How Helpers Hasten h-Relations*

Peter Sanders†
Max Planck Insitut für Informatik
Saarbrücken, Germany
sanders@mpi-sb.mpg.de

Roberto Solis-Oba‡
Department of Computer Science
The University of Western Ontario
London, ON, Canada
solis@csd.uwo.ca

Abstract

We study the problem of exchanging a set of messages among a group of processors, where messages may consist of different numbers of packets. We consider the model of half-duplex communication. Let h denote the maximum number of packets that a processor must send and receive. If all the packets need to be delivered directly, at least $\frac{3}{2}h$ communication steps are needed to solve the problem in the worst case. We show that by allowing forwarding, only $\frac{6}{5}h + \mathcal{O}(1)$ time steps are needed to exchange all the messages, and this is optimal. Our work was motivated by the importance of irregular message exchanges in distributed-memory parallel computers, but it can also be viewed as an answer to an open problem on scheduling file transfers posed by Coffmann, Garey, Johnsson, and LaPaugh in 1985.

1 Introduction

Consider a group of P processing elements (PEs) numbered 0 through $P-1$, connected by a complete network. For every pair (i, j) of processing elements there is a message consisting of $m_{ij} \geq 0$ packets that i must sent to j. For PE i let h_i be the total number of packets that it must send plus the total number of packets that it must receive. Let $h = \max \{ h_i \mid i = 0, \ldots, P-1 \}$. The *$h$-relation problem* is to send all these packets in the smallest amount of time. Our unit of communication time is the time needed to transmit one packet. We assume synchronized half-duplex communication, i.e., a PE can only either send or receive one packet at any given moment.

*A preliminary version of this paper was published in the proceedings of the 8th European Symposium on Algorithms (ESA 2000).

†Partially supported by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

‡Partially supported by Natural Sciences and Engineering Research Council of Canada grant R3050A01.

This problem has been studied in many variations and under many different names: h-relations [6], file transfer [2], edge coloring [19], and biprocessor task scheduling on dedicated processors [13]. Our original motivation was the study of the function `MPI_Alltoallv` in the Message Passing Interface (MPI) [22] and its equivalents in other message passing models for parallel computing.

The problem can be modeled using an undirected multi-graph $G = (V, E)$ called the *transfer graph*. In this graph $V = \{0, \ldots, P-1\}$, and for every pair of vertices i, j, there is an edge (i, j) of multiplicity $m_{ij} + m_{ji}$. The maximum degree $h(G)$ of G is a natural measure for the size of the problem and it yields a trivial lower bound for the time needed to exchange all the messages. When there is no confusion, we use h to denote the maximum degree of the transfer graph.

A simple reduction to the chromatic index problem [2] shows that the h-relation problem is NP-hard in the strong sense even when all messages have length 1. In the chromatic index problem, given a graph G and an integer k it is desired to know whether the edges of G can be colored using at most k colors, so that no two edges with the same color share a common endpoint. We note that a coloring of the edges of a transfer graph yields an upper bound on the value of the solution for the h-relation problem, since the color of an edge (i, j) can be interpreted as the time step in which the packet corresponding to this edge should be transmitted. The chromatic index $\chi'(G)$ of G is the minimum number of colors needed to edge-color G as described above. It is known that if G does not have multiple edges then $h(G) \leq \chi'(G) \leq h(G) + 1$ [24]. Gabow et al. [4] give an $\mathcal{O}\big(m\sqrt{P \log P}\big)$-time algorithm to edge-color a simple graph with at most $h(G) + 1$ colors, where m is the number of edges.

If a message cannot be divided into packets, then the h-relation problem is NP-hard even when the underlying transfer graph is bipartite, or when it is a tree [2]. But the problem can be solved in linear time if the transfer graph is a path.

For the case of multi-graphs, it is known [10] that the chromatic index problem cannot be approximated with a ratio smaller than $\frac{4}{3}$ unless P=NP. There are instances of the problem for which $\lceil 3h/2 \rceil$ colors are needed. Nishizeki and Sato [18] present a $\frac{4}{3}$-approximation algorithm for the problem, and Nishizeki and Kashiwagi [19] give an algorithm to edge-color a graph G using at most $1.1\chi'(G)+0.8$ colors. It is conjectured that there is a polynomial time algorithm that can find a coloring for any graph G using $\chi'(G) + 1$ colors [9, 17, 19].

