Performance Evaluation of Distributed Maximum Weighted Matching Algorithms

Can Umut Ileri
International Computer Institute
Ege University, Izmir, Turkey
Email: can.umut.ileri@ege.edu.tr

Orhan Dagdeviren
International Computer Institute
Ege University, Izmir, Turkey
Email: orhan.dagdeviren@ege.edu.tr

Abstract*—Graph matching is a fundamental graph theory problem which has a broad application range including information retrieval, pattern recognition, graph partitioning, chemical structure analysis, protein function prediction, backup placement and cellular coverage. This problem has gained attention in distributed computing as there are distributed matching algorithms with asymptotically guaranteed time bounds and approximation ratios. On the other side, we do not know the practical performance of these algorithms. In this paper, we provide a detailed performance evaluation of asynchronous distributed maximum weighted matching (MWM) algorithms. We assume a message-passing system in $\mathcal{CONGEST}$ model in which the message size is limited to $O(\log n)$ where n is the number of nodes. This model is popular for energy-efficient networks such as wireless sensor networks. We used a discrete event simulator, SimPy, to model the assumed network structures. We provide the implementations of Watthenhofer and Wattenhofer's algorithm, Hoepman's algorithm, Lotker et al.'s algorithm and Lotker et al.'s improvement algorithm. The results show that the greedy algorithm of Hoepman performed best in approximating the optimum result in all types of networks, even achieving an approximation ratio of 0.99 in some instances. To the best of our knowledge this is the first study which provides an extensive performance evaluation of distributed MWM algorithms.

***Keywords*—Graph Matching, Distributed Computing, Performance Evaluation.**

I. Introduction

Given a graph $G(V, E)$, the purpose of a matching algorithm is to find a subset of edges $M \subseteq E$ such that any vertex in V is incident to at most one edge in M. The problem of maximizing the number of edges in M is called the *maximum cardinality matching problem* (MCM). Given a function $w(e) \in \mathbf{R}^+$ where $e \in E$, the total weights of edges in M is called the weight of the matching. The problem to find the heaviest matching is called the *maximum weighted matching problem* (MWM). Matching problem has many variations (b-matching, bipartite matching) and various applications such as pattern recognition [1], information retrieval [2], graph partitioning [3]–[6], chemical structure analysis [7], protein function prediction [8], backup placement [9] and cellular coverage [10].

MWM can be computed in polynomial time in centralized settings by the famous Blossom Algorithm of Edmonds [11]. For an extensive summary of centralized matching algorithms, we refer the readers to the recent work of Lotker et al. [12]. Matching problem is studied much in distributed settings. Considering unweighted graphs, Israeli and Itai [13] proposed a $\frac{1}{2}$-approximation MCM algorithm which is still the best algorithm in terms of worst case complexity. The basic idea of the algorithm is to reduce the graph into a sparse one where each node has a degree of at most 2, and to select edges on this reduced graph. Reductions and selections are done randomly. Reduction is realized as follows: Each node randomly selects one of its neighbors and sends a *matching proposal* it. Having received at least one proposal, the node randomly accepts exactly one of them and rejects the others. Accepted edges remain in the graph and the others are neglected until the next round. Hence, each node has at most two active edges. After reduction, selection is realized as follows: Each node randomly selects one of (at most) two active edges. If an edge is selected by both of its incident nodes, the nodes are considered *matched*. Remaining nodes continue the same procedure until they are matched or do not have any active neighbor. The resulting matching is maximal, i.e. all of the unmatched edges is incident to a matched edge. The expected running time of the algorithm is $O(\log n)$. For a review of distributed MCM algorithms we refer readers to [12].

For the distributed MWM problem, which is the main focus of our work, many algorithms have been introduced in the field. Wattenhofer&Wattenhofer's distributed weighted matching algorithm [14] (hereinafter called Wattenhofer's Algorithm or simply Wattenhofer) applies Israeli&Itai's approach to the weighted graphs. Basically, locally-light edges are removed from the graph and a matching strategy similar to Israeli&Itai's approach are applied over the remaining edges. The algorithm is proved to guarantee at least $\frac{1}{5}$ of the maximum weighted matching. On the deterministic side of distributed MWM algorithms is Hoepman's greedy algorithm [15]. Each node greedily tries to match through the edge having the heaviest weight. Locally-heaviest edges (which are heavier than all of its incident edges) are guaranteed to be in the matching set with this approach. Once its heaviest neighbor is matched with another node, the node tries to match with the second heaviest and so on, until it is matched with any one or does not have any unmatched neighbor. This approach guarantees a $\frac{1}{2}$-approximation at the cost of a time complexity of $O(m)$, where m is the number of edges. Lotker et al. [16] (hereinafter called Lotker2009) combines Hoepman's deterministic and Wattenhofer's randomized approaches. Their algorithm divides the edges into classes and subclasses according to the

