Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 2458-9403
Vol. 7 Issue 6, June - 2020

An Adaptive Discrete Cuckoo Search Algorithm to Solve Structural Optimization Problems

Ali Ahmid, Thien-My Dao, and Van Ngan Le
Correspondence Address
Mechanical Engineering Department, École de Technologie Supérieure ÉTS
1100 rue Notre-Dame Ouest
Montréal (Qc) H3C 1K3
Quebec - Canada.
Emails: ali-elmbrok-salem.ahmid.1@ens.etsmtl.ca, thien-my.dao@etsmtl.ca, & vanngan.le@etsmtl.ca

***Abstract*— The Cuckoo Search optimization algorithm (CS) continues to grab the attention of the scientific community due to its simplicity and robustness. CS applied successfully to solve a wide range of hard optimization problems, and it exhibited outstanding performance. The current study presents an Adapted variant of Discrete CS Algorithm (ADCSA) that uses the rank-value approach to turn real values of random Levy walks (steps/jumps) into the equivalent discrete values. Besides, the proposed ADCSA intensification effort was enhanced by adding four different local search movements of permutation, swap, insertion and bit flip. The solution accuracy of ADCSA was validated across a benchmarking case study of a composite laminated plate. Moreover, a further structural optimization problem of customized I-beam gantry crane was solved using ADCSA. Eventually, the results of both case studies reveal that the proposed ADCSA has a considerable performance in solving discrete structural optimization problems.**

***Keywords*— Cuckoo Search; Discrete optimization; Composite laminate; Gantry crane; critical buckling load**

I. INTRODUCTION

Cuckoo Search (CS) algorithm is population-based meta-heuristic inspired by the aggressive reproduction strategy of some cuckoo bird species enhanced by Levy flights. It presented by Yang and Deb (2009) to solve a variety of continuous multimodal optimization problems. Since then, it attracted the attention due to the simplicity of implementation and the fast convergency rate and accuracy of the delivered solutions. Also, CS has a view number of parameters (almost one) to be tuned, compared to other meta-heuristics such as Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO).... etc. The size of CS applications is fascinatingly growing, where it could be observed through the number of solved optimization problems using CS in the last decade (Yang 2014),. Shehab et al. (2017) tracked the progress of published papers that uses CS in the literature. Based on different publisher's metrices, for the CS published articles between 2009 and 2016, he summarized that there are three classes of research interest. The dominant class went to the application, and it represented 67% of publications, whereas CS hybridization and CS modifications had respectively 15 % and 18% of the research interest. The applications of CS involve several main optimization problems such as Travelling Salesman Problem (TSP)(Jati and Manurung 2012, Yang and Deb 2013, Ouaarab, Ahiod et al. 2014, Zhou, Ouyang et al. 2014), and binary optimization problems, for instance, Knapsack optimization problem (Layeb 2011, Gherboudj, Layeb et al. 2012, Xin, Zhang et al. 2019), Computer vision and image detection (Agrawal, Panda et al. 2013, Loubna, Mohamed et al. 2017), Energy sector (Piechocki, Ambroziak et al. 2014, de Moura Meneses, da Silva et al. 2020), supply chain (Li, Dey et al. 2018, Li, Liu et al. 2020) and structure optimization problem (Gandomi, Yang et al. 2013, Kaveh and Bakhshpoori 2013). However, the size of CS applications is likely to escalate in the prospective researches where the fast-growing research areas of Artificial intelligence (AI) and data mining are seeking more robust optimization algorithms to build faster response models (Cobos, Muñoz-Collazos et al. 2014).

Some proposed CS modifications were presented to improve the basic CS through imposing some enhancements of step size, such as using generative scaling factor instead of using constant value (Loubna, Mohamed et al. 2017). Other proposed modifications, in the literature, were mainly focused on adjusting CS to solve discrete optimization problems where the design space is limited to certain values or options (Xin, Zhang et al. 2019, Shehab 2020). Some discrete CS variants were explicitly developed to solve particular problems, Yang and Deb (2013) used CS to solve TSP or Xin, Zhang et al. (2019) who developed a discrete binary CS to address allocation of cognitive radio network spectrum optimization problem. The further general approach of handling discreetness constraint was worked out through rounding the generated continuous values, by Levy flights, into the

nearest integer, and it seems to be work for specific structural optimization problems (Kaveh and Bakhshpoori 2013). Loubna, Mohamed et al. (2017) proposed a fascinating approach where he used a rank-value approach to handle a sizeable discrete domain of image detection problem, and the results were impressive, and it deserves attention. Even though these modifications went so far to benefit from CS special features, but it still hard to say that there is one common variant of CS that could solve different discrete nature problems.

The current work presents an Adapted version of Discrete Cuckoo Search Algorithm (ADCSA) that uses a modified rank-value approach to interpret Levy flights random steps into equivalent discrete steps. In addition, the intensification capability of ADCSA is enhanced through introducing four different local search movements of permutation, swap, insertion and bit flip. The performance of ADCSA was firstly investigated through a well-known benchmark problem of a composite laminated plate. The obtained results of ADCSA were compared across the previously published results for other meta-heuristics. Moreover, ADCSA results also compared with other two different discrete CS variants, which implemented based on the rounding and original rank-value approaches (Kaveh and Bakhshpoori 2013, Loubna, Mohamed et al. 2017). The performance of the proposed ADCSA was remarkably superior to other metaheuristics, and the obtained results demonstrated promising performance of ADCSA in solving discrete structural optimization problems.

Consequently, ADCSA applied to solve the problem of optimization of customized I-beam gantry crane, which started to grab more attention in recent years (PAVLOVIĆ, SAVKOVIĆ et al. , Ahmid, Le et al. 2017). The different dimensions of the I- beam section need to be taken from a discrete range of steel plates, whereas the span length is fixed. The objective of the optimization is minimizing the cross-section area to reduce the crane weight where it is subjected to different strength constraints. The benefits of using customized I-beam cranes, rather than using standard I and H beams, were explained too.

Finally, the rest of this paper is arranged to explain the original CS in the second section, whereas the proposed ADCSA conceptual implementation is presented in the third section. The fourth section devoted to the validation case study while the customized crane case study demonstrated in the fifth section, and the summary of the current work outcomes and findings, with possible prospective research studies, were stated in the conclusion section.

II. CUCKOO SEARCH VIA LEVY FLIGHTS (CS)

The original Cuckoo Search is a population-based metaheuristic inspired by the reproduction strategy of Cuckoo Search bird. The bird starts searching for the surrounding to find a host nest of other birds. In each candidate nest, Cuckoo bird lay just one egg, and it flies to find another one to lay the next egg. This strategy has precisely coincided with the wisdom says, “Don’t put all eggs in one basket”, which in this occasion, means that the chance of Cuckoo eggs to survive is becoming better. The host bird could discover some of the eggs, and they may discard or abounded (Yang 2014). Yang and Deb (2009) introduced the CS algorithm to simulate this natural phenomenon where the total number of candidate nests represents the population size (n), and each nest is a possible solution (Si). A fraction (Pa) of the whole population with worse fitness is going to be discarded, and this mimic the discovery of the eggs by the hosting birds. Next, new randomly generated solutions are going to substitute the discarded solutions. The top-ranked nests will remain within the next generations. The CS searching of the design space goes via a random walk that taken out from Levy probability distribution. The original CS pseudo-code is listed in Algorithm 1.

