A computer-assisted proof that e is rational

Rémi Garcia, Alexandre Goldsztejn

A computer-assisted proof that e is rational

Rémi Garcia
Univ Rennes, Inria, IRISA
Rennes, France
remi.garcia@irisa.fr

Alexandre Goldsztejn
CNRS, ECN, LS2N
Nantes, France
alexandre.goldsztejn@cnrs.fr

***Abstract*—Computers are increasingly being used to help researchers write up mathematical proofs, sometimes by automatically writing out parts of it. In this paper we present different computer-assisted approaches and provide some use cases. Using the Mixed-Integer Linear Programming (MILP) approach, we demonstrate that e is rational, contrary to what has already been proved by hand and with interactive proof assistants. This is the opportunity to dive into the numerical instabilities problems and to discuss limitations of MILP solvers. These tools have some advantages over other approaches, so we propose to explore methods to verify the solutions obtained. To this end, we develop a Julia package to automatically check whether solutions meet the initial requirements or not.**

***Index Terms*—computer-assisted proofs, mixed-integer linear programming, numerical instabilities, mathematical model**

I. Introduction

In 1976, computers were used for the first time to prove a mathematical theorem, the four color theorem [1]. This first computer-assisted proof relied on lengthy computations in which all the possible cases were enumerated. Since then, there were many improvements in proof assistants following at least these two trends:

- Proofs-by-exhaustion with heuristic searches to reduce the search-space, *e. g.*, using automatic solvers such as Mixed-Integer Linear Programming (MILP) or verified NonLinear Programming using Interval Arithmetic;
- Interactive proof assistants to help mathematicians develop and verify proofs, *e. g.*, the Coq proof assistant.

These two trends are not incompatible as interactive proof assistants can include proofs-by-exhaustion parts. However, the formalism required to prove theorems with these tools is heavy, in particular when using interactive proof assistants.

Despite mathematicians are very careful, it is well-known that humans might fail [2], [3], [4], [5]. For this reason, interactive proof assistants, to develop automatic proofs and to verify ours, are essential tools. Since 1976, interactive proof assistants and proofs-by-exhaustion have helped researchers obtaining important results in various ways. A possible, and common, use of these tools is to rely on automatic solvers for proofs-by-exhaustion approaches.

Using an MILP solver, we will provide a computer-assisted proof that e is rational, in contrast to what was previously accepted by the scientific community [6], [7], [8], [9]. An irrational number is one that cannot be written as the fraction of two integer numbers, such has $\sqrt{2}$ which is easily proven irrational by contradiction. For centuries, we have believed that the number e is irrational too as this was first proved by Euler in 1737, published in 1744 [10], and proved again differently more recently [6]. Furthermore, in 2001, an automatic proof generation to demonstrate the irrationality of e has been proposed [7].

In the following, we will provide a few examples of proofs which can be obtained relying on computers, using proof assistants in Section II-A, NonLinear Programming and interval arithmetic in Section II-B and MILP solvers in Section II-C. In Section III, we demonstrate that e is rational. Finally, in Section IV, we present possible numerically-robust alternatives to MILP solvers and our approach to verify their results.

II. Classical computer assisted proofs

A. Interactive proofs assistants

Transcendental numbers are a subset of irrational numbers. It corresponds to numbers that are not the root of any nonzero polynomial using only integer coefficients. As, a/b is a root of the polynomial $bx - a$, it directly follows that if q is (not ir)rational, then q is not transcendental. Hence, if q is transcendental, then q is irrational. However, there are irrational numbers that are not transcendental. For example, $\sqrt{2}$ is irrational and is a root of the polynomial $x^2 - 2$, thus not transcendental.

Proved by hand first by Hermite in 1874 [11], at least two computer assisted formal proofs of transcendence for e have been provided using HOL Light [8] and Coq [9] proofs assistants. As we demonstrate the opposite in Section III, we do not provide details of these so-called proofs. However, interactive proofs assistants are generally very useful and we propose to demonstrate it with a proof in Coq of Cantor's theorem.

