Improving Distributed Maximum Weighted Matchings for Communication Networks

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*—Matching problem has been attracting a growing attention in network design, especially in wireless communication networks. In this paper we present a distributed heuristic which improves the weight of a given matching. Our algorithm assumes an asynchronous communication model in which the message size is limited to $O(\log n)$ bits. To the best of our knowledge, it is the first distributed weighted matching algorithm which benefits from augmentation of alternating paths having size larger than 3 and uses small messages. We also provide results of our simulations on random geometric graphs. Computational results show that our algorithm improves the approximated solutions of other weighted matching algorithms roughly by 1-3% and closes the gap between the approximation ratio and the optimum solution by 9-26%.**

***Keywords*— Weighted matching, communication networks, distributed algorithms, approximation algorithms.**

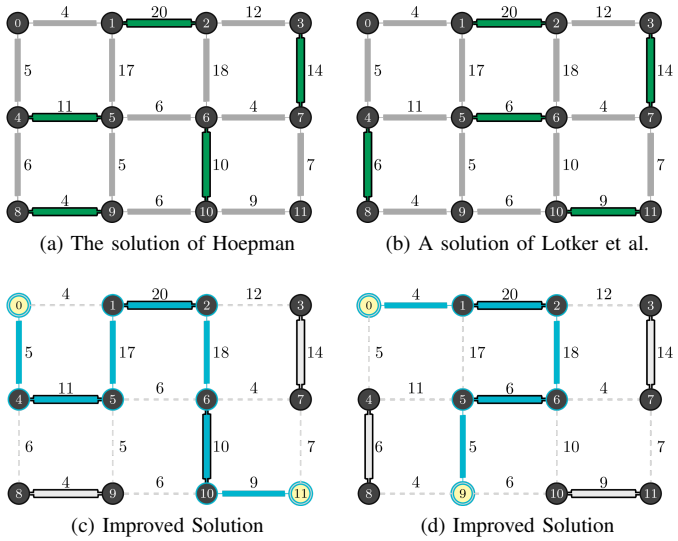
I. Introduction

Matching problem is among the most fundamental graph theoretical problems and it has been attracting a growing attention in network design, especially in wireless communication networks [1], [2]. Some of these applications are radio resource allocation in cognitive radio (CR) networks [3], [4], physical layer security [5], energy-efficient partner selection in wireless networks [6], [7], partner selection for cooperative relaying [8], and optimizing storage capacity and power consumption [9]. General aim of these applications is to pair the nodes (or *users* or *agents*) according to their preference lists obtained by a specific criteria. If nodes to be paired have distinct roles such as server-client, user-resource, etc., the problem is knows as bipartite matching in graph theory and stable marriage problem in game-theoretical approach. When each node can be paired with any node, the problem is general matching problem, whose corresponding game-theoretical approach is the roommate problem.

In this paper, we address the maximum weighted matching problem (MWM) where, given an undirected graph $G(V, E)$ and a weight function $w : E \rightarrow \mathbf{R}^+$, the aim is to find a subset of edges $M \subseteq E$ such that no two edges in M are incident to the same node and the sum of weights of edges in M is maximized.

Related Work. Although MWM can be computed in polynomial time in centralized settings by the famous Blossom Algorithm of Edmonds [10], it lacks of an exact solution in distributed settings. There have been many distributed approximation algorithms for MWM running in logarithmic time. Watthenhofer and Wattenhofer [11] refined Israeli&Itai's randomized algorithm for the maximum cardinality matching (MCM) problem [12] to weighted graphs. Each node simply neglects relatively lighter edges, randomly chooses one of the remaining neighbors. Two nodes are considered matched when they mutually select each other. Their algorithm is proved to approximate the MWM by a factor of $\frac{1}{5}$. Hoepman [13] formalized the greedy approach where each node deterministically favors its heaviest available edge and an edge is considered in the matching if it is favored by both of its incident nodes. This approach guarantees a $\frac{1}{2}$-approximation at the cost of a time complexity of $O(|E|)$. Lotker et al. [14] proposed a randomized $(\frac{1}{4} - \epsilon)$-approximation algorithm whose running time is $O(logn)$. The algorithm divides the edges into classes and subclasses according to their weights and apply an unweighted MCM algorithm (e.g. Israeli-Itai's algorithm) in each subclass in parallel. Resulting matchings of each class are then combined into a single maximal matching, giving priority to the heavier classes. Lotker et al. [15] gave a randomized synchronous $(\frac{1}{2} - \epsilon)$-approximation algorithm running in $O(\log \epsilon^{-1} \log n)$ time. The algorithm, starting from an empty matching, repeatedly applies the algorithm in [14] using a different weight function than the given weights. The values of the weight function are calculated by using the gains of alternating paths (see Section II) of length at most 3. For a performance evaluation of above MWM algorithms on asynchronous message-passing environment, we refer readers to [16].

