Noname manuscript No.
(will be inserted by the editor)

A Mixed Integer Program for Partitioning Graphs with Supply and Demand Emphasizing Sparse Graphs

Raka Jovanovic · Stefan Voß

Received: date / Accepted: date

Abstract The focus of this paper is on finding optimal solutions for the problem of maximal partitioning of graphs with supply and demand (MPGSD) for arbitrary graphs. A mixed integer programming (MIP) model is developed for the problem of interest. We also present some specific constraints that can be used in the case of tree graphs. With the goal of lowering the computational cost for solving the underlying model, a preprocessing stage is included. It is used to produce additional constraints based on shortest paths in the graph. With the aim of exploring the effectiveness of the proposed MIP formulation we have performed computational experiments for general graphs and trees. The main objective of the tests is to observe the properties and sizes of supply/demand graphs that can be solved to optimality using the proposed approach in reasonable time. The conducted computational experiments have shown that the proposed method is especially suitable for sparse graphs.

Keywords Mixed Integer Programming · Graph Partitioning · Demand Vertex · Supply Vertex

1 Introduction

Up to now literature on the problem of maximal partitioning of graphs with supply/demand (MPGSD) has mainly focused on theoretical properties. This is well documented in the following works: [1–5]. Moreover, recent research has considered heuristics and metaheuristics for the problem, as can be found in [6–8]. Another

Raka Jovanovic
Qatar Environment and Energy Research Institute (QEERI), Hamad bin Khalifa University, PO Box 5825, Doha, Qatar, E-mail: rjovanovic@qf.org.qa

Stefan Voß
Institute of Information Systems, University of Hamburg, Von-Melle-Park 5, 20146 Hamburg, Germany, E-mail: stefan.voss@uni-hamburg.de
and Escuela de Ingenieria Industrial, Pontificia Universidad Católica de Valparaíso, Chile

direction of research on the MPGSD is exploring different variations of the original problem. Examples are a parametric version [9–11], supply capacity limitations[12] and its extension by including edge capacities and flows [3,13,14]. To the best of our knowledge, one of the missing links in literature, up to now, is some mathematical programming formulation.

In this paper we focus on the development of such a formulation for the MPGSD suitable for general graphs. It is important to mention that a major focus on investigating the MPGSD was for special graph classes like trees, as it is well reflected in more theoretical elaborations like [1–3] but also in the numerical studies on heuristics like [6–8]. Moreover, the tree structure of a solution allows to define specific constraints for the MIP, reflecting certain properties of this type of graphs. The effect of using such constraints and their effectiveness is also analyzed in this article.

One of the most important contributions of MIP formulations for the problem of interest is the possibility of finding confirmed optimal solutions for arbitrary supply/demand graphs. The availability of problem instances with known optimal solutions is essential for effective development of advanced metaheuristics. In the existing literature, such problem instances exist but they are acquired using a randomized method for generating problem instances with known optimal solutions [6–8]. Although such approaches attempt to provide purely random graphs (general or of a specific type) they often have some unintentional correlations.

With the goal of evaluating the proposed MIP formulations we have implemented the MIP in CPLEX and applied it to problem instances from literature. Our computational experiments show that it is possible to find optimal solutions for graphs of significant size.

2 Maximal Partitioning of Graphs With Supply/Demand

The MPGSD is defined for an undirected graph $G = (V, E)$ with a set of nodes V and a set of edges E (an illustration of the MPGSD is given in Figure 1). The set of nodes V is split into two disjunct subsets V_s and V_d. Each node $u \in V_s$ is called supply vertex and has a corresponding positive integer value $sup(u)$. Elements of the second subset $v \in V_d$ are called demand vertices and have a corresponding positive integer value $sup(v)$. The goal is to find a partitioning $\Pi = \{S_1, S_2, .., S_n\}$ of the graph G that satisfies the following constraints. All the subgraphs in Π must be connected and contain only a single distinct supply node. As a result we have $|V_s| = n$. Each of the S_i must have a supply greater or equal to its total demand. Each demand vertex can be an element of only one subgraph, or in other words, it can only receive 'power' from one supply vertex through the edges of G. The goal is to maximize the fulfillment of demands, or more precisely to maximize the following sum.

