

Conflict Graphs in Mixed-Integer Linear Programming: Preprocessing, Heuristics and Cutting Planes

Samuel Souza Brito

Universidade Federal de Ouro Preto (UFOP)
Rua Diogo de Vasconcelos, 122, Ouro Preto - MG
samuelbrito@ufop.edu.br

Haroldo Gambini Santos

Universidade Federal de Ouro Preto (UFOP)
Rua Diogo de Vasconcelos, 122, Ouro Preto - MG
haroldo@ufop.edu.br

RESUMO

Este documento apresenta as principais contribuições da tese de doutorado de Samuel Souza Brito. A tese aborda o desenvolvimento de algoritmos baseados em grafos de conflitos para Programação Linear Inteira Mista, incluindo: (i) uma rotina de pré-processamento que reduz o número de restrições e produz automaticamente formulações mais fortes; (ii) um separador de cortes clique que produz limites duais mais fortes do que aqueles providos pelos geradores equivalentes dos resolvidores CPLEX, CBC e GLPK; (iii) um separador de cortes de ciclo-ímpar com uma nova rotina de *lifting*; (iv) uma infraestrutura eficiente para construir e manipular grafos de conflitos; e (v) duas heurísticas *diving* que geram soluções viáveis em tempos de execução restritos. Uma nova versão do CBC foi criada, incluindo esses algoritmos baseados em grafos de conflitos. A média do *gap* fechado por essa nova versão foi até quatro vezes melhor e o número de problemas resolvidos aumentou em 23.53%.

PALAVRAS CHAVE. Programação Linear Inteira Mista, Grafos de Conflitos, Cliques.

Tópicos (otimização combinatória, programação matemática)

ABSTRACT

This document presents the main contributions of the doctoral thesis of Samuel Souza Brito. The thesis addresses the development of conflict graph-based algorithms for Mixed-Integer Linear Programming, including: (i) a preprocessing routine that reduces the number of constraints and produces stronger formulations; (ii) a clique cut separator that produces dual bounds stronger than those provided by the equivalent cut generators of the CPLEX, CBC and GLPK solvers; (iii) an odd-cycle cut separator with a new lifting module; (iv) an efficient infrastructure for constructing and manipulating conflict graphs; and (v) two diving heuristics that generate feasible solutions in restricted execution times. A new version of the CBC solver was generated with the inclusion of these conflict graph-based algorithms. The average gap closed by this new version was up to four times better and the number of mixed-integer programs solved was increased by 23.53%.

KEYWORDS. Mixed-Integer Linear Programming. Conflict Graphs. Cliques.

Paper topics (combinatorial optimization, mathematical programming))

1. Introduction

This document presents a summary of the doctoral thesis of Samuel Souza Brito under the supervision of Professor Haroldo Gambini Santos. The thesis defense occurred on February 28, 2020, at the Computer Science department of *Universidade Federal de Ouro Preto*, as an event of its graduate program.

Over the last years, Mixed-Integer Linear Programming (MILP) has proven to be a powerful technique for modeling and solving a wide variety of combinatorial optimization problems, most of them with practical interest. A MILP model is formally defined as:

$$c^* = \min \{c^T x \mid Ax \leq b, l \leq x \leq u, x \in \mathbb{R}^n, x_j \in \mathbb{Z} \forall j \in I\} \quad (\text{MILP})$$

where $c \in \mathbb{R}^n$ represents the objective function coefficients, $A \in \mathbb{R}^{m \times n}$ is the constraint matrix and $b \in \mathbb{R}^m$ is the right-hand side (RHS) of the constraints. Vectors $l \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$ are the lower and upper bounds for the decision variables, respectively. Furthermore, $N = \{1, \dots, n\}$ is the index set of the decision variables x and $I \subseteq N$ contains the indices of the variables that need to be integral in every feasible solution.

An implicit structure used by modern MILP solvers in preprocessing and cut separation routines is the conflict graph. A Conflict Graph (CG) is a structure that stores assignment pairs of binary variables which cannot occur in any feasible solution. It consists of an undirected graph with a set of vertices $V = \{x_j, \bar{x}_j : j = 1, \dots, n\}$ and a set of edges $E = \{(u, v) : (u, v) \subset V^2\}$. In such structure, vertex x_j represents the assignment of the associated variable to one ($x_j = 1$), while vertex $\bar{x}_j$ corresponds to set the variable to zero ($x_j = 0$).

