Exact and approximate methods for solving the edge-strength problem

Eduardo Rodriguez-Tello[1][0000−0002−0333−0633], Eric Monfroy[2][0000−0001−7970−1368], and Claudia Vasconcellos-Gaete[2][0000−0001−9487−0573]

[1] Cinvestav Unidad Tamaulipas.
Km. 5.5 Carretera Victoria-Soto La Marina, 87130 Victoria Tamps., Mexico
`ertello@cinvestav.mx`

[2] LERIA, Université d'Angers
2 Bd de Lavoisier 49000 Angers, France
`{eric.monfroy, claudia.vasconcellos}@univ-angers.fr`

Abstract. The *Edge-Strength* (*ES*) problem is a graph labeling problem where the goal is to assign integer labels to the edges of a finite undirected graph in such a way that the maximum sum of labels between any two adjacent edges, known as *edge-strength*, is minimized. This work introduces the first methods to solve the *ES* problem exactly and approximately, including two constraint satisfaction problem (CSP) models and a simulated annealing (SAES) metaheuristic. The first CSP model is based on constrained optimization using the AllDifferent global constraint, while the second employs extensional constraints. Computational experiments on 40 standard topology graph instances demonstrate the effectiveness and robustness of these approaches. The CSP models provide exact solutions for smaller instances, while the SAES algorithm efficiently approximates solutions for larger and complex graphs. These contributions advance the state-of-the-art in solving the *ES* problem and pave the way for further research.

Keywords: Edge-strength problem · Exact solution methods · Constraint programming · Simulated Annealing · Metaheuristics

1 Introduction

There has been a growing interest in studying graph labeling problems (GLP) in recent years. Theoretically, these problems are essential since they generally belong to the $\mathcal{NP}$-hard class. They are also relevant in the industry because they can be used as abstract models that allow engineers to solve diverse practical application problems.

The *Edge-Strength* (*ES*) is a GLP, which was first stated in [6]. It can be formally stated as follows. Let $G(V, E)$ be a finite undirected graph of order $n = |V|$ and size $m = |E|$. Given an injection $\varphi : E \to \{1, 2, \ldots, m\}$, which represents a labeling of the edges of the graph, the *edge-strength* (cost) of G for φ is defined as:

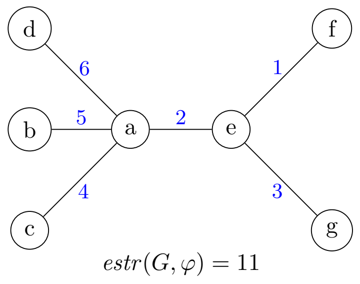
$$estr(G, \varphi) = \max\big\{\varphi(e_1) + \varphi(e_2) \,:\, e_1, e_2 \text{ are adjacent edges of } G\big\}\,, \qquad (1)$$

where $\varphi(e_i)$ denotes the label associated to the edge $e_i \in E$. Thus, the *ES* problem consist in finding a labeling φ^*, such that $estr(G, \varphi^*)$ is minimized, i.e.,

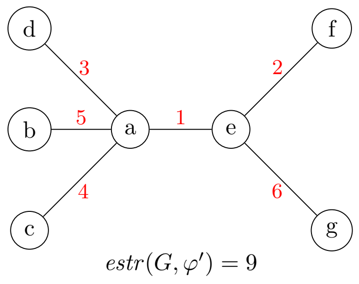
$$\varphi^* = \arg\min_{\varphi \in \Phi}\big\{estr(G, \varphi)\big\}\,,$$

where Φ is the set of all the possible labelings. The labeling φ^* satisfying this condition is known as optimum.