In this paper, we propose a distributed MWM algorithm that improves the total weight of a given matching. Note that as well as any maximal matching solution resulted from any MCM algorithm, the solutions of all MWM algorithms mentioned above can be further improved by our algorithm in a time proportional to the number of nodes. This algorithm is the first distributed weighted matching algorithm which uses small messages to detect augmenting paths of size greater than 3. We show that the augmenting gain of an augmenting path can be calculated using local information and small-sized messages. Figure 1 illustrates two examples of weight improvement. Note that even when Lotker et al.'s algorithm [15] (which is an improvement algorithm too) is used, positive weighted augmenting paths (see Sec. II) may exist. Our algorithm



(a) The solution of Hoepman (b) A solution of Lotker et al.

(c) Improved Solution (d) Improved Solution

Fig. 1. *Improvability of Distributed MWM algorithms.* In (a) and (b), possible approximate solutions of Hoepman's [13] and Lotker et al.'s [15] algorithms are illustrated. Double layered and colored edges are matched edges. In (c) and (d), the colored edges are on the positive-weighted augmenting paths. Augmenting those paths improves solutions. Note that the solution in (b) cannot be further improved by Lotker et al.'s algorithm, even though it is an improvement algorithm.

tries to find those augmenting paths. It realizes this with $(O(\log n))$-sized messages in asynchronous manner, which enables it to run right along with any other algorithm without any substantial cost of message size. It is well suited for systems where improvements on any matching value could be significant. We also give some computational results for showing off the impact of our algorithm when used right along with three of the existing MWM algorithms.

II. Communication Model and Basic Notations

We assume a strongly decentralized network environment where each node runs asynchronously with respect to its neighbors. We further assume the size of any message is limited to $O(\log n)$ bits.

Given a matching $M \subseteq E$, an edge is called *matched* (resp. *unmatched*) if it is in (resp. out of) M. A node is called *free* if none of its edges is in M.

A path whose edges alternatively traverses *matched* and *unmatched* edges is called *alternating path*. If the alternating path starts and ends in *free* nodes, it is called an *augmenting path*. The matched edges of an augmenting path can be changed to unmatched and vice versa, and the resulting set of matched edges is still a valid matching in E. This operation is called *augmenting*.

The weight of an alternating or augmenting path is the total weight of the *matched* edges on the path. *Augmenting gain* (resp. *alternating gain*) of an augmenting (resp. *alternating*) path p is denoted by $\mathcal{A}(p)$ and signifies the increase in the weight of the matching after an *augmenting* operation. Given an augmenting path $p = \{n_0, n_1, \ldots, n_k\}$ where $n_i \in V$, its augmenting gain can be formulated as follows:

$$\mathcal{A}(p) = \sum_{i=0}^{k-1} (-1)^i w(n_i, n_{i+1}) \tag{1}$$

Throughout the paper, we call an augmenting path p as *positive augmenting path* (resp. *negative augmenting path*) if it has a positive (resp. negative) augmenting gain.

III. Improving Weighted Matchings

Our algorithm assumes that a maximal matching solution is already known. Note that this can be calculated using a generic unweighted or weighted maximal matching approximation algorithm.