$$\sum_{S \in \Pi} \sum_{v \in S \cap V_d} dem(v) \tag{1}$$

while the following constraints are satisfied for all $S_i \in \Pi$

$$\sum_{v \in S_i \cap V_s} sup(v) \geq \sum_{v \in S_i \cap V_d} dem(v) \tag{2}$$

$$S_i \cap S_j = \emptyset \quad , i \neq j \tag{3}$$

$$S_i \quad is \quad connected \tag{4}$$

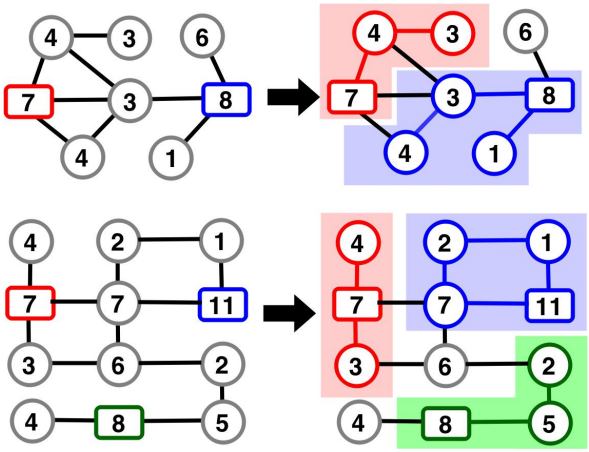

Fig. 1 Examples of problem instances for the MPGSD. Square nodes represent supply nodes and circles demand nodes. Numbers within the nodes correspond to supply and demand values, respectively. The right side shows solutions, where the same color (or connected shaded set) of nodes indicates they are a part of the same subgraph.

3 Mixed Integer Program

In this section we present a MIP formulation for the MPGSD. Initially a basic model is presented, which is then adapted to exploit the sparsity of a graph. Further, we include a simple preprocessing stage which is used to create a new set of variables for the problem. As will be seen, the preprocessing stage manages to significantly decrease the time needed for solving the MIP formulation.

3.1 Supply/Demand Related Constraints

With the goal of a more clear presentation of the MIP formulation for the MPGSD we first present the input parameters, decision variables and the corresponding constraints dedicated to the relation between supply and demand nodes. Later, we introduce the constraints and variables related to the connectivity of individual subgraphs. We start with the input parameters, which describe the supply demand graph.

- n is an integer parameter equal to the total number of nodes in the graph.
- $\hat{s}_i$ $(i = 1..n)$ is a positive integer parameter corresponding to the supply value of node i defined as $\hat{s}_i = sup(i)$ for $i \in V_s$, and $\hat{s}_i = 0$ for $i \notin V_s$.
- $\hat{d}_i$ $(i = 1..n)$ is a positive integer parameter corresponding to the demand value of node i defined as $\hat{d}_i = dem(i)$ for $i \notin V_s$, and $\hat{d}_i = 0$ for $i \in V_s$.
- s_i $(i = 1..n)$ is a binary parameter giving the information if node i is a supply node, or more formally $s_i \equiv (i \in V_s)$ (i.e., $s_i = 1$ if $i \in V_s$ and $s_i = 0$ if $i \notin V_s$).

The next step is defining the necessary decision variables as follows: $x_{ij} \equiv (i \in \hat{S}_j)$ $(i, j = 1..n)$. x_{ij} is a binary variable indicating if node i is an element of the subgraph $\hat{S}_j$ represented by node j. Note that $\hat{S}_j$ is not the same as the subgraphs S_i given in the definition of MPGSD. By using $\hat{S}_j$ we have a simpler notation in the MIP formulation since we can index supply and demand nodes jointly. The unnecessary additional variables corresponding to subgraphs represented by demand nodes, are easily removed by the automated preprocessing stage based on the model constraints. Such a stage is a part of most standard MIP solvers like CPLEX. Now we can specify the MIP formulation for the MPGSD without the connectivity constraint as follows:

