

Partial Convexification of General MIPs by Dantzig-Wolfe Reformulation

Martin Bergner^{1,*}, Alberto Caprara², Fabio Furini¹,
Marco E. Lübbecke¹, Enrico Malaguti², and Emiliano Traversi²

¹ Chair of Operations Research, RWTH Aachen University,
Templergraben 64, 52056 Aachen, Germany

{martin.bergner,fabio.furini,marco.luebbecke}@rwth-aachen.de

² DEIS, Università di Bologna,

Viale Risorgimento 2, 40136 Bologna, Italy

{alberto.caprara,enrico.malaguti,emiliano.traversi2}@unibo.it

Abstract. Dantzig-Wolfe decomposition is well-known to provide strong dual bounds for specially structured mixed integer programs (MIPs) in practice. However, the method is not implemented in any state-of-the-art MIP solver: it needs tailoring to the particular problem; the decomposition must be determined from the typical bordered block-diagonal matrix structure; the resulting column generation subproblems must be solved efficiently; etc. We provide a computational proof-of-concept that the process can be automated in principle, and that strong dual bounds can be obtained on general MIPs for which a solution by a decomposition has not been the first choice. We perform an extensive computational study on the 0-1 dynamic knapsack problem (without block-diagonal structure) and on general MIPLIB2003 instances. Our results support that Dantzig-Wolfe reformulation may hold more promise as a general-purpose tool than previously acknowledged by the research community.

1 Introduction

A considerable, if not the major, part of the computational (mixed) integer programming machinery is about outer approximating the convex hull of integer feasible points (or mixed integer sets). The addition of valid inequalities, a.k.a. cutting planes, is the traditional general-purpose device which proved powerful in strengthening the linear programming relaxations. Given that the integer hull is the ultimate object of desire, we ask: Why don't we just work with it? Being fully aware of the boldness of this question, we want to seriously re-consider it by *explicitly* constructing parts of the integer hull via a generic Dantzig-Wolfe type reformulation (decomposition). This extends previous *partial convexification* approaches which only separate a subset of facets from the integer hull.

Dantzig-Wolfe reformulation (DWR) of mixed integer programs (MIPs) became a computationally very successful—sometimes the only applicable—approach to

* Supported by the German Research Foundation (DFG) as part of the Priority Program “Algorithm Engineering” under grant no. LU770/4-1.

producing high-quality solutions for well-structured combinatorial optimization problems like vehicle routing, cutting stock, p -median, generalized assignment, and many others. Their common structure is the (bordered) block-diagonal form of the coefficient matrix, the traditional realm of Dantzig-Wolfe decomposition. Be aware that so far its use is tailored to the application and far from being a general-purpose tool: It is the *user* who does not only know *that* there is an exploitable structure present but also *what* it looks like, and *how* to exploit it algorithmically. In particular in view of the automatism with which general-purpose cutting planes are separated in all serious MIP solvers, this is an unsatisfactory situation. This raises several research questions of increasing ambition:

- When the MIP contains a *known structure* suitable to Dantzig-Wolfe reformulation, can *an algorithm* detect and exploit it?
- When the contained structure is *unknown* (to be stated more precisely later), can it still be detected and exploited?
- When it is known that the MIP *does not contain* a structure amenable to DWR in the traditional sense, can DWR still be a useful computational tool?

Re-arranging matrices into particular forms is a well-known topic. However, as we will see, when there are several choices, a “good” one may not be obvious to find at all. Besides our work, we are not aware of any attempts to systematically answer the first two questions in a DWR context, and it can be taken as a fact that the research community is very inclined to answer the last, and most interesting question in the negative. In our computational study on several of the hardest MIPLIB2003 instances, we do not only suggest first attempts to accomplish the structure detection in a DWR context. We also support the DWR’s potential of becoming a general-purpose method for improving dual bounds by giving surprisingly encouraging computational results. Of course, at the moment, our work is not intended to produce a competitive tool, but to provide a proof-of-concept and demonstrate that the direction is promising. The main findings of our work can be summarized as follows:

- A known or hidden double-bordered block-diagonal (so-called: *arrowhead*) matrix structure can be effectively recovered and prepared for use in DWR by a suitable use of (hyper-)graph partitioning algorithms.
- For the dynamic 0-1 knapsack problem, the natural formulation of which does not expose any bordered block-diagonal structure, we give impressive computational results which demonstrate the potential of our method for closing the integrality gap.
- For some of the hardest MIPLIB2003 instances our reformulations produce stronger dual bounds than CPLEX 12.2 with default cutting planes enabled.
- We provide a general-purpose implementation which reads an LP file and performs the detection, the reformulation, and the column generation itself in order to obtain a strong LP relaxation, with only mild user interaction.