A. Path finding procedure

Once a maximal matching is known, each node on the graph is in one of the following three states:

- MATCHED: The node is matched with a node and its degree is greater than one.
- LEAF: The node has a single neighbor with which it is matched.
- FREE: The node is not matched with any node.

Given a graph $G(V, E)$ and a maximal matching M, a FREE node cannot have a FREE neighbor; otherwise M would not be maximal. Suppose each LEAF node on the graph is connected to an imaginary node with an imaginary edge having zero weight. Introduced imaginary nodes are FREE nodes by definition and we call them *imaginary free nodes* (IMAGFREE). After introducing imaginary edges and nodes, M is still maximal on the new graph, say $\hat{G}$. Thus, all the augmenting paths in $\hat{G}$ with respect to a maximal matching M start and end in FREE or IMAGFREE nodes. Throughout the paper, we assume that LEAF nodes are the representatives (in G) of IMAGFREE nodes and perform their calculations.

Initiators. A node which initiates the path finding procedure is called *initiator*. FREE and LEAF nodes are initiators.

The basic idea of our distributed algorithm is to detect augmenting paths between two initiator nodes and augment the path if the total weight gained by doing so is increased.

Proposals. Let $pr(s, r)$ be a message from s to r and carry information about the current status of a path rooted from an initiator and passing through the edge (s, r). Its data field is a triplet of integers, namely $\{$*source, distance, value*$\}$. *Source* is id the initiator node of the alternating path. *Distance* is the length of the alternating path between the initiator and r, while *value* indicates the actual alternating gain of the path.

Path finding procedure begins when each initiator node chooses one of its neighbors and sends a *proposal* to it. (Different approaches can be used for candidate selection, one of which could be selecting the heaviest edge.) Throughout the algorithm, the initiated proposals are traversed through matched nodes.

A *matched* node has two basic roles: (1) it forwards the proposal coming from its matching partner to one or more unmatched edges; (2) it chooses the best proposal coming from the unmatched edges and forwards it to its matching partner.

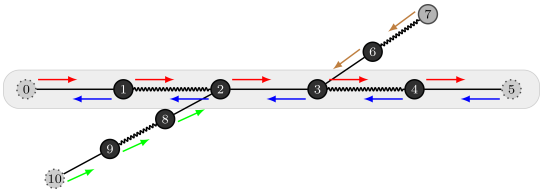


Fig. 2. *Forming of an augmenting path.* Arrows indicate proposals sent between nodes. Each color represents a different initiator. Grey-highlighted area indicates the selected augmenting path. The nodes 2 and 3 are branch nodes which decide on the direction of path.

Evaluation of Proposals. Each node on an augmenting path has to be aware of the augmenting gain of the path in order to decide whether it should update its matching partner. Furthermore, it has only local information, namely two proposals received from its neighbors on the path.

Suppose an initiator node n_0 starts a path finding procedure by sending a proposal to n_1, the proposal was forwarded to a node n_i through the path p = $\{n_0, n_1, \ldots, n_i\}$ and is to be forwarded to n_{i+1} with a proposal $pr(i, i+1)$. The actual value $\mathcal{V}$ of the proposal $pr(i, i+1)$ is the alternating gain of the path p, and is calculated using the following equation

$$\mathcal{V}(pr(i,i+1)) = \sum_{j=0}^{i} (-1)^j w(n_j, n_{j+1}) \tag{2}$$

Since n_0 is a FREE node, (n_i, n_{i+1}) is always an unmatched edge if i is even and it is a matched edge if i is odd. The factor $(-1)^j$ on Eqn. 2 guarantees that the weights of unmatched edges are added to the summation, while the weights of matched edges are subtracted from it.

Lemma 1: Actual value of a proposal originated from n_0 and sent from n_i to n_{i+1} can be calculated using only the weight of the edge (n_i, n_{i+1}) and the augmenting value of the best proposal received by n_i.

