Perfect Matching with Few Link Activations

Hugo Mirault[1], Peter Robinson[1], Ming Ming Tan[1] ✉, and Xianbin Zhu[2]

[1] School of Computer & Cyber Sciences, Augusta University, Georgia, USA
hmirault@augusta.edu perobinson@augusta.edu mtan@augusta.edu
[2] Department of Computer Science, Aalto University, Finland
xianbin.aaronzhu@gmail.com

Abstract. We consider the problem of computing a perfect matching problem in a synchronous distributed network, where the network topology corresponds to a complete bipartite graph. The communication between nodes is restricted to activating communication links, which means that instead of sending messages containing a number of bits, each node can only send a pulse over some of its incident links in each round. In the port numbering model, where nodes are unaware of their neighbor's IDs, we give a randomized algorithm that terminates in $O(\log n)$ rounds and has a pulse complexity of $O(n \log n)$, which corresponds to the number of pulses sent over all links. We also show that randomness is crucial in the port numbering model, as any deterministic algorithm must send at least $\Omega(n^2)$ messages in the standard LOCAL model, where the messages can be of unbounded size. Then, we turn our attention to the KT_1 assumption, where each node starts out knowing its neighbors' IDs. We show that this additional knowledge enables significantly improved bounds even for deterministic algorithms. First, we give an $O(\log n)$ time deterministic algorithm that sends only $O(n)$ pulses. Finally, we apply this algorithm recursively to obtain an exponential reduction in the time complexity to $O(\log^* n \log \log n)$, while slightly increasing the pulse complexity to $O(n \log^* n)$. All our bounds also hold in the standard CONGEST model with single-bit messages.

Keywords: distributed graph algorithm · perfect matching

1 Introduction

We study the fundamental problem of computing a perfect matching among the nodes of a network, where the communication capabilities are limited to link activations, which is motivated by simple biological cell networks or sensor networks of resource-restricted devices. Our model is inspired by protein-interaction networks and other biochemical signaling networks, where the alphabet of the messages between nodes are effectively restricted to a binary alphabet (i.e., signal or no signal), as demonstrated by [2].

Concretely, we consider a complete bipartite graph consisting of two vertex sets L and R, each consisting of n nodes, and we assume that each node knows whether it is in L or in R. The communication between the nodes follows a *link*

activation model, which can be interpreted as a restricted variant of the standard CONGEST model [6]: In each synchronous round, each node may *activate* any subset of its incident edges by sending a *pulse* over these edges. That is, if a node u sends a pulse over the edge $\{u, v\}$ in round r, then node v observes this activation by u at the start of round $r + 1$. We emphasize that these link activations are the only form of communication available in our model, which, in particular, prevents nodes from directly exchanging bit strings of information, as is commonly assumed in other models of distributed computation. In particular, activating the link $\{u, v\}$ does *not* allow u to transmit a sequence of bits to v in round r, but merely corresponds to a pulse that can be observed by v.

A model with similar features is the *beeping model*, introduced in [3], where, in each synchronous round, a node may choose to "beep" across *all* of its incident links, whereas, on the receiving side, each node can only distinguish between the case where none of its neighbors beeped or at least one of them did. We refer the reader to [5] for a detailed survey describing the common aspects and differences of various biological systems and distributed computing models.

To quantify the performance of algorithms in the link activation model, we analyze the *round complexity* (also known as *time complexity*), which is the worst case number of rounds, as well as the *pulse complexity*, whereby the latter corresponds to the worst-case number of pulses over all edges in any execution, for deterministic algorithms. When considering randomized algorithms, we assume that each node has access to a private source of unbiased random bits that it may query during its local computation at the beginning of each round. In that case, we also consider the *expected pulse complexity*, where the expectation is computed over the random bits of the algorithm.

While the pulse complexity is not a standard metric for distributed algorithms, it is closely related to the *message complexity*. In fact, any upper bound on the pulse complexity also implies the same bound on the message complexity. Hence, designing algorithms with low pulse complexity immediately gives rise to message-efficient algorithms that only need to send a small number of single-bit messages in the CONGEST model. Conversely, given an algorithm in CONGEST model that sends only single-bit messages, one can simulate the algorithm in our link activation model with pulse complexity equal to the message complexity.

