Advances in Engineering Software 135 (2019) 102685

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Optimum design of tied-arch bridges under code requirements using enhanced artificial bee colony algorithm

M.A. Latif, M.P. Saka*

University of Bahrain, Department of Civil Engineering, Isa Town, Bahrain

ARTICLE INFO

Keywords:
Tied-arch bridges
Built-up sections
AASHTO-LRFD provisions
Optimum structural design
Metaheuristic algorithms
Artificial bee colony algorithm
Big bang-big crunch algorithm

ABSTRACT

In this study, optimum design algorithm is presented for tied-arch bridges under AASHTO-LRFD Bridge Design Specifications provisions. It is decided that in tied-arch bridges ties, arch ribs, and bottom and top bracings are made of built-up box sections, whereas built-up I sections are utilized for floor beams and stringers. Bars are adopted for hangers. In the formulation of the optimization problem, design variables are selected as the cross sectional dimensions of steel plates not that of I and box sections. Design pools are prepared for steel plate sections in addition to the hanger bars so that the optimization algorithm can select appropriate steel plates, construct I and box built-up sections for members of 3-D tied-arch such that the weight of the bridge is minimized. In addition to design code requirements, geometrical constraints among its elements that are required for manufacturability of the bridge are also considered. The design process of tied-arch bridges differs from that of steel framed structures. It necessitates consideration of moving vehicle loads. Design optimization algorithms require the response of bridges under several design load arrangements and in the construction of influence lines. This is achieved by using open application programming interface (OAPI) facility of SAP2000. The solution of discrete nonlinear programming problem is obtained by using the proposed Enhanced Artificial Bee Colony algorithm (eABC). The proposed algorithm is compared with Standard Artificial Bee Colony (ABC) and Exponential Big Bang-Big Crunch (eBB-BC) algorithms to evaluate its performance.

1. Introduction

Bridges are integral part of infrastructure system in any country in the world. They are key elements in a transportation system and they control both the volume and the weight of the traffic carried. Bridges are expensive structures and their design and construction require a careful planning not to exceed the limited funds. Bridge designers face the difficult decision making problem of finding the balance among its capacity considering the future prediction of the traffic flow, its cost and the safety [1,2]. Depending on their structural system, bridges can be classified as arch, tied-arch, beam, truss, cantilever, suspension and cable stayed bridges. Among these, the tied-arch bridge is similar to an arch bridge where the outward horizontal forces of the arch are taken by a chord that connects both end of the arch. This chord may be used as deck of the bridge and its self-weight as well as the traffic load is transferred to the arch by hangers. With this arrangement the bridge itself becomes self-supporting structure as shown in Fig. 1. Unlike arch bridges, horizontal compression components are not transferred from the arch to the foundations. This force is transferred to a tensile force and carried by the tie member. The tied-arch bridge has different types of members such as ties, stringers, floor beams, hangers, arch ribs, bottom and arch rib bracings, and concrete deck. Ties are connected to floor beams and floor beams are connected to stringers; altogether, they form the floor system that is braced with bottom bracings. The deck rests on top of the floor system. The arch rib is braced and connected to the ties by the hangers. In general, built-up box sections are used for ties, arch ribs, and bottom and arch ribs bracings, whereas built-up I sections used for floor beams and stringers. Bars considered for hangers. Network arch bridge is a tied-arch bridge with inclined hangers and multiple intersections.

In the literature, there are few papers on the optimum design of tied-arches that are generally based on parametric studies rather than on the use of modern optimization techniques. In [3], a parametric study was carried out to find out the optimum location of cross beam locations. Although bridges were called tied-arch bridges in the study, in reality these bridges were network arch bridges because the hangers were inclined that cross each other at least twice. It is stated that optimum location of the bracing should maximize the collapse load of the bridge. The optimization of hanger arrangement of tied network arch bridges is presented in [4]. Optimization was performed through

* Corresponding author at: University of Bahrain, Civil Engineering Department, P.O. Box: 32038, Bahrain.
E-mail addresses: mpsaka@metu.edu.tr, mpsaka@uob.edu.bh (M.P. Saka).

https://doi.org/10.1016/j.advengsoft.2019.102685
Received 15 October 2018; Received in revised form 12 May 2019; Accepted 1 June 2019

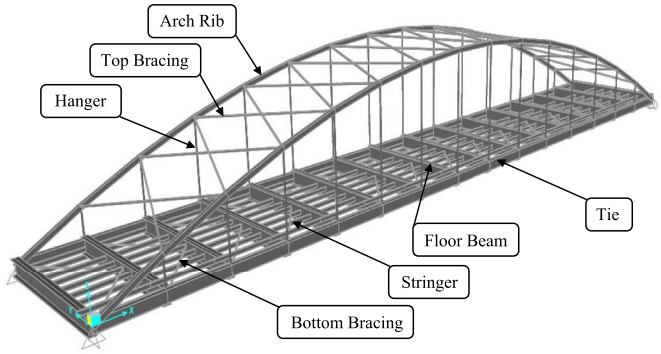

Fig. 1. Tied-arch bridge with all its elements.

execution of a simulator, evaluation of the performance objective, and adjustment of the system parameters in an iterative and directed way. Finite element simulator, ANSYS, performed the structural analysis of the virtual prototype of the model. Evaluation of structural response of the bridge is performed through a global optimization algorithm, named EVOP. The problem is formulated as a mixed integer-discrete nonlinear programming problem. Material cost of superstructure of bridge is the optimum design criteria. The design variables are taken as rise of the arch, number of hangers, cross sectional area of cables of the hangers and hanger arrangement. Constraints derived from maximal hanger forces and stress ranges are considered in the optimization problem. Optimal hanger arrangement of network arch bridge determined from global optimization technique shows significant improvement in structural performance over the bridges with vertical hangers.

Similar research is presented in [5] where the objective was to minimize the cost of superstructure particularly arches and hangers of network arch bridge by optimizing the geometric shape, rise to span ratio, cross section of arch, and the number, arrangement and cross sectional dimensions of hangers. Design constraints are formulated as per design code specifications. The minimum cost design problem turned out to be a nonlinear programming problem having a combination of continuous, discrete, and integer sets of design variables and is subjected to highly nonlinear, implicit and discontinuous constraints. Its solution is attained by using an optimization algorithm called evolutionary programming that is interfaced with finite element analysis software in order to obtain the structural response of the bridge. It is shown that through optimum design it is possible to save 38% to 40% of the total cost. In [6], an optimization model for network arch bridges was proposed where the optimum solution is obtained using a three-step algorithm. The post-tensioning forces in the hangers and the initial strains in the girder and arch that identify the initial configuration are selected as design variables. The weight of the bridge considering the cable system, girder, and arch is minimized. Parametric studies are carried out in terms of cable system configurations on several bridges with different arrangements. The effect of stiffened hangers on the longitudinal in-plane structural behavior of arch bridges was analyzed in [7] for tied-arch bridges. It is stated that stiff hangers are an efficient way to reduce the bending moments at both the arch and the deck. It is also described how stiff hangers with one hinge at the bottom or at the top combine high structural efficiency and ease of execution. Possible drawbacks of stiff hangers are also indicated. A comparative study was carried out on the instability behavior of tied-arch bridges in [8]. Finite element method is used to identify the loading curves and maximum instability strength considering geometric nonlinearity and initial stress configuration. Alternatively, classical buckling analyses are developed to verify how nonlinearities affect the instability strength. In both analyses, a procedure based on “zero displacement method” is carried out to correctly identify the initial status of the structure in presence of dead and permanent loads. Results attained from both procedures are compared to the ones obtained by using current design provisions given in Eurocode. Literature review reveals the fact that most of the research is concentrated on network arch bridges and there is no single paper on the optimum design of tied-arch bridges.

In this study, optimum design algorithm is presented for 3-D tied-arch bridges. The algorithm considers AASHTO-LRFD Bridge Design Specifications [9] provisions as design constraints as well as geometrical constraints between its elements so that manufacturability of the bridge can be satisfied. Design pools are prepared that consist of steel plates with various dimensions which are used to construct the built up I and box steel sections for its members as well as bar diameters for hanger bars. The design algorithm can select appropriate plate dimensions for the construction of box and I-sections for members of 3-D tied-arch such that the weight of the bridge is minimized.

2. Optimum design problem

The optimum design of tied-arch bridges necessitates the selection of built-up I and box sections for its various members in addition to the diameter of bars used for hangers in such a way that the tied-arch bridge with the selected sections and bars should satisfy the serviceability and strength requirements specified by AASHTO-LRFD. Furthermore, some geometrical constraints among the cross sectional dimensions of its members are also required to be satisfied for the constructability of the bridge. There are some other constraints that are related to the plates’ proportions. The built-up sections are obtained by welding steel plates with certain width and thickness to each other so that a box or I-section can be obtained. Consequently, it is necessary to prepare a design pool which contains an array of numbers corresponding to each dimension of a plate element to be used in the construction of these built-up sections shown in Fig. 2. Such design pool is shown in Table 1.

Table 1 illustrates the design pool for each steel plate element that is to be used to construct the built-up section for various types of members in a tied-arch bridge. The numbers in each cell of the table shows the starting values for particular dimension of plate section and the second values is the increment in (cm) and the third value is the last value for that dimension. For example, the design pool for the width of the plate element selected to be used as a flange element in the box section for ties starts from 20 cm goes up to 115 cm with the increment of 5 cm. Namely the pool is [20, 25, 30, 35, …, 110, 115]. The numbers given in all other cells have the same meaning. Furthermore, another pool is also prepared for hangers. Each hanger consists of four bars and the pool for the diameter of each bar varies as (5.25:0.25:10). Namely the size of the pool is 20; starting from 5.25 cm ends at 10 cm with the increment of 0.25 cm; [5.25, 5.50, 5.75,…, 9.50, 9.75, 10]. It should be noticed that each type of element of the tied-arch bridge listed in Table 1 introduces four variables into the design problem which makes 24 design variables altogether. Adding the diameter of the bars used as hanger, the total number of design variables becomes 25.

Optimum design algorithm is expected to specify the width and

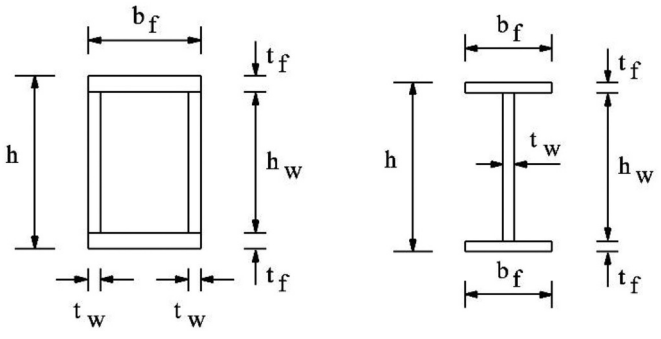

Fig. 2. Built-up box and I sections.

