

New Techniques To Improve Convergence of Decomposition Methods on Large Scale Optimization Problems

ALPÁR JÜTTNER^{*†}

Department of Operations Research
Eötvös Loránd University
Budapest, Hungary
alpar@cs.elte.hu

PÉTER MADARASI^{*}

Department of Operations Research
Eötvös Loránd University
Budapest, Hungary
madarasi@cs.elte.hu

Abstract: Dantzig-Wolfe decomposition applied to integer linear programs of special block-diagonal structure is a specific way of problem reformulation that might provide a tighter relaxation than the usual linear programming relaxation. The reformulation is an enormous integer master problem, whose variables are dealt with implicitly in a B&B method combined with a column generation procedure to compute the bounds. During the traversal of the B&B tree, branching happens on the variables of the original problem, which restricts the corresponding bounding problems on the descendent tree nodes. This paper presents refinements to the method above. On one hand, the proposed smoothed column generation procedure improves the convergence to the optimum of the LP relaxation of the decomposed problem. On the other hand, a new branching scheme moderates the so-called tail-off effect and allows a more effective execution of the usual B&B method. Numerical evaluations show that the developed method is significantly better than the traditional approach and it is competitive with the contemporary generic purpose commercial solvers.

Keywords: column generation, Dantzig-Wolfe decomposition, primal-dual methods, integer programming

1 Introduction

Column Generation and Branch-and-Price, introduced in the early results of Ford and Fulkerson [11] and Appelgren [1], are standard tools for tackling various optimization problems [2, 21, 19, 18]. It has successfully been applied to versions of traveling salesman, vehicle routing and crew scheduling problems [5], airline crew pairing [4], scheduling and fleet assignments, in telecommunication (network dimensioning, resource management and routing) and to staff scheduling problems [6], as well as to generic combinatorial optimization problems such as (integer) multicommodity flows [11], maximum stable-set [3] and graph coloring problems [15]. The aim of this work is to further widen the applicability of this approach by presenting a refinement of the aforementioned techniques that provides a faster convergence of the LP relaxation of the problem and allows a more effective execution of the usual Branch-and-Bound algorithm to find an integer optimum solution.

2 Dantzig-Wolfe decomposition

The application of Dantzig-Wolfe decomposition requires the problem described as a linear program of block diagonal structure with an additional set of constraint connecting the otherwise independent blocks.

^{*}The project has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002) and

[†]Also supported by the János Bolyai Research Fellowship program of the Hungarian Academy of Sciences.

Formally, for given matrices $A_1, \dots, A_k \in \mathbb{R}^{m \times n}$, and vectors $b_1, \dots, b_k \in \mathbb{R}^m$, $l, u \in \mathbb{R}^n$ and $c_1, \dots, c_k \in \mathbb{R}^n$ we look for

$$\min \sum_{i=1}^k c_i x_i \quad (1)$$

where

$$x_i \in \mathbb{R}^n \quad \forall i = 1, \dots, k \quad (2)$$

$$A_i x_i \leq b_i \quad \forall i = 1, \dots, k \quad (3)$$

$$\sum_{i=1}^k x_i \leq u \quad (4)$$

In addition we assume that each polyhedron $P_i := \{x \in \mathbb{R}^n : A_i x \leq b_i\}$ is bounded i.e. it is the convex combination of its set of vertices $V_i \subset \mathbb{R}^n$. Then, the linear program above is equivalent with the following (enormous) *Master Problem (MP)*.

$$\min \sum_{i=1}^k \sum_{v \in V_i} c_i v x_v^i \quad (5)$$

where

$$x_v^i \in \mathbb{R} \quad \forall i = 1, \dots, k, \forall v \in V_i \quad (6)$$

$$x_v^i \geq 0 \quad \forall i = 1, \dots, k, \forall v \in V_i \quad (7)$$

$$\sum_{v \in V_i} x_v^i = 1 \quad \forall i = 1, \dots, k \quad (8)$$

$$\sum_{i=1}^k \sum_{v \in V_i} v x_v^i \leq u \quad (9)$$

The corresponding dual linear program is

$$\max(\mathbf{1}y - uz) \quad (10)$$

where

$$y \in \mathbb{R}^k, z \in \mathbb{R}^n \quad (11)$$

$$z \geq 0 \quad (12)$$

$$y_i - vz \leq c_i v \quad \forall i = 1, \dots, k, \forall v \in V_i \quad (13)$$

