An efficient algorithm for density-balanced partitioning in distributed PageRank

Sumalee Sangamuang Pruet Boonma Juggapong Natwichai
Data Engineering and Network Technology Laboratory
Department of Computer Engineering
Chiang Mai University
Chiang Mai 50200, Thailand
Email: sumalee.sa@cmu.ac.th,{pruet,juggapong}@eng.cmu.ac.th

Abstract—**Google's PageRank is the most notable approach for web search ranking. In general, web pages are represented by web-link graph; a web-page is represented by a node, and a link between two pages is represented by an edge. In particular, t is not efficient to perform PageRank of a large web-link graph in a single computer. Distributed systems, such as P2P, are viable choices to address such limitation. In P2P-based PageRank, each computational peer contains a partial web-link graph, i.e., a sub-graph of the global web-link graph, and its PageRank is computed locally. The convergence time of a PageRank calculation is affected by the web-link graph density, i.e., the ratio of the number of edges to the number of nodes, such that if a web-link graph has high density, it will take longer time to converge. As the execution time to compute the P2P-based web ranking is influenced by the execution time of the slowest peer to compute the local ranking, the density-balanced local web-link graph partitioning can be highly desirable. This paper addresses a density-balanced partitioning problem and proposes an efficient algorithm for the problem. The experiment results show that the proposed algorithm can effectively partition graph into density-balanced subgraph with an acceptable cost.**

I. Introduction

There are several approaches to compute ranking in web search result, e.g., [1], [2], [3]. The most notable approach for web ranking is Google's PageRank [4], [5]. In particular, PageRank is a model for determining the importance of a web page through its incoming and outgoing links. In general, web pages and their links are represented by web-link graph; a web-page is represented by a node, and a link between two pages is represented by an edge. Each page considers the summation of the value from incoming links as its PageRank value while distributes its PageRank value to the other pages through the outgoing links . Therefore, PageRank calculation has to be performed iteratively until the values are converged. In particular, the convergence time of a PageRank calculation is affected by the web-link graph density, i.e., the ratio of the number of edges to the number of nodes, such that if a web-link graph has high density, it will take longer time to converge [4].

Generally speaking, a web-link graph can be very large, computing PageRank in a single computer is not effective. Peer-to-Peer (P2P) is a viable choice to address such limitation [6], [7], [8]. In P2P-based PageRank computing, each computational peer contains a local web-link graph, i.e., a sub-graph of the global web-link graph, and its PageRank is computed locally. To be able to compute the global ranking, a special node, so called world-node, is constructed to store the linkage information of the other peers. After the local web rankings of all peers are computed, a collaborative web ranking between any two peers will be proceeded to adjust the web ranking; this process is called peer-meeting. In a peer-meeting, the computation starts with merging of each two web-link graphs including the common world nodes. Then, the ranking is performed on such merged graphs. Finally, the merged graphs will be split into two local web-link graphs stored in the two participating peers as before.

The problem of partitioning a graph into k components of approximately equal size in term of the number of nodes while minimizing the cross edges between different components of the cut, called $(k, 1+\varepsilon)$-balanced partition problem, is discussed in [9], where ε is the size-unbalanced constrains. An efficient algorithm, which gives a polynomial time for partitioning a graph into k almost equal size components, was also proposed in the same literature. However, it focuses solely on approximately balancing the number of nodes in each components, which does not imply that the density will be balanced.

As the execution time to compute the P2P-based web ranking is influenced by the execution time of the slowest peer to compute the local ranking, the density-balanced local web-link graph partitioning can be highly desirable. So, this paper proposes an extension of $(k, 1+\varepsilon)$-balanced problem and an efficient algorithm to solve the problem. We define the problem, so called $(k, 1+\varepsilon, \alpha)$-balanced problem, where the density-unbalanced constrains, α, is also considered in the partitioning. Then, an efficient algorithm, so called density-balanced partitioning (DBP) algorithm for addressing such problem is proposed. The idea of the proposed work is based on binary tree decompositions which efficiently filters out the less-balanced partitioning. The proposed work is evaluated by thorough experiments against the other two algorithms [10], [9].

The organization of this paper is as follows. In Section II, the $(k, 1+\varepsilon, \alpha)$-balanced problem is presented and discussed.

