Automatically Identifying Dantzig Wolfe Decompositions Using Spectral Clustering

Prachi Shah, Suraj Shourie

1 Introduction

In the field of optimization, Linear and Mixed Integer Programming (MIP) are widely used for solving a range of problems. Historically, ideas from linear programming have inspired many of the central concepts of optimization theory, such as duality, decomposition, and the importance of convexity and its generalizations. Likewise, linear programming was heavily used in the early formation of microeconomics, and it is currently utilized in company management, such as planning, production, transportation, and technology. Although the modern management issues are ever-changing, most companies would like to maximize profits and minimize costs with limited resources.

Often, these problems in the real world are too large to be solved in reasonable time, which has motivated research in large-scale optimization. Certain special cases of linear programming, such as network flow problems and multicommodity flow problems, are considered important enough to have much research on specialized algorithms to make them tractable.

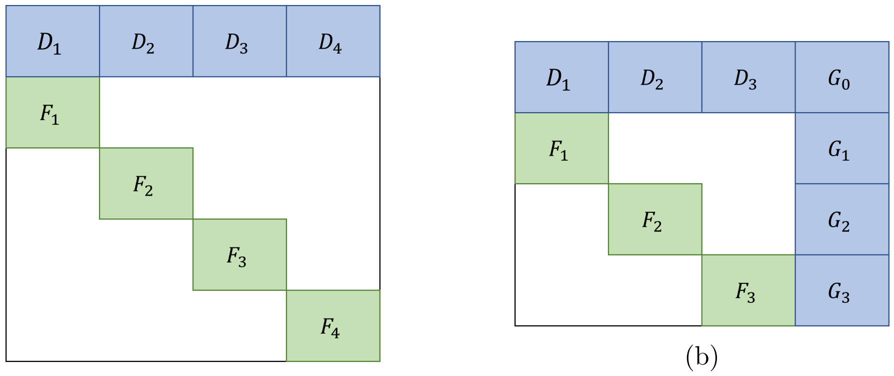
A relatively general method to solve large scale MIPs involves exploiting the known structure of the problem to decompose it into multiple sub-problems representing its several approximately-independent components. An additional advantage is that such a decomposition also makes it possible to solve in parallel. Here we will look at one such algorithm for solving large-scale integer programming, namely the Dantzig-Wolfe (DW) Decomposition [2]. This algorithm is closely linked with another approach called delayed column generation [7]. DW decomposition is typically used for problems where the constraint matrix has a block structure with linking constraints, as depicted in Fig.(1a). It can also be modified to accommodate linking variables in which case the constraint matrix has an arrowhead structure as shown in Fig.(1b).

Consider a linear programming problem of the form,

$$\begin{aligned} \text{minimize } & c_1' x_1 + c_2' x_2 \\ \text{subject to } & D_1 x_1 + D_2 x_2 = b_0 \\ & F_1 x_1 = b_1 \\ & F_2 x_2 = b_2 \end{aligned}$$

For $i = 1, 2$ define, $P_i = \{x_i \geq 0 | F_i x_i = b_i\}$. Then, broadly, DW decomposition works by reformulating the problem with sub-problems that optimize over P_1 and P_2 and the following master problem linking them.

$$\begin{aligned} \text{minimize } & c_1' x_1 + c_2' x_2 \\ \text{subject to } & D_1 x_1 + D_2 x_2 = b_0 \end{aligned}$$



(a) (b)

Figure 1: Block structure of constraint matrix for DW Decomposition

A major short-coming that prevents Dantzig-Wolfe Decomposition and column generation from being used in general MIP solvers is that it requires problem specific knowledge or user expertise to provide the decomposition as input to the structure. In this work, we propose a methodology to detect such structure automatically without any problem specific information.

In [1] the authors are the first to present a method for automatic Dantzig-Wolfe reformulation. They express the constraint matrix as Row-Net or Row-Column-Net hypergraphs. In the first case, each variable represents a node and hyper-edges represents constraints connecting variables having non-zero coefficients in them. In case of Row-Column-Net hypergraphs, each non-zero coefficient in the constraint matrix is a node and the hyper-edges represent either variables connecting non-zero elements corresponding to them or constraints connecting the coefficients present in them. They then solve the minimum cut hypergraph partitioning problem to obtain the Dantzig-Wolfe Reformulation (DWR). They show that their algorithm is able to detect arrowhead even in problems where no implicit structure is know. They demonstrate their methodology on MIPLIB [3] instances which is a standard benchmark for MIPs. Moreover, they also suggest pseudo-metrics such as relative border area that can help evaluate the quality of the decomposition apriori before deploying delayed constraint generation algorithm.

