Optimum Transputer Configurations for Real Applications requiring Global Communications

Colin J. Burgess Alan G. Chalmers

Department of Computer Science,
University of Bristol,
Bristol, BS8 1TR,
England.

Abstract

If complex problems are to be solved in reasonable computation times, then large scale parallel processing is necessary. For many of these problems, the density of the global communications dominates the performance of the parallel implementation. In these cases, the design of the interconnection network for the processors is known to play a significant part in the efficient implementation of problems on large T800 transputer systems. This paper presents a new genetic algorithm for generating optimal configurations, augmented by simulated annealing for selected refinement of difficult cases. These configurations have the further advantage that they satisfy the best known criteria for producing configurations that perform well on real applications. The paper concludes by describing the impact this might have on the design of future T9000 transputer configurations.

1 Introduction

Parallel processing offers the potential for solving many complex science and engineering problems in a fraction of the sequential processing time, by the addition of more and more processors. The parallel solution of a problem on a multiprocessor system requires the individual processors to communicate, the pattern and frequency of the communication required being determined by the nature of the problem. When a problem with a global communication pattern is implemented, every processor will need to communicate with the all other processors and the overheads that result from this global communication increase as more processors are added. There are a large number of problems, from a wide variety of disciplines, which have these global communication requirements, for example: free Lagrangian codes; boundary element methods; radiosity and spectral methods in graphics; and the modelling of molecular interactions.

Some of the best performance figures for problems requiring global communication running on T800 transputers have been achieved using irregular configurations [7, 9]. The design of these networks requires finding an optimal configuration over a very large search space. In this paper, configurations are reported that are extremely close to the theoretically limits, and have been produced using a simple genetic algorithm, and in some cases, the additional use of simulated annealing.

2 Theoretical limits on optimal configurations

Each processor in a static distributed memory multiprocessor system has a number of links available for interconnection with other processors within the system. If this number of links is not restricted then the system may be fully interconnected. In practice, however, processors have a limited number of links available, which is four for the T800 transputer.

There are a wide variety of configurations that have been proposed for the interconnection of T800 transputers. Two important metrics for measuring the effectiveness of these configurations are diameter and average interprocessor distance. The diameter, D_{Diam}, is the largest number of links in the shortest path connecting any two processors in the network. The average interprocessor distance, D_{Avg}, is the average number of links that occur in the shortest paths between any two processors. Thus the average interprocessor distance is equivalent to the average number of links a message has to cross from any source processor to its desired destination processor.

It is important to show how the maximum number of processors that can be interconnected in a configuration of a certain diameter can be derived [6, 8]. An upper bound on this value may be obtained from extremal graph theory [4]. Assuming that every processor in the configuration has the same number of links (or valency), Δ, the upper bound on the maximum number of processors in the configuration, usually called the *Moore bound*, can be found by counting the number of processors which are at distances of 0, 1, 2, ..., D_{Diam} links from a given processor. This gives at most:

$$1 + \Delta + \Delta(\Delta - 1) + \Delta(\Delta - 1)^2 + \ldots + \Delta(\Delta - 1)^{D_{Diam}-1} \qquad (1)$$

processors for a diameter D_{Diam}. Graphs achieving this bound are called Moore graphs. Biggs [3] shows that (except for D_{Diam}=1 or Δ=2), Moore graphs can exist only for the cases: D_{Diam}=2 and Δ=3, 7, or 57; but in all other cases, graphs can be found that approach these bounds.

In order to provide a useful parallel processing platform, a multiprocessor system must have access to Input/Output facilities. Most systems achieve this by designating one processor as the *system controller* with the responsibilities of providing this Input/Output interface with the other processors, the *processing elements*, performing the actual processing of the problem. The inclusion of a system controller within a configuration may be achieved either by the provision of an additional link to one or more processors or by using some of the existing links.

