Partitioning a Weighted Partial Order

Linda S. Moonen, *Department of Operations Research and Business Statistics, KULeuven*
Frits C.R. Spieksma *Department of Operations Research and Business Statistics, KULeuven*

Keywords: partial order, approximation, min-cost flow

1. Introduction

Consider a partially ordered set $(X, \prec)$. We say that two elements $i, j \in X$ are *comparable* if either $i \prec j$ or $j \prec i$. A *chain* C is defined as a subset of X such that all elements $i, j \in C$ are pairwise comparable. An *antichain* A is a subset of X such that no two elements $i, j \in A$ are comparable. The *size* of a chain (or antichain) is equal to the number of elements contained in it (Trotter [10]).

Now, the problem of partitioning a partially ordered set $(X, \prec)$ into a minimal number of chains such that each element of X belongs to at least one chain, is a well-known, fundamental problem in operations research. This problem is solvable in polynomial time, and the size of a maximum antichain is equal to the minimum number of chains needed to cover all elements of X (Dilworth [3]).

Shum and Trotter [9] generalize this problem by assuming that an integer B is given that bounds the size of a chain. Thus, in this setting no more than B elements can be in a chain. They show that the corresponding decision problem is $\mathcal{NP}$-complete (even for a fixed $B = 3$), and they study the facial structure of a formulation of this problem.

In this work we further generalize this problem by assuming that a weight w_i for each $i \in X$ is given such that $w_i \leq w_j$ if $i \prec j$. Moreover, we define the weight of a chain C as $\max_{i \in C} w_i$, and we refer to a chain containing at most B elements as a B-chain. The problem is now to partition X into a minimum-weight set of B-chains. We refer to this problem as Minimum Weight Partition into B-chains, or MWPB. Observe that when $w_i = 1$ for all $i \in X$ the problem dealt with by Shum and Trotter [9] arises.

Applications

An application of MWPB is described in Moonen and Spieksma [7]. Given are a number of rectangular shaped boxes, each with a given length ℓ_i and width d_i. These boxes need to be stacked on top of each other such that if box i is on top of box j we have $\ell_i \leq \ell_j$ and $d_i \leq d_j$. Thus, we must partition the set of boxes into a number of "pyramid-shaped" stacks; moreover, each stack can contain at most B boxes. The goal is to minimize the total area of the stacks, where the area of a stack equals the area of its largest (bottom) item. Observe that this corresponds to a special case of MWPB, where $w_i = \ell_i \times d_i$ for each i, and where the partial order induced by the lengths and widths (i.e., $i \prec j$ if and only if $\ell_i \leq \ell_j \wedge d_i \leq d_j$) can be embedded in two dimensions; in other words, the dimension of the partial order equals 2.

Other applications of MWPB can be found in the field of mutual exclusion scheduling (Baker and Coffman [1], Jansen [6]), also known as batch scheduling with job compatibilities (Boudhar [2], Finke et al. [4]). In such a problem jobs are given, each with a given processing time p_i, and for each pair of jobs it is known whether they can be processed on the same machine. A machine can process at most B jobs simultaneously, and the time a machine needs equals the maximum processing time of the jobs assigned to that machine. The problem is then to assign the jobs to the machines, respecting the compatibilities, while minimizing the total time needed by the machines to process all jobs. To represent the job compatibilities, often a graph is used; different types of graphs lead to different complexity results. In our setting, the graph corresponding to the job compatibilities is a comparability graph (see Golumbic [5]), and the weights are the processing times (i.e., $w_i = p_i$ for each i). Notice that for our results to be applicable in this mutual exclusion scheduling we need that $p_i \leq p_j$ when $i \prec j$.

Our results

In this abstract we report the following results:

- Strengthening a result from Shum and Trotter [9], we show that MWPB is $\mathcal{APX}$-hard, rendering the existence of a PTAS unlikely.
- We propose two lower bounds, each of which can be arbitrarily bad when compared to the value of the optimum. The maximum of these lower bounds, however, is shown never to be less than half the optimum value.
- We describe a simple algorithm that yields a solution with a value guaranteed not to exceed twice the optimum value. The analysis is shown to be tight.

