Placement of Communicating Processes on Multiprocessor Networks

Craig S. Steele

Computer Science
California Institute of Technology

5184:TR:85

Placement of Communicating Processes on Multiprocessor Networks

by
Craig S. Steele

In Partial Fulfillment of the Requirements
for the Degree of Master of Science

Computer Science Department
California Institute of Technology
Pasadena, CA 91125

Technical Report 5184:TR:85

April 1985

The research described in this report was sponsored by the Defense Advanced Research Projects Agency, ARPA Order number 3771, and monitored by the Office of Naval Research under contract number N00014-79-C-0597

Contents

List of Figures

Chapter 1

Introduction

My mynd ben all binsy with myndy thinking. Thinking who wer going to do what and how I myt put some thing to gether befor some 1 else done it.

Riddley Walker, Russell Hoban

1.1 Motive

For a single computation to be executed concurrently by a machine with multiple processing nodes, the problem must be divided into parts, and those parts must be placed onto the physical nodes. The first task is here referred to as *logical partitioning*; the second as *physical placement*. In the model described here, the *logical partitioning* is assumed to have been done by the programmer, who has separated the computation into a set of communicating *processes* [13]. These processes communicate by passing messages via *logical channels* that may link any pair of processes, and are also defined by the programmer. If the machine does not have a global communication system, such as shared memory or a common bus, the assignment of *logical processes* to *physical nodes* will affect the pattern and efficiency of communications.

The Caltech Cosmic Cube [13] is one such machine, composed of 64 small computers using point-to-point communications channels of intentionally limited speed and appreciable latency. The physical communications channels are arranged as the edges of a binary 6-cube; internode messages may require as many as 5 forwarding operations by intermediate nodes. For this or any other system without global communications hardware, it is desirable to place communicating logical processes on physical nodes in close proximity to reduce the cost of communications. The distance metric is defined by the communications structure.

In some cases, a natural assignment of processes to nodes may be evident. For example, the Cosmic Cube is isomorphic to a 4x4x4 three dimensional torus, and three dimensional problems with grid sizes which are multiples of 4 can be mapped readily. Irregular or ill-sized problems may not have obvious good assignments.

This paper presents a method for generating a good placement for any process and node structures, using the technique of simulated annealing [5].

1.2 The graph model

The logical partition of the problem may be represented as a graph. Each vertex of

this *logical graph* corresponds to a process; each edge represents a logical communication channel. Similarly the physical machine may be represented by a *physical graph* with vertices and edges representing processing nodes and physical communication channels, respectively. To run a computation the processes must be loaded onto the physical machine, requiring a mapping of the logical graph onto the physical graph.

1.3 Graph attributes

Each edge of the logical graph has an arbitrary weight, which is interpreted as the amount of communication between the two processes that are its endpoints. In this work, communication channels are assumed to be bidirectional, so the edges are undirected. Each vertex has a size that may be taken to represent the amount of memory it requires, expressed as a fraction of that available in a node.

Each edge of the physical graph represents the existence of a physical communication channel with a weight for the cost of message transmittal. Each node is assumed to have the same amount of memory. Both edges (communication channels) and vertices (processing nodes) can be given maximum I/O capacities.

1.4 The cost function

The mapping method attempts to find an embedding of the logical graph in the physical graph which minimizes the total cost function.

The cost function is the sum of four terms

$$cost = \sum_{logical\ channels} distance^{k_1} + c_{memutil} \sum_{nodes} memutil^{k_2} +$$

$$c_{nodeio} \sum_{nodes} nodeio^{k_3} + c_{chanio} \sum_{physical\ channels} chanio^{k_4}$$

Where

Distance is the physical distance separating logically connected processes.

Memory utilization is $\sum_{resident\ processes} memory\ size$, the total amount of memory occupied by processes on a node.

Node I/O cost is $\sum_{local\ routes} logical\ channel\ weight$, the total utilization of the processing node I/O capacity.

Channel I/O cost is $\sum_{local\ routes} logical\ channel\ weight$, the total utilization of the physical channel I/O capacity.

The summation $\sum_{local\ routes}$ for the last two terms is over the logical channels which are routed through that particular physical node or channel. The values of the coefficients c and exponents k depend on the definition of the particular problem. The coefficient of the distance term is implicitly 1; other terms are weighted relatively to it.

The first term is minimized by reducing the physical distances between communicating processes; it is analogous to an attractive force between processes. The second term is a repulsive force tending to prevent the nodes from becoming too full. The latter two terms are routing congestion terms that penalize inequitable communications loading of nodes and physical channels.

1.5 This work

Given a logical and a physical graph and parameters for a cost function, the program NET finds mappings of logical to physical graphs which approach minimum communications cost. The resultant mappings may be one-to-one (one process per node) or many-to-one (multiple processes per node), depending on the size of the processes. It is assumed that such static analysis of communications cost is adequate for many purposes. Computations with message traffic deviating only slightly from the average or with synchronized bursts of maximal communications activity are well modelled. Computations having significant temporal dependency will require more sophisticated analysis for good node load balancing.

1.6 Other applications

Though this work is motivated by the process placement problem, many other problems are readily transformed into equivalent graph embedding problems. With suitable interpretations for the graph attributes and suitable parameters for the cost function, the graph mapping program NET is widely applicable. These problems are in general NP-complete, so perfect solutions cannot be expected [3].

Data partition A slight variation of the process placement problem is partitioning data among processes to minimize interprocess communication, e.g., assignment of circuit elements to processes of a distributed logic simulator.

Logic partition Partitioning logic circuits between two packages to minimize interwiring is solved by mapping the logic graph onto a two-node physical graph with a connection cost exponent of zero.

Integer partition The integer partitioning problem is to divide a set of integers into two subsets so as to minimize the difference in the sums of each subset. It can be modelled by considering only the node utilization term of the cost function with a greater than linear exponent.

Hamiltonian circuit The problem of finding a Hamiltonian circuit in an undirected graph is the same as finding an optimal mapping of a ring of N vertices onto the given graph of N vertices.

Travelling salesman problem The travelling salesman problem can be defined as finding a minimal length mapping of an N-element ring onto a completely connected physical graph. In this case the physical graph edge weights are considered to be the distances between cities the salesman must visit.

Routing Restricting channel and node capacity to single logical connections allows solution of routing problems.

1.7 Overview of this report

- Chapter 2 describes the general optimization technique of simulated annealing.
- Chapter 3 describes the program NET, which implements a simulated annealing technique for graph embedding problems.
- Chapter 4 describes a comparative study of various medium-scale physical interconnection structures for multinode machines.
- Chapter 5 provides additional analysis of the comparative study, including derivation of a lower bound for the typical interprocess distance, and a discussion of a comparative study with physical graphs scaled to larger size.
- Chapter 6 reviews the results and proposes future research topics.

Chapter 2

Simulated annealing

2.1 Introduction

Many optimization problems of interest to computer scientists are NP-complete; that is, all methods for finding the 'best' solutions require calculations which are exponential in the size of the problem. For instance, the problem of finding the best one-to-one mapping of n processes to n nodes by exhaustive search will require all $n!$ possible configurations to be examined. This 'combinatorial explosion' of the number of possible states makes finding exact solutions for many large problems impossible. However, for practical purposes good approximations to the optimal solutions may suffice. The technique employed in this report, simulated annealing, is a method for finding such approximations.

> [Simulated annealing is] a general framework for optimization which uses computer simulation methods from condensed matter physics and an equivalence (which can be made rigorous) between the many undetermined parameters of the system being optimized and the particles in an imaginary physical system. [6]

2.2 Improving iterative improvement

Simulated annealing is a variation of the heuristic strategy of iterative improvement. Iterative improvement requires :

- a concise representation of the system of interest.
- a scalar objective function (e.g. cost or energy) that reduces the objectives of the optimization process to a single number and quantifies the trade-offs between conflicting objectives.
- a procedure for generating trial rearrangements of the system [6].

From an initial configuration, successive improvements are made by examining the cost of trial rearrangements and accepting those with lower cost. The process will continue to reduce the system cost until all of the trial configurations subject to examination are of higher cost. Iterative improvement will find a solution which is minimal within the search space of the rearrangement procedure. The search process will become 'stuck' at this local minimum, unable to reach other, possibly better, solutions.

2.2.1 Avoiding local minima

The problem of failure to locate global minima may be addressed by generating several solutions from randomly selected starting points. Alternatively, upon detection of a local minimum, a new point, near the current state but outside the local search region, may be chosen for further iterative search.

2.3 Simulated annealing

Simulated annealing adds the concept of a simulated 'temperature' to iterative improvement.

2.3.1 Thermal noise

Iterative improvement will always accept a new configuration of lower cost and reject more costly states. Simulated annealing escapes from local minima by sometimes accepting higher cost rearrangements, with a probability determined by the incremental cost and the simulated 'temperature'.

> This temperature is simply a control parameter in the same units as the cost function. The simulated annealing process consists of first 'melting' the system being optimized at a high effective temperature, then lowering the temperature by slow stages until the system 'freezes' and no further changes occur. At each temperature, the simulation must proceed long enough for the system to reach a steady state. The sequence of temperatures and the number of rearrangements ... attempted to reach equilibrium at each temperature can be considered an annealing schedule. [5]

If the initial temperature is high enough, and the cooling slow enough, the global minimum of the system will be reached.

Iterative improvement is equivalent to simulated annealing with the temperature $T = 0$. The physical analogy is 'quenching', extracting energy from a hot system as quickly as possible. Alternatively, simulated annealing is a probabilistically directed iterative improvement scheme.

2.3.2 Acceptance function

The probability of accepting a trial configuration differing by energy $\triangle E$ from the present state is denoted $P(\triangle E)$. Two probability functions from statistical mechanics are common in simulated annealing. Kirkpatrick [5] used the *Boltzmann* probability factor,

$$P(\triangle E) = min(1, e^{\frac{-\triangle E}{kT}})$$

used by Metropolis *et al.* [9] for early molecular modelling simulations, shown in Fig. 2.1. This function accepts all changes of $\triangle E \leq 0$ with $P(\triangle E) = 1$. The *'heat bath'* function

$$P(\triangle E) = \frac{1}{1 + e^{\frac{\triangle E}{kT}}}$$

shown in Fig. 2.2, is claimed (on theoretical grounds) to have faster convergence to equilibrium [4]. It is assumed that the Boltzmann constant k equals 1 in order to avoid inconveniently large temperature scales.

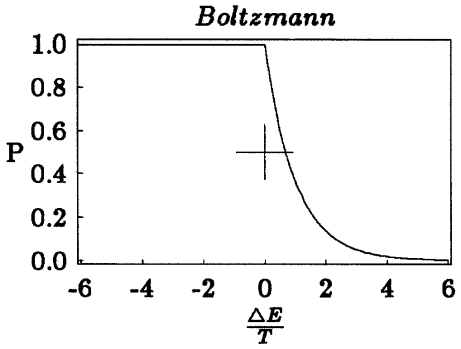

Figure 2.1: Boltzmann acceptance function

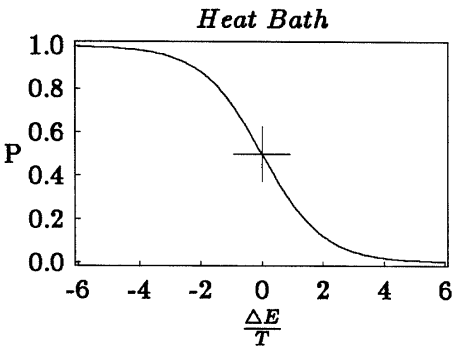

Figure 2.2: 'Heat bath' acceptance function

In my simulations I found no discernible difference between the two functions in either the speed or quality of results. The heat bath function was preferred because it accepts only 50 % of trial configurations of the same energy as the current configuration, which is somewhat more efficient and convenient in the simulation program.

Some quite unphysical acceptance functions (e.g., linear) have been evaluated with only minor differences reported [10]. Possibly even these differences can be attributed to effectively different annealing schedules.

2.3.3 Relevance of theory

The simulated systems differ significantly from physical systems in at least two ways. Simulations perform sequential rather than simultaneous evaluation of trial states and the types of possible rearrangements are relatively limited. Secondly, the number of elements is typically hundreds or thousands, tens of orders of magnitude smaller than the real systems modelled by statistical mechanics. Each state change can cause a significant disturbance of the entire system energy. In particular, the number of states near a global minimum ('ground states') is relatively small by thermodynamic standards and there are just a few distinct possible transition energies. These near minimal energy states are those of interest during optimization. Since the simulation is quite 'noisy', the exact form of the acceptance function is probably not critical. It may change the energy distribution of states at higher temperature, but almost any continuous function of the right general behavior could produce the desired low temperature result with the proper annealing schedule.

Stochastic formulation

An alternative to the thermodynamic formulation of the optimization problem is the stochastic or Markov formulation [7]. For a given temperature, a matrix of transition probabilities between states can be constructed, based on calculation of their energies. Since calculation of eigenvectors of matrices of $n!$ size is an intractable problem, this complementary approach is only of theoretical interest. The interesting problems of interest are of a middling size; too large for combinatorial search or Markov modelling; marginally small for statistical mechanics to be absolutely appropriate. The physical analogies are helpful for the development of intuition.

2.4 Issues in implementation

2.4.1 Cost function

Successful applications of simulated annealing are characterized by rapid evaluation of great numbers (thousands to millions) of possible configurations. For efficiency the cost function should be simple and additive so that small rearrangements in configuration require only local recalculation.

Terms with large relative weights will be better minimized. Terms intended as constraints must dominate in order to avoid degenerate solutions. For example, if processes are allowed to migrate without effective restriction, all processes will be put on one node to minimize communication distances. Both higher relative weighting of terms and larger exponents may be necessary to avoid such problems.

It is important to note that it is not the *total* energy contribution from energy components which determines correct weights, but the *variation* of those contributions during the simulation. A large energy term which never changes will have no effect on the acceptance probability $P(\triangle E)$.

