Egerváry Research Group
on Combinatorial Optimization

Technical reportS

TR-2018-09. Published by the Egerváry Research Group, Pázmány P. sétány 1/C, H–1117, Budapest, Hungary. Web site: www.cs.elte.hu/egres. ISSN 1587–4451.

A Primal-Dual Approach for Large Scale Integer Problems

Alpár Jüttner and Péter Madarasi

June 2018

A Primal-Dual Approach for Large Scale Integer Problems

Alpár Jüttner[⋆⋆⋆] and Péter Madarasi[‡]

Abstract

This paper presents a refined approach to using column generation to solve specific type of large integer problems. A primal-dual approach is presented to solve the Restricted Master problem belonging to the original optimization task. Firstly, this approach allows a faster convergence to the optimum of the LP relaxation of the problem. Secondly, the existence of both an upper and lower bound of the LP optimum at each iteration allows a faster searching of the Branch-and-Bound tree. To achieve this an *early termination* approach is presented. The technique is demonstrated on the Generalized Assignment problem and Parallel Machine Scheduling problem as two reference applications.

Keywords: large scale optimization, column generation, primal-dual methods, integer programming, scheduling

1 Introduction

One of the most successful approaches to solve large scale practical combinatorial optimization problems is the combination of special linear programming techniques such as Dantzig-Wolfe Decomposition, Column Generation or Lagrangian relaxation with Cutting Planes, Branch-and-Bound (B&B) or certain iterative rounding techniques. Methods of this type are collectively known as *Branch-and-Cut-and-Price (B-C-P)*.

These approaches assume that the problem to be solved is formulated as a huge but well structured linear (or integer) program (often referred to as a *master problem (MP)*), which is then decomposed into a higher and a lower level subproblem, referred to as *restricted master problem (RMP)* and *column generator (CG)* or *pricing problem*. In case of Lagrangian relaxation, they are called *Lagrangian dual problem* and *Lagrangian subproblem*, respectively [13].

⋆⋆⋆MTA-ELTE Egerváry Research Group, Budapest, Hungary. email: alpar.juttner@gmail.com.
Supported by the Hungarian Scientific Research Fund - OTKA, K109240 and by the János Bolyai Research Fellowship program of the Hungarian Academy of Sciences.

‡Department of Operations Research, Eötvös Loránd University, Budapest, Hungary. email: madarasip@cs.elte.hu. The project has been supported by the European Union, co-financed by the European Social Fund.

Beginning with the early results of Ford and Fulkerson [12], Appelgren[2], and others, especially after high performance linear programming solvers became widely available, Column Generation and Branch-and-Price are now standard tools for tackling various industrial math optimization problems. [3, 22, 20, 19]. It has successfully been applied to versions of traveling salesman, vehicle routing and crew scheduling problems [6], airline crew pairing [5], scheduling and fleet assignments, in telecommunication (network dimensioning, resource management and routing) and to staff scheduling problems [8], as well as to generic combinatorial optimization problems such as (integer) multicommodity flows [12], maximum stable-set [4] and graph coloring problems [16]. These works made extensive efforts on improving convergence of Column Generation and on developing efficient problem specific branching strategies (see Sections 2 and 3). On the other hand, several problem classes are still practically intractable, even though they seem to fit well into this framework.

The aim of this work is to discover ways to further widen the applicability of this approach by presenting a novel primal-dual solution technique that in one hand provides a faster convergence of the LP relaxation of the problem and in the other hand, allows a more effective execution of the usual Branch-and-Bound scheme to find the integer optimum solution.

The rest of the paper is organized as follows. Section 2 presents a refined approach that improves on the convergence rate solving the RMS problem in practice and, in addition, is able to provide both a lower and an upper bound of optimum at each iteration. Then, utilizing this property, Section 3 presents the *early termination* technique for B&B in order to speed up finding the integer optimum. Finally Section 4 presents the applicability of the proposed approach to two specific well-know optimization problems.

2 Primal-dual method for solving the RMS

When implementing a column generation based solution, 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 [14]. One of the first proposals for handling this issue is the BOXSTEP method proposed by Marsten at al. [15] and the *Stabilized Column Generation* proposed by du Merle et al. [9], which is considered the most promising stabilization technique.