For practical transputer configurations, two existing links from the configuration are used for connection to the system controller, since as the number of links per processor is even, using one or three links would leave another link unconnected. In such a configuration of n processors, *two* processors will have three links available and $(n-2)$ processors will have four links available for interconnection with the other processors. This results in a smaller upper bound on the maximum number of processors that can be obtained by considering the number of processors at different distances from a processor connected to the system controller. Since

such a processor has $(\Delta - 1)$ links available for interconnection with other processors, this gives:

$$1 \; + \; (\Delta - 1) \; + \; (\Delta - 1)^2 \; + \; \ldots \; + \; (\Delta - 1)^{D_{Diam}} \tag{2}$$

processors.

It is possible to consider an optimal configuration of n processors as a combination of two nodes containing the system controller connections and $n - 2$ other nodes. The two nodes can then be considered to be connected to all the other nodes by a spanning tree with a root node of valency 3, as represented by formula (2), and the other $n-2$ nodes by a spanning tree with root node of valency 4 , as represented by formula (1). It is then possible to calculate the average interprocessor distance, D_{Avg}, of this configuration, and the number of shortest paths which are equal to the diameter, P_{Diam}. These values, which are the theoretical limits for optimal configurations for various numbers of processors, are included in tables 1 and 2.

3 Generating algorithm

The genetic algorithm, developed by Goldberg [12] and also by Holland [14], is a powerful directed search technique which is loosely based on biological models of evolution. It is extremely robust in the face of conflicting information. Genetic algorithms may be used to optimise members of a population of structures, where the structures are encoded in a 'gene string'. The population of structures is evolved in a manner which is analogous to a naive view of biological evolution. In this case the determinant of evolution is the 'fitness' of each structure as measured by an appropriate fitness function. During the evolution of the population, strings which do well in terms of the fitness function are more likely to be selected for the 'breeding' process which creates new, more 'fit' populations.

3.1 Network Representation

The method for representing the network has a considerable impact on the efficiency and degree of success with genetic algorithms. The network of processors is represented by a single array of integer values. Every link from a processor is given a different numeric value, called its link number. For the convenience of coding, the n processors in an n node network are numbered from 0 to $n - 1$. Thus the link numbers for processor 0 are 0, 1, 2 and 3; for processor 1 they are 4, 5, 6 and 7 etc. The main array has $4n$ integer elements, each element representing a link from a processor. A link between two processors is represented by two of the array elements containing one another's link numbers. For example, if processors 1 and 3 are connected via link numbers 5 and 12, then link[5] = 12 and link[12] = 5. This representation is the same as used by Greenwood [13], except that the nodes are not preconnected by a minimum spanning tree.

A complete network that is normally, but not necessarily, fully connected, is formed by

systematically filling all the array elements with randomly selected values from the links that are not already connected, taking care to avoid connecting a node to itself. The two connections to the system controllers are links 0 and $4n-1$, to the processors 0 and $n-1$ respectively. These two array elements were initialised to point to themselves and were not allowed to be changed subsequently. There are several advantages to this representation.

1. It is easy to arrange for two links to be swapped.
2. After every swap the array represented a valid network, other than occasionally it was not fully connected.
3. Although not exploited in this paper, this representation is easily applicable to processors containing any number of links and to any number of connections to one or more system controllers.

3.2 Genetic algorithm

The optimising algorithm is simple and is based on randomly swapping two links and replacing the network with the new one, provided the value of the corresponding fitness function is increased. There is no crossover or other operator used, unlike many other genetic algorithms. The only precaution against getting stuck in a poor local minimum is to start with a population of independently generated random networks. This whole algorithm could be viewed either as a simple genetic algorithm with mutation within a string of integers as the only allowable operator, or as a discrete analogue of the conventional hill-climbing algorithms for continuous functions.

There is no particular constraint, either by the algorithm or by the representation of the network, on the way the fitness function is formulated. It was shown by Sharpe *et al* [17] that the minimum diameter metric was more important than the average interprocessor distance for real applications and thus the following fitness function f was used:

$$f = w_1 * D_{Diam} + w_2 * l + w_3 * P_{Diam}$$

where D_{Diam} is the diameter of the configuration, l is the total length of all the minimum length paths connecting nodes, P_{Diam} is the number of interprocessor distances with length d_{Diam}, and w_i for $i = 1, 2, 3$ are constant weights. This gives an integer value for the fitness function. The average interprocessor distance of the configuration is then given by $l/(n^2)$.