Table 1
Design pools for plate variables.

Variable (cm)	Ties (Box-section)	Stringers (I-section)	Floor beams (I-section)	Arch ribs (Box-section)	Bottom bracing (Box-section)	Top bracing (Box-section)
h	160:10:350	12:1:31	110:10:300	45:5:140	9.5:0.5:19	25.5:3:82.5
b_f	20:5:115	10:1:29	45:5:140	20:5:115	8.5:0.5:18	15.5:0.5:25
t_f	0.95:0.45:9.50	0.80:0.15:3.65	1.10:0.45:9.65	2.65:0.15:5.50	0.80:0.15:3.65	0.80:0.15:3.65
t_w	0.80:0.15:3.65	0.80:0.15:3.65	0.80:0.15:3.65	0.80:0.15:3.65	0.80:0.15:3.65	0.80:0.15:3.65

Table 2
Design variables.

Variable (cm)	Ties (Box-section)	Stringers (I-section)	Floor beams (I-section)	Arch ribs (Box-section)	Bottom bracing (Box-section)	Top bracing (Box-section)
h	I_1	I_5	I_9	I_{13}	I_{17}	I_{21}
b_f	I_2	I_6	I_{10}	I_{14}	I_{18}	I_{22}
t_f	I_3	I_7	I_{11}	I_{15}	I_{19}	I_{23}
t_w	I_4	I_8	I_{12}	I_{16}	I_{20}	I_{24}
	I_{25}: Diameter of bars for hangers					

thickness of web and flange plates for the box or I-sections required for the members of a tied-arch bridges as well as the diameter of bars to be used in hangers. Hence, the optimum design algorithm should be capable of selecting appropriate dimensions of plates for different members and the diameters for the bars used for hangers such that the weight of the bridge is the minimum while AASHTO-LRFD provisions are all satisfied. The formulation of such a design problem yields the following nonlinear discrete programming problem.

Find a vector of integer values $\boldsymbol{I}$ Eq. (1) representing the sequence numbers of dimensions plate sections from the design pool shown in Table 2 that are assigned to different type of members of the bridge

$$\boldsymbol{I}^T = [I_1,\ I_2,\ I_3,\ ...,\ I_{25}\ ] \tag{1}$$

to minimize the weight W of the frame

$$W = \sum_{i=1}^{nm} \ell_i A_i \rho \tag{2}$$

where W is the total weight of steel members (kN), nm is the total number of steel members, ℓ_i is the length of the ith steel member (m), A_i is the area of the ith steel member (m^2), ρ is the density of steel (78.5 kN/m^3).

The constraints that are required to be satisfied after the selection of members for the bridge are:

1 ***Strength Constraints***: applied to ties, stringers, floor beams, hangers, arch rib, and bottom bracing and top bracing according to AASHTO-LRFD *cl.6.8.2.3* and *cl.6.9.2.2*.

lowerRoman%1 **Ties:**

For combined tension and flexural members (*cl.6.8.2.3*), these equations shall satisfy:

$$\frac{P_u}{2P_r} + \frac{M_{ux}}{M_{rx}} + \frac{M_{uy}}{M_{ry}} \le 1.0 \quad for\ \frac{P_u}{P_r} < 0.2 \tag{3}$$

$$\frac{P_u}{P_r} + \frac{8}{9}\left(\frac{M_{ux}}{M_{rx}} + \frac{M_{uy}}{M_{ry}}\right) \le 1.0 \quad for\ \frac{P_u}{P_r} \ge 0.2 \tag{4}$$

where, P_u is the ultimate axial force (kN), P_r is the factored tensile resistance (kN), M_{ux} is the ultimate bending moment about x axis (kN.m), M_{uy} is the ultimate bending moment about y axis (kN.m), M_{rx} is the factored flexural resistance about x axis (kN.m), and M_{ry} is the factored flexural resistance about y axis (kN.m).

i **Stringers:**

The maximum design bending stress should be less than the allowable bending stress:

$$\frac{M_{ux}}{S_x} < F_b \tag{5}$$

where M_{ux} is the ultimate bending moment of the stringer about x axis (kN.m), S_x is the section modulus of the stringer about x axis (m^3), and F_b is the allowable bending stress (kN/m^2).

i **Floor Beams:**

Similarly, for floor beams the maximum design bending stress should satisfy:

$$\frac{M_{ux}}{S_x} < F_b \tag{6}$$

where M_{ux} is the ultimate bending moment of the floor beam about x axis (kN.m), S_x is the section modulus of the floor beam about x axis (m^3), and F_b is the allowable bending stress (kN/m^2).

i **Hangers:**

The design tensile stress of the hangers shall satisfy:

$$\frac{P_u}{A} < F_t \tag{7}$$

where P_u is the ultimate axial load of the hanger (kN), A is the cross sectional area of the hanger (m^2), and F_t is the allowable tensile stress (kN/m^2).

i **Arch Ribs:**

For combined axial compression and flexural members (*cl.6.9.2.2*), the following equations shall apply:

$$\frac{P_u}{2P_r} + \frac{M_{ux}}{M_{rx}} + \frac{M_{uy}}{M_{ry}} \le 1.0 \quad for\ \frac{P_u}{P_r} < 0.2 \tag{8}$$

$$\frac{P_u}{P_r} + \frac{8}{9}\left(\frac{M_{ux}}{M_{rx}} + \frac{M_{uy}}{M_{ry}}\right) \le 1.0 \quad for\ \frac{P_u}{P_r} \ge 0.2 \tag{9}$$

where, P_u is the ultimate axial force (kN), P_r is the factored compressive resistance (kN), M_{ux} is the ultimate bending moment about x axis (kN.m), M_{uy} is the ultimate bending moment about y axis (kN.m), M_{rx} is

the factored flexural resistance about x axis (kN.m), and M_{ry} is the factored flexural resistance about y axis (kN.m).

The webs slenderness should satisfy:

$$\frac{h_w}{t_w} < k\sqrt{E_s/f_a} \tag{10}$$

calculated as per *cl.6.14.4.2*, where h_w is arch rib web height (m), t_w is arch rib web thickness (m), k is the plate stability factor, E_s is the steel modulus of elasticity (kN/m^2), and f_a is the axial stress (kN/m^2).

The flanges width to thickness ratio for flange stability shall comply with:

$$\frac{b_f}{t_f} < 1.06\sqrt{E_s/(f_a + f_b)} \tag{11}$$

calculated according to *cl.6.14.4.3*, where b_f is arch rib flange width (m), t_f is arch rib flange thickness (m), E_s is the steel modulus of elasticity (kN/m^2), f_a is the axial stress (kN/m^2) and f_b is the bending stress (kN/m^2).

i **Bottom and Top Bracings:**

Eqs. (8) and (9) shall satisfy in addition to the equations of the slenderness limits for lateral bracings plates buckling calculated from *cl.6.9.4.2* as follows:

$$\frac{h_w}{t_w} < k\sqrt{E_s/(f_a + f_b)} \tag{12}$$

$$\frac{b_f}{t_f} < k\sqrt{E_s/(f_a + f_b)} \tag{13}$$

where h_w is bracing web height (m), t_w is bracing web thickness (m), b_f is bracing flange width (m), t_f is bracing flange thickness (m), k is the plate buckling coefficient, E_s is the steel modulus of elasticity (kN/m^2), f_a is the axial stress (kN/m^2) and f_b is the bending stress (kN/m^2).

1 ***Displacement Constraints***: applied to make sure that maximum displacement in the bridge would not exceed the bridge span divided by 800 as specified in AASHTO-LRFD *cl.3.4.1* and *cl.2.5.2.6.2*.

The maximum displacement under unfactored loading condition should satisfy *cl.2.5.2.6.2*

$$\delta_{max} < \delta_a \tag{14}$$

where δ_{max} is the maximum deflection (m), and δ_a is the allowable deflection (m) which is equal to the bridge span divided by 800.

1 ***Geometric Constraints***: provided to satisfy the manufacturability of the bridge.

These constraints are checked to make sure that the sections adopted are compatible geometrically as shown in Fig. 3. They include the following constraints:

$$h_{\,(Floor\ Beam)} < h_{w\,(Tie)} \tag{15}$$

$$h_{\,(Stringer)} < h_{w\,(Floor\ Beam)} \tag{16}$$

$$h_{\,(Bottom\ Bracing)} < h_{w\,(Stringer)} \tag{17}$$

$$h_{\,(Top\ Bracing)} < h_{w\,(Arch\ Rib)} \tag{18}$$

$$b_{f\,(Arch\ Rib)} < b_{f\,(Tie)} \tag{19}$$

$$b_{f\,(Tie)} - b_{f\,(Arch\ Rib)} < 0.11 \tag{20}$$

where h, h_w and b_f are the section height, web height and flange width as shown in Fig. 3 and measured in meters. Furthermore, the following constraints are also required for the built-up cross sections proportions limits of webs and flanges in which some of them to be applied in accordance with *cl.6.10.2* and *cl.6.11.2* of AASHTO-LRFD code (Eqs. (22)–(24), (28) and (29)).

For both box and I-sections:

$$b_f < h \tag{21}$$

$$b_f < 24\ t_f \tag{22}$$

$$1.1\ t_w < t_f \tag{23}$$

$$\frac{h_w}{6} < b_f \tag{24}$$

$$2\ t_f < h_w \tag{25}$$

For I-sections:

$$t_w < b_f \tag{26}$$

For box section:

$$2\ t_w < b_f \tag{27}$$

For sections without longitudinal stiffeners (stringers and bracings):

$$\frac{h_w}{t_w} < 150 \tag{28}$$

For sections with longitudinal stiffeners (floor beams and arch ribs) and tension sections that do not require longitudinal stiffeners (ties):

$$\frac{h_w}{t_w} < 300 \tag{29}$$

3. Loading combinations and bridge analysis

The design code AASHTO-LRFD necessitates that each element in the tied-arch bridge should be designed according to the load combinations given in Table 3 where DC is the dead load of structural components and non-structural attachments, LL is the HS20-44 vehicular live load, and WS is the wind load on structure.