TABLE I
THEORETICAL COMPARISON OF DISTRIBUTED MWM ALGORITHMS IN $\mathcal{CONGEST}$ MODEL

	Approx.	**Time**	**Msg.**	**Global Requirement**
Wattenhofer [14]	$\frac{1}{5}$	$O(\log^2 n)$	$O(n^2 \log^2 n)$	none
Hoepman [15]	$\frac{1}{2}$	$O(m)$	$O(m)$	none
Lotker2009 [16]	$\frac{1}{4} - \frac{\epsilon}{5}$	$O(\log n)$[a]	$O(m)$[b]	α, β, $max(w_e)$
Lotker2015 [12]	$\frac{1}{2} - \frac{\epsilon}{5}$	$O(\log n)$[a]	$O(m)$[b]	α, β, $max(w_e)$

[a] In synchronous settings.
[b] This complexity is of our implementation, as it was not provided on the paper.

weights of edges and applies a randomized algorithm similar to Israeli&Itai's and Wattenhofer's approaches. Since the classes and subclasses does not have common edges, the matching set of each class can be determined in parallel. The algorithm then combines the resulting matchings of classes into a single maximal matching by deterministically choosing one of the matched edges according to their weights. This approach guarantees a $(\frac{1}{4} - \frac{\epsilon}{5})$-approximation in $O(\log n)$ time. Lotker et al. proposed a randomized synchronous $(\frac{1}{2} - \frac{\epsilon}{5})$-approximation algorithm in [12] (hereinafter called Lotker2015) which basically improves the total weight of any given matching (including an empty matching). According to the algorithm, given a weighted graph $G(V, E)$ and a valid matching M, each unmatched edge has an *augmenting gain*, i.e. the increase in the total weight of the matching when this edge is put into M and all of its incident matched edges are removed from M. A new weight function $w'(e)$ is defined such that $w'(e)$ is equal to the augmenting gain of edge e. The augmenting gain of a matched edge is considered to be 0. If any MWM algorithm runs on graph $G(V, E)$ using the new weight function $w'(e)$, then the resulting matching M' signifies an increase in the given matching M. When M and a nonempty M' are *wrapped* into a single matching, giving priority to M', the resulting matching M'' is heavier than M. A summary of theoretical complexities of the distributed MWM algorithms is in Table I

It is important to note that the original algorithms of Lotker2009 and Lotker2015 assume synchronous network model. We implemented this algorithms in asynchronous network model; i.e. the system has neither a central scheduler nor a fixed time frame for phases. We allow each node to enter the next phase of the algorithm without waiting for other nodes to finish the phase. Each message carries information about its phase. A control mechanism exists for checking if a received message belongs to the actual phase. If not, it is put in the waiting queue to be executed later.

All of these algorithms run on $\mathcal{CONGEST}$ model [17] in which communications between nodes are realized with *messages* having size of $O(\log n)$ bits. Because of the limited message size property, this model is very popular for energy-efficient networks such as wireless sensor networks.

A. Our Contribution and Organization

In this work, we implement the aforementioned four distributed maximum weighted matching algorithms on a discrete event simulator (SimPy [18]), and compare their performances with respect to approximation ratio, running time and total message count. We omit the distributed approaches which use large messages [19]. To the best of our knowledge, this is the first study which provides the implementation and performance evaluation of distributed weighted matching algorithms. In Section II, we describe the system model. In Section III, we explain our experimental setup. We provide and discuss test results in Section IV. Finally, conclusions are drawn in Section V.

II. THE SYSTEM MODEL

Given a graph $G(V, E)$, we consider a network with n nodes and m edges, where $n = |V|$ and $m = |E|$. We assume a *fully-distributed* $\mathcal{CONGEST}$ network model [17] where each node runs algorithms *asynchronously*. Nodes are connected with weighted edges and each node has knowledge about the weight of its incident edges and the id of its neighbors. A *round* in our asynchronous model stands for a *time step*. We measure the running time of the algorithms according to the following rules:

- If a node v_i sends a message to its neighbor v_j at time t, v_j receives the message at time $t + 1$.
- If a node receives a message at time t, it tries to execute the message at time t. If the received message cannot be executed at that time for any reason (i.e. the node is waiting for another type of message or the message belongs to a future phase of the algorithm), it is delayed for one unit of time.
- Each node starts to execute algorithms at time $t_0 = 1$.

Each node $v_i \in V$ has a pointer p_i which holds the id of its matching partner. If two neighbors point to each other, they are considered *matched*. If a node is not *matched*, its pointer value is NULL.

