A Greedy Method for Optimizing the Self-Adequacy of Microgrids Presented as Partitioning of Graphs with Supply and Demand

Raka Jovanovic
Qatar Environment and Energy
Research Institute(QEERI)
PO Box 5825, Doha, Qatar
Email*: rjovanovic@qf.org.qa

Abdelkader Bousselham
Qatar Environment and Energy
Research Institute(QEERI)
PO Box 5825, Doha, Qatar
Email: abousselham@qf.org.qa

***Abstract*—In this paper we focus on solving the problem of maximal partitioning of graphs with supply and demand. The interest for this problem is due to the fact that it well represents the optimization of self-adequacy of interconnected microgrids. The chosen method is a heuristic based greedy algorithm which can be applied to very large problem instances in feasible time, which best relates to potential real life applications. To decrease the computational time of the algorithm, suitable auxiliary structures are introduced. To get the best performance, in the sense of finding high quality solutions, we have analyzed the behavior of the algorithm for three different heuristics. In the conducted experiments it has been show that the proposed method is very efficient in the case of small problem instances, for which it frequently manages to find optimal solutions. The performed test have demonstrated that the proposed method generally acquires solutions within 5-10% of the known optimal one.**

***Keywords*—Microgrid, Graph Partitioning, Greedy Algorithm**

I. Introduction

One of the main directions in smartgrid development is in the concept of interconnected microgrids [1]. For such systems the problems of maximizing self-adequacy [2], reliability, supply-security [3] and the potential for self-healing [4] are of the greatest importance. These issues frequently result in hard optimization problems that can not be solved in polynomial time. For many problems occurring in electrical grid optimization it is not necessary to use highly detailed models of the system, but often simplified graph ones can give sufficiently good approximate solutions to the original problem. One example is the use of the concept of maximum partition to optimization of the power supply and delivery networks [5], [6], [7], [8].

One of the goals of using interconnected microgrids is to make each of them as independent from the rest of the system as possible. Here we use the term independent in the sense that there is a minimum of power exchange between the connected microgrids. Such independence has two main advantages, first the entire grid is less complex due to the lower number of connections between microgrids. Secondly, each of the microgrids is more resistant to failures in the main grid which results in higher power reliability. This problem is defined as maximizing the self-adequacy of interconnected microgrids. A very interesting approach to finding approximate solutions to this problem has been presented in article[2]. Earlier, a similar concept has been used for the efficient islanding of large grids, into islands with a balanced generation/load while satisfying certain constraints. This problem has been presented in the article [9] in a form of a graph problem, and more recently, in the aspect of smartgrids in [10]. In general when optimizing certain aspects of the electrical grid, as in the case of self-adequacy of interconnected micro-grids, one of the main problems is that we are attempting to optimize very large and complex systems. In practice this results in two directions of research optimizing small detailed systems or large simplified ones.

The problem of maximum partitioning of graphs with supply and demand (MPGSD) is closely related to the optimization the self-adequacy of interconnected microgrids, although in a simplified form. In this paper we focus on developing a heuristic for finding approximate solution to it. In the literature only limited research has been conducted on solving the MPGSD and it is mostly oriented on theoretical aspects [11], [12], [6], [13]. Even in this case, the majority of the effort has been in developing algorithms for creating approximated versions of the problem. Due to the complexity of the MPGSD, the focus has been on solving the problem on some specific graph types like trees [12], [6], [13] and series-parallel graphs[11]. When modeling real electrical grids, models with such a constraint significantly limit their application. On the other hand the existing methods are, to a certain extent, constrained to solving problem instances of smaller sizes.

To fill this gap we have developed a greedy algorithm for finding high quality solutions to the MPGSD. Although heuristic based greedy algorithms manage to find solutions of lower quality than more complex meta heuristics, they have a significant advantage when computational speed is considered. Such algorithms have proven their efficiency in a wide rang of problems going from image compression [14], transport logistics [15] and facility positioning [16], etc. In the case of maximizing the self-adequacy of interconnected micro-grids the problems that are most interesting for real life applications are of significant size. Greedy algorithms are often capable to find solutions for very large problems in feasible time. Due to this fact, this type of method is very suitable for this practical problem. The main contribution of this paper is the development of a greedy algorithm dependent on two heuristic

functions that can be used for solving MPGSD. In our tests we show that the proposed algorithm can find solutions that generally have an error of 5-10% from the optimal one.

