Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Discrete Mathematics 36 (2010) 695–702
Electronic Notes in DISCRETE MATHEMATICS
www.elsevier.com/locate/endm

Column Generation based Primal Heuristics

C. Joncour(1,3), S. Michel (2), R. Sadykov (3,1), D. Sverdlov (3), F. Vanderbeck (1,3)

(1) Institut de Mathmatiques, Universit Bordeaux 1, France
(2) Institut suprieur d'tudes logistiques, Universit du Havre, France
(3) Team RealOpt, INRIA Bordeaux-Sud-Ouest, France

Abstract

In the past decade, significant progress has been achieved in developing generic primal heuristics that made their way into commercial mixed integer programming (MIP) solver. Extensions to the context of a column generation solution approach are not straightforward. The Dantzig-Wolfe decomposition principle can indeed be exploited in greedy, local search, rounding or truncated exact methods. The price coordination mechanism can bring a global view that may be lacking in some "myopic" approaches based on a compact formulation. However, the dynamic generation of variables requires specific adaptation of heuristic paradigms. The column generation literature reports many application specific studies where primal heuristics are a key to success. There remains to extract generic methods that could be seen as black-box primal heuristics for use across applications. In this paper we review generic classes of column generation based primal heuristics. We then focus on a so-called "diving" method in which we introduce diversification based on Limited Discrepancy Search. While being a general purpose approach, the implementation of our heuristic illustrates the technicalities specific to column generation. The method is numerically tested on variants of the cutting stock and vehicle routing problems.

Keywords: Primal Heuristic, Column Generation.

doi:10.1016/j.endm.2010.05.088

1 Introduction

Heuristics are algorithms that attempts to derive "good" primal feasible solutions to a combinatorial optimization problem. They include constructive methods that build a solution and improvement methods such as local search procedure that starts with an incumbent. The term "primal heuristic" generally refers to methods based on the tools of exact optimization, truncating an exact procedure or constructing solutions from the relaxation on which the exact approach relies: techniques range from greedy constructive procedures to rounding a solution of the linear programming (LP) relaxation, using the LP solution to define a target, or simply exploiting dual information for pricing choices. Alternatively, exact solvers can be used as subroutines in building heuristic solutions, for instance to explore a neighborhood in a local search procedure. Today's MIP solvers rely heavily on generic primal heuristics: high quality primal values help pruning the enumeration by bound and preprocessing; they are also essential in tackling large scale real-life applications where the exact solver is given limited running time.

Heuristics based on exact methods have found a new breath in the recent literature. The latest developments are reviewed in [5]. Let us just mention a few landmarks: the *Large Scale Neighborhood Search* [3], the *Relaxation Induced Neighborhood Search* [8], the *local branching* heuristic [12], the *feasibility pump algorithm* [1,11]. Meta-heuristic paradigms such as oscillation between intensification and diversification of the search, and use of historical memory have also inspired progress in primal heuristics. Recently such work has been extended from the context of binary integer programs to general integer programs. Our purpose is to examine possible extensions to the case where one works with a Dantzig-Wolfe reformulation of the problem involving an exponential number of columns. The above mentioned landmark heuristics have not been applied directly to the Dantzig-Wolfe reformulation because setting bounds on column values can hardly been implemented in a context of dynamic column generation (the pricing problem ignores such bounds; modifying it to enforce such bounds typically induces an harder to solve subproblem). Alternatively, one could potentially develop an implementation of classic primal heuristics based on projecting the master solution in the original variable space. But, in some applications, there might not exist a bijective relation between compact formulation and Dantzig-Wolfe reformulation. Here, we consider how one can develop constructive or improvement heuristics specifically for the column generation context.

2 An overview of column generation based heuristics

Assume a mixed integer program (IP):

$$[P] \quad \min\{cx : Ax \geq a, \underbrace{Bx \geq b, x \in \mathbb{R}^n_+ \times \mathbb{Z}^p_+}_{x \in X}\}$$

where a subset of constraints $Bx \geq b$ defines a subsystem X over which optimization is "relatively easy" while $Ax \geq a$ represent "complicating constraints". For many structured applications, $Bx \geq b$ has a *block diagonal* structure with identical blocks. The structure of $[P]$ can exploited by reformulating it as a master program:

$$[M] \quad \min\{\sum_{g \in G}(cx^g)\lambda_g : \sum_{g \in G}(Ax^g)\lambda_g \geq a;\ \sum_{g \in G}\lambda_g = K \quad \forall k;\ \lambda_g \in \mathbb{N} \quad \forall g\}$$