This problem is often tractable using the so-called *Column Generation* approach. For this, we assume that the *Pricing Problems* $\min\{c'_i x : A_i x \leq b_i\}$ can efficiently be solved for arbitrary cost functions $c_i \in \mathbb{R}^n$. Then, we start with small initial subsets $V'_1, \dots, V'_k$ of $V_1, \dots, V_k$ and find an optimal solution to the corresponding *Restricted Master Problem (RMP)* along with its dual problem. The obtained primal solution is clearly a feasible solution to the original MP. Furthermore it is optimal if and only if the dual solution of the RMP is optimal that of the MP. It is easy to see that this latter condition is equivalent with that

$$\rho_i := \min\{(c_i + z)x : A_i x \leq b_i\} \geq y_i \quad (14)$$

holds for each $i = 1, \dots, k$, which can be verified by solving the Pricing Problems with the modified cost functions $(c_i + z)$, respectively. If the inequality (14) fails for some i , then the optimal solution x^i of the Pricing Problem is added to V'_i and the above steps are iterated until an optimal solution is found.

It is also important to observe that by solving the Pricing Problem, we also get the Lagrange lower bound, namely

Claim 1 Let $\rho_1, \dots, \rho_k$ be the optimum of the Pricing Problems above for any vector $z \in \mathbb{R}^n$, $z \geq 0$. Then the value $LB(z) := \sum_{i=1}^k \rho_i - zu$ is a lower bound of the optimum of the MP.

Furthermore,

Claim 2 The optimal solution to the RMP is also optimal to the MP if and only if the optimum of the RMP is equal to $LB(z)$ with respect to the corresponding dual solution z of the RMP.

3 Improving the Convergence of Column Generation

When applying Column Generation based solution in practice, one must often face the poor convergence of the simplex-based RMP, especially towards the end of the computation. A close-to-optimal solution may be found relatively fast, but then a long time is needed to find the real optimum (called the *tail-off effect*). Another related phenomenon is the heavy oscillation of the dual variables instead of a smooth convergence to the optimal values. This is widely considered as the main reason for the poor performance [13]. One of the first proposals for handling this issue is the BOXSTEP method proposed by Marsten et al. [14] and the *Stabilized Column Generation* proposed by du Merle et al. [8].

Using the observation that a Lagrangian lower bound can also be obtained at each iteration of the column generation, another approach for dealing with the oscillation is proposed by Wentges and Pessoa [20, 16]. This technique does not use the dual solution of the master problem directly as the price vector for column generation, but instead it combines it with the preceding dual solutions. The smoothing rule proposed in [20], and reconsidered in [16] suggests $\tilde{\pi}^t = \alpha\hat{\pi} + (1-\alpha)\pi^t$ using as pricing vector, where π^t denotes the dual vector providing the best lower bound so far, while $\hat{\pi}$ is the dual vector obtained by the current RMP and $\alpha \in [0, 1)$ is a dumping factor. We propose two efficient self-adjusting scheme adapting α to the phases of the algorithm.