Although these techniques are based on the dual considerations of the RMP, they are still *primal approaches* from the perspective of the Master Problem, with the property that they maintain a (non-optimal) feasible solution during the execution.

On the other hand, Lagrangian relaxation [10, 13, 11, 23, 19] represents a completely different approach. The Lagrangian subproblem computes a lower bound to the optimum of the Master Problem, while the Lagrangian dual problem aims at finding the parameters maximizing the lower bound, which maximum is in fact equal

to the optimum of the (linear relaxation of) the Master Problem. The Lagrange relaxation[13] of the same problem combined with the standard subgradient method often provides a rapidly converging *lower bound*, while requires solving the same pricing subproblem. Unfortunately, Lagrange relaxation alone cannot produce a primal feasible solution, and the subgradient method, while it is very simple to implement still in many cases converges extremely fast, also suffers from frequent instability, tends to "stuck" and fails to eventually find the optimum.

Therefore, we propose a combination of the subgradient method with a primal approach. For the latter one we chose a linear-programming based stabilization [17]. This technique do not use the dual solution of the master problem as the price vector for column generation, but combines it with the preceding dual solutions. The smoothing rule proposed in [21], and reconsidered in [17] suggests $\tilde{\pi}^t = \alpha\hat{\pi} + (1-\alpha)\pi^t$ using as pricing vector, where π^t denotes the current dual vector, $\hat{\pi}$ is the incumbent dual vector and $\alpha \in [0, 1)$. In [17] proposes an efficient self-adjusting scheme adapting α to the phases of the algorithm.

Although the subgradient method convergences highly effectively, its instability and occasional divergence makes it impractical in case of large problems.

We improve this method be periodically inserting subgradient-based improving phases into the above primal algorithms. The initial step size of subgradient phase is calculated from the average oscillation of $(||\hat{\pi} - \pi^t||)$ of the last primal steps and it stops when no improvents is found within a constant number of streps. In this way a steady convergence of both the lower and upper bound can be ensu

3 Branching Strategies

It is well known that the branching scheme used in the conventional Branch-and-Bound method is not applicable for column generation since it would require excluding certain solutions from the pricing subproblem. Instead, various alternative branching schemes have been proposed. They are rather problem specific and partition the integer solutions of the problem in a way that is compatible with the pricing subproblem. See e.g. [7, 1, 18] for some illustrative examples. Section 4 presents such branching rules for two specific problems.

Even though the primal-dual approach above realize a considerable speedup, we still cannot afford running the column generation up to finding LP optimum at each node of the branch tree. 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.

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.

These techniques significantly reduce the time required to process one node of the B&B tree, while — if properly implemented — it increases the size of the B&B tree only marginally.

4 Reference Applications

4.1 Generalized Assignment

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

$$\sum_{1 \le j \le n} d_{ij} x_{kj}^i \le u_i \tag{1}$$

$$x_{kj}^i \in \{0,1\} \ (j = 1, \ldots, n). \tag{2}$$

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

$$\min \sum_{1 \le i \le m} \sum_{1 \le k \le k_i} z_k^i \sum_{1 \le j \le n} c_{ij} x_{kj}^i \tag{3}$$

$$\sum_{1 \le i \le m} \sum_{1 \le k \le k_i} z_k^i x_{kj}^i = 1 \ (j = 1, \ldots, n) \tag{4}$$

$$\sum_{1 \le k \le k_i} z_k^i \le 1 \ (i = 1, \ldots, m) \tag{5}$$

$$z_k^i \in \{0,1\} \ (i = 1, \ldots, m, \quad k \in K_i), \tag{6}$$

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.

For this problem, the following two branching rules are used. In each node, we fix an agent i and job j, and create two subproblems (a) job j must be assigned to agent i (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.

4.2 Parallel Machine Scheduling

In this problem, jobs $J := \{1, \ldots, n\}$ are given with processing times p_j, due times d_j and weights w_j. These jobs are to be processed by m identical machines while minimizing $\sum_{j=1}^{n} w_j \max(0, C_j - d_j)$, where C_j is the completion time of job j.