The paper is organized as follows. In the second section we give the definition of the MPGSD. In the next section we present the greedy algorithm for the problem of interest and give two corresponding heuristics. In the forth sections we show the results of our computational experiments.

II. Maximal partitioning of a graph with supply/demand

In this section we give a definition for the MPGSD as given in article [11]. It is formulated as follows:

Let $G = (V, E)$ be an undirected graph with a set of vertices V and a set of edges E. V is split into two disjunct sets V_s and V_d. Each vertex $u \in V_s$ is called a supply vertex and has a corresponding positive integer value $sup(u)$. Similarly, the vertex $v \in V_d$ is called a demand vertex and has a corresponding negative integer value $sup(v)$. Each demand vertex can receive 'power' from one supply vertex through the edges of G. The goal is to find the partition $\Pi = \{S_1, S_2, .., S_n\}$ of the graph G. All the subgraphs in Π are connected subgraphs and they only have one supply node which has a consequence $|V_s| = n$. Each of the S_i has to satisfy the constraint that the total demand must be less or equal to supply. The goal is to maximize the fulfillment of demands, or more precisely to maximize the sum

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

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

$$\sum_{v\in S_i} sup(v) \geq 0 \tag{2}$$

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

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

It has been show that the MPGSD is NP-Hard even in the case of its restriction of having only one supply node and the graph having a star structure [11].

III. Greedy Heuristic

A. Outline

In this section we present a greedy algorithm for solving the problem of interest. From the definition, as previously mentioned, the solution will consist of $|\Pi| = n$ subgraphs, where $n = |V_s|$ is the number of supply nodes. The general idea of the algorithm is to start with n disjunct subgraphs S_i, that initially contain only a supply node s_i. Next, at each step (iteration) of the algorithm one vertex $v \in V_d$ is selected and added to a selected subgraph S_i. The selection of both v and S_i should be performed in a way that the newly generated subgraph is connected, each v can be a part of at most one S_i and the Eq. 2 must be satisfied. Of course, the selection of S_i and v should be done using some heuristic that we expect will produce high quality solutions of the problem.

To formally describe such an algorithm, we shall first define the function NV for $v \in V$

$$NV(v) = \{u | u \in V \wedge (u, v) \in E\} \tag{5}$$

Here $NV(v)$ represents the set of adjacent vertices to v in G.

The idea is to slowly grow each of the subgraphs S_i by adding new vertices v to them. To make this process simpler, we can define the function NV for subgraphs S_i.

$$\hat{N}_i = NV(S_i) = \{u | u \in V \wedge \exists (v \in S_i)(u, v) \in E\} \tag{6}$$

It is obvious that if at some step of the algorithm we add a vertex v to S_i, the new subgraph will be connected if $v \in \hat{N}_i$. The problem is that the new S_i my not satisfy Eq. 2, or there may exist such an S_j for which $v \in S_j$. To avoid this, a corrected set of vertices N_i is defined, in a way that by adding $v \in N_i$ to S_i the new subgraph will satisfy all the constraints. Let sup_i be the available supply for subgraph S_i as given in Eq. 7.

$$sup_i = \sum_{v\in S_i} sup(v) \tag{7}$$

Now we can define N_i in the following way.

$$N_i = \{u | u \in \hat{N}_i \wedge sup(u) \leq sup_i\} \setminus \bigcup_{j=1}^{n} S_j \tag{8}$$

Using the sets N_i, we can specify the greedy algorithm for the MPGSD using two heuristic functions. At each step of the algorithm, the first heuristic hs is used to select the best S_i, and the second heuristic hv will be used to select the best $v \in N_i$ to be added to S_i.

B. Heuristics

The first heuristic function hv should give us the desirability of adding $v \in N_i$ to S_i. Let us define hv

$$hv(v) = |sup(v)| \tag{9}$$

In Eq. 9 vertices with high demand are considered more desirable. The logic behind this, is that it gets harder to satisfy high demands as the algorithm progresses since the available supply decreases as new vertices are added to the subgraphs. Because of this it is better to resolve high demands early.

The second heuristic function hs should give us the desirability of subgraph S_i for being expanded. There are two main properties of S_i that should be considered, the available supply sup_i and the number of potential candidates $|N_i|$.