An example of a conflict graph with three variables $\{x_1, x_2, x_3\}$ is presented in Figure 1. There is an edge linking each variable to its complement since only one must be equal to one in any feasible solution. Dashed lines in Figure 1 denote these trivial conflicts. The conflict graph of this figure generates the three edge inequalities presented to the right of it.



Figura 1: Example of a conflict graph and its associated edge inequalities.

2. Main Contributions

The thesis was motivated by the importance of MILP in solving a wide variety of combinatorial optimization problems. The development of techniques that improve the performance of MILP solvers contributes directly to solve different classes of problems, including real-world ones. Given this motivation, the main contribution of this thesis is to develop conflict graph-based techniques to accelerate the process of solving MILP models. All the proposed techniques were implemented in the state-of-the-art open-source MILP solver COIN-OR Branch-and-Cut (CBC) solver¹ and are available for the operations research community. These contributions resulted in a much faster MILP solver and received the COIN-OR Cup 2019 award².

¹<https://github.com/coin-or/Cbc>

²<https://www.coin-or.org/2019/10/24/2019-coin-or-cup-winners-announced/>

Initially, we proposed and implemented a conflict graph infrastructure, characterized by the efficient construction and handling of such graphs. Our routine for building CGs is an improved version of the conflict extraction algorithm presented by Achterberg [2007]. The basis for the improvement is a new step for detecting additional maximal cliques without increasing the computational complexity of the algorithm. We also developed optimized data structures that selectively store conflicts pairwise or grouped in cliques to handle dense CGs without incurring excessive memory usage.

After developing the infrastructure for CGs, we used the information provided by this structure to implement a preprocessing routine and two cut separators. The preprocessing routine consists of extending set packing constraints by the inclusion of additional conflicting variables. A greedy algorithm uses the information from the CG to augment the cliques formed by the set packing constraints. After executing the clique extension algorithm, all constraints that become dominated are removed. Computational results show that our preprocessing routine was responsible for reducing the number of constraints, strengthening the initial dual bounds and accelerating the process of proving optimality for a great number of instances.

The two conflict-based cut separators that we developed are responsible for separating clique and odd-cycle cuts. Our clique cut separator was capable of obtaining dual bounds at the root node LP relaxation that are 19.65% stronger than those provided by the equivalent cut generator of IBM ILOG CPLEX³ solver. Furthermore, the dual bounds produced with the execution of our clique cut separator were 362% times better than those attained by the clique cut separator of the GNU Linear Programming Kit (GLPK)⁴ solver and 422% times stronger than the dual bounds obtained by the clique cut separation routine of the COIN-OR Cut Generation Library (CGL)⁵.

Our conflict graph infrastructure, preprocessing routine and cut separators were included in the CBC solver. CBC is one of the fastest open-source MILP solvers nowadays and it is also a fundamental component used by Mixed-Integer Nonlinear solvers, such as Bonmin [Belotti et al., 2009] and Couenne [Bonami et al., 2008]. It is worth mentioning that the design and implementation of enhanced conflict graph-based cut generation routines is an important pending task in the COIN-OR development for at least one decade⁶. Experiments revealed that the average gap closed by the new version of CBC was up to four times better than its previous version. Moreover, the time spent proving the optimality for the MILP models decreased and the number of instances solved increased by 23.53%. The source codes derived from the thesis are included in the development version of the CBC solver and can be downloaded from the official repository of this solver.

Additionally, we proposed and implemented two conflict-based diving heuristics, called Conflict Diving and Modified Degree Diving. These heuristics first adjust the bounds of the variables which are more likely to cause infeasibilities. In this case, one heuristic considers the degree and the other uses the modified degree of the variables at the CG to implement the variable selection strategies. Both proposed diving heuristics presented execution times smaller than the classical diving heuristics that we evaluated in our experiments. Moreover, the heuristic that uses the modified degree in its variable selection strategy found the greatest number of feasible solutions among the heuristics evaluated.

