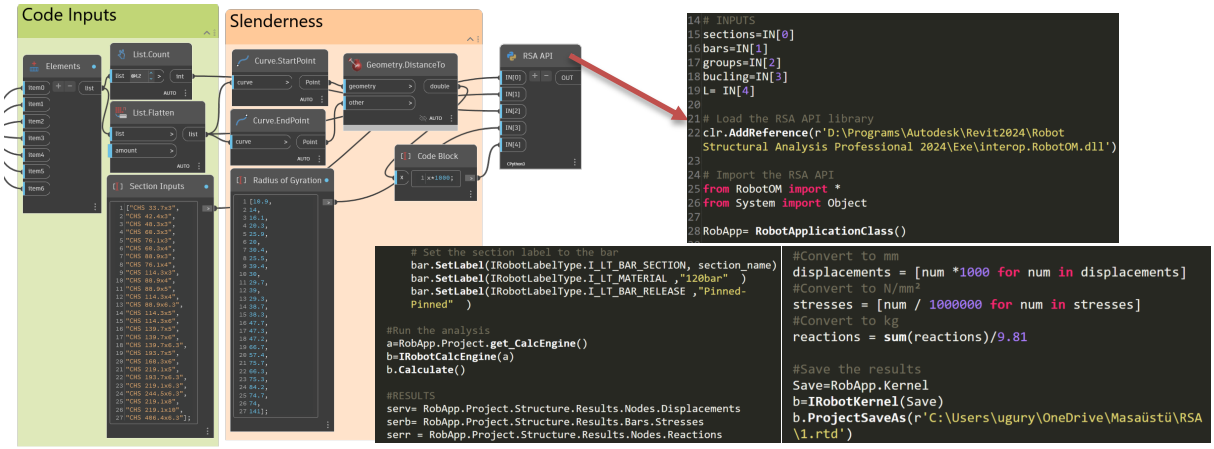
1. Generate parametric trusses by employing VP (Figure 1a);
2. Conduct structural analysis on RSA by employing RSA-API and obtain results (Figure 1b);
3. Evaluate the analysis results to rank created structures. (Figure 1c);
4. Create new populations by utilizing MAs and repeat previous steps until the stopping criteria is fulfilled. (Figure 1d).

3.1. Parametric Model Creation

Parametric modeling is a design strategy that focuses on developing flexible solutions that can respond to changing conditions or requirements. It is a process that enables alterations to model geometry and other features by only adjusting the inputs of the model. This makes it very straightforward and appropriate when combined with VP for inexperienced users with low technology literacy as it does not require substantial syntactic understanding. In Figure 1a, several attributes, such as truss height and length, number of bracings, etc., are specified using a number-integer slider or code block nodes as inputs or design decision variables. The optimization process then involves finding the best combination of the decision variables that minimize the objective function.



(a)



(b)

```python
#Sets the scoring system for the stress ratios
TensAllowableStress=172.37
bucling=final_stresses
AllowableDisplacement=50.8
violation_coef=10
stress_ratio=[]
violation = 0  # Initialize violation as a float or integer

for i in tensstress:
    ratio = i / TensAllowableStress
    stress_ratio.append(ratio)


for item1, item2 in zip(compstress, CompAllowableStress):
    ratio = item1 / item2
    stress_ratio.append(ratio)


for d in Displacement:
    ratio = d / AllowableDisplacement
    stress_ratio.append(ratio)

for r in stress_ratio:
    if r - 1 > 0:
        violation += r - 1

weight_score=weight*(1+(violation*violation_coef)

return weight_score
```

(c)

```python
class TrussProblem(Problem):
    def __init__(self):
        super().__init__(n_var=10,
                         n_obj=1,
                         n_ieq_constr=1, n_eq_constr=0,
                         xl=np.array([0]*10),
                         xu=np.array([40]*10),
                         vtype=int)


    def _evaluate(self, x, out, *args, **kwargs):
        x=x.tolist()
        weight_score= [myfunction(sublist) for sublist in x]
        G,F=zip(*weight_score)
        out["F"] = np.array(F)
        out["G"] = np.array(G)


problem = TrussProblem()

algorithm = AdaptiveEpsilonConstraintHandling(GA(pop_size=20,
                sampling=IntegerRandomSampling(),
                crossover=SBX(prob=1.0, eta=3.0, vtype=float,repair=RoundingRepair()),
                mutation=PM(prob=1.0, eta=3.0, vtype=float,repair=RoundingRepair()),
                eliminate_duplicates=True),perc_eps_until=0.5)

res = minimize(problem, algorithm,('n_gen', 20),verbose=True)
```