In the *regular* h-relation problem every message m_{ij} has length $h/(2(P-1))$. In Section 4.1 we show that this problem can be solved using a *1-factorization* [8] of the transfer graph. For the *on-line* version of the problem (when a PE knows only the messages that it must send, but it does not know the lengths of the messages that other PE's must send), there are several algorithms that transform an arbitrary instance of the h-relation problem into two "almost" regular h'-relation problems, with $h \approx h'$ [16, 23]. For all of these algorithms the communication time is about $2h$.

If $h_{\max}$ is the maximum number of packets that some PE sends *or* receives, and if we assume the full-duplex model of communication where each PE can simultaneously send and receive one packet, then the problem can be solved optimally using communication time $h_{\max}$ via bipartite edge coloring [15].

All the above results make the assumption that the packets of every message m_{ij} are directly delivered from PE i to PE j. Coffman et al. suggest that forwarding messages over different PEs might help speed-up the transmission of the messages, since this gives additional scheduling flexibility. Whitehead [25] shows that when forwarding is *needed* because some of the edges in the transfer graph are not present in the interconnection network, then the h-relation problem is NP-complete even if the transfer graph is a path with edges of arbitrary multiplicity. Goldberg et al. [5] give a randomized algorithm for the on-line h-relation problem with forwarding that with high probability finds a solution of length $\Theta(h + \log\log P)$. This algorithm assumes the full-duplex mode of communication.

In this paper we study the h-relation problem with forwarding on a complete interconnection network using half-duplex communication. We make simplifying assumptions which are warranted if the total data volume to be delivered is large. For a more detailed discussion about when our assumptions are practical refer to Section 4.2. First, we assume that the packets can be divided into smaller pieces. Our algorithm only needs to split a packet into at most five pieces. This restriction could be lifted by grouping packets into superpackets of five packets. This approach would complicate the analysis since we have to account for halfempty superpackets and it can increase the length of the solution. Our second assumption is that we schedule off-line, i.e., we do not count the cost of coordinating the processors and scheduling the delivery of packets.

It might be a little surprising that forwarding can be helpful in half-duplex communication since it increases the communication volume. We show that the net tradeoff between this disadvantage and the additional flexibility that forwarding provides, is positive.

We describe an algorithm that reduces the h-relation problem to the problem of scheduling $\lceil h/2 \rceil$ 2-*relations*. A 2-relation is a set of packets that induce a collection of cycles and paths in the transfer graph. It is easy to schedule a 2-relation in 3 time steps without forwarding, thus this approach yields a solution of length $3\lceil h/2 \rceil$. This is optimal for even h if no forwarding is allowed.[1]

We explain how to use forwarding to find a solution of length 12/5 for the 2-relation problem when P is even. This yields an algorithm for the h-relation problem with even P, that finds a solution of length $\frac{6}{5}(h+1)$. The case of odd P is more difficult. By removing some packets from the 2-relations in a "balanced" way, so that many of the removed packets can be concurrently transmitted later, it is possible to find a solution of length $\left(\frac{6}{5} + \mathcal{O}(1/P)\right)(h+1)$. We also show that there are h-relations where the above bounds cannot be improved by any algorithm regardless of their forwarding strategy.

The rest of the paper is organized in the following way. In Section 2 we present a simple algorithm that solves the h-relation problem without forwarding in $3\lceil h/2 \rceil$ time steps. Then we show how to exploit forwarding to reduce the length of the solution to $\frac{6}{5}(h+1)$ for even P and to $\left(\frac{6}{5} + \mathcal{O}(1/P)\right)(h+1)$ for odd P. We also

[1] For example, consider any transfer graph containing the vertices $\{a, b, c\}$ and $h/2$ copies of the edges (a, b), (b, c), and (c, a).

show that these bounds are almost tight. In Section 3 we explain how to modify our algorithms so that they run in strongly polynomial time. In Section 4 we outline algorithms for solving simple instances of the h-relation problem and we also discuss some practical considerations.

2 Solving the h-Relation Problem

In this section we describe a simple algorithm for the h-relation problem without forwarding, and then we show how to combine this algorithm with the idea of forwarding to get a better algorithm.