2.4.2 Annealing schedule

The quality of the solution is critically dependent on the choice of starting temperature and cooling rates. The prescription [5] is to start from a very hot system, then cool by stages, continuing the simulation at each temperature until equilibrium is reached, then reduce the temperature by some factor. Determining equilibrium is quite difficult in practice, since it presumes knowledge of the system energy that corresponds to the current temperature. Reaching equilibrium at $T = 0$ corresponds to the problem of finding a global minimum.

Cooling too fast will result in inferior solutions; cooling too slowly wastes time. Most experimenters have produced annealing schedules either interactively or by specification of an exponential decay rate. More elaborate methods are possible; mine are discussed in section 3.7.

Phase transition

Not all temperature changes will result in significant change in energy. The rate of change of the system energy E with respect to the temperature T is

$$C(T) = \frac{d\langle E(T)\rangle}{dT},$$

by analogy, the *specific heat*. Temperatures with high specific heat correspond to phase transitions of the system. Simple, 'crystalline' systems may have only one phase transition, corresponding to 'freezing', (e.g. Figure 3.4 discussed in section 3.7.2). More complex systems may have a series of phase transitions at different temperatures.

Phase hierarchies

At a given temperature the only significant trial configurations are those for which $\triangle E \approx T$ [14]. Moves resulting in larger positive changes are unlikely to be accepted; those resulting in larger negative changes will probably have already been found and accepted at previous higher temperatures. New states of smaller magnitude are unlikely to persist.

Phase transitions occur when many trial states have $\triangle E \approx T$. If appropriate parameters for the cost function are selected, a temperature hierarchy for goals and constraints may be created. For example, if we wish to place 4 processes on each node, we may first reach a transition where all 6-process states are excluded, followed by the elimination of 5-process states as the temperature is lowered. (Note that a nonlinear function is required if $\triangle E$ is to differ for these two cases.) If a cost term has a phase transition at a temperature much higher than another of conflicting impetus, it is a constraint. If two terms have comparable characteristic temperatures, they are jointly minimized. The temperature scale creates a hierarchy of importance, since higher temperature phases are encountered first.

The cost of such flexibility is potential inefficiency. Some classes of trial moves may produce energy changes appropriate to only some temperature ranges. Many problems are best addressed by separate sequential optimizations rather than simultaneous optimization of several terms. An example is hierarchic logic partitioning [5]. First circuits are partitioned into packages, minimizing the number of interpackage wires; then the packages are place on circuit boards so as to minimize wiring length. In principle both optimizations could be done together, but the efficiency of the sequential division of the problem is much greater.

2.5 Legal solutions

The legality of the resultant configuration can be motivated by high penalties in the cost function, or guaranteed by the configuration generator, ('hard' or 'soft' constraints, respectively). Cost penalties are more flexible, but may require careful parameterization to produce legal results in tightly constrained or very asymmetric systems.

2.6 Power of the method

Kirkpatrick [6] and Johnson [10] have compared simulated annealing with other available heuristics for several NP-complete problems. In general, simulated annealing produces results of quality comparable to or better than the best previously known heuristic methods, but at greater computational cost. The generality of the method is dependent on the specialization of the trial configuration generator.

The graph embedding model of Chapter 1 allows a broad range of problems to be addressed. With a fixed move set and an automatic annealing schedule, simulated annealing is an excellent tool for comparative studies.

Chapter 3

The graph mapping program

NET is a program for finding good embeddings of a logical into a physical graph. It was inspired by the bipartite logic partitioning work done by Kirkpatrick [5]. The initial version of the program required special subroutines for each physical interconnection graph; later it was generalized to accept adjacency list representations of graphs. Though the program can be applied to a variety of problems, the terminology of process placement is used here.

The basic input to the program is three (optionally four) files that define the :

- Physical graph.
- Logical graph.
- Initial mapping of the logical into the physical graph.
- (Optional) precalculated shortest path distances of the physical graph.

Additional parameters specify the cost function and annealing schedule. The output is one or more mappings of the logical to physical graph and various summary statistics.

3.1 Input specification

Process and node numbers are assumed to be small integers, starting from 0, to allow compact arrays to be used as roots for dynamic data structures. Because both graphs must be undirected, the files must list directed edge connections in both directions for correct operation. Duplication of either logical or physical channels may produce suboptimal results for congestion calculations. Most data are checked for plausible values when read.

3.1.1 Physical graph

The physical graph file, which describes the physical communication channels among the nodes, has four components:

- Node number of one end (origin) of a physical communication channel.
- Node number of the other end (destination).
- The cost of the channel. Often this is set to 1, but could vary, e.g. it would represent distance in a travelling salesman problem.

- The channel capacity (in the same units as logical channel weight, below).

Physical graphs may be disconnected; if so, a warning will be issued during the calculation of shortest paths. Calculation of the cost function may cause a floating point overflow if logically connected processes are placed in disconnected parts of a physical graph.

3.1.2 Logical graph

The logical graph, which describes the logical communication channels among the processes, has three components:

- Process number of one end (origin) of a channel.
- Process number of the other end (destination).
- The weight of the channel, i.e. the volume of message traffic per unit time.

3.1.3 Initial mapping of the logical to physical graph

The initial mapping has three components:

- Process number.
- Node number on which the process is initially placed.
- Process memory size, expressed as a fraction of the (uniform) node memory size.

If the memory size is negative, the process is considered fixed and will not be moved.

If the initial placement is not close to a legal, properly constrained configuration, the estimation of parameters for the annealing schedule will take longer due to extension of the quenching phase. The quality of the final mapping is essentially independent of the initial mapping since the system is heated to randomness. Uniform distribution of processes has proven adequate.

3.1.4 Invalid placement

For comparative studies it is convenient to tolerate 'illegal' initial mappings. In each study the same initial mapping file is used for all combinations of logical and physical graphs. Logical graphs are not tailored to fit the various sizes of physical graphs. The initial mapping may therefore place processes on nodes which do not exist in the smaller physical graphs. For example, a binary 7-cube logical graph may be placed onto a binary 7-cube with an initial one-to-one identity mapping with no difficulty, since both have 128 vertices. If the same logical graph and mapping files are used with a binary tree of height 6 (127 vertices) as the physical graph, the highest-numbered process will be located on a nonexistent node. For the comparative study the expedient solution is just to remove (after a warning) logical graph vertices that are mapped to nonexistent physical vertices. Obviously, this Procrustean treatment is inappropriate for an actual computation where all processes are indispensable.

3.2 Major internal data structures

The main internal structures correspond to the four input files. The graph structures are contained in dynamically allocated Pascal records.

3.2.1 Physical graph

Physical node (vertex) records contain memory and node I/O utilization values. Physical communication channels (edges) of the physical graph are stored as an adjacency list. Three lists contain associated processes (logical vertices) which are in one of three states:

- Resident on the node, unaffected by the trial configuration.
- Resident on the node, moved elsewhere in the trial configuration.
- Not resident on the node, but moved there by the trial move.

3.2.2 Logical graph

The process record contains two adjacency lists recording logical communication channels (edges) that begin and end at that process. The latter list of 'backpointing' references is required for efficient calculation of congestion costs and would also be useful if NET were modified to allow directed logical edges.

3.2.3 Initial configuration

The initial configuration is copied to an internal file to be used for reinitialization if more than one solution is requested. The current placement is contained in the physical node record array.

3.2.4 Physical graph shortest path and shortest distances

A fixed-size array holds the matrix of minimum distances from all physical nodes to each other. For congestion calculations, it is additionally necessary to retain sufficient information to construct the actual shortest *paths*. Therefore, another array contains heads of lists of the next vertices in the (usually multiple) shortest paths.

3.3 Physical graph distance calculation

The distance and path matrices are generated by a version of Dijkstra's all-points shortest paths algorithm [1] of $O(\ |E_{physical}|\ |V_{physical}|\ \log |V_{physical}|\ )$ time complexity, (E : edges, V : vertices). This method is superior to the simpler Floyd $O(\ |V_{physical}|^3\ )$ algorithm [1] for large sparsely connected graphs, where $|E_{physical}| \approx average\ degree \times |V_{physical}| \ll |V_{physical}|^2$. If both node I/O and channel I/O utilization energy coefficients are zero, the physical paths for each logical connection are not generated.

The actual path selected by a logical channel is recorded in a list based in the channel record. As in other data structures affected by new configuration trials, both old and new configurations are maintained.

3.4 Generation of new configurations

The generation of a new configuration proceeds in several steps. Each process is sequentially selected to be part of a trial move. A randomly selected process residing on a randomly selected neighbor of the current node is chosen. An exchange of the pair of

processes is evaluated. If the exchange is rejected, a move of one process to the adjacent node is attempted; if that move also is rejected, the complementary move is tried.

If the memory utilization cost coefficient is zero, only exchanges are made. In this case the initial mapping must place legal numbers of processes (of equal size) on nodes, since those numbers will remain constant.

3.5 Calculation of the cost function

All energy calculations are done incrementally, recomputing only the components affected by the most recently accepted move, except during initialization and statistics gathering.

The restriction of input to undirected graphs allows the logical distance cost term to be calculated more efficiently by doubling the computed unidirectional $\triangle E$.

3.6 Acceptance function

Both the Boltzmann and 'heat bath' acceptance functions have been used, with minimally different results. As mentioned in section 2.3.2, a large comparative study found no evident difference between the two functions in convergence rates or result quality. The Boltzmann function has a significant disadvantage because all moves with $\triangle E = 0$ are accepted. Because of the sequential generation of alternative moves mentioned above, if every *exchange* move attempted is accepted, the *asymmetric* moves will never be considered. Very symmetric states will be erroneously stable. Therefore, the 'heat bath' acceptance function was preferred and used for all results presented here.

3.7 Annealing schedule

The automatic generation of a suitable annealing temperature schedule is essential for convenience and comparability of results. Various program options may be selected; the most automated choices are described.

3.7.1 Exponential temperature decay

The basic annealing schedule is produced by starting at a high randomizing temperature and repeatedly lowering the temperature by some ratio (usually .9) resulting in exponential decay. Each new temperature step will alter the probability of acceptance of a given $\triangle E$ by a small amount.

Figure 3.1 shows the 'heat bath' acceptance function for a fixed $\triangle E$ with T' shown for various decay ratios. As the temperature decreases, the curve approaches a step function. The 'physical' implication is that as the temperature drops, simulated annealing increasingly resembles iterative improvement; accepting all moves which decrease energy and rejecting all which increase energy. Figure 3.2 shows the relative change in acceptance probabilities for a given $\triangle E$ due to a temperature reduction by the indicated ratio. This plot is obtained by dividing the T' curves of Figure 3.1 by the $T' = T$ base curve. An examination of the two figures shows that the difference in effect between the two commonly employed ratios of .9 and .95 is quite modest. Only for very unfavorable moves ($\frac{\triangle E}{T'} \gg 0$) is the

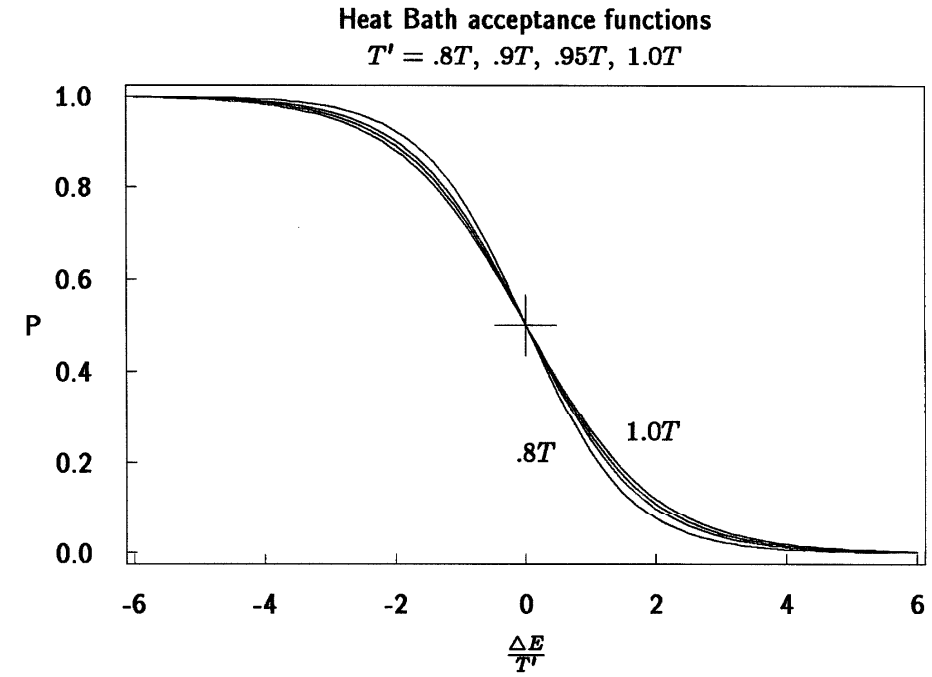

Figure 3.1: Acceptance probabilities for various decay ratios

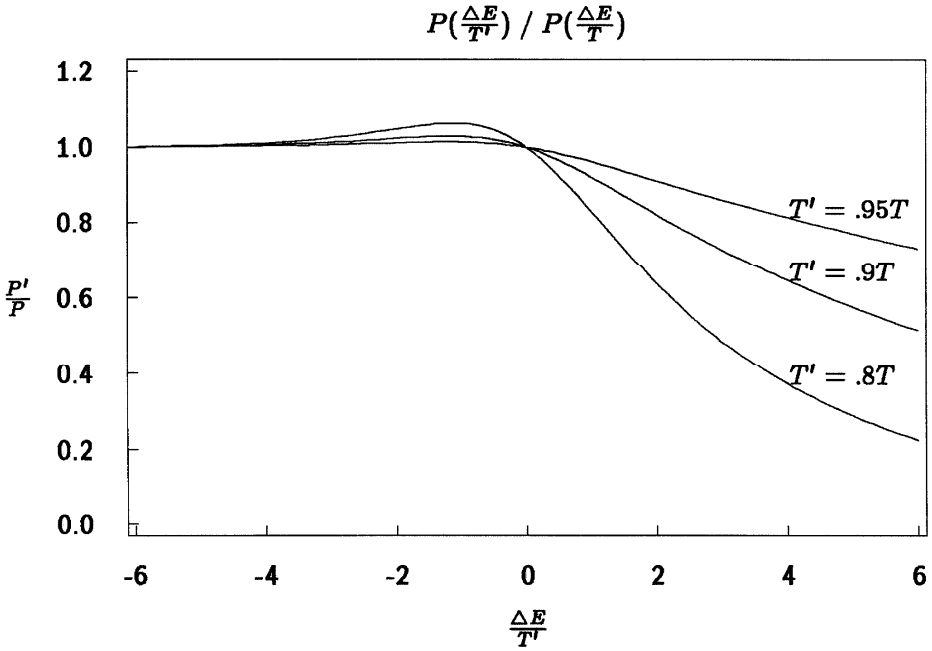

Figure 3.2: Ratio of acceptance probabilities for various decay ratios

relative probability (Figure 3.2) significantly different, but these moves are very unlikely to be accepted, as shown in Figure 3.1.