Cantor's theorem states that for any set $\mathcal{A}$, the set of all subsets of $\mathcal{A}$, $\mathcal{P}(\mathcal{A})$, has a strictly greater cardinality than $\mathcal{A}$ itself. Proving that any map $f : \mathcal{A} \to \mathcal{P}(\mathcal{A})$ is not surjective is enough as it leads to $\mathrm{card}(\mathcal{A}) < \mathrm{card}(\mathcal{P}(\mathcal{A}))$. Then, the proof with Coq starts by defining surjectivity:

```
Definition surj(X Y : Type) (f : X → Y) : ℙ := ∀ y, ∃ x, f x = y.
```

We can then state Cantor's theorem with its proof:

```
Theorem Cantor X : ¬ ∃ f : X → X → ℙ, surj f.

Proof.
  intros [f A].
  pose (g := fun x ⇒ ¬ f x x).
  destruct (A g) as [x B].
```

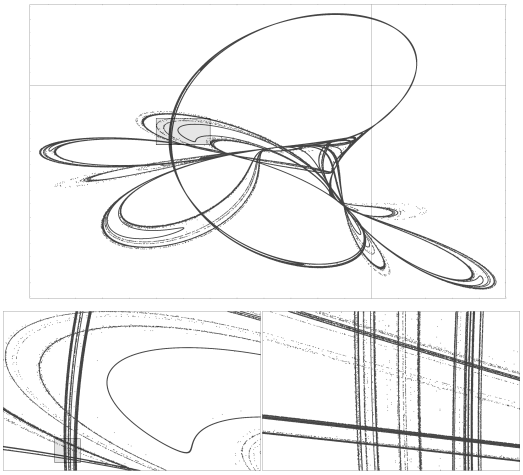

Fig. 1. Tinkerbell map's strange attractor.

```
    assert (C:g x ↔ f x x).
    {
        rewrite B. tauto.
    }
    unfold g in C. tauto.
Qed.
```

The main idea is to have a proof by contradiction where we first assumed that f is surjective. Coq automatically derives parts of the proof using `tauto` and, this way, permits to simplify the process of writing proofs.

B. Interval arithmetic

Many computer assisted proofs consist in verifying numerical hypothesis of some mathematical theorems. A typical numerical hypothesis is $f(x) \geq 0$ for all values of x in some domain. This can be carried out using the so-called interval arithmetic [12], which performs operations on intervals in such a way that the range of some mathematical expressions over an interval is enclosed by the interval evaluation of this expression. For example, one can enclose the range of $f(x) = x^2 - x + 1$ over the interval $[1, 2]$ by evaluating the function with interval arithmetic: $[1, 2] \times [1, 2] - [1, 2] + 1 = [0, 4]$ is a superset of the range $\{f(x) : x \in [1, 2]\} = [1, 3]$. This simple interval evaluation proves in particular that $f(x)$ is nonnegative inside the interval $[1, 2]$. When performed with computers, which work with floating point numbers, interval arithmetic must be implemented with correct rounding so as to enforce its containment property.

Among several fields presenting theorems that are suited to the verification using interval computations, one of the most impressive is the computer assisted proof of the presence of chaos in dynamical systems. The most simple sufficient condition for the presence of chaos for a dynamical system $x_{k+1} = f(x_k)$, with $f : I \to I$ a map of the interval I, is given by the celebrated *Period Three Implies Chaos* theorem [13]: if the map possesses a period 3 orbit, *i. e.*, $f(f(f(x))) = x$, and not a period 2 nor a period 1 orbit, *i. e.*, $f(f(x)) \neq x$ and $f(x) \neq x$, then the map f gives rise to a chaotic dynamical system of the interval I. These hypotheses are perfectly suited to verification using interval computations.

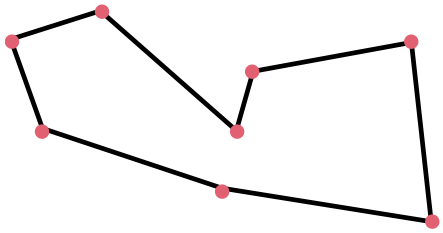

Fig. 2. Example of TSP instance and solution.

Consider for example the quadratic map $f(x) = 1 - 4(x - 1/2)^2$ of the interval $[0, 1]$ and the interval $[0.4, 0.42]$. Using floating point interval arithmetic with correct rounding we prove that $f([0.4, 0.42]) \subseteq [0.9, 1]$ and $f(f([0.4, 0.42])) \subseteq [0.0, 0.2]$. As a consequence, the interval $[0.4, 0.42]$ does not contain any period 2 nor period 1 orbit. Now, again using floating point interval arithmetic with correct rounding we evaluate $f(f(f(x))) - x$ for the intervals $[0.4, 0.4]$ and $[0.42, 0.42]$ and obtain the enclosures $[0.12, 0.13]$ and $[-0.07, -0.06]$ respectively. As a consequence, the intermediate value theorem proves that $f(f(f(x))) - x$ has a zero inside the interval $[0.4, 0.42]$, that is f has a period 3 orbit in this interval. We obtain a numerical certificate that the hypothesis of the *Period Three Implies Chaos* theorem holds true for this map, hence proving that the corresponding dynamical system is chaotic.