Our main goal of this work is to quantify the achievable round and pulse complexity for computing a perfect matching in the link activation model. To this end, we distinguish two standard assumptions pertaining to the initial knowledge of the nodes:

1. In the *port numbering* or KT_0 *model* [6, 7], we assume that nodes are anonymous and do not have unique IDs. Moreover, the incident edges to each node are numbered $1, \ldots, n$ and, at the start of each round, every node v only learns the subset of its incident port numbers that were activated by the respective other endpoint during the previous round. A crucial difficulty in this setting is that the assignment of port numbers to communication links is not known in advance to the algorithm, and may be chosen adversarially.

2. When considering the KT_1 *assumption* [1], on the other hand, we assume that every node is equipped with a unique integer ID, chosen from some range of polynomial size, and an algorithm needs to satisfy the stated properties for all possible ID assignments to the nodes. Each node knows the IDs of all its neighbors, and, since we consider a complete bipartite graph, this means that every node in L knows the IDs of all nodes in R, and vice versa. In particular, this allows a node in L to directly activate a link to a node v with some specific ID. At the start of each round $r \geqslant 2$ every node u learns the IDs of the nodes that activated a link to u in round $r-1$.

Our Contributions. We present several new algorithms and a lower bound for computing a perfect matching in complete bipartite graphs. As elaborated above, our results also hold in the CONGEST model where the per-link bandwidth restriction is just one bit. The complete proofs of our results can be found in the full version of the paper [4].

- In Section 2, we give a simple randomized algorithm that terminates in $O(\log n)$ rounds with a pulse complexity of $O(n \log n)$ with high probability. The algorithm assumes the port numbering model.
- Then, we focus on the impact of the more powerful KT_1 assumption, where nodes know the IDs of their neighbors, and show that this yields significantly improved bounds:
  - We show that the KT_1 assumption is sufficient to circumvent the lower bound of Section 4, by giving a deterministic algorithm that computes a perfect matching in $O(\log n)$ rounds and sends only $O(n)$ pulses.
  - Next, we leverage this deterministic algorithm to obtain an exponential improvement in the number of rounds, yielding a time complexity of $O(\log \log n \cdot \log^* n)$, at the cost of a slightly increased pulse complexity of $O(n \log^* n)$.
- Finally, in Section 4, we demonstrate that randomization is indeed crucial for achieving a low pulse complexity or message complexity in the port numbering model. We show that any deterministic algorithm must activate at least $\Omega(n^2)$ links in the port numbering model.

2 An Algorithm for Perfect Matching in KT_0

Throughout this section, we consider a complete bipartite graph of $2n$ nodes, $G = (V, E)$ where the vertices are partitioned into two disjoint subsets L and R. Nodes in L are referred to as *left* nodes, and nodes in R are referred to as *right* nodes. Note that each node knows whether it is a left or a right node. The algorithm is split into two stages. Stage 1 consists of $T = \lceil \log n \rceil - \Theta(\log \log n)$ phases with the goal to reduce the number of unmatched nodes to $O(\log n)$. Stage 2 deals with the remaining $O(\log n)$ unmatched nodes. Each phase consists of a constant number of rounds of communication. In phase 1, each of the left nodes randomly selects one port to send a pulse to the right nodes. If a right node receives a pulse, then it will be matched by randomly selecting one left node from

which it has received the pulse to match. We show that with high probability, after the end of phase 1, the number of unmatched nodes reduces by $c := \frac{11}{12}$. Note that the node does not know which port leads to an unmatched neighbor until a pulse is exchanged between the two connected ports. Hence, a pulse sent by an unmatched left node might hit a right node that is already matched. To ensure sufficient unmatched left nodes get matched, in phase 2, each unmatched left node now sends $\lfloor 1/c \rfloor$ pulses. Again, we can show that with high probability, after the end of phase 2, the number of unmatched nodes is reduced further by c. Ultimately, we can show that, for each subsequent phase $i \leqslant T$, the unmatched nodes send $O(\frac{1}{c^{i-1}})$ pulses, and with high probability, the number of unmatched nodes is at most $c^i n$ at the end of each phase. As a result, with high probability, the number of unmatched nodes after T phases is $O(\log n)$.