The tied-arch bridge considered has four lanes. In 3-D modeling, there are possibilities to load all the lanes or some of the lanes together. For this reason it is necessary to consider 15 arrangements to load the lanes: loading each lane alone (4 arrangements), loading two lanes (6 arrangements), loading three lanes (4 arrangements) and loading all the lanes together (1 arrangement). Determination of the unfavorable live loading condition necessitates construction of the influence lines and surfaces for the bridge. Furthermore, the response of the tied-arch bridge is also required under other loading conditions once cross sectional properties are selected from the design pool. SAP2000 is used for this purpose. SAP2000 provides the facility to allow the users to access the software built-in functions and use them for as per their needs. SAP2000-OAPI functions are compatible with different types of programming languages including visual basic and MATLAB. OAPI stands for Open Application Programming Interface, which consists of set of functions that can be used to interface with other programs to perform different tasks such as creating, modifying, running and analyzing models. This interface is implemented through applying OAPI functions in conjunction with MATLAB in this study. OAPI assures safe and accurate transfer of data importing and assigning information to SAP2000 and extracting analysis results and design parameters from SAP2000. The optimum design algorithm, which is written in MATLAB, starts assigning different plate sections for the bridge members in the model, and then this information is transferred to SAP2000 through OAPI in order to carry out the analysis of the tied-arch bridge under the required loading combinations. The result of analysis (internal forces and displacements) is transferred back to design optimization program which is also written in MATLAB to check whether the selected

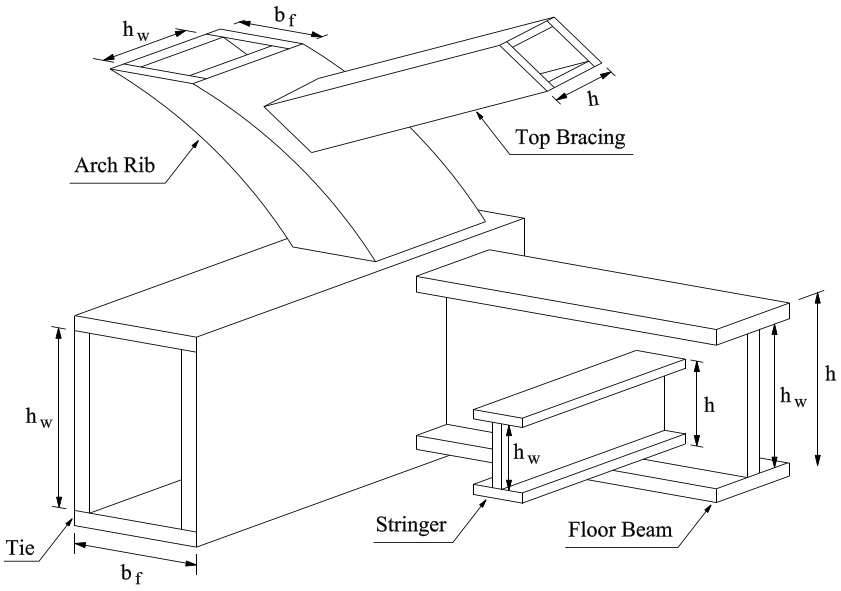

Fig. 3. Tied-arch bridge members' geometric constraints.

combination of sections constitute a feasible or infeasible solution. This process is repeated until the termination criterion is satisfied.

4. Live load analysis

Bridges are subjected to two different types of loading that are permanent and transient loading. Permanent loading are those that do not change with time while the transient loading changes with time. While structural and non-structural loading are example of permanent loads, vehicular load, pedestrian loads and wind loads are example of transient loading.

Live load consideration for bridges differs from that of buildings due to existence of vehicular traffic. The analysis of bridge live loads is complex and affected by several factors such as type of vehicles, bridge span and number of lanes. It requires influence surface calculations for three-dimensional bridge modeling. Truckload as well as a design lane load is needed to be considered for such analysis. Typical HS20-44 truckload defined in AASHTO-LRFD is shown in Figs. 4 and 5. In these figures abbreviation H stands for highway, S stands for semitrailer, 20 is the nominal truck weight in tons and 44 stands for year 1944. These truckloads are called nominal because they do not match any of the existing trucks. They exist theoretically.

AASHTO recommends that live loads are to be defined by combination of the design truckloads and design lane loads. The design lane load is considered as 9.3 kN/m^2 uniformly distributed load of along the design lanes that represents string of vehicles. It should be noticed that it is also distributed transversely over 3 m width as shown in Fig. 6. It also has two concentrated floating loads 80 kN and 116 kN that are placed to produce the maximum moment and shear respectively. Each of them is placed individually with the uniform load to generate the critical outcome [2].

The design lanes on which the live load moves are shown in Fig. 7. The tied-arch bridge considered has four design lanes with 3.6 m width that are defined on the top of the deck system as shown in the figure.

Both truck and lane loads are combined together under one vehicle class and applied to the design lanes in order to find the most severe combinations. There are two types of live load analysis in SAP2000: the influence based enveloping analysis and the step-by-step analysis. The first one is recommended by SAP2000 for most design purposes and is used in this study. In the influence based enveloping analysis the vehicles move along each lane in both directions and the vehicles are automatically located along the length and width of the lanes using the influence surfaces. The combination of design lane loading arrangement is required to be decided by the designer. In this study, 15 design lane-loading arrangements given in Table 4 are considered. These combinations are considered in the preparation of influence surfaces by SAP2000.

According to AASHTO *cl.3.6.1.1.2* multiple presence factors given in Table 5 shall be applied for these arrangements.

After design load arrangements are decided, SAP2000 automatically constructs influence surfaces for these lanes in order to place the vehicles to their maximum effect. These surfaces are interpolated from auto placement of influence unit loads along the length and the width of the lanes. Influence surfaces for bending moment, shear and axial forces for member 240 are shown in Figs. 8–11 as samples.

After computation of influence surfaces, SAP2000 constructs design envelopes for the most critical responses and finds out the maximum and minimum values of internal forces and displacement. Bending moment and shear force envelope diagrams are shown in Figs. 12 and 13.

Table 3
Loading combinations as per *cl.3.4.1*.

Name of combination	Description	Loading combination	Design calculations
Strength I	Related to vehicular normal use without wind load	1.25DC + 1.75LL	Ties, floor beams, stringers and hangers constraints
Strength III	Related to bridges exposed to wind with more than 55 mph wind velocity	1.25DC + 1.40WS	Arch ribs and bottom and top bracings constraints
Service I Live Load	Related to the normal operational use of the bridge with wind velocity of 55 mph. Also related to the deflection control of live loads.	1.00LL	Displacement constraint

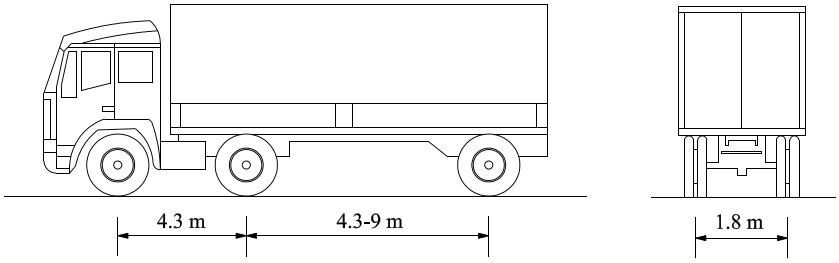

Fig. 4. Configuration of HS20-44 design truck in AASHTO.

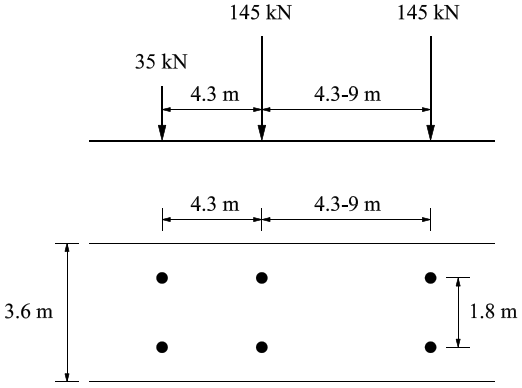

Fig. 5. Elevation and plan view for HS20-44 Vehicle loading.

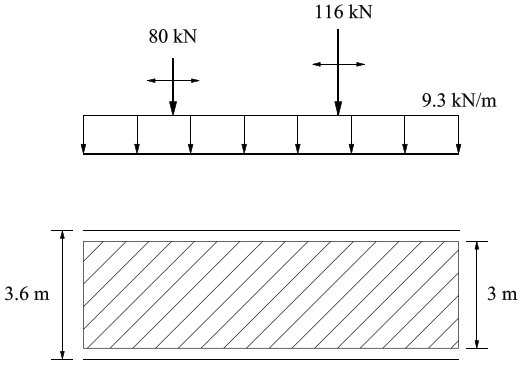

Fig. 6. Elevation and plan view of HS20-44 lane loading.

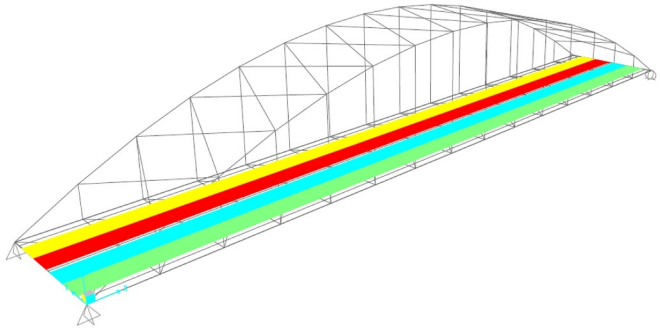

Fig. 7. Tied-arch bridge design lanes.

5. Metaheuristic algorithms

The optimum design problem described in the previous section turns out to be nonlinear discrete programming problem. Sequential linear discrete programming technique among the available mathematical programming algorithms can be used to determine its solution [10]. In this technique, nonlinear expressions are linearized about an

Table 4
Different combinations of design lane loading.

Design lane number	Loading cases 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Lane 1	●				●	●	●				●	●	●		●
Lane 2		●			●			●	●		●	●		●	●
Lane 3			●			●		●		●	●		●	●	●
Lane 4				●			●		●	●		●	●	●	●

Table 5
Multiple presence factors as per AASHTOO *Table 3.6.1.1.2-1*.

