Experiments in Routing h-Relations Using Constant Thinning And Geometric Thinning Algorithms

Anssi Kautonen
University of Joensuu, Department of Computer Science,
P.O. Box 111, 80101 Joensuu, Finland
email Anssi.Kautonen@cs.joensuu.fi

Abstract

We have experimented with two protocols designed for routing *h-relations* on complete networks under so-called OCPC assumption. These protocols, called *constant thinning* and *geometric thinning* protocol, are theoretically analyzed in [10, 11]. In this report, we estimate experimentally the efficiency of the protocols.

1 Introduction

There is need for efficient algorithms for routing in complete networks for example when simulating PRAM's on distributed memory machines (see e.g. [12]).

We assume the *OCPC* (Optical Communication Parallel Computer) model. Communication network is a complete network, thus distance between any pair of nodes is one and degree of nodes is $p-1$. Processors communicate with each other by transmitting messages [1]. A processor can transmit a message to any other processor and the transmission takes one time unit. At any time a processor can send at most one message. The message will succeed in reaching the processor, if it is the only message with that processor as its destination at that time step. If two or more processors try to send a message to the same processor, no transmission to that processor is successful and a retransmission must occur.

Any routing problem in OCPC reduces to realizing an h-relation. In an h-relation, each processor is the source as well as the destination of at most h messages. We assume that each processor knows the value of h, when the routing begins. Algorithms for solving the h-relation problem on the OCPC model have been provided by [2, 3, 4, 5, 6, 8, 9, 11].

A goal in routing h-relations is to route messages within $\theta(h)$ steps. Algorithms should be optimal for as small h as possible.

In Section 2 we present two algorithms for routing h-relations, and in Section 3 we present some experimental results. Simulations are done with a routing simulator.

2 Routing Algorithms

The simplest method to route messages is called a Greedy algorithm. In this algorithm each processors attemps to send a message at each time step until they have sended all their messages. This is not a good method, because in the end there are usually several processors, which all attempt to send their message(s) to the same processor(s). The result is a livelock. Some packets can cause mutual failure of sending until eternity.

If we want to avoid livelocks, we must regulate the transmission rate. We present two algorithms to route h-relations. They are called the constant thinning algorithm [10] and the geometric thinning algorithm [11]. Both are variations of the same idea. Packet routing is divided into phases. Each phase consists of active sending steps but also of a suitable number of passive steps.

During each phase we want that each processor sends successfully a predefined proportion (e.g. $1/e$) of the remaining packets. Also we want that each processor receives successfully predefined proportion of the packets destined for it. Thus, the effect of each phase is to reduce a k-relation to a ck-relation, where $c < 1$.

At phase i both algorithms allocate h_i time slots for packets from time window $1..\lceil \delta t h_i \rceil$. Thus, during the i'th transmission phase each of h_i selected packet is tried exactly once.

The goal of i'th phase (i=0,1,2,...) is to reduce a $(1-e^{-1/t})^i h$-relation to a $(1-e^{-1/t})^{i+1} h$-relation.

The probability that a packet chosen to be transmitted, is successfully transmitted, is

$$\left(1 - \frac{1}{\delta t h}\right)^m \geq \left(1 - \frac{1}{\delta t h}\right)^{\frac{1}{\delta t}\delta t h} \geq \frac{1}{e^{1/\delta t}},$$

where m denotes the number of packets that can potentially collide with the packet chosen to be transmitted. When the value of δ is increased the probability of successful transmission also increases.

If we choose $\delta = 1$ in CT algorithm (see below), then the expected number of successful transmissions is $h_i * e^{-1/t}$. So we can decrease the size of time window by factor $1 - e^{-1/t}$ after each phase. The bigger the $\delta > 1$ we choose the more likely all processors succeed in reducing a $(1 - e^{-1/t})^i h$-relation to a $(1 - e^{-1/t})^{i+1} h$-relation.

In the end, the situation may become unbalanced. Some processors have sent all their packets but some processors still have many packets to send. To prevent a livelock situation we have a lower bound for the size of the time window. This is the parameter h_0. The whole algorithm is as follows.

proc CT(h,h_0,t,δ)
% $t, \delta \geq 1$
for all processors **do**
  while packets remain **do**
    Transmit h packets (if so many remain) within time $[1..\lceil \delta t h \rceil]$
    $h := \max\{(1 - e^{-1/t})h, h_0\}$

In the GT algorithm, the value of t increases as a geometrical series, when the degree of the h-relation decreases. So, we decrease the size of the time window slower than what is the expected number of remaining packets. The ratio of time window size divided by the number of remaining packets increases in each phase. When processors have a lot of packets remaining, the variation in the number of successful transmissions is small. When we have only a few packets remaining, large variations in the number of successful transmission are likely. More extra space in the time window size at the end of routing process ensures that variation is not too high. The algorithm is as follows.