3.1 Monotone Adaptive Smoothing Scheme

In each iteration of the column generation method, a dual vector $\hat{\pi}$ is obtained. Instead of using $\hat{\pi}$ as the pricing vector to solve the subproblems, let us use $\tilde{\pi}_0^t = \alpha_0\hat{\pi} + (1-\alpha_0)\pi^t$, where π^t denotes the dual vector providing the best lower bound so far and $\alpha_0 \in (0, 1]$ is a given initial value. As soon as no further columns can be generated with respect to price $\tilde{\pi}_i^t$, let us change our pricing vector to $\tilde{\pi}_{i+1}^t = \alpha_{i+1}\hat{\pi} + (1-\alpha_{i+1})\pi^t$, where $\alpha_{i+1} = \min\{\alpha_0 + i\delta, 1\}$ for a given step length $\delta \in (0, 1]$ and α_0 . Note that once α_i reaches 1, the actual dual values $\hat{\pi}$ will be used without smoothing. In the practical evaluation, we set $\alpha_0 = 0.2$ and $\delta = 0.2$, which seemed to provide an overall good performance.

An important aspect of this stabilization technique is that dual values used for solving the Pricing Problems differs from those of the current RMP. Therefore, it may happen that (a) there exists no solution of the Pricing Problems violating (14) (for the dual values in use) even when the primal solution to the RMP is not optimal. Thus the algorithm may terminate only when $\alpha_i = 1$.

3.2 Dynamic Adaptive Smoothing Scheme

Let $\hat{\pi}$ and π^t denote the dual vector and the dual vector providing the best lower bound throughout the column generation method. Just as above, let us use $\tilde{\pi}^t = \alpha\hat{\pi} + (1-\alpha)\pi^t$ as pricing vector instead of $\hat{\pi}$. In the beginning, $\alpha \in (0, 1]$ is set to a certain initial value. After each iteration of the column generation, α is reset as follows.

$$\alpha := \begin{cases} \min(\alpha\mu_1, 1) & \text{if } LB(\tilde{\pi}^t) > LB(\pi^t) \\ \alpha/\mu_0 & \text{if new column added} \\ \min(\alpha\mu_1, 1) & \text{otherwise} \end{cases} \quad (15)$$

where $\mu_0 \geq 1$ and $\mu_1 > 1$ are two give constant correction factors. (In Section 5, the values $\alpha = 0.5$, $\mu_0 = 1.2$ and $\mu_1 = 1.5$ were used for the numerical evaluations.) Again, the stop contition is that the current value of α is 1 and there is no more violating column is found by solving the Pricing Problems.

4 Early Termination at Processing B&B Nodes

Exploiting the fact that the primal-dual approach maintains both a lower and an upper bound converging to the optimal value, two ideas are proposed for *early termination* of the solutions of the RMS subproblem in [7].

Early Cut. Normally, a node of the B&B tree is pruned when either the LP subproblem belonging to the node is infeasible or its LP optimum is worse than the best integer solution found so far. However, the existence of a lower bound to the LP optimum at each iterations allows us to terminate the solution as soon as the lower bound reaches the cost of the best integer.

Early Branching. When solving the RMP, we iteratively generate an increasing subset of columns of the full problem and calculate the best LP solution obtainable using only those columns. In vast majority of the cases these solutions are fractional. Therefore as soon as the LP solution of the RMP goes below the best integer solution so far, we can conclude that branching will be inevitable. Thus we stop generating further columns and branch immediately.

The implementation of *Early Cut* is pretty straightforward. On the other hand, *Early Branching* needs somewhat more considerations, because branching as soon as possible may increase the size of the branching tree considerably. Thus it may be advantageous to continue generating columns even when branching would otherwise be possible. Two possible rules are (a) setting a minimum number of columns generated before branching or (b) allow branching only if the found solution and the computed lower bounds are close enough to each other. The latter option helps overcoming the *tail-off effect*.

5 Numerical Evaluation

5.1 The Adaptive Smoothing Schemes

The effect of stabilization schemes are tested on the LP relaxation of the *Maximum Weight Perfect d -distance Matching Problem* defined below. Not only has it been chosen because it fits nicely to the Dantzig-Wolfe decomposition framework, but also because it is an interesting undiscovered combinatorial optimization problem on its own.

