Clique-based facets for the precedence constrained knapsack problem

Andreas Bley (Speaker) [*] Natashia Boland [†] Christopher Fricke [‡]
Gary Froyland [§]

1 Introduction

We consider a knapsack problem with precedence constraints imposed on pairs of items, known as the precedence constrained knapsack problem (PCKP). This problem arises as a subproblem in many scheduling and planning problems [12, 17, 13, 16]. We present a new approach for determining facets of the PCKP polyhedron based on clique inequalities. This approach generates facets that cannot be found through the existing cover-based approaches. We also propose a simple scheme to fix variables of the linear relaxation of the PCKP in a preprocessing step. It is shown that both the variable fixing and the addition of clique-based inequalities for the PCKP are highly beneficial in practical computations, reporting on results for open pit mine production scheduling problems.

Let $\mathcal{N}$ be a set of items and $\mathcal{S} \subseteq \mathcal{N} \times \mathcal{N}$ denote a partial order or set of precedence relationships on the items. A precedence relationship $(i, j) \in \mathcal{S}$ exists if item i can be placed in the knapsack only if item j is in the knapsack. Each item $i \in \mathcal{N}$ has a value $c_i \in \mathbb{Z}$ and a weight $a_i \in \mathbb{Z}^+$, and the knapsack has a capacity $b \in \mathbb{Z}^+$. The PCKP is the problem of finding a maximum value subset of $\mathcal{N}$ whose total weight does not exceed the knapsack capacity, and that also satisfies the precedence relationships. The precedence constraints can be represented by the directed graph $G = (\mathcal{N}, \mathcal{S})$, where the node set is the set of all items $\mathcal{N}$, and each precedence constraint in $\mathcal{S}$ is represented by a directed arc. Without loss of generality we may assume that G is acyclic and contains no redundant relationships, that is, $\mathcal{S}$ is the set of all immediate predecessor arcs. With

$$x_i = \begin{cases} 1, \text{if item } i \text{ is included in the knapsack} \\ 0, \text{otherwise} \end{cases} \qquad \text{for all } i \in \mathcal{N}$$

[*] bley@zib.de. Zuse Institute Berlin, Takustr. 7, D-14195 Berlin, Germany
[†] natashia.boland@newcastle.edu.au. School of Mathematical and Physical Sciences, University of Newcastle, Callaghan NSW 2308, Australia
[‡] TSG Consulting, Level 11, 350 Collins Street, Melbourne VIC 3000, Australia.
[§] g.froyland@unsw.edu.au. School of Mathematics and Statistics, University of New South Wales, Sydney NSW 2052, Australia

the PCKP may be written as

$$
\begin{array}{lrcll}
 & \max \sum_{i \in \mathcal{N}} c_i x_i & & & \\
\text{(PCKP)} & \text{s.t.} \sum_{i \in \mathcal{N}} a_i x_i & \leq & b & \\
 & x_i & \leq & x_j & \text{for all } (i,j) \in \mathcal{S} \\
 & x_i & \in & \{0,1\} & \text{for all } i \in \mathcal{N}.
\end{array}
$$

In this paper, we describe facet-defining inequalities for the polyhedron defined by the feasible solutions of (PCKP). Unlike previous work [9, 15, 18], we do not take knapsack covers as our starting point, but instead investigate clique inequalities derived from a graph representing pairwise conflict relationships between variables. These inequalities can be separated very efficiently and they proved to be computationally effective when solving practical problems containing PCKP as a subproblem.

2 Variable Fixing

Let A_i be the *minimal set of items*, including item i, that must be included in the knapsack for item i to be included, i.e., A_i is the set of all predecessors of i in the transitive closure of G. The total capacity required to include all items in a set B in the knapsack is $H(B) := \sum_{j \in A(B)} a_j$. It follows that $H(\{i\}) = \sum_{j \in A_i} a_j$ is the capacity required to include item i in the knapsack.

In a feasible solution x of (PCKP) we clearly must have $x_i = 0$ for any item i with $H(\{i\}) > b$. Based on this simple observation, in many practical instances a significant number of variables can be fixed to 0 prior to optimization, reducing the size of the PCKP and the integrality gap of the remaining integer programming formulation. Note that the values $H(\{i\})$ of *all* items $i \in \mathcal{N}$ can be computed in time $O(\mathcal{S})$ by dynamic programming. The overall variable fixing scheme therefore only takes linear time. As shown in Table 1 at the end of this paper, this simple variable fixing has a very positive computational effect in practice.