Computer assisted proofs of the presence of chaos in maps of higher dimension are more involved, *e. g.*, the computer assisted proof that the Tinkerbell map is chaotic in [14], see Tinkerbell's strange attractor in Fig. 1. Also, continuous time dynamical systems are in the scope of interval computations, *e. g.*, the computer assisted proof that Lorenz attractor is strange by Warwick Tucker [12], hence solving Smale's 14th problem.

C. Computer-assisted proofs Bird*s*

Our formal approach, which we call Bird*s* (proofs Based on mIxed-integeR linear programming Solvers), relies on Mixed-Integer Linear Programming (MILP) solvers to prove various theorems. The MILP formalism consists in describing problems as linear equations, called constraints, which continuous and/or discrete variables have to verify. MILP solvers also handle objective functions and it is then possible to define problems in which we search for variable values that minimize or maximize a linear objective.

By limiting the possibilities to linear equations only, this ensures a simple formalism: users are less prone to errors when describing their problem. Although this limits the theorems we are able to prove with MILP solvers, it still permitted to obtain interesting results which were probably out-of-reach with other tools. Furthermore, in Section III, we will demonstrate that MILP solvers are actually able to help proving theorems that could certainly have a significant impact on mathematics.

But first, we will discuss the traveling salesman problem and the influence of the MILP approach to solve it [15]. Basically,

the problem consists in, given a set of point of interest, finding the shortest possible route that visits every point of interest and that returns to the first visited one. We illustrated this with an instance (set of red dots) and a solution (black line) in Fig. 2. The starting/ending point is not represented as the choice of this point does not impact the solution.

This problem is well studied and was already mentioned in a handbook from the 19th century. Although, this problem has been proven to be NP-hard [16], MacGregor and Ormerod [17] showed that human performance on such problems is very high. However, on very large instances, using computers is mandatory and we are actually able to solve these instances with dedicated approaches and MILP solvers.

One of the classic MILP formulations for the TSP is called the Miller–Tucker–Zemlin (MTZ) formulation and relies on binary variables, $x_{i,j} \in \{0,1\}$, where $x_{i,j} = 1$ indicates that the path (or the solution) goes from point i to point j. Integer variables $u_i \in [\![1,n]\!]$ are also necessary to keep track of the order in which the n points of interest are visited.

Then, with $c_{i,j}$ being the distance between point i and point j, the complete model is as follows:

$$\min \sum_{i=1}^{n} \sum_{j=1,j\neq i}^{n} c_{i,j} x_{i,j}, \tag{1}$$

$$\text{s.t.} \sum_{i=1,i\neq j}^{n} x_{i,j} = 1, \qquad \forall j \in [\![1,n]\!], \tag{2}$$

$$\sum_{j=1,j\neq i}^{n} x_{i,j} = 1, \qquad \forall i \in [\![1,n]\!], \tag{3}$$

$$u_i - u_j + 1 \leq (n-1)\left(1 - x_{i,j}\right), \quad 2 \leq i \neq j \leq n. \tag{4}$$

These constraints ensure that each point as a unique input and a unique output and that the direct precedence between consecutive points is verified.

The above MILP-based model can be used to find the shortest path going through a set of points of interest and solvers are usually able to demonstrate that the provided path is optimal. For example, in 2007, an approach relying on MILP solvers proved that a travel of at least 7,512,218,268 meters is necessary to visit the 1,904,711 cities aggregated for the World TSP Challenge[1]. This theoretical result, proven with Bird<u>s</u>, is important as it permits to know that actual solutions obtained with dedicated approaches and MILP solvers are very close to this bound, hence almost optimal. Note that various applications, such as printed circuit design, require solving a TSP instance, thus Bird<u>s</u> is not limited to theoretical results.