III. EXPERIMENTAL SETUP

We implemented the algorithms on SimPy [18], a discrete event simulator written in Python programming language. It makes use of the coroutine-like functionality of Python language which enables a function to suspend and resume execution at certain locations [20].

We considered each node as a *process* in SimPy. We built a message passing model which uses the shared resources capability of SimPy. A central process, as we call Message-Manager, is another process which manages the transmission of messages.

We used 3 types of topology generation methods for our networks:

- *Erdös-Rényi* model: Random network model with low clustering [21].
- *Watts-Strogatz* model: Small-world random networks with high clustering [22].

TABLE II
Approximation Ratio vs. Number of Nodes, Weight Distribution: Uniform. Density is fixed at 0.1 for Erdös-Renyi and Watts-Strogatz networks, and ranges from 0.2 to 0.05 as the node count increases in geometric networks.

	Erdös-Renyi				**Watts-Strogatz**				**Geometric**			
N	**Hoepman**	**Watten**	**Lot2009**	**Lot2015**	**Hoepman**	**Watten**	**Lot2009**	**Lot2015**	**Hoepman**	**Watten**	**Lot2009**	**Lot2015**
100	0.91	0.77	0.69	0.71	0.93	0.80	0.69	0.74	0.94	0.77	0.61	0.74
200	0.93	0.76	0.62	0.74	0.94	0.75	0.63	0.72	0.94	0.76	0.61	0.74
300	0.94	0.76	0.58	0.74	0.95	0.77	0.58	0.74	0.94	0.76	0.62	0.73
400	0.95	0.75	0.54	0.74	0.95	0.76	0.56	0.75	0.94	0.77	0.62	0.74
500	0.96	0.75	0.51	0.76	0.96	0.75	0.52	0.75	0.94	0.77	0.62	0.71
600	0.96	0.75	0.50	0.76	0.96	0.75	0.49	0.63	0.94	0.77	0.62	0.72
700	0.96	0.75	0.48	0.77	0.97	0.76	0.48	0.71	0.94	0.77	0.64	0.72
800	0.97	0.75	0.46	0.77	0.97	0.76	0.46	0.78	0.93	0.77	0.63	0.73
900	0.97	0.76	0.46	0.76	0.97	0.75	0.42	0.72	0.93	0.77	0.64	0.73
1000	0.97	0.75	0.44	0.79	0.97	0.75	0.44	0.78	0.93	0.77	0.64	0.72

TABLE III
Approximation Ratio vs. Number of Nodes, Weight Distribution: Gauss. Density is fixed at 0.1 for Erdös-Renyi and Watts-Strogatz networks, and ranges from 0.2 to 0.05 as the node count increases in geometric networks

	Erdös-Renyi				**Watts-Strogatz**				**Geometric**			
N	**Hoepman**	**Watten**	**Lot2009**	**Lot2015**	**Hoepman**	**Watten**	**Lot2009**	**Lot2015**	**Hoepman**	**Watten**	**Lot2009**	**Lot2015**
100	0.93	0.75	0.84	0.85	0.94	0.76	0.86	0.88	0.96	0.74	0.76	0.84
200	0.96	0.74	0.79	0.84	0.96	0.74	0.79	0.84	0.96	0.74	0.77	0.84
300	0.96	0.73	0.74	0.83	0.97	0.74	0.75	0.83	0.96	0.74	0.78	0.84
400	0.97	0.73	0.71	0.83	0.97	0.72	0.71	0.83	0.96	0.74	0.78	0.84
500	0.97	0.72	0.65	0.85	0.97	0.72	0.67	0.83	0.96	0.76	0.79	0.84
600	0.98	0.71	0.65	0.85	0.98	0.71	0.65	0.84	0.96	0.75	0.79	0.84
700	0.98	0.70	0.61	0.85	0.98	0.71	0.61	0.84	0.96	0.75	0.79	0.83
800	0.98	0.71	0.59	0.84	0.98	0.70	0.60	0.85	0.96	0.75	0.81	0.84
900	0.98	0.70	0.58	0.85	0.98	0.70	0.58	0.85	0.96	0.75	0.81	0.85
1000	0.98	0.70	0.56	0.85	0.98	0.70	0.57	0.85	0.96	0.76	0.81	0.85

- *Geographical network model*: Given a graph $G(V, E)$ and coordinates for each node in V, any two nodes are considered connected if their euclidean distance is smaller than a constant value r.

We used NetworkX library [23] for generating *Erdös-Rényi* and *Watts-Strogatz* models. We also used its maximum weighted matching module for finding the optimum matching, which is an efficient implementation of Edmond's exact algorithm [11] and presented in [24].