3 Clique-Cuts

Extending this simple idea to pairs of items, strong facet-defining inequalities for $conv(P)$ can be identified. Let $\{i,j\}$ be a pair of distinct items with $H(\{i,j\}) > b$, i.e., the size of both items together with their predecessors exceeds the knapsack capacity. Then we clearly must have $x_i + x_j \leq 1$ in any feasible solution x of (PCKP).

For the PCKP instance under consideration we define a conflict graph based on these pairwise conflicts. This conflict graph $CG = (\mathcal{N}, E)$ contains the edge $\{i,j\} \in E$ if and only if $H(\{i,j\}) > b$. One easily finds that, for any (maximal) clique C in CG, the (maximal) clique inequality

$$\sum_{j \in \mathcal{C}} x_j \leq 1 \tag{1}$$

is valid for (PCKP). If the clique $\mathcal{C}$ is inclusion-wise maximal and the set $P(\mathcal{C}) = \cap_{i \in \mathcal{C}} A_i$ of common predecessors of the items in $\mathcal{C}$ is empty, (1) may be facet-defining for $(PCKP)$. If the

items in $\mathcal{C}$ have a common predecessor $j \in P(\mathcal{C})$, one obtains a stronger inequality of the form

$$\sum_{i \in \mathcal{C}} x_i \ \leq \ x_j. \tag{2}$$

This inequality may be facet defining if j is a maximal element in $P((C))$ with respect to the given precedence order among the items. The precise conditions when the clique inequalities (1) and the lifted clique inequalities (2) are facet defining are discussed in [10, 6].

The computational advantage of these inequalities is their simplicity, which allows for practically efficient separation algorithms via maximum weight clique computations in the conflict graph. The conflict graph can be computed in a straightforward way in $O(|\mathcal{N}|^2|\mathcal{S}|^2)$, the maximal (and weight weight minimal) common predecessor $j \in P(\mathcal{C})$ of a given clique $\mathcal{C}$ can be determined in $O(|\mathcal{S}||\mathcal{C}|)$. Although the problem of finding a maximum weight clique in a graph is NP-hard, there are numerous fast heuristics and practically efficient exact solution methods for that can be applied. (Even more, most general purpose integer programming solvers already contain efficient maximum clique algorithms for the separation of clique inequalities based on other conflict structures, which can be applied for PCKP cliques out of the box.) Bomze et al. [7] provide a comprehensive overview of such methods. In our implementation we use the Sequential Greedy heuristic and the combinatorial branch and bound algorithm proposed in [8] with a limit of at most 100 branch and bound nodes to explore. These algorithms proved to be efficient for the separation of clique inequalities in the general purpose integer programming solver SCIP [1].

4 Computational Results

We evaluated the impact of the techniques on several open pit mine production planning problems provided by our research partner BHP Billiton. In these problems, an orebody discretized into blocks is to be extracted over several periods. Each block has individual profit and tonnage. Geological restrictions concerning the admissible shape of the mine are modeled as precedence constraints among the blocks, mining and processing capacities lead to knapsack constraints for each period. A detailed description of the integer programming models can be found in [5].

Table 1 shows the impact of applying the new variable fixing and the clique-based inequalities at the root node of the branch and bound tree. All problems are solved with CPLEX 11.0 on a 2.66 GHz Intel Pentium 4 machine to an optimality gap of 1%, variable fixing and the separation of clique-inequalities are performed via CPLEX callback functions implemented in C++. The first three columns of Table 1 show the number of blocks, elementary precedence relations, and considered periods. The remaining columns display the number of branch-and-bound nodes, the integrality gap at the root node, and the overall computation time for the plain standard formulation and with variable fixing and clique inequalities. It can be seen than both techniques lead to a substantial reduction of the integrality gap at the root node and, in many cases, also to less branch-and-bound nodes and lower overall computation times. Note that even with the most aggressive variable probing and cut generation strategies CPLEX was not able to deduce the variable fixing or to derive equally strong cutting planes at the root node on its own. Similar results are obtained also for underground mining problems and for precedence constrained knapsack problems arising in the context of routing optimization in data networks [3, 4].