Since the change in probability is so small, the process is not highly sensitive to *direct* effects of the choice of the exponential decay ratio. However, if the annealing schedule is simply an exponential decay with fixed length time steps, the decay ratio will *indirectly* control the effective duration of the annealing process by changing the number of temperature steps.

$$number\ of\ temperature\ steps = \frac{\log(\frac{T_{high}}{T_{low}})}{\log(\frac{T'}{T})}$$

Changing the decay ratio from .9 to .95 will more than double the number of temperature steps and the total annealing time. If the number of trials at each temperature step is insufficient to allow a good approximation of equilibrium to be reached, a change in the ratio can have dramatic effects. The somewhat mystical air improperly associated with the choice of decay ratios should be attributed to the inherently difficult problem of determining the number of trials required at each temperature step, to be discussed in section 3.7.3.

3.7.2 Modifying exponential decay

As suggested by Kirkpatrick [5], the 'specific heat' can suggest where slower cooling might be helpful. Determination of the discrete analog of the 'specific heat' requires finding the equilibrium energy of two consecutive temperatures, which makes circular the question of determining the minimum time step needed to approach equilibrium. A practical variation of this idea is to remain at the same temperature as long as the energy is falling at a significant rate.

NET accepts two parameters to control this adaptive extension of the basic time interval. One is the minimum improvement in the energy. The other is the net rate of the *number* of positive and negative moves accepted, a hedge against a few disastrously large bad moves forcing a temperature change. The former parameter is usually set close to zero, so that any improvement will delay the temperature decrease. It predominantly affects the higher temperature schedule. The latter parameter must be significantly larger than zero to avoid 'hanging' at a near-equilibrium point. Its greatest effect is in extending the process at 'near-frozen' temperatures. This is useful if one is searching hard for perfect solutions. In practice both statistics are highly correlated. Both numbers are scaled to reflect the size of the problem and the acceptance rate of trial configurations. Large values for these parameters force pure exponential decay.

At this point it is instructive to consider a particularly good example of the adaptive annealing schedule, drawn from the one process per node comparative study of section 4.2, mapping a binary 7-cube ($|V| = 128$) onto a binary 7-cube.

Figure 3.3 shows the evolution of the system from a perfect (identity) initial mapping (marked by $\times$) to a highly disordered hot state (marked as E_H), then the slow cooling back to a perfectly ordered 'frozen' system, (E_F). The RMS (root mean square) *logical distance* (see section 4.2.3) is the typical distance between communicating processes, i.e., for vertices in the logical graph connected by an edge, what is the physical distance between the physical nodes the processes are mapped onto? A perfect mapping gives an average value of 1.

The approximate specific heat is shown in Figure 3.4. This is an 'averaged derivative' of the energy (closely related to the RMS distance) shown in Figure 3.3. Notice the prominent

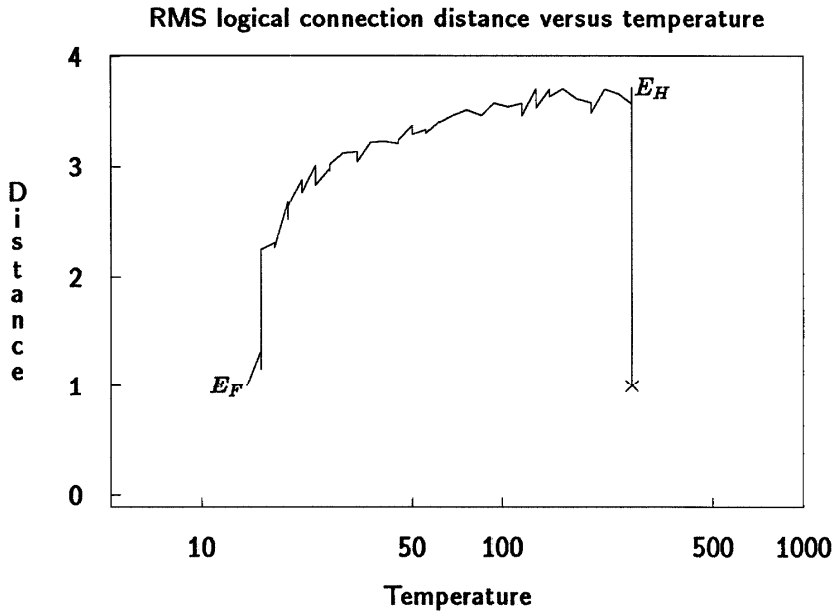

Figure 3.3: Binary 7-cube mapped onto binary 7-cube

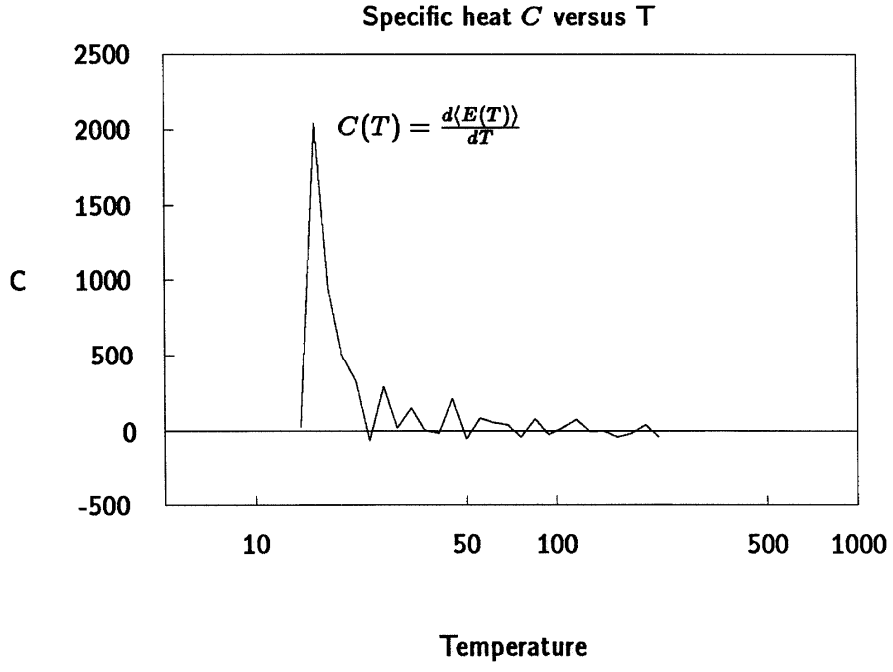

Figure 3.4: Binary 7-cube mapped onto binary 7-cube

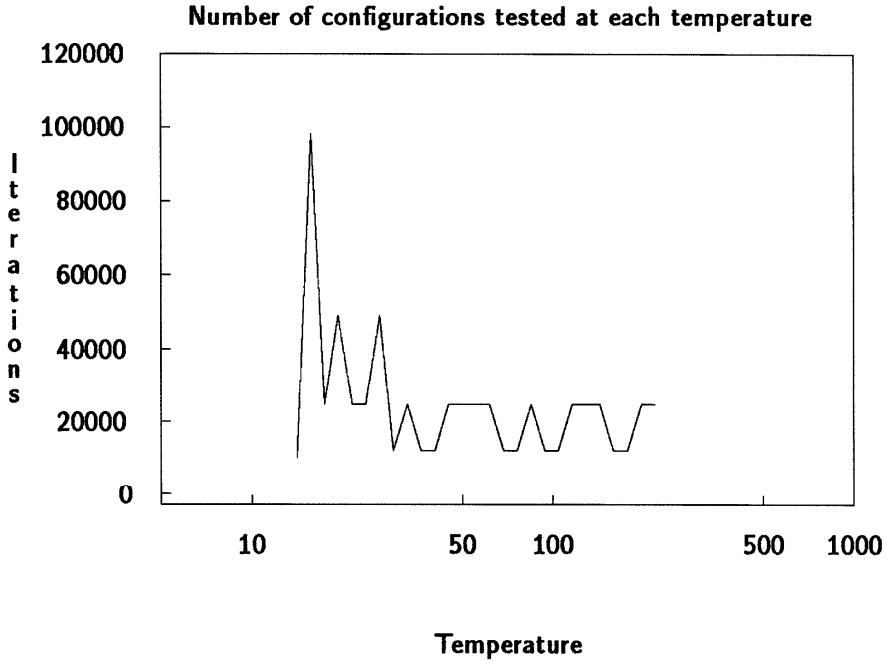

Figure 3.5: Binary 7-cube mapped onto binary 7-cube

phase transition at T = 15, the 'melting point', which is seen as the vertical drop in Figure 3.3 and as the spike of Figure 3.4. When the energy is rapidly falling at T = 15, the improvement causes the basic evaluation period of 12288 trials to be extended 7 times, as seen in Figure 3.5.

3.7.3 Relaxation time

For the statistics which determine the adjustment to the basic annealing schedule to be meaningful, the interval between evaluations must be adequately long. Traditionally, a certain number of trials is attempted for each movable entity; this number is adjusted experimentally. For NET, an input parameter specifies the number of iterations per process to define a minimum time period for further estimates.

It is assumed that the system has a characteristic 'relaxation time' which will allow it to settle close to equilibrium after a temperature change. There is no reason to believe that this time is constant across all temperatures; nevertheless a *lower* bound can be experimentally estimated.

The system is 'shocked' from a very cool, 'quenched' state. A very high temperature is estimated (section 3.7.4) and the system is simulated until a rough equilibration is found, as indicated by the appearance of both an increase and a decrease in system energy. This rather crude criterion tends to confound the mobility of the system and its level of noise, but is generally adequate.

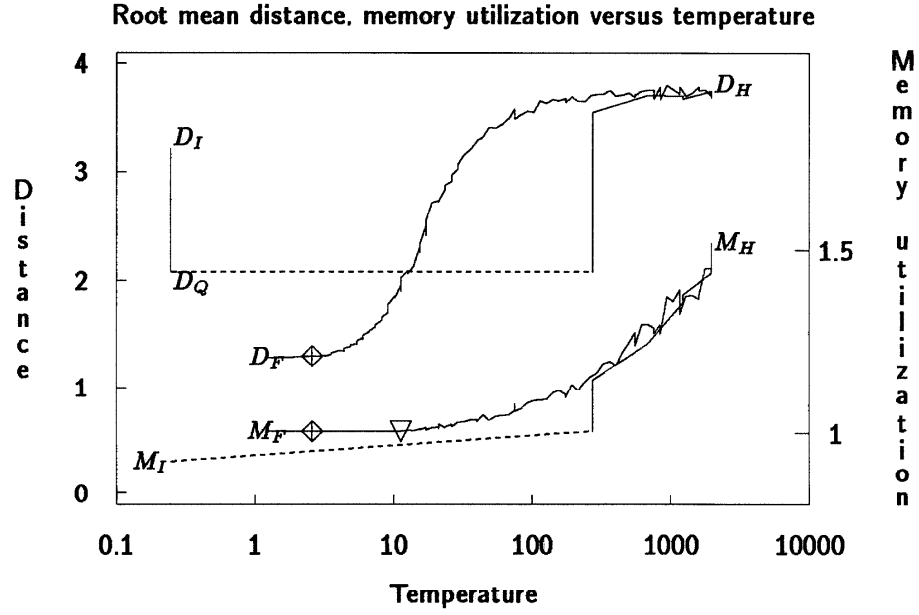

Figure 3.6: Torus, n = 3, k = 8, mapped onto torus, n = 3, k = 5

3.7.4 Estimation of high and low temperatures

The system is randomized by heating at a temperature sufficiently high to cause acceptance of a given fraction of evaluated configurations. For the heat bath acceptance function the acceptance rate will asymptotically approach 50% in a noiseless system as temperature is increased.

One technique is to heat the system with an exponentially increasing temperature until the desired acceptance rate is achieved. Alternatively, the insight [14] that T should approximate $\triangle E$ suggests that we may more quickly estimate the temperature required by estimating $\triangle E$. We survey the nearby volume of configuration space by considering the $\triangle E$ which would result from moves possible from the initial state. The standard deviation of a large number of adjacent configurations is computed, then the inverse acceptance function is applied to yield an estimate for the necessary 'hot' temperature. This method is necessarily approximate, since the initial region of configuration space does not necessarily resemble the hot system. However, errors in prediction of the hot temperature are not troublesome since it can be reestimated and tested after the system has 'equilibrated' at the predicted temperature.

3.8 An example of the complete annealing cycle

It is useful to examine an example with two energy components (from section 4.3). Figure 3.6 shows the history of finding a mapping of an 8x8x8 torus onto a 5x5x5 torus. This figure is comparable to Figure 3.3 with an additional energy component shown, the

memory utilization term. Consider the upper curve which represents the evolution of the distance term.