Levy flight is the strengthening component of CS where it offers the random walk, though steps/jumps length is selected from Levy probability distribution. The jumps (long steps) in the design space are possible because of the heavily tailed nature of Levy probability distribution (Yang 2014).

Algorithm 1: Cuckoo Search Algorithm (CS)

Initialization:

Initial and evaluate a random population of n host nests (xi, fi).

While $(\boldsymbol{convergence\ not\ met})$ ***Do***:

- Generate a new Cuckoo (population) randomly by Levy flights (Eq.1).
- Evaluate the new Cuckoo fitness (f_i).
- Choose a nest among n (say, j) randomly.
  $If\ fi > fj$
- Replace the j with the new solution.
  end
- Rank the solutions and find the current best.
- Discard Pa fraction of worst solutions.
- Substitute the discarded solutions by new ones generated by Levy flights.

End

In general, the random walk depends on the previous location, x_i^t , and the length of the step/jump, which is the second term of equation (1).

$$x_i^{t+1} = x_i^t + \alpha \otimes Lèvy(\lambda, s) \quad (1)$$

where α is the step size scale factor and $\alpha > 0$. For most optimization problems, the unity step scale factor could work well (Yang and Deb 2009). The term $Levy(\lambda)$ represents Levy probability distribution, λ is Levy exponent, and s is the step size.

$$Lèvy(\lambda, step) \sim u = t^{-\lambda}, \quad (1 < \lambda \leq 3) \quad (2)$$

The random direction of the step/jump and step size that follows Levy probability are two essential elements to generate random numbers via levy flights. The direction of the step could be randomly drawn

from the normal distribution, $N(\mu, \sigma^2)$, whereas the step length needs to be determined through the Magenta algorithm. According, the step size (s) can be determined through the following formula:

$$s = \frac{u}{|v|^{1/\lambda}} \quad (3)$$

where

$$u \sim N(0, \sigma^2),\ v \sim N(0,1). \quad (4)$$

u, v are Gaussian normal distributions. The definition of u means that the random samples are drawn from a normal distribution that has 0 mean and variance of σ^2. The variance value could be obtained from:

$$\sigma^2 = \left[\frac{\Gamma(1+\lambda)}{\lambda\Gamma((1+\lambda)/2)} \cdot \frac{\sin(\pi\lambda/2)}{2^{(\lambda-1)/2}}\right]^{1/\lambda} \quad (5)$$

where $\Gamma(n)$ is nothing more than factorial of n or $n!$

III. Adapted Discrete Cuckoo Search Algorithm (ADCSA)

CS Originally presented as population-based metaheuristic to solve unconstrained continuous optimization problems (Yang and Deb 2009). However, several discrete variants were introduced to solve a particular discrete optimization problem (Ouaarab, Ahiod et al. 2014). Others were more general, such as using rounding of the Levy flight step into the nearest integer (Kaveh and Bakhshpoori 2013). (Loubna, Mohamed et al. 2017) presented a universal approach that could generate steps with integer values to obey Levy flights random walk. A similar approach has used here, and it is explained in subsection 3.2.

The proposed ADCSA bears three main modifications to the original CS. First is using Latin HyperCube (LHC) sampling method to generate the initial population; the second is presenting discrete Levy flights representation and finally improve the neighbourhood search of the best solution through four different permutation movements. The proposed ADCSA pseudo-code is listed in Algorithm 2.

Algorithm 2: Adapted Discrete Cuckoo Search Algorithm (ADCSA)

Initialization:
- Initial and evaluate a random population of n host nests (xi, fi) using Latin Hypercube (LHC) random generator.

While $(convergence\ not\ met)$ ***Do***:
Generate a new Cuckoo (population) randomly by Levy flights (Eq.8).
Evaluate the new Cuckoo fitness (f_i).
Choose a nest among n (say, j) randomly.
If $fi > fj$
- Replace the j with the new solution.

end
Rank the solutions and find the current best.
Discard Pa fraction of worst solutions.
Substitute the discarded solutions by new ones generated by Levy flights.
If $(maximum\ successful\ runs\ number\ exceeded)$
Do permutation, swap, insertion and bit flip for the current best solution.
end
Update the best solution.
End

A. Initial Population

Li, Liu et al. (2020) investigated the impact of initialization methods on the meta-heuristics searching performance. They examined eight different ways of random initialization sampling for five meta-heuristics. Their work results revealed that CS is sensitive to the initialization method, where it performed differently in 73.68 % of the tested functions based on the used method of initialization. Moreover, they suggested that the hybridization of different sampling methods could boost the algorithm performance of searching the design space. However, in the current study, three different sampling methods, Discrete Uniform Distribution (DUD), Latin HyperCube (LHC) and hybrid DUD-LHC, were examined. The numerical experiment results exhibited a slight improvement in the overall performance of ADCSA when LHC was used compared to the other two methods; See Figure (5).

B. Discrete Levy Flights Representation

The proposed approach by Loubna, Mohamed et al. (2017) defines the design space domain by rank and value. The "rank" refers to the location of the variable within the design domain vector, D, while "value" represents the corresponding assigned variable value/option, d_i. So, the new Cuckoo,$x_{(i+1)}$ is generated based on the current solution element rank, integer number, and an integer step size that obeys Levy flights, see Eq.1. Next, the new cuckoo with rank form is transformed into the equivalent values form. Therefore, $X_i^t = [x_1, x_2, \ldots . x_n]$ represents the design variables vector, and N is the problem size. Whereas, $D = [d_1\ d_2\ \ldots . d_M]$ indicates the discrete domain of the optimization problem. So,

$$Rank\ (D) = \{1,2, \ldots . , M\},$$

$$Value\big(Rank(D)\big) = \{\ d_1, d_2, \ldots , d_M\}$$

An improved version of the value-rank approach was implemented here. The improvements went to the update strategy of the size scale factor α, and to the step size determination. Selectin the step size factor α could influence the performance of the algorithm significantly, and it linked to the problem nature (Yang and Deb 2009). Using a constant value for α, e.g. 0.01 or 1, might work, but it doesn't consider any problem characteristics such as the solution fitness/quality. So, the proposed scale factor here is examining the quality of the fitness of the individual solution, f_i, to the fitness of the best solution, f_{best}.

$$\alpha = \frac{f_i}{f_{best}} \quad (6)$$

Consequently, the new solution, x_{i+1}, will be updated according to:

$$x_{i+1} = x_i + \frac{f_i}{f_{best}}.step \otimes (x_{best} - x_i) \quad (8)$$

Now, the updated step size, second term in Eq.8, produces real values, while the current solution x_i has an integer representation (rank) of discrete values and thus the new solution, x_{i+1} is going to have real values which we couldn't use it directly as ranking values. Therefore, a transformation function was used to turn the real values of the step size into their equivalent integer values. The sigmoid function, Eq. 9, is widely used in solving classification problems by machine learning (ML) algorithms (Shalev-Shwartz and Ben-David 2014); also, it used in the binary variant of CS to solve the knapsack problem, (Ouaarab, Ahiod et al. 2014).

$$Sigmoid(z) = \frac{1}{1 + e^{-z}} \quad (9)$$

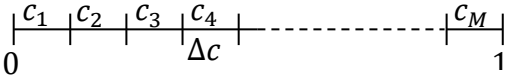
Sigmoid function, Figure(1), gives a selection probability between 0 and 1 for any input values, which is the step size in our case.