Proof: It follows from Eqn. 2 that the following equation holds for all nodes:

$$\mathcal{V}(pr(i,i+1)) = \begin{cases} \mathcal{V}(pr(i-1,i)) + w(n_i, n_{i+1}), & \text{if } (n_i, n_{i+1}) \in E \setminus M \\ \mathcal{V}(pr(i-1,i)) - w(n_i, n_{i+1}), & \text{if } (n_i, n_{i+1}) \in M \end{cases} \tag{3}$$

■

Path evaluation and augmentation. In this subsection we describe how a node locally computes the augmenting gain of an augmenting path.

Lemma 2: Suppose an augmenting path $p = \{n_0, n_1, \ldots, n_k\}$ between two initiator nodes n_0 and n_k. Let n_i be the $(i+1)^{th}$ node on p from n_0 to n_k. Any node n_i on p can calculate the augmenting gain of p using only the proposals it receives from its neighbors on the path, i.e. using the following formula:

$$\mathcal{A}(\{n_0, n_1, \ldots, n_k\}) = \mathcal{V}(pr(n_{i-1}, n_i)) + \mathcal{V}(pr(n_{i+1}, n_i)) \tag{4}$$

Proof: Let n'_i be another denotation for the nodes on the augmenting path p such that $n_i = n'_{k-i}$. In other words, $\{n_0, n_1, \ldots, n_{k-1}, n_k\} = \{n'_k, n'_{k-1}, \ldots, n'_1, n'_0\}$. Eqn. 1 can be rewritten as

$$\mathcal{A}(p) = \sum_{i=0}^{j-1} (-1)^i w(n_i, n_{i+1}) + \sum_{i=j}^{k} (-1)^i w(n_i, n_{i+1}) \tag{5}$$

which is equal to

$$\mathcal{A}(p) = \sum_{i=0}^{j-1} (-1)^i w(n_i, n_{i+1}) + \sum_{i=0}^{k-j} (-1)^i w(n'_i, n'_{i+1}) \tag{6}$$

From Eqn. 2, it follows that

$$\begin{aligned} \mathcal{A}(p) &= \mathcal{V}(pr(n_{i-1}, n_i)) + \mathcal{V}(pr(n'_{k-j}, n'_{k-j+1})) \\ &= \mathcal{V}(pr(n_{i-1}, n_i)) + \mathcal{V}(pr(n_{i+1}, n_i)) \end{aligned}$$

■

Thus, any node on an augmenting path may calculate the gain of the path simply by summing up the actual values of two proposals incoming from its neighbors on the same path.

B. The algorithm

In this section we provide the pseudo-code of our algorithm. We divided the algorithm into sub-procedures for the sake of clarity. Each node has two pointers: m_{init} and m_{impr}. m_{init} points to the initial matching partner determined by an arbitrary maximal matching algorithm. m_{impr} is set if the node is on a positive augmenting path.

Initialization. Each node on the graph runs the procedure *ImproveMatching* distributedly. Once a maximal matching is obtained and m_{init} pointers are set accordingly, initiators (FREE and LEAF nodes) run the procedure *InitiateProposal*, while MATCHED nodes run the procedure *ForwardProposal*.

Procudure *InitiateProposal*. Initiators select one of their neighbors (e.g. the one connected via heaviest edge) as candidate and send a proposal to it. Starting value of a proposal is the weight of the edge between the initiator and its candidate (If initiator is a LEAF node, starting value of the proposal is multiplied by -1, considering IMAGFREE nodes. See Sec. III-A). If, at any time, an initiator node receives a proposal from its candidate, it means an augmenting path is constructed and the augmenting weight of the path is the value of the proposal. If this augmenting value is positive, m_{impr} pointer is set accordingly. Otherwise, m_{impr} is set to NULL.

Procedure *ForwardProposal*. A MATCHED node waits for proposals. Once it receives a proposal through the matched edge, it forwards to other neighbors. If it has already received at least one proposal from its unmatched neighbors, it selects the proposal with the greatest value, forwards the proposal only to this node and informs other neighbors. If it has not received any proposal from unmatched nodes, the proposal of matched node is forwarded to all neighbors.