(d)

Figure 1. Schematic representation of the proposed methodology; (a) Parametric modeling of 120-bar truss; (b) Performing structural analysis through API and a small section of the utilized code; (c) Calculating weight score using Penalty method; (c) Employing an MAs and CHT through Python libraries

3.2. Perform Calculation and Retrieve Results through RSA-API

An integral part of the workflow for SO is conducting structural analysis. In the present investigation, an integration between Dynamo and RSA was formed via the API, allowing for the development of models and the allocation of parameters, such as materials, loads, and sections on RSA, depending on Dynamo inputs. To generate the structure on RSA, the members (lines) formed in the prior step are required to associate with the Python code that contains API demonstrated in Figure 1b. Following the generation of the bars, cross-sections and connection details must be assigned, along with the definition of support location and type, load type and magnitude, and analysis type. The API conducts the analysis,

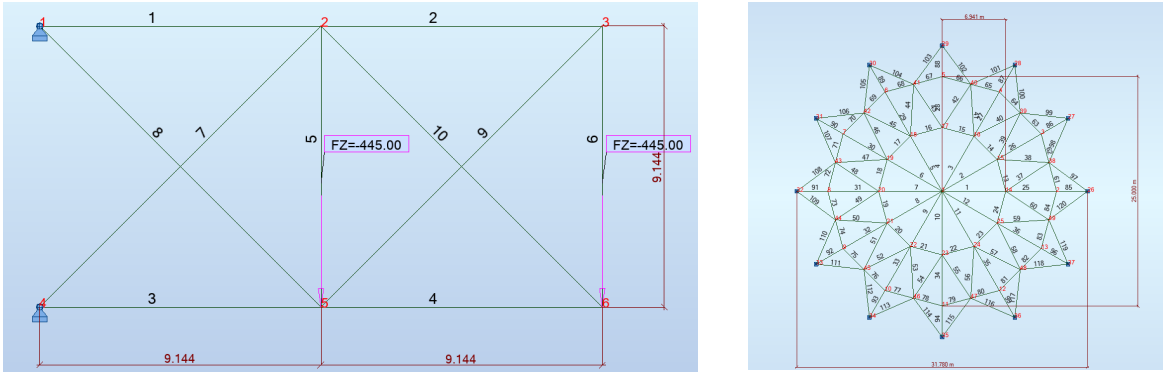
saves the model at a designated location (Figure 1b), and then retrieves the stresses, structural weight, and displacements needed for calculating the penalized objective function from RSA. To rank the design options, two different methods, namely penalty and separation based, from the open literature are obtained. Assessing the penalized objective function of every design solution is essential to rank them in SO while using penalty-based CHT. For this process formulations in [15] have been utilized as demonstrated in Figure 1c. It is also possible to use other techniques such as techniques based on the separation of objective and constraints as has been shown in Figure 1d. These methods are tested on the 10-bar truss to see their effectiveness in the result section. After obtaining the comparison chart, the method that gives better results is utilized to optimize more complex 120-bar truss structure. As can be seen in Figure 1d, Dynamo allows employing Python script in the VP environment. Thus, explained processes are carried out in the VP environment using Python scripts. More detailed information and scripts about the process can be found on the referenced GitHub location [36].

3.3. Structural Optimization

For SO, all parameters defined in VP and TP can be used as design variables in the proposed method. The present study has focused on size optimization, consequently, section indices have been set as a design parameter. Given the non-linear nature of the sizing combinatorial optimization problem, PSO and GA have been utilized as shown in Figure 1d.

4. Results

This study was validated by examining two distinct benchmark problems. The first one is the 10-bar truss example is an established benchmark problem in the subject of SO [15], [16], [37]. The geometry configuration and boundary conditions for this 2D truss are shown in Figure 2a, originated from the referenced paper [15]. A list of 41 element sections, commonly utilized in practice, has been considered as design decision variables to enhance the adaptability of the proposed workflow for practical scenarios. This list was obtained from the American Institute of Steel Construction (AISC) and inserted into RSA. The HSRO (Hollow Structural Round Sections) family was employed regarding optimization. For comprehensive instructions on importing databases and utilizing various section features, the reader is encouraged to review article [38].