For example, consider the graph $G(V, E)$ of order $n = 7$ and size $m = 6$ depicted in Figure 1(a) with the labeling φ given by the numbers shown over each edge. The distance between each pair of adjacent edges is calculated using the expression $\varphi(e_1) + \varphi(e_2)$ and presented in column 2 of Table 1. For this particular labeling φ, the edge-strength (cost) of G is $estr(G, \varphi) = 11$. For the labeling φ' of G, presented in Figure 1(b), the edge-strength is $estr(G, \varphi') = 9$ and represents the optimal solution for this particular graph (see column 3 of Table 1).



(a) Labeling φ for the graph G.



(b) Labeling φ' for the graph G.

Fig. 1. Example of an *Edge-Strength* (*ES*) problem instance.

Given its recent introduction, the *ES* problem has received less attention than other well-known graph labeling problems. Up to now, most of the research on this problem has been concentrated on the theoretical study of its properties to find exact solutions for certain specific families of graphs: paths, cycles, stars, complete graphs, bipartite complete graphs [6, 7]. A list of formulas for computing the optimal edge-strength for those specific graph families is presented in Table 2. In the case of general graphs, a lower bound can be computed using the following formula [5]:

$$estr(G) \geq m + 2(\delta(G) - 1)\,,$$

Table 1. Computing the edge-strength for two different graph labelings shown in Figure 1.

Adjacent edges	$\varphi(e_1) + \varphi(e_2)$	$\varphi'(e_1) + \varphi'(e_2)$
ab,ac	$5 + 4 = 9$	$4 + 5 = 9$
ab,ad	$5 + 6 = 11$	$4 + 3 = 7$
ab,ae	$5 + 2 = 7$	$4 + 1 = 5$
ac,ad	$4 + 6 = 10$	$5 + 3 = 8$
ac,ae	$4 + 2 = 6$	$5 + 1 = 6$
ad,ae	$6 + 2 = 8$	$3 + 1 = 4$
ae,ef	$2 + 1 = 3$	$1 + 2 = 3$
ae,eg	$2 + 3 = 5$	$1 + 6 = 7$
ef,eg	$1 + 3 = 4$	$2 + 6 = 8$

where m is the size of G and $\delta(G)$ its minimum degree. Despite this fundamental advancement, many different graph topologies exist for which optimal solutions remain unknown.

Table 2. Formulas reported in the literature for computing the optimal edge-strength for the following graph families: Path (P_n), Cycle (C_n), Star (S_n), Double star (S_{n_1,n_2}) and Complete bipartite (K_{n_1,n_2}).

Graph type	$\|V\|$	$estr(G)$	Ref.
$P_n \quad (n \geq 3)$	n	n	[6]
$C_n \quad (n \geq 3)$	n	$n + 2$	[6]
$S_n \quad (n \geq 2)$	n	$2n - 3$	[6]
S_{n_1,n_2}	$n_1 + n_2$	$2(n_1 + n_2) - 5$	[6]
$K_{n_1,n_2} \quad (n_1 > n_2 \geq 2)$	$n_1 + n_2$	$2n_1n_2 - 2n_2 + 1$	[7]

As far as we know, no exact or approximate solution algorithms have been reported in the literature for tackling the edge-strength problem. This article addresses this gap by introducing two new constraint satisfaction problem (CSP) models designed to find exact solutions for small instances of the problem. Moreover, we propose an approximation approach based on the simulated annealing algorithm for dealing with medium-scale benchmark instances. These contributions provide a comprehensive framework for tackling the edge-strength problem across different scales, advancing the state-of-the-art and opening new avenues for further research.

The remainder of this work is organized as follows. Section 2 introduces two CSP models – one using arithmetic constraints and the other using extensional constraints – designed for the exact resolution of the *ES* problem. Section 3 outlines the key implementation details of the proposed simulated annealing algorithm, called SAes. In Section 4, we describe the experimental setup used to assess the practical performance of both the exact and heuristic methods. This is followed by an analysis of the computational results, demonstrating the efficiency and robustness of our approaches across the selected benchmark problems. Finally, Section 5 summarizes the key contributions of this work and discusses potential directions for future research.