For each topology generated according to the models above, we assigned edge weights randomly using two different distributions:

- Uniform distribution with minimum weight of 10 and maximum weight of 100
- Gaussian distribution with ($\mu = 50, \sigma = 12$).

For *Erdös-Rényi* and *Watts-Strogatz* models, we generated networks with three different density values: 0.05, 0.1 and 0.2.

For each 3-tuple of topology, weight distribution and densities, we created 10 network instances.

For Lotker2009 and Lotker2015, we assumed that $\epsilon = 1/4$, which makes $\alpha = 5$ and $\beta = \frac{5}{4}$.

IV. Computational Results

A. Approximation ratio

Tables II and III provide approximation ratios of algorithms for both uniform and Gaussian weight distributions, respectively. Density is fixed at 0.1 for small-world networks and ranges from 0.2 to 0.05 as the graph size increases in geometric networks. Hoepman's greedy algorithm performs best in all types of graphs and weight distributions with an outstanding approximation ratio around 0.97, which, in other words, means that the greedy algorithm is able to approximate the optimum matching value with an error of 3%. Interestingly, its approximation ratio increases as the network size gets greater, especially in Erdös-Rényi and Wattz-Strogatz networks.

When the weights are distributed uniformly, the approximation ratios of Wattenhofer's Algorithm and Lotker2015 are similar and around 0.77. However, in case of normally distributed weights, Lotker2015's approximation ratio increases to 0.85, while Wattenhofer's performance decreases to 0.7. Recall that Lotker2015 is able to detect augmenting paths of length 3 with positive weight gain. Also note that the augmenting gain of the unmatched edges is more likely to be positive in case of normally distributed edge weights since the weights of incident edges are expected to be closer to each other. These two observations explain the relative success of Lotker2015 over Wattenhofer's when the weights are distributed normally.

In small-world networks (Erdös-Rényi and Wattz-Strogatz), Lotker2009's approximation ratio is close to that of Wattenhofer's and Lotker2009's when the size of the graph is less than 200. However its performance drops significantly in larger graphs.

TABLE IV
COMPUTATION TIME VS. NUMBER OF NODES, WEIGHT DISTRIBUTION: **UNIFORM**. DENSITY IS FIXED AT 0.1 FOR ERDÖS-RENYI AND WATTS-STROGATZ NETWORKS, AND RANGES FROM 0.2 TO 0.05 AS THE NODE COUNT INCREASES IN GEOMETRIC NETWORKS.

	Erdös-Renyi				**Watts-Strogatz**				**Geometric**			
N	**Hoepman**	**Watten**	**Lot2009**	**Lot2015**	**Hoepman**	**Watten**	**Lot2009**	**Lot2015**	**Hoepman**	**Watten**	**Lot2009**	**Lot2015**
100	30.2	33.0	125.9	353.2	27.8	33.8	110.9	335.1	58.9	45.9	222.0	959.0
200	48.2	54.3	203.2	877.3	43.8	50.4	184.5	801.7	63.4	52.4	233.1	1124.7
300	66.9	68.7	282.7	1468.9	62.7	67.8	250.7	1357.9	61.2	60.1	224.4	1086.2
400	85.6	88.7	373.7	2113.4	79.9	79.9	352.3	1940.0	57.2	53.6	215.2	992.9
500	102.6	109.7	461.4	2589.7	94.6	99.0	448.9	2406.3	56.1	51.1	214.2	920.3
600	121.4	116.4	484.0	3142.2	111.5	118.7	468.4	2526.0	56.0	53.2	208.2	951.7
700	136.8	131.9	621.6	3712.7	128.6	130.8	574.9	3165.0	54.2	53.5	216.9	905.1
800	152.7	138.7	718.3	4296.1	144.1	148.1	683.5	3938.4	54.9	50.9	206.1	889.0
900	172.0	154.3	884.0	4323.9	158.0	152.5	726.6	4320.3	49.5	53.9	200.6	814.6
1000	190.8	170.9	874.6	5064.6	173.7	167.3	822.5	5132.0	49.5	52.2	210.6	796.8

TABLE V
COMPUTATION TIME VS. NUMBER OF NODES, WEIGHT DISTRIBUTION: **GAUSS**. DENSITY IS FIXED AT 0.1 FOR ERDÖS-RENYI AND WATTS-STROGATZ NETWORKS, AND RANGES FROM 0.2 TO 0.05 AS THE NODE COUNT INCREASES IN GEOMETRIC NETWORKS