The flow of the paper is as follows: We briefly introduce the concept of partial convexification, and the overall approach. This is then applied first to a problem

not containing a matrix structure directly amenable to classical Dantzig-Wolfe reformulation, and then to general MIPs. We report on extensive computational experiments and close with a discussion of our results.

1.1 Partial Convexification and Dantzig-Wolfe Reformulations

Consider a MIP of the form

$$\max\{c^t x : Ax \leq b, Dx \leq e, x \in \mathbb{Z}^{n-q} \times \mathbb{Q}^q\} . \quad (1)$$

Let $P := \{x \in \mathbb{Q}^n : Dx \leq e\}$ and $P_{IP} := \text{conv}\{P \cap \mathbb{Z}^{n-q} \times \mathbb{Q}^q\}$ denote the LP relaxation and the integer hull with respect to constraints $Dx \leq e$, respectively. We assume for ease of presentation that P is bounded. In the classical *Dantzig-Wolfe reformulation* of constraints $Dx \leq e$ we express $x \in P_{IP}$ as a convex combination of the vertices V of P_{IP} , which leads to

$$\max\{c^t x : Ax \leq b, x = \sum_{v \in V} \lambda_v v, \sum_{v \in V} \lambda_v = 1, \lambda \geq 0, x \in \mathbb{Z}^{n-q} \times \mathbb{Q}^q\} . \quad (2)$$

It is well-known that the resulting LP relaxation is potentially stronger than that of (1) when $P_{IP} \subsetneq P$, which is a main motivation of performing the reformulation in the first place. This *partial convexification* with respect to the constraints $Dx \leq e$ corresponds to adding (implicitly) *all* valid inequalities for P_{IP} to (1), which in a sense is the best one can hope for.

The reformulated MIP (2) has fewer constraints remaining, the so-called *master constraints* $Ax \leq b$, plus the convexity constraint and the constraints linking the *original* x variables to the *extended* λ variables. On the downside of it, in general MIP (2) has an exponential number of λ variables, so its LP relaxation is solved by column generation, where the pricing or *slave MIP* problem to check whether there are variables with positive reduced cost to be added to the current *master LP* problem calls for the optimization of a linear objective function over P_{IP} . This slave MIP can either be solved by a general-purpose solver or by a tailored algorithm to exploit a specific structure, if known.

In the classical DWR setting, k disjoint sets of constraints are partially convexified, namely when the matrix D has block-diagonal form

$$D = \begin{bmatrix} D^1 & & & \\ & D^2 & & \\ & & \ddots & \\ & & & D^k \end{bmatrix},$$

where $D^i \in \mathbb{Q}^{m_i \times n_i}$ for $i = 1, \dots, k$. In other words, $Dx \leq e$ decomposes into $D^i x^i \leq e^i$ ($i = 1, \dots, k$), where $x = (x^1, x^2, \dots, x^k)$, with x^i being an n_i -vector for $i = 1, \dots, k$. Every $D^i x^i \leq e^i$ individually is partially convexified in the above spirit. We call k the *number of blocks* of the reformulation. Often enough,

constraints are not separable by variable sets as above, and a double-bordered block-diagonal (or *arrowhead*) form is the most specific structure we can hope for, i.e. the constraint matrix of (1) looks like this

$$\begin{bmatrix} D^1 & & & & F^1 \\ & D^2 & & & F^2 \\ & & \ddots & & \vdots \\ & & & D^k & F^k \\ A^1 & A^2 & \dots & A^k & G \end{bmatrix}.$$

The constraints associated with the rows of A^1 are called the *coupling constraints* and the variables associated with the columns of F^1 are called the *linking variables*. One can obtain a form without linking variables (and with additional linking constraints) by replacing each linking variable by one copy for each nonzero entry of the associated column and adding constraints imposing that all these copies be equal (see e.g., [13]). Then we are back to the above traditional setting.

1.2 Related Literature

For a general background on Dantzig-Wolfe reformulation of MIPs we refer to the recent survey [15]. The notion of *partial convexification* has been used before. For instance, Sherali et al. [12] choose small subsets of integer variables (and usually only one constraint) to directly separate cutting planes from the partial convexification P_{IP} . Our approach goes far beyond that in terms of number of constraints and variables involved in the reformulation.