(**a**) (**b**)

Figure 2. Trusses for numerical examination (a) 10-bar truss; (b) 120-bar truss loading condition

The last numerical example is the 120-bar dome truss which, was initially addressed by the referenced study [14] and later adopted by various studies to evaluate their methodologies [14], [39], [40]. The configuration, and limitations of this dome are depicted in Figure 2b, have been derived from the referenced manuscript [14]. The BS EN 10219-2:2006 standard was used to choose 27 CHS (Circular Hollow Section) sections that meet the specified limits and are available in real-world data. In addition, the buckling effect for components in compression has been considered. To compute the critic load for buckling, a Python algorithm was constructed using formulations from AISC [41]. Population size and generation number have been determined as 20 through the execution of 5 distinct run cycle for this example. The first part of optimization consists of optimizing the 10-bar truss with two different CHTs and MAs. Population size and generation number have been determined as 20 through the execution of

5 distinct run cycle for each MA and CHT techniques. According to the results presented in Figure 3, GA with penalty method as CHT yields better performance. Herewith, for the 120-bar truss, this method has been utilized.

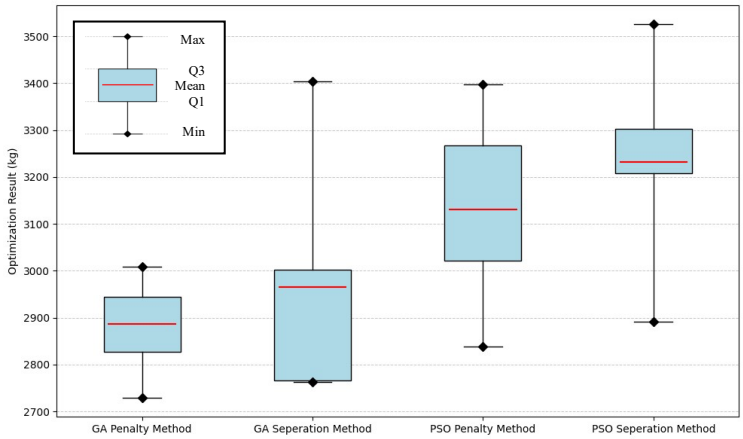


Figure 3. Box chart comparison of 10-bar truss

Ultimately, the results after SO with the proposed method are demonstrated and compared in Table 1. with the manuscripts taken from the literature. In addition to the table, the fitness score change chart and the final version of the best results with sections found have been presented in Figure 4.

Table 1. Comparison of numerical results

	10-Bar Truss				***120-Bar Truss***			
	Rajeev et al. [15]	Camp et al.[37]	Jafari et al. [21]	This Study	*Saka et al.* [14]	*Ebenau et al.* [39]	*Kooshkbaghi et al.* [40]	*This Study*
Weight (kg)	2549.2	2490.6	2302.43	2728.5	*7587*	*6923.3*	*9101.3*	*8054.4*
Analysis made	400	10000	15100	400	-	*20000*	*8400*	*400*
Separate Runs	-	24	30	20	*15*	*100*	*10*	*5*