2. Complexity and Lower Bounds

Complexity

The decision problem corresponding to MWPB can be formulated as follows:

Given an integer B, a partial order $(X, \prec)$, weights $w_i, i \in X$, and an integer K, does there exist a partition of X into B-chains such that the sum of the weights of the B-chains does not exceed K?

As stated in Section 1, Shum and Trotter [9] prove that this decision problem is $\mathcal{NP}$-complete. We can strengthen their result:

Theorem 1 *MWPB is $\mathcal{APX}$-hard, even if*

- $B = 3$, *and*
- $w_i = 1\ \forall i$, *and*
- *each element occurs in no more than 3 chains.*

We refer to Moonen and Spieksma [8] for a proof.

Lower Bounds

Consider an instance of MWPB, containing an integer B and n elements, each with a weight w_i, $1 \leq i \leq n$. We assume that the elements are ordered such that $w_1 \geq w_2 \geq \ldots \geq w_n$. Let OPT denote the value of an optimal solution to the instance. We define three lower bounds lb_i, $i = 1, 2, 3$, as follows:

1. $lb_1 = w_1 + w_{B+1} + \ldots + w_{(\lceil \frac{n}{B} \rceil - 1)B+1}$. Since the size of a chain cannot exceed B, lb_1 is obviously a lower bound for OPT.
2. When we omit the size constraint (i.e., if there is no restriction on the size of a chain), a relaxation of problem MWPB appears. Solving this relaxation gives a minimum-weight set of chains with value MWC. We set $lb_2 = MWC$.
3. $lb_3 = \max(lb_1, lb_2)$.

Theorem 2 *We can calculate the value of lb_2 by solving a min-cost flow problem.*

Proof: In order to compute the value of lb_2, we create a directed graph $D = (V, A)$. V contains $2n + 2$ nodes: 2 nodes i' and i'' for every $i \in X$, a source s and a sink t. We draw an arc from s to each node i', with cost 0. Then we add an arc from each node i'' to t with cost w_i. Next, we add an arc from a node i' to its copy i'' with cost 0, and we add arcs from nodes i'' to j' if $i \prec j$, also with cost 0. Finally we add an arc from s to t with cost 0. All nodes have supply zero, except for s which has supply n, and t, which has supply $-n$ (a demand of n). All arc capacities are equal to 1, and for the arcs from a node i' to its copy i'' we have a lower bound on the flow of 1. Now, it is easily verified that a min-cost flow in D corresponds to a solution of the relaxation of MWPB (i.e., the relaxation where the size constraint is not taken into account) and vice versa. □

Notice that this algorithm solves a weighted generalization of the classical result of Dilworth [3].

We now argue that lb_1 and lb_2 can be arbitrarily bad, even in the unweighted case. Consider lb_1, and suppose we are given a problem instance with $B = n$, such that no two elements are comparable, and suppose that each element has weight 1. One easily verifies that OPT equals n, while lb_1 equals 1.

Next, consider lb_2, and suppose we have a problem instance with $B = 1$, such that all elements are comparable, and that each element has weight 1. Again, OPT equals n, while lb_2 gives a value of 1. So, we cannot give a constant performance guarantee for either of these lower bounds. However, no instance exists where both lower bounds are arbitrarily bad. Indeed, let us now consider the maximum of these two lower bounds, lb_3.

Claim 1 *For each instance of MWPB: $lb_3 \geq \frac{1}{2}OPT$. Moreover, this bound is tight.*

We postpone the proof of this inequality to Theorem 3; we refer to [8] for an instance for which this bound is tight.

3. A 2-approximation algorithm for MWPB

In this section we propose a 2-approximation algorithm for MWPB, and show that it is tight.

Consider the following heuristic H:

Step 1. Omit the size constraint, and find a minimum-weight set of chains as described in Theorem 2.

Step 2. For each chain consisting of say K elements $i_1, \ldots, i_K$, with $i_1 \succ i_2 \succ \ldots \succ i_K$, partition it into $\lceil \frac{K}{B} \rceil$ B-chains such that elements $i_{(j-1)B+1}, i_{(j-1)B+2}, \ldots, i_{\min(jB,K)}$ form B-chain j, $j = 1, \ldots, \lceil \frac{K}{B} \rceil$.

