A Powerful Heuristic for Telephone Gossiping

Rene Beier and Jop F. Sibeyn

MPI–I–2000–1-002 May 2000

Author's Address

Rene Beier and Jop F. Sibeyn
Max-Planck-Institut für Informatik
Im Stadtwald, 66123 Saarbrücken, Germany
E-mail: rbeier, jopsi@mpi-sb.mpg.de
URL: http://www.mpi-sb.mpg.de/~jopsi/

Abstract

A refined heuristic for computing schedules for gossiping in the telephone model is presented. The heuristic is fast: for a network with n nodes and m edges, requiring R rounds for gossiping, the running time is $\mathcal{O}(R \cdot n \cdot \log n \cdot m)$ for all tested classes of graphs. This moderate time consumption allows to compute gossiping schedules for networks with more than 10,000 PUs and 100,000 connections. The heuristic is good: in practice the computed schedules never exceed the optimum by more than a few rounds. The heuristic is versatile: it can also be used for broadcasting and more general information dispersion patterns. It can handle both the unit-cost and the linear-cost model.

Actually, the heuristic is so good, that for CCC, shuffle-exchange, butterfly de Bruijn, star and pancake networks the constructed gossiping schedules are better than the best theoretically derived ones. For example, for gossiping on a shuffle-exchange network with 2^{13} PUs, the former upper bound was 49 rounds, while our heuristic finds a schedule requiring 31 rounds. Also for broadcasting the heuristic improves on many formerly known results.

A second heuristic, works even better for CCC, butterfly, star and pancake networks. For example, with this heuristic we found that gossiping on a pancake network with 7! PUs can be performed in 15 rounds, 2 fewer than achieved by the best theoretical construction. This second heuristic is less versatile than the first, but by refined search techniques it can tackle even larger problems, the main limitation being the storage capacity. Another advantage is that the constructed schedules can be represented concisely.

1 Introduction

Gossiping. Collective communication operations occur frequently in parallel computing, and their performance often determines the overall running time of an application. One of the fundamental communication problems is *gossiping* (also called total exchange or all-to-all non-personalized communication). Gossiping is the problem in which every processing unit, PU, wants to send the same packet to every other PU. Said differently, initially each of the n PUs contains an amount of data of size h, and finally all PUs know the complete data set of size $h \cdot n$. This is a very communication intensive operation. Gossiping appears in all applications in which the PUs operate autonomously for a while, and then must exchange all gathered data to update their databases. Gossiping belongs to the most investigated communication problems. Many aspects of the problem have been investigated for all kinds of interconnections networks [2, 4, 5, 6, 10, 14, 19].

We study gossiping in the *telephone model*: it is assumed that a PU can exchange data with only one other PU at a time, and that this connection is bidirectional. We focus on networks with a known but not necessarily regular structure. Such networks may represent a set of nodes in the internet, the servers of a banking institution or the processors of a parallel computer. As computing optimal gossiping schedules is NP hard for general networks, we have to resort to algorithms that find sub-optimal schedules.

Heuristics. In this paper we present two heuristics for constructing gossiping schedules and our experiences with them.

The *matching heuristic* combines simplicity and versatility and gives very good performance. It can handle both the unit-cost and the linear-cost model (all definitions are given in Section 2) and all kinds of initial packet distributions. Particularly, it is also suited for computing broadcasting schedules. The matching heuristic operates in rounds. In each round, it constructs a maximum weighted matching of the graph underlying the interconnection network. The pairs of matched PUs communicate. The non-trivial part is how to set the weights so that the gossiping time is minimized. In the linear-cost model, one also has to determine how much and which data is going to be communicated. Other interesting aspects are the value of look-ahead, and whether one might also compute approximate matchings without incurring performance losses.

The *coloring heuristic* works differently: initially a small set of matchings is constructed, and then schedules composed of these matchings are tested. Basically, the algorithm performs an exhaustive search through all possible schedules, but the order in which the schedules are tested is optimized and many less promising schedules are pruned out. In principle the coloring heuristic can be applied to any network, but it is most useful for g-regular networks that allow a g coloring: a decomposition of all edges in g perfect matchings.

Previous Work. Heuristics have been applied for computing communication schedules since many years [18]. The matching heuristic has been applied to several communication problems by Fraigniaud and Vial [7, 8, 9]. Though the underlying idea is the same, our paper goes beyond [7] in many respects. In [7], the matchings are computed for graphs that are weighted by considering the number of packets that may be transferred over each edge (for point-to-point communication, in [8] a modified weighting is applied to keep packets on a shortest path). This is a good idea, in Section 4, we consider it under the name *potential approach*, but often substantially better results can be achieved by attributing the edge weights according to more global criteria, as is done by our *BFS approach*. Furthermore, we introduce a quite sophisticated technique for gossiping in the linear-cost model; we consider the implications of using approximate matchings; we study the value of look-ahead. The efficiency of our implementation makes the heuristic effective for large graphs, and allowed us to perform sets of experiments that are sufficiently large to draw meaningful conclusions. All this is complemented with the coloring heuristic and the discovery of many new results for important classes of networks, suggesting new theoretical research.

Benchmarks. Gossiping in the unit-cost model has been studied for numerous networks. However, (almost) matching lower and upper bounds have been found only for few classes of graphs underlying the network [12, 3]. For the linear-cost model even fewer results could be found in the literature. As our algorithm is close to optimal, we need very accurate estimates to evaluate its precise performance. We have done two things. In the first place, we have written an exponential-time exhaustive search. This program gives optimal gossiping schedules for graphs with up to 20 nodes and 30 edges. In the second place, we have studied linear-cost gossiping in detail for meshes and tori. The derived schedules are almost optimal, even for odd side lengths. As far as we know, these results are new.

graph	n	m	LW	UP	HR	time
$Mesh_{80\times 80}$	6400	12640	158	158	158	22800
$Hypercube_{13}$	8192	53248	13	13	13	8839
$Knödel_{13,8192}$	8192	53248	13	13	15	10571
$Butterfly_{10}$	10240	20480	16	25	24	11021
$DeBruijn_{13}$	8192	16381	18	41	25	9144
$Pancake_7$	5040	30240	13	17	16	2688
$Random_{10000,80000}$	10000	80000	14	??	17	35393

Table 1: Quality of the matching heuristic for graphs taken from various classes. From left to right the columns give the number of PUs, the number of connections, the lower bound, the best-known upper bound, the value computed by our algorithm and the time in seconds it took to compute the schedule. All results are given for the unit-cost model.

Results. We thus obtained a set of benchmarks containing small graphs, meshes, tori, complete graphs, hypercubes, Knödel graphs, cube-connected-cycles, shuffle-exchanges, butterflies, de Bruijn graphs, star and pancake graphs, and random graphs. A small selection of the results obtained with the matching heuristic are given in Table 1. The graph properties of these classes are so diverse, that we believe that if a heuristic performs so well for all of them, it will also perform well for graphs that cannot be analyzed theoretically. Generally, the number of rounds required by the matching heuristic appears to be away from the optimum by some slowly increasing number. For networks with n PUs and m connections, the worst-case running time of the matching heuristic is bounded by $\mathcal{O}(n^4 \cdot m)$. However, for all above mentioned classes of networks, the running time is bounded by $\mathcal{O}(R \cdot n \cdot \log n \cdot m)$, where R equals the number of rounds the gossiping schedule requires. In practice, on a normal workstation, a schedule for a graph with a few thousand nodes can be computed in less than one hour.

The coloring heuristic has been applied to cube-connected-cycles, butterflies, shuffle-exchanges, de Bruijn graphs, star and pancake graphs. It is much faster than the matching heuristic: even for graphs with thousands of nodes a solution is often found in less than one minute. For the cube-connected-cycles, butterflies, star and pancake graphs the coloring heuristic outperforms the matching heuristic using no more than g matchings, where g is the degree of the graph. Particularly nice results are obtained for butterflies (e.g., 22 rounds for $Butterfly_{10}$ with 10240 nodes) and pancake graphs (e.g., 15 rounds for $Pancake_7$ with 7! nodes). A great advantage of this approach is that the schedules can be represented concisely.

Discoveries. The newly obtained results show that some of the theoretical constructions are far from optimal. For gossiping on a shuffle-exchange and de Bruijn networks of order k, the current upper bounds are $4 \cdot k - 3$ and $3 \cdot k + 2$, respectively, [12]. Our algorithm suggests that the true values are $\lceil 2\frac{1}{2} \cdot k \rceil - 3$ and $2 \cdot k - 2$, respectively. For all cube-connected-cycles, butterfly and pancake graphs the constructed schedules improve the former ones [12, 3] by several rounds. Also for broadcasting we find many new results.

2 Preliminaries

We are studying interconnection networks with n PUs and m connections. The network will be identified with its underlying graph: PUs correspond to nodes and connections to edges. The PUs can send/receive packets to/from the PUs it is connected to, its *neighbors*.

Communication Problems. *Gossiping* can be described as follows: initially each PU holds a certain amount of private information; by communicating, the PUs should establish the situation in which all PUs know all information. The complete specification of the times each of the PUs is communicating with each of its neighbors is called a *gossiping schedule*. *Broadcasting* is the simpler problem in which initially only one PU holds a piece of information that must be made known to all other PUs. Both problems are subproblems of the more general problem in which the initial amount of data of PU_i has size h_i. For gossiping all the h_i are equal, for broadcasting all but one h_i are zero.

Communication Models. For the solution of the gossiping problem the communication model is of great importance. In the *all-port* model each PU can exchange information with all its neighbors at the same time. The other extreme is the *telegraph* model, in which a PU can be involved in only one communication operation: either receiving or sending, but not both. In the *telephone* model, a PU can communicate with only one of its neighbors at a time, but it can both send and receive during this communication. In this paper we assume the telephone model, though our heuristic might easily be extended to the telegraph model.