proc GT(h,h_0,d,δ,t_{max})
% $d, \delta, t_{max} \geq 1$
for all processors **do**
    $t := 1$
    while packets remain **do**
        Transmit h packets (if so many remain) within time $[1..\lceil \delta th \rceil]$
        $h := \max\{(1 - e^{-1/t})h, h_0\}; t := \min\{t_{max}, dt\}$

A drawback of thinning is that a processor cannot successfully send a packet at those moments, when it does not even try to send. Thinning by factor t would thus imply inefficiency by factor t. But this is somewhat balanced by the success probability, which increases from $1/e$ to

$$(1 - \frac{1}{th})^{h-1} \geq \frac{1}{e^{1/t}}.$$

The expected throughput with thinning is thus characterized by the function $te^{1/t}$ whose growth

t	1.0	1.2	1.5	2.0	2.5	3.0	4.0
$te^{1/t}$	2.7	2.8	2.9	3.3	3.7	4.2	5.1

is very modest with small values of $t > 1$. The price for increased robustness seems to be rather small.

3 Experiments

3.1 Simulation experiments

We ran some experiments to test the practical efficiency of the algorithms. Experiments are done with a packet routing simulator. Results are averages of 1000 experiments. In each experiment each processor sends h packets and each processor is the destination of h packets.

The destinations of the packets are chosen in the following way at the beginning of routing: First each processor gets one packet, then each processors gets its second packet, and so on until each processor has h packets to send. Each time a processor gets a new packet, the destination of the packet is chosen uniformly at random among those processors, which have less than h packets targetted to them so far. The destination of the packet can not be the sending processor itself.

The random number generator used in these experiments is slightly modified version of Robert J. Jenkins Jr.'s ISAAC (Indirection, Shift, Accumulate, Add, and Count) pseudo-random number generator [7]. There are no cycles in the ISAAC shorter than 2^{40} values. The expected cycle length is 2^{8295} values.

Each processor has one transmitter and one receiver. During one timestep a processor can send at most one packet. The packet will succeed in reaching the receiver, if it is the only message with that receiver as its destination at that routing step. If a receiver gets a packet, then the processor sends an acknowledgement to the sender of the packet. The packets that are not acknowledged must be retransmitted.

Table 1: Comparison between constant thinning and geometric thinning algorithms.

Processors	h	h_0	δ	t^{CT}	d^{GT}	t^{GT}_{max}	CT cost	CT std	GT cost	GT std
1024	8	5	1.2	1.0	1.05	1.5	**6.11**	0.57	6.64	0.72
1024	8	10	1.2	1.0	1.05	1.5	7.92	0.82	8.61	0.93
1024	8	5	1.2	1.1	1.1	1.5	6.39	0.65	6.84	0.66
1024	8	10	1.2	1.1	1.1	1.5	8.27	0.85	*9.22*	0.99
1024	16	5	1.2	1.0	1.05	1.5	**4.89**	0.32	5.13	0.35
1024	16	10	1.2	1.0	1.05	1.5	5.62	0.43	6.10	0.51
1024	16	5	1.2	1.1	1.1	1.5	4.99	0.36	5.20	0.37
1024	16	10	1.2	1.1	1.1	1.5	5.97	0.47	*6.44*	0.48
1024	32	5	1.2	1.0	1.05	1.5	4.37	0.34	4.30	0.23
1024	32	10	1.2	1.0	1.05	1.5	4.45	0.23	4.75	0.28
1024	32	5	1.2	1.1	1.1	1.5	**4.29**	0.25	**4.29**	0.21
1024	32	10	1.2	1.1	1.1	1.5	4.64	0.26	*4.86*	0.25
1024	64	5	1.2	1.0	1.05	1.5	*4.05*	0.49	**3.79**	0.15
1024	64	10	1.2	1.0	1.05	1.5	3.84	0.15	3.98	0.15
1024	64	5	1.2	1.1	1.1	1.5	3.86	0.24	3.79	0.17
1024	64	10	1.2	1.1	1.1	1.5	3.92	0.14	4.04	0.14
1024	128	5	1.2	1.0	1.05	1.5	3.54	0.38	**3.45**	0.12
1024	128	10	1.2	1.0	1.05	1.5	3.47	0.11	3.56	0.08
1024	128	5	1.2	1.1	1.1	1.5	3.53	0.18	3.48	0.08
1024	128	10	1.2	1.1	1.1	1.5	3.53	0.09	*3.59*	0.08
1024	256	5	1.2	1.0	1.05	1.5	**3.24**	0.11	3.29	0.04
1024	256	10	1.2	1.0	1.05	1.5	3.26	0.04	3.33	0.04
1024	256	5	1.2	1.1	1.1	1.5	3.28	0.06	3.33	0.30
1024	256	10	1.2	1.1	1.1	1.5	3.31	0.05	*3.38*	0.04

It is assumed that processors have a common clock. All processors accomplish the packet transmission step at the same time.

The result of the simulation is *cost*, i.e. the number of time steps required to route all packets divided by h. This ratio should be as close to one as possible. The packet sending round and and the corresponding acknowledgement round are counted as a single round. Another result of a series of routing experiments is *the standard deviation of the cost*.

