Contemporary Engineering Sciences, Vol. 8, 2015, no. 17, 773 - 788
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ces.2015.56193

Task Scheduling for Directed Cyclic Graph Using Matching Technique

W.N.M. Ariffin

Institute of Engineering Mathematics
Universiti Malaysia Perlis
Kampus Pauh Putra 02600 Arau, Perlis, Malaysia

S. Salleh

Universiti Teknologi Malaysia
Center for Industrial and Applied Mathematics, Skudai, Johor, Malaysia

Abstract

The scheduling and mapping of task graph to processors is considered to be the most crucial NP-complete in parallel and distributed computing systems. In this paper, the theoretical graph application using matching is presented to assign a number of tasks onto two processors. This paper addresses a directed-weighted cyclic graph. The effort is to reduce the graph onto directed acyclic graph. A co-comparability graph is presented in order to assign the task onto two processors. Combining several innovative techniques lead to an efficient graph-mapping concept, called *DCGSimplify*. Our simulation model found that the proposed techniques and algorithms are easy to be implemented.

Keywords: Graph theory, Matching, Mapping

1 Introduction

The problem of scheduling a task graph of parallel program onto a parallel and distributed computing system is a well-defined NP-complete problem that has received a large amount of attention. The efficient scheduling of tasks is paramount to maximizing the benefits of executing an application in a distributed

and parallel computing. The directed acyclic graph (DAG) structure frequently occurs in many regular and irregular applications such as LU decomposition, Gaussian elimination, Laplace transforms, fast Fourier transform and instruction level parallelism [1].

The produced schedule is judged based on the performance criterion we are trying to optimize. There are two characteristics used in order to evaluate a scheduling system, which are performance and efficiency. Many studies have been done on complexities, classifications and techniques required for solving tasks assignment and scheduling problems. An optimal task assignment guarantees minimum turnaround time for a given architecture. Several approaches of optimal task assignment have been proposed, ranging from graph partitioning based tools to heuristic graph matching. In an attempt to solve the problem in the general case, a number of heuristics have been introduced. The heuristics do not guarantee an optimal solution to the problem, but they try to find near-optimal solutions most of the time [2].

The previous researches only work on directed acyclic graph. In this paper, we attempt to solve for directed cyclic graph (DCG). Since the problem involves DCG, therefore a new algorithm to reduce the DCG into the form of a DAG should be proposed. It is simpler when no cycle exists in a graph. A *DCGSimplify* algorithm is presented to reduce the graph from DCG to DAG. Then, the graph is mapped to the two processors after applying matching technique. A simulation of different number of nodes is implemented to verify our algorithm.

This paper is an extension of previous work [3] which presents a new approach of solving scheduling task based on matching technique. The rest of the paper is organized as follows: in Section 2 we describe some related works. Section 3 addresses some definitions of classical graph theory. Then, matching and two-processor assignment technique is discussed in section 4. The mathematical modeling based on DCG is introduced in section 5. The results of our work are shown in section 6. Finally, we draw some conclusions and discussion in section 7. The main contribution of this paper is the novelty reduced technique of directed cyclic graph onto acyclic graph through multicolumn nodes arrangement.

2 Related Work

Research on task allocation and scheduling problems began in the 1960's, and has become a popular research topic in the past few decades. Mona M. Arafa and Fatma A. Omara [4] proposed two hybrid genetic algorithms, named Critical Path Genetic Algorithm (CPGA) and Task Duplication Genetic Algorithm (TDGA). The problem involved mapping a DAG for a collection of computational tasks and their data precedence onto a parallel processing system. Both of the algorithms were compared to other existence algorithms. The experimental results showed that CPGA always outperforms the Multi Critical Path algorithm (MCP) and TGDA algorithm outperforms the Duplication Scheduling Heuristic algorithm (DSH) in most cases.

The increasing popularity of graph data in various domains has led to a renewed interest in developing efficient graph matching techniques, especially for processing large graphs. Linhong Zhu et al. [5] studied the problem of approximate graph matching in a large attributed graph. They proposed a novel structure-aware and attribute-aware index to process approximate graph matching in a large attributed graph. They use the index to find a set of best matching paths. From the best matching paths, they compute the best matching answer graph using a greedy algorithm.

Ullman [6] introduced a basic algorithm for subgraph isomorphism, i.e. exact matching. Tsai and Fu [7] studied error-correcting isomorphisms of attributed graphs for image analysis. They extended their pattern deformation model so that numerical attributes and probability distribution can be introduced into primitives and relations in non-hierarchical relational graphs.