$$\text{Maximize} \quad \sum_{i,j=1..n} \hat{d}_i x_{ij} \tag{5}$$

$$\text{subject to}$$

$$x_{ii} = s_i \qquad (i = 1..n) \tag{6}$$

$$x_{ij} - 1 + s_i \leq 0 \qquad (i, j = 1..n : i \neq j) \tag{7}$$

$$x_{ij} \leq s_j \qquad (i, j = 1..n) \tag{8}$$

$$\sum_{j=1..n} x_{ij} \leq 1 \qquad (i = 1..n) \tag{9}$$

$$\sum_{i=1..n} \hat{s}_i x_{ij} - \sum_{i=1..n} \hat{d}_i x_{ij} \geq 0 \qquad (j = 1..n) \tag{10}$$

$$x_{ij} \in \{0, 1\} \qquad (i, j = 1..n) \tag{11}$$

The goal of the proposed formulation is to maximize the sum given in (5) indicating the covered demand. In it $x_{ij} = 1$ states that node i is contained in one of the subgraphs. The constraints given in (6) and (7) are used to make sure a supply node j is only a part of subgraph $\hat{S}_j$. In this way symmetries are also avoided. Constraints (8) are used to fix the value of the unnecessary variables connected with subgraphs having a demand node as a representative to 0, or in other words makes sure that subgraphs $\hat{S}_i$ with a demand representative node have no nodes inside. The fact that a demand node can only be in one subgraph is ensured by (9). Constraint (10) states that the supply must be greater or equal to the total demand in a subgraph.

3.2 Subgraph Connectivity Constraints

To fully specify the MIP model for the MPGSD we also need to guarantee that each of the subgraphs is connected. To do so we need to have input parameters corresponding

to the graph's edges. This can be done using the standard adjacency matrix as follows:

$$a_{ij} \equiv ((i,j) \in E) \qquad (i,j = 1..n) \tag{12}$$

In practice – unless the graph G is very dense – it is advantageous to define the MIP using the set of edges E. In the proposed formulation we use this type of definitions, or in other words the set of edges E will be an input parameter. For practical purposes, although G is an undirected graph, it is more convenient to use directed edges in the MIP formulation, or in other words for each $(i,j) \in E$ we will add two edges (i,j) and (j,i) to the edge set. To be able to verify the connectivity of a subgraph, we will define auxiliary binary decision variables e_{ijk} that state if edge (i,j) is a part of subgraph $\hat{S}_k$. We will say that edge (i,j) is in $\hat{S}_k$ if $i,j \in \hat{S}_k$. Formally,

$$e_{ijk} \equiv (i \in \hat{S}_k) \wedge (j \in \hat{S}_k) \wedge (s_k = 1) \qquad ((i,j) \in E, \quad k = 1..n) \tag{13}$$

The values of e_{ijk} can be specified using the following linear constraints:

$$e_{ijk} \leq s_k \qquad ((i,j) \in E, \quad k = 1..n) \tag{14}$$

$$e_{ijk} \leq x_{ik} \qquad ((i,j) \in E, \quad k = 1..n) \tag{15}$$

$$e_{ijk} \leq x_{jk} \qquad ((i,j) \in E, \quad k = 1..n) \tag{16}$$

$$e_{ijk} \geq x_{ik} + x_{jk} + s_k - 2 \qquad ((i,j) \in E, \quad k = 1..n) \tag{17}$$

Using the newly defined decision variables e_{ijk} we can define additional constraints guaranteeing the connectivity within subgraphs. Due to the significant research dedicated to solving the MPGSD for trees we shall first present constraints that are specific for this type of graphs. Such constraints can be defined for each of the subgraphs by exploiting the fact that a connected subgraph of a tree is also a tree:

$$\sum_{i=1..n} x_{ik} - \frac{1}{2} \sum_{(i,j) \in E} e_{ijk} = s_k \qquad (k = 1..n) \tag{18}$$

Eq. (18) is based on the fact that in any tree with a vertex set V and edge set E the relation $|V| = |E| + 1$ is satisfied. In the same equation s_k is used instead of 1, so the constraint would also be applicable for empty subgraphs related to demand nodes. The factor $\frac{1}{2}$ is added since each edge is counted twice (once for each direction).