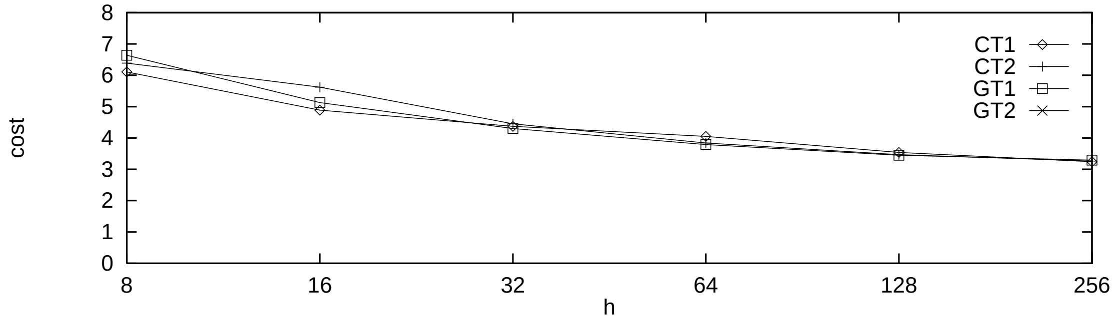

Figure 1: CT and GT results. In the figure, CT1 is a CT algorithm with parameter values $p = 1024$, $h_0 = 5$, $\delta = 1.2$, $t = 1.0$, and $h = 8...256$. CT2 is similar to CT1 except that $t = 1.1$. Curves GT1 and GT2 represent results of GT algorithm simulations. Values of parameters are $p = 1024$, $h_0 = 5$, $\delta = 1.2$, and $t_{max} = 1.5$. In GT1 $d = 1.05$, and in GT2 $d = 1.1$.

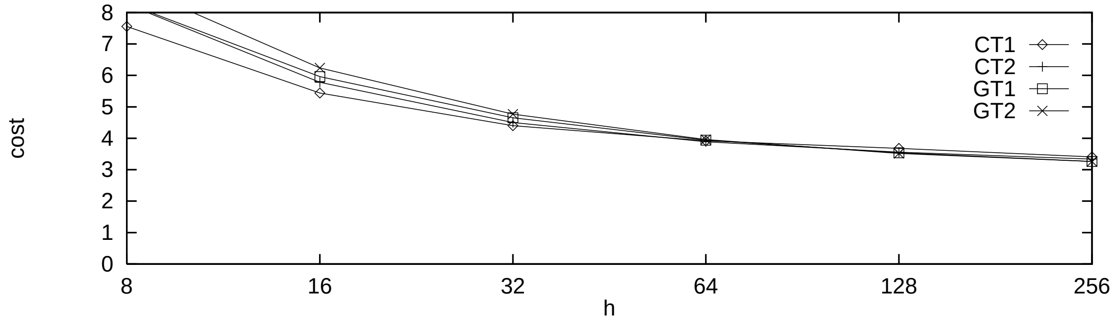

Figure 2: More CT and GT results. This is similar to Figure 1 except that $\delta = 1.1$.

Table 2: Another comparison between constant thinning and geometric thinning algorithms. Here, $\delta = 1.1$.

Processors	h	h_0	δ	t^{CT}	d^{GT}	t^{GT}_{max}	CT cost	CT std	GT cost	GT std
1024	8	5	1.1	1.0	1.05	1.5	**6.08**	0.54	6.40	0.66
1024	8	10	1.1	1.0	1.05	1.5	7.56	0.76	8.29	0.97
1024	8	5	1.1	1.1	1.1	1.5	6.39	0.65	6.79	0.75
1024	8	10	1.1	1.1	1.1	1.5	8.27	0.85	*8.92*	0.95
1024	16	5	1.1	1.0	1.05	1.5	**4.91**	0.36	5.09	0.40
1024	16	10	1.1	1.0	1.05	1.5	5.44	0.39	5.96	0.52
1024	16	5	1.1	1.1	1.1	1.5	4.98	0.36	5.16	0.35
1024	16	10	1.1	1.1	1.1	1.5	5.78	0.47	*6.24*	0.54
1024	32	5	1.1	1.0	1.05	1.5	4.58	0.42	4.33	0.25
1024	32	10	1.1	1.0	1.05	1.5	4.40	0.23	4.65	0.25
1024	32	5	1.1	1.1	1.1	1.5	4.33	0.28	**4.28**	0.23
1024	32	10	1.1	1.1	1.1	1.5	4.50	0.23	*4.77*	0.28
1024	64	5	1.1	1.0	1.05	1.5	*4.88*	0.94	3.92	0.23
1024	64	10	1.1	1.0	1.05	1.5	3.92	0.20	3.94	0.15
1024	64	5	1.1	1.1	1.1	1.5	4.20	0.46	**3.85**	0.20
1024	64	10	1.1	1.1	1.1	1.5	3.89	0.16	3.96	0.15
1024	128	5	1.1	1.0	1.05	1.5	*5.85*	3.00	3.66	0.31
1024	128	10	1.1	1.0	1.05	1.5	3.68	0.24	3.54	0.12
1024	128	5	1.1	1.1	1.1	1.5	4.27	0.89	3.56	0.22
1024	128	10	1.1	1.1	1.1	1.5	3.56	0.14	**3.52**	0.10
1024	256	5	1.1	1.0	1.05	1.5	L	L	3.35	0.23
1024	256	10	1.1	1.0	1.05	1.5	3.41	0.26	**3.26**	0.08
1024	256	5	1.1	1.1	1.1	1.5	4.07	1.10	3.29	0.14
1024	256	10	1.1	1.1	1.1	1.5	3.34	0.15	**3.26**	0.06