Cordella et al. [8] proposed a new algorithm which can handle subgraph isomorphism test efficiently in large graphs. Tong et al. [9,10] proposed an algorithm to find best effort matching in a large graph based on random walk. Another type of work that similar to Linhong Zhu et al. which use index to find matches in a large data graph efficiently, Tian and Patel [11] utilize degree information and neighborhood connectivity to filter unmatched node pairs. Zhang et al. [12] proposed another indexing approach based on graph distance. Their methods are not efficient enough to handle graphs with up to millions or billions of nodes and edges. In the preprocessing stage of [12], given a data graph, they generate a set of intersecting subgraphs may increase significantly when the size of data graph grows. Tian et al. [11] used a maximum weighted bipartite graph matching algorithm during the stage of matching important nodes, which can be costly if the bipartite graph is large.

Gutman and Wagner [13] defined the matching energy of a graph and gave some properties and asymptotic results of the matching energy. Shuli and Weigen [14] characterized the connected graph G with connectivity k has the maximum matching energy by introducing n-matching n-partite graph. Mohan and Gupta [15] established a methodology for heuristic graph matching, but restricted to a small number of test cases.

Motivated by the work done by researches [5, 11, 14, 15], a matching technique to solve task scheduling onto two processor is presented. Compared to the previous works, this paper focused on directed cyclic task graph (DCG).

3 Definitions

It is important to know few types of graph. A directed graph G is an (ordered) pair (V, E) such that V is a finite set of nodes and E is a set of ordered pairs of nodes. That is, if $e \in E$, then $e = (X_i, X_j)$ and $\{X_i, X_j\} \subseteq V$. Fig. 1 shows an example of the directed graph.

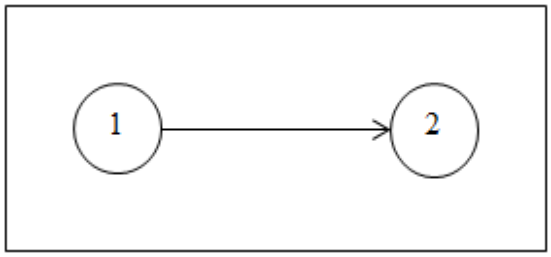
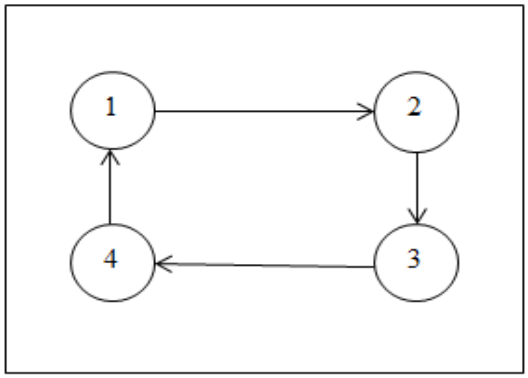
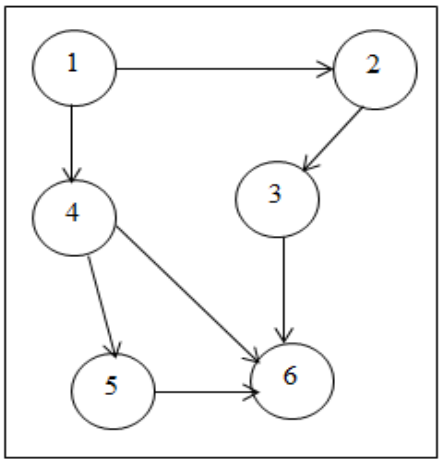


Fig. 1 A directed graph with $V = \{1, 2\}$ and $E = \{(1, 2)\}$.

If G is a directed graph, then a directed cyclic graph (DCG) is a cycle in G consists of a path from a node to itself. However, a directed acyclic graph (DAG) is a graph which does not contain any directed cycle. If $G = (V, A)$, is a DAG containing *n* nodes, then a topological (or ancestral) ordering $(X_1, ..., X_n)$ of the nodes in V is any ordering such that, if $(X_i, X_j) \in A$, then $i < j$. Both DCG and DAG are shown in Fig. 2.



(a)



(b)

Fig. 2 In this graph (a), $[1,2,3,4,1]$ is a directed cyclic and in (b) is a directed acyclic graph.