In this work, we would like to explore the effectiveness of spectral clustering[8] in obtaining the DWR. Since their methodology is based on partitioning of a hypergraph, we believe that spectral clustering is a naturally suited for this application and potentially produce high quality decompositions quickly.

2 Algorithmic Details

We propose to design a spectral clustering based algorithm to automatically reformulate a given MIP by simply rearranging the order of variables and constraints such that the resulting constraint matrix takes the block diagonal structure.

2.1 Graphical representation

Inspired by the graphical representation of constraint matrices in [1], we construct similar matrices by essentially replacing the hyperedges by cliques. However, cliques, in general, do not capture the same information as the hyperedges. Consider the case of a Row-Net graph where two variables are jointly present in multiple constraints. In case of an unweighted graph, they would be connected by an edge which is indistinguishable from the case where they were jointly present in only one constraint. We thus introduce edge weights to counteract this effect which greatly improves performance as compared to the unweighted graph. We now describe the details of the graphical representation and choice of weights.

Row-Net Graph The Row-Net Graph representation considers the problem of clustering variables into groups that define the block. The nodes of the graph represent the variables and two nodes are connected by an edge if both of the corresponding variables have non-zero coefficients in some constraint. The edge weights are computed by first assigning an importance score to every constraint and then adding the importance of every constraint the variables are jointly present in. The choice of importance score for constraints that led to the best performance was when the score was inversely proportional to the square of count of non-zero coefficients in the constraint. That is for the i-th constraint,

$$\text{score}(i) = \frac{1}{\left(\sum_{j=1}^{n} \mathbb{1}_{a_{ij} \neq 0}\right)^2}.$$

The intuition behind this is that we give lower importance to dense constraints which are more likely to be classified as a linking constraint and aggressively try to group together variables in sparse constraints. A constraint is assigned to a block if the non-zero coefficients are covered by the variables in the block. If a constraints cannot be covered by variables in any one block, it is a linking constraint.

Row-Column-Net Graph The Row-Column-Net Graph representation considers the problem of clustering non-zero coefficients in the constraint matrix. Let $Ax \leq b$ be the constraints. Than for each $a_{ij} \neq 0$, we add a node in the graph and we connect the nodes corresponding to a_{ij} and a_{kl} is either $i = k$ or $j = l$; that is if they are in the same constraint or if they are coefficients of the same variable. Similar to the Row-Net Graph, we compute importance scores for each constraint and each variable as the squared inverse of the number of non-zero elements in the row or column respectively. Coefficients in the same row are connected by edges with weight equal to the score of that constraint and symmetrically, coefficients in the same column are connected by edges with weight equal to the score of that variable. After clustering, variables and constraints are assigned to the cluster if they are covered by the coefficients in the cluster, and those spanning across different clusters are linking variables and constraints respectively.

2.2 Graph Laplacians and Spectral Clustering Algorithms

Graph Laplacians matrices are the main tool behind spectral clustering. The topic of spectral graph theory covers them in depth. For our problem we look at three different types of Graph Laplacians as mentioned in [6]. We need to define some required notations first. Note that the graphs described in 2.1 are undirected and weighted. By construction, an edge between any two vertices, v_i and v_j carries a non-negative weight $w_{ij} \geq 0$. The adjacency matrix, W, is defined as:

$$W = (w_{ij})_{i,j=1,\ldots,n} \tag{1}$$

Since the graph is undirected, $w_{ij} = w_{ji}\ \forall i, j$ and W is symmetric. Furthermore, let's define degree of a vertex $v_i \in V$, as $d_i = \sum_{j=1} w_{ij}$. Then the degree matrix, D, is defined as the diagonal matrix,

$$D = \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix} \tag{2}$$

- **Unnormalized Graph Laplacian**:
  This is the graph Laplacian covered in class. The unnormalized Graph Laplacian is defined as:
  $$L = D - W \tag{3}$$
  We implement the unnormalized spectral clustering algorithm using this graph Laplacian. To construct k clusters, we compute eigen-vectors of L corresponding to the k smallest eigen-values. Using these as features, k-means algorithm is used to find the clusters.