Since it is important to ensure that the minimum diameter dominates the value of the fitness function, w_1 is kept large. Within this constraint, two different functions $f1$ and $f2$ are used, where $f1$ optimises with respect to average processor distance and f2 with respect to P_{Diam}.

Thus for $f1$: $w_1 = 1000000$, $w_2 = 1000$ and $w_3 = 1$

and for $f2$: $w_1 = 1000000$, $w_2 = 1$ and $w_3 = 1000$

When evaluating the fitness function, successively higher diameter configurations are constructed until all the nodes are connected. Thus networks that are not fully connected gain a very high diameter, chosen to be 9, which ensures that such configurations never survive.

The choice of which two links to swap is made at random subject to the following constraints, which are to avoid generating networks which could not possibly be an improvement:

1. No two processors, of the four processors involved in the two links, can be the same.
2. The system controller links can not be moved.

The combined effect of these decisions is that the vast majority of the computation time is spent in evaluating the fitness function. It was found that function $f1$ in general gave the better results even for the minimisation of P_{Diam} values, probably because the values of P_{Diam} can vary considerably between two successive generations, whereas D_{Avg} changes more smoothly.

3.3 Simulated annealing algorithm

In a few cases, the actual configuration produced was very near to a solution with a lower diameter, that is P_{Diam} in the above formula was 1 or 2. However, it was proved that these solutions could not be improved by only swapping one pair of links, so they constitute local minima in the discrete space. Inspired by the simulated annealing approach to optimisation, the best solution so far, that is the minimum P_{Diam} for a given diameter, is chosen, and a whole population of new configurations produced. Each of the new configurations was formed by performing 8 different swaps, at random, of pairs of links of the local optimum solution. The value of 8 was chosen as a compromise to try and give a good boost away from a local minimum without destroying the basic quality of the solution. The genetic algorithm was then applied to this population. Normally, this produced a solution that had a value of P_{Diam} that was one or two smaller than the previous local minima and this enabled even more networks of lower diameter to be obtained.

4 Results

The results will be presented in two parts. The first examines the problem in detail as the number of nodes being considered approaches a Moore limit, and the second compares the results achieved with the theoretical limits, for various numbers of processors.

4.1 Approaching the Moore limit in practise

	Results Achieved			Theoretical Limit		
Processors	D_{Diam}	P_{Diam}	D_{Avg}	D_{Diam}	P_{Diam}	D_{Avg}
32	3	247	2.297	3	244	2.291
33	3	271	2.318	3	269	2.312
34	3	296	2.337	3	293	2.332
35	3	322*	2.356*	3	319	2.351
36	3	349	2.374	3	346	2.368
37	4	4	2.396	3	374	2.386
38	4	6	2.414	3	403	2.402
39	4	8	2.431	3	433	2.417
40	4	15*	2.456*	3	464	2.431

Table 1: Approaching the Moore limit

Table 1 shows the results obtained as the Moore limit of 40 processors was approached. It can be seen that for this particular case of diameter 3 configurations, where the Moore limit is 40 processors, the problem gets harder the nearer the Moore limit. However, the algorithms previously described have produce diameter 3 networks for 33 - 36 processor networks, which does not appear to have been reported before. They also failed only by a small number of paths for 37 - 40 processors. The asterisk against some results in this and later tables indicates those results that were obtained using simulated annealing in addition to the genetic algorithm.

By doing a comparison between the results obtained and the theoretically optimal solution, particularly for the values of P_{Diam} and D_{Avg}, it can be seen that the solutions for 32 -36 processors are extremely close to the theoretical limit. Viewing this research from the genetic algorithm viewpoint, it shows the power of genetic algorithms in tackling an optimisation problem with a very large search space.