There are several implementations which perform a Dantzig-Wolfe reformulation of a general MIP, and handle the resulting column generation subproblems in a generic way, like BaPCod [14], DIP [11], G12 [10], and GCG [8]. In [8] it is shown that a generic reformulation algorithm performs well when a block-diagonal matrix structure is *known and given* to the algorithm. Tebboth, in his thesis [13], derived some decomposable matrix structures from the problem given in a specific modeling language. A similar approach is taken in the G12 project. However, we do not know of a code which automatically detects a possible structure just from the matrix, that is well-suited and helpful for a Dantzig-Wolfe reformulation (or creates one by variable splitting), let alone evaluates or exploits it.

Bordered block-diagonal matrices play an important role in, e.g., numerical linear algebra. One typically tries to identify a fixed number of blocks of almost equal size with as few constraints in the border as possible. The motivation is to prepare a matrix for parallel computation, like for solving linear equation systems, see, e.g., [2], and the references therein. We are not aware of any attempts to evaluate the quality of the re-arranged matrices in terms of suitability for DWR.

We would also like to note the following possible connection to multi-row cuts which recently received some attention. Computational experiments [6] suggest that obtaining valid inequalities from more than one row of the simplex tableau holds some potential. However, in order to use this potential it seems to be

imperative to have a criterion for which rows to select. As our choice of which rows to convexify is equally important for similar reasons, the lessons we learn may help there as well.

2 Almost Automatic Detection of an Arrowhead Form

As mentioned, permuting a matrix A into arrowhead form is a common topic in numerical linear algebra. There is a folklore connection between a matrix and an associated (hyper-)graph which is also used in the graph partitioning approach by Ferris and Horn [7]. We first briefly illustrate their algorithm and then adapt it to our purpose of finding tighter LP relaxations of MIPs through DWR.

Assume the number k of blocks is given. Consider the bipartite graph G with one node r_i associated with each row i of A , one node c_j associated with each column j of A , and one edge (r_i, c_j) whenever $a_{ij} \neq 0$. Assuming the number $m+n$ of nodes is a multiple of k , find a *min k -equicut* of G , i.e. partition its node set into k subsets of equal size so that the number of edges in the cut, i.e. that join nodes in different subsets, is minimized. Finally, remove a set of nodes of G so as to ensure, for the remaining graph, that no edge joins nodes in different subsets of the partition. This yields the final arrowhead form: the row nodes removed represent the coupling constraints, the column nodes removed represent the linking variables, and the remaining row and column nodes in the i th subset of the partition define the submatrix D^i . In practice, min k -equicut is solved heuristically by using `Metis` which is an implementation of the multilevel graph partitioning algorithm in [9]. The final removal is done greedily by iteratively removing the node joined to nodes in other subsets by the largest number of edges. Moreover, to eliminate the need to require that the number of nodes be a multiple of k or to insist in having subsets of the same size, dummy nodes disconnected from the rest of the graph are added before doing the partitioning. This imposes only an upper bound on the size of each subset and on the number of subsets (note that subsets with dummy nodes only are irrelevant in the end).

We use a variant of this method because we would like to avoid the final removal phase. Specifically, we define the hypergraph H with a *node* for each nonzero entry of A . There is a *constraint hyperedge* joining all nodes for nonzero entries in the i th row of A . Moreover, there is a *variable hyperedge* joining all nodes for nonzero entries in the j th column of A . To each hyperedge we associate a cost to penalize the use of coupling constraints and linking variables.

Then, we find a heuristic min k -equicut of H , now with the objective of minimizing the sum of the costs of the hyperedges in the cut, i.e. that join nodes in different subsets of the partition. The solution already defines the arrowhead form, as the constraint hyperedges in the cut represent the coupling constraints, the variable hyperedges in the cut represent the linking variables, and the remaining constraint and variable hyperedges joining only nodes in the i th subset of the partition define the submatrix D^i . Also in this case we use dummy nodes for allowing for unequal block sizes, 20% proved useful.

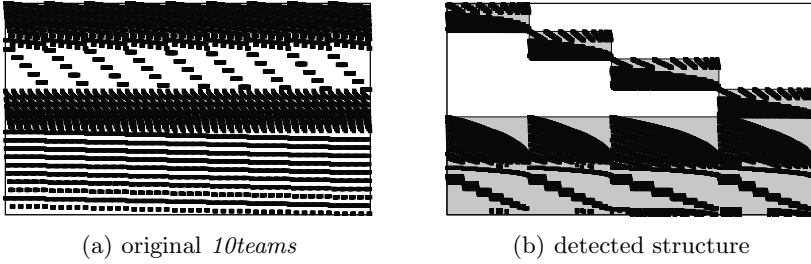


Fig. 1. (a) Matrix structure directly from the LP file (*10teams*) and (b) with a bordered block-diagonal structure detected by our algorithm

3 Computational Results

3.1 Notes on the Implementation and Experimental Setup

All experiments were done on Intel i7 quad-core PCs (2.67MHz, 8GB memory) running Linux 2.6.34 (single thread). As MIP and LP solver we used CPLEX 12.2 (single thread). Two independent jobs were run in parallel which has an impact on the CPU time of about 3% in our experiments. The min k -equicut problems to determine a reformulation (see Sect. 2) are heuristically solved using Hmetis [9].