In Stage 2, a matching for the remaining $O(\log n)$ unmatched nodes is constructed in $O(\log n)$ phases and using $O(n \log n)$ pulses. The key idea to achieve the desired number of pulses and the time complexity bound in Stage 2 is that, by exchanging $O(n \log n)$ pulses in total, each unmatched node learns *all* the ports that lead to its unmatched neighbors in each phase.

Theorem 1. *There exists a randomized algorithm that, with high probability, constructs a perfect matching on a complete bipartite graph with* $2n$ *nodes in* $O(\log n)$ *time and sends* $O(n \log n)$ *pulses.*

3 More Efficient Algorithms under the KT_1 Assumption

In this section, we present two deterministic algorithms that leverage the knowledge of their neighbors' IDs to not only break the lower bound of Section 4, but also improve over the performance of the randomized algorithm in the port numbering model given in Section 2.

Here, we assume the standard KT_1 model, where nodes have unique integer IDs, and each node starts out knowing the IDs of its neighbors. We say that node $u \in L$ (resp. R) has *rank* i, if its ID is the i-th smallest among all nodes in L (resp. R). Note that due to KT_1 assumption, each node in L can locally compute the rank of each node in R and vice versa. However, each node is unaware of its own rank. We remark that if each node knew its own rank (which is the case if we assume KT_2 [1], where each node knows the IDs of all nodes at most two hops away from it), then all nodes can output a matching instantly without any communication. Returning to our KT_1 setting, this motivates the algorithmic design strategy of informing each node of its own rank. In the CONGEST model, where each message can carry $O(\log n)$ bit, it takes just two rounds and $O(n)$ messages to inform each left node of its rank. Concretely, in the first round, each left node sends a message to the smallest rank right node r_0. In the second round, r_0 sends one message to each of the left nodes to inform each of them of their rank. On the other hand, in our link activation model, where we can only signal a pulse, a naive implementation of the above approach would take $O(\log n)$ rounds and $O(n \log n)$ pulses, which would not improve over the port

numbering algorithm of Section 2: When using the convention that a 0 bit sent over a link corresponds to non-activation of the link, and a 1 bit corresponds to a pulse, in round i, node r_0 sends the i-th bit of the binary representation of j to the left node of rank j. This can be completed in $O(\log n)$ rounds. [3] We now describe the algorithm Fast-Interval-Matching, which achieves optimal pulse complexity of $O(n)$ and terminates in $O(\log n)$ rounds.

Interval Construction: We describe a procedure called Interval-Construction that partitions the nodes in L and R into equal size intervals. As we will instantiate this method for subgraphs of different sizes, we state it under the assumption that $|L| = |R| = N$, where N is any positive integer that is greater than 2. We use $l_0, \ldots, l_{N-1}$ to denote the nodes in L ordered in increasing order of their IDs, and analogously define the order $r_0, \ldots, r_{N-1}$ of the nodes in R. Note that this means the rank of l_i as well as r_i is i. Each node in L activates a link to the node r_0, who has the smallest ID in R. We define $s = \lceil \log N \rceil$.

Helper Inform Step: r_0 recruits a set H of s *helpers*, by contacting the nodes with the s-smallest ranks in L, and informing these nodes of their respective rank in L. In more detail, r_0 executes $O(\log s)$ rounds, where in each round i, r_0 sends the i-th bit of rank j to l_j, for each $j = 0, \ldots, s-1$ in parallel.

Partition Step: Our next goal will be to split R into N/s intervals,[4] denoted by $R_0, \ldots, R_{N/s-1}$. For every $i \in [0, N/s-2]$, interval R_i contains exactly s nodes, whereby $R_{N/s-1}$ contains at most $s-1$ nodes. We will obtain the intervals $L_0, \ldots, L_{N/s-1}$ by partitioning L analogously. To implement this partitioning process, we make use of the helper nodes to identify every node $r_{j \cdot s}$, for $j \in \{0, \ldots, n/s-1\}$, as the *leader of interval* R_j. To this end, we need to make sure that each of these interval leaders learns its own rank. Concretely, for every $r_{j \cdot s}$, each helper l_i sends the i-th bit of $r_{j \cdot s}$' rank to $r_{j \cdot s}$ (where 0 and 1 bits are encoded as pulses or absence of pulses, as described above). Note that all helpers perform this operation in parallel, and thus this requires just 1 round. Afterwards, each interval leader $r_{j \cdot s}$ activates a link to $l_{j \cdot s}$. Since the *leader of* L_j, which is $l_{j \cdot s}$, knows the rank of $r_{j \cdot s}$, a pulse is sufficient for it to learn its own rank. Then, we add all edges $\{l_{j \cdot s}, r_{j \cdot s}\}$ to the matching, for each j.