4.2 Comparison with the theoretical limits

The genetic algorithm described was run for a number of networks containing different numbers of processors. A wide selection of the results can be found in table 2. These results are better than any previously published results with respect to both minimum diameter, average distance for a given minimum diameter, and comparable to the best results for minimum average distance irrespective of the diameter.

As far as the authors are aware, this is the only genetic algorithm approach that has achieved a diameter 4 for 64 processors, which was also achieved by another method [8], and is the only paper to report having found a diameter 6 configuration 256 processor networks.

The computation times required varied considerably with the number of processors required and the number of different initial configurations processed. In order to obtain good solutions, this required more members in the initial population the nearer the number of

	Results Achieved			Theoretical Limit		
Processors	D_{Diam}	P_{Diam}	D_{Avg}	D_{Diam}	P_{Diam}	D_{Avg}
12	2	43	1.514	2	43	1.514
13	2	53	1.550	2	53	1.550
14	3	2	1.602	3	1	1.592
32	3	247	2.297	3	244	2.291
36	3	349	2.374	3	346	2.368
40	4	15*	2.456*	3	464	2.431
53	4	171	2.694	4	13	2.578
64	4	492	2.885	4	365	2.821
128	5	899	3.523	5	7	3.409
256	6	1505	4.192	5	12200	4.154

Table 2: Comparison of results with theoretical limits

processors was to a Moore limit. As a guide, for values above 30 processors and well below the Moore limit, a population of 40 configurations was used for networks up to 64 processors and the best solutions were found after 10 to 40 thousand swaps on each configuration, that is up to about a million evaluations of the fitness function. Unfortunately the time to evaluate the function is of the order n^3 to n^4, so for the larger networks only a population of 10 configurations was used. Of course, these networks only ever need to be generated *once* in order that they may be used as the means for configuring multiprocessor systems and thus the computation time is not so important.

When optimising a network that is near a Moore limit, or when using simulated annealing, a larger population of up to 400 initial configurations was used. However, despite starting with a population of 2000 configurations, it was still not possible to get a diameter 3 solution for 37 or more nodes. However, he Moore limit is a theoretical limit and thus gives no guarantee that diameter 3 solutions exist for 37-40 processors.

A number of completely different techniques have been proposed to obtain graphs with similar properties to these configurations. For example, Toug and Steiglitz [18] proposed a local search algorithm for finding graphs with small diameters. In their paper, the graphs studied by Trufanov [19] are used as the initial graphs and then a set of perturbations are carried out in order to try to achieve graphs with a lower diameter. Prior *et al* [16] used Hamiltonian graphs as their initial graph and then used genetic algorithms to "search" for an improved graph. These techniques do not produce exactly what is required for practical configurations and, as insufficient detail is presented in these papers, it is difficult to compare them in detail with the methods used here.

5 Existing Configurations

In this section we compare the results of AMP configurations generated by a variety of methods and some other configurations which are often cited in the literature. The AMP configura-

tions make use of existing links to include the system controller, while the other configurations use additional links.

Table 3 gives the diameters of a number of processors arranged in different configurations. The row labelled AMP_{GAS} shows the results that have been achieved using the simple genetic algorithm presented in this paper. AMP_{GA} and AMP_{GAcSP} show previous results achieved using genetic algorithms as presented in [17] and [21] respectively. The final row of AMP configurations, labelled AMP_{DFS}, are those obtained using a heuristic depth-first search strategy implemented in Prolog as presented in [8]. Diameters of other well known configurations are also shown in the table for comparison.