5.1.1 Prefect d -Distance Matching

Imagine a go-kart game in an amusing park with k different cars, d of which is on the track while the remaining ones are in a pool to be chosen by the next person when one cars arrives back. n persons waits in a queue, each of them having a priory of cars they want to sit in. Namely, for each person s and cart t , we are given the amount w_{st} which the person s is willing to pay in order to go one round with car t . Thus our goal is to assign each people to a car in such a way that that car is available when to person is the next in the queue and we want to maximize the income.

Formally, this problem can be stated as follows.

Definition 3 Let us given a bipartite graph $G=(S,T;E)$ with $S = \{1, \dots, n\}$, $T = \{1, \dots, k\}$ and $d \in \mathbb{N}$ $d \leq k$. $M \subseteq E$ is called Perfect d -distance Patching, if $d_M(s) = 1$ for each $s \in S$ and $|\{(i, t) \in M : s \leq j < s + d\}| \leq 1$ holds for each $s \in S$ and $t \in T$.

For a given weight function $w : E \rightarrow \mathbb{R}$, the Minimum Weight Perfect d -distance Matching Problem looks for a perfect d -distance matching M minimizing the total weight $\sum_{st \in M} w_{st}$.

	Runtime (s)			Columns		
	DASS	MASS	NSS	DASS	MASS	NSS
$d = 10$ $ S = 200$ $ T = 25$	7.21	9.69	11.98	2143	3012	3694
	6.3	9.92	13.71	2126	3083	3935
	6.45	9.56	12.83	2176	2934	3716
	6.66	8.12	12.49	2114	2761	3702
	7.18	9.26	12.86	2178	2994	3693
$d = 5$ $ S = 200$ $ T = 30$	26.56	33.32	54.45	4065	5001	6314
	42.12	55.32	92.31	5446	6705	7282
	28.44	41.39	123.87	4450	5223	9457
	35.86	46.88	91.16	4779	5618	7711
	29.97	108.38	144.24	3931	7798	9024
$d = 5$ $ S = 200$ $ T = 20$	24.04	43.87	77.92	3261	4893	6472
	26.66	35.51	69.13	3807	4269	6206
	19.96	29.79	48.59	3117	4044	5252
	29.06	38.08	64.62	3598	4466	6022
	23.01	32.13	57.9	3188	4119	5692
$d = 5$ $ S = 200$ $ T = 10$	16.22	31.45	60.38	2294	3607	5334
	19.82	37.76	64.54	2714	3993	5504
	17.51	38.27	57.47	2274	3962	5002
	29.34	31.59	60.28	3884	3461	5058
	22.14	36.25	64.26	2891	3626	5092
$d = 3$ $ S = 200$ $ T = 3$	0.93	105.1	111.42	263	4907	5221
	1.24	136.01	126.01	296	5583	5329
	1.33	133.95	136.32	306	5425	5583
	1.17	114.22	121.29	288	5161	5383
	1.46	137.92	143.48	339	5917	6039
$d = 50$ $ S = 200$ $ T = 60$	1.38	1.77	2.16	1751	3794	4419
	1.44	2.03	2.37	1733	4049	4619
	1.37	2.09	2.21	1806	4168	4487
	1.31	1.91	2.26	1718	3951	4559
	1.24	1.75	2.73	1707	3469	5608
Average	14.58	44.11	61.51	2488.1	4399.77	5513.63

Figure 1: Practical evaluation of the proposed Smoothing Schemes. Columns *DASS*, *MASS* and *NSS* correspond to the *Dynamic Adaptive Smoothing Scheme* ($\alpha = 0.5$, $\mu_0 = 1.2$, $\mu_1 = 1.501$), *Monotone Adaptive Smoothing Scheme* ($\alpha_0 = 0.2$, $\delta = 0.2$) and to the case when no smoothing is used, respectively.