Interval Inform Step: In one additional round, the leader of L_j sends a pulse to each node in interval R_j, and similarly the leader of R_j sends a pulse to the nodes in the L_j, for every j. This completes the Interval-Construction procedure.

Parallel Interval Matching: Finally, we compute a matching in parallel on each interval: Each leader in R_j, i.e., node $r_{j.s}$, sends a pulse to the nodes in L_j sequentially, by activating one link per round. This allows each node in L_j to deduce its rank relative to the other nodes in L_j. In particular, node $l_{j \cdot s+k}$ will have its link activated after k rounds, prompting it to activate the link to $r_{j \cdot s+k}$,

[3] An alternate approach for each left node to learn its own rank is by performing the standard binary search. That is, each left node l initially guesses its rank i and sends a pulse to the right node r of rank i. In two rounds, r can inform l if its guess is equal, higher, or lower than the correct value.

[4] To simplify the notation, we assume that N/s is an integer.

to which it becomes matched. Since there are s nodes per interval this completes in $O(s)$ rounds.

Theorem 2. *Fast-Interval-Matching computes a perfect matching in $O(\log n)$ rounds and with $O(n)$ pulse complexity, assuming the KT_1 model.*

We now employ procedure Interval-Construction as a building block for obtaining an exponential improvement in the time complexity, while increasing the message complexity by a factor of $\log^* n$.

Theorem 3. *There is a deterministic algorithm that computes a perfect matching in $O(\log^* n \cdot \log\log n)$ rounds and with a pulse complexity of $O(n \log^* n)$ under the KT_1 assumption.*

4 A Lower Bound for Deterministic Algorithms

In this section, we return to the port numbering model and show that randomization is crucial for any perfect matching algorithm that uses a small number of communication links. We make several assumptions that strengthen our lower bound, namely, we consider the standard LOCAL model [6], where nodes may send messages of arbitrary size in each round, and deterministically label the nodes in L with IDs from $\{1, \ldots, n\}$ and the nodes in R with IDs from $\{n+1, \ldots, 2n\}$.

Theorem 4. *Any deterministic algorithm that constructs a perfect matching in the complete bipartite graph of $2n$ nodes has a message complexity of $\Omega(n^2)$ in the LOCAL model.*

Acknowledgements. Peter Robinson was supported in part by National Science Foundation (NSF) grant CCF-2402836. Ming Ming Tan was supported in part by National Science Foundation (NSF) grant CCF-2348346. Xianbin Zhu was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. CityU 11213620].

References

1. Baruch Awerbuch, Oded Goldreich, David Peleg, and Ronen Vainish. A trade-off between information and communication in broadcast protocols. *J. ACM*, 37(2):238–256, 1990.
2. Raymond Cheong, Alex Rhee, Chiaochun Joanne Wang, Ilya Nemenman, and Andre Levchenko. Information transduction capacity of noisy biochemical signaling networks. *science*, 334(6054):354–358, 2011.
3. Alejandro Cornejo and Fabian Kuhn. Deploying wireless networks with beeps. In *International Symposium on Distributed Computing*, pages 148–162. Springer, 2010.
4. Hugo Mirault, Peter Robinson, Ming Ming Tan, and Xianbin Zhu. Perfect matching with few link activations. *arXiv preprint (to appear)*, 2025.
5. Saket Navlakha and Ziv Bar-Joseph. Distributed information processing in biological and computational systems. *Communications of the ACM*, 58(1):94–102, 2014.
6. David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, Philadelphia, 2000.
7. Jukka Suomela. Survey of local algorithms. *ACM Comput. Surv.*, 45(2):24:1–24:40, 2013.