	Processors							
	8	13	16	32	40	53	64	128
AMP_{GAS}	2	2	3	3	4	4	4	5
AMP_{GA}	2	2	3	4	4	4	5	6
AMP_{DFS}	2	2	3	3	4	4	4	5
AMP_{GAcSP}	-	-	-	4	-	-	-	-
Hypercube	3	-	4	5 †	-	-	6 †	7 †
Torus	3	-	4	6	7	-	7	12
Ternary Tree	4	4	5	6	6	8	8	10
Mesh	4	-	6	10	11	-	14	22
Ring	4	6	8	16	20	26	32	64
Chain	7	12	15	31	39	52	63	127

Table 3: Comparison of configuration diameters

	Processors							
	8	13	16	32	40	53	64	128
AMP_{GAS}	1.28	1.55	1.66	2.30	2.46	2.69	2.88	3.52
AMP_{GA}	1.28	1.55	1.69	2.30	2.47	2.72	2.92	-
AMP_{DFS}	1.28	1.55	1.73	2.31	2.53	2.76	2.92	3.58
AMP_{GAcSP}	-	-	-	2.47	-	-	-	-
Hypercube	1.50	-	2.00	2.50†	-	-	3.00†	3.50†
Torus	1.50	-	2.00	3.00	3.50	-	4.00	5.64
Ternary Tree	1.97	2.56	2.91	3.93	4.25	4.77	5.01	6.25
Mesh	1.75	-	2.5	3.88	4.23	-	5.26	7.94
Ring	2.00	3.23	4.00	8.00	10.00	13.25	16.00	32.00
Chain	2.63	4.31	5.31	10.67	13.99	17.66	21.33	42.66

Table 4: Comparison of average interprocessor distances

Note that the values marked with a † in the table for the hypercube for more than 16 processors cannot be compared directly as, unlike the other configurations, they assume more than four links are available on each processor.

6 Implications for T9000 networks

The next generation of parallel processors, such as the T9000 transputer [**?**], are now appearing on the market. These new processors have moved towards being able to construct a system which allows dynamically reconfigurable networks based on the concepts of *virtual links* and *wormhole routing*. Virtual links provides the routing requirements of a multiprocessor configuration in the hardware, thereby removing the need for this to be included in the system software. Virtual links, also known as logical links, may be established between any two processes within the system and the hardware router will be responsible for ensuring that messages reach their destinations. Although virtual links over reconfigurable physical links appears to offer full connectivity between processors and even processes, such a scheme is not without problems. Despite the fair manner used for interleaving messages on to the physical links, multiplexing large numbers of virtual channels on to a few physical channels is still likely to lead to serious contention under heavy message loading. Removing the burden of having to write the multiplexing software from the programmer is liable to aggravate the problem, as he or she is less likely to be aware of the problems of this contention and thus create virtual links with "gay abandon". Proposed automatic placement of processes on to processors may help, but even this will still depend on the ratio of virtual links to physical links. Certainly, with hardware routing, the efficiency of the individual processors should be enhanced, which will improve the scalability of a given problem on a multiprocessor system. Limiting the number of processors that can be connected to a single routing chip means that for large numbers of processors, a number of these routing chips will have to be joined together themselves in larger networks.

The best method for designing the configuration for a large number of T9000 processors with associated routing chips is not at all obvious and does not directly follow from T800 configurations. As a result of an initial study, based on the limited technical data available so far [**?**], it would seem that additional metrics may well be needed in order to produce optimal designs. It would also appear that there are a number of good networks for a given number of T9000 processors, each based on a differing number of C104 routing chips, and differing only in the message bandwidth between processors, rather than the diameter or average interprocessor distance.

As an example, it would appear that a diameter 2 network for 1040 T9000 processors could be built using 260 C104's in the following way:

1. Construct a basic building block of 16 T9000's, with one link from each T9000 being connected to each of 4 C104 routing chips. This gives a network of 16 processors with minimum distance of 1 routing chip between any two processors. The whole building block then has 64 free links.

2. Connect each building block to a different block of the same type, giving a configuration of diameter 2, using 65 building blocks and $65 \times 16 = 1040$ processors and 260 routing chips.