A straightforward integer programming formulation of the problem can be given as follows.

$$\min \sum_{st \in M} w_{st} x_{st} \quad (16)$$

where

$$x \in \mathbb{N}^{n \times k} \quad (17)$$

$$\sum_{t: st \in M} x_{st} = 1 \quad \forall s \in S \quad (18)$$

$$\sum_{it \in E: s \leq i < s+d} x_{it} \leq 1 \quad \forall s \in S, t \in T \quad (19)$$

By relaxing the integrality constraints (17) and Equations (18), we obtain the following Pricing

Problems for each $t \in T$.

$$\min \sum_{st \in M} w'_{st} x_{st} \quad (20)$$

$$\sum_{it \in M: s \leq i < s+d} x_{it} \leq 1 \quad \forall s \in S \quad (21)$$

The optimal solution to this problem can be found in time $O(n)$ by the following dynamic programming approach. Let f_s^t denote the minimal cost partial distance matching of $\{1, \dots, s\} \in S$ to $t \in T$ i.e.

$$f_s^t := \max \left\{ \sum_{i \in A, it \in M} w'_{it} : A \subseteq \{1, \dots, s\}, (i, j \in A, i \neq j \Rightarrow i - j \geq d) \right\}$$

These quantities can be calculated using the following recursion.

$$f_s^t := \begin{cases} 0 & \text{if } s \leq 0, \\ f_{s-1}^t & \text{if } s \geq 1 \text{ and } st \notin E, \\ \min(f_{s-1}^t, f_{s-d}^t + w'_{st}) & \text{if } s \geq 1 \text{ and } st \in E. \end{cases}$$

To test the proposed smoothing methods, random instances of the Perfect Distance Matching problem have been generated and solved with column generation using the proposed smoothing schemes. The first column of Table 5.1.1 contains the six considered parameter settings and each row corresponds to a random instance respecting the parameters on its left. Based on our computational experiments, the Dynamic Adaptive Smoothing Scheme significantly and consistently outperforms the Monotone Adaptive Smoothing Scheme, which is more efficient than the regular column generation method (when no smoothing is applied).

5.2 The Effect of Early Branch and Early Cut

In order to demonstrate the applicability of these techniques to speed up the execution of the Branch and Bound scheme, we tested them on the *Generalized Assignment Problem*.

In this problem, we are given n jobs to be assigned to m agents. Each agent i has capacity u_i , and when job j is assigned to agent i , it requires capacity d_{ij} and costs c_{ij} . The solution consists of matching each job to exactly one agent, so that the capacities of the agents are respected and the total assignment cost is minimized.

Let $K_i = \{x_1^i, x_2^i, \dots, x_{k_i}^i\}$ be the set of all feasible assignment of jobs to agent i , that is $x_k^i = (x_{k1}^i, x_{k2}^i, \dots, x_{kn}^i)$ satisfies

$$\sum_{1 \leq j \leq n} d_{ij} x_{kj}^i \leq u_i \quad (22)$$

$$x_{kj}^i \in \{0, 1\} \quad (j = 1, \dots, n). \quad (23)$$

Let $z_k^i \in \{0, 1\}$ ($i = 1, \dots, m$, $k \in K_i$) indicate whether assignment x_k^i is selected for agent i . Using these notations, the generalized assignment problem can be formulated as follows.

$$\min \sum_{1 \leq i \leq m} \sum_{1 \leq k \leq k_i} z_k^i \sum_{1 \leq j \leq n} c_{ij} x_{kj}^i \quad (24)$$

$$\sum_{1 \leq i \leq m} \sum_{1 \leq k \leq k_i} z_k^i x_{kj}^i = 1 \quad (j = 1, \dots, n) \quad (25)$$