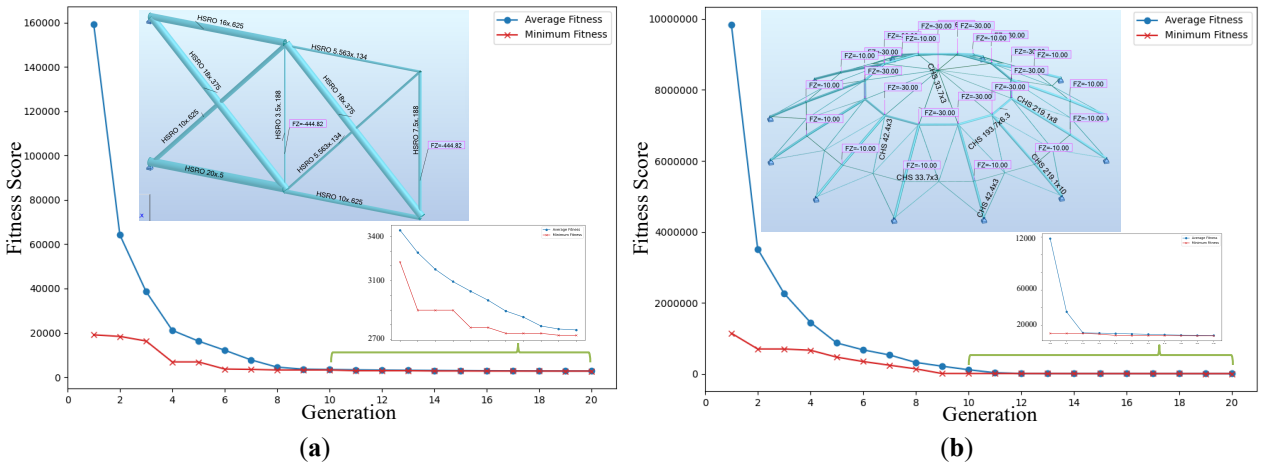


Figure 4. The final version of the trusses along with fitness score change by generation: (a) 10-bar truss; (b) 120-bar truss

4. Conclusion and Discussion

Despite being a well-known topic, SO with meta-heuristic algorithms is not frequently utilized or applied to real-world projects. Here, this problem is approached by using technological advancements, such as Parametric design, and VP in BIM-based projects through Dynamo's ability to become an interoperable bridge between RSA and BIM during the optimization process. Results reveal that the weight obtained after optimization is slightly lower than previous study [7] but still higher than benchmark studies. This is because: (i) discrete and real-world cross-sections selected as design variables, in contrast to the

benchmark problems; (ii) the benchmark studies involved significant amount of more analyses than the present study. Nevertheless, this study was able to achieve within 12% of the best results achieved in both benchmark trusses -on average- with over an order of magnitude less function evaluations and it is considered enough to prove the applicability of proposed workflow.

For future deployments, several enhancements can be made on the study. The main problem is the time required for creating the geometry from the Dynamo plugin directly on RSA for structural analysis. Using the RSA-API significantly reduced the time from 8 hours (for more details, refer to [7]) to 45 minutes per 400 analyses while optimizing the 10-bar truss; however, conducting one analysis for the 120-bar problem with the API still took about 50 seconds. Multi operation techniques can be employed to speed up the process of model creation on RSA using additional API. This will allow the designer to perform more analysis within a shorter time frame, which is expected to improve the results. Moreover, frame structures are not included in this study. Performing SO with different types of optimization techniques (size, shape, topology, layout optimization etc.) on space frames along with multiple types of materials can expand the potential of the proposed workflow.

Overall, the outcomes demonstrated the potential of modern technologies, making it possible for scholars to incorporate programming into their projects by removing the need for a significant amount of syntax knowledge.

Acknowledgements

A big thanks to the Department of Digital Engineering and Construction (DEC) at Karlsruhe Institute of Technology (KIT) for the opportunities that they provide for this research. Also deserving of praise are the members of the Dynamo forum who are constantly trying to help and respond to new questions and debate other options.

References

[1] L. L. Beghini, A. Beghini, N. Katz, W. F. Baker, and G. H. Paulino, "Connecting architecture and engineering through structural topology optimization," *Eng Struct*, vol. 59, pp. 716–726, Feb. 2014, doi: 10.1016/j.engstruct.2013.10.032.

[2] D. Davis and B. Peters, "Design ecosystems: Customising the architectural design environment with software plug-ins," *Architectural Design*, vol. 83, no. 2, pp. 124–131, Mar. 2013, doi: 10.1002/ad.1567.

[3] D. Vermeulen, "Structural Dynam(o)ite-Optimized Design and Fabrication Workflows w/ Dynamo Structural Dynam(o)ite Optimized Design and Fabrication Workflows with Dynamo," 2018. [Online]. Available: www.linkedin.com/in/dietervermeulen

[4] R. Maalek and S. Maalek, "Repurposing existing skeletal spatial structure (SkS) system designs using the Field Information Modeling (FIM) framework for generative decision-support in future construction projects," *Sci Rep*, vol. 13, no. 1, p. 19591, Nov. 2023, doi: 10.1038/s41598-023-46523-z.

[5] S. W. Choi, B. K. Oh, and H. S. Park, "Design technology based on resizing method for reduction of costs and carbon dioxide emissions of high-rise buildings," *Energy Build*, vol. 138, pp. 612–620, Mar. 2017, doi: 10.1016/j.enbuild.2016.12.095.