Subgraphs with a higher value of sup_i are considered more desirable, in the sense that they should be selected earlier. The reason for this is that it is expected that more vertices need to be added to such subgraphs than to ones with a lower supply. As the algorithm progresses the number of non satisfied demand vertices drops and in later stages of the algorithm there may not be enough available demand to reach sup_i. Using this logic we define the heuristic hs_1 as

$$hs_1(S_i) = sup_i \tag{10}$$

The other potential problem with the proposed algorithm is that a subgraph S_i will not be able to expend further due to being cut of by other subgraphs. This is a consequence of the fact that each vertex can be a part of only one subgraph, which creates a possibility that all of the neighbors of subgraph S_i will be added to other subgraphs. Because of this we shall consider subgraphs with a low value of $|N_i|$ highly desirable. We can define a new heuristic hs_2

$$hs_2(S_i) = \frac{1}{|N_i|} \quad , |N_i| \neq 0 \tag{11}$$

We can also define a third heuristic function hs_3 that balances the two effects

$$hs_3(S_i) = \frac{s_i}{|N_i|} \quad , |N_i| \neq 0 \tag{12}$$

At this point wish to point out that for the heuristic functions hs_2, hs_3 the case when $|N_i| = 0$ will be considered least desirable and will never be selected.

C. Implementation

For the iterative algorithm outlined in the previous section, it is important to emphasize that the subgraphs S_i and corresponding sets of neighboring vertices N_i will be changing with the progress of the algorithm. To reflect this fact, we will add the new notations S_i^j, N_i^j and sup_i^j corresponding to the state of S_i, N_i and sup_i at iteration j. It is essential to have an efficient procedure for calculating N_i^j. In the following text such a procedure is presented.

The initial state of the sets N_i^0 consist of all the neighbors of the corresponding supply vertex s_i that are not supply vertices, as given in Eq. 13.

$$N_i^0 = NV(s_i) \setminus V_s \tag{13}$$

The N_i^0 correspond to the subgraphs $S_i = \{s_i\}$

The update procedure at iteration $j+1$, when a vertex $v \in N_i^j$ is added subgraphs S_i^j, for N_i^j, S_i^j and sup_i^j should reflect the following changes:

1) v should not be a member of any N_i^{j+1}
2) v should be added to S_i^{j+1}
3) The available supply for subgraph S_i^{j+1} sould be decreased $sup_1^{j+i} = sup_i^j + sup(v)$
4) Add to N_i^{j+1} all neighbors of v that are not in any of S_i^{j+1}
5) remove all elements of N_i^{j+1} which have a demand higher that the new available supply sup_i^{j+1}

All the necessary updates rules except the forth one are trivial. This update rule can easily be implemented using a brute force approach but by doing so the main advantage of a greedy algorithm, the computational speed, is lost due to number of checks.

Such computational cost can easily be avoided by adding an auxiliary structure that helps tracks the used vertices by updating the set of edges E for graph G, using a similar procedure like in articles [17], [18]. The idea is adding a set of edges E_i that gives only connections to non used elements of V_d at iteration i of the algorithm. To make such a calculation possible we shall define the functions NV_i and NE_i for $v \in V$ that correspond to iteration i

$$NV_i(v) = \{u | u \in V \wedge (u,v) \in E_i\} \tag{14}$$
$$NE_i(v) = \{(u,v) | u \in V \wedge (u,v) \in E_i\} \tag{15}$$

Here $NV_i(v)$ represents the set of adjacent vertices to v in $G_i(V, E_i)$, and analogy $NE_i(v)$ represents the set of edges in $G_i(V, E_i)$ which contain v. In the same way as in Eq. 8 we define extensions of $NV_i(S)$, $NE_i(S)$ for sets. Using these functions we have the initial set E_0

$$E_0 = E \setminus \bigcup_{j=1}^{n} NE_(s_i) \tag{16}$$

The update function for E_j, at iteration $j+1$ when vertex v is added to some subset

$$E_{j+1} = E_j \setminus NE_i(v) \tag{17}$$

Now the update rule 4, at iteration $j+1$, can be expressed trivially as

$$N_i^{j+1} = N_i^j \cup NV_i(v) \tag{18}$$

To have a more clear presentation of the propose algorithm we give it a form of the following pseudo-code.