³<https://www.ibm.com/analytics/cplex-optimizer>

⁴<https://www.gnu.org/software/glpk/>

⁵<https://github.com/coin-or/Cgl>

⁶<https://projects.coin-or.org/Events/wiki/CoinORSoCIdeaList>

2.1. Published Papers, Conference Presentations and Awards

The following publications were derived from the thesis: (i) a journal article [Brito e Santos, 2021] that presents our efficient conflict graph infrastructure and the clique and odd-cycle cut separators, as well as extensive computational experiments; (ii) a conference article [Brito et al., 2015] that introduces the preliminary versions of our algorithm for building CGs and the two cut separation routines; (iii) a conference abstract [Brito e Santos, 2018] that shows some computational results obtained with the inclusion of the preliminary versions of our graph-based techniques in the CBC solver; and (iv) a conference abstract [Brito et al., 2019] that presents our two conflict-based diving heuristics and employs machine learning techniques to produce an adaptive diving algorithm.

The work developed in the thesis was also awarded as the winner of the COIN-OR CUP 2019⁷. The COIN-OR Cup recognizes and celebrates the best contributions to open-source operations research software development and use associated with the COIN-OR software collection. The award ceremony took place at the INFORMS Annual Meeting in Seattle, USA.

Our cut separators were also successfully applied to three classical combinatorial optimization problems: Capacitated Vehicle Routing [Pecin et al., 2017], Project Scheduling [Araujo et al., 2020] and Nurse Rostering [Santos et al., 2016]. The use of these routines contributed to the solution of several hard instances for the first time in the literature.

3. Computational Results

This section presents some of the main results obtained in the thesis. The instances used in the experiments consist of 320 MILP models divided into five groups: set *bmc* contains instances of Bandwidth Multicoloring Problem [Dias et al., 2016]; set *bpwc* has instances of Bin Packing Problem with Conflicts [Sadykov e Vanderbeck, 2013]; set *miplib* is composed by instances of the current and previous versions of the Mixed-Integer Problem Library (MIPLIB) benchmark set [Gleixner et al., 2018]; set *rostering* is formed by instances of Nurse Rostering problem [Haspeslagh et al., 2014]; and set *timetabling* contains instances of Educational Timetabling [Fonseca et al., 2017].

3.1. Building Conflict Graphs

An experiment was conducted to compare the performance of our algorithm for building CGs, named as *ICE*, against the pairwise inspection scheme of Atamtürk et al. [2000], referred to here as *PI*, and the clique extraction algorithm of Achterberg [2007], denoted as *CE*. In this experiment, *PI* failed to construct graphs for eight instances since it needed more than 32 GB of memory to construct and store CGs for these instances. They have some set packing and set partitioning constraints formed by a large number of variables, whose pairwise storage of conflicts results in excessive memory consumption.

ICE built and stored CGs for the considered instance problems spending, on average, 252.47 MB of memory and 7.60 seconds. In comparison with *PI*, these results represent a decrease of 85.66% in memory consumption and 87.20% in the execution time.

Additionally, *ICE* was capable of found more conflicts than *CE* in 27 instances. The number of new conflicts refers to the number of new edges that were included in the conflict graphs. It is the case of instance *uELGN_BPWC_3_2_18*, where more than 8.7 million of new edges were inserted in the associated CG. The highest increase of conflicts, in percentage, was obtained in instance *istanbul-no-cutoff*, in which *ICE* is responsible for detecting 127.78% more conflicts.

⁷<https://www.coin-or.org/2019/10/24/2019-coin-or-cup-winners-announced/>

3.2. Clique Strengthening

We also evaluated the ability of our preprocessing routine to produce strengthened formulations and to reduce the size of MILPs with respect to the number of constraints. To assess this, we ran our clique strengthening routine limiting the execution of the algorithm to constraints with at most 128 variables. After performing clique strengthening, we used the *COIN-OR Linear Program Solver* (CLP)⁸ to solve the LP relaxation, and then we calculated the gap closed.