Cost Models. Next to the communication model, the cost model is of great importance. In the *unit-cost model* it is assumed that it takes one time unit to start-up communication with a neighbor, but that the actual data transfer takes negligible time. In this case, it is natural to assume that all communication is performed in discrete *rounds*, and for a given graph the goal is to determine a gossiping schedule that minimizes R, the required number of rounds. For large data sets or slow connections, this model may not be realistic. A two-parameter model gives a more accurate description of the actual communication behavior: transferring a packet of size s to a neighbor takes $1 + \tau \cdot s$ time. τ is the time it takes to transfer one packet divided by the start-up time. Under this *linear-cost model*, it is not always optimal to exchange the maximum amount of information. Our heuristic is taking care of this. It might also be profitable to have PUs work asynchronously. The heuristic can be modified to incorporate this possibility, but it would lead to a considerable increase in computation time. Therefore we continue to assume that the rounds are synchronized. In that case, the goal is to determine a gossiping schedule that minimizes

$$T = R + \tau \cdot \sum_{t<R} \max_{i<n}\{s_{i,t}\},$$

where $s_{i,t}$ denotes the number of packets PU_i is sending in round t. While R is called the number of *rounds*, $S = \sum_{t<R} \max_{i<n}\{s_{i,t}\}$ will be called the number of *steps*.

Graph Classes. We are considering graphs of several classes. Here we mention only some fundamental parameters. Definitions and more details can be found in [6, 12, 3, 15]. n and m denote the number of nodes and edges, LW and UP the current lower and upper bounds for R.

$Mesh_{a\times b}$**:** 2-dimensional $a \times b$ mesh. $n = a \cdot b$, $m = 2 \cdot a \cdot b - a - b$, $LW = a + b - 2$.

$Torus_{a\times b}$**:** 2-dimensional $a \times b$ torus. $n = a \cdot b$, $m = 2 \cdot a \cdot b$, $LW = (a + b)/2$.

$Hypercube_k$**:** k-dimensional hypercube. $n = 2^k$, $m = k/2 \cdot 2^k$, $LW = k$, $UP = k$.

$Knödel_{\Delta,k}$**:** Knödel graph. $n = k$, $m = k \cdot \Delta/2$, $LW = \lceil \log k \rceil$, $UP = \lceil \log k \rceil$, for $\Delta = \lfloor \log k \rfloor$.

CCC_k**:** k-dimensional cube-connected-cycles. $n = k \cdot 2^k$, $m = 3/2 \cdot k \cdot 2^k$, $LW = \lceil 5 \cdot k/2 \rceil - 2$, for $k \geq 5$, $UP = 5 \cdot \lceil k/2 \rceil$.

SE_k**:** k-th shuffle-exchange graph. $n = 2^k$, $m = 3/2 \cdot 2^k - 3$, for k even, and $m = 3/2 \cdot 2^k - 2$, for k odd, $LW = 2 \cdot k - 1$, $UP = 4 \cdot k - 3$.

$Butterfly_k$**:** k-th butterfly. $n = k \cdot 2^k$, $m = 2 \cdot k \cdot 2^k$, $LW = 1.741 \cdot k$, for $k \gg 1$, $UP = 5 \cdot \lceil k/2 \rceil$.

$DeBruijn_k$**:** k-th de Bruijn graph. $n = 2^k$, $m = 2 \cdot 2^k - 3$, $LW = 1.317 \cdot k$, for $k \gg 1$, $UP = 3 \cdot k + 2$.

$Star_k$**,** $Pancake_k$**:** k-th star or pancake graph. $n = k!$, $m = (k - 1) \cdot k!$, $UP = k + \sum_{i=3}^{k-1} \lceil \log i \rceil$, for $k \geq 3$.

$Random_{a,b}$**:** Random graph from $\mathcal{G}_{a,b}$. $n = a$, $m = b$, $LW = \lceil \log_2 a \rceil + \mathrm{odd}(a)$.

Here $odd(n) = n \bmod 2$. Bounds for gossiping in the linear-cost model are rare. Obviously, on a network with n PUs, every PU must receive $h \cdot (n - 1)$ packets. Thus, for any schedule, $S \geq h \cdot (n - 1)$. Because $R \geq \lceil \log n \rceil + odd(n)$ [13], the following trivial lower bound holds for all h, τ and any network:

$$T \geq \lceil \log n \rceil + odd(n) + \tau \cdot h \cdot (n - 1). \tag{1}$$

3 Gossiping on Meshes and Tori

In a d-dimensional mesh the PUs are laid out on a d-dimensional grid. Each PU is connected with its at most $2 \cdot d$ neighbors. A torus is a mesh with additional 'wrap-around' connections, connecting the PUs on the outsides with the PUs on the opposite outside. Meshes and tori are so simple, that almost optimal schedules can be derived for them even for the linear-cost model. In a different context gossiping on meshes has been studied in [11]. A path (one-dimensional mesh) with n PUs is denoted by P_n, a cycle (one-dimensional torus) by C_n, an $a \times b$ mesh by $M_{a,b}$ and an $a \times b$ torus by $T_{a,b}$. The PUs are indexed by their positions in the grid. The indices for every dimension start with 0.

Lemma 1 *For gossiping on paths and cycles of length n,*

$$
\begin{aligned}
T(P_n) &= n - 1 + \tau \cdot h \cdot (2 \cdot n - 3), \text{ for every even } n \geq 2, \\
T(P_n) &= n + \tau \cdot h \cdot (2 \cdot n - 3), \text{ for every odd } n \geq 5, \\
T(C_n) &= n/2 + \tau \cdot h \cdot (n - 1), \text{ for every even } n \geq 2, \quad (2) \\
T(C_n) &= \lfloor n/2 \rfloor + 2 + \tau \cdot h \cdot (n + 1), \text{ for every odd } n \geq 3. \quad (3)
\end{aligned}
$$

Proof: The first three schedules alternatingly use the edges $(2 \cdot i, 2 \cdot i + 1)$ and the edges $(2 \cdot i + 1, 2 \cdot i + 2)$. A node is always sending all packets that are unknown to the receiving node. The first round consists of one step, all other rounds are two steps long. Only for the case of gossiping on paths of odd length, the situation is slightly different: one extra round is required, and during the last two rounds the PUs send at most one packet. For cycles of odd length, node i remains idle during round i. This leaves a unique maximum matching: giving the active edges for round i. Whenever node 0 is communicating with node 1, it has a choice of 3 packets to send. It chooses the 2 that it knows longest. All rounds consist of 2 steps, except for the first and the last round, which take one step each. □

(3) shows that, on a cycle of even length, gossiping can be performed optimally: both the number of rounds and the number of steps are minimal. For the paths the number of steps is almost twice as large as the lower bound.

Lemma 2 *For gossiping on $a \times b$ meshes and tori,*

$$
\begin{aligned}
T(M_{a,b}) &\leq a + b - 1 + \tau \cdot h \cdot (a \cdot b + a - 1), \text{ for } a, b \geq 2 \text{ even}, \quad (4) \\
T(M_{a,b}) &\leq a + b - 1 + \tau \cdot h \cdot (a \cdot b + 3/2 \cdot a - 3), \text{ for } a \geq 2 \text{ even}, b \geq 3 \text{ odd}, \\
T(M_{a,b}) &\leq a + b + \tau \cdot h \cdot (2 \cdot a \cdot b - a - 3), \text{ for } a, b \geq 5 \text{ odd}, \quad (5) \\
T(T_{a,b}) &= a/2 + b/2 + \tau \cdot h \cdot (a \cdot b - 1), \text{ for } a, b \geq 2 \text{ even}, \quad (6) \\
T(T_{a,b}) &\leq \lfloor a/2 \rfloor + b/2 + 2 + \tau \cdot h \cdot (a \cdot b + 1), \text{ for } a \geq 3 \text{ odd}, b \geq 2 \text{ even}, \quad (7) \\
T(T_{a,b}) &\leq \lfloor a/2 \rfloor + \lfloor b/2 \rfloor + 4 + \tau \cdot h \cdot (a \cdot b + 2 \cdot a + 1), \text{ for } a, b \geq 3 \text{ odd}. \quad (8)
\end{aligned}
$$

Proof: All schedules consist of two phases, In phase 1 the gossiping is performed within the rows. In phase 2, gossiping is performed in the columns. The cost of these phases is estimated with Lemma 1. For tori the rows and columns constitute cycles, for meshes they are paths. For meshes, if a is even, then phase 2 is performed in pairs of adjacent columns that together constitute cycles of length $2 \cdot b$. If also b

is even, the same applies to phase 1. When a is even, either two or four (depending on the parity of b) PUs on each cycle hold the same information at the beginning of phase 2. Thus, for the analysis of phase 2, we may assume packets of size $h \cdot a/2$. If a and b are even, then the first round of phase 2 is omited. □

The result of (6) is optimal. The number of rounds is always optimal or close to optimal. Only for meshes with a and b odd the number of steps is a factor two too large. For 3×3 meshes we have an explicit construction with $R = 5$ and $S = 11$, which is optimal. In the following we describe an algorithm that gives a better trade-off between the numbers of rounds and steps for general odd a and b.

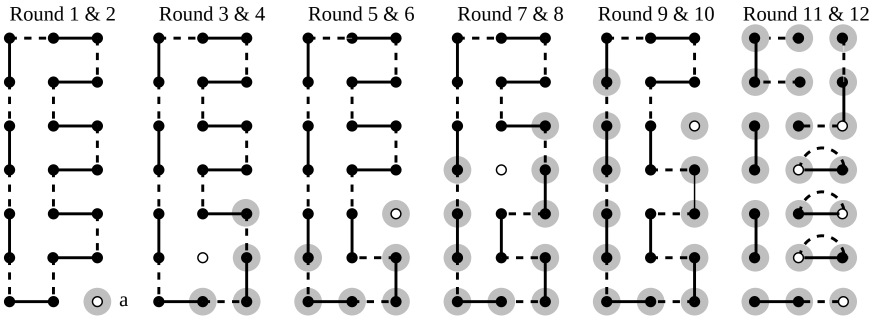

Figure 1: Gossiping schedule for a 3×7 mesh. The solid and dotted lines represent the active link in the first and second round in each figure, respectively. In the first 10 rounds we gossip along the changing cycle. Excluded nodes are marked with small circles. They will be short of at most 6 superpackets before the last 2 rounds. The nodes that know superpacket a are marked with grey discs. The first round uses one step, the last round uses 4 steps and all other rounds use 2 steps. Altogether we need 25 steps, 5 more than the minimum for $n = 20$.

Lemma 3 *For gossiping on a $(a \times b)$ mesh with a and b odd:*

$$T(M_{a,b}) \leq a + (3 \cdot b + 3)/2 + \tau \cdot h \cdot (\lceil a/3 \rceil \cdot (3 \cdot b + 4) + 2 \cdot a - 3).$$

Proof: The algorithm consists of two phases. In phase 1, the gossiping is performed within the rows. In phase 2, the gossiping is performed in vertical strips consisting of two columns each, except for one strip of width three.