Thanks to technical advances on MILP solvers [18], the Bird<u>s</u> approach will be better with time and we will be able to prove more and more TSP results just by giving today's mathematical models to tomorrow's MILP solvers. Obviously, the Bird<u>s</u> approach is not limited to solving TSP and in the next section we will demonstrate its efficiency on proving theoretical results.

[1]https://www.math.uwaterloo.ca/tsp/world/

III. e IS ACTUALLY RATIONAL

Using the Bird<u>s</u> approach, we demonstrate that e is rational. To do so, we define an MILP-based mathematical model which, once solved, would lead to conclude that $e = x/y$ where $x, y \in \mathbb{N}$. The main idea is to have a constraint in the model which corresponds to

$$e = \frac{x}{y}, \tag{5}$$

where x and y are integer variables. However, divisions cannot be used in MILP-based model and we have to rewrite it as

$$x - e \times y = 0, \tag{6}$$

where $y \neq 0$.

This constraint should not hold as e is believed to be rational [6], [7], [8], [9]. Hence, we replace (6) with

$$\min t \tag{7}$$

$$\text{s.t. } |x - e \times y| \leq t, \tag{8}$$

where $t \in \mathbb{R}_+$ is a continuous variable. This way, we are sure to have a feasible model in any case. Then, the number e is rational if and only if the minimum of this problem is 0.

An absolute value is used in (8) and this nonlinear operation needs to be linearized as follows:

$$x - e \times y \leq t \qquad \text{and} \qquad -(x - e \times y) \leq t. \tag{9}$$

This way, we have a linear model which we solved with CPLEX 22.1.1 [19] using the JuMP [20] modeling language (Julia) on a computer with an Intel® Core™ i7-1365U CPU.

In less than a second, the solver returns that 0 is an optimal solution for this model, *i. e.*, the solution corresponds to $t = 0$ thus $x - e \times y = 0$ and e is rational. Although unexpected, Bird<u>s</u> definitely proved that previous research on the rationality of e was wrong as the simplicity of our model makes it easy to understand and check for correctness.

Unfortunately, a closer look at the results shows that the solver returned $x = 49170.99999499567 \notin \mathbb{N}$. MILP solvers have, generally harmless, integrality tolerance to speed up internal computations. Hence, any real number close enough to an integer is considered to be an integer. This tolerance level can be set with a parameter and for mathematical proofs, zero tolerance is necessary, thus we fixed the solver parameter to 0. We solved the model with this new parameter and obtained 0 as an optimal solution, again. However, this time, the model correctly returned $x, y \in \mathbb{N}$.

This means that, with the Bird<u>s</u> approach, we demonstrated that e is actually rational and that centuries of research on this topic were wrong. Furthermore, we have the exact values for x and y such that $e = x/y$:

$$x = 1084483, \qquad y = 398959. \tag{10}$$

Regrettably, although this may go unnoticed since few people know more than a few digits of the number e, x/y differs from the exact value of e:

$$\frac{x}{y} = 2.7182818284585633, \tag{11}$$

$$e = 2.718281828459045. \tag{12}$$

For the second time, Bird̲s failed us.

In addition to integrality tolerance, MILP solvers usually have multiple other tolerances. Some of them cannot be completely removed but we fixed them to the least possible value. We have therefore set the simplex feasibility and optimality tolerances to 10^{-9} and the absolute MIP gap tolerance to 0. This permits to enhance the working precision of internal subroutines and to ensure that the solver will not stop until the best known solution and the best possible solution are equal.

With these settings, we solved our model again and for the third time, the solver returned 0 as an optimal solution. However, although the integrality tolerance is equal to 0, the provided y is not an integer:

$$x = 28245729, \tag{13}$$

$$y = 10391023 + 1.8626 \cdots \times 10^{-9} \notin \mathbb{N}. \tag{14}$$

This inconsistency between the integrality tolerance setting and the solver result might be a bug of the solver or come from limits of modern computers. In any case, it seems that our approach failed again and that maybe "Bird̲s aren't real".

IV. Mathematically-sound tools

In previous section, we used MILP solvers and tried to prove that e is rational by solving an MILP-based model. At first glance, it seemed that Bird̲s were able to prove the rationality of e but we found errors in the results each time.

Interestingly, it was not even possible to remove some numerical tolerances of the solver and, in our last experiment, the solver actually returned results that were inconsistent with its parameters. We believe that these problems have their origin in floating-point arithmetic [21], [22].