Despite the low impact on improving the values of the LP relaxation, there was a significant reduction in the number of constraints of the MILPs from *bmc* and *bpwc*. The percentage of constraints eliminated in instances from set *bmc* was more than 85% and above 71% in instances from *bpwc*. For 48 instances of *miplib*, our preprocessing routine was able to close the gap by up to 99% and reduce the number of constraints by up to 99.91%. The execution of the clique strengthening routine in set *rostering* allowed a significant improvement in the values of LP relaxations, closing the gap by up to 89%. A significant improvement was also observed in the value of the LP relaxation on instances of set *timetabling*. The preprocessing routine was able to reduce the number of constraints of instances *trd445c* and *trdta0010* in 98% and 80%, respectively. Moreover, the integrality gaps in these instances were completely closed.

3.3. Clique Cut Separator

We evaluated the ability of our clique cut separator, referred to here as *bkclq*, in tightening the LP relaxations. For this, we compared the performance of our clique cut separator against the clique cut separators implemented in three MILP solvers. The first clique cut separator that we compared, referred to here as *cglclq*, is used by the CBC solver and provided by the COIN-OR Cut Generation Library (CGL). The second clique cut separator compared in this experiment, named here as *glpclq*, is provided by the open-source solver of the GNU Linear Programming Kit (GLPK) version 4.65. The last clique cut separator that we compared was the one included in the commercial solver IBM ILOG CPLEX version 12.8, referred to here as *cpyclq*, considering the “very aggressively” strategy for separating cuts. All cut separators were executed at the root node LP relaxation of the instance problems, considering at most 50 iterations of the cut generation loop and a time limit of 10,800 seconds.

The average gap closed by the clique cut separators at each iteration of the cut generation loop is presented in Figure 2. Differences in the performances of the cut separators can be seen in the earlier steps of the separation process.

After 50 iterations, the average gap closed by the cut separators was 4.33% for *cglclq*, 5.05% for *glpclq*, 15.27% for *cpyclq* and 18.27% for *bkclq*. Thus, the average gap closed by our clique cut separator was 4.22 times better than the clique cut separator of CGL, 3.62 times better than the clique cut separator provided by GLPK and 19.65% better than the clique cut separator of CPLEX.

Besides producing better dual bounds, *bkclq* was responsible for completely closing the gap for a greater number of instances. Cut separators *bkclq*, *cpyclq* and *glpclq* completely closed the gap for 8, 5 and 2 instances, respectively. Cut separator *cglclq* was not able to completely close the gap for any instance.

3.4. Improving COIN-OR Branch-and-Cut Solver

A new version of the CBC solver was generated, including our algorithms and data structures in the source code of its development version. This new version of CBC is denoted here as *cbc+cg*. We investigated the performance improvement of the new version of CBC against its

⁸<https://github.com/coin-or/Clp>

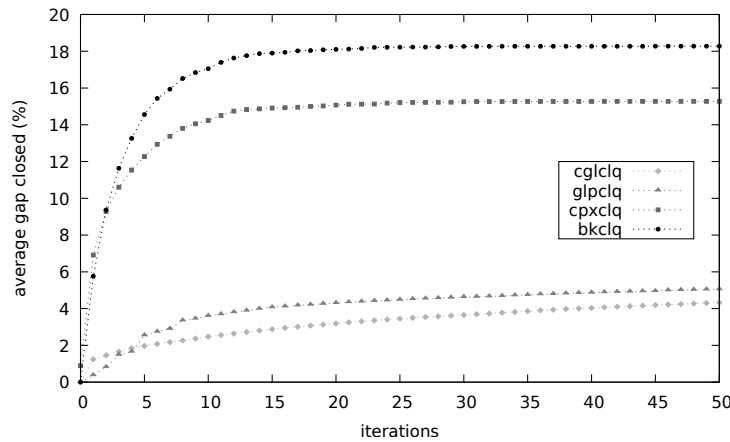


Figura 2: Average gap closed by each clique cut separator.

previous version, referred to here as *cbc*. Each version was run on the 320 MILPs of our dataset considering a time limit of 10,800 seconds.

The inclusion of our conflict graph-based algorithms and data structures in CBC contributed significantly to improve the dual bounds obtained by this solver. The percentage of gap closed by *cbc+cg* was greater than or equal to that obtained by *cbc* in all 320 instances. Consequently, the average gap closed by CBC increased from 58.86% to 68.76%, representing an improvement of 16.82%. Reductions in the execution times were observed in several instances.