We first consider gossiping on $3 \times b$ meshes, showing that generally

$$T(M_{3,b}) \leq (3 \cdot b + 3)/2 + \tau \cdot h \cdot (3 \cdot b + 4).$$

Because b is odd, the $3 \times b$ strip cannot be turned into a cycle of length $3 \cdot b$. If we would use a fixed cycle of length $3 \cdot b - 1$, then the excluded PU still would have to receive all packets at the end, requiring $h \cdot a \cdot b$ steps. Therefore, we use $\lceil (3 \cdot b - 1)/4 \rceil$ different cycles. During round 1 and 2, the excluded PU is located at position $(0, 2)$. Hereafter, it shifts up one row every other round, alternating between column 1 and 2. Because these changes are local, they do not disturb the dissemination for most of the packets. After $(3 \cdot b - 1)/2$ rounds, all but four packets are known by all nodes except by those that missed two gossiping rounds. The other four packets are known by at least $2 \cdot b - 2$ nodes and can be disseminated to all other nodes within two rounds and six steps. The construction is illustrated in Figure 1, details are given in [17].

In our case the packets are actually superpackets, consisting of $h \cdot a/3$ real packets each. This means that the number of steps has to be multiplied by $h \cdot a/3$. The gossiping on the $2 \times b$ strips can be performed faster, but it should be modified to also require about $3 \cdot b/2$ rounds. This can be easily achieved by sending $h \cdot a/3$ packets in the first round and $2 \cdot h \cdot a/3$ packets in the further rounds. □

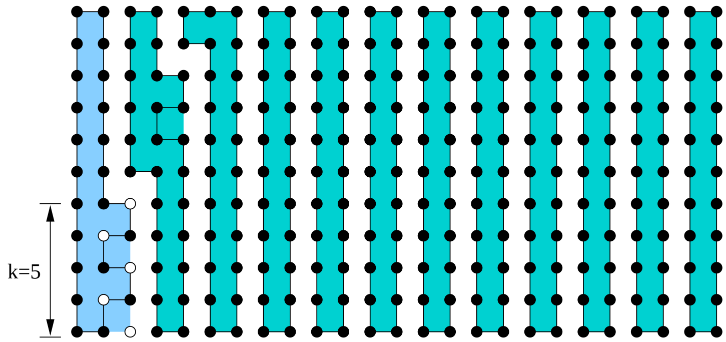

Figure 2: A 25×11 mesh divided into 12 strips. All strips contain an even number of nodes except for the leftmost strip with 27 nodes. The constructed cycles have length 26 at most and finish gossiping within 13 rounds. The nodes drawn as small circles are idle for 4, 2, 2, 2 and 3 rounds respectively.

This construction asymptotically minimizes the number of steps at the expense of $\lfloor b/2 \rfloor + 4$ additional rounds. Better trade-offs are possible: both the number of steps and the number of rounds can be made asymptotically optimal.

Lemma 4 *For gossiping on $a \times b$ meshes, a and b odd, the following result can be achieved for all $3 \leq k < b$:*

$$T(M_{a,b}) \leq a + b + \lceil k/2 \rceil + \tau \cdot h \cdot ((a+1) \cdot (b + b/k + k/2 + 7/2). - 13/2)$$

Proof: We use vertical strips of width 2 for most of their height and width 3 for some consecutive rows. The leftmost strip contains $2 \cdot b + k$ nodes with $3 \leq k < b$ odd. All other strips are smaller with an even number of nodes. In each strip we gossip on cycles of even length for $(2 \cdot b + k - 1)/2 = R'$ rounds. The routing in the leftmost strip is most critical. As in the proof of Lemma 3, the gossiping in the other strips can be tuned so that it has no impact on the duration of the rounds.

Each node in the leftmost strip is starting with a superpacket of size $h \cdot \lceil a/2 \rceil$. The cycle is changing, using k different idle nodes v_i, $0 \leq i < k$. The v_i are idle for l_i successive rounds, v_0 first, then v_1, and so on. The idea is illustrated in Figure 2. The l_i are chosen so, that $\sum_i l_i = R'$, $\lfloor R'/k \rfloor - 1 \leq l_i \leq \lceil R'/k \rceil + 1$ for all k and l_i even for $0 \leq i < k-1$. If node v_i becomes idle in round r, then the two superpackets it received in round $r-1$ from its neighbor v are resent by v to v_{i-1} in round $r+1$. This causes a delay of two rounds for all packets passing v in this direction. Because all l_i are even for $i < k-1$, only packets traveling in a counterclockwise sense are concerned. Thus, only nodes in column 1, and none of the v_i, will be short of some packets due to this delay. They can be informed by adjacent nodes from column 0 in one additional round which is also used to supply the v_i with the at most $2 \cdot l_i$ superpackets they have missed. Phase 1 takes $a + \tau \cdot h \cdot (2 \cdot a - 3)$ time. The first R' rounds of phase 2 require $\lceil a/2 \rceil \cdot (2 \cdot b + k - 2)$ steps, the additional round $\lceil a/2 \rceil \cdot 2 \cdot \max_i\{l_i\} \leq (a+1) \cdot (b/k + 5/2)$ steps. □

For $k = \sqrt{b}$, the $\sqrt{b/2}$ additional rounds as well as the $a \cdot (3/2 \cdot \sqrt{b} + 7/2)$ additional steps are lower-order terms. The results can be immediately generalized to higher dimensional meshes and tori:

Lemma 5 *For gossiping on d-dimensional $a_1 \times \cdots \times a_d$ meshes and tori,*

$$\begin{aligned} T(M_{a_1,\ldots,a_d}) &\leq \sum_{i=1}^{d} a_i - d + 1 + \tau \cdot h \cdot (\prod_{i=1}^{d} a_i + a_1 - 1), \ \textit{all } a_i \geq 2, \ \textit{all } a_i \ \textit{even}, \\ T(T_{a_1,\ldots,a_d}) &\leq \sum_{i=1}^{d} a_i/2 + \tau \cdot h \cdot (\prod_{i=1}^{d} a_i - 1), \ \textit{all } a_i \geq 2, \ \textit{all } a_i \ \textit{even}. \end{aligned}$$

Proof: For d-dimensional meshes, phase j is performed in cycles of length $2 \cdot a_i$. For $j > 1$, the first round can be omited. At the beginning of phase j, the superpackets are of size $h \cdot \prod_{i=1}^{j-1} a_i/2$. Thus, phase j requires $(2 \cdot a_j - 2) \cdot h \cdot \prod_{i=1}^{j-1} a_i/2 = h \cdot \prod_{i=1}^{j-1} a_i \cdot (a_j - 1)$ steps. Summing over all phases gives the result. The analysis for tori is similar. □

4 The Matching Heuristic: Description and Analysis

4.1 Description

Given a undirected graph representing the underlying network, the heuristic computes a gossiping schedule, which for each round specifies the active edges and the routed packets. For each round, based on the current data distribution in the network, the heuristic first determines the edges that are going to be used. Then, for the linear-cost model, it selects the packets that are going to be transferred. Finally, the data distribution as it arises after routing the selected packets is determined. Such rounds are repeated until the gossiping has been completed.

In the considered telephone model, a node can exchange data with only one neighbor per round. Thus, for every round, the set of active edges must form a matching of the graph. Actually, the heuristic constructs a maximum-weight matching for a graph whose edges are weighted as a function of the packet distribution in the network: the more useful it appears to use an edge, the higher its weight. In the unit-cost model, there is no limit on the number of packets that can be exchanged during a round between two communicating PUs. On the other hand, in the linear-cost model, for each round t its number of steps s_t has to be fixed. Choosing s_t equal to the maximum number of packets any PU wants to transfer to a matched neighbor might be inefficient, because many other PUs may run out of packets in fewer than s_t steps. Choosing s_t too small is inefficient, because then the start-up costs are not amortized optimally. Thus, s_t must be chosen as a trade-off between extra start-up costs and wasted transfer capacity. Once s_t has been fixed, we have to decide for each active edge which packets to transfer. For this purpose each packet is assigned a priority and the at most s_t edges with highest priority are transferred. The operations performed in each round can be summarized as follows:

Algorithm ROUND_HEURISTIC

1. Compute the weights for all edges.

2. Construct a maximum weighted matching. Matched edges are active in this round.

3. In the linear-cost model: Fix the number of steps for this round.

4. In the linear-cost model: For each active edge, choose the set of packets to be transferred.

5. Calculate the packet distribution as it arises after transferring all selected packets.

The crux of the heuristic lies in step 1: how to set the edge weights? We use two different methods.

Potential Approach. The weight of an edge (v, w) is set equal to its *potential*, defined as the number of packets known by either v or w, but not by both of them.

Lemma 6 *Using the potential approach, calculating the edge weights of a graph with n nodes and m edges takes $\mathcal{O}(n \cdot m / \log n)$ time and $\mathcal{O}(n^2 / \log n)$ space.*

Proof: An array of n bits is used to store for each node the packets it already knows. For n nodes we need $\mathcal{O}(n^2)$ bits storage. For each edge, the exclusive-or of two arrays of n bits can easily be computed in $\mathcal{O}(n)$ time. Using that $\log n$ bits fit in one word, and that `exor` can be performed on words, the time and storage bound can be reduced by a factor $\log n$. `number_of_ones` is not a standard operation, but it can be determined in constant time by precomputing a table of size n and table-lookup. □

BFS Approach. The potential approach is simple, requires little storage and is very fast, but as a pure local, greedy approach it lacks a global view. The Breadth-First-Search (BFS) approach, though far more expensive, is much better in that respect.

Definition 1 *The* dispersion region *$DR(p, t)$ of a packet p is the set of nodes that know p at the beginning of round t. For a node v, $dist_v(p, t)$ denotes the shortest distance in the graph from v to a node $w \in DR(p, t)$. The set of* border-crossing edges *$bce(p, t)$ is defined as $bce(p, t) = \{(v, w) \in E | \ v \in DR(p, t) \text{ and } w \notin DR(p, t)\}$. For a node $v \notin DR(p, t)$, $bce_v(p, t)$ consists of all edges in $bce(p, t)$ that lie on a shortest path from $DR(p, t)$ to v.*

Obviously, the subgraphs induced by the dispersion regions are connected. The above notions are illustrated in Figure 3.