Then, in Section III, the BDP algorithm is proposed. Section IV presents the experiment results of our proposed work. Finally, the conclusion and the outlook for the future work are given in the last section.

II. $(k, 1+\varepsilon, \alpha)$-BALANCED PARTITION PROBLEM

A. Basic Notations

In this section, the basic notations are defined as follows.

A web link graph is represented as a directed graph which is an ordered pair $G = (V, E)$. The set of nodes of G is denoted as $V(G)$ while the set of edges of G is denoted as $E(G)$. An element in $E(G)$ is an ordered pair of (u, v) such that $u, v \in V(G)$. A subgraph $G' = (V', E')$ of G is a graph that satisfied following conditions: $V' \subseteq V$ and $E' \subseteq E$.

Components of graph G are its subgraphs, each component is denoted as $G_i = (V_i, E_i)$; $i \in I^+$. The set of component nodes is denoted as $V(G_i)$ where $V(G_i) \subseteq V(G)$. On the other hand, the set of component edges is denoted as $E(G_i)$ where $E(G_i) \subseteq E(G)$.

A partition P of G is a set of the components of G which satisfies the following properties: $\bigcup_{i=1}^{k} V(G_i) = V(G)$ and $\bigcap_{i=1}^{k} V(G_i) \subseteq V(G)$; where k is the set cardinality. The density of component G_i, $d(G_i)$, is the ratio of the cardinality of the edges set and the node set of G_i , i.e.,

$$d(G_i) = \frac{|E(G_i)|}{|V(G_i)|}. \tag{1}$$

B. Problem Definitions

In [9], the problem of (k, v)-balanced partitioning is presented. In this problem, a graph is partitioned into k components and each components has less than $v \cdot \frac{n}{k}$ nodes where n is the number of the nodes in the graph. When $v \geq 2$, it is possible to find an approximation algorithm, with ratio of $O(log\ n)$, that can solve this problem efficiently [11]. However, they argue that there is no studies on the problem when $v < 2$; therefore, they focus on $(k, 1+\varepsilon)$-balanced partitioning problem, where $\varepsilon \in (0, 1)$ is the size-unbalanced constrains. In particular, the problem is defined as followed: Given a graph $G = (V, E)$, find partitions P where each components contains at most $(1+\epsilon)\frac{n}{k}$ nodes. It is proved that $(k, 1)$-balanced partition problem is a NP-hard problem, by reduction from 3-Partition problem. However, it is possible to find a polynomial time approximation algorithm for $(k, 1+\varepsilon)$-balanced partition problem [9].

Based on the aforementioned problem, this work defines the density-balanced partitioning problem, i.e., $(k, 1+\varepsilon, \alpha)$-balanced problem, as followed: Given a graph $G = (V, E)$, find partition P where each component contains at most $(1+\epsilon)\frac{n}{k}$ nodes and the density unbalanced constrain α is not violated. n is the number of nodes in G and k is the number of components in P. The density unbalanced constrain assures that the differences between any two component's densities is not greater than α. This problem has at least the same complexity as the aforementioned problem; in particular, if $\alpha = n$, the algorithm that can solve this problem can also solve the aforementioned problem in [9].

III. PROPOSED ALGORITHM

Given the combinatorial nature of partitioning problems, it is not efficient to consider all possible partitions. To reduce the number of possible partitions, the proposed algorithm, called density-balanced partitioning (DBP) algorithm, prunes the search space, i.e., all possible partitions, by using binary tree decomposition. In particular, a web-link graph is decomposed to binary trees with recursive edges separation algorithm where each node of the tree contains a subgraph of the original web-link graph. Then, the trees are filtered by size-unbalanced constrain ε and density-unbalanced constrain α. Only the trees that satisfied the two constrains will be considered as the results. Finally, the output partitions are extracted from the trees.

Following definition shows the properties of the binary tree decomposition.