$$p_i(\alpha.s.\Delta x x_{bset}) = \frac{1}{1 + e^{-\alpha.s.\Delta x x_{bset}}} \quad (10)$$

The next step is dividing the interval [0,1] to the desired number of classes (ranks), C, and this gives each class (rank) a range of selection, $\triangle c$.

$$C = |D| \implies C = M$$

Then, the obtained selection probability,p_i, is compared across all ranges to determine the class to which this p_i belongs.



$$\Delta c = \frac{1}{M}$$

$$(k-1).\Delta c \le c_i < k.\Delta c\,, \qquad (i, k = 1{:}M)$$

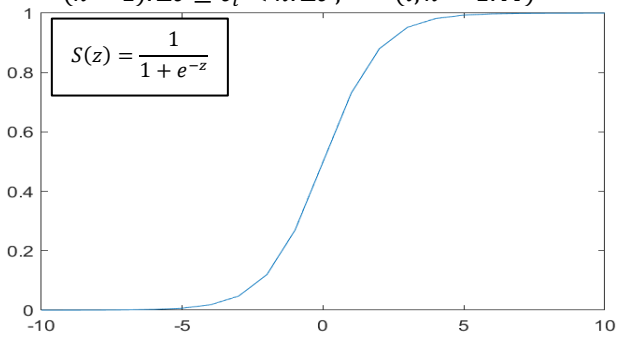


Figure 1: Sigmoid transformation function.

Despite the successful transformation of the original step size into its equivalent integer size, the new solution x_{i+1} might exceed the domain bounds when equation (8) applied. However, we experience a similar situation every day when we use the clock arithmetic to keep the time; e.g. the clock now is (8:00 am), and we want to know where will the hour hand be in 5 hours? $(8\ am + 5 \equiv 13)$ Obviously, it exceeds the clock bounds, which is 12, but intuitively we say it is 1:00 pm. The mathematical interpretation of this, that the remainder of dividing 13 by 12, is one, and this is the typical definition of modulo function that we are going to use to reflect a meaningful value for out-off bounds ranks. Therefore,

$$rank(x_{i+1}) = mod\big((x_i + s), |D|\big) + 1 \quad (11)$$

where s represents the integer value of the levy flights steps/jumps, and 1 is the minimum rank value if $mod\big((x_i + s), |D|\big) = 0$. Eventually, the ranked new solution, $rank(x_{i+1})$, reversed to its assigned rank values or $value(rank(x_{i+1}))$ form that we could use to determine the objective optimization function directly.

C. *Neighbourhood Search*

Yang (2013) expect that CS intensification could be improved by using local search Levy flights or hybridization CS with other local optimization algorithms (e.g. Tabu Search). The primary purpose of enhancing the intensification feature of any meta-heuristic is to ensure that the obtained solution is not a local optimum, and there was no possible global optimal solution left behind the current best solution. However, the proposed ADCSA turns the optimization problem into a pure permutation problem, as a result of using the rank-value approach. The ranked solution has an integer representation that we could permute to produce a new ranked solution. Based on this, four different permutation operators employed in ADCSA to improve the search of the current best solution neighborhood.

1) Random permutation

Random permutation operator is selecting randomly two elements of the solution vector and reverses the order of the other elements in between. Let's that we have a six dimensions solution vector as follow:

$$x = [1\ 2\ 3\ 4\ 5\ 6]$$

So, a possible permutation is:

Before permutation:

$$x = [1\ \mathbf{2}\ 3\ 4\ \mathbf{5}\ 6]$$

After permutation:

$$x = [1\ \mathbf{5}\ 4\ 3\ \mathbf{2}\ 6]$$

2) Swap(mutation)

Swap operator also knows as mutation, selects randomly two elements and switch over their positions in the solution vector.

Before swap:

$$x = [1\ 2\ \mathbf{3}\ 4\ 5\ \mathbf{6}]$$

After swap:

$$x = [1\ 2\ \mathbf{6}\ 4\ 5\ \mathbf{3}]$$

3) Insertion

The insertion operator is randomly selecting an element of the solution vector and insert it randomly between the other two elements.

before insertion:

$$x = [1\ 2\ 3\ 4\ \mathbf{5}\ 6]$$

after insertion:

$$x = [1\ \mathbf{5}\ 2\ 3\ 4\ \ 6]$$

4) Bit flip

Bit flip operator selects a random element (bit) of the solution vector and changes its rank order randomly.

before bit flip:

$$x = [1\ 2\ 3\ 4\ 5\ 6]$$

after a bit flip:

$$x = [1\ 2\ \mathbf{2}\ 4\ 5\ 6]$$

Random permutation and swap operators were used in solving a well-known benchmark problem of structural engineering of composite laminated design optimization by Genetic Algorithm (GA), (Le Riche and Haftka 1993). Whereas, insertion and bit flip operators were used efficiently, as a part of an improved Tabu search algorithm in searching of the neighbourhood of large design space optimization problems (He, de Weerdt et al. 2019). Lastly, these operators were integrated into ADCSA structure in the way that they will not be activated until a certain number of successful runs is reached.

D. Convergence Criteria

ADCSA designed to stop after a certain number of successful runs without any solution improvement is reached. Otherwise, it continues searching for the optimal solution until the predefined maximum number of iterations is exceeded.

IV. Numerical Experiments

In order to examine the performance of the proposed ADCSA in solving discrete structural optimization problems, two different case studies were selected from the literature. The first case study is a benchmark problem used here as a numerical experiment to validate the solution accuracy of ADCSA and to compare its performance with other meta-heuristics in the literature. Thus, a benchmark problem of a composite laminated plate subjected to bi-directional compression loading was used in this experiment (Le Riche and Haftka 1993). To ensure the robustness of ADCSA, another discrete structure case study of customized I-beam gantry crane subjected to yield criteria (Ahmid, Le et al. 2017).

A. Validation numerical experiment

The benchmark optimization problem of a composite laminated plate subjected to compression loading is extensively used in the literature to investigate the performance of a new or modified meta-heuristics (Aymerich and Serra 2008, Koide, França et al. 2013).In a more recent study, the same benchmark problem used to compare the performance of five different meta-heuristics (Ahmid, Thien-My et al. 2019). The original optimization problem introduced by (Le Riche and Haftka 1993)for a laminated rectangular plate simply supported. The in-plane compression loading conditions were applied in the direction of both axes x,y, see Figure (2) . The optimization objective is maximizing the critical buckling loading capacity of the plate subjected to design and manufacturing constraints. Moreover, the number of plies, N_p, and thickness of each ply, t_p, are imposed while the fiber orientation of each plies group, θ_p, needs to be chosen from a discrete domain of available orientations, $D = [0°; \pm 45°; 90°]$. The design variable vector, X, is formed by the number of plies groups that meet the symmetry and balanced constraints. The symmetrical laminate means that both sides about the mid-plane have the same number of the plies and this reduces the number of plies, to be optimized, into the half of the total number of plies, Np=2 .Balanced laminate is symmetrical one where each group of two plies, with same fiber orientation, on one side has a similar group on the other side, and this downsize the number of the optimized plies to another half. Thus, the final number of design variables (optimized plies) will be equal to $N_p = 4$ .The formula of buckling load factor λ_b , which developed according to Classical Lamination Theory (CLT), has been used implicitly to determine the objective function of critical buckling load, λ_{cb} , (Riche,et al. 1993), Therefore,

$$\lambda_b(\mathrm{p},\mathrm{q}) = \pi^2 \frac{[D_{11}(p/a)^4 + 2(D_{12}+2D_{66})(p/a)^2 + D_{22}(q/b)^4]}{(p/a)^2 N_x + (q/b)^2 N_y} \quad (12)$$

where D_{ij} is the bending stiffness, N_x, N_y are in-plane compression loads in x, y. The variables p , q denote the buckling modes in both x, y directions. The critical buckling load factor λ_{cb} , is defined as the minimum obtained value of $\lambda_b(\mathrm{p},\mathrm{q})$, (Rao 2009).