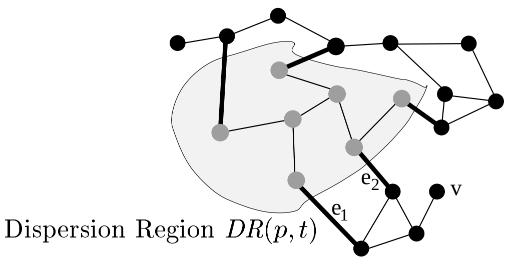

Figure 3: The dispersion region $DR(p, t)$ for some packet p. The edges of $bce(p, t)$ are drawn bold. $dist_v(p, t) = 3$ and $bce_v(p, t) = \{e_1, e_2\}$.

The weight attributed to an edge is given as the sum of the contributions by each of the data packets p. Only border-crossing edges can disseminate p further and will be provided with weight. Consider an edge $e \in bce(p,t)$. How useful is e for the rapid dissemination of p? Packet p should preferably be routed on shortest paths from $DR(p,t)$ to all other nodes: if, for a node v, an edge $e \in bce_v(p,t)$ is chosen to be active in round t, then $dist_v(p,t+1) = dist_v(p,t) - 1$. If e lies on many of these shortest paths it is more useful. The larger $dist_v(p,t)$ is, the more priority should be given to forwarding p towards v. These criteria motivate the following choice of the weight, involving parameters *Dist_Exp* and *Num_Exp*, that is attributed by all nodes $v \notin DR(p,t)$ to every edge $e \in bce_v(p,t)$:

$$weight(v,p,t) = \frac{dist_v(p,t)^{Dist\_Exp}}{|bce_v(p,t)|^{Num\_Exp}}, \tag{9}$$

In round t, for all data packets p, we have to compute $dist_v(p,t)$ and $bce_v(p,t)$ for all nodes v. We use a modified breadth first search algorithm, so nodes are considered in order of increasing $dist_v(p,t)$. The edges in $bce_v(p,t)$ are maintained in sorted lists and computed as follows. For all nodes $v \in DR(p,t)$ the set $bce_v(p,t)$ is empty. For nodes v with $dist_v(p,t) = 1$, $bce_v(p,t)$ consists of all incident edges that connect v to a node in $DR(p,t)$. For larger $dist_v(p,t)$ the algorithm computes the union of the sets $bce_{w_i}(p,t)$, for all nodes w_i adjacent to v with $dist_{w_i}(p,t) = dist_v(p,t) - 1$. If the number of these w_i equals j and $\sum_i |bce_{w_i}(p,t)| = l$, then this union can be computed in $\mathcal{O}(l \cdot \min\{\log j, \log(m \cdot j/l) + 1\})$. Thus, the calculation of the $bce_v(p,t)$ can easily be incorporated into the BFS search.

Lemma 7 *Computing the edge weights for a graph with n nodes and m edges using the BFS approach without considering the time to maintain the sets of border-crossing edges $bce_v(p,t)$ takes $\mathcal{O}(n\cdot(n+m))$ time and $\mathcal{O}(n^2/\log n)$ space. Computing the $bce_v(p,t)$ takes $\mathcal{O}(n^3 \cdot m)$ time and $\mathcal{O}(n \cdot m)$ space.*

Proof: The modified BFS algorithm is called for all n packets. Without maintaining the $bce_v(p,t)$ the time for one call is $\mathcal{O}(n+m)$. Each of the n dispersion region can be maintained with n bits. For a node v, $bce_v(p,t)$ is the union of at most n sets with at most m elements each. This computation takes $\mathcal{O}(n \cdot m)$ time. The $bce_v(p,t)$ are computed for all p and v, giving a running time of $\mathcal{O}(n^3 \cdot m)$. At any given time, at most n sets $bce_v(p,t)$ are stored, each of maximal size m. Working with bit arrays, a factor $\log n$ is saved again for time and storage. □

Because the gossiping takes at most n rounds, the total time-consumption is bounded by $\mathcal{O}(n^4 \cdot m)$. Figure 4 gives a graph with $m = \Theta(n^2)$, for which the computation of the contributions to the weights caused by a single node p takes $\Omega(n^4)$ time. So, the estimate in Lemma 7 might actually be sharp.

Linear-Cost Model. In step 2 of ROUND_HEURISTIC, a maximum weighted matching $\mathcal{M}$ is constructed that determines the active links. Thereupon, in the unit-cost model, a PU sends all packets that are new to the receiver. In the linear-cost model, the packets that are going to be routed along the active links are determined in step 3 and 4. We now describe how this is done.

Let $\mathcal{P}(v)$ denote the set of packets known by a node v, and let *Transfer_Volume*$(s, \mathcal{M})$ be the number of packets that can be sent in s steps along all edges in $\mathcal{M}$:

$$Transfer\_Volume(s, \mathcal{M}) = \sum_{(v,w)\in\mathcal{M}} \min\{s, |\mathcal{P}(v) \backslash \mathcal{P}(w)|\} + \min\{s, |\mathcal{P}(w) \backslash \mathcal{P}(v)|\}.$$

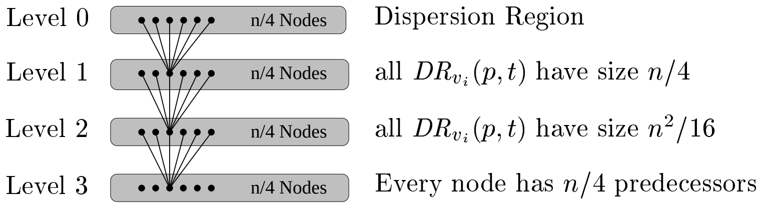

Figure 4: Example for a set of border-crossing edges of maximal size. Each of the levels consists of $n/4$ nodes. In each level, all nodes have the same distance to the dispersion region. Edges exist between each pair of nodes in adjacent levels. There are $n^2/16$ border-crossing edges altogether. For each node in level 3 the set of border-crossing edges is computed from $n/4$ sets each of size $n^2/16$. This takes $\Omega(n^4)$ time. When these costs apply for the dispersion regions of all packets, then $\Omega(n^5)$ time is needed.

We want to maximize the number of transferred packets per cost unit. Let s_{opt} be the value of s, $1 \leq s < n$ for which the expression $\mathit{Transfer\_Volume}(s, \mathcal{M})/(1+\tau \cdot h \cdot s)$ is maximized. This value s_{opt} can be computed in $\mathcal{O}(n)$ time. We limit the round to s_{opt} steps. s_{opt} depends on τ, the ratio of transfer costs to start-up costs: larger start-up costs result in longer rounds and vice versa. s_{opt} is the best choice for the current round, but does not guarantee optimality on a longer time scale. In particular the choice of s_{opt} is not optimal, if the gossiping can be finished in the current round. The heuristic tests for this possibility.

Now we have to choose the packets that are going to be transferred. This is done by assigning weights to the packets and then picking for each PU the at most s_{opt} packets with the highest weights larger than zero. For a node v with $e = (v, w) \in \mathcal{M}$, a data packet p it is holding is given the weight that is assigned to e during the BFS search for p. If the edge weights are stored for each of the data packets, then these weights can be determined without additional work. However, this may require $\Omega(n \cdot m)$ storage. It is better to compute the packet weights only after the active edges have been selected. In this way, less than n weights must be stored for each of the $n/2$ edges in $\mathcal{M}$.

4.2 Refinements and Extensions

Look-Ahead. For tori and hypercubes with an equal packet distribution, the heuristic computes equal weights for all edges, independently of the parameters used. In this case the weighted matching in step 2 of ROUND_HEURISTIC is not more than a maximum cardinality matching. For hypercubes many different maximal cardinality matchings exist, and only a few lead to an optimal gossiping schedule. If we assume that in case of equal weights, the constructed matching depends on the order in which edges are stored in memory, the heuristic cannot find an optimal schedule for some inputs without look-ahead. For some graphs the quality of the calculated schedules indeed depends on the input order of the edges. The optimal results in Table 1 for hypercubes and tori are found for the most natural ordering of the edges, but if they are permuted before applying the heuristic, a few extra rounds are required.

The above considerations show that a round-by-round optimization cannot always lead to optimal results. A more refined approach considers several matchings for a round, computes the resulting distribution of packets l rounds later, compares them and then chooses the most promising matching. We use two methods for generating a set of matchings. In step 2, ROUND_HEURISTIC constructs a maximum weighted matching $\mathcal{M}_{\text{opt}}$. To obtain a suboptimal matching we may randomly choose a small number of edges from $\mathcal{M}_{\text{opt}}$, temporarily set their weights to 0 and compute a new weighted matching. Another method uses different parameters for (9) which leads to different edge weights. Unfortunately, there is no guarantee that also the resulting matchings are different, and the cost for recomputing the edge weights is high. Starting with several possible matchings $\mathcal{M}_1, \ldots, \mathcal{M}_j$, we obtain packet distributions $\mathcal{D}_1, \ldots \mathcal{D}_j$ after l rounds. We should select the matching $\mathcal{M}_i$ that leads to the packet distribution $\mathcal{D}_i$ for which the gossiping can be finished fastest. For this selection, we should define a function that attributes some measure of *cost* to packet distributions. This function should be relatively simple, because its evaluation determines the overall running time. Accuracy is not so important, because it constitutes a secondary heuristic. For a packet distribution $\mathcal{D}$, $dist_v(p, \mathcal{D})$ denotes the distance in the graph from the node v to the dispersion region of the data packet p under $\mathcal{D}$. For a parameter $Dist\_Exp'$, that may be different from $Dist\_Exp$ in (9), we define the following function, that can be evaluated in $\mathcal{O}(n \cdot (n+m))$ time:

$$cost(\mathcal{D}) = \sum_{p<n} \sum_{v<n} dist_v(p, \mathcal{D})^{Dist\_Exp'}.$$

Approximate Matching. Since constructing the maximum weighted matching in step 2 consumes up to 60% of the running time, we are interested in approximation algorithms with a smaller time complexity. [16] introduced a $\mathcal{O}(n+m)$ algorithm that computes a matching with weight at least 1/2 of the optimum. We use the simpler $\mathcal{O}(m \cdot \log n)$ algorithm from [1].