2 Modeling the problem as a COP or CSP

2.1 A COP - arithmetic model

We propose a first model based on constrained optimization (COP) that relies on the efficiency of the global constraint *AllDifferent* to treat inequalities [1, 14].

In the *ES* problem, we have m variables $l_{e_1}, \ldots, l_{e_m}$; each l_{e_i} has a domain $[1..m]$, and represents the label of edge e_i. Like in any other labeling problems, we must first state that each label is unique (2). Then, the COP model proposed minimizes the maximum sum of two labels of adjacent edges (3), where $\mathcal{L}^*(G)$ represents the edge-strength of G.

$$\mathit{AllDifferent}(\{l_{e_1}, \ldots, l_{e_m}\}) \ , \tag{2}$$

$$\mathcal{L}^*(G) = \mathit{minimize}\left(\max\{l_{(u,v)} + l_{(v,w)} \mid \forall((u,v),(v,w)) \in E(G) \times E(G)\}\right) \ . \tag{3}$$

However, our preliminary experiments showed that this COP model could be inefficient. Indeed, all the constraints (except the *AllDifferent* constraint) are in the objective function, and thus, constraint solvers cannot prune the search space properly. We thus propose a CSP model based on extensional constraints, i.e., table constraints [11].

2.2 A CSP model with extensional constraints

The main idea of this CSP model is to convert the problem into a constraint satisfaction problem, looking for a labeling $\mathcal{L}(G) \leq k$ for a given positive integer k. Thus, given k, we can deduce which pairs of labels are permitted for adjacent edges. Let's call $\mathcal{T}(m,k)$ this set of pairs of permitted labels:

$$\mathcal{T}(m,k) = \{(l,l') \in [1..m] \times [1..m] \mid l + l' \leq k\}$$

We need the same variables as in the COP model. Thus, we consider m finite domain variables $l_{e_1}, \ldots, l_{e_m}$ with domains $[1..m]$, representing the labels of the edges of G. We still need to state that each label is unique by using constraint (2).

Then, we have to enforce that the sum of labels of each pair of adjacent edges is smaller than or equal to k:

$$\forall((u,v),(v,w)) \in E(G) \times E(G),\ (l_{(u,v)}, l_{(v,w)}) \in \mathcal{T}(m,k) \tag{4}$$

The CSP model is thus given by:

$$\mathcal{M}_{CSP,m,k} = Constraint\ (2)\ \wedge\ Constraint\ (4)$$

Note that the model $\mathcal{M}_{CSP,m,k}$ is parameterized by the number of edges m of the graph and the integer k.

By instantiating $\mathcal{M}_{CSP,m,k}$ for a graph G with $|E(G)| = m$, we obtain a finite domain CSP instance $\mathcal{I}_{CSP,G,m,k}$. Solving $\mathcal{I}_{CSP,G,m,k}$ with a standard finite domain solver, we are thus able to decide whether G has an edge-strength less than or equal to a given integer k.

2.3 Minimizing k with the CSP model

A standard incremental optimization can be used: starting with k equal to the lower bound of the instance, we try to find a labeling with an edge-strength value smaller than k. If satisfiable, we have the minimum edge-strength; otherwise, we increment k and continue the same iterative process. However, this optimization is not efficient at all.

We thus propose an optimization based on a property of the *ES* problem.

Property 1. If there is a labeling with edge-strength value k, there is also a labeling with edge-strength value $k+1$. If no labeling results in a cost $k+1$, there is also no solution for value k.

Proof (Sketch of the proof).