In case of general graphs defining connectivity constraints is more complex. One of the standard methods of ensuring graph connectivity in a MIP formulation is by introducing flow variables [15]. In the proposed mathematical model for MPGSD this concept will be adopted. In case of the problem of interest, this type of formulation needs to be extended to the setting of multiple subgraphs, whose structures are dependent on the decision variable. The general idea is to set the supply node in each of the subgraphs as a source of the flow, and check if the flow has reached all the nodes in the subgraph. To do this a flow variable f_{ijk} is defined for each edge $(i,j) \in E$ and subgraph $\hat{S}_k$. All the flow variables are positive integer numbers. The value of f_{ijk} indicates how much flow is moved from node i to node j in subgraph $\hat{S}_k$. Using the new

decision variables the connectivity can be guaranteed using the following constraints:

$$f_{ijk} - Me_{ijk} \leq 0 \qquad ((i,j) \in E), \quad k = 1..n) \tag{19}$$

$$f_{ijj} = 0 \qquad ((i,j) \in E) : s_j = 1) \tag{20}$$

$$\sum_{i=1..n} x_{ik} - \sum_{i=1..n} f_{kik} = 1 \qquad (k = 1..n : s_k = 1) \tag{21}$$

$$\sum_{i=1..n} f_{ijk} - \sum_{i=1..n} f_{jik} \geq x_{jk} \qquad (j,k = 1..n : s_k = 1 \wedge s_j = 0) \tag{22}$$

$$\sum_{i=1..n} f_{ijk} \geq x_{jk} \qquad (j,k = 1..n : s_k = 1) \tag{23}$$

Constraints (19) guarantee that flow is only allowed through edges that are inside subgraph $\hat{S}_k$ (M represents a sufficiently large number). Eq. (20) and (21) specify the flow properties of the supply node. To be specific, (20) states that there is no input flow entering the source node, and (21) states that the total flow out of node j is equal to the number of demand nodes in the subgraph ($|\hat{S}_j| - 1$). Constraints (22) are used to ensure that each node that receives flow consumes at least one unit of it. The final constraint states that each node in a subgraph has an input flow at least equal to 1.

3.3 Preprocessing

In the previous two subsections a complete MIP for the MPGSD is given. To be exact, the definition consists in the maximization of (5), based on the decision variables x_{ij}, e_{ijk} and f_{ijk}, corresponding input parameters and the constraints given in (6)-(10), (14)-(17), (19)-(23). In general a MIP formulation having the same set of constraints may be more effective in case it has a lower number of decision variables. As previously mentioned, in the presented formulation many of the decision variables x_{ij} can be removed, e.g. all those representing subgraphs having demand nodes as representatives. To be exact, instead of defining variables x_{ij} over the set $\{1,..,n\} \times \{1,..,n\}$, they can be defined over the following set

$$X = \{(i,j) \mid (i \in V) \wedge (j \in V_s)\} \tag{24}$$

The set X can be also used to create a restriction of the edge set for defining variables e_{ijk} and f_{ijk}. In practice this change produces a less significant decrease in computational time, needed for solving the MPGSD in a standard MIP solver like CPLEX, than expected. This is due to the fact that such variables are automatically removed by software of this type in the preprocessing stage.