A *schedule* of a single machine is an $s = (j_1, j_2, \ldots, j_k)$ sequence of jobs which induces completion times $C_{j_i} = \sum_{l=0}^{i} p_{j_l}$ and costs $c(s) = \sum_{i=1}^{k} w_{j_i} \max(0, C_{j_i} - d_{j_i})$ The column generation formulation of this problem consists of the set of variables x_k^s for each machine $k = 1, \ldots, m$ and for each possible s_k schedule of this machine. The formulation is as follows.

$$\min \sum_{k=1}^{m} \sum_{s \in S_k} c(s) x_s \tag{7}$$

$$\sum_{k=1}^{m} \sum_{s \in S_k} \chi_s(i) x_s = 1 \quad \forall i = 1, \ldots, j \tag{8}$$

$$\sum_{s \in S_k} x_s = 1 \quad \forall k = 1, \ldots, m \tag{9}$$

$$x_s \in \{0, 1\} \quad \forall k = 1, \ldots, m, \quad s \in S_k \tag{10}$$

Where χ_s is the characteristic vector of s, i.e. $\chi_s(i) := |\{j \in s : j = i\}|$.

The corresponding pricing subproblem consists of finding a schedule for a single machine with an additional constant "price" y_j of processing job j. In order to make it solvable by a standard dynamic programming approach[17], we also allow multiple processing of a job by a single machine, but limit the maximum number of jobs processed by a single machine to be at most n.

To obtain an integer solution we apply a branch and bound method with the following branching rule.

In every node of the branching tree, for each machine k, we specify a series of subsets J_k^i of allowed jobs as the i^{th} job to be processed. This problem can also be solved using standard dynamic programming technique in time $O(n^2 T)$, by calculating the values

$$c(t, l) := \begin{cases} +\infty & \text{if } t < 0, \\ 0 & \text{if } k = 0, t = 0, \\ \min\limits_{j \in J_k^l} \left(w_j \max(0, t - d_j) - y_j + c(t - p_j, l - 1) \right) & \text{otherwise} \end{cases} \tag{11}$$

for each values of $k = 0, \ldots, n$ and $t = 0, \ldots, T$. The computed value $c(t, l)$ is the cost of the optimal sequence consisting of j jobs with the last job finished in time t.

In the root node of the branching tree we set $J_k^i := J$ for all k and i. Then we apply two different kind of branchings.

1. If a job j appears in the (fractional) schedule of more than one machines, then we choose a machine k and create two subproblems by (a) assigning job j solely to machine k and (b) the disallowing processing job j by machine k. These constraints can be enforced by removing the job j from the corresponding constraint sets $J_{k'}^i$.

2. If more than one job appears in the (fractional) schedule of a certain machine k at a position i, we chose one and create two subproblem by either (a) allowing this job only to be processed at positon k and (b) disallowing it to be processed at position k.

It is easy to see that if neither of the above branching rules are applicable for an *optimal* solution of the linear problem obtained by the column generation, then it is an optimal integer solution of the current subproblem.

In order to apply the early branching approach, we generate columns until the objective function value becomes lower than the best integer found so far. Then we continue generating, until either an optimal solution is found or one of the branching rules becomes applicable. Then we choose the biggest non integer variable and branch according to that.

5 Computational results – Generalized Assignment

To evaluate the efficiency of the proposed method, computational tests were carried out with the GA problem.

For each specific problem, the a separation problem must be determined and implemented s.t. the implemented solver can use it as an oracle. In the case of GA, the separation problem is the knapsack problem, which has been solved using Pisinger's minknap [24], a state of the art knapsack problem solver.

On the other hand a branching rule should be determined and implemented, as well. In our test the following rule was used: choose a fraction variable ,say , corresponding to job j and machine i, and generate two branches (a) job j must be processed on machine i (b) job j is forbidden to be processed on machine i. 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 branching [Early] 2) the proposed method without early branching [No Early] 3) CPLEX with default setup [CPLEX]. The runtime, the number of columns and the number of B&B nodes are shown in Figure 1, where each row corresponds to a random instance.

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	0.9	0.89	72.07
529.72	716.92	721.08	865.98	709.06
11.81	9.35	20.4	9.36	82.57
29.23	26.48	49.43	46.58	35.26
147.8	75.28	158.27	81.79	165.01
45.01	41.94	49.51	47.77	101.76
176.9	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 2: Effect of the early ideas on GA. The number of machines is 100, and that of jobs is 110.