$\Longrightarrow$ consider two adjacent edges (u,v) and (v,w) such that $l_{(u,v)} + l_{(v,w)} = \max\{l_{(u,v)}+l_{(v,w)} \mid \forall((u,v),(v,w)) \in E(G)\times E(G)\} = k$. Consider $l_{(u,v)} < l_{(v,w)}$ and thus, $l_{(u,v)} < m$. Then, we swap the label $l_{(u,v)}$ with the label $l_{(u,v)}+1$ which was on edge (x,y). Thus, the sum of each pair of adjacent edges with x or y in common is decreased by one and is thus strictly less than k. The sum of each pair of adjacent edges with u or v in common is increased by 1, and a fortiori, $l_{(u,v)} + l_{(v,w)} = k+1$. We thus obtain a labeling of cost $k+1$.

$\Longleftarrow$ we have shown that "labeling of cost k" $\rightarrow$ "labeling of cost $k+1$", which is the same as "no labeling of cost k" $\vee$ "labeling of cost $k+1$", which is also the same as "no labeling of cost $k+1$" $\rightarrow$ "no labeling of cost k".

Using the previous property, we propose a dichotomic algorithm (see Algorithm 1) for optimization of k between the standard lower and upper bounds, using the model $\mathcal{M}_{CSP,m,k}$:

3 Simulated annealing algorithm

Simulated Annealing (SA) is a versatile probabilistic optimization method proposed independently in [10, 16]. Since its introduction, it has demonstrated its effectiveness in approximating globally optimal solutions for numerous NP-hard problems [4, 8, 9]. The core principle of this optimization approach involves occasionally accepting a neighboring solution that worsens the current one, with

Algorithm 1: Optimize Function

```
   Data: $G, lb, ub$
1  $best\_k \leftarrow ub$;
2  while $lb < ub$ do
3      $k \leftarrow (ub + lb) \div 2$;
4      if solve($\mathcal{I}_{CSP,G,|E(G)|,k}$) is SAT then
5          $ub \leftarrow k$;
6          $best\_k \leftarrow ub$;
7      else
8          $lb \leftarrow k + 1$;
9  return $best\_k$
```

this acceptance governed by a carefully managed probability. As the algorithm progresses, the likelihood of accepting such non-improving moves gradually decreases [2].

We illustrate in Algorithm 2 the pseudo-code of our SA implementation for solving the *ES* problem. Next, we describe its main components.

Search space. Given a graph $G = (V, E)$ of size $|E| = m$, the search space Φ for the *ES* problem is composed of all possible labelings (solutions) of G, $\varphi : E \rightarrow \{1, 2, \ldots, m\}$. Therefore, $m!/2$ possible solutions exist for such a graph, given that every one of the $m!$ possible labelings can be inverted to obtain an equivalent *edge-strength* (cost).

Solutions representation. In our SAES algorithm a labeling (solution) φ is represented as a vector l of integers with length m, which is indexed by the edges and whose i-th value $l[i]$ denotes the label assigned to the edge e_i.

Evaluation function. The quality $estr(G, \varphi)$ of the labeling φ is evaluated by using the evaluation function (1). Every edge in the graph G must be analyzed to compute it. As a result, $\mathcal{O}(|E|)$ instructions must be executed by this *complete evaluation scheme*. Nevertheless, the proposed SAES algorithm employs an *incremental evaluation* of neighboring solutions. To this end, the edge-strength of each edge in the graph is stored using an appropriate data structure. Indeed, suppose that the labels of two different non-adjacent edges (e_u, e_v) are exchanged in a labeling φ to produce a neighboring solution φ'. We should only recompute the edge costs that change to obtain the new edge-strength of φ'. It takes only $\mathcal{O}(|\mathcal{A}(e_u)| + |\mathcal{A}(e_v)|)$ operations, where $|\mathcal{A}(e_i)|$ represents the number of adjacent edges to e_i. Therefore, our SAES algorithm can analyze thousands of neighboring