Loaded lanes numbers	Multiple presence factor
1	1.20
2	1.00
3	0.85
4 and above	0.65

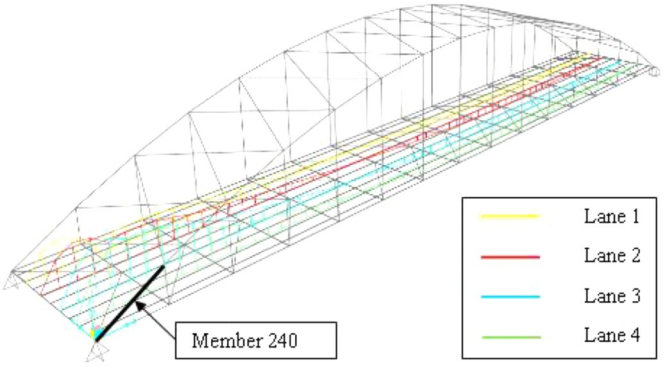

Fig. 8. Influence lines for bending moment for member 240 when all lanes loaded.

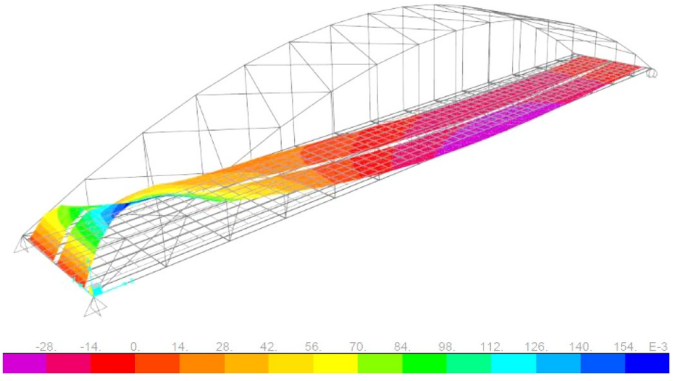

Fig. 9. Influence surface for bending moment for member 240 for all lane loading.

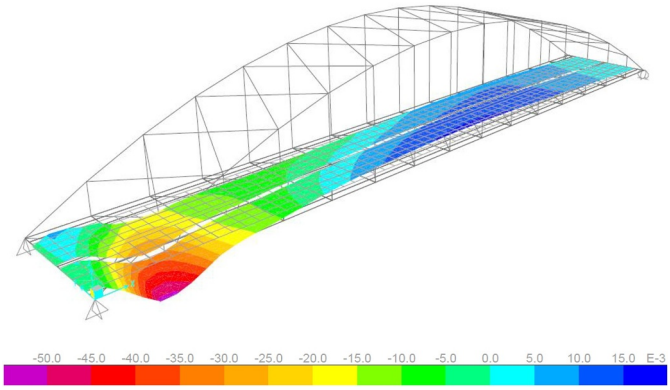

Fig. 10. Influence surface for shear force for member 240 for all lane loading.

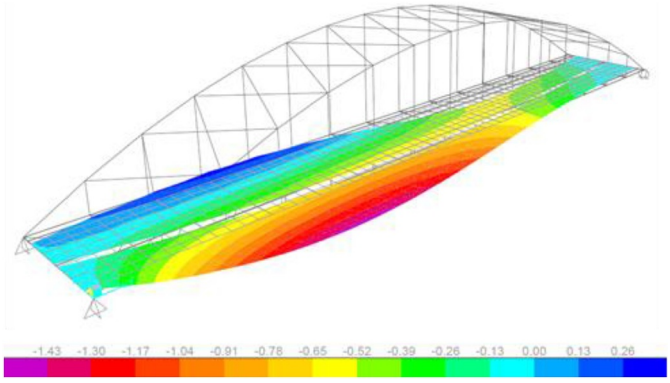

Fig. 11. Influence surface for axial force for member 240 for all lane loading.

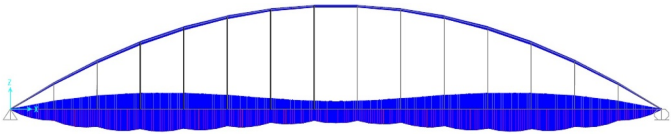

Fig. 12. Bending moment design envelope.

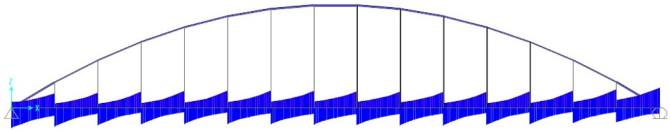

Fig. 13. Shear force design envelope.

initial point using a first-order Taylor's series expansion and linearized problem is solved using integer linear programming techniques. However, in the case of discrete design variables this cannot be carried out because design variables are discrete and non-integer. Therefore, it becomes necessary to introduce additional zero-one variables in order to handle the non-integer-discrete variables, which increases the size of the optimization problem. Furthermore, quality of the initial point has immense effect in finding the optimum solution. Selection of a poor initial point might even cause not to have a convergence during the iterations [11,12]. On the other hand stochastic search methods that are known as metaheuristic techniques are quite efficient in determining the solution of discrete programming problems [13–17]. The fundamental properties of metaheuristic algorithms are that they imitate certain strategies taken from nature, social culture, biology or laws of physics, which are used to direct the search process. Their goal is to explore the search space effectively using these governing mechanisms in order to find near optimal solutions if not global optimum. They also utilize some strategies to avoid being trapped in confined areas of search space. Furthermore, they do not even require an explicit relationship between the objective function and the constraints. They are not problem specific and proven to be very efficient and robust in obtaining the solution of practical engineering design optimization problems with both continuous and non-integer discrete design variables [18–22].

- Constraint handling

Metaheuristic algorithms are developed to obtain the solution of unconstrained optimization problems. However, almost all of the structural design problems are constrained optimization problems. It is apparent that it becomes necessary to transform the constrained optimum design problem into unconstrained one if one intends to use metaheuristic algorithms for obtaining its solution. One way to achieve this is to utilize a penalty function. The following penalty function is a typical expression adopted in large number of studies to achieve this transformation.

$$W_p = W(1 + C)^{\varepsilon} \tag{30}$$

where W is the value of objective function of optimum design problem given in Eqn. (2), W_p is the penalized weight of structure, C is the value of total constraint violations which is calculated by summing the violation of each individual constraint, ε is penalty coefficient which is selected depending on the optimization problem under consideration. Generally, a value of 1 or 2 is advised in the literature [18,19].

$$C = \sum_{i=1}^{nc} c_i \tag{31}$$

$$c_i = \begin{cases} 0 & if \quad g_j \le 0 \\ g_j & if \quad g_j > 0 \end{cases} j = 1, ...,nc \tag{32}$$

where c_i is the ith constraint violation, g_j is the jth constraint function and nc is the total number of constraints in the optimum design problem. It should be pointed out that all constraints in the design problem must be normalized before metaheuristic algorithms are applied.

Among the large amount of metaheuristic techniques available in the literature two population based approached are selected to find the solution of the design problem described from (1) to (29) due to their wide application and proven efficiency in obtaining the solution of discrete structural design optimization problems. These are artificial bee colony and big bang-big crunch algorithms. Furthermore, an enhancement is also suggested to artificial bee colony algorithm to improve its performance.

5.1. Artificial bee colony algorithm (ABC)

The artificial bee colony algorithm was developed by Karaboga and Basturk [23–25]. This algorithm mimics the foraging behavior of honeybee swarms. The algorithm initiates the search process by first forming an artificial bee colony total number of which is decided by the user. The colony is divided in three main groups that are named as employed bees, onlooker bees and scout bees. Each of these groups has different task to execute. The task of *employed bees* is to locate food source, evaluate its amount of nectar and keep the location of better sources in their memory. After they complete their work and fly back to hive they share this information to other bees in the dancing area by dancing. The dancing time represents the amount of nectar in the food source. The *onlooker bees* observe the dance and act accordingly. They might decide to fly to the food source or not depending on whether they find the source worthwhile. The *scout bees* explore new food sources near the hive randomly. The employed bee whose food source has been consumed becomes a scout bee. It is necessary that each employed bee in the colony travels to one food source and no other employed bees should fly to this source. Consequently, the total number of employed

bees in the artificial bee colony algorithm is equal to number of food sources. When the food source is exhausted, onlooker and employed bees of this food source are replaced by scout bees. Then, these bees start searching new food sources by making random search. It is apparent that each food source represents a possible solution for the optimization problem. The nectar amount of a food source symbolizes the quality of the solution, which is identified by its fitness value.

- Standard artificial bee colony algorithm (sABC)

The standard artificial bee colony algorithm consists of four stages. These stages are initialization phase, employed bees phase, onlooker bees phase and scout bees phase. These stages are summarized below as given in [17] for the optimization problem of $Min.\ z = f(\mathbf{x})$ where $\mathbf{x}$ is vector of n design variables:

1 ***Initialization phase***: Initialize all the vectors of the population of food sources, $x_p, p = 1, \ldots, np$ using random search Eq. (33) where np is the population size (total number of artificial bees). Each food source is a solution vector consisting of n variables ($x_{pi}, i = 1, \ldots, n$) is a potential solution to the optimization problem.

$$x_{pi} = x_{\ell i} + rand\ (0,\ 1)(x_{ui} - x_{\ell i}) \tag{33}$$

where $x_{\ell i}$and x_{ui} are upper and lower bounds on x_i.$rand(0,\ 1)$ is a random number between 0 and 1.

1 ***Employed bees phase:*** Employed bees search new food sources by using (34).

$$v_{pi} = x_{pi} + \phi_{pi}(x_{pi} - x_{ki}) \tag{34}$$

where $k \neq i$is a randomly selected food source, φ_{pi} is a random number in range $[-1,1]$. After producing the new food source (solution vector), its fitness is calculated. If its fitness is better than $x_{pi}$the new food source replaces the previous one. The fitness value of the food sources is calculated according to (35)

$$fitness(x_p) = \begin{cases} \frac{1}{1+f(x_p)} & if \quad f(x_p) \geq 0 \\ 1 + abs(f(x_p)) & if \quad f(x_p) < 0 \end{cases} \tag{35}$$

where $f(x_p)$is the objective function value of food source x_p .

1 ***Onlooker bees phase:*** Unemployed bees consist of two groups. These are onlooker bees and scouts. Employed bees share their food source information with onlooker bees. Onlooker bees choose their food source depending on the probability valueP_p, which is calculated using the fitness values of each food source in the population as shown in (36).

$$P_p = \frac{fitness(x_p)}{\sum_{p=1}^{np} fitness(x_p)} \tag{36}$$

After a food source $x_{pi}$for an onlooker bee is probabilistically chosen, a neighborhood source is determined by using Eq. (34) and its fitness value is computed using (35).