	Erdös-Renyi				**Watts-Strogatz**				**Geometric**			
N	**Hoepman**	**Watten**	**Lot2009**	**Lot2015**	**Hoepman**	**Watten**	**Lot2009**	**Lot2015**	**Hoepman**	**Watten**	**Lot2009**	**Lot2015**
100	30.5	36.2	177.6	396.0	28.0	33.0	155.9	329.6	61.7	41.6	308.5	1014.0
200	48.6	50.1	306.4	934.1	46.2	53.1	282.1	795.7	64.3	55.1	368.1	1176.5
300	68.3	73.4	443.2	1568.8	60.3	68.5	415.0	1478.4	59.9	61.7	351.3	1128.9
400	84.9	91.1	529.1	2226.5	78.9	91.9	535.4	2102.6	58.4	53.8	320.8	1245.8
500	102.2	106.6	658.8	3911.6	95.3	107.8	633.0	3038.2	57.2	53.6	311.2	1124.4
600	121.8	117.6	741.2	5078.4	109.2	117.8	774.8	4570.0	54.8	52.0	304.4	1024.2
700	135.4	131.6	920.6	6289.8	126.2	135.4	871.6	5443.6	55.6	53.4	302.2	1058.0
800	155.8	157.0	967.2	6975.0	144.2	154.6	960.4	6751.8	49.6	48.0	279.4	1018.0
900	170.4	154.2	1088.0	8181.4	158.2	165.0	1116.0	8032.4	49.2	54.4	309.4	1002.4
1000	185.0	170.2	1255.4	9211.0	174.8	185.4	1172.4	8547.8	51.2	54.8	282.6	984.4

TABLE VI
TOTAL MESSAGE COUNT VS. NUMBER OF NODES, WEIGHT DISTRIBUTION: **UNIFORM**. DENSITY IS FIXED AT 0.1 FOR ERDÖS-RENYI AND WATTS-STROGATZ NETWORKS, AND RANGES FROM 0.2 TO 0.05 AS THE NODE COUNT INCREASES IN GEOMETRIC NETWORKS

	Erdös-Renyi				**Watts-Strogatz**				**Geometric**			
N	**Hoepman**	**Watten**	**Lot2009**	**Lot2015**	**Hoepman**	**Watten**	**Lot2009**	**Lot2015**	**Hoepman**	**Watten**	**Lot2009**	**Lot2015**
100	751.5	3435.4	5073.4	9050.6	743.6	3546.8	5052.0	9034.8	1711.8	8843.7	10091.4	19710.9
200	2768.6	13587.1	16845.1	32444.7	2807.7	13932.0	17053.0	32873.5	3421.8	17310.7	20201.9	39547.4
300	6081.4	31313.8	34993.5	70070.0	6110.3	31391.7	35001.0	70674.6	4888.6	24962.1	29282.8	56624.8
400	10858.6	56585.5	59137.1	122301.7	10828.8	56412.8	59852.2	122371.7	6047.1	29844.5	36487.4	70193.5
500	16896.9	88608.6	88441.7	187426.1	16811.8	87747.6	89032.5	187685.5	7135.9	35565.1	43230.5	81412.5
600	24055.1	126879.3	124530.7	267690.7	24055.0	126962.2	124300.0	239513.2	8137.3	40716.5	49671.2	95599.8
700	32494.2	170849.8	165867.2	361087.5	32291.0	173115.8	165926.6	342774.1	9130.1	45719.8	56517.7	107760.8
800	42251.8	226209.2	213032.0	469449.9	42016.8	226973.0	212898.5	468854.8	10049.8	50014.9	62440.0	119214.8
900	53131.2	284697.6	266850.3	575553.5	53409.8	283869.7	265504.7	560081.0	10779.5	53548.6	67351.5	127480.1
1000	65188.6	351954.8	326368.0	724479.0	64594.8	350013.0	325613.8	725492.7	11438.7	56777.7	72225.0	135745.8

TABLE VII
TOTAL MESSAGE COUNT VS. NUMBER OF NODES, WEIGHT DISTRIBUTION: **GAUSS**. DENSITY IS FIXED AT 0.1 FOR ERDÖS-RENYI AND WATTS-STROGATZ NETWORKS, AND RANGES FROM 0.2 TO 0.05 AS THE NODE COUNT INCREASES IN GEOMETRIC NETWORKS