- **Symmetric Normalized Graph Laplacian**:
  $$L_{sym} := D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2} \tag{4}$$
  Using this, we implement the normalized spectral clustering algorithm in [4]. The algorithm follows similarly as the unnormalized spectral clustering except that the feature vector for each row is normalized by division by the euclidean norm of the row.

- **Random-Walk Normalized Graph Laplacian**
  $$L_{rw} := D^{-1} L = I - D^{-1} W \tag{5}$$
  This definition of Laplacian used in the the unnormalized spectral clustering algorithm leads to the normalized spectral clustering algorithm in [5]

Since W and D are symmetric, all the three definitions of the Laplacians are also symmetric. Moreoever, they are positive semi-definite. Consequently, they have n non-negative real-valued eigen-values. The graph Laplacian is used to find the number of connected components in the original graph G. In-fact, The multiplicity of the eigenvalue 0 corresponds to the number of connected components in the original graph.

Unnormalized spectral clustering algorithm solves the relaxation of the minimum cut problem on the graph where the objective function is normalized by the size of each cluster. On the other hand, both of the normalized spectral clustering algorithms solve the problem minumum cut problem where the objective is normalized by the total weight of edges in each cluster. Refer to [6] for further details.

3 Results

We demonstrate our algorithm on 8 of the MIPLIB instances used by [1] for a fair comparison. For each instance, we run the algorithm for both graph types, all 3 choices of Laplacians and number of clusters, $k \in \{2, \ldots, 10\}$.

We evaluate the quality of the resulting decompositions using relative border area, proposed in the same paper. The relative border area is defined as the proportion of total area of the constraint matrix occupied by the linking variables and constraints. Let m, m_l, n, n_l be the number of constraints, linking constraints, variables and linking variables respectively. Then the border area is computed as,

$$\text{Relative Border Area} = \frac{(m_l \times n) + (m \times n_l) - (m_l \times n_l)}{(m \times n)} \tag{6}$$

Practically, for solving Dantzig Wolfe reformulations, it is desirable to have blocks of similar sizes. Decompositions with similar sized blocks also lend themselves to efficient parallelization. However, spectral decomposition does not give any guarantees on the variance in the size of blocks. Thus, we also evaluate decompositions on the following metric,

$$\text{Block Size Ratio} = \frac{\max_{1 \le i \le k} m_i \times n_i}{\min_{1 \le i \le k} m_i \times n_i} \tag{7}$$

where m_i, n_i denote the number of constraints and variables in block i.

For our first set of results, we analyse compare the quality of the decompositions for different Graph Laplacians based on the balance of size of blocks generated. Table: 1 shows the Block Size Ratios for the resulting decomposition across graph types and number of clusters that give the lowest Relative Border Area. Random-walk Laplacian performs terribly in this regard. It often generates decompositions where a cluster consists of a singe variable and constraint resulting in the unusually high ratio for some problems. Overall, Symmetric Laplacian gives the most balanced clusters and should be the algorithm of choice when similarity in size of the resulting blocks is very important.

Problem	**Unnormalized**	**Random-Walk**	**Symmetric**
aflow30a	1.44	76.82	1.12
fiber	2.19	1722.94	1.82
gesa2	1.24	1.24	1.24
glass4	3.68	3.45	1.01
harp2	1.06	163836	4.76
noswot	16.55	18.25	2.19
rout	2.26	2.26	2.26
timtab1	13249	538.04	2.37

Table 1: Block Size Ratio for smallest Border Area

From all the decompositions obtained for a problem, we select the "Best" based on the following criteria -

- We first eliminate decompositions where the size of blocks vary greatly. In particular, we filter out the results where the Block Size Ratio exceeds 5.
- Of the resulting decompositions, we select the one which has the smallest Relative Border Area.