Concerning the implementation, the overall algorithm detects an arrowhead form in the matrix, splits linking variables in order to obtain a bordered block-diagonal form, and then performs a Dantzig-Wolfe reformulation of the resulting blocks. Certain parameters must be given to the algorithm like the number k of blocks or the weights of the hyperedges in the k -equicut problem. We experimented with very few different settings and selected good decompositions instance dependent to demonstrate the concept. In the full version of this paper we plan to show that the whole process can be automated without any user interaction at all.

3.2 A Class of Structured MIPs without Block-Diagonal Form

We first consider a class of problems which has a natural MIP formulation with a coefficient matrix which is very structured, but not bordered block-diagonal at all. A *dynamic MIP* extends a MIP

$$\max\{c^t x : Ax \leq b, x \in \mathbb{Z}^{n-q} \times \mathbb{Q}^q\}$$

by a time horizon $[0, T]$ and a time interval $[s_j, f_j] \subseteq [0, T]$ in which each variable x_j is *active*. In the dynamic MIP all constraints must be satisfied at any instant in the time horizon restricting attention to the variables that are active at that instant (with the other variables fixed to 0). It is elementary to observe that attention can be restricted to the instants $I := \{s_1, \dots, s_n, f_1, \dots, f_n\}$ in which variables become active or inactive, yielding the following MIP formulation of the dynamic MIP:

$$\max\{c^t x : A^i x \leq b \ (i \in I), x \in \mathbb{Z}^{n-q} \times \mathbb{Q}^q\}, \quad (3)$$

where A^i is obtained from A by setting to 0 all the entries of columns associated with variables not active at $i \in I$.

A simple example of a dynamic MIP is the *dynamic 0-1 knapsack* problem, in which $q = 0$ and $Ax \leq b$ takes the form $a^t x \leq b$, $0 \leq x \leq 1$ for some $a \in \mathbb{Q}_+^n$ and $b \in \mathbb{Q}_+$. In words, we have a set of items each with a size a_j and active for time interval $[s_j, f_j]$ and a container of capacity b and we must select a set of items to pack in the container (for the whole interval in which they are active) so that the total size packed in each instant is at most b . This problem is also known as *resource allocation* or *unsplittable flow on a line*. The latter name is due to the fact that the time dimension may in fact be a (one-dimensional) spatial dimension, as it is the case in the following real-world application that we encountered in railway service design [5]. We are given a railway corridor with m stations $1, \dots, m$ and n railcars, the j th having weight a_j and to be transported from station s_j to station f_j , gaining profit c_j in this case. Moreover, we have a train that travels from station 1 to station m and can carry at the same time railcars for a total weight b . The problem is to decide which railcars the train carries in order to maximize the profit, and is clearly a dynamic 0-1 knapsack problem.

The dynamic 0-1 knapsack problem was recently proven to be strongly NP-hard and approximable within a constant [3] but a polynomial-time approximation scheme is not known.

In Table 1 we report the results obtained on a hard representative instance in each of the 0-1 dynamic knapsack classes considered in [4] when the parameter k varies. This parameter is chosen in such a way to enforce a decomposition in blocks of equal size of 128, 64, and 32 constraints each (this proved to provide best bounds as compared to computation times; for a larger number of much smaller blocks, computation times went up considerably). As one can see, on average, the DWR approach is able to close almost all of the root node's integrality gap. However, it takes an order of magnitude longer to obtain these results (see below). Nonetheless, the excellent quality of the dual bound helps in faster obtaining an integer optimal solution, as we see next.

Even though we do not wish to advocate for a principal “battle” of branch-and-price “versus” branch-and-cut (because branch-price-and-cut is probably the most versatile computational tool anyway), the direct comparison is instructive. To this end, we developed a very basic depth-first branch-and-price algorithm. Each node's linear programming relaxation is solved via DWR. We branch on the most fractional *original* variable and compare this setting to CPLEX 12.2 default branch-and-cut. Table 2 lists the results. We report the time required by branch-and-bound and, in case this exceeds the time limit of one hour, the relative gap between the best dual bound and the best primal solution found. The table shows the clear advantage of using the DWR. All instances can be solved to optimality within minutes, whereas CPLEX reaches the one-hour time limit for all instances except I65 (where profits equal volume). The results on the other instances in the six classes are analogous.