2.1 An Algorithm Based on Bipartite Edge Coloring

We first explain how to translate an h-relation into $\lceil h/2 \rceil$ 2-relations. Besides laying the ground for our main result described in the next section, this also yields a good algorithm without forwarding. Since a 2-relation can be solved in 3 units of time, the original h-relation can be solved using communication time at most $3\lceil h/2 \rceil$. To see that a 2-relation can be solved in 3 units of time, we note that a 2-relation induces a collection of disjoint cycles and paths in the transfer graph. Every path and even length cycle can be decomposed into two disjoint matchings, while an odd length cycle can be partitioned into three matchings. It is easy to see that the packets in a matching can be transmitted in one unit of time.

Since in any instance of the h-relation problem at least h rounds are needed to send the messages, the above algorithm achieves a performance ratio no worse than $\frac{3\lceil h/2 \rceil}{h} \leq \frac{3}{2}(1 + \frac{1}{h})$.

The algorithm for translating an h-relation into 2-relations first converts an instance of the h-relation problem into an edge coloring problem on a bipartite graph with maximum degree $\lceil h/2 \rceil$. Consider the transfer graph $G = (V, E)$. Since the sum of the degrees of the vertices in any graph is even, any graph has an even number of vertices with odd degree. Add an edge between every pair of vertices of odd degree, so that every vertex in the graph has even degree. This new graph is Eulerian, and so we can find a collection of edge-disjoint cycles covering all edges in linear time. By traversing these cycles we can assign orientations to the edges of the graph so that for every vertex its in-degree and out-degree are the same. Let $G' = (V, E')$ denote the resulting directed graph. In this graph the maximum in-degree and out-degree are $\lceil h/2 \rceil$.

Now build an undirected bipartite graph $\bar{G} = (L, R, \bar{E})$ by making two copies $L = R = V$ of the vertices of G', and adding an edge from vertex $u \in L$ to $v \in R$ whenever $(u, v) \in E'$. In this bipartite graph the two copies of any vertex v have the same degree, and hence the maximum degree is $\lceil h/2 \rceil$. Next, compute an optimum edge coloring for $\bar{G}$. This can be done in time $\mathcal{O}(hP \log h)$ [3]. The coloring of the edges induces a decomposition of $\bar{E}$ into $\lceil h/2 \rceil$ disjoint matchings in which every matching consists of edges of the same color. The edges in each matching induce a 2-relation in G. These 2-relations together cover all edges in G.

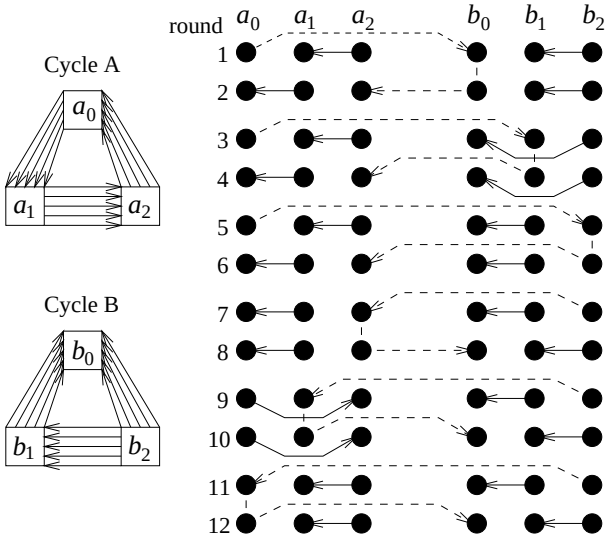

Figure 1: Example for scheduling two 3-cycles. Dashed lines represent forwarded messages.

2.2 Exploiting Forwarding

We now show how the method described in the previous section can be refined to use forwarding to reduce the time needed for exchanging the messages.