We present the details regarding the best decomposition thus obtained in Table 2, the visualizations for the same are presented in Figures 2-9. Observe that most of these resulting decompositions are derived using the Normalized Symmetric Laplacian and row-col-net graphs. The number of clusters varies in the range of 2 to 10. We also compare the relative border area against that of the

Problem	Laplacian	Graph	K	Linking Cons	Linking Vars	Block Size Ratio	Rel. Border Area	[1] Rel. Border Area	Rel. Border Area from $\hat{K}$
aflow30a	symmetric	row-col-net	10	7%	1%	3.7	7.5%	5.8%	8.1%
fiber	default	row-net	2	6%	0%	2.8	6.1%	6.1%	12.7%
gesa2	default	row-col-net	2	0%	2%	1.2	2.1%	2.1%	5.1%
glass4	symmetric	row-col-net	4	0%	5%	4.3	5.0%	3.7%	–
harp2	symmetric	row-col-net	9	35%	0%	4.8	34.8%	34.8%	34.8%
noswot	symmetric	row-col-net	2	4%	0%	2.2	3.8%	4.9%	4.6%
rout	symmetric	row-col-net	5	5%	0%	1.0	5.5%	5.5%	5.5%
timtab1	symmetric	row-col-net	3	5%	3%	2.8	7.9%	3.3%	11.6%

Table 2: Best selection results

best decomposition generated in [1]. While our results are comparable to those in [1] on most of the problems, we do much worse on timetab1.

In spite of the fact that our spectral clustering does not out-perform the hypergraph partitioning algorithm of [1], it possesses an advantage. In case of the hypergraph partitioning algorithm, there is no way to a priori predict the number of clusters which will result in a good decomposition. On the other hand, in case of spectral clustering, looking at the eigen-values of the Laplacian can help choose a promising value of k. For our last experiment, we limit our choices to the value of k that maximizes the increase in eigen-values. Let $\lambda_1, \ldots, \lambda_n$ be the eigen-values in ascending order. Then, given we want the number of clusters between 2 and 10, we define

$$\hat{K} = \arg \max_{2 \leq i \leq 10} \{\lambda_{i+1} - \lambda i\} \tag{8}$$

We then select the best decomposition using the same criteria described above, but limiting the value of $k = \hat{K}$. The results are presented in the last column of Table 2. As expected, these are worse in terms of relative border area as compared to the earlier results. This technique, however, saves the computational effort of generating and evaluating decompositions for every value of k which may be high for large sized problems. Moreover, this can be used to predict if at all good decompositions exist for a given problem. When applied to a problem like harp2 for instance, this can lead to an early conclusion that decomposition based methods are likely to fail to solve the problem.

4 Conclusion

We have shown that spectral clustering can be applied for identifying block structure in constraint matrices without any user input. We evaluate three different graph Laplacians in terms of their ability to produce balanced clusters of similar sizes. From our experiments, Symmetric Normalized Laplacian produces the best results. We also compare our results to that of [1] that use partitioning on hypergraphs to generate clusters. While our results are comparable to theirs for most problems, our algorithm under-performs on some. However, spectral clustering has an added advantage that the eigenvalues generated in the process can be used to predict the number of clusters that would result in good decompositions. On the other hand, there is no way to a choose a promising number of clusters in case of the hypergraph partitioning algorithm without expert knowledge regarding the problem structure. This crucial advantage makes spectral clustering a promising algorithm for this application and further research in this direction may improve the algorithm such that it outperforms existing methods.