where G is the set of *generators* of X and K is the number of identical blocks in $Bx \geq b$. We assume a bounded subsystem, thus G is an enumeration of the feasible solutions to X, i.e. $X = \{x^g\}_{g \in G}$. $|G|$ is typically exponential in the input size. Reformulation $[M]$ is solved by branch-and-price: at each node of the branch-and-bound tree the linear relaxation of $[M]$ is solved by column generation with a pricing problem of the form: $\min_{x \in X}\{(c - \pi A)\, x\}$, where π is the dual solution associated with constraints $Ax \geq a$ in $[M]$.

The most commonly used generic primal heuristic in this column generation context is the so-called *restricted master heuristic*. The column generation formulation is restricted to a subset of generators $\overline{G}$ and associated variables, and it is solved as a static IP. The restricted set $\overline{G}$ can either be generated heuristically, or be made of the columns generated during the master LP solution, or a mixture of both. The main drawback of this approach is that the resulting restricted master integer problem is often infeasible (the columns of $\overline{G}$ – typically defined by the LP solution – may not be combined to an integer solution). In an application specific context, an ad-hoc recourse can be designed to "repair" infeasibility. Such implementation has been developed for network design [6], vehicle routing [2,7,20] problems.

Simple *greedy heuristic* strategies have also been used: one iteratively adds a greedy selected column (the one with the best so-called "pseudo-cost") to the partial solution until feasibility is reached. Column selection criteria can make use of the pricing procedure: for instance selecting columns that have the smallest ratio of reduced cost per unit of constraint satisfaction (based on dual price estimates that can be re-evaluated in the course of the algorithm). One can also use the master LP solution as a base for column selection. This gives rise to so-called *rounding heuristics*. A standard rounding strategy for

the common set covering type master formulation consists of 3 steps: (*i*) an initial partial solution is obtained by rounding downwards the master LP solution, (*ii*) columns whose LP values are closest to their rounded part are then considered for round up while feasible, (*iii*) an ad-hoc constructive procedure is used to complete the solution. *Local search* can be used to improve the solutions while implementing some form of diversification. Typically one will remove some of the columns selected in the primal solution and reconstruct a solution with one of the above techniques. Greedy and rounding heuristics (sometimes coupled with local search) have been successfully applied (f.i in [4,19]). However, reaching feasibility remains a difficult issue that is often handled in an application specific manner along with greedy selection of columns, rounding directions or neighborhood definition. Deriving general purpose approaches is difficult. A generic way to complete a partial solution in search for feasibility is to generate further columns through pricing. This is precisely the strength of *diving heuristics*.

3 Diving heuristics

A *diving heuristic* is a depth-first heuristic search in the branch-and-price tree. At each node, a branch is heuristically selected based on greedy or rounding strategies. After branching the master LP is re-optimized (exactly or approximately). The branching rule used in such context is typically quite different from the one used in an exact approach: in a diving heuristic, one is not concerned with balancing the tree and one can overcome issues of variable fixing that are not compatible with column generation. Generating new columns in the process of re-optimization is an important feature for the success of the approach as it allow to construct feasible solutions. Observe that fixing a partial solution does not impair column generation: the residual master can be tackled in the same way as the original master LP as no specific bounds on master variables have been set. In particular, columns previously selected in the partial solution remain in the formulation and may be selected again at a further stage.

Diving heuristics admit many possible implementation variants as illustrated in previous application specific studies: they were used on vehicle routing [15,18], crew rostering [14,13], cutting stock [9,19], and lot-sizing [9] problems. A key feature is the column selection mode that drives the heuristic: using greedy, random or rounding strategies (rounding down, up, to the closest integer, or based on a threshold [9,13,15,18]). Note that several columns can be taken into the solution simultaneously, and a constructive heuristic can be

applied in an attempt to complete the solution at any stage [4,17]. A template of such diving procedure is given in Table 1.