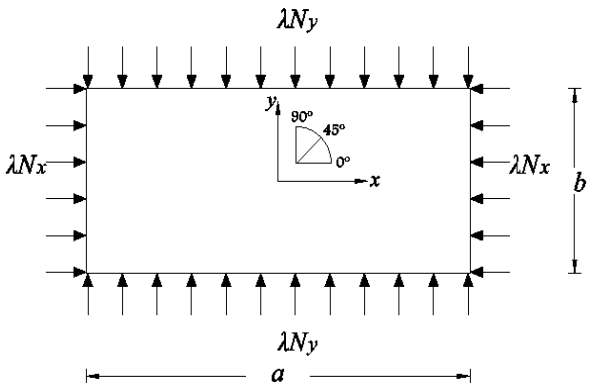


Figure 2: Simply supported plate subjected to biaxial loading (Ahmid, Thien-My et al. 2019).

The orthotropic material properties, dimensions, and loading conditions of the composite laminated

plate used in this experiment are listed in Tables 1 and 2.

1) Experiment Setting

The proposed ADCSA code written in Matlab 2019b programming language. In addition, the other two discrete variants of presented by Kaveh and Bakhshpoori (2013) and Loubna, Mohamed et al. (2017) were implemented and programmed using

```
==========================================================================
User: Ali Ahmid ……..15-Feb-2020 00:24:25
==========================================================================
Machine Information:
CPU Processor: Intel(R) Core (TM) i7-4790 CPU @ 3.60GHz
CPU clock speed: 3601 MHz
CPU Cache size (L2): 1024 KB
Number of physical CPU cores: 4
Installed physical memory (RAM): 16 GB
operating System Type: Windows
Operating System Version: Microsoft Windows 7 Enterprise
==========================================================================
```

Figure 3: The specifications of PC-machine used in the current comparison study.

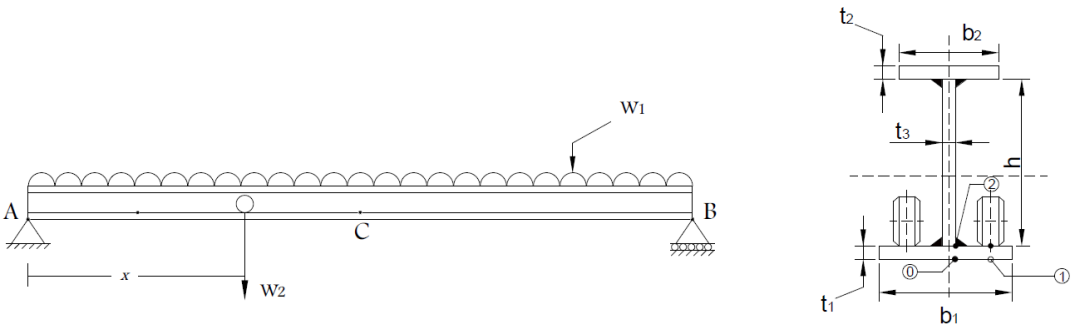


Figure 4: The crane beam dimensions and loading conditions, (Ahmid, Le et al. 2017)

Matlab 2019b. All implemented variants tested on the same PC-machine with the listed specifications in Figure (3). The experiment initialized with the same number of nests, $n_{Nest} = 100$, and same discovery rate, $P_a = 0.25$. For each variant, the experiment has been repeated 200 times to overcome the stochastic behaviour of meta-heuristics as recommended in the original reference, (Le Riche and Haftka 1993).

Furthermore, an assessment of three different random initialization methods of Discrete Uniform Distribution (DUD), Latin hypercube (LHC) and hybrid DUD-LHC were conducted. The results demonstrated slightly better performance of ADCSA when LHC used to generate the initial population, see Figure . Thus, it used to generate the initial population of ADCSA in the executed validation experiments.

2) ADCSA Performance assessment criteria

In the literature, there were different measures used to assess meta-heuristics performance. Elapsed time is not only the measure used to evaluate the computational solution cost where the success rate (or reliability) and the average number of runs required to find the optimal solution (price) were commonly used too. Furthermore, normalizing the solution price, price/ reliability, could reveal valuable information about the solution cost (Le Riche and Haftka 1993, Ahmid, Thien-My et al. 2019). Lastly, where such an optimization problem has multi-optimal solutions, the term of practical optima is used. It is devoted to considering the near-optimal solutions of 0.1% error to the best-known optimal solution (Aymerich and Serra 2008).

B. Customized I-beam gantry crane problem

The customized I-beam gantry crane design is another structural design problem that started to attract the attention, (PAVLOVIĆ, SAVKOVIĆ et al. , Ahmid, Le et al. 2017, Alhorani 2020). The original problem statement says that for a welded I-beam profile, gantry crane built by welding three different steel plates that have the same length, but they were diverse in their thickness and width. The live loading condition was applied to the crane, see Figure. The nomenclature of different crane dimensions and loads are given as: b_1, t_1 and $b_2\ t_2$ are lower and upper flanges widths and thicknesses respectively, while h, t_3 are the width and thickness and width of the web. W_1represents the crane weight and W_2 is the live load. Lastly, L is the crane span, and x is the distance of W_2 measured from the crane left end. The optimization objective is reducing the crane weight by minimizing the crane cross-section area, which is defined by:

$$A_{cs} = b_1.t_1 + b_2.t_2 + h.t_3 \quad (13)$$

The crane design is subjected to (14) bending and buckling criteria, which result

in a set of design constraints. Hence,

$$g_1 = \sigma_{comb\_max} - \sigma_{Tallowed} \leq 0$$

$$g_2 = 1.9 - f_{Buckling} \leq 0 \quad (15)$$

$$g_3 = h/t_3 - 260 \leq 0 \quad (16)$$

$$g_4 = b_2/2t_2 - 260/\sqrt{\sigma_y} \leq 0 \quad (17)$$

$$g_5 = \delta_v - L/600 \leq 0 \quad (18)$$

$$g_6 = (\Delta\sigma)_{comb_{max}} - \Delta\sigma_{allowed} \leq 0 \quad (19)$$

These constraints were imposed by using the exterior penalty function that transforms the objective function, A_{cs}, into:

$$F(X, r_g) = A_{cs}(X) + r_g \left[\sum_{j=1}^{m} (max\{0, g_j(X)\})^2\right] \quad (20)$$

where $X = [b_1\ t_1\ b_2\ t_2\ h\ t_3]$ and r_g is penalty multiplier for inequality constraints g_i.