At the end of the executions, the average gap closed by *cbc+cg* was four times better than the one obtained by *cbc* in instance set *bmc*. Furthermore, the average gap closed by the new version of CBC was 54.01% better in instance set *bpwc*, and 66.45% better in *rostering*. In *timetabling*, the average gap closed by CBC increased in 14.74%. The smallest improvement in the average gap closed was 8.95%, obtained in the instances of *miplib*.

We also investigated the evolution of the number of instances solved by each version of CBC over time. These results are provided in Figure 3.

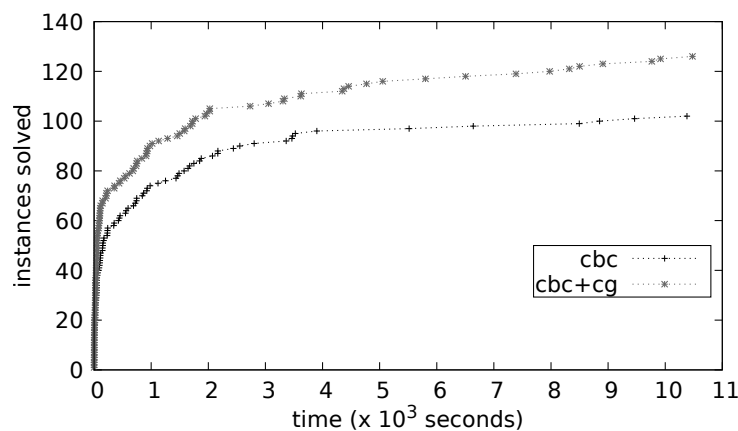


Figura 3: Number of instances solved over three hours.

The previous version of CBC was able to prove the optimality for 102 instances, while the new version proved the optimality for 126 instances. This result represents an increase of 23.53% in the number of instances solved. Moreover, the use of our conflict graph-based algorithms and data structures not only increased the number of instances solved but also decreased the execution time necessary for doing so.

3.5. Conflict-Based Diving Heuristics

We evaluated our two conflict-based diving heuristics concerning the ability to generate feasible integer solutions. For this, we ran Conflict Diving and Modified Degree Diving for each instance, stopping the executions when one of the following conditions holds: (i) an integer feasible solution is found; (ii) the LP relaxation is infeasible; or (iii) execution time reached 10,800 seconds.

For comparison purposes, we implemented four of the most common diving heuristics: Fractional Diving, Coefficient Diving, Linesearch Diving and Vectorlength Diving [Berthold, 2006]. The diving heuristic that found the greatest number of feasible solutions was the Modified Degree Diving, which was able to produce by up 72.41% more solutions than the other heuristics. Even though this heuristic does not consider the objective coefficient of the variables, it found the best solutions for 83 instances. Moreover, Conflict Diving and Modified Degree Diving obtained the best execution times.

The relation between the number instances where a feasible solution was found and the execution time of the heuristics is presented in Figure 4. In this figure it is possible to observe that Modified Degree Diving generated a greater number of feasible solutions, spending smaller execution times when compared with the other diving heuristics.

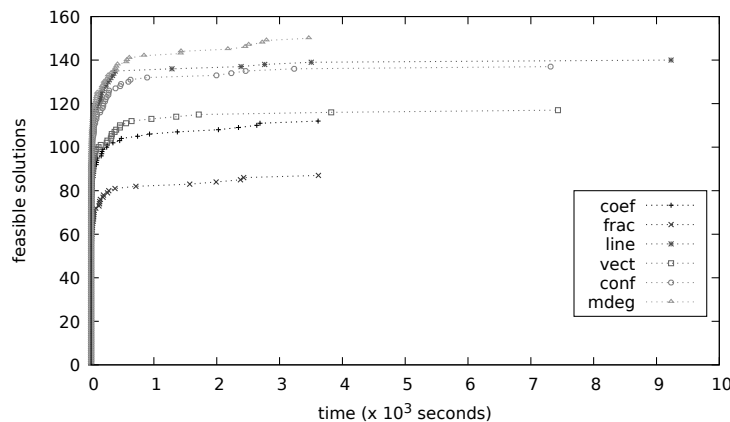


Figura 4: Number of feasible solutions found by each diving heuristic over the time.