Broadcasting. The heuristic is also suitable for computing broadcasting schedules. The algorithm is the same but now the distribution of only one data packet determines the edge weights. With the potential approach, all edge weights are set to 0 or 1. With the BFS approach optimal results can achieved for many graph classes. Since the edge weights are computed n times faster, the computation of the maximum weighted matching dominates the running time. Fortunately, even the matching is much easier, since in many cases there are only few edges with non-zero weights, particularly during the first rounds. As also the storage requirements are much smaller than for gossiping, broadcasting schedules can be computed for graphs with up to one million nodes.

5 The Matching Heuristic: Practical Behavior

5.1 Running Time

In order to analyze the running time, we have tested graphs with up to 16384 nodes from numerous classes of graphs (in total we performed 93 measurements, at least seven for every class, except for pancake and star graphs). We focus on the unit-cost model: for the linear-cost model, the heuristic takes at most twice as long. The total time consumption T_{total} has two main contributions: the time T_M for constructing the maximum weighted matchings; and the time T_H for all the rest. T_M varies considerably, but the matching can be viewed as an external routine. Therefore, it is not unreasonable to focus on T_H. Inspired by theoretical considerations, we have tested several functions that might describe T_H as a function of n, m and the number of required rounds R. Somewhat surprisingly, for all classes of graphs, T_H can be approximated to within a few percent by a single function of just two parameters:

$$T_{\text{app}}(n, m, R) = \alpha \cdot R \cdot n \cdot m \cdot \log(n) + \beta \cdot R \cdot n^2. \quad (10)$$

The values of α and β depend on the class of graphs. Table 2 gives the optimal values for the considered classes of graphs. The first term is probably due to the cost for maintaining the sets of border-crossing edges, the second term gives the cost for handling all nodes. In some sense the values of α and β give a measure for the hardness of the graph classes, their ratio tells which of the two terms dominates. For meshes, for example, it can be understood that the parameters are small (many easy steps) and that the second term dominates (few border-crossing edges).

graph class	$\alpha \cdot 10^{-10}$	$\beta \cdot 10^{-8}$	$D_{\text{avg}} \cdot 100$	$D_{\text{max}} \cdot 100$	T_M/T_{total}
Mesh $n \times n$	122	226	2.04	4.56	30% – 47%
Torus $n \times n$	513	235	0.54	1.31	32% – 50%
Hypercube	939	254	1.88	3.33	17% – 21%
CCC	1092	188	4.35	6.21	19% – 31%
Shuffle-Exch.	1371	123	3.80	8.47	28% – 43%
Butterfly	1188	174	2.23	3.63	17% – 27%
De Bruijn	1370	136	2.86	6.25	33% – 45%
Star	494	197	0.09	0.12	17% – 22%
Pancake	586	211	1.02	1.56	14% – 18%
Random	669	233	4.07	13.76	34% – 82%

Table 2: Parameters and properties for different graph classes. α and β are the optimized choices for the parameters in (10). D_{avg} and D_{max} give the average and maximum values of $|T_{\text{app}}(n, m, R) - T_H|/T_H$. The last column states how much of the total running time is used for constructing the matching.

5.2 Quality of Computed Schedules

The quality of the heuristic heavily depends on the choice of the parameters. Particularly important is *Dist_Exp* from (9) which determines the influence of the distance between nodes and dispersion regions. We used values in the range from 0.25 to 60. The optimal value depends on the the graph class, the size of the graph and the cost model. For larger graphs larger values of *Dist_Exp* tend to give better results. For the linear-cost model, values between 0.5 and 2.5 are suitable. Better results are achieved when *Dist_Exp* decreases from round to round. When using approximate matching in step 2, then the optimum of *Dist_Exp* is usually higher than for exact matching. For meshes, the best choice is *Dist_Exp* $= 4$. For butterflies, the best choice is *Dist_Exp* $= 2$.

For the unit-cost model, a selection of the results is given in Table 1. For cube-connected-cycles, shuffle-exchanges, butterflies and de Bruijn graphs further results are presented in Table 5. For meshes, tori and hypercubes the computed schedules are optimal, though we should remark that for hypercubes a few extra rounds are required if the nodes are indexed differently. Generally, for all cases in which the lower bound is sharp, our heuristic comes rather close to it. Studying the developments for the graph classes in Table 5 gives the impression that with increasing R

the heuristic looses a round every now and then. The series are not long enough to quantify this statement.

			$\tau = 2.0$		$\tau = 0.5$		$\tau = 0.1$		$\tau = 0$	
graph class	n	m	R	S	R	S	R	S	R	S
$Mesh_{20\times 20}$	400	360	62	497	49	517	40	612	38	2713
$Torus_{21\times 21}$	441	882	34	488	30	486	28	528	23	1023
CCC_7	896	1344	24	902	22	904	23	943	20	1139
SE_{10}	1024	1533	63	2047	50	2051	37	2073	23	3933
$Butterfly_7$	896	1792	39	1044	33	1107	22	1110	17	1229
$DeBruijn_{10}$	1024	2045	46	1221	39	1270	31	1513	18	2733
$Random_{1000,8000}$	1000	8000	19	1009	18	1014	16	1028	13	1281

Table 3: R and S values achieved by the matching heuristic for various τ values in the linear-cost model for graphs taken from various classes.

For the linear-cost model we found the results in Table 3. These are typical examples, not the best we could find. The adaptiveness of the heuristic is exposed clearly: with decreasing τ the number of steps becomes less important and gradually increases. At the same time the number of rounds decreases. Comparing the results for $\tau = 0$ and $\tau = 0.1$, shows that often S can be reduced considerably without increasing R by much. Apparently, computing schedules for the linear-cost model is much harder than for the unit-cost model: the deviations from the optimum values (as far as known) are considerable. For example, for a 20×20 mesh, the heuristic finds schedules with $T(\tau = 2) = 1056$, $T(\tau = 0.5) = 308$, $T(\tau = 0.2) = 153$, $T(\tau = 0.1) = 102$, respectively. All of these are about 25% more than required by the schedule underlying (4), which has $R = 39$ and $S = 419$ for all τ. All results were computed with the BFS approach. For Knödel, Star and Pancake graphs, S is optimal for all k independently of τ. For cube-connected-cycles and butterflies S is optimal for all even k.

5.3 Refinements and Extensions

Look-Ahead. Look-ahead is most useful for small graphs. Probably, the reason is that, even though the number of matchings that lead to an optimal gossiping grows with increasing graph sizes, the ratio to all possible matchings is rapidly decreasing. So it becomes much harder to find them by randomly testing suboptimal matchings. One example for an improvement using look-ahead is the 5×5 mesh. This is the only instance of meshes for which the simple heuristic was not able to find an optimal schedule. For random graphs with 30 nodes and 60 edges, look-ahead improves the result for about 40% of the graphs. We tested a maximum of ten different matchings per round, computed another two rounds look-ahead using only the optimal matching and compared the resulting distributions of packets. Sometimes the result with look-ahead is worse than without. This is due to the fact, that even when, $cost(\mathcal{D}_1) < cost(\mathcal{D}_2)$, for two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$, it may nevertheless take longer to complete $\mathcal{D}_1$ than $\mathcal{D}_2$.

Approximate Matching. Using approximative methods, the time for constructing the weighted matching decreases from about 30% to less than 1% of the overall running time. The quality of the schedules depends on the graph class. The larger

k	CCC_k		SE_k		$Butterfly_k$			$DeBruijn_k$		
	Opt	HR	Opt	HR	LB	UB	HR	LB	UB	HR
3	6	*6*	5	*5*	5	5	*5*	4	6	**4**
4	9	*9*	7	*7*	7	7	*7*	6	8	**5**
5	11	*11*	9	*9*	8	9	*9*	7	9	**7**
6	13	*13*	11	*11*	10	11	**10**	8	11	**8**
7	16	*16*	13	*13*	11	13	**12**	10	12	**9**
8	18	*18*	15	*15*	13	15	**14**	11	14	**11**
9	21	*21*	17	*17*	15	17	**16**	12	15	**12**
10	23	*23*	19	*19*	16	19	**17**	13	17	**14**
11	26	*26*	21	*21*	18	21	**19**	15	18	**15**
12	28	*28*	23	24	19	23	**22**	16	20	**17**
13	31	*31*	25	26	21	25	**23**	18	21	**18**
14	33	*33*	27	28	23	27	**24**	19	23	**20**

Table 4: Broadcasting in four different graph classes. Given is the best result of the heuristic together with the current lower and upper bound, or the optimum value, if they are identical. For SE_k and $DeBruijn_k$, the results hold for broadcasting from node 0 and several other nodes. For CCC_k and $Butterfly_k$ the results hold for any source node. Italic printing indicates that the number matches the best previously obtained value, bold printing indicates that the number improves the best previous value. The "lower bounds" are not really lower bounds: they are computed with the formulas in [12], which only hold asymptotically. Thus it may happen that for $DeBruijn_7$ the heuristic requires fewer rounds than given by the lower bound.

the graphs, the bigger the loss of quality. For meshes of almost all sizes the calculated schedules are still optimal or at most one round away from it. For shuffle-exchange graphs we loose at most one round, for de Bruijn graphs one or two. Generally, the differences are not big, but if performance is important, then exact matching is to be preferred. Approximate matching is particularly interesting, when the weighted graph is constructed with the much faster potential approach. Approximate matching is even more suited for broadcasting: the edge weights are computed in $\mathcal{O}(m+n)$ per round, much less than the time for exact matching. Furthermore, for broadcasting, the quality of the computed schedules is almost the same.

Broadcasting. Broadcasting schedules cannot only be computed much faster than for gossiping, the structure of the problem is also so much simpler, that their lengths are much closer to the optimum. Some results are given in Table 4. For cube-connected-cycles and for shuffle-exchange graphs, optimal schedules are known, so we could not hope to improve them. Nevertheless, it is very positive that our heuristic, in a matter of minutes!, finds almost optimal results. For butterflies and de Bruijn graphs, our heuristic improves the former constructions by a few rounds.

6 The Coloring Heuristic

The coloring heuristic is an alternative general gossiping heuristic. Initially, the computer or the user constructs a set $\mathcal{S}$ of p matchings $\mathcal{M}_i$, $0 \leq i < p$, that appear suitable for gossiping. Then the program tests for sequences $(\mathcal{M}_{i_0}, \mathcal{M}_{i_1}, \ldots, \mathcal{M}_{i_{R-1}})$ whether this is a gossiping schedule, until a solution has been found. By making the right choice of $\mathcal{S}$, by pruning most of the sequences and by enumerating the remaining ones in a non-trivial order, this simple idea can be turned into an approach that beats the matching heuristic in speed and performance for several classes of graphs.