Definition 1 (Binary Tree Decomposition): Let G = (V,E) be a graph, a binary tree decomposition T of graph G is a tree T that satisfies the following properties

(I) Root of tree T contains all nodes of V(G)

(II) For each internal node of tree T contains $V(G_i) \subseteq V(G)$

(III) Child nodes c of a parent node p contains the subset of $V(G_p)$ in the parent nodes such that $\bigcup_{i \in c} V(G_i) = V(G_p) \wedge \bigcap_{i \in c} V(G_i) = \emptyset$

(IV) For each external node of tree T contains an individual node of G

From the definition, a graph will be decomposed into binary trees where the root nodes of the trees contain all of the nodes in the graph (I). Then, the content of a tee node is split to its child nodes (II, III). This separation of tree node's content is performed by edge separation algorithm, which will be explained next. Finally, tree nodes' content is split until each node contains only one node from the graph (IV). Figure 1 shows examples of tree decomposition from a graph G. There are many tree decompositions from a graph G but only two example shows in this figure.

Definition 2 shows the properties of desired edge separation. The edge separator is used to separate a content of a node (see Definition 1 (III)) in a binary tree into two parts and assigned to the child nodes. Only the separations that meet the desired properties in the definition will be considered.

Definition 2 (Edges Separation): Let G = (V,E), an edges separator is a cut that partitions $V(G)$ into $V(G_1)$ and $V(G_2)$ such that the following properties are satisfied:

(I) $V(G_1) \cup V(G_2)$ = $V(G) \wedge V(G_1) \cap V(G_2) = \emptyset$,

(II) $-1 \leq |V(G_1)|$ - $|V(G_2)| \leq 1$,

(III) $|E(G_i)| \leq threshold.$

From the definition, edge separator partitions a graph into two sub-graph (I) where the different between the number of nodes in each subgraph is less than one (II). Finally, the size of edge separation must less than a threshold. Figure 2 shows an example of edge separator where the value of threshold is two.

Algorithm 1 shows the pseudo code of the proposed DBP algorithm. From the input graph G, a list of decomposed

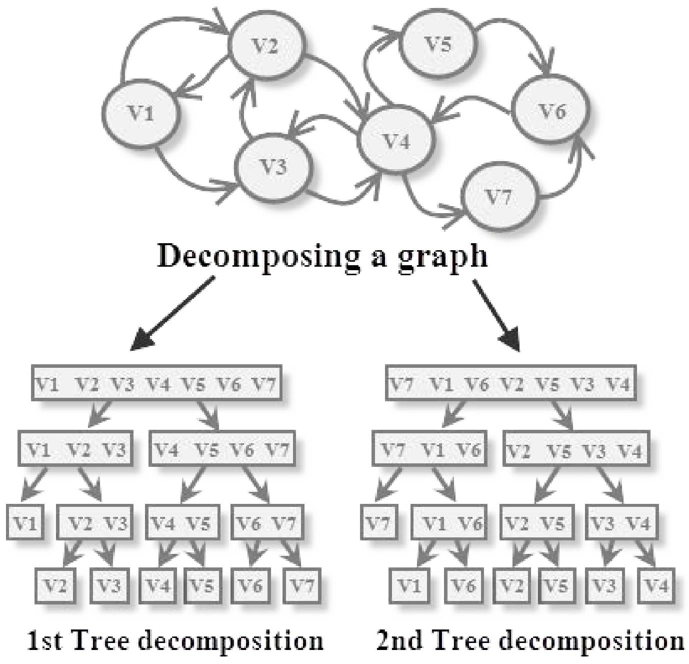

Fig. 1: An example of binary tree decomposition.

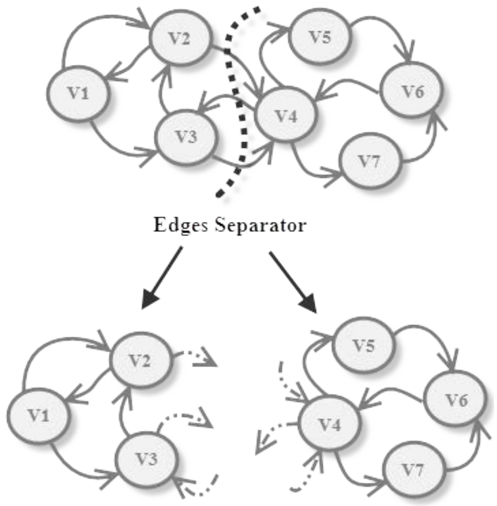

Fig. 2: An example of edge separator.

binary trees is created by using BTD function (line 3). This BTD function generates trees that satisfied the requirement in Definition 1 and 2. Then, each tree is filtered by its number of nodes with the node size constraint (line 6-10). Next, the maximum different in density between any two remaining nodes is measured (line 12-16). This value will be use to decide whether the decomposed binary tree contains the valid partition where the different in density is less than the density-unbalanced constrain. Also, the tree will not be included in the result if the number of components inside the tree is not k (line 17-19). Finally, the result which contains all valid partitions is returned. The partitions will be selected based on the value of k. The running time of this algorithm is $\mathcal{O}(|V(G)|^3)$.