Table 6
Tied-arch bridge parameters.

No.	Parameter	Value (m)
1	Bridge height	20
2	Bridge span	126
3	Bridge width	20
4	Spacing between floor beams	8.4
5	Spacing between stringers	2.15
6	Slab thickness	0.15

Table 7
Number of members in tied-arch example.

No.	Type of member	Number of members
1	Ties	30
2	Stringers	135
3	Floor beams	16
4	Hangers	28
5	Arch ribs	30
6	Bottom bracings	30
7	Top bracings	26
	Total number of members	295

Table 8
Adopted values for algorithm parameters.

Algorithm	Population	Other parameters
ABC	20	Limit = 250
eABC	20	Limit = 250
eBB-BC	20	Alpha = 0.25

1 ***Scout bees phase:*** The unemployed bees who choose their food sources randomly called scouts. Employed bees whose solutions cannot be improved after predetermined number of trials become scouts and their solutions are abandoned. These scouts start to search for new solutions by using random search.

The artificial bee colony algorithm is shown to be competitive to other population-based algorithms such as genetic algorithm and particle swarm optimizer [26]. However, it is reported that while its performance is good at exploration but poor at exploitation [27]. Several variants of artificial bee colony algorithm are developed in order to remedy this insufficiency [28–36]. The use of artificial bee colony algorithm in the development of structural optimization algorithms is documented in [37–42].

- Enhanced artificial bee colony algorithm (eABC)

In standard artificial bee colony algorithm onlooker bees carry out exploitation (local search). Design variables i and k are selected randomly and the new value for the design variable i is calculated by using Eq. (34). This newly calculated value is required to be within the upper and lowers bounds of the design variable i. In case this holds and in the meantime the fitness value of this new design variable is less than the fitness value of current design variable i, then the newly calculated value replaces the current value of the design variable i. The success of this strategy heavily depends on the value of design variable k. If the value of design variable k is such that it yields a better solution when it is used in Eq. (34) with design variable i, then local search is successful. However most of the time it does not. For this case, there is no remedy in the standard artificial bee colony algorithm. It just keeps the old value of the design variable. Hence, the iteration is wasted computationally without producing a better value. In this study, the following enhancement is suggested for the local search.

Table 9
Optimum results obtained by three metaheuristic algorithms for different seed values.

Algorithm	ABC			eABC			eBB-BC		
Seed value	Penalized weight (kN)	Value of penalty term		Penalized weight (kN)	Value of penalty term		Penalized weight (kN)	Value of penalty term	
0	6441.44	0		6734.67	0		6460.83	0	
1	6669.27	0		**6273.19**	0		**6383.62**	0	
2	7054.92	0		6403.72	0		6642.71	0	
3	6650.47	0		6481.48	0		6727.93	0	
4	6820.44	0		6924.79	2.2×10^{-16}		6453.08	0	
5	6519.68	2.2×10^{-16}		6557.31	0		6638.61	0	
6	6764.30	0		6614.35	0		6700.05	7.2×10^{-6}	
7	6744.87	0		6665.10	0		6945.17	0	
8	6530.87	0		6491.70	0		6542.34	0	
9	**6375.34**	0		6806.24	0		6535.48	0	
Best	6375.34	No. of adopted solutions	10	6273.19	No. of adopted solutions	10	6383.62	No. of adopted solutions	10
Worst	7054.92			6924.79			6945.17		
Median	6659.87			6585.83			6590.48		
Average	6657.16			6595.26			6602.98		
STD	201.40			195.31			164.66		

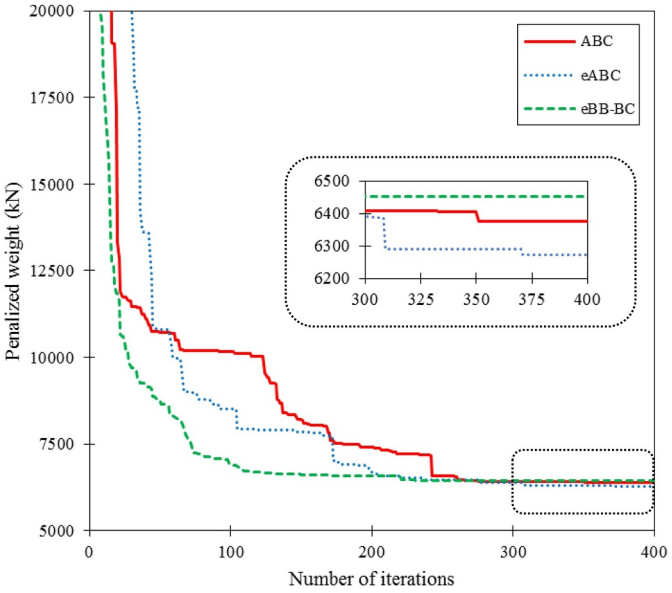

Fig. 14. Design histories of best solutions obtained by three metaheuristic algorithms.

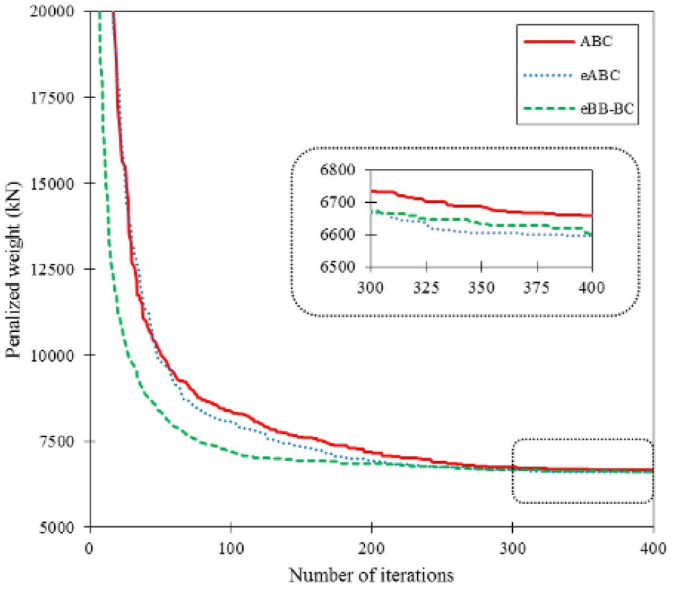

Fig. 15. Design histories of average solutions obtained by three metaheuristic algorithms.

$$
\begin{aligned}
&\textit{if} \quad f(x_{pi}) \quad < \quad f(x_{ki}) \\
&\quad v_{pi} = x_{pi} + \phi_{pi}\,(x_{pi} - x_{ki}) \\
&\textit{else} \\
&\quad v_{pi} = x_{ibest} \\
&\textit{end}
\end{aligned} \tag{37}
$$

where $k \neq i$ is a randomly selected food source, $f(x_{pi})$and $f(x_{ki})$ are the corresponding objective function values for design variables i and k, φ_{pi} is a random number in range $[-1,1]$, $x_{ibest}$is the best value of the design variable i attained until this iteration. The enhanced artificial bee colony algorithm follows the steps of standard artificial bee colony algorithm listed in the previous section; it uses Eq. (37) instead of Eq. (34) when local search is unsuccessful.

5.2. Big Bang-Big Crunch algorithm (BB-BC)

Big Bang-Big Crunch algorithm was developed by Erol and Eksin [43], which mimics the evolution of the universe. The Big Bang theory attempts to explain how the universe developed from a very tiny, dense state into its current state. The energy released during the big bang causes the expansion of the universe. The Big Crunch theory on the other hand suggests that this expansion could not be forever because of mutual gravitational attraction of matters, which would eventually reverse the expansion and cause the universe to contract.

The Big Bang-Big Crunch algorithm consists of two phases as its name implies similar to Big Bang and Big Crunch theory. First phase is the big bang in which the randomly generated candidate solutions are randomly distributed in the search space. In the second phase, these points are contracted to a single representative point that is the weighted average of randomly distributed candidate solutions.

- Standard BB-BC algorithm (sBB-BC)
  1 Decide the size of initial population *np* and generate initial population randomly similar to other evolutionary algorithms. These candidate solutions are scattered in the design domain. This is called the big bang phase.
  2 Apply big crunch operation. This operation takes the current position of each candidate solution in the population and its associated penalized fitness function value and computes a center of mass as given in (38). The center of mass is the weighted average of the candidate solution positions with respect to the inverse of the penalized fitness function values.

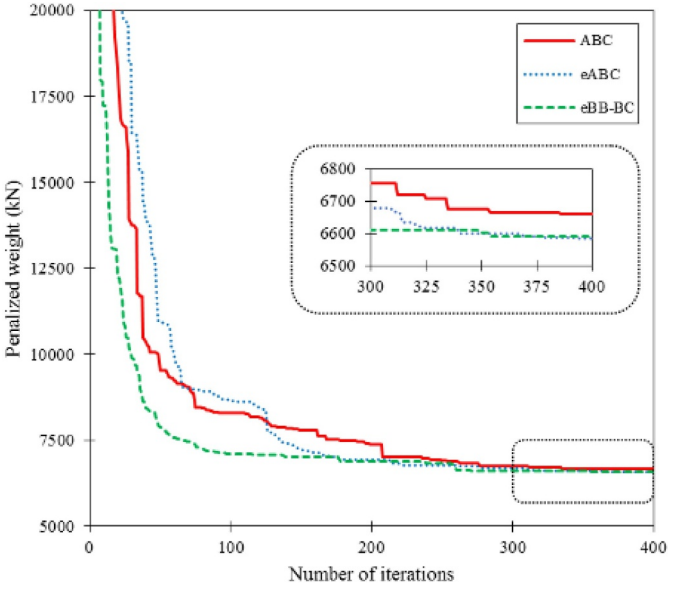

Fig. 16. Design histories of median solutions obtained by three metaheuristic algorithms.

Table 10
Optimum dimensions of plate elements for different members of tied-arch bridge.

Member type	h (cm)	b_f (cm)	t_f (cm)	t_w (cm)
Tie	240.00	55.00	2.30	0.80
Stringer	16.00	10.00	0.95	0.80
Floor beam	230.00	75.00	5.60	0.80
Arch rib	80.00	55.00	2.65	1.85
Bottom bracing	13.50	11.50	0.95	0.80
Top bracing	28.50	18.00	1.40	0.80
Hanger	4Φ7.00 (cm)			

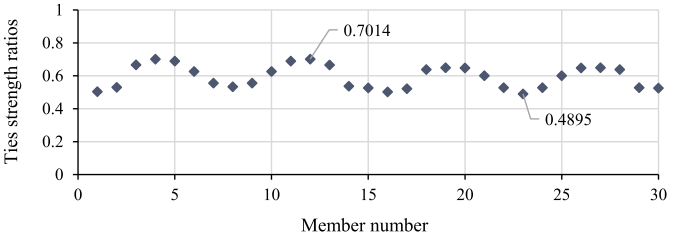

Fig. 17. Strength ratios for tie members.