The co-comparability graph, $G' = (V', A')$ is an undirected graph, that can be defined as $V' = V$ and $A' = \{(i, j)\}$ if there is no path from i to j or from j to i in G. In the mathematical discipline of graph theory, a matching or independent edge set in a graph is a set of edges without common vertices. It may also be an entire graph consisting of edges without common vertices.

Let a task graph G be a DCG composed of N nodes $n_1, n_2, n_3, ..., n_N$. Each node is termed a task of the graph which in turn is a set of instructions that must be executed. A node has one or more inputs. A node with no parent is called source node while a node with no child is called destination node. The graph also has E directed edges representing the communication between the nodes. The weight on an edge is called the communication cost of the edge and denoted by $c(n_i, n_j)$.

4 Matching and two-processors assignment

Matching has several applications in the real world. One of good examples is the personnel assignment application. The problem can be modeled using matching if a bipartite graph G is constructed with bipartition (X, Y), as shown in Fig. 3 where $X = \{x_1, x_2, ..., x_n\}$ and $Y = \{y_1, y_2, ..., y_n\}$. A node x_i is connected to y_j if and only if the candidate x_i is qualified for job y_j. The problem now becomes one of determining whether G having a perfect matching.

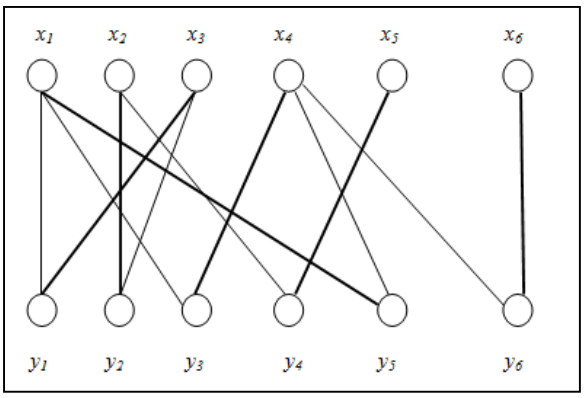


Fig. 3 Personnel assignment problem using matching.

The matching technique presents an interesting relationship between the two processors scheduling problem and the size of maximum matching in the complement of the task graph. This relationship was established by Fuji *et al.* [16] could be expanded to include cases where communication is considered. Matching can be described as given an undirected graph $G = (V, A)$, where V is a set of nodes and A is a set of edges. A subset T of A is called a matching in G if its elements are edges and no two are adjacent in G. The two ends of an edge in T are said to be matched under T. A matching M is a maximum matching if G has no matching T' with $|T'| > |T|$.

The problem that we want to investigate in this paper is to schedule a task graph onto two processors P_1 and P_2 . We present a task graph, a directed cyclic graph initially. The graph is then reduced to a directed acyclic graph, $G = (V, A)$ where V is a set of nodes and A is a set of arcs (edges). We can say that $v \in S$ is a maximal task in S if there does not exist task $v \in S$ such that v precedes u in G. In order to obtain the relationship between matching and scheduling in this study, we propose a co-comparability graph, $G' = (V', A')$. This paper is an extension of our previous work in [3] which consider a problem of finding an optimal schedule for G onto two processors with communication cost, which is to find the maximum number of disjoint task pairs, can be reduced to the maximum matching problem in G' . A co-comparability graph is presented in our work in order to show the matching technique before mapping the nodes onto two processors. In this paper, we verify our proposed algorithm through the simulation model.

5 The model for task scheduling problem

In this paper, a task graph of weighted-directed cyclic graph is presented. The main goal of our study is to obtain a minimum total completion time of a task scheduling. In order to achieve the goal, a good matching algorithm and mapping strategy of task that will be assigned to processor should be implemented. Our approach is shown in Fig. 4.

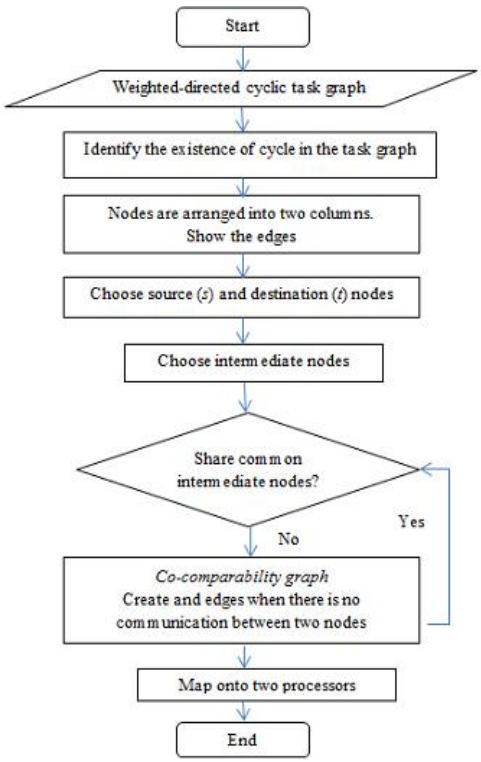


Fig. 4 The flowchart of *DCGSimplify* technique.