Algorithm 1 DBP

```
Input: Graph $G = (V, E)$, number of component $k$, size unbalanced constrain $\varepsilon$, density unbalanced constrain $\alpha$.
Output: The density balanced partitions of G.
 1: $TreeList \leftarrow \emptyset$
 2: $PartitionList \leftarrow \emptyset$
 3: $TreeList \leftarrow BTD(G)$
 4: for each $T_i \in TreeList$ do
 5:     $v_{T_i} \leftarrow V(T_i)$
 6:     for each $node_j \in v_{T_i}$ do
 7:         if $|node_j| \geq (1+\varepsilon)\frac{|V(G)|}{k}$ then
 8:             $v_{T_i} \leftarrow v_{T_i} \setminus node_j$
 9:         end if
10:     end for
11:     $md \leftarrow 0$
12:     for each $node_n \in v_{T_i}$ do
13:         for each $node_m \in v_{T_i}, node_m \neq node_n$ do
14:             $md \leftarrow \max(md, |d(node_m) - d(node_n)|)$
15:         end for
16:     end for
17:     if $|v_{T_i}| > 0 \wedge md \leq \alpha \wedge component(v_{T_i}) = k$ then
18:         $PartitionList \leftarrow PartitionList \cup \{T_i\}$
19:     end if
20: end for
21: return $PartitionList$
```

IV. Experiment Results

This section shows the experiment results to evaluate DBP algorithm. In this experiments, DBP is compared with the algorithm in [9], labeled as ABP, and a greedy algorithm used in [10], labeled as GBP. ABP is similar to DBP but does not consider α. This GBP algorithm sorts the nodes in a graph based on their degree. Then, the algorithm assigns the nodes to components in a round-robbin manner until every node is assigned to a component. The experiments evaluate the impact of α on the density-unbalanced value, impact of α on the number of result partitions from a graph, impact of graph size on density-unbalanced value, and the impact of graph size on the execution time. The density-unbalanced value is measured from the different between highest density and lowest density of components in the partition generated from a graph. If this value is small, it means that the components in a partition has similar density. The result partitions from the tree algorithms are the partitions that satisfied the number of components (i.e., k) and the number of nodes per component (i.e, $1+\varepsilon$) constraint.

A. Simulation Setup

The data set used in the experiments was synthesized from GraphStream, a dynamic graph library base on Java. Two types of graph are generated for the data set, directed random graph and Watts-Strogatz graph , a random graph with small-world

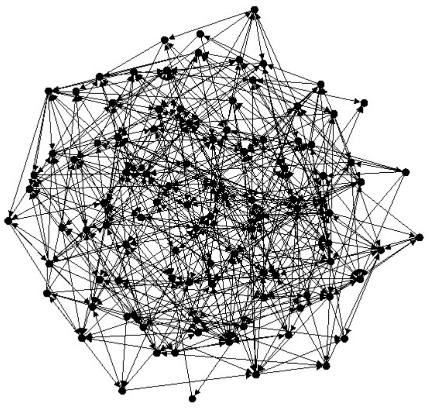

Fig. 3: A directed random graph.

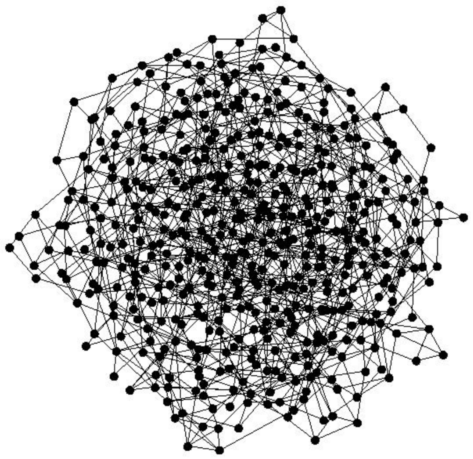

Fig. 4: A Watts-Strogatz graph.

properties base on Watts and Strogatz model [12] . Figure 3 and Figure 4 show examples of a random graph and a Watts-Strogatz graph, respectively. The graph size used in the experiments is in the range of 1000 to 1800 nodes while the average degree of node in a random graph is about 10 and Watts-Strogatz graph is about 4. Section IV-B shows results from directed random graphs while Section IV-C shows results from directed Euclidean graphs. The value of k and ε used in the experiments are 4 and 0.5, respectively. The resulting numbers are from 16 experiments.