Once a proposal is received from one of the unmatched neighbors, it is checked for validity. Validity check, which is explained in the next paragraph, is required for the prevention and avoidance of the proposal cycles. If a proposal is considered valid, it is included in the set of proposals, say P. Among all proposals in P, each node selects the one with the greatest

Algorithm 1 Main algorithm

```
1: procedure ImproveMatching(G(V,E), M)
2:     solve InitialMaximal Matching
3:     if status ∈ {Free, Leaf} then InitiateProposal()
4:     elseif status ∈ {Matched} then ForwardProposal()
```

Algorithm 2 Initiation (by Free and Leaf nodes)

```
1: procedure InitiateProposal
2:     k ← select best candidate
3:     if M1_STATUS = Free then w_prop ← w(me, k)
4:     else w_prop ← −w(me, k)
5:     send prop(w_prop, myid, 1) to k
6:     repeat
7:         receive msg from k
8:         if msg.type = PROP and msg.value > 0 then
9:             m_impr ← k
10:        else if (msg.type = PROP and msg.value <= 0) or
       msg.type = CANCEL then
11:            m_impr ← NULL
12:    until termination
```

Algorithm 3 Forwarding (by Matched nodes)

```
1: procedure ForwardProposal
2:     M ← the matching partner
3:     U ← { set of neighbors } \ M
4:     while TRUE do
5:         prop ←receive PROP message from a neighbor
6:         if prop received from M then
7:             if there are props from any node in U then
8:                 c ← select best prop incoming from U
9:                 forward prop to c
10:                forward c's proposal to M
11:            else
12:                forward prop to all u ∈ U
13:                bcast ← TRUE, c ← NULL
14:        else
15:            if proposal is valid then
16:                accept prop and add it to the proposal list
17:            c_new ← select the best prop from U
18:            if c_new ≠ c_old or prop value is changed then
19:                forward c_new's proposal to M
20:                if received a proposal from M then
21:                    forward M's proposal to c_new
22:                    cancel proposal to c_old
23:            if bcast = TRUE and c_new ≠ NULL then
24:                cancel proposals to all u ∈ U \ c_new
25:                bcast ← FALSE
```

value and forwards it to its matched node. If, at any time, the sender or value of the best proposal in P changes, the new best proposal is forwarded to the matching partner, and sender of the old best proposal is informed with a cancel message.

Proposal validation and cycle prevention. Our algorithm can be regarded as a search algorithm where an initiator node searches for a path to another initiator. During the path finding procedure, forwarding of proposals may create cycles. We omit the detailed analysis of cycle prevention due to space constraints. Note that by checking the distance and source of the proposal, as well as limiting the number of forwarded proposals of a specific initiator by a constant, say τ, we prevent all probable cycles of proposals.

C. Analysis

Time complexity. Suppose a graph $G(V, E)$ and a maximal matching $M \subseteq E$ such that all the augmenting paths with respect to M are cycle-free. If there is an alternating path p having distance d from a free node n_0 to a matched node n_{d-1}, then, after at most d steps, either n_{d-1} receives the proposal related to p or the proposal would have been declined by a node on the path from n_0 to n_{d-1}.

Considering the case where cycles of proposals may exist, if we use the cycle prevention method discussed above, we guarantee that the number of forwarding a proposal originated from a specific initiator is limited by a constant τ. Thus, an initiator receiver the proposal of another initiator in at most $\tau \times |V|$ time. This concludes that the time complexity of our algorithm is $O(\tau n)$ where n is the number of nodes.

Message complexity. The number of messages required for the algorithm is highly dependent on the initial maximal matching solution. The number of simultaneous path finding procedures is equal to the number of initiators. In case the given initial solution is a perfect matching on a graph where the minimum degree of nodes is at least 2, no message is sent. When initiator nodes exist, each of them may send exactly one PROP message in our implementation of the algorithm. Other nodes may send more than one messages. Each PROP message initiated by an initiator node may be forwarded by a Matched node at most τ times. Thus, at the worst case, the message complexity of the algorithm is $O(n_{init}(n - n_{init})\tau)$, where n_{init} is the number of initiator nodes. The best case scenario for a graph having Free or Leaf nodes is where all Matched nodes receive their best proposal before all the other proposals it may receive, and hence forwards only one message. In this case, the message complexity is $O(n)$.