In this section a new set of constraints is defined that manages to remove a large number of decision variables. More precisely, the decision variables x_{ij} corresponding to supply/demand node pairs for which it is impossible to be a part of any feasible solution are removed. The idea is that a demand node i cannot be covered by a supply node j because it is "too far from it." To be exact, the demand node i cannot be in a subgraph with a supply node j if the sum of all demands on the shortest path between

i and j is greater than $sup(j)$. Using this idea we can add an additional binary input parameter p_{ij} $(i, j = 1..n)$ precalculated in the following way:

$$dist(i, j) = \sum_{k \in SV(i,j), k \neq j} dem'(k) \qquad (i \in V, j \in V_s) \tag{25}$$

$$p_{ij} \equiv (dist(i, j) \leq sup(j)) \tag{26}$$

In Eq. (25), function $dem'(i)$ is an extended version of the dem in which $dem'(j) = M$, where M is a sufficiently large number, for $j \in V_s$. In the same equation $SV(i, j)$ is the set of nodes on the path with a minimal weight connecting i and j. The values of the $dist(i, j)$, and as a consequence the values of p_{ij}, can be calculated using Dijkstra's algorithm. The MIP model can now be extended by the following constraint:

$$x_{ij} \leq p_{ij} \qquad (i, j = 1..n) \tag{27}$$

In practical implementations the additional constraint will not be used but a restricted set of decision variables x_{ij}, which is defined over the following set.

$$X' = \{(i, j) \mid ((i, j) \in X) \wedge (dist(i, j) \leq sup(j))\} \tag{28}$$

Our tests have shown that this preprocessing stage has a computational cost neglectable compared to solving the MIP. The set X' can also be used to restrict the number of variables of types e_{ijk} and f_{ijk}. That is, instead of defining these variables over the set $\{1, .., n\} \times \{1, .., n\} \times \{1, .., n\}$ they will be defined over the following set:

$$T = \{(i, j, k) \mid ((i, j) \in E) \wedge ((i, k) \in X') \wedge ((j, k) \in X')\} \tag{29}$$

4 Results

In this section, we present the results of the performed computational experiments based on the proposed MIP formulation for the MPGSD. The model has been implemented using IBM ILOG CPLEX Optimization Studio Version: 12.6.1.0, and executed using the default solver settings. The calculations have been done on a machine with 2 Intel(R) Xenon(R) CPU 3.30 GHz, 96GB of DDR3-1333 RAM, running on Microsoft Windows 7 Professional 64-bit. In all the implemented models we have used a sparse representation of the decision variables x_{ij}, e_{ijk} and f_{ijk}, as presented in the previous section. The tests have been performed on the same data sets as in [8,6] (which can be downloaded from http://mail.ipb.ac.rs/~rakaj/home/graphsd.htm). The test data consists of 24 different graph sizes having 2-100 supply nodes and 6-2000 demand nodes, and for each size there are 40 different problem instances. There are two such data sets, one for general graphs and one for trees. Note that the used general graphs are relatively sparse.

The goal of the conducted tests is to explore the limits, in the sense of graph properties and size, of the proposed MIP formulations for finding optimal solutions. Further, we also aim to evaluate the effectiveness of the previously presented preprocessing procedure and the tree-specific connectivity constraints. The results of our computational experiments can be seen in Tables 1 and 2 for trees and general graphs,

respectively. The notation for the columns is as follows: GEN - flow-based connectivity constraints, TRE-P - tree-specific connectivity constraints combined with preprocessing, GEN-P - GEN combined with preprocessing. All times are in milliseconds.

Table 1 Comparison of computational times for the different MIP formulations for trees.

Demand	*Average*			*Median*			*Maximal*		
nodes	GEN	TRE-P	GEN-P	GEN	TRE-P	GEN-P	GEN	TRE-P	GEN-P
2 Supply nodes									
6	15	16	19	10	10	10	78	51	90
10	25	34	29	20	20	20	74	113	130
20	53	83	57	30	38	30	273	282	274
40	84	160	138	68	103	70	343	424	350
5 Supply nodes									
15	30	29	21	21	20	20	244	240	62
25	50	126	87	40	71	31	270	342	318
50	333	260	215	348	292	202	544	637	471
100	492	562	437	497	493	434	786	1232	1091
10 Supply nodes									
30	62	68	44	50	31	30	271	302	272
50	202	225	153	141	264	93	462	439	377
100	792	793	490	717	621	503	1549	1956	852
200	1958	3993	1438	1790	2778	1187	5421	35186	5910
25 Supply nodes									
75	317	190	100	263	261	60	563	394	302
125	1126	783	454	1050	606	423	2113	2621	796
250	6792	11509	2129	5394	6556	1731	18185	62775	10654
500	-	-	15252	-	-	9279	-	-	88598
50 Supply nodes									
150	897	348	217	889	351	152	1472	1079	400
250	4400	2008	882	4280	1536	749	9335	8034	1805
500	70907	-	9339	57412	-	5086	346809	-	72544
1000	-	-	-	-	-	-	-	-	-
100 Supply nodes									
300	3562	515	342	3447	490	384	5498	877	544
500	30960	6630	1852	28472	4678	1728	89122	26093	4204
1000	-	-	65634	-	-	38839	-	-	305700
2000	-	-	-	-	-	-	-	-	-