3.2 The results of experiments

In the Tables 1-4 are comparisions between the CT algorithm and the GT algorithm. In the Table 5 are the results of experiments, in which is tried to find good values for parameters of the CT algorithm. In the Tables 6-8 are the results of experiments, in which is tried find good values for parameters of the GT algorithm.

The results presented in tables are part of the results we got in larger experiments. The values of the parameters chosen for tables seem to give good results.

In the experiments 5-8 (see Tables 5-8) is tried to find sensible ranges for the values of parameters when $p = 1024$ and $h = 8...256$. For each parameter it is tried to find such value that increasing the value of the parameter increases the cost, and decresing the value

Table 3: CT versus GT comparison, where number of processors varies from 128 to 4096.

Processors	h	h_0	δ	t^{CT}	d^{GT}	t^{GT}_{max}	CT cost	CT std	GT cost	GT std
128	49	3	1.1	1.1	1.1	1.2	4.26	1.05	*4.51*	1.40
128	49	7	1.1	1.1	1.1	1.2	3.68	0.21	**3.67**	0.19
128	49	3	1.1	2.0	1.1	1.5	4.17	0.16	3.81	0.59
128	49	7	1.1	2.0	1.1	1.5	4.44	0.18	3.73	0.18
256	64	4	1.1	1.1	1.1	1.2	4.35	1.01	3.78	0.37
256	64	8	1.1	1.1	1.1	1.2	**3.70**	0.19	3.65	0.16
256	64	4	1.1	2.0	1.1	1.5	4.15	0.12	3.68	0.28
256	64	8	1.1	2.0	1.1	1.5	*4.40*	0.15	3.71	0.16
512	81	4	1.1	1.1	1.1	1.2	*5.18*	2.46	4.00	0.61
512	81	9	1.1	1.1	1.1	1.2	3.69	0.17	**3.64**	0.14
512	81	4	1.1	2.0	1.1	1.5	4.13	0.11	3.69	0.27
512	81	9	1.1	2.0	1.1	1.5	4.37	0.12	3.66	0.13
1024	100	5	1.1	1.1	1.1	1.2	4.20	0.65	4.00	0.56
1024	100	10	1.1	1.1	1.1	1.2	3.67	0.15	3.63	0.14
1024	100	5	1.1	2.0	1.1	1.5	4.11	0.09	**3.61**	0.19
1024	100	10	1.1	2.0	1.1	1.5	*4.32*	0.10	3.64	0.11
2048	121	5	1.1	1.1	1.1	1.2	*4.53*	0.82	4.05	0.53
2048	121	11	1.1	1.1	1.1	1.2	3.65	0.13	**3.59**	0.11
2048	121	5	1.1	2.0	1.1	1.5	4.08	0.08	3.64	0.22
2048	121	11	1.1	2.0	1.1	1.5	4.32	0.10	3.61	0.09
4096	144	6	1.1	1.1	1.1	1.2	*4.43*	0.75	3.99	0.49
4096	144	12	1.1	1.1	1.1	1.2	3.64	0.13	**3.57**	0.10
4096	144	6	1.1	2.0	1.1	1.5	4.08	0.07	3.60	0.18
4096	144	12	1.1	2.0	1.1	1.5	4.29	0.09	3.58	0.08

of parameter also increases the cost or causes a livelock. The simulations show however that the change of one parameter changes the optimal value of the other parameters.

In the tables the best and the worst result of each p, h pair is highlighted. The best result is in bold text, and the worst result is in italic text.

In the Table 1, the simulations of CT and GT are compared. In these experiments, we had 1024 processors and $\delta = 1.2$. The value of h varies from 8 to 256 and the value of h_0 is either 5 ($= 1/2 * \log_2 p$) or 10 ($= \log_2 p$). The CT algorithm is tested with 2 differrent values of t. In the GT algorithm experiments, the value of parameter d varies between 1.05 and 1.1. The value of t_{max} is 1.5. The Figure 1 describes partly the results shown in Table 1. A logarithmic scale (for h) is used in the figure.