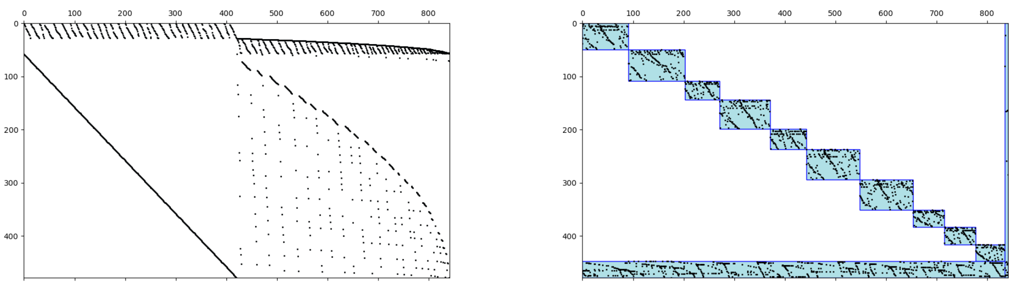


Figure 2: Reformulation for aflow30

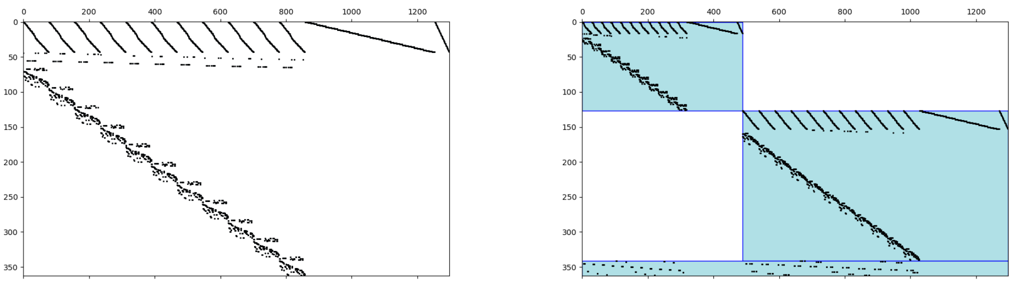


Figure 3: Reformulation for fiber

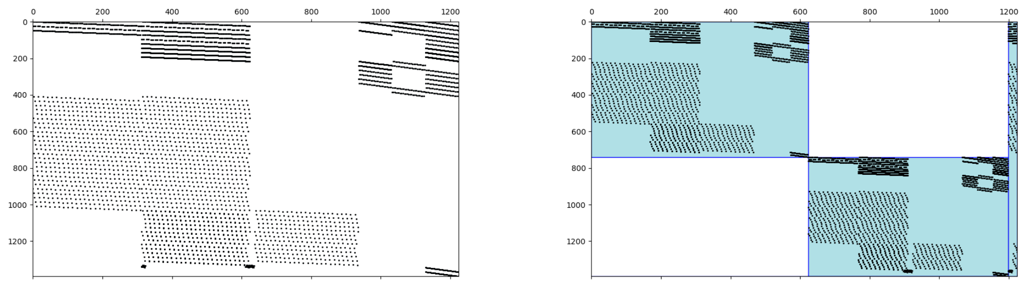


Figure 4: Reformulation for gesa2

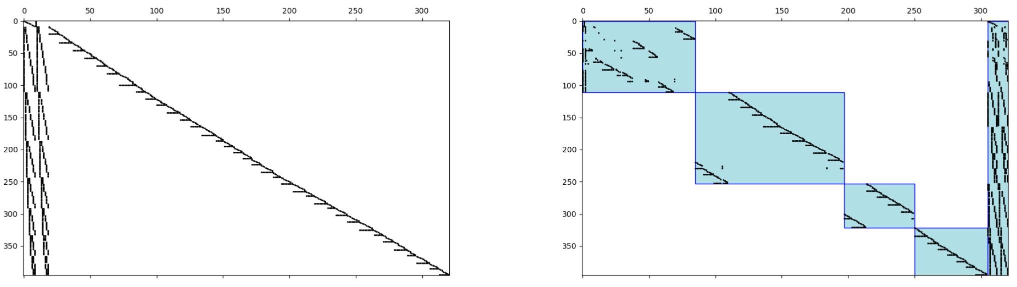


Figure 5: Reformulation for glass4

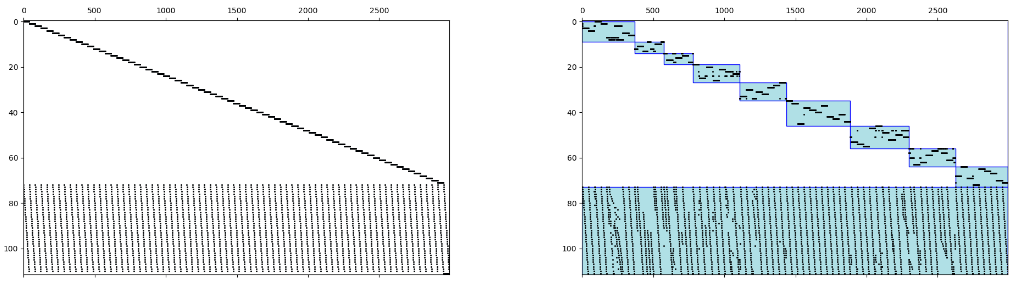


Figure 6: Reformulation for harp2

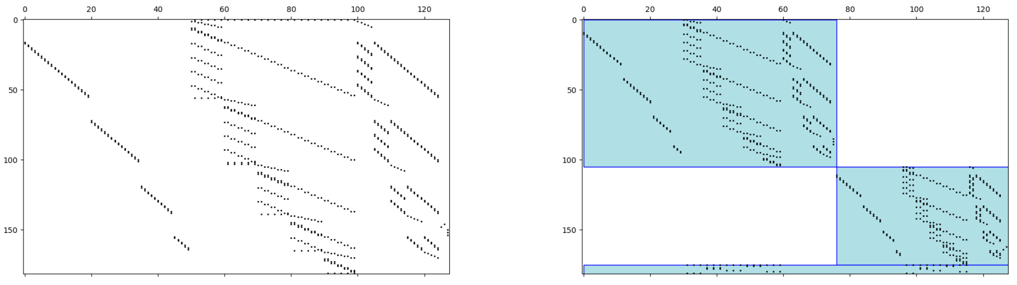


Figure 7: Reformulation for noswot

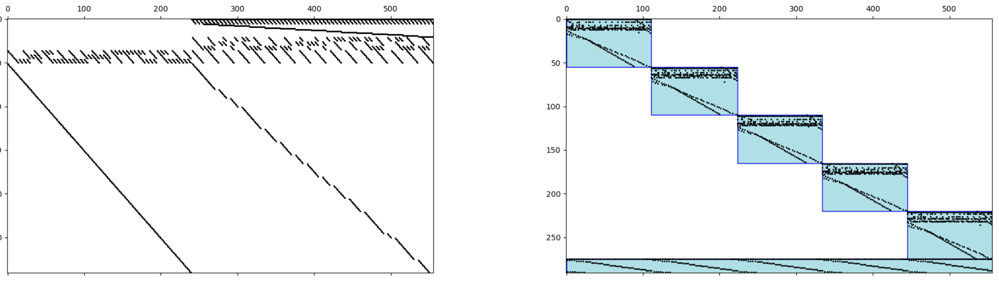


Figure 8: Reformulation for rout

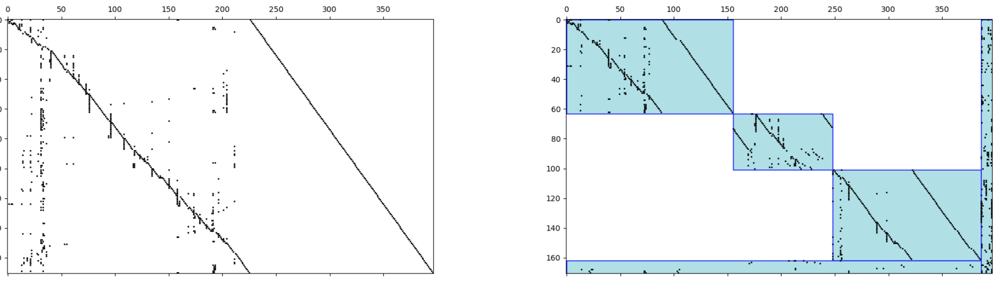


Figure 9: Reformulation for timtab1

References

[1] Martin Bergner et al. “Automatic Dantzig–Wolfe reformulation of mixed integer programs”. In: *Mathematical Programming* 149.1 (2015), pp. 391–424.

[2] George B Dantzig and Philip Wolfe. “Decomposition principle for linear programs”. In: *Operations research* 8.1 (1960), pp. 101–111.

[3] Ambros Gleixner et al. “MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library”. In: *Mathematical Programming Computation* (2021). DOI: 10.1007/s12532-020-00194-3. URL: https://doi.org/10.1007/s12532-020-00194-3.

[4] Andrew Ng, Michael Jordan, and Yair Weiss. “On spectral clustering: Analysis and an algorithm”. In: *Advances in neural information processing systems* 14 (2001).

[5] Jianbo Shi and Jitendra Malik. “Normalized cuts and image segmentation”. In: *IEEE Transactions on pattern analysis and machine intelligence* 22.8 (2000), pp. 888–905.

[6] Ulrike Von Luxburg. “A tutorial on spectral clustering”. In: *Statistics and computing* 17.4 (2007), pp. 395–416.

[7] *Wikipedia: Delayed Column Generation*. URL: https://en.wikipedia.org/wiki/Column_generation.

[8] *Wikipedia: Spectral Clustering*. URL: https://en.wikipedia.org/wiki/Spectral_clustering.