Data set			Standard Formulation				With Fixing				With Fixing and Cliques			
N	$\|\mathcal{E}\|$	T	Vars	B&B Nodes	Root Node Gap (%)	Time (sec)	Fixed Vars	B&B Nodes	Root Node Gap (%)	Time (sec)	Added clique cuts	B&B Nodes	Root Node Gap (%)	Time (sec)
Shallow Pits														
67	190	5	335	407	5.2	4.8	46	432	2.8	2.8	14	142	1.5	2.6
67	190	10	670	41723	9.6	531.4	103	59319	4.6	444.3	38	102700	2.9	1589.7
115	368	5	575	225	4.1	11.2	40	97	2.3	8.6	18	50	1.2	9.3
115	368	10	1150	5925	5.1	224.0	106	2882	2.6	88.5	34	1996	1.5	73.6
182	615	5	910	100	2.0	22.9	45	214	1.8	17.8	14	93	1.2	10.8
182	615	10	1820	4136	3.4	505.6	101	954	2.0	119.5	29	1356	1.3	143.9
354	1670	5	1770	3	1.0	12.2	21	1	0.9	9.1	8	1	0.8	9.0
354	1670	10	3540	239	1.1	120.5	93	300	1.0	92.9	10	431	1.0	123.5
Deep Pits														
66	312	5	330	62	3.2	1.1	38	50	2.4	0.9	6	3	1.1	0.4
66	312	10	660	1567	7.2	42.0	94	1750	5.1	35.0	13	539	1.9	11.5
90	544	5	450	2102	6.2	9.2	54	75	4.5	1.6	9	1	0.9	0.2
90	544	10	900	10473	9.1	213.4	140	1387	5.2	34.2	25	599	2.2	21.8
166	1551	5	830	460	25.6	34.7	209	312	5.6	11.9	21	187	4.1	11.6
166	1551	10	1660	44359	37.0	2477.8	475	13545	6.8	317.9	21	3663	4.7	117.9
420	5900	5	2100	2514	37.2	487.2	620	445	9.4	112.3	11	462	5.3	153.8
420	5900	10	4200	71268	55.2	61224.5	1385	3739	9.2	1023.8	19	6417	5.6	3114.7

Table 1: Computational results for open pit mine production scheduling with fixing and cuts added at the root node

References

[1] Achterberg, T.: Constraint Integer Programming, PhD Thesis, Technische Universität Berlin, 2007.

[2] Balas, E.: Disjunctive Programming. Ann. Discr. Math. **5**, 3-51 (1979)

[3] A. Bley. *Routing and Capacity Optimization for IP Networks.* PhD thesis, Technische Universität Berlin, 2007.

[4] A. Bley. An integer programming algorithm for routing optimization in IP networks. In *Proceedings of the 16th Annual European Symposium on Algorithms (ESA 2008), Karlsruhe, Germany*, pages 198–209, 2008.

[5] A. Bley, N. Boland, C. Fricke, and G. Froyland. A strengthened formulation and cutting planes for the open pit mine production scheduling problem. Technical Report, submitted, 2008.

[6] N. Boland, C. Fricke, G. Froyland, R. Sotirov, and A. Bley, Clique-based facets for the precedence constrained knapsack problem. Technical Report, submitted, 2008.

[7] I. Bomze, M. Budinich, P. Pardalos and M. Pelillo, The maximum clique problem, In *Handbook of Combinatorial Optimization*, volume 4, Kluwer Academic Press, 1999.

[8] R. Borndörfer and Z. Kormos, An Algorithm for Maximum Cliques, Unpublished working paper, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1997.

[9] Boyd, E.A.: Polyhedral results for the precedence-constrained knapsack problem. Discrete Appl. Math. **41**, 185-201 (1993)

[10] C. Fricke, Applications of Integer programming in Open Pit Mining, PhD Thesis, University of Melbourne, 2006.

[11] Garey, M.R. and Johnson, D.S.: Computers and intractability: A guide to the theory of NP-completeness, W. H. Freeman and Company, San Francisco (1979)

[12] Ibarra, O.H., Kim, C.E.: Approximation algorithms for certain scheduling problems. Math. Oper. Res. **4**, 197-204 (1978)

[13] Johnson, D.S., Niemi, K.A.: On knapsacks, partitions, and a new dynamic programming technique for trees. Math. Oper. Res. **8**, 1-14 (1983)

[14] Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization, Wiley-Interscience, New York (1988)

[15] Park K., Park, S.: Lifting cover inequalities for the precedence-constrained knapsack problem. Discrete Appl. Math. **72**, 219-241 (1997)

[16] Shaw, D.X., Cho, G., Chang, H.: A depth-first dynamic programming procedure for the extended tree knapsack problem in local access network design. Telecommunication Systems **7**, 29-43 (1997)

[17] Stecke, K.E., Kim, I.: A study of part type selection approaches for short-term production planning. Int. J. Flexible Manufacturing Systems **1**, 7-29 (1988)

[18] Van de Leensel, R.L.M.J., van Hoesel, C.P.M., van de Klundert, J.J.: Lifting valid inequalities for the precedence constrained knapsack problem. Math. Program. **86**, 161-185 (1999)