Algorithm 2: Simulated annealing algorithm (SA)

```
   Data: $\mathcal{N}, f, l, \alpha$
1  $t_i, t_f \leftarrow$ ComputeTemperatures();
2  $\varphi \leftarrow$ GenerateInitialSolution();
3  $\varphi^* \leftarrow \varphi$;
4  $t_k \leftarrow t_i$;
5  $k \leftarrow 0$;
6  while $\neg$StopCriterion() do
7      Select randomly $\varphi' \in \mathcal{N}(\varphi)$;
8      $\Delta f \leftarrow f(\varphi') - f(\varphi)$;
9      Generate a random $u \in [0, 1]$;
10     if $(\Delta f <= 0)$ or $(e^{-\Delta f / t_k} > u)$ then
11         $\varphi \leftarrow \varphi'$;
12         if $f(\varphi') < f(\varphi^*)$ then
13             $\varphi^* \leftarrow \varphi'$;
14     if TemperatureLength() then
15         $t_k \leftarrow \alpha t_{k-1}$;
16     $k \leftarrow k + 1$;
17 return $\varphi^*$
```

solutions, employing only a small fraction of the time required by the complete evaluation scheme.

Initial solution. The initial solution is the starting labeling used for the algorithm to begin the search for better configurations in the search space Φ. In our implementation, the starting solution is randomly generated.

Initial and final temperatures. Our algorithm's initial and final temperatures are computed using the following formulas proposed in [3].

$$t_i = \Delta f_{min} + \frac{\Delta f_{max} - \Delta f_{min}}{10} \ , \tag{5}$$

$$t_f = \Delta f_{min} \ , \tag{6}$$

where Δf_{min} and Δf_{max} are the minimum and maximum cost difference between consecutive visited labelings during a random walk, of length $\frac{m(m-1)}{4}$, through the search space.

Neighborhood function. The main objective of the neighborhood function in a local search algorithm is to identify the set of potential solutions that can be reached from the current solution [13]. In our SAES algorithm, we implemented the neighborhood function $\mathcal{N}_1(\varphi)$ that is formally defined as follows:

$$\mathcal{N}(\varphi) = \{\varphi' = swap(\varphi, e_u, e_v) \ : \ e_u, e_v \in E \text{ and } e_v \notin \mathcal{A}(e_u)\} \, , \tag{7}$$

where $swap(\varphi, e_u, e_v)$ is a function allowing the exchange of the labels of a pair of non-adjacent edges e_u and e_v to produce a new labeling φ'. The operation that replaces the incumbent solution with a new neighboring labeling is called a move.

Acceptance criterion. Our SAes algorithm employs the Metropolis condition [12]. It systematically accepts moves to improving or equal quality neighboring labelings and could execute worsening moves with a probability that depends on the increase of the evaluation function value and the temperature t_k (see line 10 in Algorithm 2).

Temperature length. The maximum number of neighboring solutions visited at each temperature value ($maxConfigurations$) depends directly on the number of edges (m) of the graph because more moves are required for denser graphs: $maxConfigurations = \beta \times m$.

Cooling schedule. A geometry cooling schedule is used in our simulated annealing implementation: $T_k = \alpha \times T_{k-1}$ [10].

Termination Condition. The SAes algorithm stops when the current temperature reaches the computed final temperature T_f value.

4 Computational experiments

In this section, computational experiments for assessing the practical usefulness of the COP/CSP models and the SAes algorithm introduced above are presented. We employed a benchmark set composed of 40 instances. It includes graphs with standard topologies (path, cycle, star, double star, complete bipartite, cycle power, hypercube, complete, Möbius ladder, triangulated grid, r-level t-ary tree, wheel, Petersen, Cartesian products). These graphs' order ($|V| = n$) ranges from 5 to 20, while their size ($|E| = m$) spans from 4 to 32 edges.

These experiments were run on a computer equipped with an Intel® Xeon® E5-2630 v4 processor at 2.20 GHz, 64 GB of RAM, and a Linux operating system. The COP/CSP models were coded in Python using the PyCSP^3 v2.4 library [15]. The allotted execution time for completing the `optimization()` function over each benchmark instance was 120 hours, which can lead to several calls to the CSP solver (same instance, with different k values).