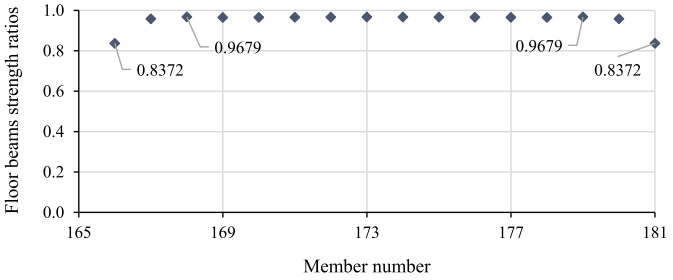

Fig. 18. Strength ratios for floor beams.

$$x_i^{cm} = \left[\sum_{j=1}^{np} \frac{x_{i,j}}{f_j}\right] \div \left[\sum_{j=1}^{np} \frac{1}{f_j}\right] \quad (38)$$

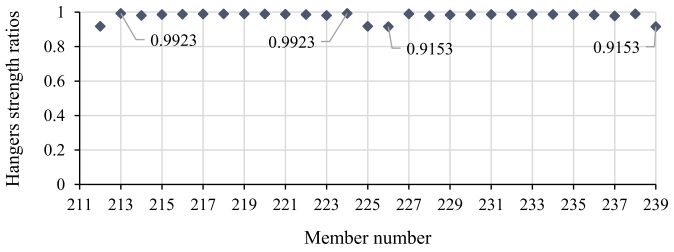

Fig. 19. Strength ratios for hangers.

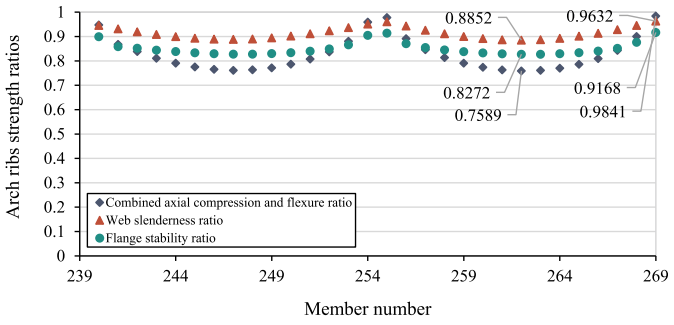

Fig. 20. Strength ratios for arch rib elements.

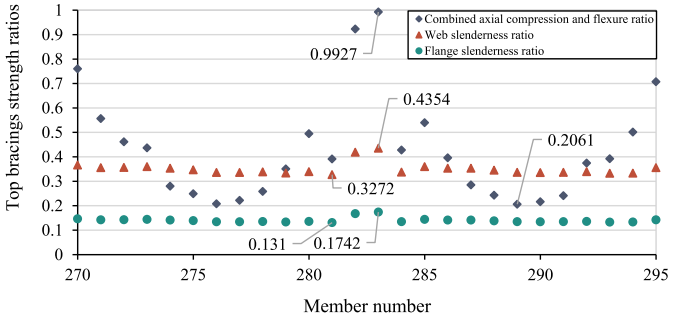

Fig. 21. Strength ratios for top bracing elements.

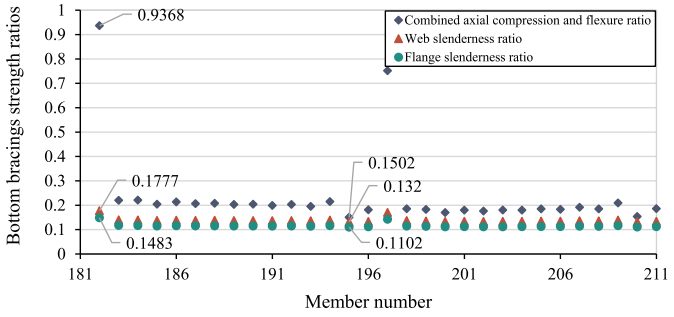

Fig. 22. Strength ratios for bottom bracing elements.

where x_i^{cm} is the ith component of the center of mass point at any iteration, $x_{i,j}$ is the ith component of the jth candidate solution at any iteration in n dimensional search space, f_j is the penalized fitness function value of candidate solution j and np is the size of the initial population.

1 Treat the current center of mass as a hub for the next big bang. Compute the new position of candidate solutions in the next iteration using the following expression considering that they are normally distributed around the center of mass x_i^{cm}

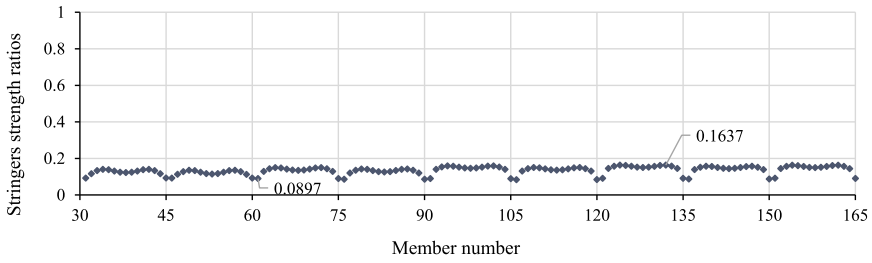

Fig. 23. Strength ratios for stringers.

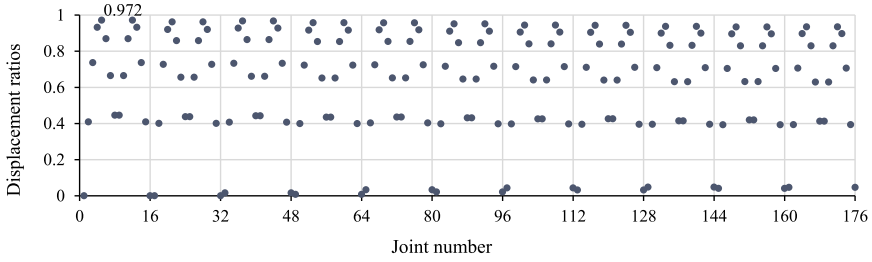

Fig. 24. Displacement ratios of joints.

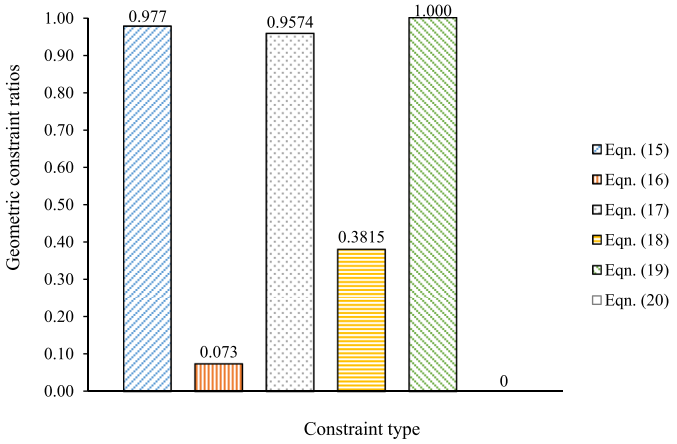

Fig. 25. Ratios for geometric constraints for different member types connecting together.

$$x_i^{next} = x_i^{cm} + \frac{r\alpha\,(x_i^{\max} - x_i^{\min})}{s} \tag{39}$$

where x_i^{next} is the position of the new candidate solution i, r is the random number from a standard normal distribution, α is the parameter that limits the size of the search space, $x_{\max}$ and $x_{\min}$ are the upper and lower bounds on the design variables and s is the iteration number. In the case where x_i^{next} is to be selected from a discrete set then it becomes necessary to work with integer numbers. In this case x_i^{next} is calculated as

$$I_i^{next} = I_i^{cm} + round\left[\frac{r\alpha\,(I_i^{\max} - I_i^{\min})}{s}\right] \tag{40}$$

where I_i^{cm} the value of ith discrete design variable in the fittest individual and $I_i^{\min}$ and $I_i^{\max}$ are its lower and upper bounds.

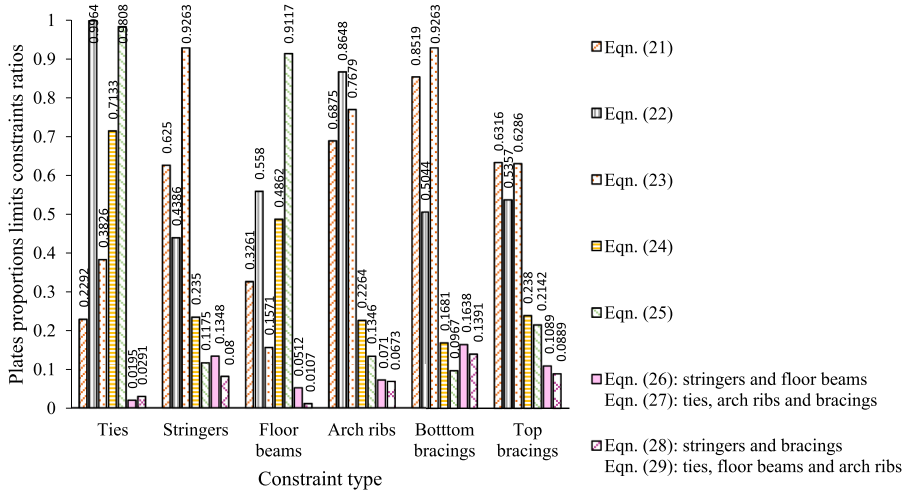

Fig. 26. Ratios for plates' proportions limits constraints.