	Erdös-Renyi				**Watts-Strogatz**				**Geometric**			
N	**Hoepman**	**Watten**	**Lot2009**	**Lot2015**	**Hoepman**	**Watten**	**Lot2009**	**Lot2015**	**Hoepman**	**Watten**	**Lot2009**	**Lot2015**
100	761.0	3902.0	8466.3	12111.3	740.1	3946.7	8464.2	11993.3	1719.2	9507.9	16850.0	25489.6
200	2810.2	15323.9	28551.1	42546.2	2817.9	15888.3	28671.3	42309.6	3535.7	19738.3	34350.4	52034.2
300	6238.1	35110.9	58288.9	89745.0	6162.6	34797.1	58915.0	89988.2	4860.0	27289.0	47763.7	71854.0
400	10767.2	61659.0	95485.1	150979.3	10890.1	62392.3	96880.1	153103.7	10142.8	34010.6	60892.6	110004.2
500	16628.8	96654.2	134175.8	276411.0	16723.7	97358.0	139536.4	238227.1	7134.6	40378.6	73306.6	130379.2
600	24207.2	136523.6	193331.4	396684.4	23970.0	137955.2	191170.6	394047.2	8073.4	45357.0	83248.0	148635.6
700	32746.6	185536.2	247769.0	526139.2	32860.2	188475.0	248040.4	526800.0	9226.0	50270.4	93056.0	161381.2
800	42965.4	251776.6	305900.0	671498.6	42705.0	244472.8	311612.0	674437.8	10005.0	55037.4	103961.8	178673.8
900	53775.6	309016.2	376844.8	839197.8	54166.6	312700.6	376003.2	839720.8	10858.0	60849.8	115391.2	198741.6
1000	67127.8	383844.8	445429.6	1018570.8	67126.4	386209.6	454900.0	1024084.4	11491.0	64283.8	121260.0	213855.0

TABLE VIII
The effect of density on approximation performance in small-world networks. Node size is fixed at 300. D is the density.

	Weight Distribution: Uniform								**Weight Distribution: Gauss**							
	Erdös-Renyi				**Watts-Strogatz**				**Erdös-Renyi**				**Watts-Strogatz**			
D	**Hoep**	**Watt**	**Lot09**	**Lot15**	**Hoep**	**Watt**	**Lot09**	**Lot15**	**Hoep**	**Watt**	**Lot09**	**Lot15**	**Hoep**	**Watt**	**Lot09**	**Lot15**
0.10	0.93	0.76	0.59	0.75	0.94	0.77	0.61	0.76	0.96	0.74	0.76	0.84	0.97	0.73	0.73	0.83
0.20	0.96	0.76	0.49	0.80	0.95	0.75	0.51	0.80	0.97	0.72	0.66	0.85	0.98	0.71	0.64	0.85
0.30	0.97	0.75	0.45	0.83	0.97	0.76	0.46	0.82	0.98	0.70	0.57	0.85	0.98	0.70	0.57	0.84
0.40	0.98	0.75	0.43	0.85	0.98	0.75	0.45	0.84	0.98	0.70	0.55	0.85	0.98	0.70	0.53	0.84
0.50	0.98	0.76	0.42	0.86	0.98	0.76	0.42	0.86	0.98	0.70	0.51	0.84	0.98	0.70	0.49	0.85
0.60	0.98	0.75	0.41	0.87	0.98	0.76	0.41	0.87	0.98	0.69	0.47	0.84	0.99	0.69	0.47	0.84
0.70	0.98	0.75	0.41	0.86	0.99	0.76	0.42	0.87	0.99	0.67	0.45	0.84	0.98	0.68	0.46	0.85
0.80	0.99	0.75	0.42	0.88	0.98	0.74	0.42	0.87	0.99	0.68	0.45	0.84	0.99	0.68	0.44	0.85
0.90	0.99	0.76	0.41	0.86	0.99	0.75	0.41	0.88	0.99	0.67	0.42	0.85	0.99	0.68	0.43	0.85
1.00	0.99	0.76	0.43	0.87	0.99	0.76	0.42	0.88	0.99	0.67	0.42	0.84	0.99	0.68	0.41	0.86

B. Time

Tables IV and V provide computational results for the total time required for each algorithm in different types of networks. Density is fixed at 0.1 as well. The results show that Hoepman's and Wattenhofer's algorithms converge quicker than other algorithms. Total time step required in a network with n nodes is $0.2n$ for Hoepman, $0.15n$ for Wattenhofer and around $0.8n$ for Lotker2009. Asynchronous version of Lotker2009 performs slightly worse than Hoepman and Wattenhofer, while, Lotker2015 performs dramatically worse than other algorithms as the required number of time step is 5 times of the total number of nodes in small-world networks.

Recall that Lotker2015 is originally a synchronous algorithm, consisting of phases and solves a black-box MWM algorithm (which is Lotker2009 in our simulations) at each phase. At the end of each phase, nodes update their status and have to inform their neighbors through messages. Besides, the calculation of augmenting gain of an edge requires the actual weight function value of its incident matched edges, which amounts to further messages. As we see in the next subsection, the algorithm uses higher number of messages than other algorithms. Since we assume that transmission delay of message takes a unit of time, this explains the high amount of time requirement of the algorithm.