IV. Computational Results

Using SimPy [17], a discrete event simulator, we built an asynchronous networking environment for simulating distributed algorithms. Due to space restrictions, we report in Table I only the random geometric network instances having size between 100 and 1000, and an average degree of 10. For edge weights, we randomly assigned a number from 20 to 100 to each edge. For obtaining initial maximal matchings, we solved instances by algorithms of [13], [11] and [15]. r_{init} indicates the approximation ratio of these algorithms. Once a maximal matching is found, we ran our algorithm to improve the total weight of the matching. r_{impr} signifies the improved approximation ratio. Values in ΔGap columns are relative improvements with respect to initial maximal matching solutions. t_{impr} columns indicate the total running time of the improvement algorithm assuming that the transmission of a message takes one unit of time. m_{init} and m_{impr} are the number of messages transmitted during initial and improvement matching algorithms, respectively. Values in each cell are averages of 10 different instances with the same size.

TABLE I
COMPUTATIONAL RESULTS ON SYNTHETIC GEOMETRIC NETWORKS. AVERAGE DEGREE: 10

	Init. Matching: Hoepman						Init. Matching: Wattenhofer						Init. Matching: Lotker et al.					
N	r_{init}	w_{impr}	ΔGap	t_{impr}	m_{init}	m_{impr}	r_{init}	w_{impr}	ΔGap	t_{impr}	m_{init}	m_{impr}	r_{init}	w_{impr}	ΔGap	t_{impr}	m_{init}	m_{impr}
100	0.93	0.95	0.28	64.7	729	2893	0.79	0.82	0.12	84.9	3564	3023	0.91	0.92	0.14	205.4	11381	3068
200	0.93	0.95	0.23	93.5	1491	6592	0.79	0.80	0.06	120.3	7276	6886	0.92	0.93	0.17	240.6	23114	6521
300	0.92	0.94	0.25	112.4	2259	8933	0.78	0.80	0.09	108.4	10929	10186	0.92	0.93	0.08	255.2	34781	9882
400	0.93	0.95	0.25	99.2	3025	12234	0.78	0.80	0.06	129.6	14581	14929	0.92	0.93	0.06	281.3	46415	13527
500	0.93	0.94	0.24	108.9	3780	15487	0.79	0.81	0.09	135.1	18356	17987	0.91	0.92	0.09	284.9	58091	16842
600	0.92	0.94	0.27	115.5	4544	19590	0.79	0.80	0.07	127.4	21675	22219	0.92	0.93	0.09	294.1	69442	19710
700	0.93	0.94	0.26	132.9	5318	22725	0.78	0.81	0.13	119.5	25611	24609	0.92	0.92	0.07	303.9	81440	23232
800	0.93	0.95	0.26	123.4	6052	25587	0.78	0.80	0.08	145.7	29494	29493	0.92	0.93	0.12	312.1	93125	26547
900	0.93	0.95	0.28	130.7	6830	29508	0.79	0.80	0.09	156.0	32854	34341	0.92	0.93	0.11	316.5	104937	31348
1000	0.93	0.95	0.26	147.1	7592	32330	0.79	0.80	0.08	135.7	36339	37620	0.92	0.93	0.09	327.4	116809	33944
Avrg	0.93	0.95	**0.26**				0.79	0.80	**0.09**				0.92	0.93	**0.10**			

Our heuristic improved the approximation ratios by 1-3% in average. When Hoepman's algorithm is used to obtain the initial matching, the gap between the approximation and the optimum is closed by 26% in average. In case of initializing with Wattenhofer's algorithm and Lotker et al.'s improvement algorithm, this metric amounts to 9% and 10%, respectively.

The improvement heuristic requires 30 messages per node in average and increases with the node count in a linear way. The number of transferred messages for the proposed heuristic is lower than the messages required in Lotker et al.'s algorithm for all cases and Wattenhofer's algorithm for most of the cases. Note that the running time of our heuristic increases in a sublinear manner with the increasing node count. The running time is highly dependent on the number of initiators (FREE and LEAF nodes) at the end of the initial matching. So to say, when the number of initiators increases, the number of simultaneous path finding procedures steps up and results in longer running times. Since smaller graphs may possibly have higher number of initiators than larger graphs, the fluctuations in the running time columns of the tables are not considered as anomalies.