Our SA algorithm was implemented in C++ and compiled using $g{+}{+}$ v14.2 with the -$O3$ optimization flag. Given its stochastic nature, we conducted 10 independent sequential runs for each selected benchmark instance. The algorithm requires only two input parameters: β and α. The parameter β determines the maximum number of neighboring solutions evaluated at each temperature level ($maxConfigurations$), while α controls the temperature reduction after processing $maxConfigurations$ solutions. The values for α and β were chosen experimentally and fixed to 0.99 and 5, respectively.

The detailed results from our experiments can be found in Table 3. The first three columns in this table display the name of the graph, its order (n), and its size (m). Column 4 lists the optimal edge-strength cost (Opt) found by the proposed COP model. The lower (lb) and upper (ub) bounds established by the CSP model, as well as the best cost ($B^{\star}$) attained by it, are presented in columns 6 to 8. Columns 10 to 13 register the average cost (Avg), standard deviation (Std), average CPU time in seconds, and the best cost ($B^{\star}$) reached by our SAes algorithm in 10 executions with each instance. The computational time (t) in seconds expended by the COP/CSP models are also presented in columns 5 and 9.

We observe several key findings based on data provided in Table 3. To begin, we will delve into the analysis of the COP model, which uses basic arithmetic constraints (see Section 2.1). It successfully found the optimal solutions for simpler graph structures such as path, cycle, and star graphs with a small number of edges (m). However, as the complexity of the graph increased, the expended computational time rose significantly. For instance, the path graph with 19 edges took 78.5 seconds, and the cycle graph with 15 edges took 93.0 seconds, indicating a substantial computational burden. Despite these challenges, the COP model demonstrated its capability of reaching the optimal solutions when given sufficient time, making it suitable for small to medium-sized graphs. Nonetheless, it is important to note that it did not resolve 13 out of 40 instances within the maximum allotted 120 hours (rows marked with symbol "–"), highlighting its limitations when dealing with larger and more complex graphs.

Turning to the CSP model (see Sections 2.2 and 2.3), which employs extensional constraints and a dichotomic search algorithm, the results indicated a more efficient handling of larger graph instances than the COP model. The CSP model found the optimal or near-optimal solutions with relatively lower computational times for all the 40 tested instances. For example, it efficiently solved instances like the complete bipartite and double star graphs. However, for highly complex graphs like cycle power and hypercube, the CSP model's bounds (lb and ub) still left some gaps, indicating that while it is faster, it may not always reach the exact optimal solution quickly. The CSP model was, in fact, able to handle graphs with up to 32 edges in a much shorter time frame.

Analyzing the results of our Simulated Annealing algorithm implementation (SAes), it is evident that this heuristic approach is highly efficient for larger instances, providing near-optimal solutions within very short computational times. For instance, it consistently found optimal or close to optimal solutions for all graph types, including highly complex graphs such as the Möbius ladder and Cartesian products for which their solutions were still unknown. The average CPU time for SAeswas the lowest among the three tested solution approaches, demonstrating its effectiveness in quickly approximating solutions. The standard deviation values were also minimal, indicating the robustness and reliability of our metaheuristic in providing consistent results across multiple executions. Notably, even for graphs with up to 32 edges, SAesshowed remarkable performance.

5 Conclusions and future work

This work addressed the edge-strength (ES) problem in graphs, presenting the first exact and approximate methods to solve this problem. The key contribu-

Table 3. Experimental results reached with the introduced COP/CSP models and the SAES algorithm over 40 graphs with diverse standard topologies.