Initialization and quenching

The initial mapping (D_I) is poor, with a large typical logical distance. The system is quenched by annealing at $T = 0$, (plotted on the logarithmic X axis as .25 for convenience), dropping quickly to a local minimum at D_Q.

Estimation of the equilibration period

To estimate the number of cycles required to bring the system into approximate equilibrium, the quenched system is run at the high temperature estimated from the initial mapping. The dashed horizontal line to the right of D_Q records the instantaneous change in temperature. The next, vertical, segment indicates the rapid increase in the energy of the system in the next basic time interval as it responds to this thermal shock. The estimate of the desired high temperature, (such that the average acceptance probability is .48), is revised three times as the line snakes to the upper right. Finally, at $T = 1963$ the revised value for T is close to the current value, and the equilibration phase is concluded. The length of this phase is the minimum length of future temperature steps, the equilibration period.

Heating

As a further check on our high temperature estimate, the system is run for at least one period to confirm that the acceptance rate exceeds the required value. If not, the temperature is increased exponentially (by the inverse of the decay ratio) until the rate is adequate. In this example one period suffices, barely visible as a short vertical bar by the D_H inscription. The randomized system's RMS distance is close to the average distance of this physical graph, 3.6, (see section 5.4.1).

Cooling

The bulk of the plot is the cooling phase, shown as the noisy curve from D_H to the diamond near D_F. During this phase the energy is assumed to be a quasi-equilibrium value for the temperature. There is a marked phase transition just above $T = 10$, also shown by a jump in specific heat in Figure 3.7. This phase transition is less sharply defined than that of Figures 3.3 and 3.4 since the logical and physical graphs are not isomorphic.

Freezing and termination

A low temperature is estimated from the initial mapping, and marked on the curves with the diamond symbol. When the annealing schedule reaches this temperature (reestimated as necessary), it is assumed that the system is near the final configuration, and termination due to energy stagnation becomes possible. If the energy does not show a decrease in several consecutive equilibration periods it is assumed that the system is frozen, and the annealing process is done. This conclusion is marked as D_F.

The significantly lower value of the curve at D_F relative to D_Q is an indication of the improvement possible through simulated annealing as contrasted to simple iterative improvement.

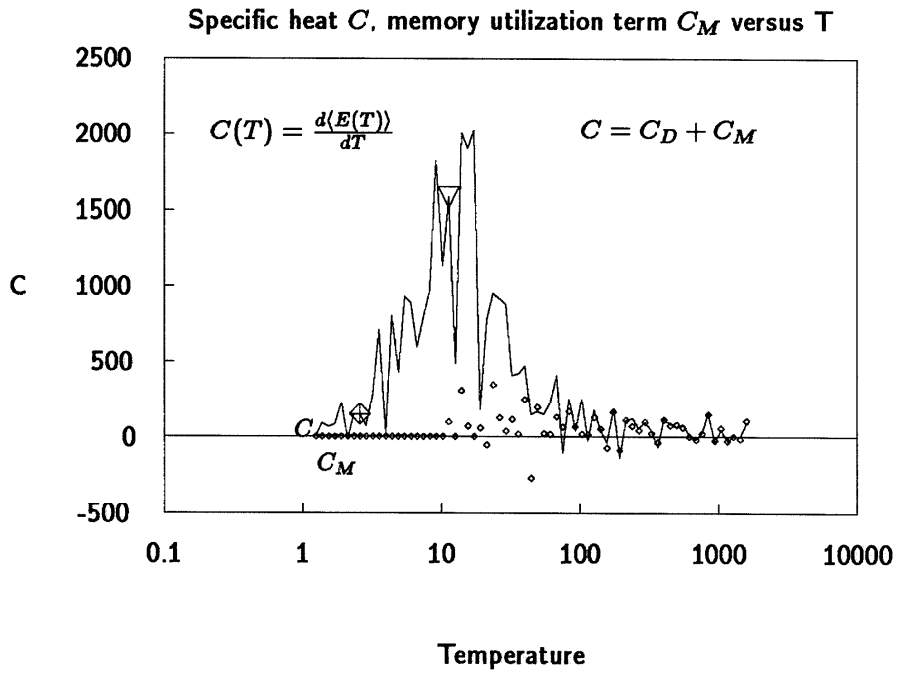

Figure 3.7: Torus, n = 3, k = 8, mapped onto torus, n = 3, k = 5

Memory utilization energy

The lower curve of Figure 3.6 represents the memory utilization energy term. It is intended to prevent the number of processes assigned to a node from exceeding the memory available, in this case no more than four of the equally sized processes. (Alternatively, if the logical graph vertex weight is considered as runtime, the term represents a penalty for load imbalance.)

The curve is marked much as the distance term curve. The corresponding scale is on the right axis of the plot. M_I marks the initial mapping, which is a uniform distribution of four processes per node. Since the memory utilization is already minimized, the quenched value M_Q (not marked) is the same as the initial value M_I.

After the quenching phase, the coefficient of the memory term is reestimated, in this case increasing by $\approx$ 40%, (see section 3.10). The line to the right of M_I slants upward, indicating a discontinuity in the memory utilization energy due to the new coefficient, and to the right, due to the temperature increase at the start of the equilibration phase. No changes in the system configuration are made, only in parameter values.

The equilibration and heating phases (leading up to M_H) are quite similar to the D line. In the cooling phase (from M_H to M_F), the inverted triangle identifies the point at which the memory utilization energy first reached its minimum value.

Temporal evolution

Figure 3.8 shows the temporal development of the same mapping as the previous two plots. In this linear plot, the total energy (E) is seen to be the sum of the distance (D) and

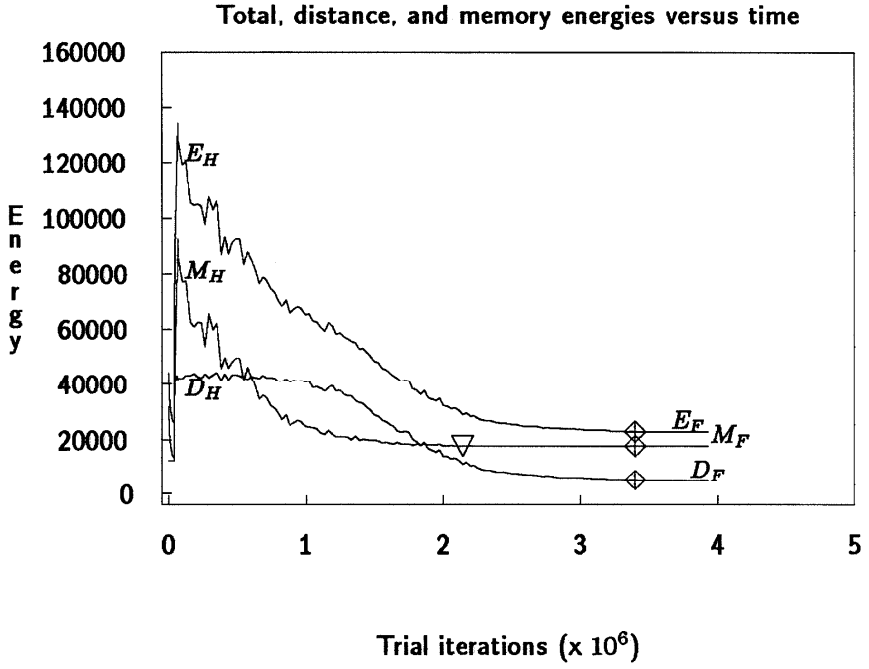

Figure 3.8: Torus, n = 3, k = 8, mapped onto torus, n = 3, k = 5

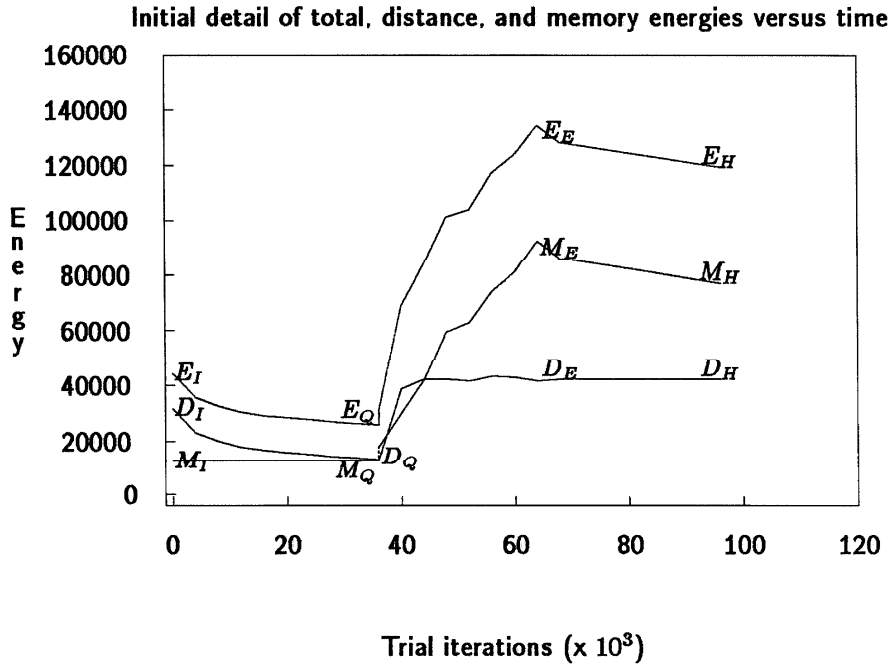

Figure 3.9: Torus, n = 3, k = 8, mapped onto torus, n = 3, k = 5

memory utilization (M) terms. Essentially the first half of the process (before the inverted triangle) was devoted to minimizing the memory utilization component, while the second half served to reduce the distance component.

Figure 3.9 is an expanded detail of the start of Figure 3.8. The quenching phase lies between E_I and E_Q. The increase in the memory utilization coefficient is visible as the small dashed vertical segment just above the M_Q point where the M term jumps discontinuously. The number of iterations between E_Q and E_E is the equilibration phase, which becomes the unit time step (28000 iterations/step instead of the previous 4000 iterations/step) thereafter. The heating phase is the single period between E_E and E_H. The cooling phase comprises most of Figure 3.8.

3.9 Memory doesn't freeze

The memory utilization term does not have a phase transition. This reflects the fact that the most likely state is always a uniform distribution, though high temperature broadens the variation about that state. Alternatively, it may be observed that the 'legal' number of processes per node is dependent on the average statistics of the system rather than the detailed behavior of neighboring nodes. A dramatic collective phenomenon is unlikely with such weakly coupled elements.

Figure 3.7 shows the components of the specific heat due to both the distance and memory utilization terms, C_D and C_M respectively drawn as lines and small diamonds. The memory utilization term displays no peak to suggest a phase transition.

3.10 Estimation of energy term weights

Different energy components represent different, usually conflicting minimization goals for the system being optimized. Incorrect weighting of the different terms can cause inefficient annealing schedules, unmet constraints, and poor optimization results. The following method simplifies the weight selection process.

3.10.1 Define a hierarchy of constraints and goals

It is necessary to determine the relative importance of the conflicting goals to be optimized. For example, the multiple process per node study of section 4.3 requires that no more than 4 of the processes be assigned to a single node to avoid overutilization of available memory. This is made a 'soft' constraint by assigning a sufficiently large value to the coefficient of the memory utilization term. The relative weightings of the terms produce a hierarchy of importance. Terms of relatively greater weight are minimized at higher temperature, earlier in the cooling schedule. After the temperature has dropped further, the goal or constraint embodied in that term is effectively 'frozen' in a local minimum, and unlikely to change at lower temperatures.

Consider Figure 3.6. The memory utilization component has a relative weight of 3 compared to the distance term. The memory term is minimized at $T = 11.2$, at the inverted triangular mark. The distance term is essentially minimized at $T = 2.6$, the start of the freezing phase, marked by the diamond. The ratio of the two temperatures is 4.3, reasonably close to the specified relative weight 3.

Such simultaneous solution of conflicting goals is relatively expensive. As the system is cooled the acceptance rate will drop as more classes of moves are effectively prohibited by satisfied constraints. A fixed set of moves will result in a low average acceptance rate. An adaptive configuration generator might improve this situation by not trying types of moves expected to have low probability of acceptance at given temperatures. The additional complexity is unlikely to be worthwhile except at very cool temperatures.

3.10.2 Estimation of absolute weights

The total values of the energy terms do not affect the annealing process, only the changes in the terms. The basic problem is to roughly balance the typical delta values of the energy components, e.g., if

$$\triangle E = \triangle D + \triangle M$$

then the goal is to find a $c_{memutil}$ coefficient (section 1.4) so that

$$\triangle D \approx \triangle M$$

The actual coefficient is the product of the relative weight given by the user and the absolute coefficient estimated by the program to yield parity.

The absolute coefficients are estimated by sampling the local configuration space, configurations differing from the current configuration by only one move, to determine typical delta energy values. The absolute coefficients are estimated to give all energy terms delta values comparable to that of the distance term.

A first estimate is made at the initial mapping; a second after quenching. The primary purpose of the quenching phase is to force all initial configurations into a rough equilibrium to allow good estimation of the typical delta values of energy components.

This method gives excellent results in most cases; some are more recalcitrant, (see section 4.3.4).

3.11 Performance

The space requirements for NET are quite readily calculated. The evaluation rate of trial configurations is somewhat more complicated, but is not extremely variable. The total run time cannot be predicted without detailed knowledge of the mapping problem, but a lower bound can be easily calculated *after* NET estimates high and low temperatures for the problem, by calculating the number of temperature steps using the formula of section 3.7.1.

3.11.1 Time

The simple move set allows efficient calculations. For the graphs of sections 4.2 and 4.3 the generation and evaluation of a new configuration takes 3 - 4 milliseconds, corresponding to 15000 - 18000 moves per CPU-minute on a VAX-11/750.

The major procedures of the main iterative loop are relatively balanced in computation costs. For a test using both distance and memory utilization terms, with about a 40% acceptance rate, the following approximate profile was measured.