Our developed algorithm is very simple and constructed using the following steps:

Step1: Identify the existence of cycle(s) in the task graph.
Step2: All the nodes are arranged in two columns like wrapped-butterfly topology, inspired by DPillar [17].
Step3: A source and destination nodes are chosen from the graph.
Step4: Obtain the path from source to destination nodes such that they do not share common intermediate nodes. The produced graph now is a directed acyclic graph.
Step5: An edge is created when there is no communication between two nodes. We call it as a *co-comparability graph*.
Step6: The connected nodes are then mapped onto two processors simultaneously, while the other tasks are freely mapped to any available processor.

These steps are constructed to solve the problem, named Model 1. The hypothesis on *DCGSimplify* is efficient to construct and the *co-comparability graph* based on the *DCGSimplify* is efficient and scalable to different sizes of graphs. The next section will discuss the implementation of the algorithm and the task graph with same communication cost and computational cost.

5.1 Model 1

The proposed algorithm involves few stages of solution. Firstly, we define the source and destination nodes (in our simulation, we named both nodes as source and target). Next, we want to obtain a DAG by choosing the intermediate nodes. Then, a co-comparability graph is obtained and matching technique is applied to the task graph. The *DCGSimplify* algorithm is outlined in Algorithm 1.

Algorithm 1: DAG and apply matching technique

Given a graph $G(V,E)$, obtain a DAG, $G(V,A)$.

Construct its co-comparability graph, $G'(V',A')$.

In G', obtain a set of edges in maximum matching M.

Let $S_1 \leftarrow V, M_1 \leftarrow M$ and $i \leftarrow 1$.

for $i = 1$ to $|V| - |M|$

if there exists a maximal task u in S_i that is not included in any of the task pairs of M_i, then

$\alpha_i \leftarrow u$

$S_{i+1} \leftarrow S_i - \{u\}$

$M_{i+1} \leftarrow M_i$

if there exists a task pair (u,v) in M_i such that u and v are maximal in S_i, then

$\alpha_i \leftarrow (u, v)$

$S_{i+1} \leftarrow S_i - \{u, v\}$

$M_{i+1} \leftarrow M_i - \{u, v\}$

if there are two task pairs (u_1, u_2) and (v_1, v_2) in M_i such that u_1 and v_1 are maximal in S_i, u_2 and v_2 are independent, then

$\alpha_i \leftarrow (u_1, v_1)$

$S_{i+1} \leftarrow S_i - \{u_1, v_1\}$

$M_{i+1} \leftarrow M_i \cup \{(u_2, v_2)\} - \{(u_1, v_1)\}$

//Map the task graph onto processors.

Each α_i can be mapped according to its indexed number or can be mapped simultaneously by two processors when α_i is a pair of tasks.

An arbitrary task graph, with eight number of nodes is proposed as in Fig. 5. To improve the flexibility of the propose task graph graph, multicolumn (multilayer) of the graph is constructed by replicating the single graph , times. In this study, two-column of task graph, i.e. is presented. The problem is stated as follows:

Given a graph $G(8, E, 2)$, *how the tasks can be mapped onto two available processors?*

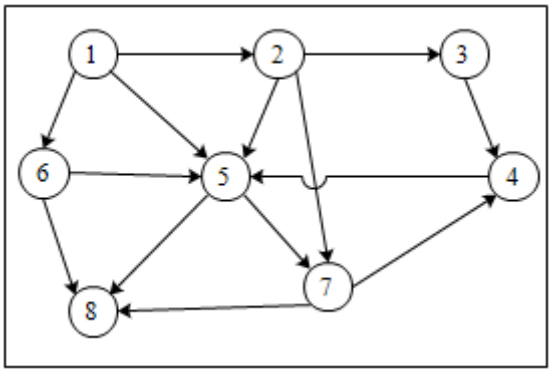


Fig. 5 Model 1: A DCG with eight number of nodes.