			COP		**CSP**				**SAES**			
Instance	*n*	*m*	*Opt*	*t*	*lb*	*ub*	$B^\star$	*t*	*Avg*	*Std*	$B^\star$	*t*
path10	10	9	10	0.8	10	10	10	0.9	10.0	0.0	10	0.1
path15	15	14	15	1.3	15	27	15	0.9	15.2	0.4	15	0.1
path20	20	19	20	78.5	20	20	20	2.6	20.5	0.5	20	0.1
cycle10	10	10	12	1.7	11	12	12	0.9	12.0	0.0	12	0.1
cycle15	15	15	17	93.0	16	29	17	1.1	17.2	0.4	17	0.1
cycle20	20	20	–	–	21	30	22	1.6	22.5	0.5	22	0.1
star5	5	4	7	2.2	5	7	7	0.9	7.0	0.0	7	0.1
star7	7	6	11	2.4	7	11	11	0.9	11.0	0.0	11	0.1
star10	10	9	17	5.5	10	17	17	0.9	17.0	0.0	17	0.1
star15	15	14	–	–	15	27	27	0.6	27.0	0.0	27	0.2
dStar6-3	9	8	13	1.6	9	15	13	0.9	13.0	0.0	13	0.1
dStar5-5	10	9	15	4.3	10	17	15	0.9	15.0	0.0	15	0.1
dStar6-6	12	11	19	87.2	12	21	19	0.9	19.0	0.0	19	0.1
dStar8-4	12	11	19	71.6	12	21	19	0.9	19.0	0.0	19	0.1
bipartite3x3	6	9	14	2.5	10	14	14	0.9	14.0	0.0	14	0.1
bipartite4x3	7	12	19	79.6	13	23	19	0.9	19.0	0.0	19	0.1
bipartite5x3	8	15	–	–	16	29	25	0.7	25.0	0.0	25	0.1
bipartite4x4	8	16	–	–	17	24	26	1.0	26.0	0.0	26	0.1
cyclePow15-2	10	30	–	–	31	59	47	1.2	47.0	0.0	47	0.2
hypercube3	8	12	–	–	13	23	18	1.5	18.0	0.0	18	0.1
k5	5	10	17	8.1	11	17	17	0.9	17.0	0.0	17	0.2
k6	6	15	–	–	16	29	26	0.6	26.0	0.0	26	0.2
mobLadder6	6	9	14	2.2	10	17	14	0.9	14.0	0.0	14	0.1
mobLadder8	8	12	18	53.0	13	23	18	0.9	18.0	0.0	18	0.1
mobLadder10	10	15	22	15139.2	16	29	22	0.9	22.0	0.0	22	0.1
mobLadder12	12	18	–	–	19	35	26	0.6	26.0	0.0	26	0.1
triangle6	6	9	14	3.1	10	17	14	0.9	14.0	0.0	14	0.1
triangle8	8	13	20	295.8	14	25	20	1.0	20.0	0.0	20	0.1
tree2x2	7	6	9	2.3	7	9	9	0.9	9.0	0.0	9	0.1
tree2x3	13	12	19	316.0	13	23	19	0.9	19.0	0.0	19	0.1
tree3x2	15	14	19	724.9	15	19	19	0.9	19.0	0.0	19	0.2
wheel5	5	8	13	2.6	9	13	13	0.9	13.0	0.0	13	0.1
wheel7	7	12	–	–	13	18	19	1.0	19.0	0.0	19	0.2
petersen	10	15	22	14425.1	16	29	22	1.0	22.0	0.0	22	0.1
c4xc4	16	32	–	–	33	63	50	2.3	50.0	0.0	50	0.4
p2xp3	6	7	–	–	8	11	10	0.6	10.0	0.0	10	0.1
p3xp3	9	12	17	18.5	13	17	17	0.9	17.0	0.0	17	0.1
p2xc3	6	9	–	–	10	14	14	0.6	14.0	0.0	14	0.1
p3xc3	9	15	–	–	16	23	23	0.7	23.0	0.0	23	0.1
st3xst3	9	12	17	15.3	13	23	17	1.0	17.0	0.0	17	0.1

tions are the development of two constraint satisfaction problem (CSP) models and a metaheuristic based on simulated annealing (SAes). The CSP models include an arithmetic constraints-based model and a model using extensional constraints, while the SAes metaheuristic is designed to handle medium-sized instances efficiently.