In Table 2, we have another CT versus GT comparison. The difference to Table 1 is

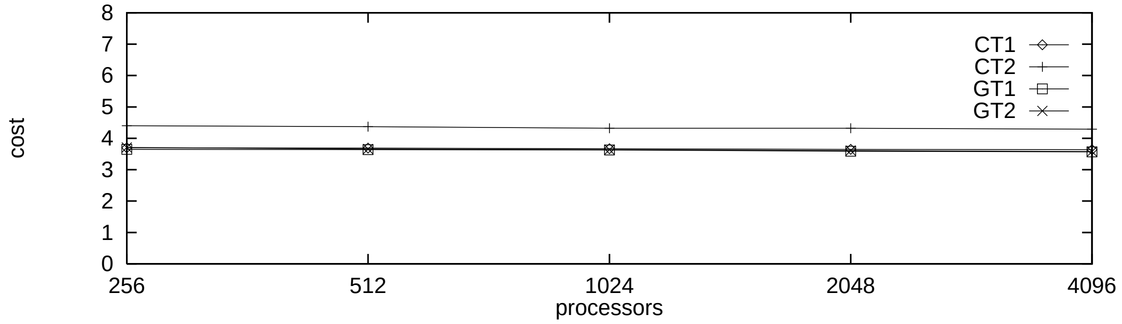

Figure 3: Now the values of CT1 parameters are $p = 128...4096$, $h = \log_2^2 p$, $h_0 = \log_2 p$, $\delta = 1.1$, and $t = 1.1$. CT2 is similar to CT except that $t = 2.0$. In GT1 and GT2, the values of parameters are $p = 128...4096$, $h = \log_2^2 p$, $\delta = 1.1$, and $d = 1.1$. In GT1 $t_{max} = 1.2$, and in GT2 $t_{max} = 1.5$.

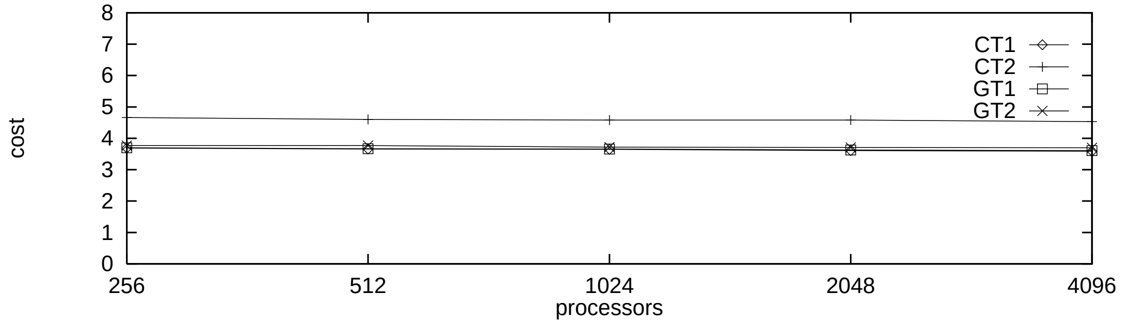

Figure 4: The curves represent the results of experiments which are similar to experiments reported in Figure 3 except that $\delta = 1.2$.

that $\delta = 1.1$. In the table, value 'L' means that the cost of simulation is more than 100. This is possible, when h_0 is chosen to be very small. (In the $O(h)$ time proof in [10], a very safe value $h_0 > 100 \ln p$ was used.) The results of Table 2 are partly illustrated in Figure 2.

In Table 3, there are results of another kind of comparison between the CT and GT algorithms. In these experiments the number of processors varies from 128 to 4096. Value of $h = \log_2^2 p$, where p is the number of the processors. The tested values of h_0 are $\log_2 p$, and $1/2 * \log_2 p$.

In Table 4 are the results of experiments which are similar to experiments reported in Table 3 except that $\delta = 1.2$.

The results of experiments 1-4 show that the CT and the GT algorithms are almost equally good. The CT algorithm seems to have a little higher livelock risk. The experiments have also showed that finding good values for the parameters of CT algorithm is more difficult than finding good values for the parameters of GT algorithm.

In Table 5 are the results of the experiments in which is tried to find the best value for the parameters δ and h_0 of CT. In these experiments the number of the processors is 1024. The value of h varies from 8 to 256, $t = 1.0$,, $h_0 = 5...15$, and $\delta = 1.0...1.4$.

Table 4: Another CT versus GT comparison, where number of processors varies from 128 to 4096.