Eventually, the material used for the crane is 350W structure steel with yielding strength S_y =350 MPa, density ρ=7850 kg/m^3, Young's modulus E=200 GPa, shear modulus G=77 GPa and Poisson's ratio v=0.3. The dimensions intervals and loads of the crane optimized here are:

$b_1 \epsilon\ [150:10:490]$, (mm)
$t_1\ \epsilon\ [6:2:74]$, (mm)
$b_2\ \epsilon\ [150:10:490]$, (mm)
$t_2\ \epsilon\ [6:2:74]$, (mm)
$h\ \epsilon\ [600:20:1280]$, (mm)
$t_3\ \epsilon\ [2:36]$, (mm)
$L\ = 8$, (m)
$W_2 = 10{,}20{,}40$, (ton)

V. Results and Disscusions

The obtained results of the validation experiment and the case study of customized I-beam gantry crane are illustrated and discussed in the following subsections.

A. *Validation of experiment results*

The results of the three initialization methods, they mentioned in section 3.1, were statistically compared and depicted in standard division graph in Figure (5).

Moreover, the two variants of discrete CS by Kaveh and Bakhshpoori (2013) and Loubna, Mohamed et al. (2017) where implemented, as described, and they were given two abbreviations, RDCS and ADCS, respectively. Consequently, they were applied for the same experiment with the same number of experiments. The number of 30 runs without improving was used as convergency criteria to break the variant searching loop. The proposed ADCSA was also examined with the same experiment setting, and the obtained results of all discrete CS were illustrated in Figures 5-8. Finally, the summary of the comparison

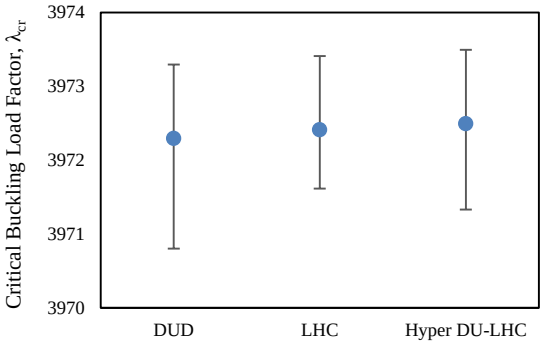


Figure 5 Standard deviation plot for different initialization methods

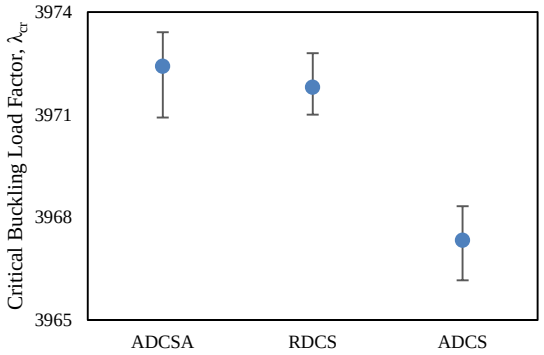


Figure 6 Standard deviation plot for different initialization methods

of different published results and the proposed algorithms were listed in Table 3. The results reveal that the proposed ADCSA outperforms the other presented DCS algorithms and other meta-heuristics in literature as solving algorithm for the composite laminated plate. ADCSA exhibited a fast convergence rate where it needs around 12 iterations to find the optimal solution. Moreover, ADCSA delivers an accurate solution with So, the accuracy of the proposed ADCSA is examined, and it has shown significant performance in solving the NP-optimization problem of structural engineering.98% reliability (successful rate) at 41 iterations solution cost.

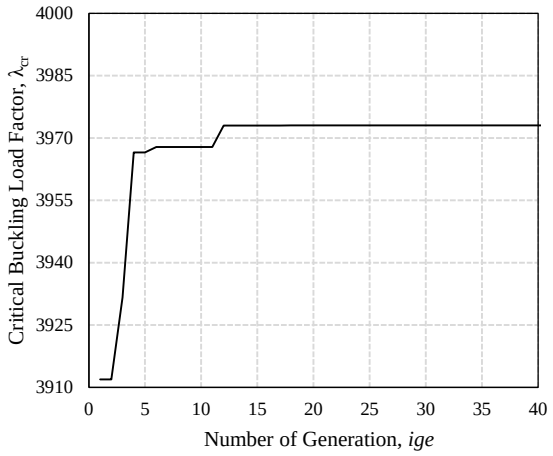


Figure 8. ADCSA meta-heuristic Convergence in the first successful run.

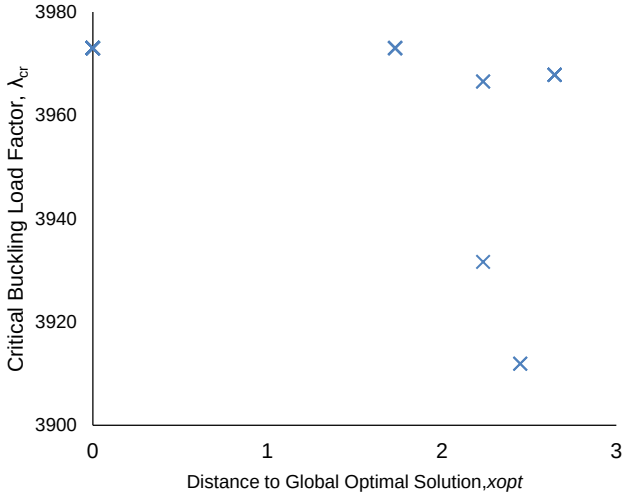


Figure 9 Distance to global optimal for ADCSA meta-heuristic first successful run

The next section is devoted to demonstrating the results of applying ADCSA to other structural engineering optimization problems.

A. Customized I-beam Gantry crane Results

The obtained results of discrete crane design optimization using ADCSA were compared to previously published one of the continuous optimization approaches. Furthermore, both results compared to an equivalent standard I-beam profile for the same optimization strength constraints (CISA). The comparison study produced nine different examples of 8 m crane, where the three different types of crane beams were subjected to three live loads of 10,20 and 40 tons. The results of this comparison listed in Figures 10-17 and the abbreviations CC, DC and ES

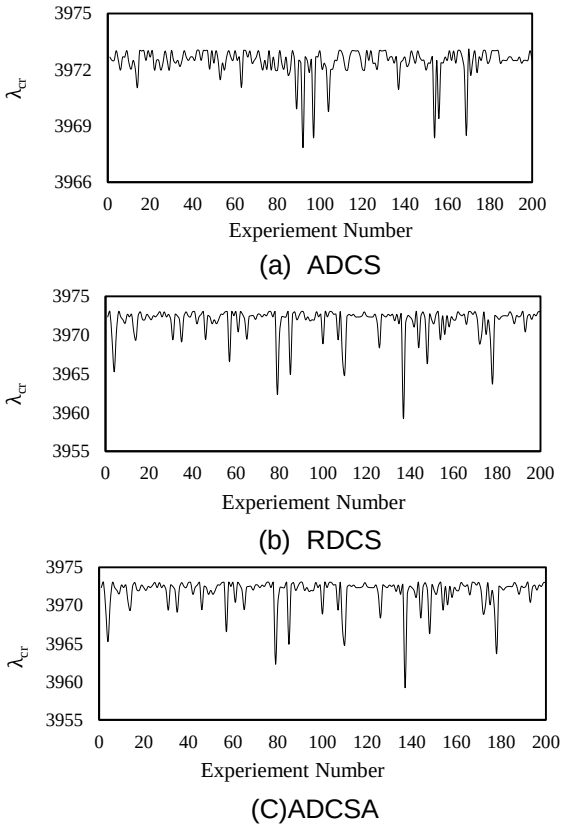


Figure 7 The number of experiments vs. critical buckling load for ADCSA, RDCS, ADCS.