Theorem 3 *H is a 2-approximation algorithm for problem MWPB.*

Proof: We assume that the elements are ordered such that $w_1 \geq w_2 \geq \ldots \geq w_n$. Suppose we find a solution using heuristic H with value v_H, where in the first step we find a decomposition into p chains, $C_1, ..., C_p$. In the second step we partition each of these p chains into a number of B-chains. The maximal elements of the B-chains that contain the maximal elements of C_ℓ are referred to as i_ℓ, $1 \leq \ell \leq p$. All other maximal elements of B-chains are referred to as j_ℓ, $1 \leq \ell \leq k$. Assume, without loss of generality, that $j_1 \leq j_2 \leq \ldots \leq j_k$. Notice that we can associate to each item j_ℓ a set of B items that belong to the same chain found in Step 1 as j_ℓ, and are the smallest B items that dominate j_ℓ. Let us refer to this set of items as $S(j_\ell)$, $1 \leq \ell \leq k$.

Claim 2 $j_\ell \geq \ell B$

Argument: Consider the sets $S(j_\ell)$, $\ell = 1, \ldots, k$. Since these sets are pairwise disjoint, the number of items that must precede j_ℓ, $1 \leq \ell \leq k$, equals at least ℓB. □

The inequality from Claim 2 implies $\sum_{\ell=1}^{k} w_{j_\ell} \leq \sum_{\ell=1}^{k} w_{\ell B} \leq lb_1$. And, obviously, $\sum_{\ell=1}^{p} w_{i_\ell} = lb_2$. Thus, we have that $v_H = \sum_{\ell=1}^{p} w_{i_\ell} + \sum_{\ell=1}^{k} w_{j_\ell} \leq lb_1 + lb_2 \leq 2OPT$. Also, notice that $lb_3 + lb_3 \geq lb_1 + lb_2 \geq v_H \geq OPT$, implying Theorem 3. □

In [8], instances are described that imply tightness of this 2-approximation algorithm.

4. Computational results

We implemented the 2-approximation algorithm in C++, using the CPLEX network solver to solve the min-cost flow problems, and we tested it on a number of real-world and randomly generated problem instances of MWPB. We use 3 different data sets for the experiments. The first data set contains 50 real-world instances provided to us by Bruynzeel Storage Systems, the second data set contains 50 randomly generated instances, and the third data set contains 50 randomly generated instances that have small clique-width (see Moonen and Spieksma [7]). The problem instances for all three data sets contain between 20 and 200 elements. We solve each problem instance for 5 different values of B, so we have 250 experiments for each data set. (In the real-world setting of Bruynzeel Storage System, B equals 12.) Since the computation times were 0.00 seconds, for all instances, we omit them from the tables with results.

Table 1: Results for lower bounds

B	*Data set 1*		*Data set 2*		*Data set 3*	
	$lb_3 = lb_1$	$lb_3 = lb_2$	$lb_3 = lb_1$	$lb_3 = lb_2$	$lb_3 = lb_1$	$lb_3 = lb_2$
3	100%	0%	96%	4%	96%	4%
6	100%	0%	74%	26%	74%	26%
9	98%	2%	50%	50%	28%	72%
12	94%	6%	42%	58%	14%	86%
15	94%	6%	26%	74%	8%	92%

In Table 1 we compare the lower bounds. As lb_3 is defined as the maximum of lb_1 and lb_2, we want to know how many times lb_3 equals lb_1, and how many times it equals lb_2. So in Table 1 we give, for each of the three data sets, the percentage of the number of times that lb_3 equals lb_1 and the number of times that lb_3 equals lb_2.

Table 2: Results for data set 1: real-world instances

B	lb_1	lb_2	lb_3	v_H	Δ_3 (%)	
					avg_{Δ_3}	max_{Δ_3}
3	1085.06	96.02	1085.06	1096.12	1.41	5.21
6	556.12	96.02	556.12	574.74	3.85	14.81
9	379.96	96.02	380.22	396.66	5.51	21.60
12	293.76	96.02	294.82	316.08	7.04	24.27
15	240.40	96.02	242.18	259.64	7.78	20.10

Tables 2, 3, and 4 show a comparison between the values of the three lower bounds and the value of the 2-approximation algorithm. In the first column we give the value of B, and in the next four columns we show the average values of the three lower bounds (lb_1, lb_2, and lb_3) and the 2-approximation algorithm (v_H). Of