The experimental results demonstrate the effectiveness of these methods. The CSP models can find exact solutions for smaller instances. In contrast, the SAes algorithm performs well for larger and more complex graphs, providing near-optimal solutions in significantly less computational time. These findings indicate that our proposed approaches offer a comprehensive framework for tackling the *ES* problem across different scales.

For future research, several avenues can be explored to advance the study of the *ES* problem further:

- Developing other metaheuristic algorithms, such as genetic algorithms or particle swarm optimization, to compare their performance against our simulated annealing implementation.
- Investigating the fitness landscape of the es problem to understand its structure and optimization challenges, potentially leading to more efficient solution methods.
- Studying graph instances with more edges and other complex topologies to evaluate the proposed models' and algorithms' scalability and applicability in various industrial contexts.

Acknowledgments. The first author thankfully acknowledge a sabbatical leave granted by Cinvestav (01/09/2024 - 31/08/2025), as well as the courtesies and facilities of the LERIA, Université d'Angers France.

References

1. Beldiceanu, N., Carlsson, M., Rampon, J.X.: Global constraint catalog. Tech. Rep. Research Report SICS T2005-08, Swedish Institute of Computer Science, Kista. (May 2005), `https://hal.science/hal-00485396`
2. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys **35**(3), 268–308 (sep 2003). https://doi.org/10.1145/937503.937505
3. Connolly, D.T.: An improved annealing scheme for the QAP. European Journal of Operational Research **46**(1), 93–100 (1990). https://doi.org/10.1016/0377-2217(90)90301-Q
4. Franzin, A., Stützle, T.: Revisiting simulated annealing: A component-based analysis. Computers & Operations Research **104**, 191–206 (2019). https://doi.org/10.1016/j.cor.2018.12.015
5. Ichishima, R., Oshima, A., Takahashi, Y.: Bounds for the edge-strength of graphs. Memoirs of the Kokushikan University Information Science **41**, 9–15 (2020)
6. Ichishima, R., Oshima, A., Takahashi, Y.: The edge-strength of graphs. Discrete Mathematics Letters **3**, 44–49 (2020)
7. Ichishima, R., Oshima, A., Takahashi, Y.: Some new results on the edge-strength and strength of graphs. Discrete Mathematics Letters **12**, 22–25 (2023)

8. Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. Operations Research **37**(6), 865–892 (1989). https://doi.org/10.1287/opre.37.6.865
9. Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by simulated annealing: An experimental evaluation; part ii, graph coloring and number partitioning. Operations Research **39**(3), 378–406 (1991). https://doi.org/10.1287/opre.39.3.378
10. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
11. Lecoutre, C.: Optimization of simple tabular reduction for table constraints. In: Stuckey, P.J. (ed.) Principles and Practice of Constraint Programming, CP 2008. Lecture Notes in Computer Science, vol. 5202, pp. 128–143. Springer (2008)
12. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H.: Equation of state calculations by fast computing machines. Journal of Chemical Physics **21**(6), 1087–1092 (1953)
13. Talbi, E.: Metaheuristics: From design to implementation. John Wiley & Sons (2009)
14. van Hoeve, W.: The alldifferent constraint: A survey. In: 6th Annual Workshop of the ERCIM Working Group on Constraints. pp. 1–12 (2001). https://doi.org/10.48550/arXiv.cs/0105015
15. XCSP3 Team: PyCSP^3 v2.4 (2024), `https://www.pycsp.org/`
16. Černý, V.: A thermodynamic approach to the traveling salesman problem: An efficient simulation algorithm. Journal of Optimization Theory and Applications **45**(1), 41–51 (1985)