The program essentially consists of R nested loops, implemented recursively. At the top level, we start with one packet in every node. The operations in the loop at level j, $0 \leq j <= R-2$, can be summarized as follows:

1. Consider all $\mathcal{M}_i$, $0 \leq i < p$, and filter out those that appear useless.
2. Sort the surviving matchings according to their estimated usefulness.
3. Apply the highest ranked matching that has not been tried before to the current data set and proceed to level $j+1$.

At level $R-1$ the resulting data set is tested for completeness.

Choosing the Matchings. A good idea is to perform a minimum edge coloring of the graph (whence the name of the heuristic) and then completing the sets of edges with the same color to a maximal cardinality matching. Sometimes fewer matchings will do, sometimes one should better add some more, but this approach gives the smallest number of matchings that together contain all edges at least once. Clearly this approach is most suited for regular graphs of degree g that allow a coloring with g perfect matchings. Examples are cube-connected-cycles, butterflies, star and pancake graphs.

Pruning Sequences. If, for given $\mathcal{S}$ and R, solutions exist at all, then typically there are many of them. The goal is to minimize the time for finding one. So, we should focus on parts of the search space where solutions lie most densely, pruning out less promising sequences, even if we may miss some solutions by this. An elementary observation is that we should have $\mathcal{M}_{i_j} \neq \mathcal{M}_{i_{j-1}}$, for all $1 \leq j < R$. This reduces the number of sequences from p^R to $p \cdot (p-1)^{R-1}$. For most classes of graphs it was effective to also impose $\mathcal{M}_{i_j} \neq \mathcal{M}_{i_{j-2}}$, for all $2 \leq j < R$. This reduces the number of sequences to $p \cdot (p-1) \cdot (p-2)^{R-2}$. Adding the condition that each matching occurs at least once in every subsequence of $p+1$ matchings reduces the number of sequences even much stronger. Another filtering technique is the following: if W packets are transferred when using the best matching, then we consider only matchings for which at least $f \cdot W$ packets are transferred, for some $0 < f < 1$. If it is applied with the right choice of f, this technique is the most effective of all.

Focusing. In the current implementation the usefulness of a matching is estimated by the number of packets that it allows to transfer (as in the potential approach). For the matching heuristic there is a trade-off between time and quality. Here a more elaborate ranking of the matchings is useful only if this leads to solutions faster. We doubt that this is the case, because the precise order of the matchings at the earlier stages of the recursion is not so important, while the latter stages can be rapidly tested exhaustively.

Broadcasting. The coloring heuristic has also been implemented for broadcasting. It is remarkably fast, but it cannot compete with the matching heuristic in quality: the idea of working with a fixed set of matchings is limited to gossiping.

k	CCC_k			SE_k			$Butterfly_k$			$DeBruijn_k$		
	LB	UB	HR	LB	UB	HR	LB	UB	HR	LB	UB	HR
3	7	10	**7**	5	9	**5**	**6**	10	**6**	4	11	**4**
4	9	10	**9**	7	13	**7**	7	10	**7**	6	14	**6**
5	11	15	**13**	9	17	**10**	8	15	**11**	7	17	**8**
6	13	15	**14**	11	21	**12**	10	15	**12**	8	20	**10**
7	16	20	**19**	13	25	**15**	11	20	**16**	10	23	**12**
8	18	20	**18**	15	29	**17**	13	20	**17**	11	26	**14**
9	21	25	**23**	17	33	**20**	15	25	**21**	12	29	**16**
10	23	25	**23**	19	37	**23**	16	25	**22**	13	32	**18**
11	26	30	**29**	21	41	**26**	18	30	**26**	15	35	**20**
12	28	30	**28**	23	45	**28**	19	30	**27**	16	38	**23**
13	31	35		25	49	**31**	21	35		18	41	**25**
14	33	35		27	53	**35**	23	35		19	44	**28**

Table 5: Gossiping in four different graph classes. Given is the best heuristic result together with lower and former upper bound. The new lower bound for BF_3 was found with the exact algorithm: trying all possible sequences of five maximal matchings, shows that none of them can be used for gossiping.

7 Discoveries and Hypotheses

The heuristics were applied to regular graphs with known gossiping schedules for testing their performance. Then it turned out that in many cases the schedules they find are better than the best schedules in the literature. All results that could be computed in a reasonable amount of time are given in Table 5 and Table 6.

For CCC_k, [12] gives an upper bound of $5 \cdot \lceil k/2 \rceil$, our coloring heuristic achieves better for all k. It appears that for even k the lower bound can be matched. The results for odd k are less good. It may even be the case that R does not increase monotonously with k. For SE_k, [12] gives an upper bound of $4 \cdot k - 3$. Our matching heuristic achieves much better. The results suggest that going from k to $k + 1$ increases the number of rounds by 3 if k is even and by 2 if k is odd. This would give an upper bound of $\lceil 5/2 \cdot k \rceil - 3$. For $Butterfly_k$, [12] gives an upper bound of $5 \cdot \lceil k/2 \rceil$. Our coloring heuristic achieves somewhat better. The results suggest that going from k to $k + 1$ increases the number of rounds by 4 if k is even and by 1 if k is odd. This would give an upper bound of $3 \cdot \lceil k/2 \rceil + k - 3$.

For $DeBruijn_k$, [12] gives an upper bound of $3 \cdot k + 2$. Our matching heuristic achieves much better. The results suggest that going from k to $k + 1$ increases the number of rounds by 2. This would give an upper bound of $2 \cdot k - 2$. Our results for $Star_k$ and $Pancake_k$ are better than those in [3] (for $k = 3, 4, 5, 6, 7, 8$, the best construction in [3] gives $R = 3, 6, 9, 13, 17, 21$, respectively), but it is hard to draw conclusions from this except for the fact that apparently pancakes are better suited for gossiping than star graphs. This means that there cannot be a single general optimal Caley graph gossiping strategy. In Table 4, for broadcasting, the differences of the heuristic results with the lower bounds are so small, that they appear to be sharp.

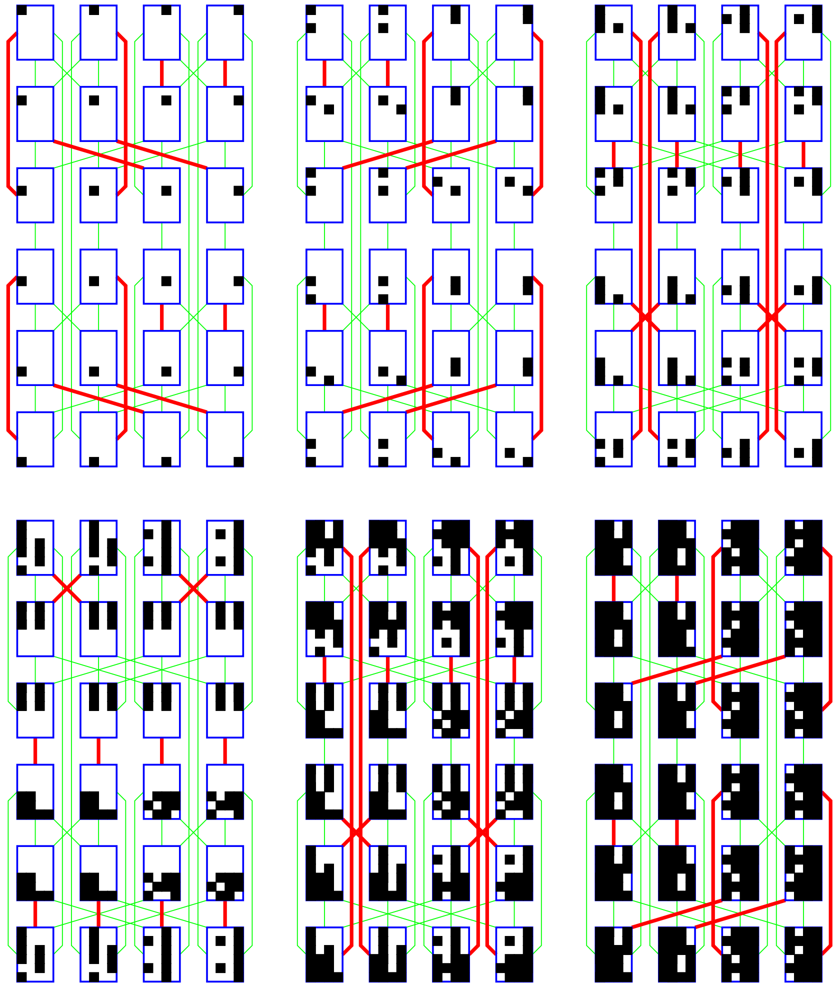

Figure 5: Gossiping on *Butterfly*$_3$ with $n = 24$ and $m = 48$. Indicated is the situation at the beginning of each of the six rounds. In each picture, the matched edges are drawn bold. The packet originating from some node v is placed at the position within the rectangles that corresponds to the position of node v in the network. The rounds take $1+2+4+8+6+4 = 25$ steps, two more than the trivial minimum.

8 Schedules and Examples

The matching heuristic constructs explicit schedules. These might be stored and used for gossiping. However, only for small networks these are suited for human interpretation. Figure 5 gives a schedule for *Butterfly*$_3$ with $R = 6$ and $S = 25$ as it was computed by the matching heuristic. This graph is already of a complexity that excludes any construction by hand: one has to resort to general approaches that are far from optimal.