Theorem 1 *The h-relation problem can be solved using communication time $\frac{6}{5}(h+1)$ if P is even, and using time $(\frac{6}{5}+\frac{2}{P})(h+1)$ if P is odd.*[2] *This bound is almost tight since for any value h there are problem instances for which at least time $\frac{6}{5}h$ is needed for exchanging the messages when P is even, and at least time $\frac{6}{5}(1+\frac{3}{5P})h$ is needed when P is odd.*

The remainder of Section 2 is dedicated to proving this result. The first part of the algorithm used for the upper bound is the same as that described in the previous section: we build the bipartite graph $\bar{G}$, color it, and derive $\lceil h/2 \rceil$ 2-relations from the colors. But now we use a more sophisticated algorithm for scheduling the 2-relations. Each 2-relation defines a collection of disjoint paths and cycles in the transfer graph G. To avoid some tedious case distinctions, we add dummy edges closing all paths to cycles. It is easy to schedule the packets in an even length cycle so that they can be transmitted in communication time 2. But it is not so easy to find a good schedule for the messages in an odd length cycle. We use forwarding of some data to solve this problem. Note that from now on we have to reintroduce the direction of a packet.

For example, consider the two 3-cycles shown in Figure 1, where the packets have been split in five pieces of size 1/5 each. Clearly, no algorithm without forwarding can exchange all the data in less than 3 time steps since at least two PEs will be idle at any point in time. The idea is to exploit the idle times for forwarding data. The schedule in Figure 1 shows how to transmit all these pieces in 12 rounds of length

[2] In the extended paper [21] we outline how the bound for odd P can be slightly improved to $(\frac{6}{5}+\frac{9}{10P})(h+1)$.

1/5 each, thus using total communication time 12/5. During the first 6 rounds, PE a_2 sends three pieces of packet (a_2, a_0) to PE a_0 with the help of the PEs in cycle B. In the last 6 rounds, PE b_2 sends 3 pieces of packet (b_2, b_0) to PE b_0 with help from the PEs in cycle A.

In the next two sections we show how to generalize this idea to reduce the time needed to exchange the messages for any number P of PEs.

2.3 Even Number of PEs

When the number P of PEs is even, there is an even number of odd length cycles (ignoring the directions of the edges). We pair the odd cycles and use the PEs in one cycle to help forward the messages of the other cycle, just like we did in Figure 1. We now explain how to schedule the packets in a pair of odd length cycles A and B. As in the example of Figure 1, the packets are split into 5 pieces, and these pieces are exchanged in 12 rounds of length 1/5 each as described below.

If one of the cycles consist only of an idle node, this node can easily help the other cycle by forwarding three of its pieces. Otherwise, let us name the PEs of cycle A as $a_0, \ldots, a_{|A|-1}$ in such a way that $a_{|A|-1}$ sends a packet to a_0 (and not vice versa). Similarly, let $b_0, \ldots, b_{|B|-1}$ denote the PEs in cycle B, and let $b_{|B|-1}$ send a packet to b_0. Figure 2 summarizes the schedule for exchanging the packets of cycles A and B.

In rounds $2i$ and $2i+1$, for $i \in \{0, 1, 2\}$, b_i forwards one piece of the packet from $a_{|A|-1}$ to a_0. Concurrently, three pieces of every other packet in cycle A are transmitted directly: in round $2i$ one piece of the packet between a_{2k} and a_{2k+1} is transmitted, and in round $2i+1$, one piece of the packet between a_{2k+1} and a_{2k+2} is transmitted, for every $0 \leq k < |A|/2$. Thus, within six rounds, three pieces of every packet in cycle A are transmitted.