$E_c = E$
for all $s_i \in V_s$ **do**
  $S_i = \{s_i\}$
  $N_i = NV(s_i, E)$
  $E_c = E_c \setminus NE(s_i, E)$
  $sup_i = sup(s_i)$
end for

Remove all s_i from all N_i

while $(Sum(sup_i) > 0)$ **and** $Sum(|N_i|) > 0)$ **do**

  $k = max\_hs_(\Pi)$
  $v = max\_hv(S_k)$

  $S_k = S_k \cup \{v\}$
  $N_k = N_k \cup NV(v, E_c)$
  $E_c = E_c \setminus NE(v, E_c)$
  $sup_k = sup_k - sup(v)$

  Remove v from all N_j
  Remove all $w \in N_k$ from which $|sup(w)| > sup_k$

end while

The solution of the MPGSD will be the partition $\Pi = \{S_1, S_2, .., S_n\}$, the pseudo code calculates the elements for each S_i. In it NV and NE correspond to the appropriate functions given in the previous text, but also include the set of edges of the graph as a parameter. Functions $max\_hs_(\Pi)$ returns the index i of the subgraph $S_i \in \Pi$ with the highest value of the heuristic function hs. Similarly, function $max\_hv(S_k)$ returns the vertex $v \in S_k$ with the highest value of hv.

TABLE I. COMPARISON OF HEURISTICS FOR THE MPGSD

Dem X		hs_1			hs_2			hs_3		Opt
Sup	Avg	Hit	Best	Avg	Hit	Best	Avg	Hit	Best	
2 X 6	53.5	27	33	55.3	32	38	54.4	30	36	56.9
2 X 10	77.4	8	24	81.2	14	36	79.6	11	31	85.9
2 X 20	156.2	12	25	159.1	14	32	158.0	11	28	163.5
3 X 9	68.3	12	25	72.3	19	37	71.8	17	33	75.9
3 X 15	120.2	5	25	121.2	10	27	120.5	5	23	130.3
3 X 30	246.4	4	12	249.2	3	27	252.4	5	20	259.2
5 X 15	114.3	2	19	117.2	7	31	116.5	4	28	125.1
5 X 25	183.2	0	11	194.3	4	33	190.2	2	16	202.6
5 X 50	403.0	0	3	415.2	3	25	412.0	1	15	429.0
10 X 30	224.4	1	9	231.6	3	26	228.3	1	19	246.7
10 X 50	387.0	0	8	392.0	0	19	392.2	0	15	424.1
10 X 100	808.6	0	6	821.7	0	21	823.3	0	15	852.9
25 X 75	555.4	0	5	574.7	0	22	572.0	0	14	618.6
25 X 125	954.6	0	2	982.1	0	21	979.7	0	17	1054.0
25 X 250	2005.4	0	0	2066.5	0	28	2052.0	0	13	2146.2
50 X 150	1102.8	0	1	1147.0	0	30	1134.0	0	9	1238.7
50 X 250	1914.6	0	0	1958.0	0	20	1961.7	0	20	2111.6
50 X 500	3948.0	0	2	4041.3	0	23	4034.9	0	17	4217.7

TABLE II. COMPARISON OF HEURISTICS FOR THE MPGSD

Sup X Dem	hs_1			hs_2			hs_3		
	Avg	Max	StDev	Avg	Max	StDev	Avg	Max	StDev
2 X 6	6.6	36.7	11.3	3.5	36.7	8.0	4.6	36.7	9.2
2 X 10	9.5	45.1	10.3	5.4	33.0	6.9	7.1	31.8	7.2
2 X 20	4.3	30.1	7.5	2.7	29.0	5.2	3.4	29.0	5.3
3 X 9	9.9	39.7	10.6	4.9	20.2	5.7	5.5	17.5	5.7
3 X 15	7.9	22.9	5.9	6.9	29.7	7.9	7.7	22.9	6.1
3 X 30	5.1	34.1	7.1	4.0	38.5	8.0	2.6	13.9	2.8
5 X 15	8.7	22.7	5.6	6.2	19.6	6.3	6.9	21.4	5.9
5 X 25	9.6	26.3	6.0	4.0	17.7	3.9	6.2	16.3	4.0
5 X 50	6.1	21.0	4.6	3.2	15.0	4.0	4.0	13.1	3.3
10 X 30	9.0	20.1	4.7	6.1	18.6	4.5	7.4	19.0	4.6
10 X 50	8.8	20.8	4.6	7.6	21.7	5.0	7.6	17.0	3.6
10 X 100	5.2	17.8	3.4	3.7	9.6	2.7	3.5	9.7	1.9
25 X 75	10.2	22.9	4.1	7.1	18.1	3.9	7.5	19.0	3.3
25 X 125	9.4	14.1	1.9	6.8	16.1	2.8	7.0	11.6	2.0
25 X 250	6.6	10.7	1.9	3.7	10.9	2.3	4.4	7.2	1.4
50 X 150	11.0	16.7	2.6	7.4	16.7	2.6	8.5	13.1	2.2
50 X 250	9.3	13.8	1.8	7.3	12.9	1.7	7.1	9.7	1.3
50 X 500	6.4	10.5	1.8	4.2	7.4	1.4	4.3	6.0	0.9