Unlike in small-world networks, the density is not fixed in our geometric network instances and the graphs become sparser when the number of nodes increases. In the *geometric* column of Tables IV and V, the total time required for the algorithms have a non-increasing trend, which means the required time depends on the density of the network, rather than the total number of nodes.

When we compare the results in Table IV with Table V, we see that Lotker2009 and Lotker2015 require more computation time when the weights have a Gaussian distribution. In these algorithms, classes and subclasses are formed with respect to the weights of edges. Thus, edges having similar weights, which is more-likely in case of normally distributed weights, will be in the same subclass while some classes may be empty. Consequently, the algorithm will require more steps to solve the *crowded* subclasses.

C. Total message count

In Table I, we showed that the message complexities of the algorithms are highly dependent upon the number of edges. Therefore we can compare the message sizes of the algorithms with respect to the number of edges in the network instances. Note that $m = \frac{Dn(n-1)}{2}$ where D is the density, and n and m are the number of nodes and edges, respectively.

In Hoepman's algorithm, at most two messages are expected to be transmitted through each edge. Our results in Tables VI and VII verify this expectation as the total number of messages in a network is 1.5 times of the total number of edges, in average.

In Wattenhofer's and Lotker2009's algorithms, the number of messages are expected to be similar since both algorithms depend on the same approach: Israeli&Itai's MCM algorithm. Our results certify this statement, too, as they have the same trend in Tables VI and VII. However, since Lotker2009 solves randomized MCM algorithm for a certain number of subclasses in parallel, the size of messages in Lotker2009 is greater than that of Wattenhofer. In other words, even though the number of messages are close to each other, the number of bits transmitted until termination is relatively higher in Lotker2009.

The difference between the number of messages required for Wattenhofer and Lotker2009 in geometric network topologies is due to the change in density. In geometric instances, while the size of the network becomes greater, the graph becomes sparser. Thus the difference between these algorithms in the geometric network instances means that Wattenhofer's message complexity is dependent on the number of nodes, while that of Lotker2009 is dependent on the number of edges.

As we expected and explained in the previous subsection, Lotker2015 requires the highest number of messages, which is roughly 14 messages per edge in small-world networks and 25 messages per edge in geometric networks.

D. The effect of density on approximation ratio

In the tables discussed so far, we listed results of small-world instances with a fixed density of 0.1 for the purpose of clarity. The table in Table VIII provides the results when the graph size is fixed at 300 and density (D) ranges from 0.1 to ~ 1 (complete graph).

Results show that the success of Hoepman gets even better when the graph becomes denser and it reaches an approximation ratio of 0.99 in average in complete graphs.

Wattenhofer's approximation ratio is not affected by the density under uniform weight distribution since the algorithm omits the locally-lighter edges at the beginning of the algorithm, regardless of the density. When the weights are distributed normally, its approximation ratio drops slightly.

Lotker2009's performance drops from 0.6 to 0.42 in uniform weight distribution and from around 0.73 to 0.41 in Gaussian weight distribution as the graphs become denser. The reason might be that the size of each subclass gets large in case of denser graphs and since the edges are chosen randomly in each subclass, randomization causes the elimination of *good* edges at early phases, as in Wattenhofer's algorithm.

V. Conclusion

We presented the extensive performance evaluations of four asynchronous distributed weighted maximum matching algorithms running in $\mathcal{CONGEST}$ model. We implemented algorithms in a discrete event simulator and compared their performances with respect to approximation ratio, convergence time and message count in different types of random networks.

Results show that Hoepman's greedy algorithm outperforms all other algorithms in all performance criteria and in all network topologies. Despite its theoretical approximation guarantee of $\frac{1}{2}$ of the optimum, it had an outstanding approximation ratio of 0.97 in sparse small-world network instances and even reaches 0.99 in denser instances. It realizes this using very low number of messages compared to the other algorithms.

Wattenhofer's algorithm and Lotker's improvement algorithm (Lotker2015) had similar approximation ratios (around 0.78) in networks having uniform weight distribution, while Lotker2015 had 15% better approximation ratios than Wattenhofer in case of normally distributed weights. However Lotker2015 required a great amount of time and messages to achieve these results, while Wattenhofer's time requirement is the lowest in all of the algorithms. Lotker2015's high message and time requirement is mainly due to the facts that it was designed as a synchronous algorithm and it requires relatively high amount of information sharing between neighbors.