V. CONCLUSION

In this paper, we presented a distributed heuristic which improves the approximation performance of any maximum weighted matching algorithm. To the best of our knowledge, our algorithm is the first distributed weighted matching algorithm which uses small messages ($O(\log n)$ bits) to detect augmenting paths of size greater than 3. We analyze the time and message complexities of the proposed algorithm. In our simulations, we tested the practical performance of our algorithm on top of three well-known weighted matching algorithms and on different types of networks. Improvement algorithm achieves best approximation ratios when it is applied on top of Hoepman's algorithm. It closed the gap between approximations of these algorithms and the optimum solution by a percentage varying between 9 and 26 while consumed reasonable amout of resources.

ACKNOWLEDGMENT

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

REFERENCES

[1] Y. Gu, W. Saad, M. Bennis, M. Debbah, and Z. Han, "Matching theory for future wireless networks: fundamentals and applications," *IEEE Communications Magazine*, vol. 53, no. 5, pp. 52–59, May 2015.

[2] S. Bayat, Y. Li, L. Song, and Z. Han, "Matching theory: Applications in wireless communications," *IEEE Signal Processing Magazine*, vol. 33, no. 6, pp. 103–122, Nov 2016.

[3] S. Bayat, R. H. Louie, B. Vucetic, and Y. Li, "Dynamic decentralised algorithms for cognitive radio relay networks with multiple primary and secondary users utilising matching theory," *Transactions on Emerging Telecommunications Technologies*, vol. 24, no. 5, pp. 486–502, 2013.

[4] A. Leshem, E. Zehavi, and Y. Yaffe, "Multichannel opportunistic carrier sensing for stable channel access control in cognitive radio systems," *IEEE Jour. on Selec. Are. in Com.*, vol. 30, no. 1, pp. 82–95, Jan. 2012.

[5] S. Bayat, R. H. Y. Louie, Z. Han, B. Vucetic, and Y. Li, "Physical-layer security in distributed wireless networks using matching theory," *IEEE Trans. on Info. For. and Sec.*, vol. 8, no. 5, pp. 717–732, May 2013.

[6] M. W. Baidas and M. M. Afghah, "A matching-theoretic approach to energy-efficient partner selection in wireless networks," in *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)*, Aug 2015, pp. 1260–1265.

[7] S. M. H. Aejaz and A. Springer, "Partner selection algorithms for improving the network lifetime and decreasing the overall transmit energy expenditure in a wireless sensor network," in *2016 International Conference on Broadband Communications for Next Generation Networks and Multimedia Applications (CoBCom)*, Sept 2016, pp. 1–8.

[8] C. Hasan, E. Altman, and J. M. Gorce, "Partner selection for decode-and-forward cooperative relaying: A matching theoretic approach," in *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Sept 2013, pp. 2275–2280.

[9] A. Roumy and D. Gesbert, "Optimal matching in wireless sensor networks," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 1, no. 4, pp. 725–735, 2007.

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

[11] M. Wattenhofer and R. Wattenhofer, *Distributed Computing: 18th International Conference, DISC 2004, Amsterdam, The Netherlands, October 4-7, 2004. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, ch. Distributed Weighted Matching, pp. 335–348.

[12] A. Israeli and A. Itai, "A fast and simple randomized parallel algorithm for maximal matching," *Inf. Proc. Let.*, vol. 22, no. 2, pp. 77 – 80, 1986.

[13] J. Hoepman, "Simple distributed weighted matchings," *CoRR*, vol. cs.DC/0410047, 2004.

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

[15] 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

[16] C. U. Ileri and O. Dagdeviren, "Performance evaluation of distributed maximum weighted matching algorithms," in *Digital Information and Communication Technology and its Applications (DICTAP), 2016 Sixth International Conference on*. IEEE, 2016, pp. 103–108.

[17] 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.