B. Directed Random Graph Results

This section shows results from directed random graphs.

Figure 5 shows impact of α on the density-unbalanced values. The graph size this experiment is 1000. The value of α between 0.1 - 0.85 is shown in the X-axis while the density-unbalanced values are shown in the Y-axis. The result shows that when the value of α is small, DBP allows partition to have small density-unbalanced value, as small as 0, compared with ABP and GBP, which has average value of about 0.2. However, then the value of α increased, the result of DBP becomes similar with ABP. GBP is the worst. DBP allows

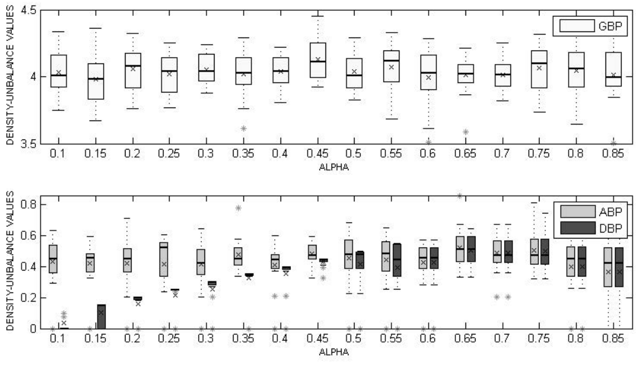

Fig. 5: Impact of α on density-unbalanced value.

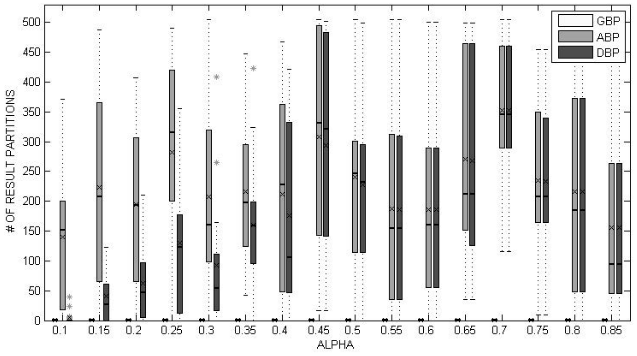

Fig. 6: Impact of α on number of partitions.

components in a graph to have very similar density compared with ABP and GBP.

Figure 6 shows impact of α on the number of result partitions from a graph. The graph size this experiment is 1000. The value of α between 0.1 - 0.85 is shown in the X-axis while the result partitions are shown in the Y-axis. The result from GBP is always be 1 partitions from a graph (show as dashed line in the figure). This shows that, given a graph size of 1000 nodes and value of k as 4, ABP always find many possible parititions from the graph which is not less than DBP. DBP, on the other hand, might not be able to find any partition when the value of α is very small, e.g., around 0.1-0.14. This is because when the value of α is very small, there is no partitions that can satisfy density-unbalanced constrain (see Algorithm 1, lines 12-19). However, when the value of α increased, the number of result partitions of DBP is also increased. Because of density-unbalanced constraint, DBP might not be able to find valid partitions.

Figure 7 shows impact of graph size on the density-unbalanced value. In this experiment, the best density-unbalanced value of each algorithm is presented. The result shows that, regardless of the graph size, DBP always outperform ABP and GBP, significantly.