In case of both graph types we observe the average, median and maximal time for finding the optimal solution for all the problem instances in one graph size. In the results we have only considered graph sizes for which optimal solutions have been found for all the test instances within a time limit of 600 seconds.

In case of tree graphs we have compared the performance of MIP formulations based on the flow approach to graph connectivity with and without the use of preprocessing. In Table 1 we have also included results when using the tree-specific con-

Table 2 Comparison of computational times for the different MIP formulations for general graphs.

Demand Nodes	*Average* GEN	*Average* GEN-P	*Median* GEN	*Median* GEN-P	*Maximal* GEN	*Maximal* GEN-P
2 Supply nodes						
6	57	38	30	20	260	250
10	188	95	220	50	320	338
20	329	234	376	192	539	634
40	543	504	538	526	786	731
5 Supply nodes						
15	187	111	132	66	383	375
25	1547	1141	919	702	9286	6399
50	8573	6925	5832	5153	29474	27048
100	8401	7740	5889	5413	24814	26972
10 Supply nodes						
30	632	341	588	360	1203	643
50	-	39439	-	26114	-	164163
100	-	-	-	-	-	-
200	-	-	-	-	-	-
25 Supply nodes						
75	68667	3533	37314	1402	527270	14985
125	-	-	-	-	-	-
250	-	-	-	-	-	-
500	-	-	-	-	-	-
50 Supply nodes						
150	-	31452	-	16538	-	260955
250	-	-	-	-	-	-
500	-	-	-	-	-	-
1000	-	-	-	-	-	-
100 Supply nodes						
300	-	98181.8	-	61098	-	540174
500	-	-	-	-	-	-
1000	-	-	-	-	-	-
2000	-	-	-	-	-	-

nectivity constraint (instead of flow-based) combined with the preprocessing stage. Note that in the results given in Table 1, the presented times only correspond to the time needed to solve the MIP. To be more precise the preprocessing time is excluded, since it was neglectable compared to solving the MIP. As it can be seen from the results in Table 1, the MIP formulation is very suitable for solving MPGSD on trees. It manages to solve almost all the problem instances to optimality. The decrease of computational time achieved by adding the preprocessing stage is substantial in case of all but the smallest graphs. The speedup reaches even up to ten times in case of the largest graphs. The tree-specific constraints have produced results worse than the flow-based approach for connectivity. Even in combination with the preprocessing stage the tree-specific model scales poorly, and often performed worse than the flow-based approach without preprocessing.

In case of general graphs the proposed MIP model is significantly less effective and for a wide range of problem instances does not manage to find optimal solutions. A similar positive effect of using the preprocessing stage has been exhibited as in the case of trees. It is important to point out that the proposed MIP formulation is most suitable for graphs having a low ratio of the number of demand nodes over the number of supply nodes ($|V_d|/|V_s|$). In case of the tested data it made it possible to solve all the test instances having this ratio equal to 3. In case of trees the formulation was able to handle all instances with this ratio equal to 10.

Another important observation about the computational times for finding optimal solutions is that it is highly dependent on the specific problem instance in case of both trees and general graphs. The difference between the median and maximal calculation time is significant. For a notable number of graph sizes it is increased more than three times, and in some cases even more than ten times.