[6] United Nations Environment Programme, "Global Status Report for Buildings and Construction: : Towards a Zero-emission, Efficient and Resilient Buildings and Construction Sector," Nairobi, 2022. Accessed: Feb. 26, 2023. [Online]. Available: www.globalabc.org.

[7] F. Yavan, R. Maalek, and V. Toğan, "Structural Optimization of Trusses in Building Information Modeling (BIM) Projects Using Visual Programming, Evolutionary Algorithms, and Life Cycle Assessment (LCA) Tools," *Buildings 2024, Vol. 14, Page 1532*, vol. 14, no. 6, p. 1532, May 2024, doi: 10.3390/BUILDINGS14061532.

[8] S. Kirkpatrick, "Optimization by simulated annealing: Quantitative studies," *J Stat Phys*, vol. 34, no. 5–6, pp. 975–986, Mar. 1984, doi: 10.1007/BF01009452/METRICS.

[9] J. H. Holland, "Genetic Algorithms," *Sci Am*, vol. 267, no. 1, pp. 66–73, 1992, [Online]. Available: http://www.jstor.org/stable/24939139

[10] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput Intell Mag*, vol. 1, no. 4, pp. 28–39, Nov. 2006, doi: 10.1109/MCI.2006.329691.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.

[12] T. Dede, M. Kripka, V. Togan, V. Yepes, and R. V. Rao, "Usage of Optimization Techniques in Civil Engineering During the Last Two Decades," 2019.

[13] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, "Metaheuristic Algorithms: A Comprehensive Review," *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, pp. 185–231, Jan. 2018, doi: 10.1016/B978-0-12-813314-9.00010-4.

[14] M. P. Saka and M. Ulker, "Optimum design of geometrically nonlinear space trusses," *Comput Struct*, vol. 41, no. 6, pp. 1387–1396, Jan. 1991, doi: 10.1016/0045-7949(91)90276-R.

[15] S. Rajeev and C. S. Krishnamoorthy, "Discrete Optimization of Structures Using Genetic Algorithms," *Journal of Structural Engineering*, vol. 118, no. 5, pp. 1233–1250, May 1992, doi: 10.1061/(ASCE)0733-9445(1992)118:5(1233).

[16] A. A. Groenwold, N. Stander, and J. A. Snyman, "A regional genetic algorithm for the discrete optimal design of truss structures," *Int J Numer Methods Eng*, vol. 44, no. 6, pp. 749–766, Feb. 1999, doi: 10.1002/(SICI)1097-0207(19990228)44:6<749::AID-NME523>3.0.CO;2-F.

[17] V. Goodarzimehr, U. Topal, A. K. Das, and T. Vo-Duy, "Bonobo optimizer algorithm for optimum design of truss structures with static constraints," *Structures*, vol. 50, pp. 400–417, Apr. 2023, doi: 10.1016/J.ISTRUC.2023.02.023.

[18] Z. Aydın, "Size, layout and tendon profile optimization of prestressed steel trusses using Jaya algorithm," *Structures*, vol. 40, pp. 284–294, Jun. 2022, doi: 10.1016/J.ISTRUC.2022.04.014.

[19] P. Singh, R. Kottath, and G. G. Tejani, "Ameliorated Follow The Leader: Algorithm and Application to Truss Design Problem," *Structures*, vol. 42, pp. 181–204, Aug. 2022, doi: 10.1016/J.ISTRUC.2022.05.105.

[20] F. K. J. Jawad, C. Ozturk, W. Dansheng, M. Mahmood, O. Al-Azzawi, and A. Al-Jemely, "Sizing and layout optimization of truss structures with artificial bee colony algorithm," *Structures*, vol. 30, pp. 546–559, Apr. 2021, doi: 10.1016/J.ISTRUC.2021.01.016.

[21] M. Jafari, E. Salajegheh, and J. Salajegheh, "Optimal design of truss structures using a hybrid method based on particle swarm optimizer and cultural algorithm," *Structures*, vol. 32, pp. 391–405, Aug. 2021, doi: 10.1016/J.ISTRUC.2021.03.017.

[22] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-wesley, 1989.

[23] J. Blank and K. Deb, "Pymoo: Multi-Objective Optimization in Python," *IEEE Access*, vol. 8, pp. 89497–89509, 2020, doi: 10.1109/ACCESS.2020.2990567.

[24] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, pp. 1942–1948, 1995, doi: 10.1109/ICNN.1995.488968.