are mentioning the three different types of I-beams crane: Continues optimized Custom (CC), Discrete optimized Custom (DC) and Equivalent Standard I-beam (ES) respectively.The results obtained here reveal that the crane cross-section profile of the discrete optimization approach followed the same configurations pattern achieved using a continues optimization approach. It always shows narrow and thick lower flange, wider and thinner upper flange and tall and very thin web, see Figure 17. Furthermore, both approaches of optimal custom crane did not violate any imposed constraints for the three live loads. At the same time, the equivalent standard I-beam failed to remain within the strength limits of tension, and fatigue stresses constraints for the 40-ton case, see Figure 10-11. The lateral buckling of 10-ton live load almost reached the limit for the three I-beam types, see Figure 12. The local buckling of the top flange became critical for CC I-beam, while it was never critical for DC or ES I-beam types. On the other hand, the web slenderness was not critical for any CC I-beam loading cases while it reaches the limits in the first loading case (10-ton) for the other two types, see Figure 13. Finally, the results reveal that the discrete optimization approach could reduce the weight from 61 - 69 % of the equivalent standard I-beam crane structure.

Table 1: Comparison of different performance measures for ADCSA and other meta-heuristics

Meta-heuristic	Price	Reliability, %	Normalized Price	Elapsed Time, sec

ADCSA	41	98%	42	6
ADCS (Luboan2017)	71	56.5%	126	18
RDCS (Kaveh2013)	243	93.5%	259	118
GA (LeRiche1993)	371	98.9%	375	NA
GA (Ahmid 2019)	252	88%	286	16
ACO (Ahmid2019)	88	76.5%	115	4

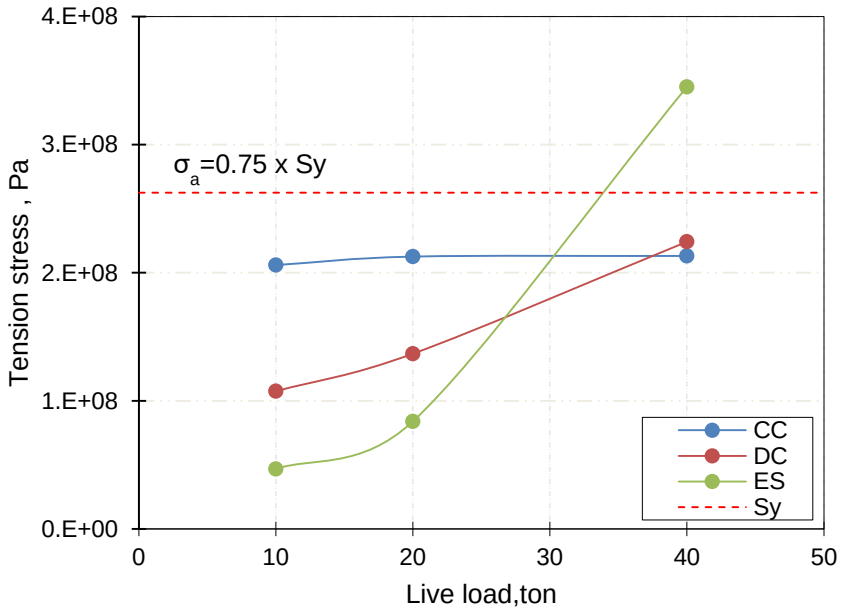


Figure 10: The tension stresses vs. live loads for different types of I-beam crane

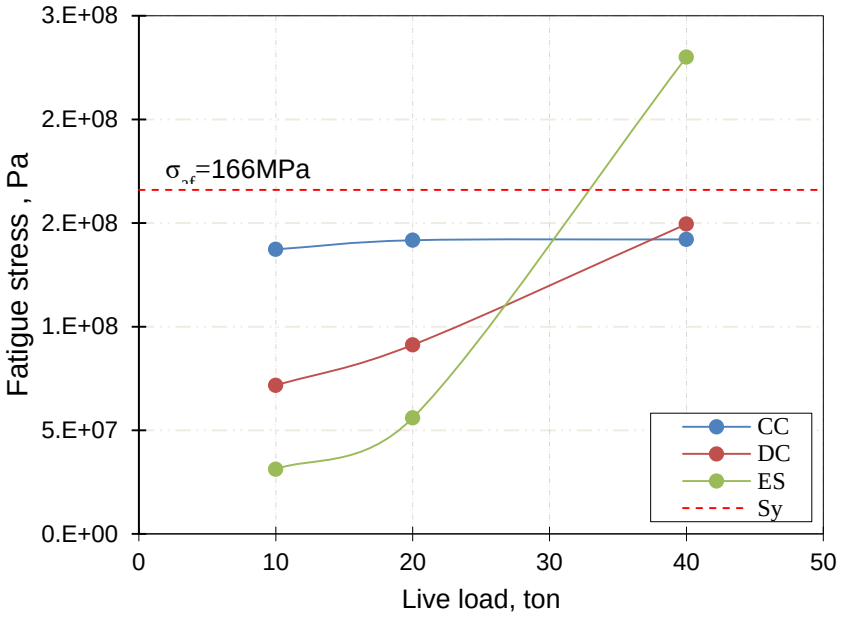


Figure 11: The fatigue stresses vs. live loads for different types of I-beam crane

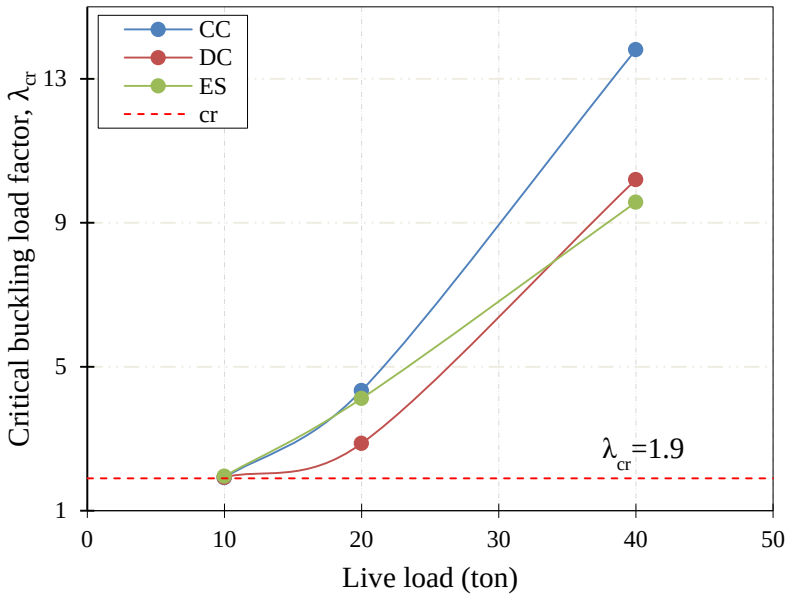


Figure 12: The critical buckling load factor vs. live loads for different types of I-beam crane