Processors	h	h_0	δ	t^{CT}	d^{GT}	t^{GT}_{max}	CT cost	CT std	GT cost	GT std
128	49	3	1.2	1.1	1.1	1.2	3.83	0.58	3.70	0.32
128	49	7	1.2	1.1	1.1	1.2	3.73	0.18	3.73	0.18
128	49	3	1.2	2.0	1.1	1.5	4.38	0.14	**3.63**	0.20
128	49	7	1.2	2.0	1.1	1.5	*4.66*	0.19	3.79	0.18
256	64	4	1.2	1.1	1.1	1.2	3.67	0.25	**3.61**	0.19
256	64	8	1.2	1.1	1.1	1.2	3.69	0.14	3.70	0.14
256	64	4	1.2	2.0	1.1	1.5	4.38	0.12	**3.61**	0.19
256	64	8	1.2	2.0	1.1	1.5	*4.66*	0.14	3.77	0.14
512	81	4	1.2	1.1	1.1	1.2	3.70	0.28	3.60	0.20
512	81	9	1.2	1.1	1.1	1.2	3.66	0.13	3.67	0.12
512	81	4	1.2	2.0	1.1	1.5	4.32	0.09	**3.59**	0.11
512	81	9	1.2	2.0	1.1	1.5	*4.60*	0.14	3.77	0.12
1024	100	5	1.2	1.1	1.1	1.2	3.69	0.26	3.59	0.13
1024	100	10	1.2	1.1	1.1	1.2	3.65	0.11	3.66	0.10
1024	100	5	1.2	2.0	1.1	1.5	4.33	0.08	**3.57**	0.10
1024	100	10	1.2	2.0	1.1	1.5	*4.58*	0.11	3.72	0.10
2048	121	5	1.2	1.1	1.1	1.2	3.64	0.23	**3.54**	0.11
2048	121	11	1.2	1.1	1.1	1.2	3.61	0.09	3.63	0.08
2048	121	5	1.2	2.0	1.1	1.5	4.29	0.07	**3.54**	0.08
2048	121	11	1.2	2.0	1.1	1.5	4.58	0.09	3.71	0.09
4096	144	6	1.2	1.1	1.1	1.2	3.59	0.19	**3.53**	0.10
4096	144	12	1.2	1.1	1.1	1.2	3.59	0.08	3.61	0.07
4096	144	6	1.2	2.0	1.1	1.5	4.30	0.06	3.54	0.07
4096	144	12	1.2	2.0	1.1	1.5	*4.53*	0.07	3.70	0.07

The results of the experiments, in which we tried to find the best value for the parameter δ of GT, are reported in Table 6. In these experiments, the number of processors is 1024. The value of h varies from 8 to 256, $h_0 = 5...10$, $d = 1.05...1.1$, $\delta = 1.0...1.4$, and $t_{max} = 1.5$.

In Tables 7 and 8 are reported the results of experiments, in which we tried to find the best values for d and t_{max} of GT. In the both experiments the number of processors is 1024. The value of h varies from 8 to 256, and $h_0 = 5$. In the experiments of Table 7, values of the parameters are $\delta = 1.1...1.2$, and $t_{max} = 1.5...2.0$. The values of the parameter d used in the experiments are 1.05, 1.1, 1.15, 1.2, and 1.25. In Table 8 $\delta = 1.1...1.2$, $d = 1.15$, and the tested t_{max} values are 1.25, 1.5, 1.75, and 2.0.

The results in Table 5 show how the cost of the simulation depends on the values of

Table 5: Finding best value for δ in CT algorithm.

h	h_0	1.0	1.05	1.1	1.2	1.3	1.4
8	5	**5.88**	6.07	6.08	6.11	6.44	6.43
8	10	7.32	7.59	7.56	7.92	8.23	8.53
8	15	8.89	9.21	9.51	9.83	10.39	*10.65*
16	5	5.11	4.92	4.91	**4.89**	4.97	5.04
16	10	5.31	5.44	5.44	5.62	5.80	6.01
16	15	6.07	6.25	6.44	6.62	6.92	*7.12*
32	5	5.99	4.72	4.58	4.37	**4.24**	4.27
32	10	4.44	4.40	4.40	4.45	4.53	4.68
32	15	4.63	4.71	4.77	4.87	5.07	*5.20*
64	5	*LL*	5.53	4.88	4.05	**3.78**	3.83
64	10	4.25	4.01	3.92	3.84	3.90	4.00
64	15	4.01	3.97	3.97	4.01	4.13	4.23
128	5	*LL*	*LL*	5.85	3.54	3.49	3.66
128	10	4.92	4.00	3.68	**3.47**	3.56	3.72
128	15	3.87	3.66	3.57	3.54	3.65	3.80
256	5	*LL*	*LL*	*LL*	**3.24**	3.42	3.64
256	10	*LL*	4.28	3.41	3.26	**3.24**	3.64
256	15	4.26	3.55	3.30	3.29	3.54	*3.65*

the parameters h_0 and δ in CT. In the case $p = 1024$, values 1.2 and 1.3 for the parameter δ seem to yield best results. Values close to 1.0 cause a high livelock risk. Safe values for δ are larger as the h becomes larger. Optimal value for δ depends also on h_0. Small values of δ requires large h_0, and for larger δ values, a small h_0 is enough. On small values of h, the cost of the simulation depends mostly on how optimal value for h_0 has been found. The experiments also show that the larger h is the larger h_0 is required. Best values for t are close one, and the cost increses as the the value of t increases.

The GT algorithm gives similar kind of results that the CT algorithm gave. The results are reported in Tables 6, 7, and 8.

To achive good results, the most important thing to find a good combination of δ and h_0. The Table 6 shows that the best values for δ are between 1.1 and 1.3. Value of h_0 must be increased as the h becomes larger. When using the GT algorithm, smaller values of h_0 give better results than in the CT case. A good value for h_0 depends on δ, but most cases value 10 is enough.