Although, DBP can finder better density balanced partition; however, it comes with a cost. Because its computational complexity is $\mathcal{O}(|V(G)|^3)$, higher than $\mathcal{O}(|V(G)|^2)$ of ABP and $\mathcal{O}(|V(G)|log(|V(G)|))$ of GBP, it always has higher

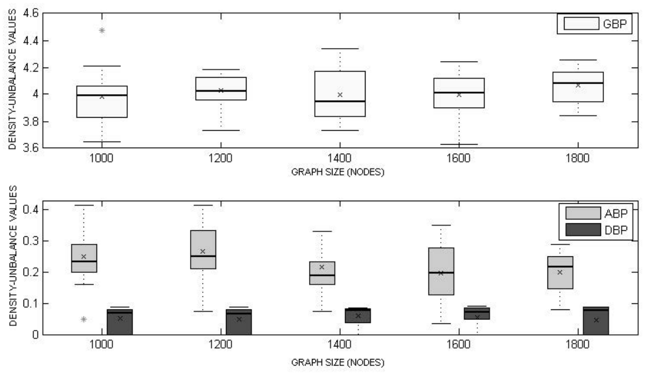

Fig. 7: Impact of graph size on density-unbalanced value.

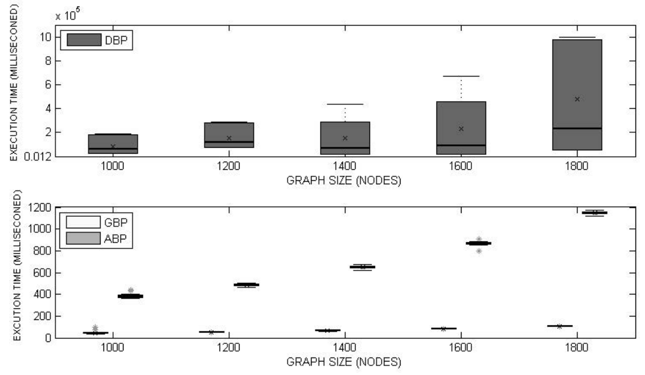

Fig. 8: Impact of graph size on execution time.

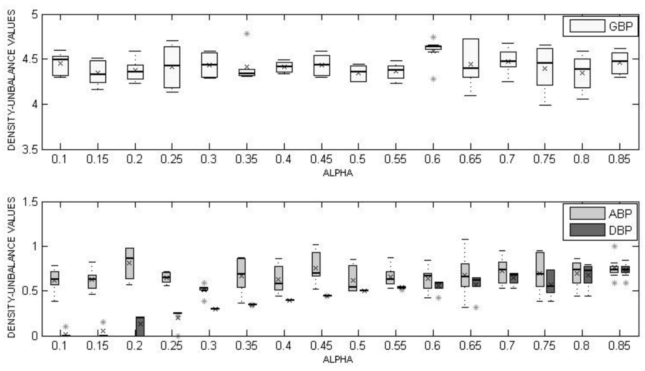

Fig. 9: Impact of α on density-unbalanced value.

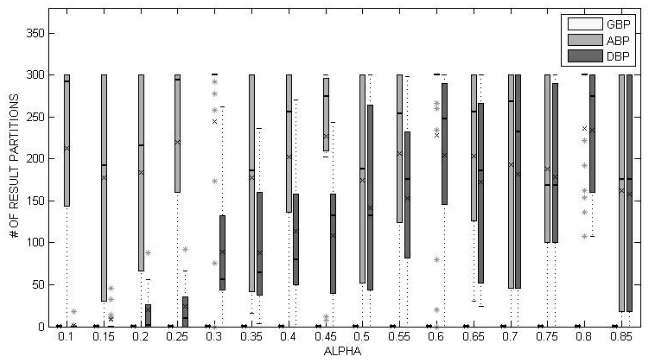

Fig. 10: Impact of α on number of partitions.

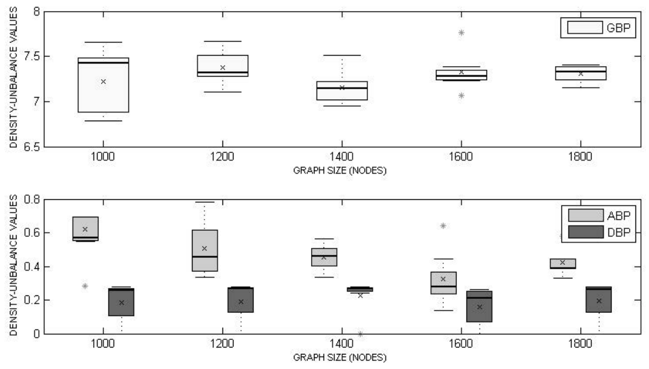

Fig. 11: Impact of graph size on density-unbalanced value.