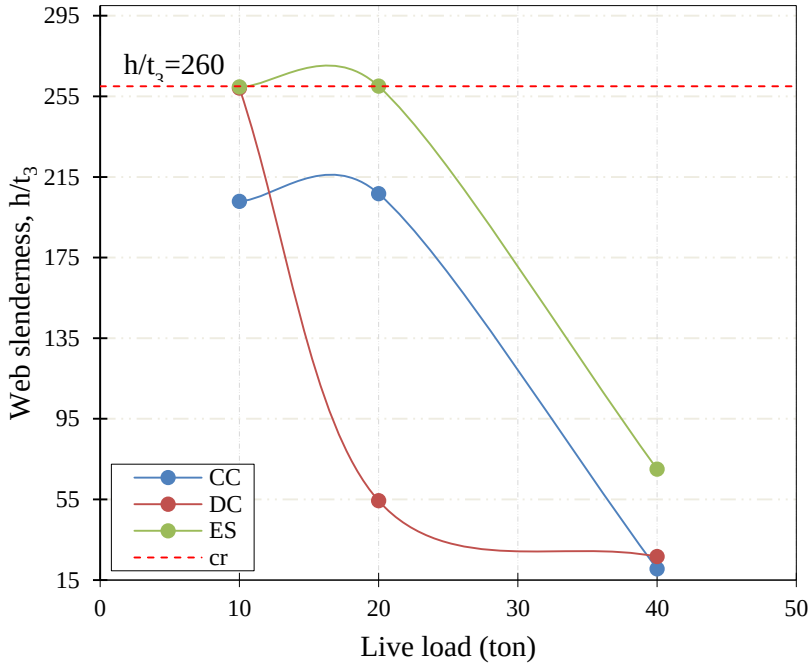


Figure 13: The web slenderness vs. live loads for different types of I-beam crane

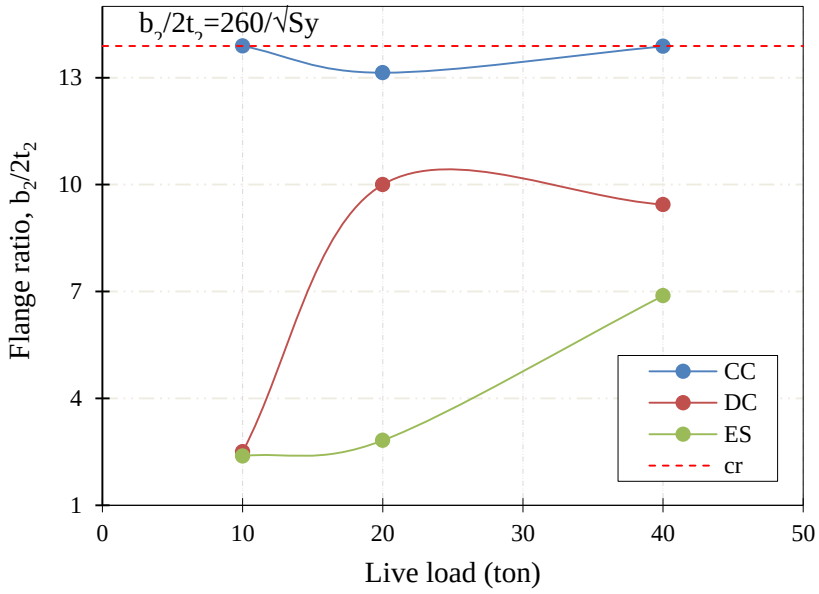


Figure 10: The flange ratio vs. live loads for different types of I-beam crane.

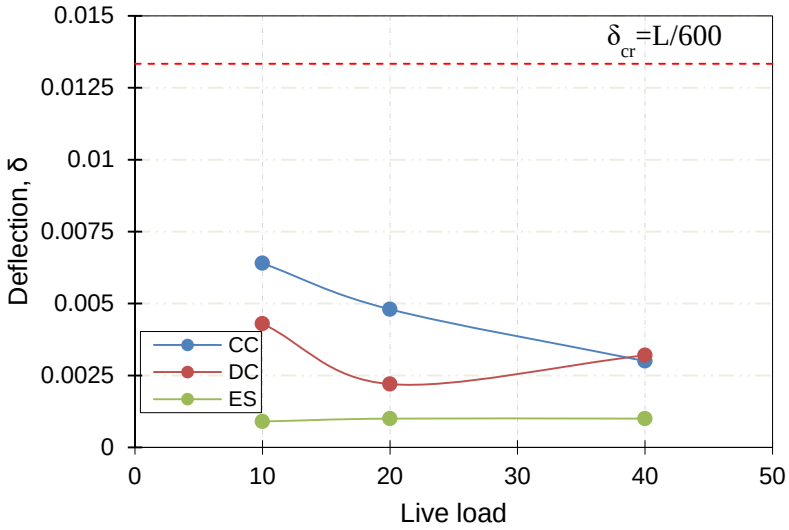


Figure 15: The deflection vs. live loads for different types of I-beam crane

Table 4: Different optimal solution configuration of customized I-beam gantry crane

Design Approach		t_1 (mm)	b_1 (mm)	t_2 (mm)	b_2 (mm)	t_3 (mm)	h (mm)	Area (m^2)
CC	**10 tons**	27.82	150.01	6.99	194.2	3	608.64	.00736
	20 tons	37.88	150.16	8.38	220.18	3.19	826.46	.0102
	40 tons	52.62	150.04	9.08	252.14	4.38	1137.03	.0152
ES	**10 tons**	56.9	270.3	56.9	270.3	31.5	826.46	.0513
	20 tons	54.1	305.2	54.1	305.2	30	796.8	.0572
	40 tons	40	550	40	550	16	1120	.0621
DC	**10 tons**	54	150	34	170	3	620	.01574
	20 tons	46	170	10	200	14	760	.0177
	40 tons	46	190	16	290	10	1080	0.02418

