

Primal Heuristics for Branch-and-Price Algorithms

Marco Lübbecke and Christian Puchert

Abstract In this paper, we present several primal heuristics which we implemented in the branch-and-price solver `GCG` based on the `SCIP` framework. This involves new heuristics as well as heuristics from the literature that make use of the reformulation yielded by the Dantzig-Wolfe decomposition. We give short descriptions of those heuristics and briefly discuss computational results. Furthermore, we give an outlook on current and further development.

1 Introduction

A wealth of practical decision and planning problems in operations research can be modeled as mixed integer programs (MIPs) with a special structure; this structure has been recently successfully exploited in *branch-and-price* algorithms [6] which are based on *Dantzig-Wolfe decomposition* [5]; we assume the reader to be familiar with both concepts. *Primal heuristics* are a very important aspect of MIP solving: In practice, the decision maker is often satisfied with a “good” solution to her problem; besides, such solutions can help accelerate the solution process. We developed and tested heuristics specially tailored for branch-and-price that exploit the fact that two different formulations of the MIP are available via the Dantzig-Wolfe decomposition: the original MIP, and the extended reformulation.

We implemented our heuristics in the branch-and-price solver `GCG` [8] which is based on the well-known `SCIP` framework [1]. Whereas today’s branch-and-price algorithms are almost always tailored to the application and are thus “reserved to experts,” the goal of the `GCG` project is to provide a more generally accessible solver

Marco Lübbecke

RWTH Aachen University, Operations Research, Templergraben 64, 52056 Aachen, Germany
e-mail: marco.luebbecke@rwth-aachen.de

Christian Puchert

RWTH Aachen University, Operations Research, Templergraben 64, 52056 Aachen, Germany
e-mail: puchert@or.rwth-aachen.de

that is available to a much broader range of users. Our work contributes towards this goal by providing primal heuristics for branch-and-price.

In the remainder, we will present three types of such heuristics—called extreme point, restricted master, and column selection heuristics—and discuss computational results. This work is based on the master’s thesis of the second author [11].

2 Preliminaries

We want to solve MIPs of the form

$$\begin{aligned} \min \quad & c^T x \\ \text{s. t.} \quad & Ax \geq b \\ & x \in X, \end{aligned} \tag{1}$$

with $X = \{x \in \mathbb{R}^n : Dx \geq d, x_i \in \mathbb{Z} \text{ for all } i \in I \subseteq \{1, \dots, n\}\}$. Typically, $Ax \geq b$ are some “complicated” or “linking” constraints, whereas the set X often has a block structure $X = X_1 \times \dots \times X_K$ with $X_k = \{x^k \in \mathbb{R}^{n_k} : D^k x^k \geq d^k, x_i^k \in \mathbb{Z} \text{ for all } i \in I_k\}$, $k = 1, \dots, K$, $I_k \subseteq \{1, \dots, n_k\}$, $\sum_{k=1}^K n_k = n$.

Via Dantzig-Wolfe decomposition, (1) is reformulated to a *master problem*

$$\begin{aligned} \min \quad & \sum_{k=1}^K \sum_{p \in P_k} c^p \lambda_{kp} \\ \text{s. t.} \quad & \sum_{k=1}^K \sum_{p \in P_k} a^p \lambda_{kp} \geq b \\ & \sum_{p \in P_k} \lambda_{kp} = 1 \quad \text{for all } k \\ & \lambda_k \in \mathbb{R}_+^{|P_k|} \quad \text{for all } k, \end{aligned} \tag{2}$$

where integrality of the x_i is enforced by constraints $\sum_{p \in P_k} \lambda_{kp} x^{kp} = x^k$ for all k and $x_i^k \in \mathbb{Z}$ for all $i \in I_k$ (*convexification*) or $\lambda_k \in \mathbb{Z}_+^{|P_k|}$ for all k (*discretization*). The master problem (2) bears too many variables to explicitly work with and column generation is typically applied. The pricing subproblems to generate variables as needed read as

$$\begin{aligned} \min \quad & (c^k)^T x^k - (\pi^k)^T A^k x^k - \pi_0 \\ \text{s. t.} \quad & x^k \in X_k, \end{aligned} \tag{3}$$

one for each block k .

3 Extreme Points Crossover

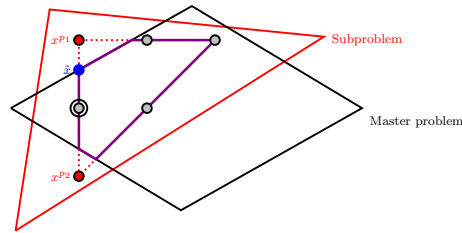
Each component \tilde{x}^k of the solution \tilde{x} yielded by solving the master problem (2) is a convex combination $\tilde{x}^k = \sum_{p \in P_k} \tilde{\lambda}_{kp} x^{kp}$, with P_k being the (index) set of extreme points obtained by solving the k -th pricing problem. These are integer feasible and satisfy the constraints $D^k x^k \geq d^k$, while \tilde{x}^k satisfies all linear constraints but is usually not integer feasible. This leads to the question: Can we exploit the integer feasibility of the x^{kp} to obtain a solution \tilde{x} satisfying *all* constraints? This is what extreme point based heuristics try to achieve. The extreme points we are particularly interested in are those with $\tilde{\lambda}_{kp} > 0$.