Approximate CPU time consumed		
Procedure	Purpose	CPU fraction
calcconnectionenergy	calculate distance energy term	24%
calcmemutilenergy	calculate memory utilization term	16%
newconfig	generate trial configuration	24%
newchosen	acceptance decision	15%
acceptnewconf	accept new configuration	10%
rejectnewconf	reject new configuration	11%
		100%

Some of the procedures perform linear searches of lists, so performance decreases with increasing average graph degree and number of resident processes. These costs become dominant when these lists grow to hundreds of elements each, e.g., 512 processes per node. Substituting sequential for random selection in more places could mitigate this linear increase; more fundamental changes to the data structures would compromise the program's applicability for large sparse graphs.

Congestion terms

Congestion calculations require dynamic generation of new paths and calculation of the costs incurred at each physical channel and/or node along the path. When the system is in a disordered state (the typical case), the path length is comparable to the average length of the physical graph. For graphs examined in section 4.3, enabling the congestion terms increases the per-move evaluation time by approximately an order of magnitude.

3.11.2 Space

NET is limited by the $O(\ |V_{physical}|^2\ )$ array(s) used for the shortest distance (and optionally, path generation arrays). Because the new configuration loop runs sequentially through the processes rather than the nodes, the locality of access is poor and virtual memory paging is inefficient. The largest physical graphs that have been used contain 1024 vertices.

For a particular physical graph, if an analytic formula exists, the distance calculation can be done procedurally at increased time cost. Generality and speed are traded off against space.

For congestion calculations the path space used is $O(\ |E_{logical}|\ |average\ path\ length|\ )$ with a fairly large coefficient due to the pointers used for dynamic lists. For dense graphs, path storage requirements can be significant.

Chapter 4

Comparisons of physical graphs

All cases are unique, and very similar to others.

T.S. Eliot

4.1 Introduction

In this chapter the methods described in the previous chapter are applied in two comparative studies. In each, a set of logical graphs and a set of physical graphs are selected, representing 'typical' computations and possible machine architectures, respectively. For each combination, the NET program produces an embedding of the logical into the physical graph. The resultant mappings set upper bounds on the best possible (least-costly) mappings and provide a basis for comparison of machine structures.

The first study ('1 → 1') allows at most one process per physical node; the second ('4 → 1') at most four processes per node. For the 1 → 1 study, process exchange is the only allowed move; thus the constraint of one process per node is enforced by a restricted configuration generator, a 'hard' limit. The second study enforces the four process limit by a memory utilization penalty term, a 'soft' limit. The relative weight of the two energy terms is estimated, with minimal interactive input, by the method of section 3.10. Neither study considers channel or node I/O utilization.

The logical graphs for the two studies contains about 128 and 512 vertices, respectively. The physical graphs contain about 128 nodes in both cases. The exact sizes of the graphs are determined by their 'natural' sizes, e.g., 127 vertices for a binary tree of height 6.

This chapter explains the details of the two experiments and presents the results with some interpretation. More analysis is found in the next chapter.

4.2 Mappings of one process to one node

Ten logical graphs and ten physical graphs of about 128 vertices each constitute the test sets. Input parameters to NET are identical for each mapping; actual values such as high and low temperatures and the number of iterations in the basic interval are determined automatically.

Since the energy function has only one term, (distance, see section 4.2.3), the quenching phase required for estimation of relative weights of energy terms is omitted from the annealing schedule (see section 3.10.2).

4.2.1 Logical graphs used for test cases

Ten logical graphs are used as 'typical' test problems. Most logical graphs have unitary edge weights. *Weighted* trees have edge weights increasing with height (defined as distance to leaves). The weight of an edge to the parent vertex is the sum of the weights of edges to child vertices. For example, in the weighted binary tree, the edge weights are 2^{height} (with the height of edges to leaves defined as zero). Weighting the tree produces problems of interpretation (discussed below), but is probably more representative of the root congestion likely to be seen in real problems.

Toroidal meshes (tori) are defined by specifying the number of dimensions, n, and the number of distinct coordinates in each dimension, k [12]. The graphs are cyclic in each dimension, i.e., for a given dimension the vertex at coordinate 0 communicates with the vertices at coordinates 1 and $k-1$. The degree (number of edges per vertex) of the graph is $2n$. The binary n-cube is a degenerate $k = 2$ torus with redundant edges suppressed so that the degree is n rather than $2n$.

Physical graphs with less than 128 nodes cause proportionate reduction in the size of logical graphs due to 'illegal' initial mappings of processes to nonexistent physical nodes, as discussed in section 3.1.4. This expedient avoids the difficulty of tailoring logical graphs to physical graphs of varied size, but produces some perturbation of the quality of results. Toroids will have cycles broken by the removal of vertices from the logical graph. In particular, mappings of one-dimensional toroids (rings) can suffer appreciably from the loss of the cycle.

The logical graphs used in the single-process per node study are:

A tree (bush) of height 1, with 120 children. $|V| = 121$.

A 10-tree of height 2. $|V| = 111$.

A weighted 10-tree of height 2. $|V| = 111$.

A binary tree of height 6. $|V| = 127$.

A weighted binary tree of height 6. $|V| = 127$.

A binary 7-cube. $|V| = 128$.

A randomly-connected graph with degree uniformly distributed between 1 and 8. $|V| = 128$.

A torus, n = 3, k = 5, i.e. 5x5x5 three-dimensional toroidal mesh. $|V| = 125$.

A torus, n = 2, k = 11, i.e. 11x11 two-dimensional toroidal mesh. $|V| = 121$.

A torus, n = 1, k = 128, i.e. a ring. $|V| = 128$.

4.2.2 Physical graphs

The physical graphs used in this study are :

A binary 7-cube. $|V| = 128$.

A torus, n = 3, k = 5. $|V| = 125$.

An shuffle-exchange graph augmented by a ring, derived from the NYU Ultracomputer project [11]. $|V| = 128$.

A torus, n = 2, k = 11. $|V| = 121$.

A shuffle-exchange graph. $|V| = 128$.

A cyclic Sneptree[8], a binary tree of height 6, augmented with additional channels from leaves to all other nodes. This network has two distinct Hamiltonian circuits. Node degree ≤ 4. $|V| = 127$.

An 11-tree of height 2. $|V| = 133$.

A 3-tree of height 4. $|V| = 121$.

A binary tree of height 6. $|V| = 127$.

A torus, n = 1, k = 128, i.e. a ring. $|V| = 128$.

All edges are of unit weight. The average and maximum distances of these graphs are given in section 5.4.1.

4.2.3 Cost functions

Two variant studies were done for these logical and physical graphs, one using a squared cost function, the other a linear cost function. The cost function here is the sum of distance terms only

$$cost = \sum_{logical\ channels} distance^{k_1}$$

where *distance* is the physical distance separating logically connected processes, and $k_1 =$ 2 or 1 respectively. At each basic time interval, the NET program calculates a *normalized root mean* statistic for each energy term E_i. For the distance term this is

$$\sqrt[k_1]{\frac{\sum_{logical\ channels} distance^{k_1}}{|Logical\ channels|}}$$

For graphs with unit edge costs, this is the RMS logical distance for the squared cost function, and the true average logical distance for the linear cost function. The plots of the previous chapter display RMS distances. In addition, final statistics from each run include a true average for each energy component. This statistic is used in all following comparative figures.

4.2.4 Comparison of mapping results

In the following figures, each statistic represents a single mapping for each possible combination of logical and physical graphs. The 10x10 tables are split across pages into two 10x5 parts because of size.

Average logical distance for squared cost

Figures 4.1 and 4.2 represent the average logical distance for the mappings, i.e., the average physical distance between logically adjacent vertices. The actual statistic shown is the *reciprocal* of the average distance, so that 'bigger is better'. The area of the circle is proportionate to the inscribed number. 1 is a lower bound (not necessarily achievable) for the average distance, and therefore an upper bound for the inverse average distance displayed. This bound is met for mappings of 7-cube $\rightarrow$ 7-cube and ring $\rightarrow$ ring. Self-mappings of trees and the two and three dimensional tori fail to reach perfect mappings. Section 5.2.1 introduces a tighter lower bound for the average distance.

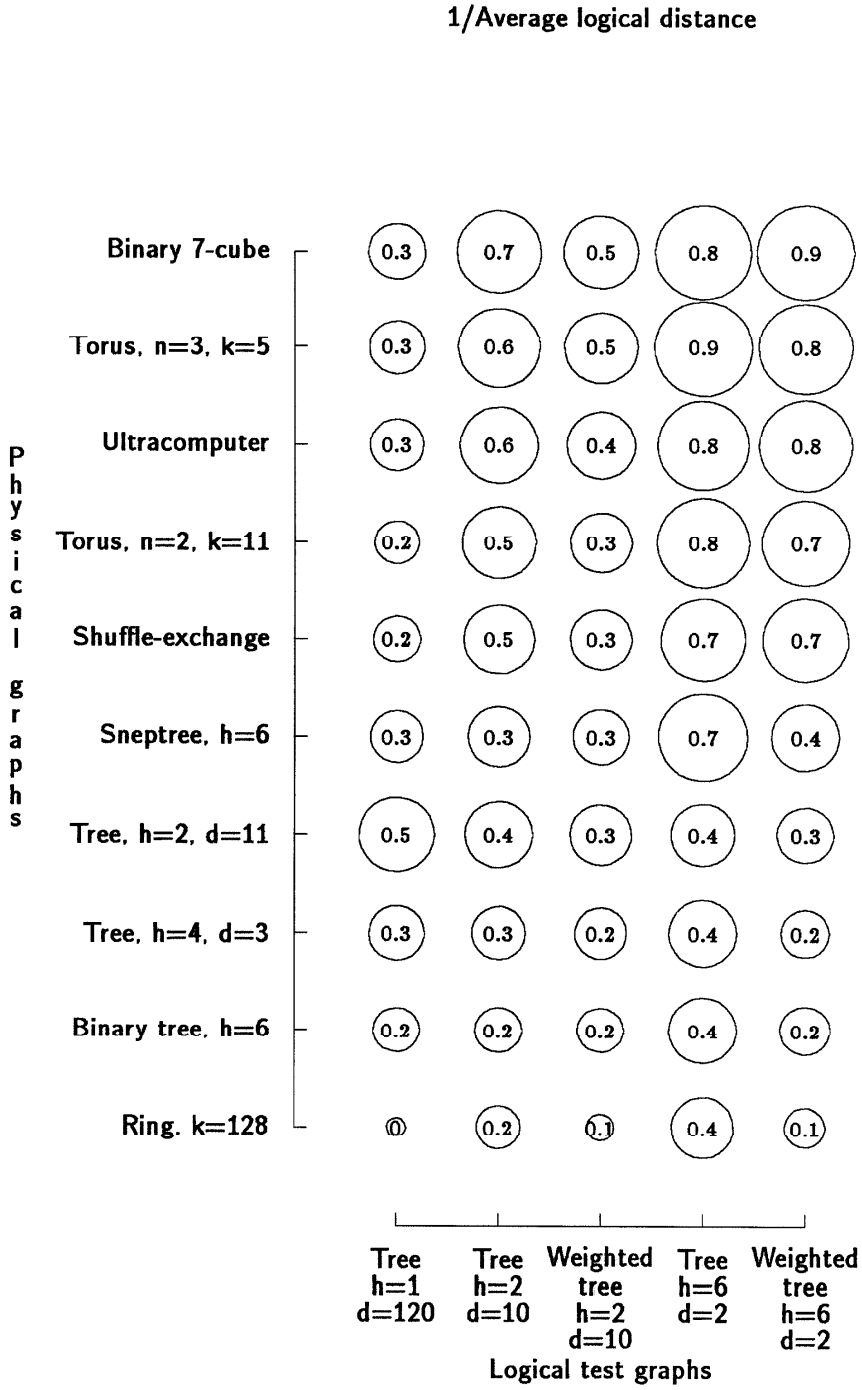

Figure 4.1: Inverse average distances for $1 \rightarrow 1, E = \sum distance^2$ series

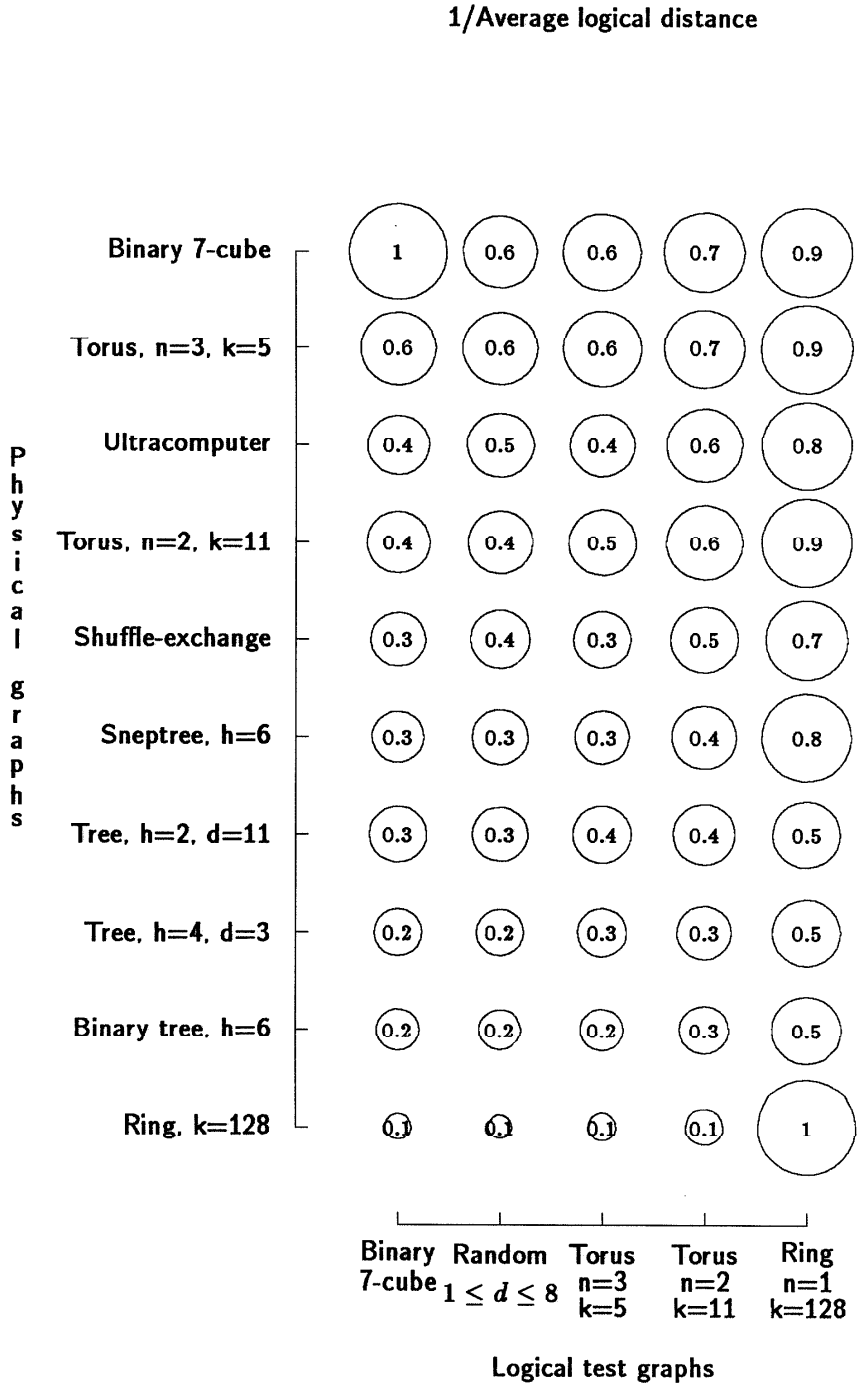

Figure 4.2: Inverse average distances for $1 \rightarrow 1, E = \sum distance^2$ series