To solve the problem, a matching technique is presented for the directed multicolumn graph. In our approach, we begin by arranging the nodes in two-column. The purpose of doing this is as alternative way of finding a directed acyclic graph. The nodes of graph as shown in Fig. 5 is then arranged as in Fig. 6. The communication (edges) between the nodes also presented in Fig. 6.

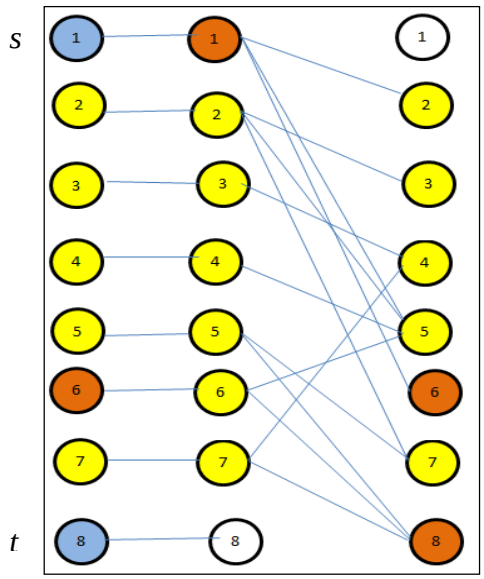


Fig. 6 Nodes are arranged in duplicated column/muticolumn.

The next step requires us to find the directed acyclic graph. As an example, we want to find the shortest path from the source node 1, denote as *s* to the destination node 8, denote as *t*. So, the path from *s* to *t* must be found such that all nodes must be chosen and they must not share common intermediate nodes. As can be seen from Fig. 6, there are three nodes can be chosen from the source node 1. Let say, node 2 is chosen from node 1. Next, from node 2, node 3, 5 and 7 can be reached. Next, node 3 is chosen. From node 3, only node 4 can be chosen. Next, node 5 is chosen from node 4. After that, node 7 is chosen from node 5. Note that there are two nodes that are not chosed yet. So, again from the source node, node 6 is chosen. From node 6, either node 5 or node 8 can be chosen. Notice that, same intermediate nodes are not allowed. Thus, node 5 is strictly not available to be chosed. Next, from node 6, node 8 can be reached, which is the destination node. Now, the directed acyclic graph is obtained. It is illustrated as in Fig. 7.

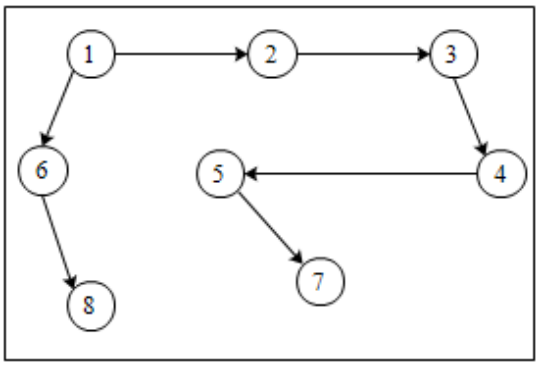


Fig. 7 The reduced directed cyclic graph to directed acyclic graph.

Next, we need to map the tasks onto processor. In order to map the tasks onto two processors, the co-comparability of the graph should be obtained. Based on the definition stated in previous section, an edge is created when there is no path or communication from *u* to *v*. Since there is no path between nodes 5-6, 5-8 and 7-8, therefore we create a path to connect node 5 and node 6 also node 7 and node 8. Finally, the co-comparability graph can be illustrated as follows:

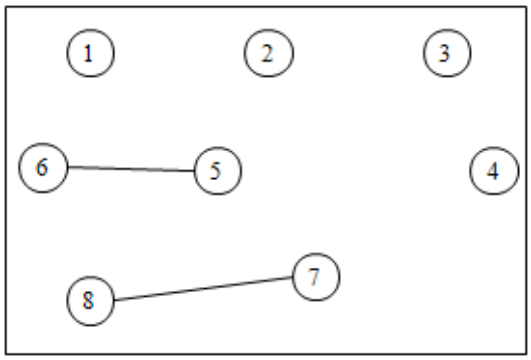
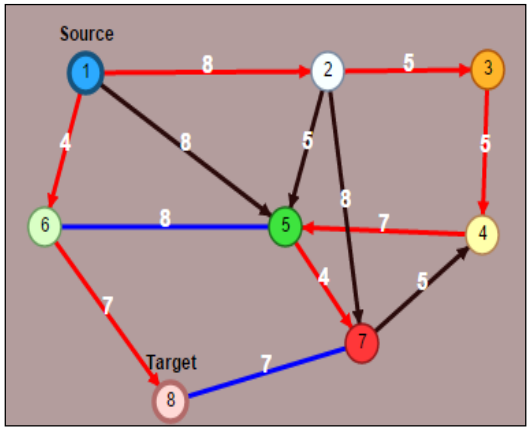


Fig. 8 The co-comparability of the directed acyclic graph.