We usually consider MILP-based approaches as mathematical concepts with continuous variables [23], *e. g.*, $t \in \mathbb{R}$ in previous section. Nevertheless, models will be implemented on computers and actually solved using floating-point numbers. In many cases, this has no negative impact on the solution returned by the solver. In some cases, however, classic MILP solvers fail to deliver correct solutions and/or termination certificates, *i. e.*, certificates that the problem is infeasible or that the returned solution is optimal, for example.

Guarantees on solutions is possible and a few solvers have them to ensure reliable results. For example, SoPlex [24] is a linear programming solver which "provides special support for the exact solution of LPs with rational input data." The SCIP MILP solver [25] can use SoPlex as a back-end to port exact solutions to MILP-based model.

However, exact solutions will not suffice in the case presented in Section III: techniques outside of the scope of MILP solvers are necessary to detect that t, although bounded, can always be closer to 0. In an ideal world where infinite precision is possible everywhere, exact solvers will not be able to terminate when optimizing the proposed model.

Interval arithmetic could be a second approach to explore. For example, ibex[2] is a library that provides solutions using reliable algorithms handling nonlinear constraints. In the case of our model for proving the (ir)rationality of e, using interval arithmetic will always lead to $t \in [0, \varepsilon]$ as the returned solution, *i. e.*, the optimal value for t could be 0 but might also never reach 0. Thus, the statement "e is rational" remains a possibility but is not guaranteed nor excluded.

We believe that limits of exact and interval arithmetic methods call for additional checks. However, detecting beforehand all the cases for which the solver will not terminate or return unsatisfactory solutions is certainly not possible and outside checks seem preferable. Another benefit of outside checks is that using efficient MILP solvers, which work correctly in most cases, remains a reasonable choice.

Working on checking and repairing solutions (with post-processing) has been proposed previously [26]. However, repairing solutions is not always possible, in particular in our example with e as [26] requires reasonable bounds on integer variables, which we do not have. Instead, we propose that solvers integrate a simple solution-checking post-process to return a "bug" certificate if necessary. Although hard to admit, one way or another, bugs will always occur and we think it is important to acknowledge this possibility.

To help this solution-checking, we propose a Julia package built on top of JuMP, CheckSolve[3], to automatically check if the provided solution meets the model's constraints. With the model presented in Section III, using CheckSolve we were able to tell that the obtained solutions did not met the constraints by around 10^{-7}, or 10^{-9} once limiting all the numerical tolerances.

V. Conclusion

Starting in 1976, computers have been used to verify existing proofs, to automatically derive proofs or part of them. We also rely on their processing power to perform enumerations which could not be handled by hand. In this work, we presented a way to use an MILP-based approach in order to prove that e is rational. This result is not consistent with existing works which have proven with reliable methods that e is irrational and even transcendental.

A closer look at our result permits to highlight limitations of MILP solvers which, we believe, cannot be easily avoided. This calls for using different approaches such as interval arithmetic or exact solvers. However, in most cases MILP solvers have no issue and efficiently provide interesting results, thus we want to keep using them despite their flaws.

This led us to propose a Julia package, CheckSolve, to automatically verify solutions obtained with solvers. Our approach does not currently permit to verify that a solution said to be optimal is actually so. This is a perspective for future work. Furthermore, we currently rely on floating-point arithmetic for the solution checking while verified interval arithmetic would be preferable.

[2]https://github.com/ibex-team/ibex-lib

[3]https://github.com/remi-garcia/CheckSolve

References

[1] K. Appel and W. Haken, "Every planar map is four colorable," *Bulletin of the American Mathematical Society*, vol. 82, no. 5, pp. 711–712, 1976.

[2] B. Dreben and A. Kanamori, "Hilbert and Set Theory," *Synthese*, vol. 110, no. 1, pp. 77–125, 1997. [Online]. Available: http://www.jstor.org/stable/20117587

[3] A. Neeman, "A counterexample to a 1961 "theorem" in homological algebra," *Inventiones Mathematicae*, vol. 148, no. 2, pp. 397–420, May 2002.

[4] N. Mnev, "On D.K. Biss' papers "The homotopy type of the matroid Grassmannian" and "Oriented matroids, complex manifolds, and a combinatorial model for BU"," 2007.

[5] J.-C. Yoccoz, "Une erreur féconde du mathématicien Henri Poincaré : Le prix en l'honneur des 60 ans du roi Oscar et la découverte des orbites homoclines," *La lettre du Collège de France*, no. 28, pp. 38–42, Apr. 2010.