The Table 7 shows that good values for d are close to one (1.05 ... 1.1), and the Table 8 shows that good values for t_{max} are between 1.5 and 2.0. The choice of values of the parameters d and t_{max} has less effect on the simulation cost than the choice δ and h_0, but they can used in finetuning the algorithm.

Table 6: Finding best value for δ in GT algorithm.

h	h_0	d	1.0	1.05	1.1	1.2	1.3	1.4
8	5	1.05	**6.34**	6.40	6.46	6.66	6.82	7.03
8	5	1.1	6.53	6.57	6.75	6.80	7.09	7.33
8	10	1.05	8.02	8.08	8.32	8.64	9.05	9.48
8	10	1.1	8.47	8.59	8.93	9.21	9.66	*9.93*
16	5	1.05	5.06	5.05	5.05	5.14	5.23	5.37
16	5	1.1	**5.04**	5.08	5.22	5.19	5.36	5.53
16	10	1.05	5.79	5.89	5.97	6.08	6.32	6.52
16	10	1.1	6.03	6.11	6.25	6.44	6.74	*6.85*
32	5	1.05	4.39	4.34	4.34	4.29	4.37	4.45
32	5	1.1	4.32	**4.27**	4.29	4.29	4.41	4.54
32	10	1.05	4.60	4.60	4.68	4.74	4.87	4.99
32	10	1.1	4.62	4.70	4.78	4.88	5.07	*5.17*
64	5	1.05	4.33	4.06	3.91	3.79	3.85	3.97
64	5	1.1	4.10	3.94	3.84	**3.78**	3.88	4.02
64	10	1.05	3.97	3.95	3.94	3.97	4.09	4.21
64	10	1.1	3.92	3.93	3.97	4.03	4.18	*4.32*
128	5	1.05	4.94	4.21	3.65	**3.46**	3.59	3.76
128	5	1.1	4.34	3.92	3.56	3.48	3.62	3.83
128	10	1.05	3.67	3.57	3.54	3.56	3.70	3.85
128	10	1.1	3.56	3.53	3.52	3.59	3.76	*3.95*
256	5	1.05	*LL*	*LL*	3.36	3.29	3.48	3.71
256	5	1.1	*LL*	3.93	3.30	3.33	3.55	3.78
256	10	1.05	3.63	3.37	**3.26**	3.33	3.52	3.73
256	10	1.1	3.43	3.30	**3.26**	3.39	3.59	3.81

The experiments show that both algorithms give good results, if the parameters are chosen corretly. However, finding the best values for the parameters is not easy because changing the value of one parameter effects on the optimal values of the other parameters. The values in the tables give limits where to start looking for the best values for parameters.

4 Conclusion

There is a “tail” phase at the end of the simulation. In this phase, most of the processors have sent all their packets successfully but some processors still have few packets remaining to send. Also some processors are still receiving packets. This phase can cause significant increase to the overall routing cost, there are either a lot of collisions or many steps, when

Table 7: Finding best value for d in GT algorithm.

h	δ	t_{max}	1.05	1.1	1.15	1.2	1.25
8	1.1	1.5	**6.46**	6.66	6.84	6.87	6.87
8	1.2	1.5	6.75	6.80	6.89	6.98	6.97
8	1.1	2.0	6.51	6.80	7.14	7.30	7.40
8	1.2	2.0	6.63	6.99	7.40	7.60	*7.69*
16	1.1	1.5	**5.05**	5.14	5.17	5.19	5.21
16	1.2	1.5	5.22	5.19	5.22	5.24	5.27
16	1.1	2.0	5.10	5.29	5.42	5.43	5.47
16	1.2	2.0	5.15	5.40	5.53	5.63	*5.67*
32	1.1	1.5	**4.27**	4.29	4.28	4.29	4.31
32	1.2	1.5	4.29	4.29	4.31	4.34	4.34
32	1.1	2.0	4.34	4.38	4.38	4.39	4.40
32	1.2	2.0	4.31	4.42	4.48	4.53	*4.55*
64	1.1	1.5	3.91	3.79	3.80	3.80	3.81
64	1.2	1.5	3.84	**3.78**	3.81	3.83	3.84
64	1.1	2.0	3.92	3.86	3.80	3.81	3.84
64	1.2	2.0	3.80	3.84	3.87	3.91	*3.96*
128	1.1	1.5	*3.65*	3.56	3.53	5.52	3.53
128	1.2	1.5	**3.46**	3.48	3.52	3.53	3.56
128	1.1	2.0	3.63	3.50	3.47	3.48	3.50
128	1.2	2.0	3.47	3.51	3.55	3.60	*3.65*
256	1.1	1.5	3.36	3.30	3.29	3.32	3.34
256	1.2	1.5	3.29	3.33	3.37	3.39	3.40
256	1.1	2.0	3.32	**3.25**	**3.25**	3.29	3.32
256	1.2	2.0	3.29	3.35	3.42	3.46	*3.50*

the processors still having packets attempt to send them too rarely.