The matching heuristic has taught us that gossiping can be performed efficiently by a sequence of matchings picked from a small set of matchings that are mutually almost edge-disjoint. We have formalized this observation in the coloring heuristic which specifically searches for such schedules. For CCCs, butterflies, star and pancake graphs this leads to schedules that are as good or better as those found with the matching heuristic. Before going in detail we consider *Pancake*$_4$ (see [3] for a

k	CCC_k		$Butterfly_k$	
	R	Schedule	R	Schedule
3	7	0120120	6	012320
4	9	012021202	7	0123023
5	13	2020120120210	11	02103231023
6	14	01202120212020	12	012030230123
7	19	2012010201202120210	16	0123012321032310
8	18	012120212021201212	17	01230123103230123
9	23	01212021202120212021210	21	012301231012321032031
10	23	01202012021202120210202	22	0123012301230123013210
11	29	01201201201202120212021202121	26	01230132012310231203210321
12	28	0121202120212021202120212101	27	012301230123012301230132103
k	$Star_k$		$Pancake_k$	
	R	Schedule	R	Schedule
3	3	010	3	010
4	6	012010	5	02102
5	9	012310320	8	01230130
6	13	0123402413203	11	02102432104
7	18	012345024153012540	15	012345021025012
8	22	0123456031526402143506	20	01234560245602456043

Table 6: Schedules computed with the coloring heuristic and the resulting R and S for four classes of graphs and various k.

definition and the best algorithm) as an example. The matchings are defined by a 3-coloring: color c, $0 \leq c < 3$, consists of all edges between node (i_0, i_1, i_2, i_3) and the node with the first $c+2$ entries of its index reversed. The coloring heuristic finds four schedules with $R = 5$ and $S = 23$: 02102, 12012, 20120 and 21021, the first of these is illustrated in Figure 6. This shows that $Pancake_4$ is a minimum linear gossip graph for 24 nodes. Actually, it is the example given in [6]. In the following we describe the matchings for the four most interesting classes of graphs, schedules based on them are given in Table 6.

Cube-Connected-Cycles. The nodes are indexed by two-tuples (i, j), where $0 \leq i < 2^k$, gives the index of the k-cycle on which this node is lying and where $0 \leq j < k$, gives the index of this node within its cycle. We will speak of *cross edges* for the edges between (i, j) and $(i \pm 2^j, j)$, and of *cycle edges* for the edges between (i, j) and $(i, (j \pm 1) \bmod k)$. We use three matchings covering all edges of CCC_k exactly once. For even k, $\mathcal{M}_0$ contains the edges between $(i, 2 \cdot j)$ and $(i, 2 \cdot j + 1)$, and $\mathcal{M}_1$ the edges between $(i, 2 \cdot j - 1)$ and $(i, 2 \cdot j)$. $\mathcal{M}_2$ contains all cross edges. For odd k the matchings must be slightly modified. $\mathcal{M}_0$ and $\mathcal{M}_1$ each contain $\lfloor k/2 \rfloor$ cycle edges: in $\mathcal{M}_0$, the nodes $(i, k-1)$ remain unmatched, in $\mathcal{M}_1$, the nodes $(i, 0)$. $\mathcal{M}_0$ additionally contains the edges between $(i, k-1)$ and $(i \pm 2^{k-1}, k-1)$, $\mathcal{M}_1$ the edges between $(i, 0)$ and $(i \pm 1, 0)$. $\mathcal{M}_2$ contains all cross edges, except for those in $\mathcal{M}_0$ and $\mathcal{M}_1$, plus the edges between $(i, 0)$ and $(i, k-1)$.

Butterflies. The nodes are indexed again by two-tuples (i, j), $0 \leq i < 2^k$, and $0 \leq j < k$. The cycle edges are defined as before. The cross edges are now running from a node (i, j) to $(i \pm 2^j, (j + 1) \bmod k)$. For even k, $\mathcal{M}_0$ and $\mathcal{M}_1$ are taken as

Figure 6: Gossiping on $Pancake_4$ with $n = 24$ and $m = 36$. The rounds take $1 + 2 + 4 + 6 + 10 = 23$ steps, the optimum.

for CCC_k. $\mathcal{M}_2$ contains the cross edges running from $(i, 2 \cdot j)$ to $(i \pm 2^{2 \cdot j}, 2 \cdot j + 1)$, $\mathcal{M}_3$ the edges from $(i, 2 \cdot j - 1)$ to $(i \pm 2^{2 \cdot j - 1}, 2 \cdot j)$. For odd k, the matchings are somewhat mixed up. $\mathcal{M}_0$ and $\mathcal{M}_1$ contain $\lfloor k/2 \rfloor$ cycles edges from each cycle: for $0 \leq i < 2^{k-1}$, $\mathcal{M}_0$ contains the edges between $(i, 2 \cdot j)$ and $(i, 2 \cdot j + 1)$, for all $0 \leq j \leq (k-3)/2$, for $2^{k-1} \leq i < 2^k$, the edges between $(i, 2 \cdot j - 1)$ and $(i, 2 \cdot j)$, for all $1 \leq j \leq (k-1)/2$. Additionally $\mathcal{M}_0$ contains the edges from $(i, k-1)$ to $(i + 2^{k-1}, 0)$ for all $0 \leq i < 2^{k-1}$. For $0 \leq i < 2^{k-1}$, $\mathcal{M}_1$ contains the edges that $\mathcal{M}_0$ contains for $2^{k-1} \leq i < 2^k$ and vice-versa for the other i. The remaining edges are attributed to $\mathcal{M}_2$ and $\mathcal{M}_3$. These are: the cross edges exept for those starting in $(i, k-1)$ plus the edges between $(i, k-1)$ and $(i, 0)$. $\mathcal{M}_2$ contains these latter edges for $2^{k-2} \leq i < 2 \cdot 2^{k-2}$ and $3 \cdot 2^{k-2} \leq i < 4 \cdot 2^{k-2}$, $\mathcal{M}_3$ for the other i. The allocation of the cross edges is uniquely determined by this. The construction is clarified in Figure 7.

Star and Pancake Graphs. These are the ideal graphs for the coloring heuristic: a minimum cardinality coloring with perfect matchings is so to say part of the definition of the graphs. For $Star_k$ and $Pancake_k$, the nodes are indexed with k-tuples $(i_0, i_1, \ldots, i_{k-1})$, where the set $\{i_l | 0 \leq l < k\}$ constitutes a permutation of $\{0, 1, \ldots, k-1\}$. For $Star_k$, $\mathcal{M}_c$, $0 \leq c \leq k-2$, contains all edges between nodes $(i_0, i_1, \ldots, i_{k-1})$ and the node with i_0 and i_{c+1} exchanged. For $Pancake_k$, $\mathcal{M}_c$, $0 \leq c \leq k-2$, contains all edges between nodes $(i_0, i_1, \ldots, i_{k-1})$ and the node with $i_0, \ldots, i_{c+1}$ replaced by $i_{c+1}, \ldots, i_0$.

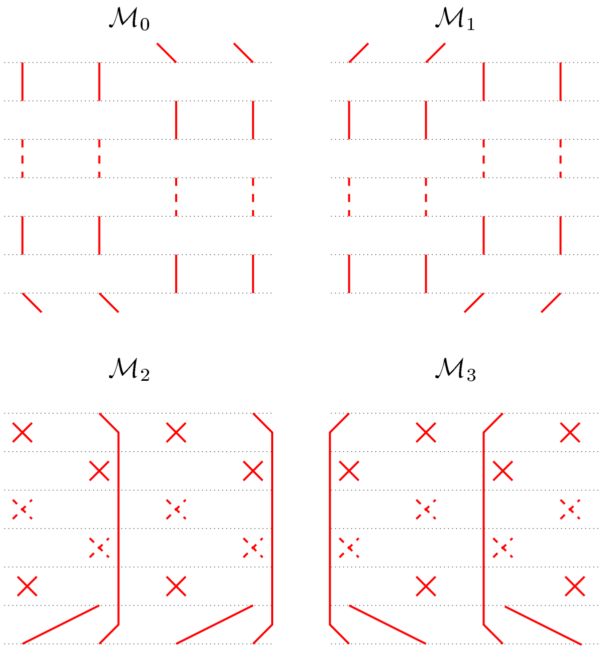

Figure 7: Symbolic representation of the four matchings for $Butterfly_k$, for k odd. The drawn lines indicate matched edges. The vertical lines are cycle edges, the crosses are cross edges. The four columns in each picture indicates the following ranges of i values: $0 \leq i < 2^{k-2}$, $2^{k-2} \leq i < 2 \cdot 2^{k-2}$, $2 \cdot 2^{k-2} \leq i < 3 \cdot 2^{k-2}$ and $3 \cdot 2^{k-2} \leq i < 4 \cdot 2^{k-2}$, respectively.

9 Conclusions and Further Work

We have presented heuristics for gossiping in the telephone model with unit or linear costs. The matching heuristic computes almost optimal schedules and due to its relative efficiency it can do so even for large graphs. Together with the coloring heuristic it leads to improved upper bounds for various important classes of interconnection networks. Generally, these heuristics may become valuable tools for the development of better gossiping and broadcasting schedules.

Not withstanding the achievements, there remains much work to be done. The matching heuristic is fast, but for very large graphs, it is still too slow. If somewhat larger deviations from optimality are acceptable, then the number of dispersion regions can be strongly reduced by first gossiping in small subgraphs. Also one might apply the potential approach in most steps and the BFS approach only in a few critical steps. Parallelization comes without loss of quality: the computation of the edge weights as it is performed in the BFS approach can easily be distributed over the processors of a parallel computer or a cluster of workstations. On the other hand, if the computed schedules are not good enough, then deeper look-ahead might help a bit. This might be developed into a full branch-and-cut approach. More refined structural analysis of the graph might bring more. For the linear-cost model, improvements might be achieved by allowing non-synchronous operations. The analysis should be extended to more practical classes of graphs. Next to the random graphs, whose theoretical properties still should be further studied, we should look at random geometric graphs, planar graphs and real-world examples.

References

[1] Avis, A., 'A Survey of Heuristics for the Weighted Matching Problem,' *Networks*, 13, pp. 475–493, 1983.

[2] Bermond, J.-C., P. Fraigniaud, 'Broadcasting and Gossiping in de Bruijn Networks,' *SIAM Journal on Computing*, 23(1), pp. 212–225, 1994.

[3] Berthomé, P., A. Ferreira, S. Perennes, 'Optimal Information Dissemination in Star and Pancake Networks,' *IEEE Transactions on Parallel and Distributed Systems*, 7(12), pp. 1292–1300, 1996.

[4] Delmas, O., S. Perennes, 'Circuit-Switched Gossiping in 3-Dimensional Torus Networks,' *Proc. 2nd Euro-Par Conference*, LNCS 1123, pp. 370–373, Springer-Verlag, 1996.

[5] Fertin, G., 'Trade-Offs for Odd Gossiping,' *Proc. 6th Colloquium on Structural Information & Communication Complexity*, Proceedings in Informatics 5, pp. 137–151, Carleton Scientific, 1999.