[6] A. E. Parks, "pi, e, and Other Irrational Numbers," *The American Mathematical Monthly*, vol. 93, no. 9, pp. 722–723, Nov. 1986.

[7] M. Beeson, "Automatic Derivation of the Irrationality of *e*," *Journal of Symbolic Computation*, vol. 32, no. 4, pp. 333–349, Sep. 2001.

[8] J. Bingham, "Formalizing a Proof that e is Transcendental," *Journal of Formalized Reasoning; Vol 4*, no. 1, pp. 71–84, 2011.

[9] S. Bernard, Y. Bertot, L. Rideau, and P.-Y. Strub, "Formal proofs of transcendence for e and pi as an application of multivariate and symmetric polynomials," in *Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs*, ser. CPP 2016. ACM, Jan. 2016.

[10] J. L. Coolidge, "The Number e," *The American Mathematical Monthly*, vol. 57, no. 9, pp. 591–602, Nov. 1950.

[11] C. Hermite, *Sur la fonction exponentielle*. Gauthier-Villars, 1874. [Online]. Available: http://eudml.org/doc/203956

[12] W. Tucker, *Validated Numerics: A Short Introduction to Rigorous Computations*. Princeton University Press, 2011.

[13] T.-Y. Li and J. A. Yorke, "Period three implies chaos," *The American Mathematical Monthly*, vol. 82, no. 10, pp. 985–992, 1975.

[14] A. Goldsztejn, W. Hayes, and P. Collins, "Tinkerbell Is Chaotic," *SIAM Journal on Applied Dynamical Systems*, vol. 10, no. 4, pp. 1480–1501, 2011.

[15] W. Cook, *In pursuit of the traveling salesman*. Princeton [u.a.]: Princeton University Press, 2014.

[16] R. M. Karp, *Reducibility among Combinatorial Problems*. Springer US, 1972, pp. 85–103.

[17] J. N. Macgregor and T. Ormerod, "Human performance on the traveling salesman problem," *Perception & Psychophysics*, vol. 58, no. 4, pp. 527–539, Jun. 1996.

[18] T. Koch, T. Berthold, J. Pedersen, and C. Vanaret, "Progress in mathematical programming solvers from 2001 to 2020," *EURO Journal on Computational Optimization*, vol. 10, p. 100031, 2022.

[19] CPLEX, "CPLEX User's Manual," 2022. [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer

[20] I. Dunning, J. Huchette, and M. Lubin, "JuMP: A Modeling Language for Mathematical Optimization," *SIAM Review*, vol. 59, no. 2, pp. 295–320, May 2017.

[21] J.-M. Muller, N. Brunie, F. de Dinechin, C.-P. Jeannerod, M. Joldes, V. Lefèvre, G. Melquiond, N. Revol, and S. Torres, *Handbook of Floating-Point Arithmetic*. Springer International Publishing, 2018.

[22] "IEEE Standard for Floating-Point Arithmetic," 2019.

[23] L. A. Wolsey, *Integer Programming*. Wiley & Sons, Incorporated, John, 2020.

[24] A. Gleixner and D. E. Steffy, "Linear programming using limited-precision oracles," *Mathematical Programming*, vol. 183, no. 1–2, pp. 525–554, Nov. 2019.

[25] K. Bestuzheva, M. Besançon, W.-K. Chen, A. Chmiela, T. Donkiewicz, J. van Doornmalen, L. Eifler, O. Gaul, G. Gamrath, A. Gleixner, L. Gottwald, C. Graczyk, K. Halbig, A. Hoen, C. Hojny, R. van der Hulst, T. Koch, M. Lübbecke, S. J. Maher, F. Matter, E. Mühmer, B. Müller, M. E. Pfetsch, D. Rehfeldt, S. Schlein, F. Schlösser, F. Serrano, Y. Shinano, B. Sofranac, M. Turner, S. Vigerske, F. Wegscheider, P. Wellner, D. Weninger, and J. Witzig, "The SCIP Optimization Suite 8.0," Optimization Online, Technical Report, Dec. 2021. [Online]. Available: http://www.optimization-online.org/DB_HTML/2021/12/8728.html

[26] A. Neumaier and O. Shcherbina, "Safe bounds in linear and mixed-integer linear programming," *Mathematical Programming*, vol. 99, no. 2, pp. 283–296, Mar. 2004.