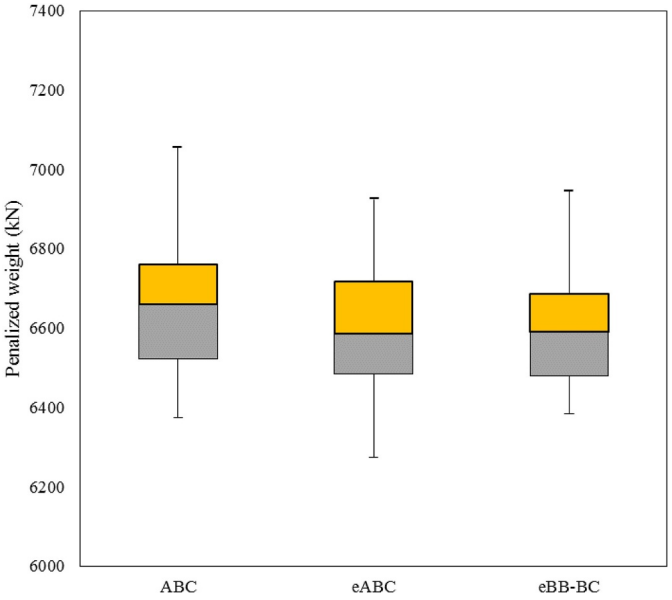

Fig. 27. Boxplot of final distribution of the optimum results attained by each algorithm.

Table 11
Total number of fitness evaluations and CPU time for each execution.

Algorithm	ABC	eABC	eBB-BC
CPU time (hours)	134	133	133
Number of fitness evaluations	8010	8010	8000

Table 12
The ranks of the three metaheuristic algorithms applied in Friedman Test calculations.

Algorithm Seed value	ABC Penalized Weight (kN)	Rank	eABC Penalized Weight (kN)	Rank	eBB-BC Penalized Weight (kN)	Rank
0	6441.44	1	6734.67	3	6460.83	2
1	6669.27	3	**6273.19**	1	**6383.62**	2
2	7054.92	3	6403.72	1	6642.71	2
3	6650.47	2	6481.48	1	6727.93	3
4	6820.44	2	6924.79	3	6453.08	1
5	6519.68	1	6557.31	2	6638.61	3
6	6764.30	3	6614.35	1	6700.05	2
7	6744.87	2	6665.10	1	6945.17	3
8	6530.87	2	6491.70	1	6542.34	3
9	**6375.34**	1	6806.24	3	6535.48	2
Avg. Rank		2.00		1.70		2.30

Table 13
Friedman Test results.

Calculated Chi Square Value	**1.80**
Critical Chi Square Value	5.99
Mean Rank Between Algorithms	2.00

1 Repeat steps 2 and 3until termination criteria is satisfied.

Big-bang big-crunch algorithm is used to develop structural optimization algorithms for various types of structures such as space trusses [44,45], Schwedler and ribbed domes [46] and swaying frames [47]. Performance evaluation of big-bang big crunch algorithm on benchmark engineering optimization problem is carried out in [48]. Enhancements to big bang-big crunch algorithm to improve its performance is suggested in [49] and hybrid differential evolution and big bang-big crunch algorithm is developed in [50].

- An exponential big bang-big crunch algorithm (eBB-BC)

It is shown in the literature that metaheuristic algorithms are general and they are capable of attaining the optimum solution of a wide range of various engineering design optimization problems. However, it is also revealed in the literature that their performances differ depending on the unique feature of each design problem. In order to achieve the best performance of any metaheuristic algorithm certain adjustments may be necessary. It is reported in [49] that the standard BB-BC algorithm shows poor performance in discrete design optimization of steel frame structures due to ineffective manipulations of its two parameters search dimensionality and step size. It is stated that in discrete optimization both average search dimensionality ratio and average step size parameters tend to become zero after certain number of iteration. As a result of this the algorithm would not be able to generate a new solution and continues replicating the former ones. Noticing this drawback of the standard algorithm, Eq. (40) is replaced with the following expression.

$$I_i^{next} = I_i^{cm} + round\left[\frac{r_i^n \alpha\,(I_i^{\max} - I_i^{\min})}{s}\right] \tag{41}$$

In this new formulation the use of *n*th power a random number is inspired from statistical distribution where n≥ 2. This new expression eliminates diminishing of search dimensionality in the beginning stages of the search process and increases the same parameter towards the latest stages. Furthermore, it enables large step size from time to time at later optimization stages which prevents the algorithm to getting stuck in the local optimum. Expressions (40) and (41) are replaced by selecting the type of statistical distribution used to sample the random number instead of normal distribution where *n* is taken as 3. Hence, Eq. (41) has changed into the following

$$I_i^{next} = I_i^{cm} \pm round\left[\alpha \times E(\lambda = 1)_i^3 \frac{(I_i^{\max} - I_i^{\min})}{s}\right] \tag{42}$$

which is referred to as exponential BB-BC; namely eBB-BC. The use of an exponential distribution (E) together with the third power of random number is found effective in the discrete design optimization of steel frame structures [49]. The probability density function for an exponential distribution is given as follows.

$$f(x) = \begin{cases} \lambda e^{-x} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{43}$$

where λ is a real, positive constant. The mean and variance of the exponential distribution are given as $1/\lambda$ and $1/\lambda^2$, respectively. It is reported that λ is taken equal to one. The basic difference between the normal distribution and the exponential distribution is that while the former generates positive and negative real numbers, the later only generates positive numbers. Consequently, the rounded term on the right hand side of Eqn. (42) could be added to or subtracted from I_i^{cm}under equal probability to allow for both increase and decrease in the value of a design variable. The efficiency of these modifications is verified on number of design examples in [49]. This algorithm is adopted for attaining the optimum solution of the design problem described in (1) to (29).

6. Design example

The tied-arch bridge shown in Fig. 1 is considered as a design example. The geometric dimensions of the bridge are given in Table 6. The tied-arch bridge consisting of members listed in Table 7 is modeled as 3-D structure in SAP2000 Software, which consists of 295 members. This bridge is taken from [51].

The bridge is simply supported and all members are considered to

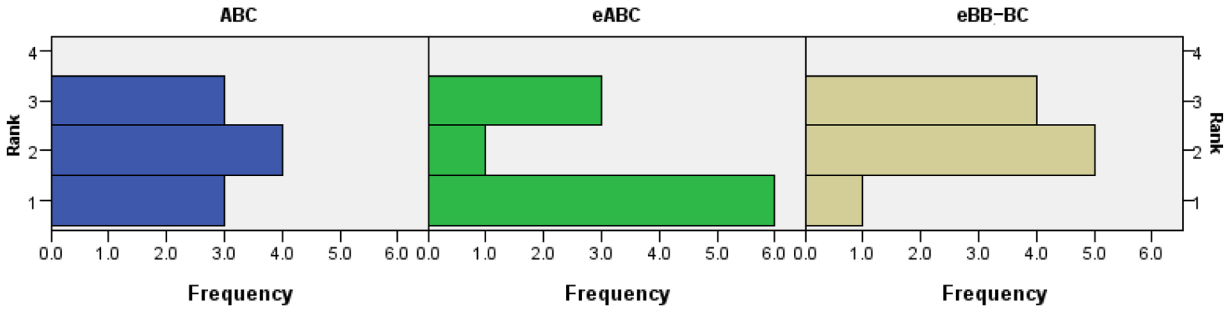

Fig. 28. Ranking frequency for each algorithm.

be rigidly connected. The steel members are modeled as frame elements while the deck is modeled as shell element. The steel grade of A588 is considered for ties, A514 for arch ribs, and A36 for the remaining sections. It is assumed that the floor beams are stiffened with longitudinal and transverse stiffeners and that the arch ribs sections are stiffened with longitudinal stiffeners. The size of the required stiffeners can be decided later by the designer after obtaining the optimum members' sections dimensions.

The tied-arch bridge is designed three times by using three metaheuristic algorithms; namely artificial bee colony algorithm (ABC), enhanced artificial bee colony algorithm (eABC) and exponential big bang-big crunch algorithm (eBB-BC). Maximum number of iterations is taken as 400. The selected values of parameters for each metaheuristic algorithm are given in Table 8. Each metaheuristic algorithm is repeated ten times with ten different seed values. The minimum weights of the tied-arch bridge attained in each run are shown in Table 9. The design histories of each metaheuristic algorithm for the best, average and median solutions are illustrated in Figs. 14–16.

It is apparent that from Table 9 and Figs. 14–16 that the enhanced artificial bee colony algorithm yields the lightest tied-arch bridge with 6273.19 kN weight which is 1.63% less than the optimum weight obtained by artificial bee colony algorithm and 1.76% less than the one attained by exponential big bang-big crunch algorithm. The optimum dimensional values of plate elements for the optimum solution of 6273.19 kN are listed in Table 10. Plate elements with these dimensions are to be used to construct the box and I-sections for various members of the tied-arch bridge that are shown in Table 2.

The strength constraints ratios for ties, floor beams, hangers, arch rib, top bracings, bottom bracings and stringers are shown in Figs. 17–23. Displacement ratios are given in Fig. 24. Geometric constraints ratios for members and plates are shown in Figs. 25 and 26.

Inspection of Figs. 17–24 reveals the fact that the strength constraint ratios of floor beam, hangers, arch rib, top bracing and bottom bracing are all close to one that indicates that they are fully stressed. Furthermore, the displacement constraints are also very close to their upper limit. The same is also observed in Figs. 25 and 26 for geometric constraints for different member types and plate's proportions limits. Therefore, it is apparent that the optimum design provides most economical material use. It is also found that the total weight obtained in this research (6273.19 kN) is less than the one obtained from the literature (7000.3 kN) [51] which results in 10.37% material saving.

A boxplot of final distribution of optimum results by each algorithm given in Table 9 is shown in Fig. 27. The boxplot shows that the distribution of the results for the three algorithms is uniform. It also shows that the median of eABC and eBB-BC are close to each other and better than standard ABC.

Total fitness evaluations and CPU time for each execution is given in Table 11. Total number of fitness evaluations in ABC and eABC is slightly larger than for eBB-BBC. However, the difference is not significant compared to the total number of structural analysis.

Friedman test was applied for multiple comparison as explained in references [52–54] in order to classify the statistical significance among the three metaheuristic algorithms. The null hypothesis states that the distribution of ABC, eABC and eBB-BC are the same. In order to apply the test, the algorithms results are ranked as shown in Table 12. The final results of Friedman Test are presented in Table 13. This table shows that the value of the Friedman's Chi Square is 1.80, which is less than the critical value 5.99. This means that the null hypothesis is accepted thus the distribution of the three algorithms is the same with a mean rank of 2.00. The critical value of Chi Square is obtained from the Chi Square distribution table available in statistics books considering 0.05 level of significance.

While eBB-BC showed a better convergence history as shown in Figs. 14–16, eABC had the lower average ranking of 1.70. This means that eABC has the lowest value in our experiments in terms of finding the minimum weight that leads to the most economical design. The following figure shows the ranking frequency per each algorithm (Fig. 28).