IV. RESULTS

In this section we give an evaluation of the proposed algorithm and a comparison of the effect of using different heuristic functions. The algorithm has been implemented in C# using Microsoft Visual Studio 2012. The calculations have been done on a machine with Intel(R) Core(TM) i7-2630 QM CPU 2.00 Ghz, 4GB of DDR3-1333 RAM, running on Microsoft Windows 7 Home Premium 64-bit.

To be able to have a good comparison of the different heuristic functions we have analyzed the behavior of each of them for a wide range of graph sizes. We have observed graphs having 2-50 supply nodes and 6-500 demand nodes. For each of the test sizes 40 different problem instances are generated and we observe the average solution quality for each size.

The problem instances (graphs) for n supply and m demand vertices have been generated using the following algorithm. First we would generate an array containing $n+m$ integer random weights uniformly distributed inside of the interval $[-1,-18]$. Next $(n+m)*1.5$ random edges would be added to the graph but making sure that the graph is connected. In the next step n random vertices would be selected to be seeds for n subgraphs (partitions). The subgraphs would be grown using a iterative method until $(n+m)*(0.95)$ vertices of the original graph are contained in one of the subgraphs. The growth of subgraph S_i is done by adding a random neighboring

vertex that does not belong to any of the other subgraphs. Finally, for each of the subgraphs S_i a random vertex $v \in S_i$ is selected and its weight w is set using the following formula

$$w = |\sum_{a \in S_i} sup(a)| + sup(v) \tag{19}$$

For each of the test graphs, generated using this method, the optimal solution is known and its covered demand is equal to the sum of supplies of all supply nodes.

To evaluate the proposed greedy algorithm, we have run the algorithm with the three different heuristic functions on each of the test sizes. We have observed several different aspects of performance of the heuristics for each group of 40 problem instances. The first group of evaluations is given in Table I, where we show the average number of covered demand, the number of times each heuristic has found the optimal solution and the number of times it has found the best solution compared to the other proposed heuristics. From these results it is evident that hs_2, hs_3 manage to significantly out preform the hs_1. For the 18 different pairs of (n, m) not in one case did the heuristic based only the amount of available supply manage to get better results that the other two methods. Out of the two other heuristics hs_2 had a better performance but the difference is less significant. From this fact, we can conclude that the problem of avoiding cutting off subgraphs from the rest of the graph is of great importance. In our tests we have also observed the number of times each heuristic has managed to find the optimal partitioning. It is notable that in case of small problems, the best performing heuristic has managed to find optimal solutions in more that 25% of the problem instances, in some cases even over 50%. For larger graph sizes none of the proposed heuristics managed to acquire the optimal solutions.

In the Table II, we present statistics for normalized error of the solutions acquired by the proposed algorithm compared to the known optimal ones. More precisely we have calculated the error in percent corresponding to the three heuristics hs_1, hs_2 and hs_3 compared to the optimal solution, for each of the 40 problem instances inside of one problem size. In Table II we present the values for the average error, maximal error and the standard deviation. This type analysis also confirms that hs_2 is the best performing heuristic having average errors between 2.7% and 7.6%, which is close to hs_3 and significantly better than hs_1. When we observe the average maximal error, it is notable that although hs_3 has an over all worse behavior than hs_2, it is much more robust. We say this because hs_3 has had the smallest maximal error in 14 out of 18 tested graph sizes, which means that it rarely acquires very bad solutions when compared to the other two heuristics.

V. Conclusion

In this paper, we have presented a heuristic based greedy algorithm for solving the MPGSD which is closely related to the optimization of self-adequacy of interconnected microgrids. The proposed algorithm is two step in the sense that at each iteration two heuristic are used one for selecting which partition should be expanded an a second one that selects which vertex should be added. In the article we have shown that by using some auxiliary structures it is possible to make such an algorithm very computationally efficient.

We have performed tests on a wide range of different graphs sizes and shown that the proposed algorithm can find high quality solutions to the problem of interest. More precisely the best performing heuristic has managed to have an average error,when compared to the know optimal solution, in between 2.7% and 7.6%. Our test have also shown that in the case of small problem instances the proposed algorithm can in many cases find optimal solutions.