Runtime (s)			Colums		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
147.8	81.79	165.01	2714	3120	861	547	19315
45.01	47.77	101.76	2519	2771	287	347	8357
176.9	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 1: Efficiency comparison on GA. The number of machines is 100, and that of jobs is 110.

To evaluate the contribution of each idea to the speed up, Figure 2 shows the runtimes when using 1) the proposed method including early branching [Early] 2) early cut only [Early Cut] 3) early branch only [Early Branch] 4) column generation with neither early cut nor early branching [No Early] 5) Default CPLEX [CPLEX]. Again, each row corresponds to a random instance with 100 machines and 110 jobs.

6 Conclusion

This paper presented an improved primal-dual method for solving the RMS problem of typical large scale combinatorial optimization problems which in turn allows imple-

menting the B&B scheme with only partially solved subproblems. Our initial practical evaluation shows promising improvements on the reference applications compared to the existing solutions.

References

[1] R. Anbil, R. Tanga, E. L. Johnson, *A Global Approach to Crew-pairing Optimization*. IBM Systems J. 31, 71-78, 1992

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

[3] 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.

[4] 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.

[5] T. G. Crainic, J.-M. Rousseau, *The column generation principle and the airline crew pairing problem*, Infor, 25 (1987), 136–151.

[6] G. Desaulniers, J. Desrosiers, I. Ioachim, M. M. Solomon, F. Soumis, D. Villeneuve. *A unified framework for deterministic time con strained vehicle routing and crew scheduling problems.* In: T. G. Crainic, G. Laporte, eds. Fleet Management and Logistics. Kluwer, Norwell, MA, pp. 57-93, 1998.

[7] M. Desrochers, F. Soumis, *A Column Generation Approach to the Urban Transit Crew Scheduling Problem*. Transportation Science 23, 1-13, 1989.

[8] B. Dezső, A. Jüttner, P. Kovács, *Column Generation Method for an Agent Scheduling Problem*, Electronic Notes in Discrete Mathematics 36 (2010) 829836.

[9] O. du Merle, D. Villeneuve, J. Desrosiers, P. Hansen. *Stabilized column generation*. Discrete Math. 194 229237. 1999

[10] M. L. Fisher, J. S. Shapiro, Constructive Duality in Integer Programming, SIAM J. Appl. Math., 27 (1974), 31–52.

[11] M. L. Fisher, The Lagrangian Relaxation Method for Solving Integer Programming Problems, Management Science, 27 (1981), 1–18.

[12] L. R. Ford, D. R. Fulkerson, *A suggested computation for maximal multicommodity network flows.* Management Science, 5 (1958), 97–7101.

[13] A. M. Geoffrion, *Lagrangean relaxation for integer programming*, Mathematical Programming Study, 2 (1974), 82–114.

[14] M. E. Lübbecke, J. Desrosiers *Selected Topics in Column Generation*, Operations Research Vol. 53, No. 6, 2005, pp. 10071023

[15] R. E. Marsten, W. W. Hogan, J. W. Blankenship. *The Boxstep method for large-scale optimization.* Operations. Research. 23 389405. 1975

[16] A. Mehrotra, M. A. Trick, *A column generation approach for graph coloring.* INFORMS J. Comput, 8 (1996), 344–354.

[17] A Pessoa, R Sadykov, E Uchoa, F Vanderbeck, *Automation and combination of linear-programming based stabilization techniques in column generation* HAL, 2014, hal-01077984

[18] 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

[19] 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.

[20] 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.

[21] P. Wentges. *Weighted dantzigwolfe decomposition for linear mixed-integer programming.* International Transactions in Operational Research, 4(2):151162, 1997

[22] W. E. Wilhelm, *A Technical Review of Column Generation in Integer Programming*, Optimization and Engineering, 2 (2001), 159–200.

[23] L. A. Wolsey, Integer programming duality, Mathematical Progamming, 20 (1981), 173–195.

[24] D. Pisinger, A minimal algorithm for the 0-1 knapsack problem, Operations Research, 45, 758-767 (1997)