7. Conclusions

The optimum design algorithm developed in this study for tied-arch bridges determines the dimensions of steel built-up box and I-sections for its different members under AASHTO-LRFD provisions. The new algorithm differs from the previous ones on the point that it does not make selection directly from the design pool of available steel sections but it selects plate elements from a design pool that are used indirectly in the construction of built-up steel sections. The mathematical modeling of such optimum design problem leads to formulate a highly constrained discrete non-linear programming problem. Three different metaheuristic algorithms are employed to determine its solution so that the optimum designs accomplished can be compared.

The design algorithm models tied-arch bridge as a three-dimensional model that necessitates the consideration of design lane loading in order to construct influence surface of internal actions. Unfavorable loading cases for traffic loading are based on the influence surfaces that cannot be carried out in two-dimensional modeling. Therefore, the design algorithm uses realistic modeling in the optimum design of tied-arch bridges. On the other hand, the construction of these influence surfaces for internal actions requires structural analysis of the bridge under several different loading conditions that is computationally quite expensive. SAP2000-OAPI provides automated solution for such a task. The metaheuristic algorithms are coded in MATLAB and SAP2000 is used through OAPI to obtain the response of the bridge whenever it is needed. Among the metaheuristic algorithms employed to find the optimum solution, the proposed enhanced artificial bee colony algorithm showed lowest best, mean, average and worst values of final solution than the other two that are exponential big bang-big crunch and standard artificial bee colony algorithms in finding the minimum weight. Friedman test has shown that the distribution of the three algorithms is the same. Although eBB-BC showed a better convergence history, eABC has the lower mean ranking of 1.70. Overall, it is shown that the optimum design problem of tied-arch bridges under AASHTO-LRFD provisions can be modeled realistically considering the influence

surfaces and its solution can be obtained by means of metaheuristic algorithms.

Acknowledgment

This paper is based on a research supported by The University of Bahrain, Deanship of Graduate Studies and Scientific Research Funding (Research Grant project 2014/09) which is gratefully acknowledged. Authors also extend their appreciation and thanks to Dr. S. K. Azad for providing MATLAB code for eBB-BC.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.advengsoft.2019.102685.

References

[1] Barker RM, Puckett JA. Design of highway bridges-an LRFD approach. USA: John Wiley & Sons; 2013.
[2] Taly N. Highway bridge superstructure engineering - LRFD approaches to design and analysis. USA: CRC Press; 2015.
[3] Ney L, Goyet V, Maquoi R. Optimum bracing design of the arches of tied-arch bridges. J Constr Steel Res 1991;18(3):239–49.
[4] Islam N, Ahsan R. Optimization of hanger arrangement of network arch bridges. Proceedings of IABSE-JSCE joint conference on advances in bridge engineering-II. 2010. p. 107–19. August 8-10.
[5] Islam N, Rana S, Ahsan R, Ghani SN. An optimized design of network arch bridge using global optimization algorithm. Adv Struct Eng 2014;17(2):197–210.
[6] Bruno D, Lonetti P, Pascuzzo A. An optimization model for design of network arch bridges. Comput Struct 2016;170:13–25.
[7] Garcia-Guerroro JM, Jorquera-Lucerga JJ. Effect of stiff hangers on the longitudinal structural behavior of tied-arch bridges. Appl Sciences 2018;8(2):258.
[8] Lonetti P, Pascuzzo A, Aiello S. Instability design in tied-arch bridges. Mech Adv Mater Struct 2018. https://doi.org/10.1080/15376494.2017.1410911.
[9] AASHTO-LRFD. Bridge design specifications. 5th ed Washington: American Association of State Highway and Transportation Officials; 2010.
[10] Rao SS. Engineering optimization: theory and practice. 4th ed. John Wiley & Sons; 2009.
[11] Saka MP. Optimum design of steel frames using stochastic search techniques based on natural phenomena: a review. In: Topping BHV, editor. Chapter 6 in civil engineering computations: tools and techniques. Saxe-Coburgh Publications; 2007. p. 105–47.
[12] Saka MP, Geem ZW. Mathematical and metaheuristic applications in design optimization of steel frame structures: an extensive review. Math Prob Eng 2013;2013:271031.
[13] Yang X-S. Nature-inspired metaheuristic algorithms. Luniver Press; 2008.
[14] Yang X-S. Engineering optimization: an introduction with metaheuristic applications. John Wiley; 2010.
[15] Lamberti L, Pappalettere C. Metaheuristic design optimization of skeletal structures: a review. Comput Technol Rev 2011;4:1–32.
[16] Saka MP. Recent developments in metaheuristic algorithms: a review. Comput Technol Rev 2012;5:31–78.
[17] Saka MP, Dogan E, Aydogdu I. Review and analysis of swarm-intelligence based algorithms, chapter 2 in swarm intelligence and bio-inspired computation, theory and applications. In: Yang X-S, Cui Z, Xiao R, Gandomi AH, Karamanoglu M, editors. Elsevier; 2013. p. 25–47.
[18] Hasançebi O, Çarbaş S, Doğan E, Erdal F, Saka MP. Performance evaluation of metaheuristic search techniques in the optimum design of real size pin jointed structures. Comput Struct 2009;87(5–6):284–302.
[19] Hasançebi O, Çarbaş S, Doğan E, Erdal F, Saka MP. Comparison of non-deterministic search techniques in the optimum design of real size steel frames. Comput Struct 2010;88(17–18):1033–48.
[20] Lagaros ND. A general purpose real-world structural design optimization computing platform. Struct Multidiscipl Optim 2014;49(6):1047–66.
[21] Yang X-S, Bekdaş G, Nigdeli SM, editors. Metaheuristics and optimization in civil engineering, modeling and optimization in science and technologies 7. Springer; 2016.
[22] Kaveh A. Applications of metaheuristic optimization algorithms in civil engineering. 2nd ed. Springer; 2017.
[23] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 2007;39:459–71.
[24] Karaboga D, Basturk B. On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 2008;8:687–97.
[25] Karaboga D, Akay B. A modified artificial bee colony (ABC) algorithm for constrained optimization problems. Appl Soft Comput 2011;11:3021–31.
[26] Zhu G, Kwong S. Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl Math Comput 2010;217:3166–73.
[27] Kang F, Li J, Ma Z, Li H. Artificial bee colony algorithm with local search for numerical optimization. J Softw 2011;6:490–7.
[28] Gao W, Liu S. A modified artificial bee colony algorithm. Comput Oper Res 2012;201(39):687–97.
[29] Wu B, Qian C, Ni W, Fan S. Hybrid harmony search and artificial bee colony algorithm for global optimization problems. Comput Math Appl 2012;64(8):2621–34.
[30] Gao W, Liu S, Huang L. A global best artificial bee colony algorithm for global optimization. J Comput Appl Math 2012;236:2471–753.
[31] Li G, Niu P, Xiao X. Development and investigation of efficient artificial bee colony algorithm for numerical function optimization. Appl Soft Comput 2012;12:320–32.
[32] Kang F, Li J, Li H. Artificial bee colony algorithm and pattern search hybridized for global optimization. Appl Soft Comput 2013;13:1781–91.
[33] Gao W, Liu S, Huang L. A novel artificial bee colony algorithm with powell's method. Appl Soft Comput 2013;13:3763–75.
[34] Wang H, Wu Z, Rahnamayan S, Sun H, Liu Y, Pan J-S. Multi-strategy ensemble artificial bee colony algorithm. Inf Sci 2014;279:587–603.
[35] Karaboga D, Gorkemli B. A quick artificial bee colony (QABC) algorithm and its performance on optimization problems. Appl Soft Comput 2014;23:227–38.
[36] Kiran MS, Findik O. A directed artificial bee colony algorithm. Appl Soft Comput 2015;26:454–62.
[37] Hadidi A, Azad SK, Azad SK. Structural optimization using artificial bee colony algorithm. 2nd International conference on engineering optimization, September 6-9. 2010.
[38] Sonmez M. Artificial bee colony algorithm for optimization of truss structures. Appl Soft Comput 2011;11:2406–18.
[39] Sonmez M. Discrete optimum design of truss structures using artificial bee colony algorithm. Struct Multidiscipl Optim 2011;43:85–97.
[40] Degertekin SO. Optimum design of geometrically non-linear steel frames using artificial bee colony algorithm. Steel Compos Struct 2011;12:505–22.
[41] Garg H. Solving structural engineering design optimization problems using artificial bee colony algorithm. J Ind Manag Optim 2014;10(3):777–94.
[42] Aydogdu I, Akin A, Saka MP. Design optimization of real world steel space frames using artificial bee colony algorithm with levy flight distribution. Adv Eng Software 2016;92:1–14.
[43] Erol O, Eksin I. A new optimization method: big bang-big crunch. Adv Eng Software 2006;37:106–11.
[44] Camp CV. Design of space trusses using big bang-big crunch optimization. J Struct Eng ASCE 2007;133:999–1008.
[45] Kaveh A, Talatahari S. Size optimization of space trusses using big bang-big crunch algorithm. Comput Struct 2009;87:1129–40.
[46] Kaveh A, Talatahari S. Optimal design of Schwedler and ribbed domes via hybrid big bang-big crunch algorithm. J Constr Steel Res 2010;66(3):412–9.
[47] Kaveh A, Abbasgholiha H. Optimum design of steel sway frames using big bang–big crunch algorithm. Asian J Civil Eng 2011;12:293–317.
[48] Kazemzadeh Azad S, Hasancebi O, Erol OK. Evaluating efficiency of big-bang big-crunch algorithm in benchmark engineering optimization problems. Int J Optim Civil Eng 2011;1:495–505.
[49] Hasancebi O, Azad SK. An exponential big bang-big crunch algorithm for discrete design optimization of steel frames. Comput Struct 2011;110-111:167–79.
[50] Prayago D, Cheng M-Y, Wu Y-W, Herdany AA, Prayogo H. Differential big bang-big crunch algorithm for construction engineering design optimization. Autom Constr 2018;85:290–304.
[51] Namin AA. Structural evaluation of tied-arch and truss bridges subjected to wind and traffic loading MSc Thesis Gazimagusa, Northern Cyprus: Eastern Mediterranean University; 2012
[52] Demsar J. Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 2006;7:1–30.
[53] Garcia S, Fernandez A, Luengo J, Herrera F. A study of statistical technique and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Comput 2009;13:959–77.
[54] Derrac J, Garcia S, Molina D, Herrera F. A practical tutorial on the use of non-parametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 2011;1:3–18.