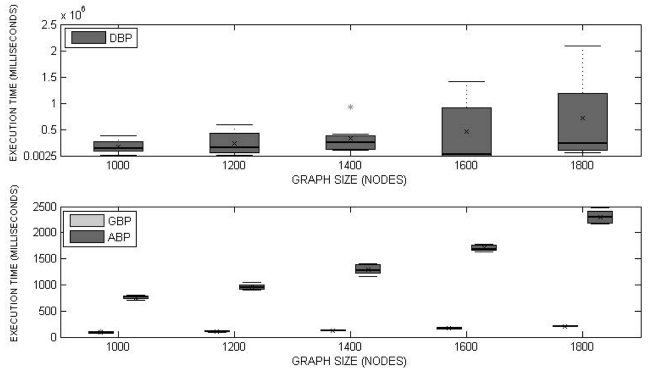

Fig. 12: Impact of graph size on execution times.

execution time. Nevertheless, the complexity of DBP is still polynomial and should be acceptable.

C. Watts-Strogatz Graph Results

This section shows results from Watts-Strogatz graphs.

Figure 9, 10, 11 and 12 show similar results as of Figure 5, 6, 7 and 8, respectively. The experiment results in this section show that DBP does not depend on type of graph.

The experiment results from Section IV-B and IV-C show that DBP can find partitions with smaller density-unbalanced values, compared with ABP and GBP. However, the number of result partitions from DBP might be smaller than that of ABP and GBP because it exposes more constraint on partition filtering. In addition, it consumes more computing power, but still in an acceptable region.

V. Conclusion and Future Work

This paper discusses a graph partitioning problem called $(k, 1+\varepsilon, \alpha)$-balanced problem, which has constrains on the number of components (k), the number of nodes in each component ($1+\varepsilon$) and the different of density in each component (α). As it is an NP-Hard problem, thus, an efficient algorithm, DBP, is proposed to address the problem. The experiment results show that DBP, compared with the other two algorithms, is very effective, i.e., it can generate the solutions with smaller density-unbalanced value for two different types of graph. However, its execution time is higher, but still in an acceptable region. P2P-based PageRank on a web-link graph that is partitioned by DBP will have lower execution time, because there will be no big different on the execution time of each peer. Future work includes an extension of the proposed algorithm where the execution time is smaller while maintaining the effectiveness of the algorithm.

Acknowledgment

The work is partially supported by Coordinating Center for Thai Government Science and Technology Scholarship Students (CSTS), National Science and Technology Development Agency (NSTDA), Thailand.

References

[1] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 107 – 117, 1998.

[2] P. K. Desikan, J. Srivastava, V. Kumar, and P. N. Tan, "Hyperlink analysis - techniques and applications," Army High Performance Computing Center, Technical Report;, 2002.

[3] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM*, vol. 46, no. 5, pp. 604–632, 1999.

[4] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," Stanford University, Technical Report;, 1998.

[5] A. N. Langville and C. D. Meyer, "Deeper inside pagerank," *Internet Mathematics*, vol. 1, p. 2004, 2004.

[6] J. X. Parreira and G. Weikum, "JXP global authority scores in a P2P network," in *Proceedings of the Eight International Work-shop on the Web and Databases (WebDB 2005)*, Baltimore, Maryland, USA, 2005, pp. 31–36.

[7] K. Sankaralingam, S. Sethumadhavan, and J. C. Browne, "Distributed pagerank for p2p systems," in *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*. IEEE Computer Society, 2003, p. 58.

[8] S. Shi, J. Yu, G. Yang, and D. Wang, "Distributed page ranking in structured p2p networks." in *Proceedings of the 2003 International Conference on Parallel Processing*, 2003.

[9] K. Andreev and H. Racke, "Balanced graph partitioning," *Theory of Computing Systems*, vol. 39, pp. 929–939, November 2006.

[10] S. Sangamuang, P. Boonma, and J. Natwichai, "A P2P-based incremental web ranking algorithm," in *Proceedings of the 2011 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, Oct 2011, pp. 123–127.

[11] G. Even, J. S. Naor, S. Rao, and B. Schieber, "Fast approximate graph partitioning algorithms," in *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1997, pp. 639–648.

[12] S. H. S. Duncan J. Watts2, "Collective dynamics of 'small-world' networks," *Nature*, pp. 440–442, June 1998.