Table 1. Quality of the dual bound obtained by our DWR approach on the dynamic 0-1 knapsack problem, in comparison to a state-of-the-art IP solver. Listed are the instance name, the number of constraints and variables, the number k of blocks, the number ℓ of linking variables, and number c of coupling constraints. Under the heading *LP* one finds the relative integrality gap of the LP relaxation (in percent). The relative integrality gaps of DWR and CPLEX with default cuts applied are listed under *DWR* and *CPLEX+cuts*, respectively. The percentage of the LP gap closed is given under *%closed* for both approaches. The last row lists arithmetic means of the columns.

<i>instance</i>	rows	cols	k	ℓ	c	LP	DWR		CPLEX+cuts	
						gap	gap	%closed	gap	%closed
<i>I45</i>	1280	3433	10	243	0	15.720	0.002	99.987	1.306	91.695
<i>I55</i>	1280	8266	10	189	0	13.944	0.000	100.000	0.638	95.421
<i>I65</i>	1280	3434	10	243	0	11.182	0.011	99.903	0.098	99.127
<i>I75</i>	1280	4771	10	199	0	13.783	0.026	99.808	0.796	94.228
<i>I85</i>	1280	8656	10	159	0	13.057	0.001	99.994	0.410	96.860
<i>I95</i>	1280	5209	10	243	0	12.306	0.000	100.000	0.756	93.860
<i>I45</i>	1280	3433	20	505	0	15.720	0.007	99.956	1.306	91.695
<i>I55</i>	1280	8266	20	402	0	13.944	0.029	99.793	0.638	95.421
<i>I65</i>	1280	3434	20	509	0	11.182	0.009	99.915	0.098	99.127
<i>I75</i>	1280	4771	20	430	0	13.783	0.009	99.934	0.796	94.228
<i>I85</i>	1280	8656	20	344	0	13.057	0.001	99.994	0.410	96.860
<i>I95</i>	1280	5209	20	515	0	12.306	0.000	100.000	0.756	93.860
<i>I45</i>	1280	3433	40	977	0	15.720	0.047	99.703	1.306	91.695
<i>I55</i>	1280	8266	40	825	0	13.944	0.020	99.857	0.638	95.421
<i>I65</i>	1280	3434	40	976	0	11.182	0.000	100.000	0.098	99.127
<i>I75</i>	1280	4771	40	880	1	13.783	0.031	99.775	0.796	94.228
<i>I85</i>	1280	8656	40	756	0	13.057	0.025	99.807	0.410	96.860
<i>I95</i>	1280	5209	40	1041	0	12.306	0.004	99.970	0.756	93.860
means						13.332	0.012	99.911	0.667	95.199

3.3 MIPLIB2003

In order to assess the generality of the proposed method we tested the algorithm on MIPLIB2003 instances [1]. We selected a subset of instances, for which (a) the optimum is known, (b) the density is between 0.05% and 5%, (c) the number of non-zeros is not larger than 20,000, and (d) the percentage of discrete variables is at least 20%. We are consistently capable of improving over the LP relaxation in terms of the dual bound. Moreover, on average, our DWR approach closes a larger percentage of the integrality gap than CPLEX with default cuts applied, see Table 3. We do not report on computation times because the experience is the same as for the RAP (and not the focus here): Often, we need an order of magnitude more time (even though occasionally, we are competitive with CPLEX).

Table 2. Comparison of the dual bounds obtainable by branch-and-price as compared to branch-and-cut for the dynamic 0-1 knapsack problem, and the computation times needed. All instances are solved to optimality by branch-and-price. Listed are the instance name, the times to obtain the various dual bounds: The linear relaxation from DWR, the linear relaxation with CPLEX and default cuts applied, the optimal solution with branch-and-price (DWR), and the time needed with branch-and-cut (CPLEX). The time limit (TL) was set to one hour. The final columns list the remaining integrality gap of CPLEX at time limit.

instance	time			CPLEX B&B		
	DWR	CPLEX+cuts	DWR B&P	CPLEX B&B	gap	%closed
<i>I45</i>	33.75	2.72	111.06	TL	0.35	97.79
<i>I55</i>	179.73	4.53	2483.31	TL	0.16	98.87
<i>I65</i>	7.31	1.50	16.28	1.67	0.00	100.00
<i>I75</i>	30.02	2.63	97.84	TL	0.03	99.80
<i>I85</i>	55.62	4.53	105.59	1357.88	0.00	100.00
<i>I95</i>	29.21	3.01	29.22	TL	0.05	99.55
means	55.94	3.15	473.88	2626.59	0.10	99.34