[6] Fraigniaud, P., J.G. Peters, 'Minimum Linear Gossip Graphs and Maximal Linear (Δ, k)-Gossip Graphs,' Techn. Rep. CMPT TR 94-06, Simon Fraser University, Burnaby, B.C., 1994. Available at `http://fas.sfu.ca/pub/cs/techreports/1994/`.

[7] Fraigniaud, P., S. Vial, 'Approximation Algorithms for Broadcasting and Gossiping,' *Journal of Parallel and Distributed Computing*, 43(1), pp. 47–55, 1997.

[8] Fraigniaud, P., S. Vial, 'Heuristic Algorithms for Personalized Communication Problems in Point-to-Point Networks,' *4th Colloquium on Structural Information and Communication Complexity*, pp. 240–252, Carleton Scientific, 1997.

[9] Fraigniaud, P., S. Vial, 'Comparison of Heuristics for One-to-All and All-to-All Communication in Partial Meshes,' *Parallel Processing Letters*, 9(1), pp. 9–20, 1999.

[10] Gvozdjak, P., J.G. Peters, 'Gossiping in Inclined LEO Satelite Networks,' *Proc. 6th Colloquium on Structural Information and Communication Complexity*, Proceedings in Informatics 5, pp. 166–180, Carleton Scientific, 1999.

[11] Juurlink, B., J.F. Sibeyn, P.S. Rao, 'Gossiping on Meshes and Tori,' *IEEE Transactions on Parallel and Distributed Systems*, 9(6), pp. 513–525, 1998.

[12] Hromkovič, J., R. Klasing, B. Monien, R. Peine, 'Dissemination of Information in Interconnection Networks,' *Combinatorial Network Theory*, D.-Z. Du, D.F. Hsu (eds.), pp. 125–212, Kluwer Academic Publishers, 1996.

[13] Knödel, W., 'New Gossips and Telephones,' *Discrete Mathematics*, 13, p. 95, North-Holland, 1975.

[14] Labahn, R., I. Warnke, 'Quick Gossiping by Telegraphs,' *Discrete Mathematics*, 126, pp. 421–424, North-Holland, 1994.

[15] Leighton, T., *Introduction to Parallel Algorithms and Architectures: Arrays-Trees-Hypercubes*, Morgan-Kaufmann Publishers, San Mateo, California, 1992.

[16] Preis, R., 'Linear Time 1/2-Approximation Algorithm for Maximum Weighted Matching in General Graphs' *Proc. 16th Symposium on Theoretical Aspects of Computer Science*, LNCS 1563, pp. 259–269, Springer-Verlag, 1999.

[17] Beier, R., 'Eine Heuristik für das Gossiping-Problem.' *Master Thesis*, Computer Science Department, Universität des Saarlandes, to appear June 2000.

[18] Scheuermann, P., G. Wu, 'Heuristic Algorithms for Broadcasting in Point-to-Point Computer Networks,' *IEEE Transactions on Computers*, C-33(9), 1984.

[19] Šoch, M., P. Tvrdk, 'Optimal Gossip in Store-and-Forward Noncombining 2-D Tori,' *Proc. 3rd International Euro-Par Conference*, LNCS 1300, pp. 234–241, Springer-Verlag, 1997.

I N F O R M A T I K

Below you find a list of the most recent technical reports of the Max-Planck-Institut für Informatik. They are available by anonymous ftp from `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Anja Becker
Im Stadtwald
66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-2000-4-001	J. Kautz, W. Heidrich, K. Daubert	Bump Map Shadows for OpenGL Rendering
MPI-I-2000-1-001	E. Althaus, O. Kohlbacher, H. Lenhof, P. Müller	A branch and cut algorithm for the optimal solution of the side-chain placement problem
MPI-I-1999-4-001	J. Haber, H. Seidel	A Framework for Evaluating the Quality of Lossy Image Compression
MPI-I-1999-3-005	T.A. Henzinger, J. Raskin, P. Schobbens	Axioms for Real-Time Logics
MPI-I-1999-3-004	J. Raskin, P. Schobbens	Proving a conjecture of Andreka on temporal logic
MPI-I-1999-3-003	T.A. Henzinger, J. Raskin, P. Schobbens	Fully Decidable Logics, Automata and Classical Theories for Defining Regular Real-Time Languages
MPI-I-1999-3-002	J. Raskin, P. Schobbens	The Logic of Event Clocks
MPI-I-1999-3-001	S. Vorobyov	New Lower Bounds for the Expressiveness and the Higher-Order Matching Problem in the Simply Typed Lambda Calculus
MPI-I-1999-2-008	A. Bockmayr, F. Eisenbrand	Cutting Planes and the Elementary Closure in Fixed Dimension
MPI-I-1999-2-007	G. Delzanno, J. Raskin	Symbolic Representation of Upward-closed Sets
MPI-I-1999-2-006	A. Nonnengart	A Deductive Model Checking Approach for Hybrid Systems
MPI-I-1999-2-005	J. Wu	Symmetries in Logic Programs
MPI-I-1999-2-004	V. Cortier, H. Ganzinger, F. Jacquemard, M. Veanes	Decidable fragments of simultaneous rigid reachability
MPI-I-1999-2-003	U. Waldmann	Cancellative Superposition Decides the Theory of Divisible Torsion-Free Abelian Groups
MPI-I-1999-2-001	W. Charatonik	Automata on DAG Representations of Finite Trees
MPI-I-1999-1-007	C. Burnikel, K. Mehlhorn, M. Seel	A simple way to recognize a correct Voronoi diagram of line segments
MPI-I-1999-1-006	M. Nissen	Integration of Graph Iterators into LEDA
MPI-I-1999-1-005	J.F. Sibeyn	Ultimate Parallel List Ranking ?
MPI-I-1999-1-004	M. Nissen, K. Weihe	How generic language extensions enable “open-world” desing in Java
MPI-I-1999-1-003	P. Sanders, S. Egner, J. Korst	Fast Concurrent Access to Parallel Disks
MPI-I-1999-1-002	N.P. Boghossian, O. Kohlbacher, H.-. Lenhof	BALL: Biochemical Algorithms Library
MPI-I-1999-1-001	A. Crauser, P. Ferragina	A Theoretical and Experimental Study on the Construction of Suffix Arrays in External Memory

MPI-I-98-2-018	F. Eisenbrand	A Note on the Membership Problem for the First Elementary Closure of a Polyhedron
MPI-I-98-2-017	M. Tzakova, P. Blackburn	Hybridizing Concept Languages
MPI-I-98-2-014	Y. Gurevich, M. Veanes	Partisan Corroboration, and Shifted Pairing
MPI-I-98-2-013	H. Ganzinger, F. Jacquemard, M. Veanes	Rigid Reachability
MPI-I-98-2-012	G. Delzanno, A. Podelski	Model Checking Infinite-state Systems in CLP
MPI-I-98-2-011	A. Degtyarev, A. Voronkov	Equality Reasoning in Sequent-Based Calculi
MPI-I-98-2-010	S. Ramangalahy	Strategies for Conformance Testing
MPI-I-98-2-009	S. Vorobyov	The Undecidability of the First-Order Theories of One Step Rewriting in Linear Canonical Systems
MPI-I-98-2-008	S. Vorobyov	AE-Equational theory of context unification is Co-RE-Hard
MPI-I-98-2-007	S. Vorobyov	The Most Nonelementary Theory (A Direct Lower Bound Proof)
MPI-I-98-2-006	P. Blackburn, M. Tzakova	Hybrid Languages and Temporal Logic
MPI-I-98-2-005	M. Veanes	The Relation Between Second-Order Unification and Simultaneous Rigid E-Unification
MPI-I-98-2-004	S. Vorobyov	Satisfiability of Functional+Record Subtype Constraints is NP-Hard
MPI-I-98-2-003	R.A. Schmidt	E-Unification for Subsystems of S4
MPI-I-98-2-002	F. Jacquemard, C. Meyer, C. Weidenbach	Unification in Extensions of Shallow Equational Theories
MPI-I-98-1-031	G.W. Klau, P. Mutzel	Optimal Compaction of Orthogonal Grid Drawings
MPI-I-98-1-030	H. Brönniman, L. Kettner, S. Schirra, R. Veltkamp	Applications of the Generic Programming Paradigm in the Design of CGAL
MPI-I-98-1-029	P. Mutzel, R. Weiskircher	Optimizing Over All Combinatorial Embeddings of a Planar Graph
MPI-I-98-1-028	A. Crauser, K. Mehlhorn, E. Althaus, K. Brengel, T. Buchheit, J. Keller, H. Krone, O. Lambert, R. Schulte, S. Thiel, M. Westphal, R. Wirth	On the performance of LEDA-SM
MPI-I-98-1-027	C. Burnikel	Delaunay Graphs by Divide and Conquer
MPI-I-98-1-026	K. Jansen, L. Porkolab	Improved Approximation Schemes for Scheduling Unrelated Parallel Machines
MPI-I-98-1-025	K. Jansen, L. Porkolab	Linear-time Approximation Schemes for Scheduling Malleable Parallel Tasks
MPI-I-98-1-024	S. Burkhardt, A. Crauser, P. Ferragina, H. Lenhof, E. Rivals, M. Vingron	q-gram Based Database Searching Using a Suffix Array (QUASAR)
MPI-I-98-1-023	C. Burnikel	Rational Points on Circles
MPI-I-98-1-022	C. Burnikel, J. Ziegler	Fast Recursive Division
MPI-I-98-1-021	S. Albers, G. Schmidt	Scheduling with Unexpected Machine Breakdowns
MPI-I-98-1-020	C. Rüb	On Wallace's Method for the Generation of Normal Variates
MPI-I-98-1-019		2nd Workshop on Algorithm Engineering WAE '98 - Proceedings
MPI-I-98-1-018	D. Dubhashi, D. Ranjan	On Positive Influence and Negative Dependence
MPI-I-98-1-017	A. Crauser, P. Ferragina, K. Mehlhorn, U. Meyer, E. Ramos	Randomized External-Memory Algorithms for Some Geometric Problems
MPI-I-98-1-016	P. Krysta, K. Loryś	New Approximation Algorithms for the Achromatic Number
MPI-I-98-1-015	M.R. Henzinger, S. Leonardi	Scheduling Multicasts on Unit-Capacity Trees and Meshes
MPI-I-98-1-014	U. Meyer, J.F. Sibeyn	Time-Independent Gossiping on Full-Port Tori