The logical and physical graph axes are labelled in the order listed above. In the tree labels, d is the maximum number of descendents, e.g. $d = 3$ for a 3-tree. Physical graphs are ordered in decreasing overall quality of result.

It is important to note that for weighted trees the average logical distance statistic differs qualitatively from the actual cost function being minimized. For the cases presented here, mappings onto trees and the ring differ significantly for the weighted and unweighted binary trees. The distances of the heavily weighted edges near the root are minimized well, while the unit weight edges to leaves may have relatively poor mappings without large cost. For this reason the results of weighted and unweighted trees are not directly comparable.

Maximum logical distance for squared cost

Figures 4.3 and 4.4 similarly display the maximum physical distance for logically connected processes. This measure could be significant if, for example, the computation were synchronized so that worst-case communications latency controlled the overall iterative cycle speed.

Distribution of logical distances

The average and maximum logical distances are convenient summaries of mapping quality. Much more information is contained in the distributions of distance in the mappings, i.e. what percentage of logical connections is of physical distance $0, 1, 2, \ldots$, etc..

Figure 4.5 is the distribution of distances for the mapping of a binary 7-cube onto a ring. Only the points marked with an inscribed circle are data points; other values are zero. The RMS and average distances are marked. This is an unusual distribution, reflecting a highly symmetric optimal mapping. Distances which are powers of two from 1 to 16 have 14.3% of the logical connections, distance 32 has 28.6%, other distances no connections. Four subcubes (5-cubes) of the 7-cube have been mapped uniformly around the 128-element ring. Each distance indicates the embedding of a higher dimensional cube in the ring; number of connections doubles at distance 32 because the four subcubes have closed the cycle.

A different presentation of the same data is Figure 4.6, a star plot with radii proportional in length to the percentage at each distance. Adjacent radii are joined by a peripheral line; since most values of this particular distribution are zero, only the line joining the ends of the distance radii 1 and 2 is distinct.

This and other, less singular, radial plots for all the mappings are shown in Figures 4.7 and 4.8. For clarity, only distances between 0 and 12 are shown in these figures; only the ring has a possible distance in excess of 12 (maximum distance 64). The embedding of a 7-cube onto a 7-cube, (in the upper left corner), shows a perfect mapping, with all logical connections at distance 1. Several of the mappings onto trees or the ring show bi- or multi-modal distributions, (see section 4.3.5).

Even without close study, the eye can discern patterns based on shape of the radial plots. For example, examination of the rightmost column of Figure 4.8 might lead one to hypothesize (correctly) that a perfect embedding of the ring (a Hamiltonian circuit) exists for all physical graphs except the shuffle-exchange graph and trees. The failure to obtain perfect mappings for the cyclic graphs is in part attributable to the severing of the 128-element ring in smaller physical graphs, as noted in section 4.2.1.

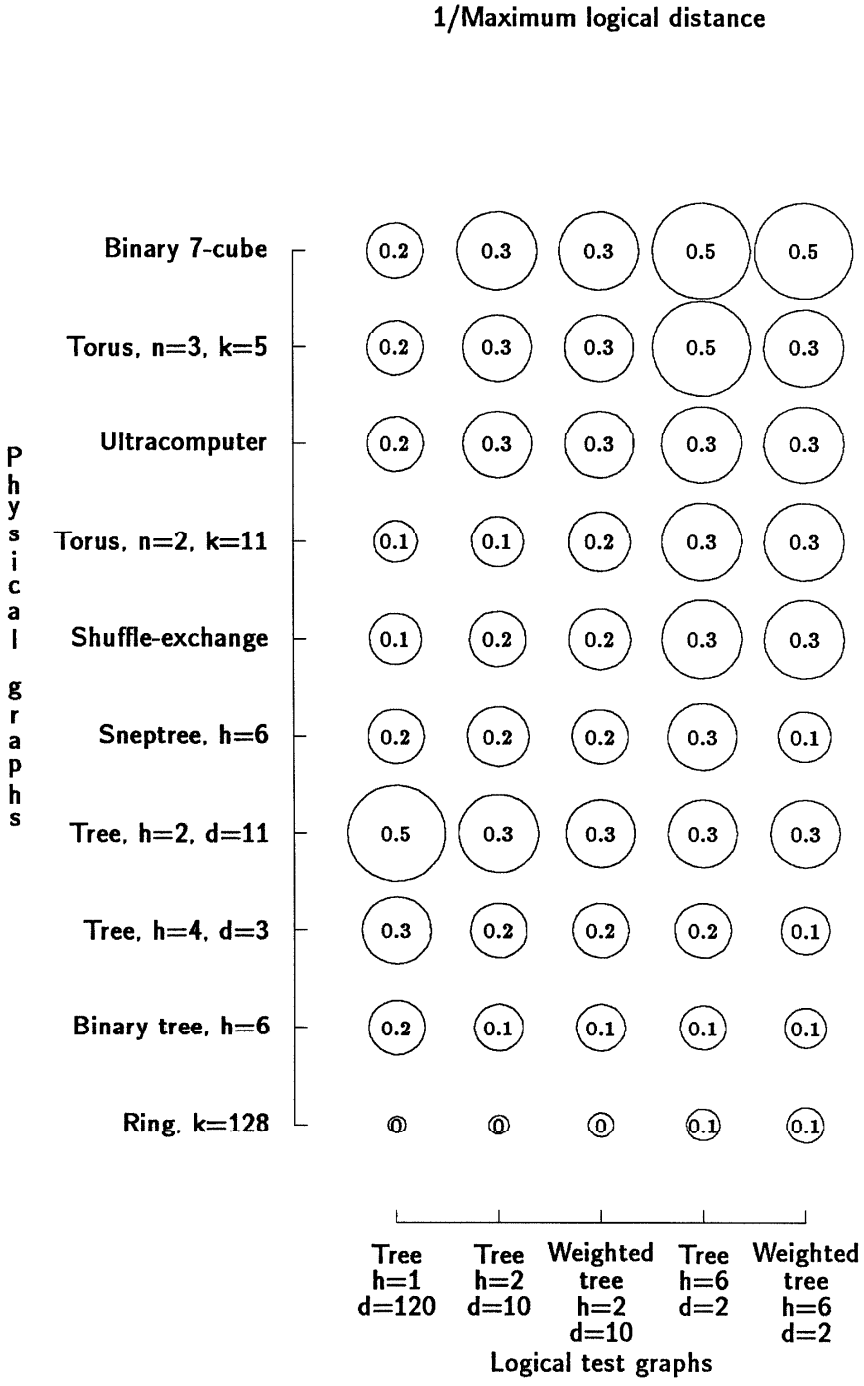

Figure 4.3: Inverse maximum distances for $1 \rightarrow 1, E = \sum distance^2$ series

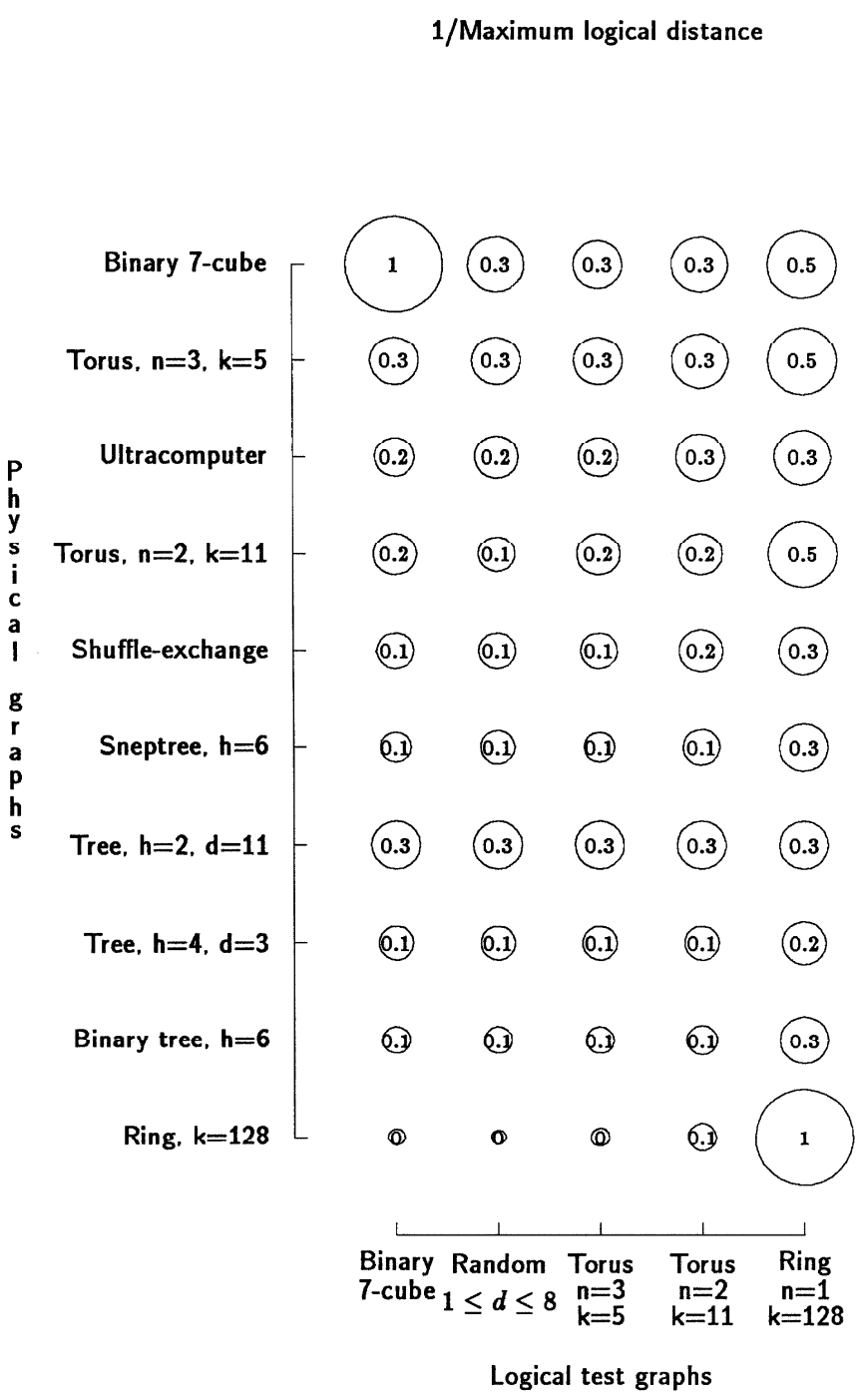

Figure 4.4: Inverse maximum distances for $1 \rightarrow 1, E = \sum distance^2$ series