If one wants better practical approximation ratios for the distributed maximum weighted matching problem, an effective approach could be to develop an improvement algorithm on top of the greedy algorithm. Note that although Lotker2015 is an important improvement algorithm, it cannot improve the weight of a solution of Hoepman's greedy algorithm since the augmenting gain of all of the unmatched edges would be non-positive. An improvement algorithm running on top of the greedy algorithm and detecting longer augmenting paths with positive augmenting gain may give even better approximations for the optimum solution.

Acknowledgment

Authors would like to thank TUBITAK for the project grant (215E115) and the PhD research grant (BIDEB-1001).

References

[1] C. Laitang, K. Pinel-Sauvagnat, and M. Boughanem, *DTD Based Costs for Tree-Edit Distance in Structured Information Retrieval*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 158–170. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-36973-5_14

[2] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," *International journal of pattern recognition and artificial intelligence*, vol. 18, no. 03, pp. 265–298, 2004.

[3] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

[4] S. Jin, J. Zhang, P. S. Yu, S. Yang, and A. Li, "Synergistic partitioning in multiple large scale social networks," in *Big Data (Big Data), 2014 IEEE International Conference on*. IEEE, 2014, pp. 281–290.

[5] B. Monien, R. Preis, and R. Diekmann, "Quality matching and local improvement for multilevel graph-partitioning," *Parallel Computing*, vol. 26, no. 12, pp. 1609–1634, 2000.

[6] H. N. Gabow, *Data structures for weighted matching and nearest common ancestors with linking*. University of Colorado, Boulder, Department of Computer Science, 1990.

[7] Y. Wang, F. Makedon, J. Ford, and H. Huang, "A bipartite graph matching framework for finding correspondences between structural elements in two proteins," in *Engineering in Medicine and Biology Society, 2004. IEMBS'04. 26th Annual International Conference of the IEEE*, vol. 2. IEEE, 2004, pp. 2972–2975.

[8] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, no. suppl 1, pp. i47–i56, 2005.

[9] M. M. Halldórsson, S. Köhler, B. Patt-Shamir, and D. Rawitz, "Distributed backup placement in networks," in *Proceedings of the 27th ACM Symposium on Parallelism in Algorithms and Architectures*, ser. SPAA '15. New York, NY, USA: ACM, 2015, pp. 274–283. [Online]. Available: http://doi.acm.org/10.1145/2755573.2755583

[10] B. Patt-Shamir, D. Rawitz, and G. Scalosub, "Distributed approximation of cellular coverage," *Journal of Parallel and Distributed Computing*, vol. 72, no. 3, pp. 402–408, 2012.

[11] J. Edmonds, "Paths, trees and flowers," *Canadian Journal of Mathematics*, pp. 449–467, 1965.

[12] Z. Lotker, B. Patt-Shamir, and S. Pettie, "Improved distributed approximate matching," *J. ACM*, vol. 62, no. 5, pp. 38:1–38:17, Nov. 2015. [Online]. Available: http://doi.acm.org/10.1145/2786753

[13] A. Israeli and A. Itai, "A fast and simple randomized parallel algorithm for maximal matching," *Information Processing Letters*, vol. 22, no. 2, pp. 77 – 80, 1986.

[14] M. Wattenhofer and R. Wattenhofer, *Distributed Weighted Matching*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 335–348.

[15] J. Hoepman, "Simple distributed weighted matchings," *CoRR*, vol. cs.DC/0410047, 2004. [Online]. Available: http://arxiv.org/abs/cs.DC/0410047

[16] Z. Lotker, B. Patt-Shamir, and A. Rosén, "Distributed approximate matching," *SIAM Journal on Computing*, vol. 39, no. 2, pp. 445–460, 2009.

[17] D. Peleg, "Distributed computing," *SIAM Monographs on discrete mathematics and applications*, vol. 5, 2000.

[18] N. Matloff, "Introduction to discrete-event simulation and the simpy language," *Davis, CA. Dept of Computer Science. University of California at Davis. Retrieved on August*, vol. 2, p. 2009, 2008.

[19] T. Nieberg, "Local, distributed weighted matching on general and wireless topologies," in *Proceedings of the fifth international workshop on Foundations of mobile computing*. ACM, 2008, pp. 87–92.

[20] G. v. Rossum *et al.*, "Python programming language," 1989. [Online]. Available: http://python.org

[21] P. Erdös and A. Rényi, "On random graphs, i," *Publicationes Mathematicae (Debrecen)*, vol. 6, pp. 290–297, 1959.

[22] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-worldnetworks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[23] D. A. Schult and P. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conferences (SciPy 2008)*, vol. 2008, 2008, pp. 11–16.

[24] Z. Galil, "Efficient algorithms for finding maximum matching in graphs," *ACM Computing Surveys (CSUR)*, vol. 18, no. 1, pp. 23–38, 1986.