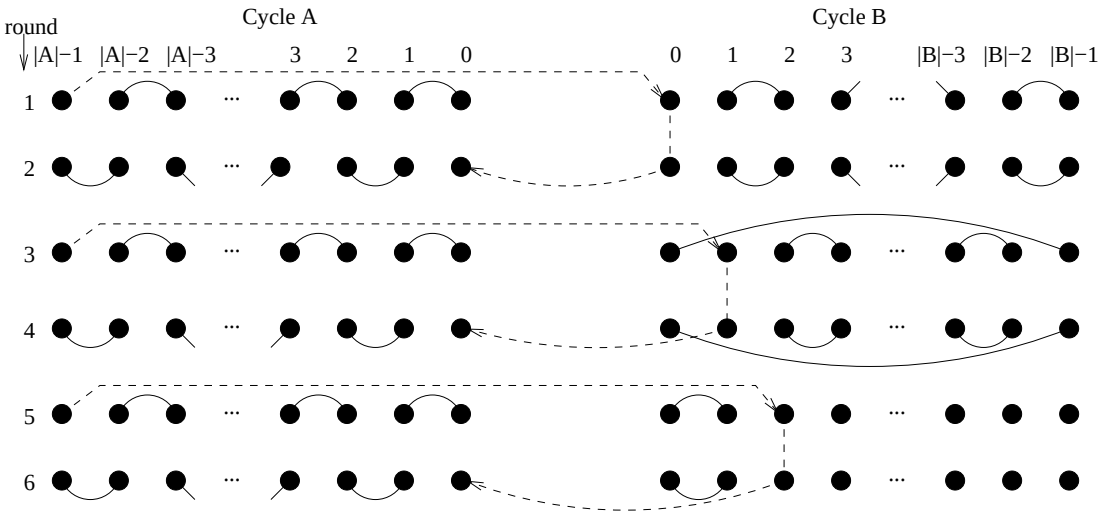

Figure 2: How odd cycle B helps odd cycle A by forwarding three pieces of a packet. Undirected edges must be given the direction of the corresponding edge in the transfer graph.

As for cycle B, in rounds one and two, two pieces of the packet between b_{2j+1} and b_{2j+2} are transmitted for every $0 \leq j < |B|/2$. In rounds three and four, two pieces of the packet between b_{2j} and $b_{2j+1 \bmod |B|}$ are transmitted for every $0 < j < |B|/2$. In rounds five and six, only two pieces of the packet between b_0 and b_1 are transmitted. Thus, within six rounds two pieces of every packet in cycle B are transmitted.

In rounds seven through twelve, cycles A and B switch their roles, so that after round twelve all five pieces of the packets are transmitted. The total time needed to exchange all the messages in the transfer graph G is $\frac{12}{5}\lceil h/2 \rceil = \frac{6}{5}(h+1)$ This establishes the upper bound of Theorem 1 for even P.

2.4 Odd Number of PEs

If the number of PEs is odd, there will be an odd number of odd cycles in each 2-relation, so it is not possible to pair cycles as before. It is not difficult to see that there are 2-relations with an odd number of PEs that cannot be scheduled in twelve rounds of length 1/5 each.

We solve this problem as follows. In those 2-relations which contain idle PEs (1-cyles) we use one idle PE as an unpaired cycle and proceed as in the case of even P. For the remaining 2-relations, we choose one of the cycles A and remove one packet from it. If A is an odd length cycle, the removal of a packet transforms it into a path whose packets can be scheduled in communication time 2 without help from another cycle. Moreover, all the remaining odd length cycles can be paired and their packets exchanged as described in the previous section. If the chosen cycle A has even length, then we can pair it with an odd cycle B. A simple modification of the algorithm described in the previous section can be used to transmit all the packets of A and B in 12/5 units of time.

What remains to be done is to schedule the packets that have been removed. We maintain the invariant that all the removed packets form a matching M in the transfer graph G. Whenever we select a cycle, we try to choose it so that it has a packet that maintains this invariant. If this is possible, we just add this packet to M. Otherwise, if no packet can be added to the matching, then we transmit all the packets in M in one unit of time, emptying the matching M. The process is repeated until all messages are transmitted.

Using this algorithm, we can prove that an additional step for emptying the matching M is only required rarely:

Lemma 1 *Whenever M needs to be emptied, it contains at least $\lceil P/4 \rceil$ edges.*

Proof. Whenever the algorithm removes a packet there are P candidate edges E' to be removed from the cycles. (Recall that we do not remove packets from 2-cycles with idle PEs.) Every edge $e \in M$ can have a common endpoint with at most 4 edges from E'. Hence, if $|M| < P/4$ there must be at least one candidate edge in E' that can be added to M. ∎

To summarize, the solution produced by the above algorithm needs time $\frac{12}{5}\lceil h/2 \rceil$ for transmitting the packets in the cycles, plus $\lceil h/2 \rceil / \lceil P/4 \rceil$ units of time for for

emptying the matchings M. The total length of the solution is then less than $(\frac{6}{5} + \frac{2}{P})(h+1)$.

2.5 Lower Bound

In this section we concentrate on the case of odd P. The case of even P is similar. Consider the following instance of the problem with $P = 3k$, for some odd $k > 0$. For every $0 \leq i < k$, there are messages $m_{3i,3i+1}$, $m_{3i+1,3i+2}$, and $m_{3i+2,3i}$ of length $h/2$, for some even h. All other messages are empty. Consider any algorithm $\mathcal{A}$ for exchanging these messages. Let $D(t)$ denote the number of packets being directly routed by $\mathcal{A}$ to their destinations at time t. Note that $D(t) \leq P/3$.