Without a lot more experience, it is impossible to say if this would be a sensible or useful

network, but if the bandwidth between processors became a limiting factor, then the number of links between different building blocks could be increased and the total number of processors reduced, until, in the limit, you just had two building blocks, with 16 connections between them and thus a network of 32 processors and 2 C104's. Alternatively, it is also possible to build networks based on 4 layers, each layer involving one connection to a processor. If very high global communication is required, involving probably a smaller ratio of processors to routing chips, then it could be that the most optimal configuration, based perhaps on some measure of average interprocessor bandwidth, could be achieved using an irregular configuration, as is the case for T800 networks.

The optimal design of networks for large numbers of T9000 processors seems to be an open research area, requiring a combination of theory, design and, in due course when such systems can be built, verification using real applications.

References

[1] D. P. Agrawal and G. C. Pathak. Evaluating the performance of microcomputer configurations. *IEEE Computer*, 19:23–37, 1986.

[2] J. C. Bermond, C. Delorme, and J. J. Quisquater. Strategies for interconnection networks: some methods from graph theory. *Journal of Parallel and Distributed Computing*, 3:433–449, 1986.

[3] N. Biggs. *Algebraic Graph Theory. Cambridge Tracts in Math. No. 67*, Cambridge University Press, London, 1974.

[4] B. Bollobás. *Extremal Graph Theory.* Academic Press, London, 1978.

[5] A. G. Chalmers. A Minimum Path system: a communication-efficient system for distributed-memory multiprocessors. *South African Journal of Science*, 89:175–181, Apr. 1993.

[6] A. G. Chalmers. *A Minimum Path system for parallel processing.* PhD thesis, University of Bristol, Department of Computer Science, Aug. 1991.

[7] A. G. Chalmers, S. Fiddes, and D. J. Paddon. Parallel panel methods. In H. S. M. Zedan, editor, *13th Occam Users Group conference*, pages 313–321, IOS Press, York, 1990.

[8] A. G. Chalmers and S. Gregory. Constructing minimum path configurations for multiprocessor systems. *Parallel Computing*, 19:343–355, Apr. 1993.

[9] A. G. Chalmers and D. J. Paddon. Parallel radiosity methods. In D. L. Fielding, editor, *4th North American Transputer Users Group*, pages 183–193, IOS Press, Ithaca, NY, Oct. 1990.

[10] L. Davis, editor. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, 1991.

[11] W. Dzwinel. The search for an optimal multiprocessor interconnection network. *Parallel Computing*, 17:95–100, Apr. 1991. Short Communication.

[12] D. E. Goldberg. *Genetic Algorithm in search, optimisation and machine learning.* 1989.

[13] A. Greenwood. *Parallel Genetic Algorithms for AMP configurations.* Master's thesis, University of the West of England, Bristol Transputer Center, Dec. 1992.

[14] J. H. Holland. *Adaptions in natural and artificial systems.* University of Michigan Press, Ann Arbor, 1975.

[15] M. Maekawa. Optimal processor interconnection topologies. *ACM SIGARCH*, 9:171–185, May 1981.

[16] D. Prior, N. Radcliffe, M. Norman, and L. Clarke. *What Price Regularity?* Technical Report ECSP-TR-3, Edinburgh Concurrent Supercomputer Project, Edinburgh University, Jan. 1989.

[17] P. Sharpe, A. Chalmers, and A. Greenwood. Genetic algorithms for generating minimum path configurations. *Microprocessors and Microsystems*, 1994. To appear.

[18] S. Toueg and K. Steiglitz. The design of small-diameter networks by local search. *IEEE Transactions on Computers*, 28(7):537–542, July 1979.

[19] S. V. Trufanov. Some problems of distance on a graph. *Eng. Cybern*, 60–66, 1967.

[20] C. von Conta. Torus and other networks as communication networks with up to some hundred points. *IEEE Transactions on Computers*, 33(7):657–666, 1983.

[21] T. Warwick and E. P. K. Tsang. *Using a Genetic Algorithm to tackle the processor configuration problem.* Technical Report CSM-190, University of Essex, Apr. 1993.

[22] L. D. Wittie. Communication structures for large networks of microcomputers. *IEEE Transactions on Computers*, 30(4):264–273, 1981.

Ver 0.9 4/11/94 As sent to conference