Choosing a good decomposition. We experimented with a few parameter settings for our algorithm to detect an arrowhead structure. Depending on these settings, different arrowhead forms are produced (for the same input matrix), and these perform differently in the subsequent Dantzig-Wolfe reformulation. Parameters are (a) the number k of blocks, and the penalties for hyperedges which (b) split continuous and (c) discrete variables, as well as (d) penalties for hyperedges which couple constraints. We experimented with $k \in \{2, 3, 4, 5\}$, penalized (throughout all experiments) hyperedges corresponding to continuous and discrete with cost 1 and 2, respectively; and finally we used two different settings for penalizing coupling constraints: mildly (cost 5) and extremely (cost 10^5). Every combination was applied to each instance, and the best result in terms of dual bound is reported. In other words, in this paper, we give only a proof-of-concept *that* a good decomposition can be chosen. *How* to find a good decomposition, and even the meaning of “good,” are highly non-trivial issues. E.g., the “right” number k of blocks is far from obvious for instances that do not have a natural (bordered) block-diagonal form. We have preliminary computational experience that a “visually appealing” decomposition performs better than others. We give details on measures for this intuition, and on how to automate the detection and decomposition process in the full version.

Table 3 shows that in almost all cases the dual bound found by our DWR approach is much better than that of the continuous relaxation, and often even improves on CPLEX’s root node bounds with default cuts applied. The time required to compute our bound is not competitive with the time required by the general-purpose solver to solve the instance, but there remains the possibility that for some instances the significantly stronger dual bound helps in solving the instance to integer optimality.

Table 3. Comparison of the dual bounds provided by our automatic DWR reformulation approach and the general-purpose MIP solver CPLEX for 23 selected instances of MIPLIB2003. The headings have the same meanings as in Table 1.

<i>instance</i>	rows	cols	<i>k</i>	ℓ	<i>c</i>	LP	DWR		CPLEX+cuts	
						gap	gap	%closed	gap	%closed
10teams	2025	230	4	0	107	0.758	0.000	100.000	0.000	100.000
<i>aflow30a</i>	842	479	2	0	28	15.098	14.700	2.634	5.353	64.547
<i>aflow40b</i>	2728	1442	5	0	39	13.899	13.899	0.000	6.471	53.441
fiber	1298	363	2	2	21	61.550	1.067	98.266	1.894	96.923
<i>fixnet6</i>	878	478	4	3	14	69.850	18.882	72.967	6.064	91.318
gesa2-o	1224	1248	5	65	0	1.177	0.000	99.986	0.207	82.379
gesa2	1224	1392	3	65	0	1.177	0.000	99.986	0.100	91.507
glass4	322	396	3	16	0	33.334	26.713	19.862	33.334	0.000
<i>harp2</i>	2993	112	5	0	39	0.614	0.614	0.000	0.371	39.599
manna81	3321	6480	2	78	0	1.010	0.000	100.000	0.000	100.000
mkc	5325	3411	2	0	29	8.514	0.153	98.200	3.778	55.625
modglob	422	291	2	18	3	1.493	0.000	100.000	0.142	90.480
noswot	128	182	5	21	3	4.878	0.488	90.000	4.878	0.000
<i>opt1217</i>	769	64	4	0	16	25.134	25.134	0.000	0.000	100.000
p2756	2756	755	4	39	13	13.932	0.269	98.070	7.467	46.407
pp08a	240	136	2	16	0	62.608	2.172	96.530	2.525	95.967
pp08aCUTS	240	246	2	16	0	25.434	2.172	91.459	3.823	84.968
rout	556	291	5	0	16	8.881	0.681	92.335	8.858	0.262
<i>set1ch</i>	712	492	3	20	8	41.311	2.086	94.950	0.923	97.765
timtab1	397	171	2	13	0	96.248	14.473	84.963	39.050	59.428
timtab2	675	294	4	25	0	92.377	24.426	73.559	46.004	50.200
<i>tr12-30</i>	1080	750	3	24	0	89.119	2.713	96.955	0.682	99.235
vpm2	378	234	2	7	0	28.078	1.706	93.924	6.443	77.053
arithm. mean						30.281	6.624	74.115	7.755	68.570

4 Discussion

We have performed the first systematic computational study with an automatic partial convexification by a Dantzig-Wolfe type reformulation of subsets of rows of *arbitrary* mixed integer programs. While it is clear from theory that a partial convexification can improve the dual bound, it has not been considered a generally useful computational tool *in practice*. Thus, the most unexpected outcome of our study is that already a fairly basic implementation, combined with a careful choice of the decomposition, is actually capable of competing with or even beating a state-of-the-art MIP solver in terms of the root node dual bound. Interestingly, to the best of our knowledge for the first time, the “careful choice of the decomposition” is done almost entirely by an algorithm, only mildly helped by the user. A fully automated detection will be presented in the full paper.