Table 3: Results for data set 2: random instances

B	lb_1	lb_2	lb_3	v_H	Δ_3 (%)	
					avg_{Δ_3}	max_{Δ_3}
3	6518.64	1508.44	6589.06	6935.18	9.72	24.09
6	3317.18	1508.44	3338.78	3877.36	15.66	26.17
9	2259.08	1508.44	2314.28	2867.24	15.03	27.10
12	1726.68	1508.44	1818.90	2348.64	14.21	30.51
15	1411.90	1508.44	1585.92	2066.94	13.05	31.92

Table 4: Results for data set 3: instances with small clique-width

B	lb_1	lb_2	lb_3	v_H	Δ_3 (%)	
					avg_{Δ_3}	max_{Δ_3}
3	3215.70	1357.98	3219.66	3607.64	11.13	22.76
6	1686.28	1357.98	1765.54	2200.00	18.57	27.85
9	1177.96	1357.98	1451.42	1762.92	15.96	30.33
12	926.42	1357.98	1389.50	1574.78	11.15	27.43
15	775.38	1357.98	1364.50	1470.34	7.44	28.52

course, as can be seen in the third column, the value of B does not influence lb_2. Finally, the column labelled Δ_3 shows the average (avg_{Δ_3}) and the maximum (max_{Δ_3}) difference between the values of v_H and lb_3. These differences are all given in percentages (i.e., $\frac{v_H - lb_3}{v_H} \cdot 100\%$).

From these results we see that the performance of the lower bounds is very different for the different data sets. For the first data set, that contains the real-world problem instances, lb_1 is clearly better than lb_2: in 97.20% of all experiments, the value of lb_1 is larger than the value of lb_2. If we look at the second data set, we see that lb_1 still performs better compared to lb_2, but the percentage of experiments for which lb_1 is larger than lb_2 is only 57.60% for data set 2. However, for data set 3 we see that lb_2 performs slightly better than lb_1: in 56.00% of all experiments the value of lb_2 is larger than the value of lb_1.

Next we compare the values of lb_3 with the values of the approximation algorithm. The difference between the value of the approximation algorithm and the value of lb_3 could get as large as 100%, however, the maximum difference among all experiments from the three data sets is equal to 24.27% for data set 1, 31.92% for data set 2, and 30.33% for data set 3. The average difference for the three data sets equal 5.12% for data set 1, 13.53% for data set 2, and 12.85% for data set 3.

5. Conclusions

In this abstract we discuss the problem of partitioning a weighted partially ordered set into chains of bounded size. We established its complexity, proposed three lower bounds, and presented a 2-approximation algorithm for solving it. The approximation algorithm is tested on a number of real-world and randomly generated problem instances.

References

[1] Baker, B.S. and E.G. Coffman, Jr. (1996). Mutual exclusion scheduling. *Theoretical Computer Science* **162** 225-245.

[2] Boudhar, M. (2003). Scheduling a batch processing machine with bipartite compatibility graphs. *Mathematical Methods of Operations Research* **57** 513-527.

[3] Dilworth, R.P. (1950). A decomposition theorem for partially ordered sets. *Annals of Mathematics* **51** 161-166.

[4] Finke, G., V. Jost and M. Queyranne (2004). Batch processing with interval graph compatibilities between tasks. In *Proceedings of Discrete Optimization Methods in Production and Logistics*, Omsk-Irkutsk, 88-94.

[5] Golumbic, M.C. (1980). *Algorithmic graph theory and perfect graphs.* Academic Press, New York.

[6] Jansen, K. (2003). The mutual exclusion scheduling problem for permutation and comparability graphs. *Information and Computation* **180** 71-81.

[7] Moonen, L.S. and F.C.R. Spieksma (2006). Exact algorithms for a loading problem with bounded clique width. *INFORMS Journal on Computing* **18** 455-465.

[8] Moonen, L.S. and F.C.R. Spieksma (2005). Partitioning a weighted partial order. Research Report 0538, Department of Operations Research and Business Statistics, KULeuven.

[9] Shum, H. and L.E. Trotter, Jr. (1996). Cardinality-restricted chains and antichains in partially ordered sets. *Discrete Applied Mathematics* **65** 421-439.

[10] Trotter, W.T. (1992). *Combinatorics and partially ordered sets: dimension theory.* The Johns Hopkins University Press, Baltimore.