Table 1
Depth first search diving heuristic

Algorithm 1

Step 1: *Solve the current master LP.*

Step 2: *Select columns into the current solution at heuristically set values.*

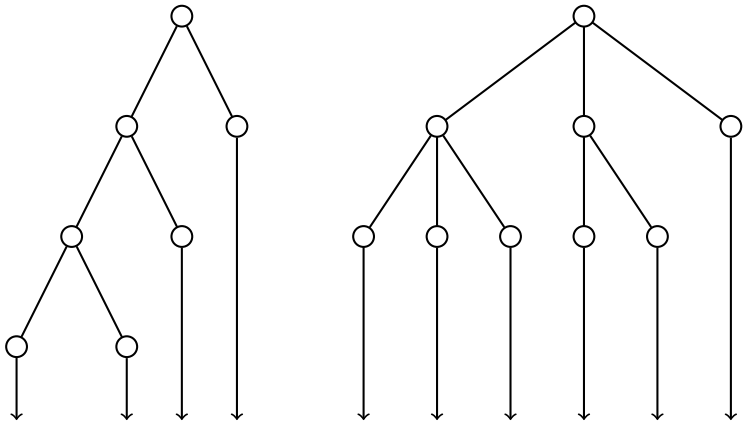
Step 3: *If the partial solution defines a complete primal solution, record this solution and goto* Step 6.

Step 4: *Update the master (right hand side) and the pricing problems (setting new bounds on subproblem variables to generate proper columns).*

Step 5: *If residual master problem is shown infeasible through preprocessing, goto* Step 6. *Else, return to* Step 1.

Step 6: *Stop.*

In our implementation, columns are selected one at the time in Step 2, selecting in the current master LP solution the λ_g variable value closest to integer (among roundings that are feasible for the master program). The selected column g is taken in the solution at a value equal to the closest integer to λ_g. An original feature not found in the above mentioned references is a limited backtracking that implements a diversification mechanism. The solution obtained through the initial depth-first exploration of the tree is considered as a reference solution around which we implement a *Limited Discrepancy Search (LDS)* [16]. The latter is controlled by two parameters: **maxDepth** and **maxDiscrepancy**: up to depth **maxDepth**, we considered to deviate from the reference solution in up **maxDiscrepancy** ways. Specifically, in Step 2, we avoid selecting columns present in a tabu list (of size $\leq$ **maxDiscrepancy**) that consists of columns selected in previous branches from which we wish to diversify the search. In step 6, we backtrack, while the current depth is greater than **maxDepth** or the current Discrepancy level = **maxDiscrepancy**. If such backtracking is not possible, we stop. Otherwise, we create a new branch defined by a tabu list made of columns that were tabu at the ancestor node or were selected in previous child nodes of the ancestor node. The resulting exploration tree is illustrated in Figure 1 for two parameter settings.



(a) **maxDepth** = 3, **maxDiscrepancy** = 1

(b) **maxDepth** = 2, **maxDiscrepancy** = 2

Fig. 1. Illustration of a tree search with bounded depth and discrepancy

4 Computational results

We have build our diving heuristic into *BaPCod* [21], a generic Branch-and-Price Code that we developed. To test its usefulness, we compare the exact branch-and-price algorithm with and without the use of the primal heuristic at the root node, and the use of the diving heuristic only. Table 2 presents computational results on standard instances for the cutting stock problem (CSP) [22], bin packing problem with conflicts (BPWC) [10] (considering "hard instances" with a conflict graph of density of 10 to 40%), the graph vertex coloring problem (VCP) and classical vehicle routing problem (VRP) using the class A instances of `http://neo.lcc.uma.es/radi-aeb/webvrp/`. For **maxDepth** = 3 and **maxDiscrepancy** = 2, Table 2 reports the problem class, the instances size, the number of instances tested (#inst.), the number of unsolved instances within 30 minutes (#unsolv.), the average size of the branch-and-price tree (#nodes, using the branching scheme of [23]), the average solution time (in seconds). For the pure heuristic (DH only), we report the number of instances for which the solution found is not optimal (#unsolv.) and the average gap of the solution found with the optimum. Even when the DH solution has no gap with the optimum, it can have a gap with the column generation dual bound and hence no prove of optimality is known, which induces branching.

Table 2
Comparing branch-and-price with and without our diving heuristic (DH).

			pure B-a-P			DH + B-a-P			DH only		
problem	size	#inst.	#unsolv.	#nodes	time	#unsolv.	#nodes	time	#unsolv.	gap	time
CSP	50	10	0	245	26				0	0%	2
CSP	80	10	0	417	192				0	0%	14
BPWC	60–500	280	9	148	167	0	2.9	102	2	0.02%	75
VCP	11–211	18	0	17	116	0	1.2	34	0	0%	26
VRP	An32-80	27							17	5.25%	70

References

[1] Tobias Achterberg and Timo Berthold. Improving the feasibility pump. *Discrete Optim.*, 4(1):77–86, 2007.