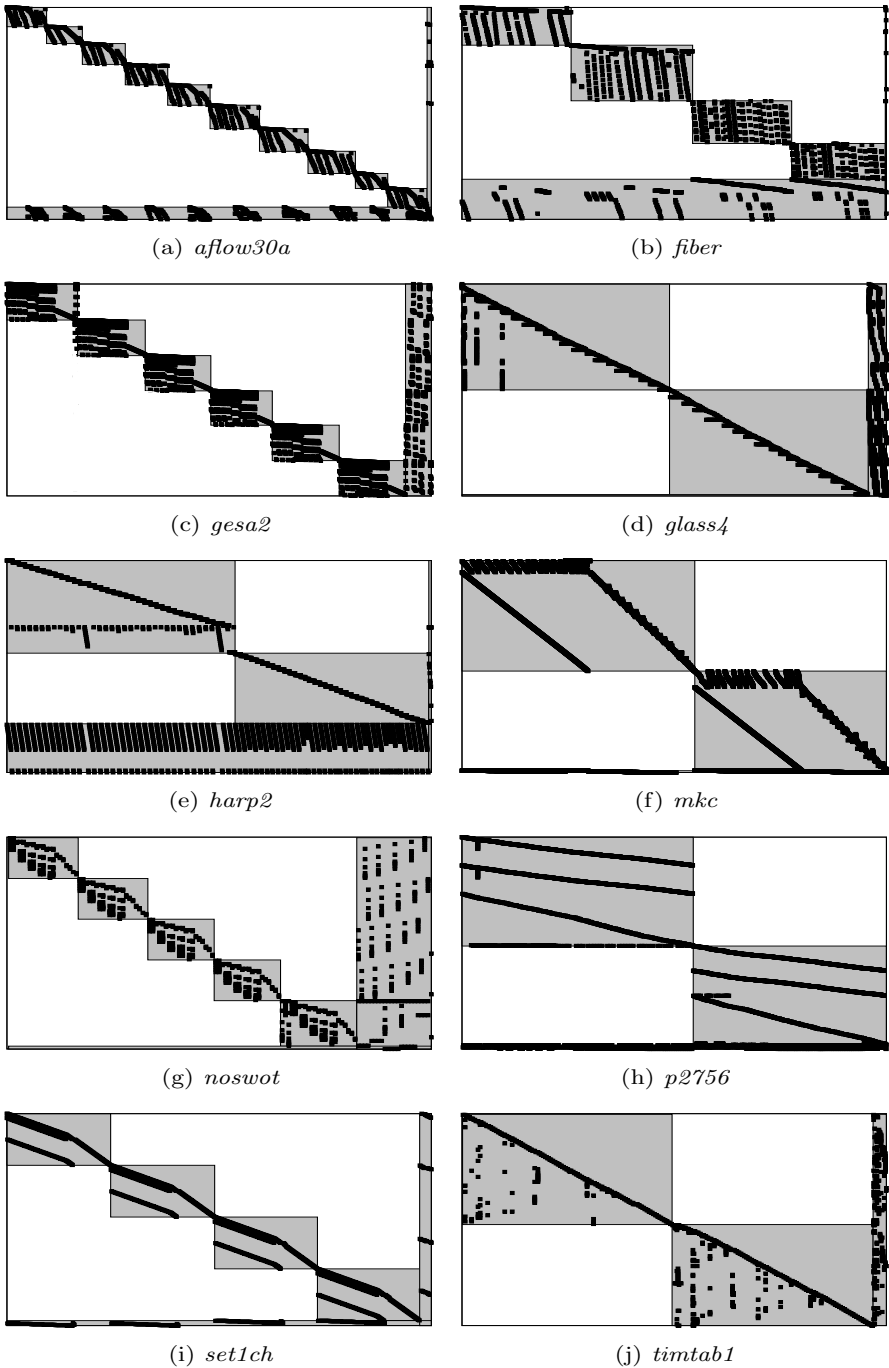


Fig. 2. Detected matrix structures for selected MIPLIB2003 instances

One should not deny that an inner approximation of the integer hull still has considerable disadvantages, as the process is not reversible: we cannot easily get rid of the extended formulation. Also, the choice of the decomposition is final (at present). This “single shot” contrasts cutting plane algorithms which can iteratively increase the number of classes of valid inequalities considered. Therefore, one must carefully decide whether to use a DWR approach or not. A remote goal would be to be able to make this important decision based on the instance only. If in, say, even only a small fraction of “all” MIPs a DWR approach pays, we would have already enriched the generic MIP toolbox.