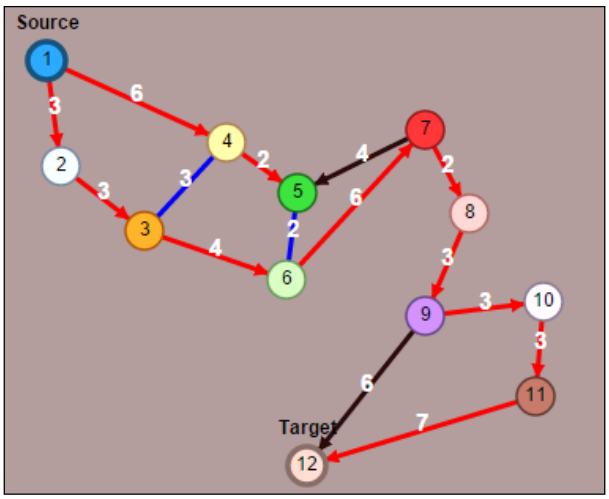
From Fig. 8, we can map the nodes onto processors. Node 5 and node 6 are mapped onto two different processors, and the same to node 7 and node 8. The remainders are freely assigned to available processor.

6 Results

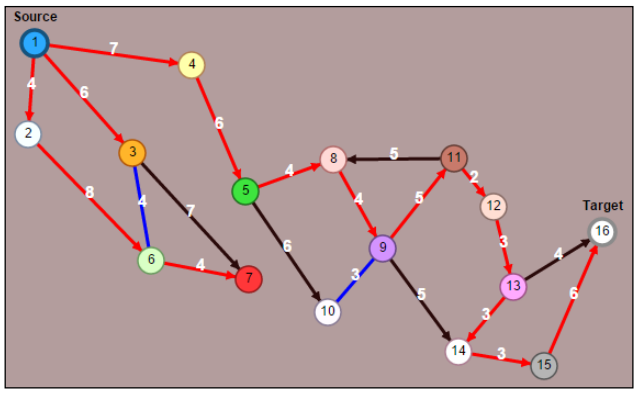
To evaluate our proposed algorithm, we have implemented it using Intel(R) Core(TM) i5 (2.3 GHz) using Javascript programming language with D3 library. The algorithm is applied to test for several number of nodes. In this paper, we show our simulation model for eight, twelve, sixteen and twenty number of nodes. Our simulation model is able to solve for a large number of nodes. After we run the simulation model, we obtained the results as follows:



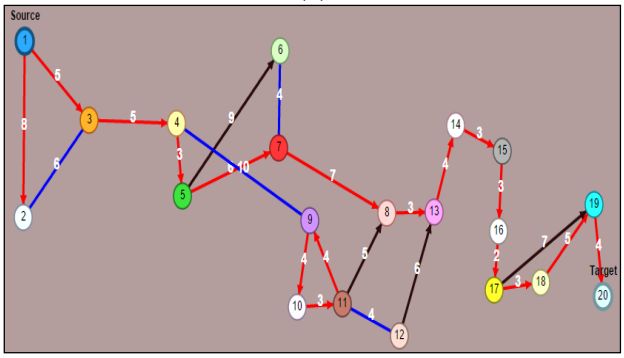
(a)



(b)



(c)

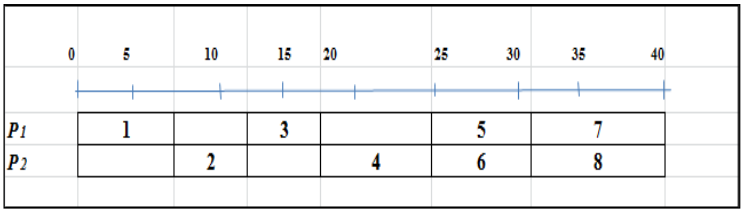


(d)

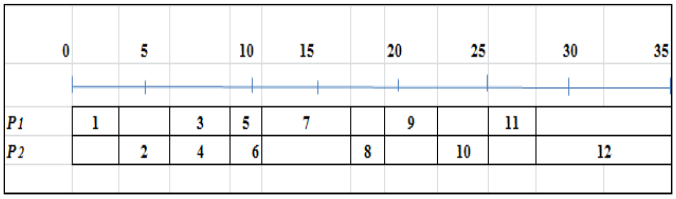
Fig. 9 The DCG to DAG colored with red line and matching colored with blue line for 8, 12, 16 and 20 nodes in (a), (b), (c) and (d) respectively.