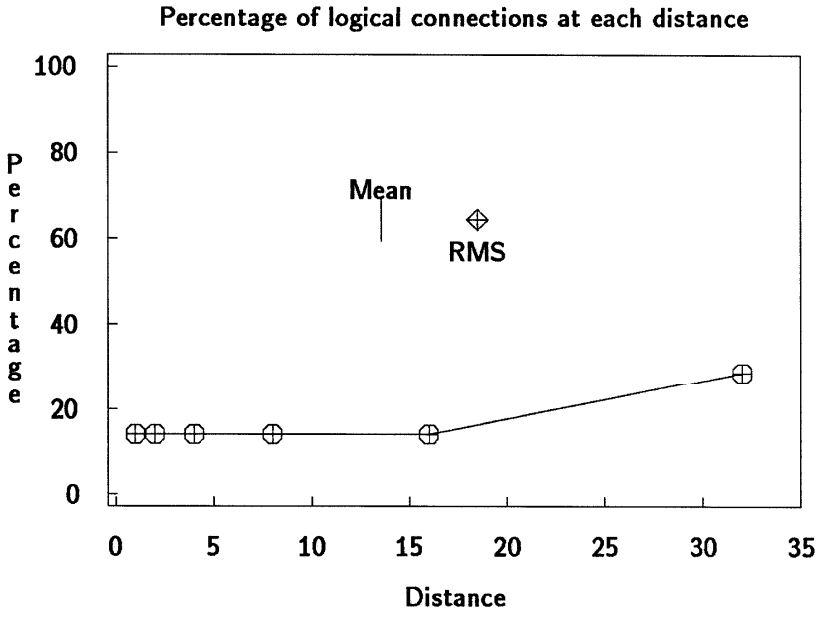

Figure 4.5: Distance distribution, 7-cube → ring

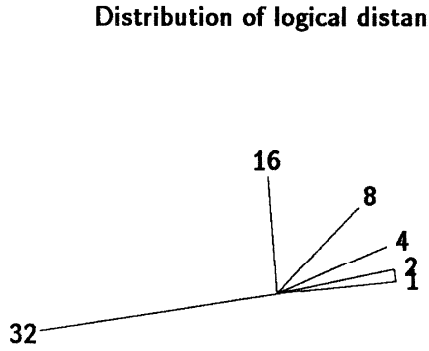

Figure 4.6: Distance distribution, 7-cube → ring

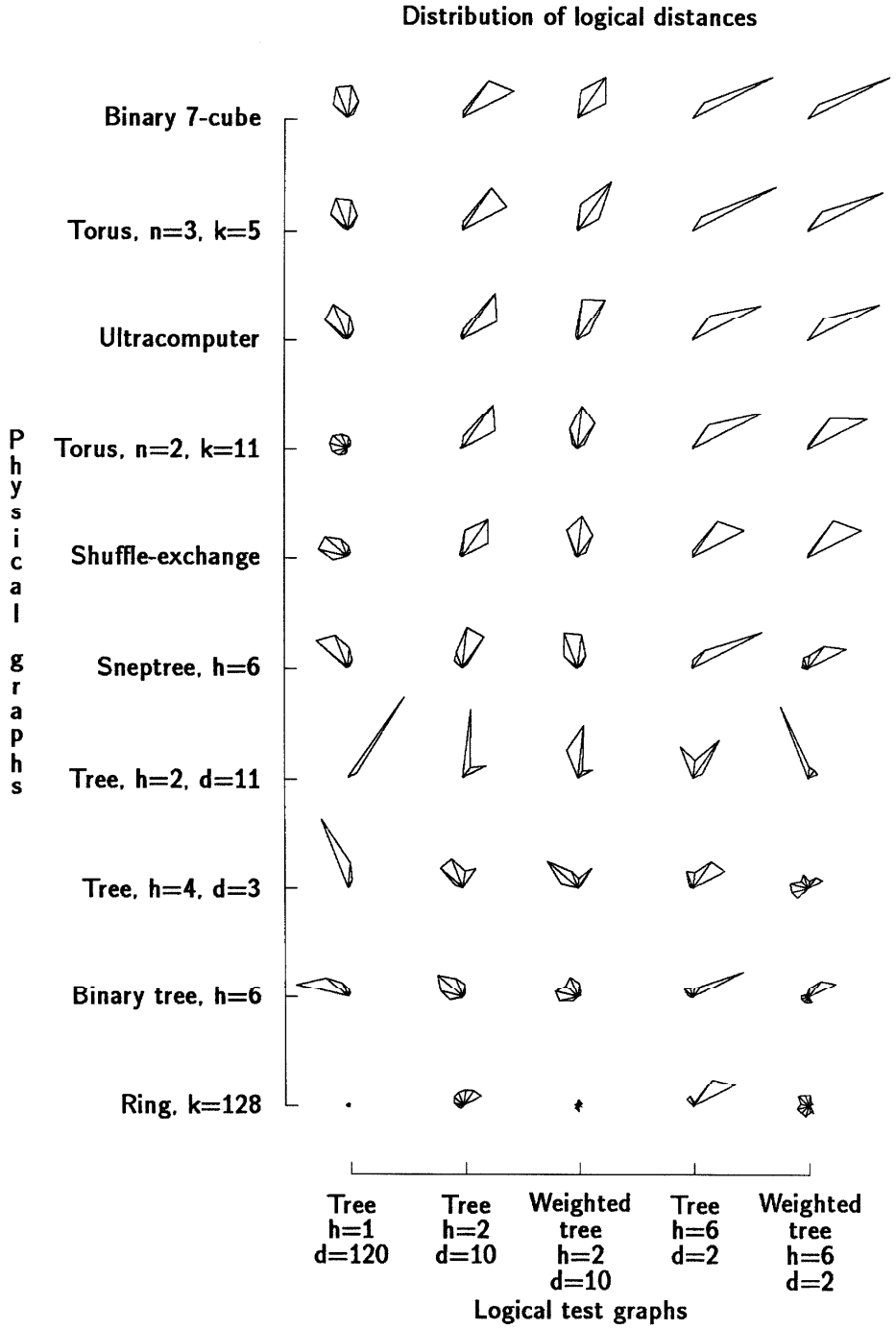

Figure 4.7: Distance distribution, $E = \sum distance^2$

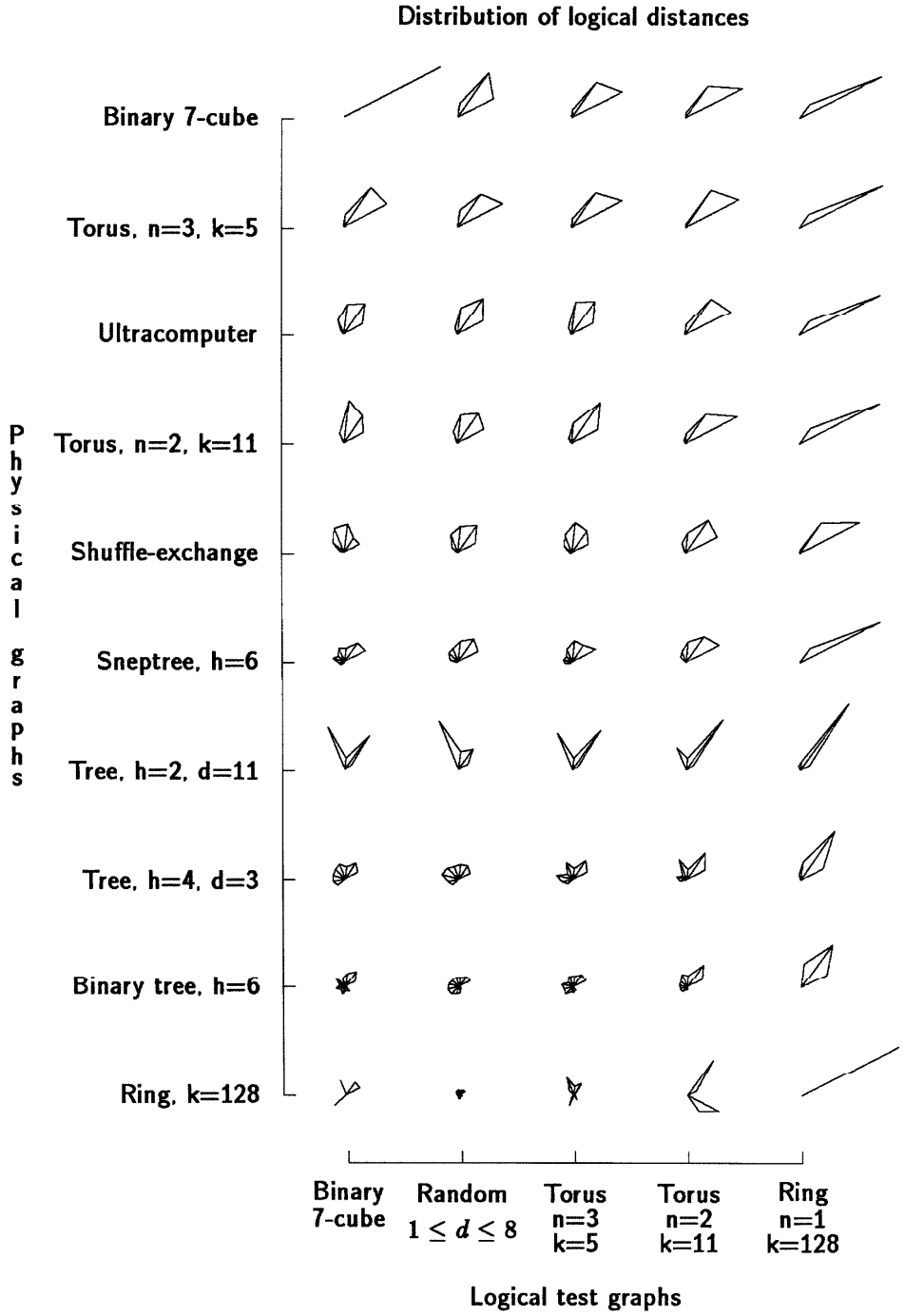

Figure 4.8: Distance distribution, $E = \sum distance^2$

4.2.5 Stability of results with linear cost

This variant $1 \rightarrow 1$ experiment has the same sets of graphs and parameters except that the exponent for the distance term is 1. The change in the energy function causes remarkably little difference in the resultant average logical distances. Figure 4.9 shows a comparison of all the results obtained with both values of exponents. A diamond is drawn for each combination of logical and physical graph, with the value obtained with $k_1 = 2$ as the x-coordinate, and the value for $k_1 = 1$ as the y-coordinate. If the results are identical for both exponents, the diamond will be on the $x = y$ line. The least squares regression of y on x line, (slope = .99) overlaps the $x = y$ line, (slope = 1). There is no significant trend in the average; of the 100 points drawn only a handful are far from the identity line. Comparing the linear to the squared cost results, we find the average is smaller in 40 instances, greater in 49, and equal (to four digits) in 11.

The plot of the maximum distances (Figure 4.10) shows much more striking behavior. The maximum distance for the linear case is smaller than that of the squared case in 0 instances, greater in 57, and equal in 43. The least squares regression (upper line) indicates 33% increased maximum distance with the linear cost function. This result is consistent with the intuition that larger exponents will cause severe penalization of the longest connections.

The overall similarity of results in the two variant experiments illustrates that the simulated annealing method used is robust and not excessively sensitive to variation in the exact shape of the energy function.

The squared distance cost function was chosen for the $4 \rightarrow 1$ study of the next section because of its success in reducing the maximum distances without significant increase in the average distance.

4.3 Mappings of four processes to one node

This ('$4 \rightarrow 1$') study limits the number of processes per node to no more than four by using a memory utilization penalty. Node memory capacity is a constraint which must be absolutely satisfied. Reduction of the physical distance of logical channels is a desired goal. This study demonstrates the ability of the NET program to find solutions to problems with a two-level hierarchy of a constraint and a goal.

Fewer logical and physical graphs are used in this than in the $1 \rightarrow 1$ studies. Four logical graphs and seven physical graphs of about 512 and 128 vertices respectively constitute the test sets.

The parameters controlling the annealing schedule are identical to those for the $1 \rightarrow 1$ except for one trivial change : the minimum number of trials per process per temperature step is a factor of four smaller since the number of processes is quadrupled. Two additional parameters define the memory utilization energy term. The exponent is constant (4) for all mappings. The 'relative memory weight' requires adjustment from the initial value in 9 of the 28 cases. All other parameters are unchanged.

4.3.1 Logical graphs used for test cases

Four graphs of types used in the $1 \rightarrow 1$ logical set are used as test problems :

A weighted binary tree of height 8. $|V| = 511$.

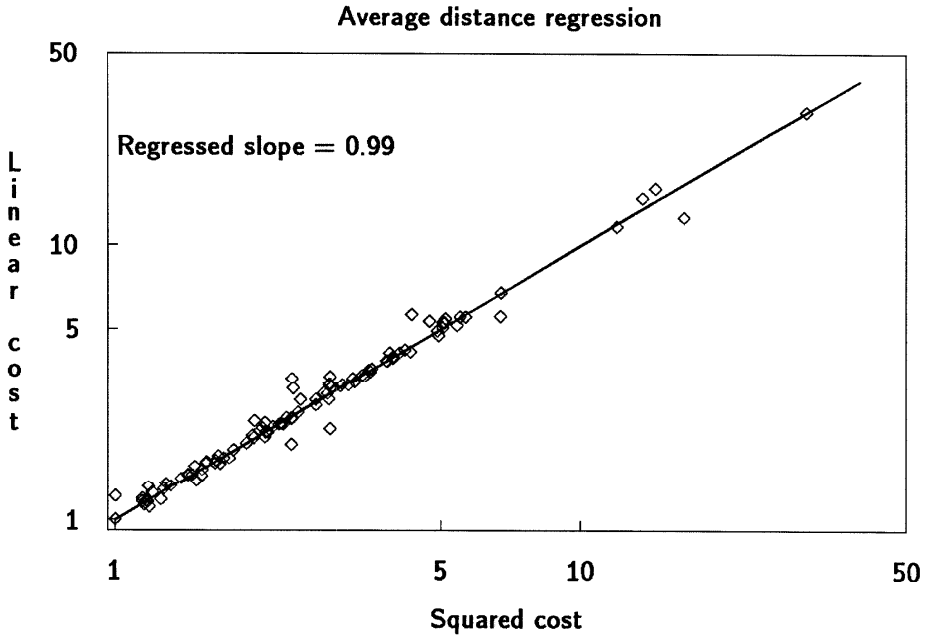

Figure 4.9: Regression of average distances for $distance^2$, $distance^1$ cost functions

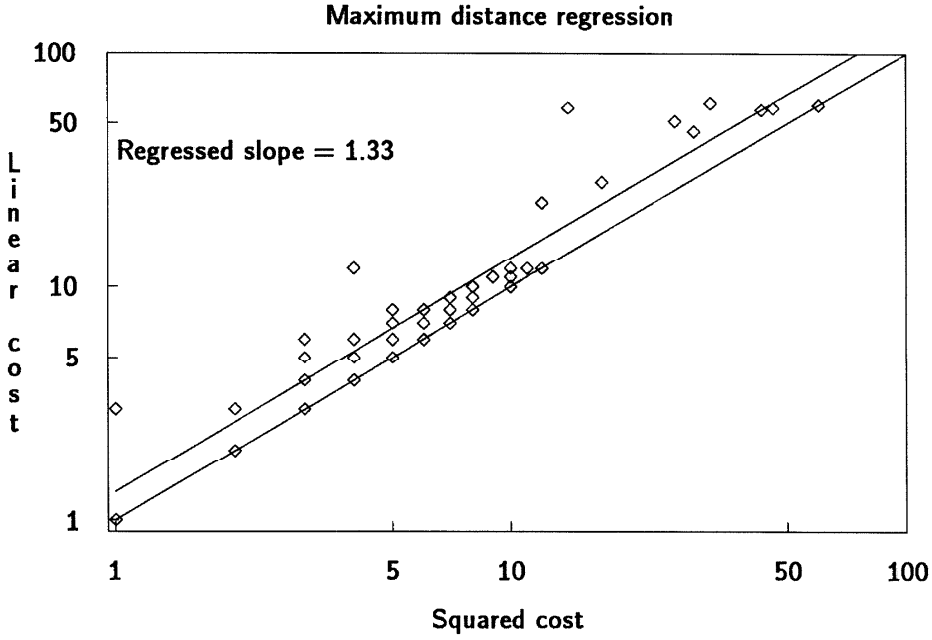

Figure 4.10: Regression of maximum distances for $distance^2$, $distance^1$ cost functions

A randomly connected graph with degree uniformly distributed between 1 and 8. $|V| = 512$.