It may happen that there is a number of coordinates in which these extreme points are identical. Especially when their values are also shared by \tilde{x} , it seems worth exploring whether there are more integer feasible or feasible points taking the same values. Therefore, *Extreme Points Crossover* performs a neighborhood search on the extreme points by *crossing* them: For each block, it takes those x^{kp} with the highest $\tilde{\lambda}_{kp}$, fixes the variables in which the selected points are identical and solves a sub-MIP.

As its name indicates, the heuristic can be compared to the Crossover heuristic discussed in [4, 12] and implemented in SCIP, which considers a number of already known feasible solution for crossing. The main advantage of our heuristic against this one is that it does not need any previously found feasible solution.

Fig. 1 Extreme Points

Crossover: An LP feasible solution \tilde{x} is a convex combination of $x^{p1}, x^{p2} \in \text{conv}(X)$. All the three points have a coordinate in common which is also shared by the encircled feasible point.



4 Restricted Master

When we solve the master problem, we dynamically add variables that improve the current solution. As the total number of master variables may be exponentially high compared to the number of original variables, this means that the master problem may soon reach a size where searching for feasible solutions would become very time-consuming. This is what the *Restricted Master* heuristic (suggested by Joncour et al. [9]) tries to overcome. It searches for an integer feasible master solution by restricting the formulation again to a subset of promising variables, hence regarding a problem which is of considerably smaller size than the current master formulation.

That is, we solve a sub-MIP of the master formulation (without adding any new variables to it) where all variables not contained in this subset are fixed to zero. As our subset, we chose those master variables which are part of the master LP solution, i.e., the λ_p with $\tilde{\lambda}_p > 0$.

5 Column Selection

Column Selection—also suggested by Joncour et al. [9]—is a constructive approach to build feasible solutions from scratch, with only the knowledge of extreme points previously generated by the pricing problems. In the discretization approach, integrality is enforced by integrality constraints on the master variables. This in particular means that choosing one x^{kp_k} , $p_k \in P_k$, for each block k and building the Cartesian product will lead to a point $\bar{x} = x^{1p_1} \times \dots \times x^{Kp_K}$ that is feasible w.r.t. the integrality and the subproblem constraints; only the master constraints may be violated.

The heuristic constructs a product as above by successively choosing x^{kp_k} previously generated by the subproblems, while trying to avoid master infeasibility. Technically, this happens by increasing the corresponding master variable λ_{kp_k} in the master problem by one.

There are different approaches trying to achieve that, we implemented the following two:

- *Greedy Column Selection*: Starting with $\bar{\lambda} = 0$, increase a master variable such that the number of currently violated master constraints is decreased as much as possible. As a tie breaker, one can optionally use the objective coefficients of the λ_{kp_k} variables;
- *Relaxation Based Column Selection*: Starting with a rounded down master LP solution $\lfloor \tilde{\lambda} \rfloor$, we increase master variables, preferring those which actually have been rounded (i.e., the λ_{kp_k} with $\tilde{\lambda}_{kp_k} > 0$); this method can also be viewed as a rounding heuristic on the master variables.

6 Results

We ran GCG with the above mentioned heuristics on four types of combinatorial optimization problems: bin packing, vertex coloring, capacitated p -median and resource allocation. These problems bear a bordered block-diagonal structure or at least can be brought into one which can be easily exploited by GCG. Apart from that, we also tested how several of the default SCIP heuristics [4]—such as rounding heuristics, RENS, diving heuristics, and improvement heuristics—perform within the branch-and-price scenario (that is, when applied to the original x variables).

Previously, no primal heuristics had been implemented in GCG; only the default SCIP heuristics on the master variables were used. Now, the new heuristics as well as the SCIP heuristics on the original variable space are available to GCG.

The heuristics did unfortunately not so much improve the solution time of GCG. This is actually no big surprise since it already performed well on especially the bin packing and the smaller coloring instances before the heuristics had been implemented. However, solutions could be found and the number of branch-and-bound nodes could be reduced on a large number of the problems on which the heuristics were tested; in particular, they were successful on some instances where GCG previously failed.