$$\sum_{1 \leq k \leq k_i} z_k^i \leq 1 \quad (i = 1, \dots, m) \quad (26)$$

$$z_k^i \in \{0, 1\} \quad (i = 1, \dots, m, \quad k \in K_i), \quad (27)$$

Runtime(s)			Columns		Nodes		
Early	No Early	CPLEX	Early	No Early	Early	No Early	CPLEX
12.94	17.63	56.02	2713	2895	69	125	6014
142.99	280.69	164.37	2581	3259	876	1733	19111
34.56	39.51	92.91	2210	2515	255	333	9552
0.89	0.89	72.07	2352	2352	1	1	8732
529.72	865.98	709.06	3394	4417	3186	4259	83565
11.81	9.36	82.57	2601	2705	67	69	8467
29.23	46.58	35.26	2540	2802	173	331	3219
147.80	81.79	165.01	2714	3120	861	547	19315
45.01	47.77	101.76	2519	2771	287	347	8357
176.90	221.49	33.04	2704	3042	1048	1248	2971
0.52	0.52	36.04	2132	2132	1	1	3395
46.84	62.91	374.85	2822	3156	253	411	43262
1179.21	1675.12	1922.96	31282	35166	7077	9405	215960

Figure 2: Efficiency comparison on GA. The number of machines is 100, and that of jobs is 110.

where the first set of constraints provides that each job is assigned to exactly one agent, while the second one enforces that at most one feasible assignment is chosen for all the agents.

The corresponding Pricing Problem consists of finding a feasible assignment to one of the agents with minimum reduced cost, which can be reformulated as a binary knapsack problem. In our experiments we used the "Minknap" algorithm developed by Pisinger [23].

In order to implement the Branch and Bound scheme, we apply an appropriate branching rule. Note that usual branching techniques used by the generic MIP solvers are inapplicable here because forcing some variables to be zero would essentially require finding the optimal solution to the Pricing problem with a set of excluded solutions, which does not seem to be a tractable problem.

Instead, we applied the following branching rule. In each node, choose a fraction variable, say x_{ij} , corresponding to job j and machine i and generate two branches in which (a) job j must be assigned to agent i and another one in which (b) job j is not allowed to be assigned to agent i . Thus in each node we are given a subproblem, where some agent-job pairs are bounded and other ones are forbidden. All these restrictions can easily be incorporated to the knapsack problem, and the columns representing forbidden assignments can be avoided. Many different strategies were tried out to select the fractional variable, and choosing one being closest to $1/2$ seems to be the most efficient way.

Random test instances were generated with 100 machines and 110 jobs, and all of them were solved using (1) the proposed method including early cut and branching, see column "Early" (2) the proposed method without early cut and branching, see column "No Early" (3) CPLEX with default parameters, see column "CPLEX".

The executions time, the total number of generated columns and the number of the nodes of the search tree are shown in Figure 2, where each row corresponds to a random instance. Note that in most cases, the proposed method is significantly faster and explores fewer nodes of the search tree than CPLEX.

Finally, Figure 3 demonstrates that applying Early Branching might increase both the number of nodes of the search tree and the runtime. However, using it together with Early Cut improves the performance of the proposed method both in terms of execution time and the number of nodes.

References

- [1] L. H. Appelgren, *A Column Generation Algorithms for a Ship Scheduling Problem*, Transportation Science 3, 53-68, 1969

Runtime(s)				
Early	Early Cut	Early Branch	No Early	CPLEX
12.94	15.63	19.34	17.63	56.02
142.99	191.93	307.36	280.69	164.37
34.56	39.02	44.71	39.51	92.91
0.89	0.89	0.90	0.89	72.07
529.72	716.92	721.98	865.98	709.06
11.81	9.35	20.40	9.36	82.57
29.23	26.48	49.43	46.58	35.26
147.80	75.28	158.27	81.79	165.01
45.01	41.94	49.51	47.77	101.76
176.90	209.02	196.19	221.49	33.04
0.52	0.51	0.52	0.52	36.04
46.84	57.87	54.35	62.91	374.85
1179.21	1384.84	1622.06	1675.12	1922.96

Figure 3: Effect of the early ideas on GA. The number of machines is 100, and that of jobs is 110.