Choosing the right values for parameters for routing is very important especially in the "tail" phase. Choosing too small values for h_0, δ, d or t_{max} can cause a livelock situation or considerable slowdown to routing. Too small pameters mean also high standard deviation.

On the other hand, too big values for parameters will slowdown the routing, because there are many steps, when the processors do not attempt packet transmission.

Finding the optimal values for parameters is difficult, because the change of one parameter has effects the optimal values of the other parameters. Parameters should be chosen so that most the of the processors will succeed in reducing a $(1 - e^{-1/t})^i h$-relation to a $(1-e^{-1/t})^{i+1}h$-relation during one phase. When too small δ is chosen, many processors fail to achieve this goal, and then large h_0 is required to avoid livelock. Thus, in order get a

Table 8: Finding best value for t_{max} in GT algorithm.

h	d	1.25	1.5	1.75	2.0
8	1.1	**6.35**	6.85	7.07	7.13
8	1.2	6.65	6.88	7.32	*7.40*
16	1.1	**4.93**	5.16	5.30	5.42
16	1.2	5.07	5.22	5.43	*5.54*
32	1.1	**4.26**	4.28	4.33	4.39
32	1.2	4.27	4.31	4.43	*4.48*
64	1.1	*3.98*	3.80	**3.79**	3.80
64	1.2	3.81	3.81	3.85	3.88
128	1.1	*3.83*	3.53	3.48	**3.47**
128	1.2	3.49	3.51	3.54	3.55
256	1.1	*3.52*	3.29	**3.26**	**3.26**
256	1.2	3.32	3.37	3.40	3.42

good result, a compromise between a combination of small δ, large h_0, and a combination large δ, small h_0, must be found.

With high values of h the cost becomes lower. It approaches to e, because the probability of a single successful transmission approaches to $1/e$ and expected number of steps required approaches to $h * e$. The amount of extra time steps needed (the number of time steps required to send all packets - $h * e$) approaches to 0, when h approaches to infinity. Also the length of tail increases very slowly, when h increases.

Future work is to find a method for calculating optimal values for h_0, δ and t_{max}, when p and h are known.

The cost of routing can be as low as 3.3, when the number of processors is 1024 and $h = 256$. This is better than the Penalty algorithm of [9] (3.9) and the GGT algorithm of [4, 10] (5.6). In all tested cases, the GT algorithm seems to be better than the Penalty algorithm or the GGT algorithm.

Acknowledgements

I would like to thank professor Martti Penttonen and Dr. Ville Leppänen for discussion and ideas.

References

[1] R. J. Anderson and G. L. Miller. Optical communication for pointer based algorithms. Technical Report CRI-88-14, Computer Science Department, University of Southern

California, LA, 1988.

[2] A. Czumaj, F. Meyer auf der Heide, and V. Stemann. Shared memory simulations with triple-logarithmic delay. *Lecture Notes in Computer Science*, 979:46–59, 1995.

[3] M. Dietzfelbinger and F. Meyer auf def Heide. Simple, efficient shared memory simulations. In *Proceedings of the 5th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 110–119, Velen, Germany, June 30–July 2, 1993. SIGACT and SIGARCH. Extended abstract.

[4] M. Geréb-Graus and T. Tsantilas. Efficient optical communication in parallel computers. In *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 41–48, San Diego, California, June 29–July 1, 1992. SIGACT/SIGARCH.

[5] L. A. Goldberg, M. Jerrum, T. Leighton, and S. Rao. A doubly logarithmic communication algorithm for the completely connected optical communication parallel computer. In *Proceedings of the 5th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 300–309, Velen, Germany, June 30–July 2, 1993. SIGACT and SIGARCH.

[6] L. A. Goldberg, Y. Matias, and S. Rao. An optical simulation of shared memory. In *Proceedings of the 6th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 257–267, Cape May, New Jersey, June 27–29, 1994. SIGACT and SIGARCH.

[7] R. J. Jenkins Jr. Isaac and rc4. http://ourworld.compuserve.com/homepages/bob_jenkins/isaac.htm, 1993.

[8] R. K. Karp, M. Luby, and F. Meyer auf der Heide. Efficient PRAM simulation on a distributed memory machine. *Algorithmica*, 16(4/5):517–542, October/November 1996.

[9] A. Kautonen, V. Leppänen, and M. Penttonen. Simulations of PRAM on complete optical networks. *Lecture Notes in Computer Science*, 1124:307–310, 1996.

[10] A. Kautonen, V. Leppänen, and M. Penttonen. Constant thinning protocol for routing h-relations in complete networks. *Lecture Notes in Computer Science*, 1470:993–998, 1998.

[11] A. Kautonen, V. Leppänen, and M. Penttonen. Thinning protocols for routing h-relations in complete networks. In *Proc. of International Workshop on Randomized Algorithms*, pages 61–69, Brno, 1998.

[12] L. G. Valiant. General purpose parallel architectures. In *Handbook of Theoretical Computer Science, Ed. Jan van Leeuwen, Elsevier and MIT Press*, volume 1. Elsevier Science, 1990.