There are at most $P - 2D(t)$ other PEs available for forwarding packets. Since $P - 2D(t)$ is odd, these PEs can handle at most $\frac{P-2D(t)-1}{2}$ packets at any time. Since forwarded packets have to be sent at least twice, we define *the progress* made by $\mathcal{A}$ at time t towards delivering the packets to their final destinations to be $D(t) + \frac{P-2D(t)-1}{4}$. The integral[3] of the progress over the total communication time T of the solution produced by $\mathcal{A}$ must be equal to the the total volume $hP/2$ of the data. Hence,

$$\begin{aligned} h\frac{P}{2} &= \int_{t=0}^{T} \left(D(t) + \frac{P - 2D(t) - 1}{4} \right) dt = \frac{1}{2} \int_{t=0}^{T} D(t)dt + T\left(\frac{P}{4} - \frac{1}{4}\right) \\ &\leq \frac{1}{2} \int_{t=0}^{T} \frac{P}{3} dt + T\left(\frac{P}{4} - \frac{1}{4}\right) = T\left(\frac{5P}{12} - \frac{1}{4}\right) \quad . \end{aligned}$$

Solving this for T yields $T \geq h/(\frac{5}{6} - \frac{1}{2P}) \geq \frac{6}{5}(1 + \frac{3}{5P-3})h$. ∎

3 A Strongly Polynomial Time Algorithm

The algorithms described in the previous section do not have strongly polynomial running times since the number of edges in the bipartite multi-graph $\bar{G}$ depends on the lengths of the messages that must be exchanged. In this section we show that it is possible to modify our algorithms so that their time complexities are independent of the message lengths, therefore, obtaining strongly polynomial time algorithms.

An instance of the h-relation problem can be represented by a weighted transfer digraph $G = (V, E)$ in which the weight w_{ij} of an edge (i, j) is equal to the length of the message m_{ij}. Let m be the number of edges and δ be the maximum degree of G. Define the *load* of a processing element i as the total length of the messages that must be sent and received by i. As before h is the largest PE load.

We construct a bipartite graph $\bar{G} = (L, R, \bar{E})$. For every edge $(i, j) \in E$ we add edges (i, j), $i \in L$, $j \in R$, and (j, i), $j \in L$, $i \in R$, with weight $\bar{w}_{ij} = \lfloor w_{ij}/2 \rfloor$. Next, we build an unweighted multi-graph G' containing only the edges of G with odd weight. With G' we proceed as before: add dummy edges to make all degrees

[3] We integrate over continuous time rather than sum over time steps in order to cover asynchronous algorithms for which there is no natural notion of time steps. Note that we get no problems regarding the existence of the integral since the integrand is continuous almost everywhere.

even, and find a collection of Euler cycles to orient the edges in G'. Finally, for every edge $(i,j) \in G'$ we increment the corresponding weight $\bar{w}_{ij}$ in the bipartite graph $\bar{G}$.

Now we find a matching $\bar{M}$ of $\bar{G}$ that covers all nodes corresponding to PEs with maximum load. Such a matching must exist because if we replace every edge (i,j) of $\bar{G}$ by $\bar{w}_{ij}$ copies of weight 1 each, we obtain a bipartite graph that can be colored with $\lceil h/2 \rceil$ colors. Each color in this coloring induces a matching covering all nodes corresponding to PEs with maximum load. The matching $\bar{M}$ can be found by computing a maximum cardinality matching on the subgraph of $\bar{G}$ formed by those edges incident to nodes from PEs with maximum load. This can be done in $\mathcal{O}\big(m \log \delta + \frac{m}{\delta} \log \frac{m}{\delta} \log^2 \delta\big)$ time using the algorithm in [11].

Let $w_{\min}$ be the smallest weight in $\bar{M}$. Let h' be the second largest load among the PEs. We exchange the messages in $\bar{M}$ as described in the previous section, using *super-packets* formed by $\min\{h - h', w_{\min}\}$ packets each. After doing this we modify the bipartite-graph $\bar{G}$ by decreasing the weight of every edge in $\bar{M}$ by $\min\{h - h', w_{\min}\}$ and discarding all edges of weight zero. Let h be the new largest PE load. The process is repeated until all messages have been sent. Note that each iteration of this process either removes one edge from $\bar{G}$, or it increases by at least one the number of PEs with largest load. Therefore the process requires $\mathcal{O}(m + P)$ iterations.