[25] URL-11, "A Gentle Introduction to Particle Swarm Optimization - MachineLearningMastery.com." Accessed: Aug. 14, 2023. [Online]. Available: https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization/

[26] Z. Han, C. Ning, and Y. Wei, "MOPSO for BIM: a multi-objective optimization tool using particle swarm optimization algorithm on a BIMbased visual programming platform," *Hello, Culture*, pp. 39–51, 2019.

[27] J. Blank and K. Deb, "Pymoo: Multi-Objective Optimization in Python," *IEEE Access*, vol. 8, pp. 89497–89509, 2020, doi: 10.1109/ACCESS.2020.2990567.

[28] D. Savic, "Single-objective vs. Multiobjective Optimisation for Integrated Decision Support," Lugano, Switzerland: International Congress on Environmental Modelling and Software, Jun. 2002.

[29] K. Deb and K. Deb, "Multi-objective Optimization," in *Search Methodologies*, Boston, MA: Springer US, 2014, pp. 403–449. doi: 10.1007/978-1-4614-6940-7_15.

[30] N. D. Lagaros, M. Kournoutos, N. A. Kallioras, and A. N. Nordas, "Constraint handling techniques for metaheuristics: a state-of-the-art review and new variants," *Optimization and Engineering*, vol. 24, no. 4, pp. 2251–2298, Dec. 2023, doi: 10.1007/S11081-022-09782-9/FIGURES/3.

[31] I. Rahimi, A. H. Gandomi, F. Chen, and E. Mezura-Montes, "A Review on Constraint Handling Techniques for Population-based Algorithms: from single-objective to multi-objective optimization," *Archives of Computational Methods in Engineering*, vol. 30, no. 3, pp. 2181–2209, Apr. 2023, doi: 10.1007/S11831-022-09859-9/TABLES/9.

[32] Ö. Yeniay, "Penalty Function Methods for Constrained Optimization with Genetic Algorithms," *Mathematical and Computational Applications*, vol. 10, no. 1, pp. 45–56, Apr. 2005, doi: 10.3390/mca10010045.

[33] J. Liu, K. L. Teo, X. Wang, and C. Wu, "An exact penalty function-based differential search algorithm for constrained global optimization," *Soft comput*, vol. 20, no. 4, pp. 1305–1313, Apr. 2016, doi: 10.1007/s00500-015-1588-6.

[34] T. P. Runarsson and Xin Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294, 2000, doi: 10.1109/4235.873238.

[35] K. Deb, "An efficient constraint handling method for genetic algorithms," *Comput Methods Appl Mech Eng*, vol. 186, no. 2–4, pp. 311–338, Jun. 2000, doi: 10.1016/S0045-7825(99)00389-8.

[36] "ugurfeyzullah/Structural-optimization-with-VP." Accessed: Mar. 10, 2024. [Online]. Available: https://github.com/ugurfeyzullah/Structural-optimization-with-VP

[37] C. V. Camp and B. J. Bichon, "Design of Space Trusses Using Ant Colony Optimization," *Journal of Structural Engineering*, vol. 130, no. 5, pp. 741–751, May 2004, doi: 10.1061/(ASCE)0733-9445(2004)130:5(741).

[38] "Robot Structural Analysis 2018 Help | Section Database | Autodesk." Accessed: Apr. 14, 2024. [Online]. Available: https://help.autodesk.com/view/RSAPRO/2018/ENU/?guid=GUID-727CAC1A-7ABE-4986-B7A5-4E31ADF1A6AA

[39] C. Ebenau, J. Rottschäfer, and G. Thierauf, "An advanced evolutionary strategy with an adaptive penalty function for mixed-discrete structural optimisation," *Advances in Engineering Software*, vol. 36, no. 1, pp. 29–38, Jan. 2005, doi: 10.1016/j.advengsoft.2003.10.008.

[40] M. Kooshkbaghi and A. Kaveh, "Sizing Optimization of Truss Structures with Continuous Variables by Artificial Coronary Circulation System Algorithm," *Iranian Journal of Science and Technology - Transactions of Civil Engineering*, vol. 44, no. 1, pp. 1–20, Mar. 2020, doi: 10.1007/S40996-019-00254-2/FIGURES/17.

[41] AISC, "Specification for the Design, Fabrication & Erection of Structural Steel for Buildings." American Institute of Steel Construction, New York, 1961.