[2] Y. Agarwal, K. Mathur, and H. M. Salkin. A set-partitioning-based exact algorithm for the vehicle routing problem. *Networks*, 19(7):731–749, 1989.

[3] Ravindra K. Ahuja, Özlem Ergun, James B. Orlin, and Abraham P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Appl. Math.*, 123(1-3):75–102, 2002. Workshop on Discrete Optimization, DO'99 (Piscataway, NJ).

[4] G. Belov and G. Scheithauer. A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths. *European J. Oper. Res.*, 141(2):274–294, 2002. Cutting and packing.

[5] Timo Berthold. Primal Heuristics for Mixed Integer Programs. Master's thesis, Technische Universität Berlin, 2006.

[6] A. Chabrier. Heuristic branch-and-price-and-cut to solve a network design problem. In *Proceedings CPAIOR*, Montreal, Canada, may 2003.

[7] A. Chabrier, E. Danna, and C. Le Pape. Coopération entre génération de colonnes et recherche locale appliquées au problème de routage de véhicules. In *Huitièmes Journées Nationales sur la résolution de Problèmes NP-Complets (JNPC)*, pages 83–97, Nice, France, may 2002.

[8] Emilie Danna, Edward Rothberg, and Claude Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Math. Program.*, 102(1, Ser. A):71–90, 2005.

[9] Zeger Degraeve and Raf Jans. A new Dantzig-Wolfe reformulation and branch-and-price algorithm for the capacitated lot-sizing problem with setup times. *Oper. Res.*, 55(5):909–920, 2007.

[10] Albert E. Fernandes Muritiba, Manuel Iori, Enrico Malaguti, and Paolo Toth. Algorithms for the bin packing problem with conflicts. *INFORMS Journal on Computing*, page ijoc.1090.0355, 2009.

[11] Matteo Fischetti, Fred Glover, and Andrea Lodi. The feasibility pump. *Math. Program.*, 104(1, Ser. A):91–104, 2005.

[12] Matteo Fischetti and Andrea Lodi. Local branching. *Math. Program.*, 98(1-3, Ser. B):23–47, 2003. Integer programming (Pittsburgh, PA, 2002).

[13] Michel Gamache, François Soumis, Gérald Marquis, and Jacques Desrosiers. A column generation approach for large-scale aircrew rostering problems. *Oper. Res.*, 47(2):247–263, 1999.

[14] Martin Grötschel, Ralf Borndörfer, and Andreas Löbel. Duty scheduling in public transit. Jäger, Willi (ed.) et al., Mathematics—key technology for the future. Joint projects between universities and industry. Berlin: Springer. 653-674 (2003)., 2003.

[15] Oktay Gunluk, Tracy Kimbrel, Laszlo Ladanyi, Baruch Schieber, and Gregory B. Sorkin. Vehicle Routing and Staffing for Sedan Service. *Transportation Science*, 40(3):313–326, 2006.

[16] William D. Harvey and Matthew L. Ginsberg. Limited discrepancy search. In *Proc. IJCAI-95, Montreal, Quebec*, pages 607–613. Morgan Kaufmann, 1995.

[17] Krzysztof C. Kiwiel. An Inexact Bundle Approach to Cutting-Stock Problems. *INFORMS Journal on Computing*, page ijoc.1090.0326, 2009.

[18] Sophie Michel. *Optimisation des tournées de véhicules combinées à la gestion de stock*. PhD thesis, Université Bordeaux 1, France, 2006.

[19] Nancy Perrot. *Integer Programming Column Generation Strategies for the Cutting Stock Problem and its Variants*. PhD thesis, Université Bordeaux 1, France, 2005.

[20] É. D. Taillard. A heuristic column generation method for the heterogeneous fleet VRP. *RO Oper. Res.*, 33(1):1–14, 1999.

[21] F. Vanderbeck. Bapcod - a generic branch-and-price code, 2008. http://wiki.bordeaux.inria.fr/realopt/.

[22] François Vanderbeck. Computational study of a column generation algorithm for bin packing and cutting stock problems. *Math. Program.*, 86(3, Ser. A):565–594, 1999.

[23] François Vanderbeck. Branching in branch-and-price: a generic scheme. *Mathematical Programming*, 2010. doi 10.1007/s10107-009-0334-1.