4 Special Cases and Practical Considerations

4.1 Faster Exchange of Easy h-Relations

Some h-relations can be routed in less than $6h/5$ units of time. If the chromatic index of the transfer graph G is smaller than $12h/11$ such a schedule can be found using the edge coloring algorithm of Nishizeki and Kashiwagi [19] without using forwarding. It is an open problem to improve this by combining such general edge coloring algorithms with the idea of forwarding.

Hall et al. [7] observe that an h-relation can be routed in h steps if $P/4$ PEs have no data to send or receive — these PEs can be used as helpers. It is an interesting question whether the more general statements holds that h steps suffice when there are at most $\frac{3}{4} \cdot Ph/2$ packets to send.

Another simple but useful observation is that instances with $\ell := \min_{i \neq j}\{m_{ij} + m_{ji}\} > 0$ can be decomposed into two instances, one of which consists solely of messages of length ℓ. The packets in this regular instance can be scheduled using total communication time $(P-1)\ell$ ($P\ell$ for odd P). To our surprise, there seemed to be no optimal algorithm for this purpose in the parallel processing literature, except for the case that P is a power of 2 [14]. Here is a simple optimal algorithm for this problem. For odd P, there are P rounds $0,\ldots,P-1$. In round i, PE j exchanges its data with PE $(i-j) \bmod P$. In each round, the PE with $i - j = j \bmod P$ is (unavoidably) idle. Each PE is idle exactly in one round. For even P, PEs $0,\ldots,P-2$ execute the above algorithm except that the otherwise idle PE exchanges data with PE $P-1$. We do not go into more detail here since this algorithm turns out to

be equivalent to a long known graph-theoretical result, namely the factorization of a clique into 1-factors [12, 8].

4.2 Is This Practical?

For moderate P and long messages, our algorithm might indeed be useful.

Let us first look at a prototypical application where message lengths of h-relations can be very uneven. Consider the main communication phase of sample sort [1] for sorting $n \gg P^2B$ elements where B is the number of elements fitting into a packet. Sample sort needs an h-relation with $h \approx 2n/(PB)$. Although randomization makes sure that all PEs have to communicate about the same amount of data, the individual message lengths can vary arbitrarily for worst case inputs. In this case, our algorithm can be a large factor faster than a naive direct exchange and still yields a factor of up to $2/\frac{6}{5} = \frac{5}{3}$ speedup over two-phase algorithms. The number of PEs should be moderate for several reasons. Firstly, we assume that the network itself is not a bottleneck which is usually only the case for machines up to around 64 PEs.[4] Secondly, the scheduling overhead grows with P. However, in some cases it might be possible to amortize the scheduling overhead over multiple calls for h-relations with identical structure. Many iterative numerical algorithms working on irregular data are of this type.

Message lengths are even more critical. The main requirement is that one fifth of a packet is still so large that the startup overhead for communication is small compared to the transmission overhead itself. For so large packets the assumption of synchronized communication is quite realistic since most communication subsystems have a built-in flow control mechanism that enforces synchronization for large data transmissions. Also, measurements with MPI [20] indicate that synchronized communisation of large packets is among the fastests protocols on most parallel machines.

Similar observations can be made for the particular assumption of half-duplex communication. Although many machines allow full-duplex communication on the physical level of the network, currently none of the measurements in [20] indicate that MPI can profit from that. (Otherwise the Benchmark for bidirectional communication with `MPI_Sendrecv` should allow close to twice the bandwidth of the corresponding unidirectional communications with `MPI_Ssend` and `MPI_Recv`.)

References

[1] G. E. Blelloch, C. E. Leiserson, B. M. Maggs, C. G. Plaxton, S. J. Smith, and M. Zagha. A comparison of sorting algorithms for the connection machine CM-2. In *ACM Symposium on Parallel Architectures and Algorithms*, pages 3–16, 1991.

[2] E. G. Coffman, M. R. Garey, D. S. Johnson, and A. S. LaPaugh. Scheduling file transfers. *SIAM Journal on Computing*, 14(3):744–780, 1985.