Fig. 9 (a) shows eight nodes with communication cost. For example, we have $c(n_1, n_2) = 6$. Initially we have a directed cyclic graph. Then, our algorithm tries to find the directed acyclic graph (red line). Then, we apply the co-comparability and

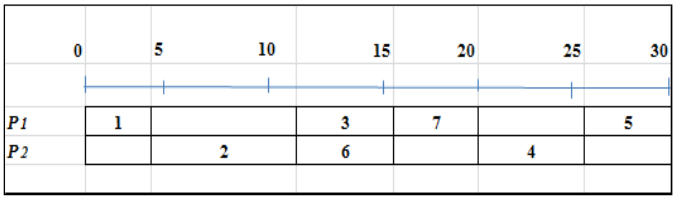
matching colored with blue line. Based on Fig. 9, we map the nodes onto two processors as follows:

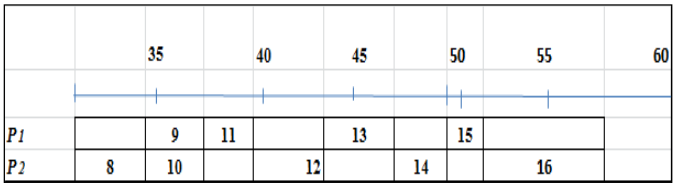


(a)

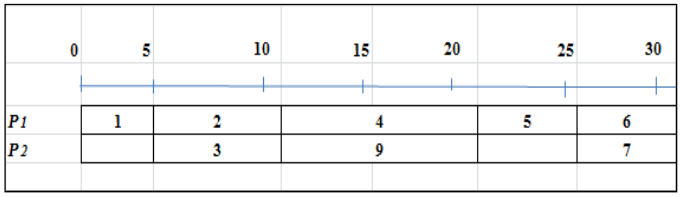


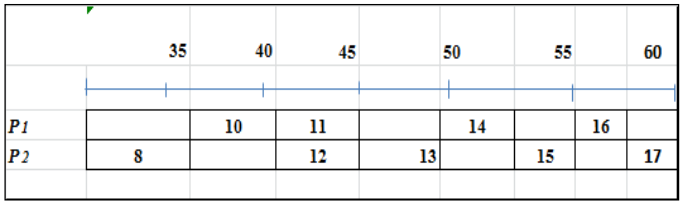
(b)

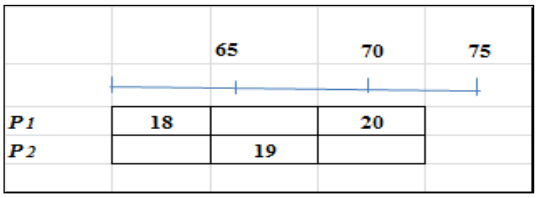




(c)







(d)

Fig. 10 The schedule length for (a) eight nodes, (b) twelve nodes, (c) sixteen nodes and (d) twenty nodes.

From the results obtained in Fig. 10, we can tabulate the total length (communication cost) to complete all the tasks as shown in Table 1.

Table 1. Total length of completing all tasks.

Number of Nodes (Tasks)	**Total Length(Unit)**
8	40
12	35
16	57
20	72

7 Conclusion and Discussion

Based on the work done in this paper, we can conclude that the technique of matching for a directed cyclic graph onto two processors is easy to implement and

efficient. This is the novelty of this paper. We begin finding the solution from the directed cyclic graph. The quality techniques used for finding the acyclic graph and co-comparability graph from the directed cyclic graph led to an efficient graph-mapping concept. The technique is valid and applicable for larger number of nodes. Our next paper will put an attention on graph partitioning and map it onto $n > 2$ processors.

Acknowledgements. This work was supported by the Ministry of Education, Malaysia for Research Acculturation Grant Scheme (RAGS) under project code 9018-00036.