Most of the feasible solutions found by the diving heuristics were discovered in the earlier execution times. For example, the feasible solutions for 125 of 150 instances found by Modified Degree diving were generated in less than 100 seconds.

4. Final Considerations

This document presented some of the main contributions and results obtained in the thesis. All details of the development and computational results are available in the full text of the thesis, available at the UFOP thesis repository⁹.

⁹<http://www.repositorio.ufop.br/handle/123456789/12386>

Referências

- Achterberg, T. (2007). *Constraint Integer Programming*. PhD thesis, Technische Universität Berlin, Berlin, Germany.
- Araujo, J. A., Santos, H. G., Gendron, B., Jena, S. D., Brito, S. S., e Souza, D. S. (2020). Strong bounds for resource constrained project scheduling: Preprocessing and cutting planes. *Computers & Operations Research*, 113:104782. ISSN 0305-0548.
- Atamtürk, A., Nemhauser, G. L., e Savelsbergh, M. W. (2000). Conflict graphs in solving integer programming problems. *European Journal of Operational Research*, 121(1):40 – 55. ISSN 0377-2217.
- Belotti, P., Lee, J., Liberti, L., Margot, F., e Wächter, A. (2009). Branching and bounds tightening techniques for non-convex minlp. *Optimization Methods and Software*, 24(4-5):597–634.
- Berthold, T. (2006). *Primal heuristics for mixed integer programs*. diploma thesis, Technische Universität Berlin, Berlin, Germany.
- Bonami, P., Biegler, L. T., Conn, A. R., Cornuéjols, G., Grossmann, I. E., Laird, C. D., Lee, J., Lodi, A., Margot, F., Sawaya, N., e Wächter, A. (2008). An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186–204. ISSN 1572-5286.
- Brito, S. S., Santos, H. G., e Vanden Berghe, G. (2019). Machine learning based diving for mixed integer programming: Decision trees. In *30th European Conference on Operational Research (EURO2019)*, Dublin, Ireland.
- Brito, S. S. e Santos, H. G. (2018). Improving coin-or cbc mip solver using conflict graphs. In *23rd International Symposium on Mathematical Programming (ISMP2018)*, Bordeaux, France.
- Brito, S. S. e Santos, H. G. (2021). Preprocessing and cutting planes with conflict graphs. *Computers & Operations Research*, 128:105176. ISSN 0305-0548.
- Brito, S. S., Santos, H. G., e Poggi, M. (2015). A computational study of conflict graphs and aggressive cut separation in integer programming. *Electronic Notes in Discrete Mathematics*, 50:355 – 360. ISSN 1571-0653. LAGOS'15 – VIII Latin-American Algorithms, Graphs and Optimization Symposium.
- Dias, B., de Freitas, R., Maculan, N., e Michelon, P. (2016). Constraint and integer programming models for bandwidth coloring and multicoloring in graphs. In *Proceedings of the XLVIII Brazilian Symposium on Operations Research*, p. 4116–4127.
- Fonseca, G. H., Santos, H. G., Carrano, E. G., e Stidsen, T. J. (2017). Integer programming techniques for educational timetabling. *European Journal of Operational Research*, 262(1):28 – 39.
- Gleixner, A., Hendel, G., Gamrath, G., Achterberg, T., Bastubbe, M., Berthold, T., Christophel, P., Jarck, K., Koch, T., Linderoth, J., Lübbecke, M., Mittelman, H. D., Ozyurt, D., Ralphs, T. K., Salvagnin, D., e Shinano, Y. (2018). MIPLIB 2017.
- Haspeslagh, S., De Causmaecker, P., Schaerf, A., e Stølevik, M. (2014). The first international nurse rostering competition 2010. *Annals of Operations Research*, 218(1):221–236. ISSN 0254-5330.
- Pecin, D., Pessoa, A., Poggi, M., e Uchoa, E. (2017). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, 9(1):61–100.
- Sadykov, R. e Vanderbeck, F. (2013). Bin packing with conflicts: A generic branch-and-price algorithm. *INFORMS Journal on Computing*, 25(2):244–255.
- Santos, H. G., Toffolo, T. A. M., Gomes, R. A. M., e Ribas, S. (2016). Integer programming techniques for the nurse rostering problem. *Annals of Operations Research*, 239(1):225–251.