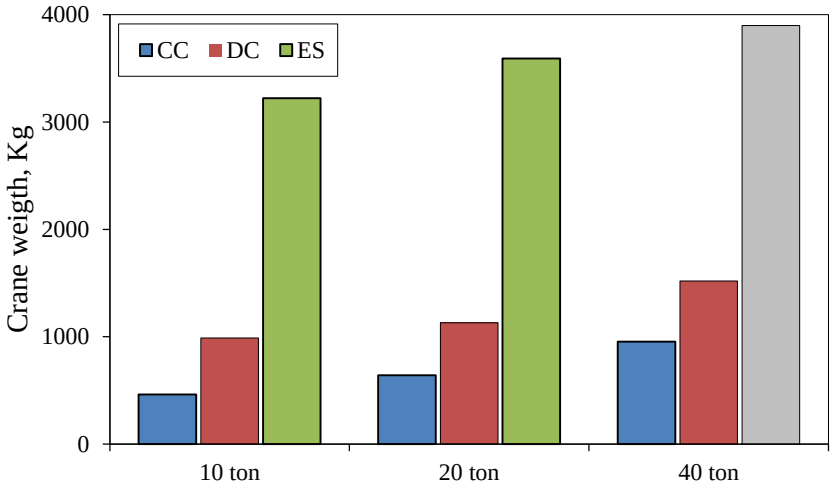


Figure 16: The crane weight vs. live loads for different types of I-beam crane

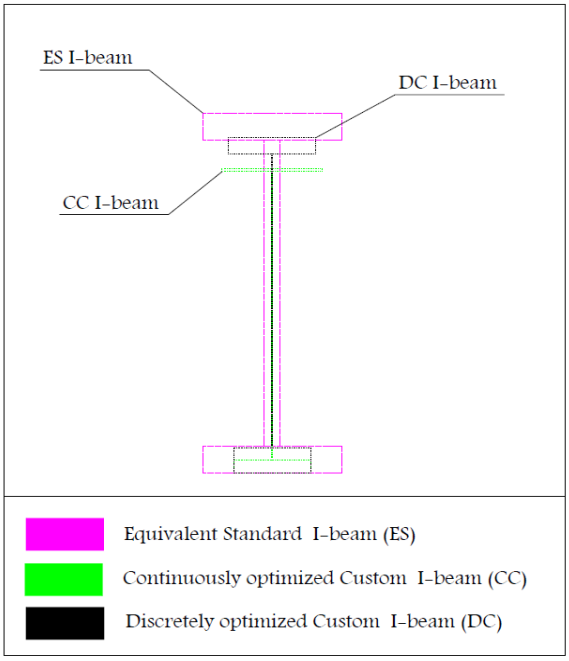


Figure 11: The three different types of I-beam crane for the case of 8 m x10 ton.

VI. CONCLUSION

A new variant of the Adapted Discrete Cuckoo Search Algorithm (ADCSA) presented and examined for two different case studies of maximizing the critical buckling load of composite laminated plate and a customized I-beam gantry crane design optimization. The validation results demonstrated a high accuracy of the ADCSA solution at a reasonable cost. Furthermore, the initialization methods experiment conducted here illustrated a slight effect of the initial population generation on the performance ADCSA. The use of the LHC sampling approach improved the reliability slightly compared to ADCSA initialized using DUD or Hybrid DUD-LHC. The results obtained for the customized I-beam gantry crane shown that the crane cross-section profile of the discrete optimization approach followed the same configurations pattern obtained using a continues optimization approach. It always shows narrow and thick lower flange, wider and thinner upper flange and tall and very thin web. Additionally, the saving in the cross-section area is noticeable compared to the equivalent standard I-beams. Eventually, the proposed ADCSA has been applied for two different structural optimization problems so far and examining it for other engineering problems could be prospective work. Furthermore, investigating the different initialization methods on the proposed algorithm to find better performance deserves a try.

REFERENCES

Agrawal, S., et al. (2013). "Tsallis entropy based optimal multilevel thresholding using cuckoo search algorithm." Swarm and Evolutionary Computation **11**: 16-30.

Ahmid, A., et al. (2017). "An optimization procedure for overhead gantry crane exposed to buckling and yield criteria." International Journal of Technology and Engineering **8**(2): 11.

Ahmid, A., et al. (2019). "Comparison Study of Discrete Optimization Problem Using Meta-Heuristic Approaches: A Case Study." International Journal of Industrial Engineering **1**(2): 97-109.

Alhorani, R. A. (2020). "Mathematical models for the optimal design of I-and H-shaped crane bridge girders." Asian Journal of Civil Engineering **21**(4): 707-722.

Aymerich, F. and M. Serra (2008). "Optimization of laminate stacking sequence for maximum buckling load using the ant colony optimization (ACO) metaheuristic." Composites Part A: Applied Science and Manufacturing **39**(2): 262-272.

Cobos, C., et al. (2014). "Clustering of web search results based on the cuckoo search algorithm and Balanced Bayesian Information Criterion." Information Sciences **281**: 248-264.

de Moura Meneses, A. A., et al. (2020). "Application of Cuckoo Search algorithm to Loading Pattern Optimization problems." Annals of Nuclear Energy **139**: 107214.

Gandomi, A. H., et al. (2013). "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems." Engineering with computers **29**(1): 17-35.

Gherboudj, A., et al. (2012). "Solving 0-1 knapsack problems by a discrete binary version of cuckoo search algorithm." International Journal of Bio-Inspired Computation **4**(4): 229-236.

He, L., et al. (2019). "Time/sequence-dependent scheduling: the design and evaluation of a general purpose tabu-based adaptive large neighbourhood search algorithm." Journal of Intelligent Manufacturing: 1-28.

Jati, G. K. and H. M. Manurung (2012). Discrete cuckoo search for traveling salesman problem. 2012 7th International Conference on Computing and Convergence Technology (ICCCT), IEEE.

Kaveh, A. and T. Bakhshpoori (2013). "Optimum design of steel frames using Cuckoo Search algorithm with Lévy flights." The Structural Design of Tall and Special Buildings **22**(13): 1023-1036.

Koide, R. M., et al. (2013). "An ant colony algorithm applied to lay-up optimization of laminated composite plates." Latin American Journal of Solids and Structures **10**(3): 491-504.

Layeb, A. (2011). "A novel quantum inspired cuckoo search for knapsack problems." International Journal of Bio-Inspired Computation **3**(5): 297-305.

Le Riche, R. and R. T. Haftka (1993). "Optimization of laminate stacking sequence for buckling load maximization by genetic algorithm." AIAA journal **31**(5): 951-956.

Li, Q., et al. (2020). "Influence of initialization on the performance of metaheuristic optimizers." Applied Soft Computing: 106193.

Li, Z., et al. (2018). "Discrete cuckoo search algorithms for two-sided robotic assembly line balancing problem." Neural Computing and Applications **30**(9): 2685-2696.

Loubna, B., et al. (2017). "A Novel adaptive Discrete Cuckoo Search Algorithm for parameter optimization in computer vision." Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial **20**(60): 51-71.

Ouaarab, A., et al. (2014). "Discrete cuckoo search algorithm for the travelling salesman problem." Neural Computing and Applications **24**(7-8): 1659-1669.

PAVLOVIĆ, G., et al. "Analysis and Optimization Design of Welded I-girder of the Single-beam Bridge Crane."

Piechocki, J., et al. (2014). "Use of Modified Cuckoo Search algorithm in the design process of integrated power systems for modern and energy self-sufficient farms." Applied Energy **114**: 901-908.

Rao, S. S. (2009). Engineering optimization: theory and practice, John Wiley & Sons.

Shalev-Shwartz, S. and S. Ben-David (2014). Understanding machine learning: From theory to algorithms, Cambridge university press.

Shehab, M. (2020). Introduction of Diffusion MRI and Cuckoo Search Algorithm. Artificial Intelligence in Diffusion MRI, Springer**:** 1-12.

Shehab, M., et al. (2017). "A survey on applications and variants of the cuckoo search algorithm." Applied Soft Computing **61**: 1041-1059.

Xin, Z., et al. (2019). Spectrum Allocation of Cognitive Radio Network Based on Improved Cuckoo Search Algorithm. Proceedings of the 2nd International Conference on Computer Science and Software Engineering.

Yang, X.-S. (2013). Cuckoo search and firefly algorithm: Theory and applications, Springer.

Yang, X.-S. (2014). Nature-inspired optimization algorithms, Elsevier.

Yang, X.-S. and S. Deb (2009). Cuckoo search via Lévy flights. 2009 World congress on nature & biologically inspired computing (NaBIC), IEEE.

Yang, X.-S. and S. Deb (2013). "Multiobjective cuckoo search for design optimization." Computers & Operations Research **40**(6): 1616-1624.

Zhou, Y., et al. (2014). "A discrete cuckoo search algorithm for travelling salesman problem." International Journal of Collaborative Intelligence **1**(1): 68-84.