References

[1] S. Baskiyar and C. Dickinson, Scheduling Directed A-cyclic Task Graphs on a Bounded Set of Heterogeneous Processors Using Task Duplication, *Journal Parallel Distrib. Comput.*, **65** (2005), 911 - 921. http://dx.doi.org/10.1016/j.jpdc.2005.01.006

[2] H. El-Rewini and T.G. Lewis, *Introduction to Parallel Computing*, Prentice-Hall, 1994.

[3] W. N. M. Ariffin and S. Salleh, The Matching Technique of Directed Cyclic Graph for Task Assignment Problem, *AIP Conference Proceedings*, **1635** (2014), 387. http://dx.doi.org/10.1063/1.4903612

[4] Mona M. Arafa and Fatma A. Omara, Genetic Algorithms for Task Scheduling Problem, *Journal Parallel Distrib. Comput.*, **70** (2010), 13 - 22. http://dx.doi.org/10.1016/j.jpdc.2009.09.009

[5] Linhong Zhu, Wee Keong Ng, James Cheng, Structure and Attribute Index for Approximate Graph Matching in Large Graphs, *Information Systems*, **36** (2011), 958 - 972. http://dx.doi.org/10.1016/j.is.2011.03.009

[6] J.R. Ullmann, An Algorithm for Subgraph Isomorphism, *The Journal of the ACM*, **23** (1976), no. 1, 31 - 42. http://dx.doi.org/10.1145/321921.321925

[7] W. H. Tsai, K. Fu, Error-correcting Isomorphisms of Attributed Relational Graphs for Pattern Analysis, *IEEE Transaction on Systems, Man and Cybernetics*, **9** (1979), no. 12, 757-768. http://dx.doi.org/10.1109/tsmc.1979.4310127

[8] L.P. Cordella, P. Foggia, C. Samsone, M. Vento, A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, **26** (2004), no. 10, 1367 - 1372.
http://dx.doi.org/10.1109/tpami.2004.75

[9] H. Tong, C. Faloutsos, B. Callagher, T. Eliassi-Rad, Fast Best-Effort Pattern Matching in Large Attributed Graphs, *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, New York, NY, USA, (2007), 737 - 746.
http://dx.doi.org/10.1145/1281192.1281271

[10] H. Tong, C. Faloutsos, Center-piece Subgraphs: Problem Definition and Fast Solutions, *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and data Mining*, ACM, New York, NY, USA, (2006), 404 - 413. http://dx.doi.org/10.1145/1150402.1150448

[11] Y. Tian, J.M. Patel, Tale: A Tool for Approximate Large Graph Matching, *Proceedings of the 24th International Conference on Data Engineering*, *IEEE Computer Society*, Washington, DC, USA, (2008), 963 - 972.
http://dx.doi.org/10.1109/icde.2008.4497505

[12] S.Zhang, S. Li, J. Yang, Gaddi, Distance Index Based Subgraph Matching in Biological Networks, *Proceedings of the 12th International Conference on Extending Database Technology ACM*, New York, NY, USA, (2009), 192 - 203.
http://dx.doi.org/10.1145/1516360.1516384

[13] I. Gutman, S. Wegner, The Matching Energy of a Graph, *Discrete Appl. Math.*, **160** (2012), 2177 - 2187.
http://dx.doi.org/10.1016/j.dam.2012.06.001

[14] L. Shuli, Y. Weigen, The Matching Energy of Graphs with Given Parameters, *Discrete Applied Mathematics*, **162** (2014), 415 - 420.
http://dx.doi.org/10.1016/j.dam.2013.09.014

[15] R. Mohan, A. Gupta, Graph Matching Algorithm for Task assignment Problem, I*nternational Journal of Computer Science, Engineering and Applications* (IJCSEA), **1** (2011), no. 6, 105 - 118.
http://dx.doi.org/10.5121/ijcsea.2011.1609

[16] M. Fujii, T. Kasami, K. Ninomiya, Optimal Sequencing of Two Equivalent Processors, *SIAM Journal of Appl. Math.*, **17** (1969), no. 4, 784 - 789.
http://dx.doi.org/10.1137/0117070

[17] Y. Liao, J. Yin, D. Yin, L. Goa, D. Pillar: Dual-port Server Interconnection Network for Large Scale Data Centers, *Computer Networks*, **56** (2012), 2132 - 2147. http://dx.doi.org/10.1016/j.comnet.2012.02.016

Received: July 11, 2015; Published: September 1, 2015