- [2] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W.P. Savelsbergh, P. H. Vance, *Branch-and-Price: Column Generation for Solving Huge Integer Programs*, Operations Research, 46 (1998), 316–329.
- [3] J.-M. Bourjolly, G. Laporte, H. Mercure. *A combinatorial column generation algorithm for the maximum stable set problem*. Oper. Res. Lett. 20 (1997), 21–29.
- [4] T. G. Crainic, J.-M. Rousseau, *The column generation principle and the airline crew pairing problem*, Infor, 25 (1987), 136–151.
- [5] G. Desaulniers, J. Desrosiers, I. Ioachim, M. M. Solomon, F. Soumis, D. Villeneuve. *A unified framework for deterministic time constrained vehicle routing and crew scheduling problems*. In: T. G. Crainic, G. Laporte, eds. Fleet Management and Logistics. Kluwer, Norwell, MA, pp. 57-93, 1998.
- [6] B. Dezső, A. Jüttner, P. Kovács, *Column Generation Method for an Agent Scheduling Problem*, Electronic Notes in Discrete Mathematics 36 (2010) 829836.
- [7] A. Jüttner, P. Madarasi, *A Primal–Dual Approach for Large Scale Integer Problems*, 10th Japanese-Hungarian Symposium on Discrete Mathematics and Its Applications May 22-25, 2017, Budapest, Hungary
- [8] O. du Merle, D. Villeneuve, J. Desrosiers, P. Hansen. *Stabilized column generation*. Discrete Math. 194 229237. 1999
- [9] M. L. Fisher, J. S. Shapiro, *Constructive Duality in Integer Programming*, SIAM J. Appl. Math., 27 (1974), 31–52.
- [10] M. L. Fisher, *The Lagrangian Relaxation Method for Solving Integer Programming Problems*, Management Science, 27 (1981), 1–18.
- [11] L. R. Ford, D. R. Fulkerson, *A suggested computation for maximal multicommodity network flows*. Management Science, 5 (1958), 97–7101.
- [12] A. M. Geoffrion, *Lagrangian relaxation for integer programming*, Mathematical Programming Study, 2 (1974), 82–114.
- [13] M. E. Lübbecke, J. Desrosiers *Selected Topics in Column Generation*, Operations Research Vol. 53, No. 6, 2005, pp. 10071023

- [14] R. E. Marsten, W. W. Hogan, J. W. Blankenship. *The Boxstep method for large-scale optimization*. Operations. Research. 23 389405. 1975
- [15] A. Mehrotra, M. A. Trick, *A column generation approach for graph coloring*. INFORMS J. Comput, 8 (1996), 344–354.
- [16] A Pessoa, R Sadykov, E Uchoa, F Vanderbeck, *Automation and combination of linear-programming based stabilization techniques in column generation* HAL, 2014, hal-01077984
- [17] P. H. Vance, C. Barnhart, E. L. Johnson, G. L. Nemhauser, *Airline Crew Scheduling: A New Formulations and Decomposition Algorithm*. Operations Research 45, 188-200, 1997
- [18] F. Vanderbeck, L. A. Wolsey, *Reformulation and Decomposition of Integer Programs*, In: M. Jünger et al. (eds.), 50 Years of Integer Programming 1958–2008, Springer-Verlag Berlin Heidelberg 2010, pp. 431–502.
- [19] D. Villeneuve, J. Desrosiers, M. E. Lübbecke, *On Compact Formulations for Integer Programs Solved by Column Generation*, Annals of Operations Research 139 (2005), 375–388.
- [20] P. Wentges. *Weighted Dantzig-Wolfe decomposition for linear mixed-integer programming*. International Transactions in Operational Research, 4(2):151162, 1997
- [21] W. E. Wilhelm, *A Technical Review of Column Generation in Integer Programming*, Optimization and Engineering, 2 (2001), 159–200.
- [22] L. A. Wolsey, Integer programming duality, Mathematical Programming, 20 (1981), 173–195.
- [23] D. Pisinger, A minimal algorithm for the 0-1 knapsack problem, Operations Research, 45, 758-767 (1997)