[4] But note that some machines consist of multiprocessor nodes connected by a crossbar. On such machines our algorithm might be useful for scheduling data exchanges between nodes.

[3] R. Cole, K. Ost, and S. Schirra. Edge-coloring bipartite multigraphs in $O(E \log D)$ time. *submitted for publication*, 2000.

[4] H. N. Gabow and O. Kariv. Algorithms for edge coloring bipartite graphs and multigraphs. *SIAM Journal on Computing*, 11(1):117–129, 1982.

[5] L. A. Goldberg, M. Jerrum, T. Leighton, and S. Rao. Doubly logarithmic communication algorithms for optical-communication parallel computers. *SIAM Journal on Computing*, 26(4):1100–1119, August 1997.

[6] M. D. Grammatikakis, D. F. Hsu, M. Kraetzl, and J. Sibeyn. Packet routing in fixed-connection networks: A survey. *Journal of Parallel and Distributed Processing*, 54:77–132, 1998.

[7] J. Hall, J. Hartline, A. R. Karlin, J. Saia, and J. Wilkes. On algorithms for efficient data migration. In *12th ACM-SIAM Symposium on Discrete Algorithms*, pages 620–629, 2001.

[8] F. Harary. *Graph Theory.* Addison Wesley, 1969.

[9] D. S. Hochbaum, T. Nishizeki, and D. B. Shmoys. A better than "best possible" algorithm to edge color multigraphs. *Journal of Algorithms*, 7:79–104, 1986.

[10] Ian Holyer. The NP-completeness of edge-coloring. *SIAM Journal on Computing*, 10(4):718–720, 1981.

[11] A. Kapoor and R. Rizzi. Edge-coloring bipartite graphs. *Journal of Algorithms*, 34(2):390–396, 2000.

[12] D. König. *Theorie der endlichen und unendlichen Graphen.* Akademische Verlagsgesellschaft, 1936.

[13] M. Kubale. Preemptive versus nonpreemptive scheduling of biprocessor tasks on dedicated processors. *European Journal of Operational Research*, 94:242–251, 1996.

[14] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing. Design and Analysis of Algorithms.* Benjamin/Cummings, 1994.

[15] G. Lev, N. Pippenger, and L. Valiant. A fast parallel algorithm for routing in permutation networks. *IEEE Trans. on Comp.*, C-30, 2:93–100, 1981.

[16] W. Liu, C. Wang, and K. Prasanna. Portable and scalable algorithms for irregular all-to-all communication. In *16th ICDCS*, pages 428–435. IEEE, 1996.

[17] S. Nakano, X. Zhou, and T. Nishizeki. Edge-coloring algorithms. In *Computer Science Today*, number 1000 in LNCS, pages 172–183. Springer, 1996.

[18] T. Nishizeki and M. Sato. An algorithm for edge-coloring multigraphs. *Trans. Inst. Electronics and Communication Eng.*, J67-D(4):466–471, 1984. (in Japanese).

[19] Takao Nishizeki and Kenichi Kashiwagi. On the 1.1 edge-coloring of multigraphs. *SIAM Journal on Discrete Mathematics*, 3(3):391–410, August 1990.

[20] R. Reussner, P. Sanders, L. Prechelt, and M. Müller. SKaMPI: A detailed, accurate MPI benchmark. In *EuroPVM/MPI see also* `http://liinwww.ira.uka.de/~skampi/`, number 1497 in LNCS, pages 52–59, 1998.

[21] P. Sanders and R. Solis-Oba. How helpers hasten h-relations. Technical Report 559, The University of Western Ontario, Department of Computer Science, March 2001.

[22] M. Snir, S. W. Otto, S. Huss-Lederman, D. W. Walker, and J. Dongarra. *MPI – the Complete Reference*. MIT Press, 1996.

[23] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Conference Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computation*, pages 263–277, Milwaukee, Wisconsin, 11–13 May 1981.

[24] V. G. Vizing. On an estimate of the chromatic class of a p-graph (in russian). *Diskret. Analiz*, 3:23–30, 1964.

[25] J. Whitehead. The complexity of file transfer scheduling with forwarding. *SIAM Journal on Computing*, 19(2):222–245, April 1990.