		Restricted Master	Column Selection	Extreme Point
BIN	Solutions found [% inst.]	81.48	74.07	50.00
	Best solution [% inst.]	29.63	0.00	11.11
	Running time [sec, total]	4	17	3226
	Running time [sec, geom. mean]	1.00	1.04	16.48
COLEASY	Solutions found [% inst.]	76.67	60.00	60.00
	Best solution [% inst.]	10.00	3.33	46.67
	Running time [sec, total]	6	14	329
	Running time [sec, geom. mean]	1.02	1.09	3.86
COLHARD	Solutions found [% inst.]	43.48	52.17	4.35
	Best solution [% inst.]	30.43	39.13	4.35
	Running time [sec, total]	40	47	1168
	Running time [sec, geom. mean]	1.48	1.66	5.39
CPMP	Solutions found [% inst.]	0.00	0.00	71.43
	Best solution [% inst.]	0.00	0.00	7.79
	Running time [sec, total]	10	324	681
	Running time [sec, geom. mean]	1.00	2.37	4.75
RAP	Solutions found [% inst.]	92.86	95.24	92.86
	Best solution [% inst.]	16.67	2.38	42.85
	Running time [sec, total]	3	0	67
	Running time [sec, geom. mean]	1.00	1.00	1.52

Table 1 Results: For each heuristic and each test set, we indicate in how many percent of the instances the heuristics found a solution, how often it found the best known solution and how much time it spent over all instances in total and on average, respectively.

Besides, we also had test runs on some MIPLIB2003 [2] and MIPLIB2010 [10] instances; recently, there has been made progress in detecting structures in such general MIPs even if it is not apparently visible (see [3] for details). Our results there indicate that the heuristics are also of use in that general setting; in particular, we recently succeeded with applying diving heuristics to MIPLIB2003 instances. The GCG versions of *Coefficient Diving* and *Fractional Diving*—which use the master LP instead of the original LP—were able to reduce the computational effort for the instances `10teams`, `modglob`, `p2756`, `pp08aCUTS`, and `rout`.

7 Conclusion & Outlook

The heuristics presented here are able to generate feasible solutions for quite a number of problem instances. Yet, there still lies some potential in them. Currently, we are improving the *Extreme Points Crossover* heuristic and the GCG diving heuristics.

In addition, we are developing a column generation based variant of the *Feasibility Pump* heuristic by Fischetti et al. [7]. While the original *Feasibility Pump* decom-

poses the problem into linear and integer constraints, our heuristic distinguishes—like the Dantzig-Wolfe decomposition—between master and subproblem constraints. That is, by alternately solving modified master problems and subproblems, we try to construct two sequences of points, hopefully converging towards each other and yielding a feasible solution.

Often, like in stochastic programs or other multi-stage decision problems, a matrix has a staircase structure, i.e., we have *linking variables* belonging to multiple blocks. The presence of such variables is undesirable since in order to apply a Dantzig-Wolfe decomposition, one needs to copy these variables and introduce additional master constraints. A way to overcome this issue is to heuristically fix them, which is what *fifty-fifty* heuristics do. Another possibility is letting diving heuristics branch on those variables, hoping that they will get fixed by doing so.

The results show that our heuristics have now become an important part of GCG; a number of them are now included by default. This motivates us to further pursue our research in this important area of computational integer programming.

References

1. Achterberg, T.: SCIP: Solving constraint integer programs. *Mathematical Programming Computation* **1**(1), 1–41 (2009). <http://mpc.zib.de/index.php/MPC/article/view/4>
2. Achterberg, T., Koch, T., Martin, A.: MIPLIB 2003. *Operations Research Letters* **34**(4), 361–372 (2006). <http://mipilib.zib.de>
3. Bergner, M., Caprara, A., Furini, F., Lübbecke, M., Malaguti, E., Traversi, E.: Partial convexification of general MIPs by Dantzig-Wolfe reformulation. In: Günlük, O., Woeginger, G.J. (eds.), *Integer Programming and Combinatorial Optimization (IPCO 2011)*, LNCS, 6655, pp. 39–51. Springer, Berlin (2011)
4. Berthold, T.: Primal Heuristics for Mixed Integer Programs. Master’s thesis. Technische Universität Berlin (2006)
5. Dantzig, G.B., Wolfe, P.: Decomposition Principle for Linear Programs. *Operations Research* **8**(1), 101–111 (1960)
6. Desrosiers, J., Lübbecke, M.: A Primer in Column Generation. In: Desaulniers, G., Desrosiers, J., Solomon, M.M. (eds.), *Column Generation*, pp. 1–32. Springer, US (2005)
7. Fischetti, M., Glover, F., Lodi, A.: The feasibility pump. *Mathematical Programming* **104**, 91–104 (2005)
8. Gamrath, G., Lübbecke, M.: Experiments with a Generic Dantzig-Wolfe Decomposition for Integer Programs. In: Festa, P. (ed.), *Experimental Algorithms*, LNCS, 6049, pp. 239–252. Springer, Berlin (2010)
9. Joncour, C., Michel, S., Sadykov, R., Sverdlov, D., and Vanderbeck, F.: Column Generation based Primal Heuristics. *Electronic Notes in Discrete Mathematics* **3**, 695–702 (2010). ISCO 2010 – International Symposium on Combinatorial Optimization.
10. T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelman, T. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter. MIPLIB 2010. *Mathematical Programming Computation*, 3(2):103–163, 2011.
11. Puchert, C.: Primal Heuristics for Branch-and-Price Algorithms. Master’s thesis. Technische Universität Darmstadt (2011)
12. Rothberg, E.: An Evolutionary Algorithm for Polishing Mixed Integer Programming Solutions. ILOG Inc. (2005)