In the future we plan to improve the proposed algorithm by incorporating a local search method similar to 2-opt and including a certain level of randomization. The other direction of our work will be in adapting the MPGSD to better reflect the problems occurring in microgrids. This would be done by adapting the problem to a stochastic environment, removing the constraint of uniqueness of supply and including time dependence for supply/demand.

References

[1] N. Hatziargyriou, H. Asano, R. Iravani, and C. Marnay, "Microgrids," *Power and Energy Magazine, IEEE*, vol. 5, no. 4, pp. 78–94, July 2007.

[2] S. Arefifar, Y. Mohamed, and T. H. M. EL-Fouly, "Supply-adequacy-based optimal construction of microgrids in smart distribution systems," *Smart Grid, IEEE Transactions on*, vol. 3, no. 3, pp. 1491–1502, Sept 2012.

[3] S. Arefifar, Y.-R. Mohamed, and T. EL-Fouly, "Optimum microgrid design for enhancing reliability and supply-security," *Smart Grid, IEEE Transactions on*, vol. 4, no. 3, pp. 1567–1575, Sept 2013.

[4] ——, "Comprehensive operational planning framework for self-healing control actions in smart distribution grids," *Power Systems, IEEE Transactions on*, vol. 28, no. 4, pp. 4192–4200, Nov 2013.

[5] N. Boulaxis and M. Papadopoulos, "Optimal feeder routing in distribution system planning using dynamic programming technique and gis facilities," *Power Delivery, IEEE Transactions on*, vol. 17, no. 1, pp. 242–247, Jan 2002.

[6] T. Ito, X. Zhou, and T. Nishizeki, "Partitioning trees of supply and demand," *International Journal of Foundations of Computer Science*, vol. 16, no. 04, pp. 803–827, 2005.

[7] A. B. Morton and I. M. Mareels, "An efficient brute-force solution to the network reconfiguration problem," *Power Delivery, IEEE Transactions on*, vol. 15, no. 3, pp. 996–1000, 2000.

[8] J.-H. Teng and C.-N. Lu, "Feeder-switch relocation for customer interruption cost minimization," *Power Delivery, IEEE Transactions on*, vol. 17, no. 1, pp. 254–259, Jan 2002.

[9] K. Sun, D.-Z. Zheng, and Q. Lu, "A simulation study of obdd-based proper splitting strategies for power systems under consideration of transient stability," *Power Systems, IEEE Transactions on*, vol. 20, no. 1, pp. 389–399, 2005.

[10] J. Li, C.-C. Liu, and K. P. Schneider, "Controlled partitioning of a power network considering real and reactive power balance," *Smart Grid, IEEE Transactions on*, vol. 1, no. 3, pp. 261–269, 2010.

[11] T. Ito, E. D. Demaine, X. Zhou, and T. Nishizeki, "Approximability of partitioning graphs with supply and demand," *Journal of Discrete Algorithms*, vol. 6, no. 4, pp. 627 – 650, 2008, selected papers from the 1st Algorithms and Complexity in Durham Workshop (ACiD 2005).

[12] N. S. Narayanaswamy and G. Ramakrishna, "Linear time algorithm for tree t-spanner in outerplanar graphs via supply-demand partition in trees," 2012.

[13] M. Kawabata and T. Nishizeki, "Partitioning trees with supply, demand and edge-capacity," *IEICE Transactions*, vol. 96-A, no. 6, pp. 1036–1043, 2013.

[14] R. Jovanovic and R. A. Lorentz, "Adaptive lossless prediction based image compression," *Appl. Math*, vol. 8, no. 1, pp. 153–160, 2014.

[15] R. Jovanovic and S. Voss, "A chain heuristic for the blocks relocation problem," *Computers & Industrial Engineering*, vol. 75, no. 0, pp. 79 – 86, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360835214001909

[16] R. Jovanovic, M. Tuba, and D. Simian, "Ant colony optimization applied to minimum weight dominating set problem," in *Proceedings of the 12th WSEAS international conference on Automatic control, modelling & simulation*. World Scientific and Engineering Academy and Society (WSEAS), 2010, pp. 322–326.

[17] R. Jovanovic and M. Tuba, "An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem," *Applied Soft Computing*, vol. 11, no. 8, pp. 5360 – 5366, 2011.

[18] ——, "Ant colony optimization algorithm with pheromone correction strategy for the minimum connected dominating set problem." *Comput. Sci. Inf. Syst.*, pp. 133–149, 2013.