In certain cases, the proposed MIP approach manages to outperform the ant colony optimization algorithm presented in [6]. The greatest advantage is in case of trees having the ratio $|V_d|/|V_s| = 3$, where GEN-P manages to find optimal solutions in all the cases for execution times lower or similar to the ones given for ACO [6] in which not all of them are discovered. The advantage even grows with the increase of the number of supply nodes. For instance, in case of graphs with 100 supply and 300 demand nodes GEN-P manages to find all the optimal solutions within 13.6 seconds while ACO in 120 finds only 17. By observing the results in [6], it is noticeable that for small (2,5 supply nodes) general and tree graphs the proposed MIP is advantageous to ACO. It is important to point out that this is not an exact comparison due to the differences in the computational experiments. In case of larger problem instances, especially for general graphs, the scaling of execution time for ACO is much better and it frequently manages to find optimal solutions. For example, in case of general graphs having 10/200 supply/demand nodes ACO manages to find all the optimal solutions within 60 seconds while MIP did not even in 24000.

5 Conclusion

In this paper we have presented a MIP formulation for the MPGSD. Our computational experiments have shown that its use is very efficient in case of trees and graphs having a low ratio of $|V_s|/|V_d|$. In such cases it is even more efficient than the ant colony optimization approach given in [6]. In the future we plan to explore the effectiveness of using a MIP formulation for the different variations of the MPGSD. We also intend to explore the potential of incorporating the presented MIP formulation into a hybrid metaheuristic for solving large scale problem instances.

References

1. Narayanaswamy, N.S., Ramakrishna, G.: Linear time algorithm for tree t-spanner in outerplanar graphs via supply-demand partition in trees. In: CoRR. (2012) abs/1210.7919.
2. Ito, T., Zhou, X., Nishizeki, T.: Partitioning trees of supply and demand. International Journal of Foundations of Computer Science **16**(4) (2005) 803–827

3. Kawabata, M., Nishizeki, T.: Partitioning trees with supply, demand and edge-capacity. IEICE Transactions **96-A**(6) (2013) 1036–1043
4. Ito, T., Demaine, E.D., Zhou, X., Nishizeki, T.: Approximability of partitioning graphs with supply and demand. Journal of Discrete Algorithms **6**(4) (2008) 627 – 650
5. Ito, T., Zhou, X., Nishizeki, T.: Partitioning trees of supply and demand. Lecture Notes in Computer Science **2518** (2002) 612–623
6. Jovanovic, R., Tuba, M., Voß, S.: An ant colony optimization algorithm for partitioning graphs with supply and demand. Technical report (2014)
7. Popa, A.: Modelling the power supply network - hardness and approximation. Lecture Notes in Computer Science **7876** (2013) 62–71
8. Jovanovic, R., Bousselham, A., Voß, S.: A heuristic method for solving the problem of partitioning graphs with supply and demand. Annals of Operations Research (2015)
9. Morishita, S., Nishizeki, T.: Parametric power supply networks. Lecture Notes in Computer Science **7936** (2013) 245–256
10. Morishita, S., Nishizeki, T.: Parametric power supply networks. Journal of Combinatorial Optimization **29**(1) (2015) 1–15
11. Lin, M., Li, W., Feng, Q.: Parameterized minimum cost partition of a tree with supply and demand. Lecture Notes in Computer Science **9130** (2015) 180
12. Jovanovic, R., Bousselham, A., Voß, S.: Partitioning of supply/demand graphs with capacity limitations: an ant colony approach. Journal of Combinatorial Optimization (2015)
13. Inoue, K., Nishizeki, T.: Spanning distribution forests of graphs. Lecture Notes in Computer Science **9130** (2014) 117–127
14. Kawabata, M., Nishizeki, T.: Spanning distribution trees of graphs. IEICE Transactions on Information and Systems **97**(3) (2014) 406–412
15. Morgan, M., Grout, V.: Finding optimal solutions to backbone minimisation problems using mixed integer programming. In: Proceedings of the 7th International Network Conference (INC 2008). (2008) 53–64