There are some possible immediate extensions concerning the implementation. Even though we have mainly solved the LP relaxation so far, our tool is able to perform a true branch-and-price. Only further experimentation can show whether the advantage in the root node can be retained throughout the search tree, not only for dynamic MIPs but also for general MIPs (it is also conceivable that an advantage becomes visible *only* further down the tree). If one is only interested in a strong dual bound, the addition of generic cutting planes is a natural next step.

All the questions initially posed in the introduction are computationally and conceptually extremely hard, and at present one cannot hope for conclusive answers to any of them. We therefore think that our work spawns a number of interesting research directions worth pursuing further.

1. The most important task, both from a theoretical and a practical point of view, is to characterize a *good decomposition*. This can also help in quickly deciding whether it is worth trying a reformulation or not.
2. We have seen that the matrix needs not contain any (known) apparent structure in order to make the method perform well. In particular our third question from the introduction needs re-formulation in the light of our results: what does the fact that a model is suitable for application of DWR mean?
3. *Extended formulations* are a topic on its own in combinatorial optimization, mainly used as a theoretical vehicle to obtain stronger formulations. As DWR is a particular kind of extended formulation it is natural to ask: Can an approach like ours turn this into a computational tool?
4. We complained that, once chosen, a decomposition is static. Is there a computationally viable way for dynamically updating an extended formulation, like our DWR?

Taking into account that state-of-the-art solvers make successful use of cutting planes for over 15 years now, it is clear that outer approximations of the integer hull have a prominent headway in experience over inner approximations. We hope to have inspired further research and experimentation on the topic of this paper.

References

1. Achterberg, T., Koch, T., Martin, A.: MIPLIB 2003. *Oper. Res. Lett.* 34(4), 361–372 (2006)
2. Aykanat, C., Pinar, A., Çatalyürek, Ü.V.: Permuting sparse rectangular matrices into block-diagonal form. *SIAM J. Sci. Comput.* 25, 1860–1879 (2004)
3. Bonsma, P., Schulz, J., Wiese, A.: A constant factor approximation algorithm for unsplittable flow on paths. *CoRR*, abs/1102.3643 (2011)
4. Caprara, A., Furini, F., Malaguti, E.: Exact algorithms for the temporal knapsack problem. Technical report OR-10-7, DEIS, University of Bologna (2010)
5. Caprara, A., Malaguti, E., Toth, P.: A freight service design problem for a railway corridor. *Transportation Sci.* (2011) (in press)
6. Espinoza, D.G.: Computing with multi-row Gomory cuts. *Oper. Res. Lett.* 38, 115–120 (2010)
7. Ferris, M.C., Horn, J.D.: Partitioning mathematical programs for parallel solution. *Math. Program.* 80(1), 35–61 (1998)
8. Gamrath, G., Lübbecke, M.E.: Experiments with a generic dantzig-wolfe decomposition for integer programs. In: Festa, P. (ed.) SEA 2010. LNCS, vol. 6049, pp. 239–252. Springer, Heidelberg (2010)
9. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Comput.* 20(1), 359–392 (1998)
10. Puchinger, J., Stuckey, P.J., Wallace, M.G., Brand, S.: Dantzig-Wolfe decomposition and branch-and-price solving in G12. *Constraints* 16(1), 77–99 (2011)
11. Ralphs, T.K., Galati, M.V.: DIP – decomposition for integer programming (2009), <https://projects.coin-or.org/Dip>
12. Sherali, H.D., Lee, Y., Kim, Y.: Partial convexification cuts for 0-1 mixed-integer programs. *European J. Oper. Res.* 165(3), 625–648 (2005)
13. Tebbboth, J.R.: A Computational Study of Dantzig-Wolfe Decomposition. PhD thesis, University of Buckingham (2001)
14. Vanderbeck, F.: BaPCod – a generic branch-and-price code (2005), <https://wiki.bordeaux.inria.fr/realopt/pmwiki.php/Project/BaPCod>
15. Vanderbeck, F., Wolsey, L.: Reformulation and decomposition of integer programs. In: Jünger, M., Lieblich, T.M., Naddef, D., Nemhauser, G.L., Pulleyblank, W.R., Reinelt, G., Rinaldi, G., Wolsey, L.A. (eds.) 50 Years of Integer Programming 1958–2008. Springer, Berlin (2010)