A torus, n = 3, k = 8, i.e. 8x8x8 three-dimensional toroidal mesh. $|V| = 512$.

A torus, n = 2, k = 22, i.e. 22x22 two-dimensional toroidal mesh. $|V| = 484$.

All processes are assigned a vertex weight ('memory size') of .25 node units, restricting 'legal' solutions to at most four processes per node.

4.3.2 Physical graphs

The physical graphs are a subset of the $1 \rightarrow 1$ studies, all non-binary trees being removed.

4.3.3 Cost function

The cost function is the sum of two terms

$$cost = \sum_{logical\ channels} distance^2 + c_{memutil} \sum_{nodes} memutil^4$$

where *memutil* (memory utilization) is $\sum_{resident\ processes} memory\ size$, the total amount of memory occupied by processes assigned to a node.

This energy function enforces a limit of four processes per node by a nonlinear penalty for overutilization of node memory. The distance exponent of 2 is chosen because the $1 \rightarrow 1$ studies demonstrated that this value reduced many maximum distances with minor effect of the average values. The memory utilization exponent of 4 is chosen to be larger than the distance exponent to reduce the tendency toward degenerate collapse of the system.

The same purpose could be accomplished by starting from a legal initial mapping and employing only process exchange moves, as in the $1 \rightarrow 1$ study, obviating the necessity of a memory utilization energy term. (Indeed, this is done in section 5.3.2.) For logical graphs with equal process sizes, it is easy to find a legal initial mapping, and process exchanges conserve that legality. The more general method of this study is required if process sizes differ.

4.3.4 Variation in relative memory utilization term weights

The relative weight of the memory utilization and distance energy terms is controlled by an input parameter as described in section 3.10. The initial value for all mappings was chosen arbitrarily as 3. This value sufficed for 19 of the 28 possible mappings, including all cases for toroidal graphs.

Physical graphs lacking translational symmetry tend to have unequally utilized channels. End nodes of preferred physical channels will tend to attract above average numbers of processes, as decreased distance costs will offset increased memory utilization costs. This problem can be remedied by increasing the relative memory weight, repeatedly if necessary.

The Ultracomputer and shuffle-exchange networks each have one case requiring an increase of the initial value. The Sneptree and binary tree require such adjustments in 7 of 8 cases, presumably reflecting congestion near the root bottleneck.

Since the difficulty stems from the asymmetry of the physical graph, it is likely that this adjustment can be estimated automatically by determining the variation in channel

utilization over all shortest paths. Alternatively, this problem can be dealt with directly (but slowly) by using channel congestion costs to penalize overutilized routes.

Of course, in this study with just two energy terms legal solutions can be obtained by choosing a uniform relative weight greater or equal to the greatest case. However, an excessively large weight for an energy term will unnecessarily raise the peak temperature and slow the solution cycle.

4.3.5 Comparison of mapping results

In the following displays, the statistics represent a single mapping for each possible combination of logical and physical graphs.

Average logical distance for 4 → 1 study

Figure 4.11 represents the average logical distance for the mappings. Unlike the 1 → 1 study the average distance does not have an obvious lower bound of 1. A lower bound will be developed in section 5.2.1 and compared to these actual results in section 5.2.2. For the moment, consider the mapping of the n=2, k=22 torus into the n=2, k=11 torus. The best possible embedding would yield an average distance of .5 for an (inverse) display statistic of 2. The actual average distance is .89, giving a displayed statistic of 1.1.

Maximum logical distance for 4 → 1 study

Figure 4.12 displays the maximum physical distance for logically connected processes.

Distribution of logical distances

Star plots of distance distributions for all the mappings are shown in Figure 4.13 As in the 1 → 1 display, the distance scale extends counterclockwise from 0 to 12. Since up to four processes may reside in a single node, distances of 0 are possible. For the embedding of the n=2, k=22 torus into the n=2, k=11 torus, a perfect mapping would have half the connections at distance 0, half at distance 1. In fact, only 23% of the connections are at distance 0, with 66% at distance 1, 10% at distance 2, and the (invisible) remainder at distances 3 and 4.

As in the previous study, the weighted tree → tree mapping is bimodal.

Why trees are bad

Since NET driven by a localized, nonhierarchic cost function, trees are especially troublesome. Each tree vertex is distinguished from its neighbors by its distances from the root and leaves. Correct mappings of similar trees require global knowledge, i.e., information about the position of the root. It would be easy to improve the program's performance on these special cases, since NET does extensive analysis of graph vertex degree to estimate the 'star' lower bound for the average distance, (section 5.2.1). This information could be used to generate initial mappings by a matching of vertex degrees, as is commonly done to test for graph isomorphism [2].

Trees are self-similar. If a particular section of a mapped logical graph is moved vertically on a similar physical tree graph the local energy will be unchanged, unless the section touches the root or leaf nodes. This self-similarity can induce large local minima in the

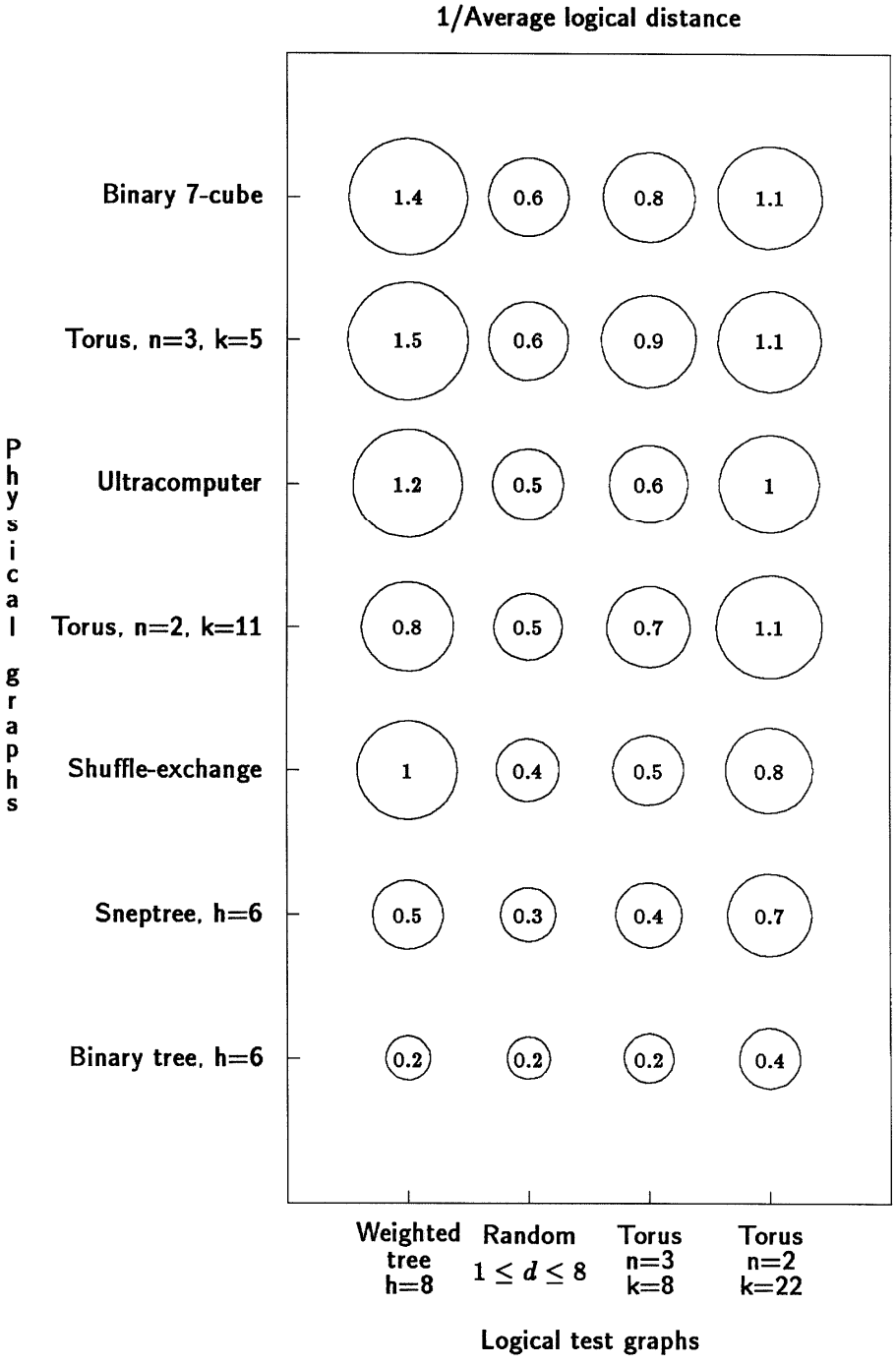

Figure 4.11: Inverse average distances for $4 \rightarrow 1$ series

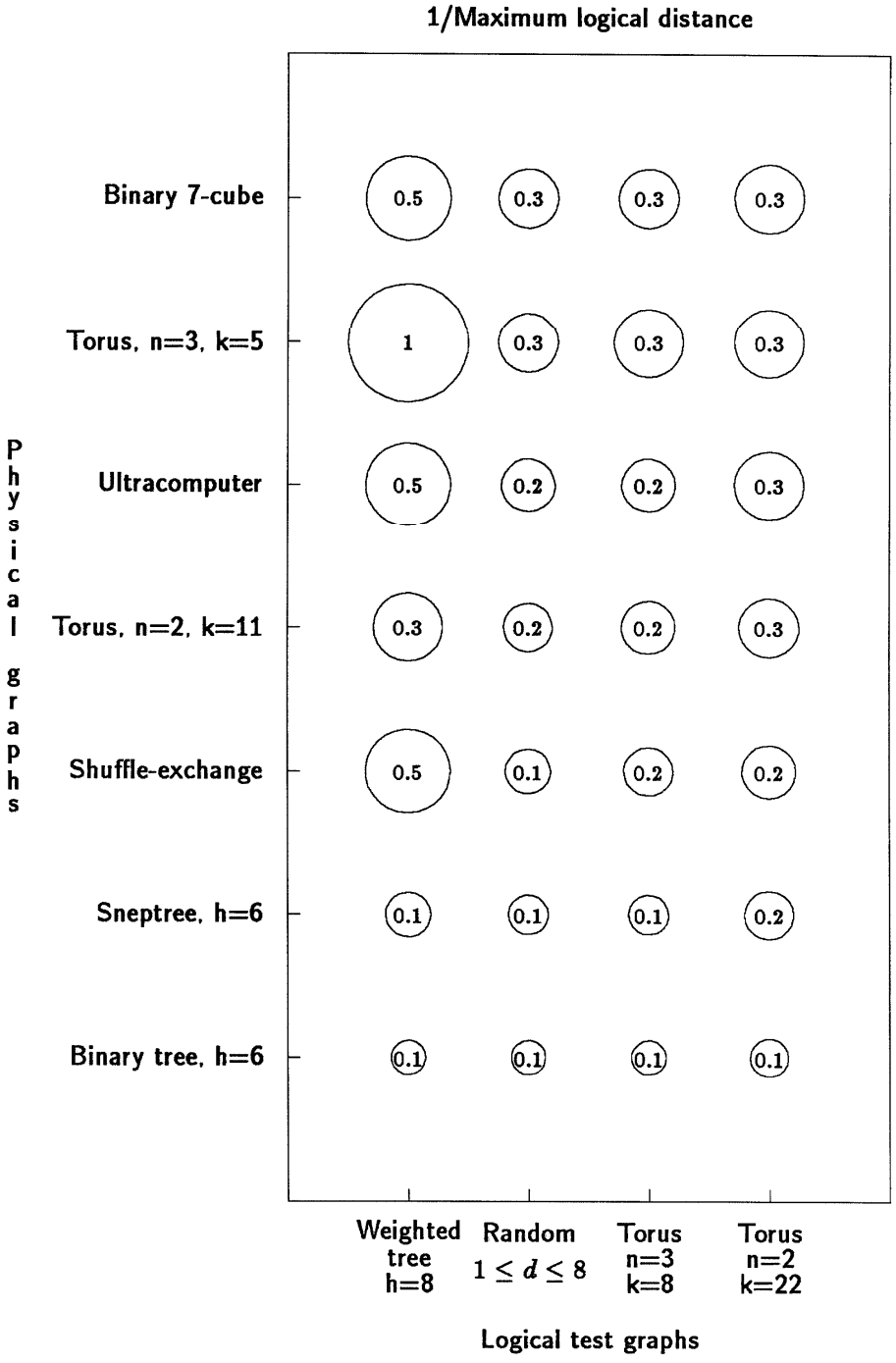

Figure 4.12: Inverse maximum distances for $4 \rightarrow 1$ series

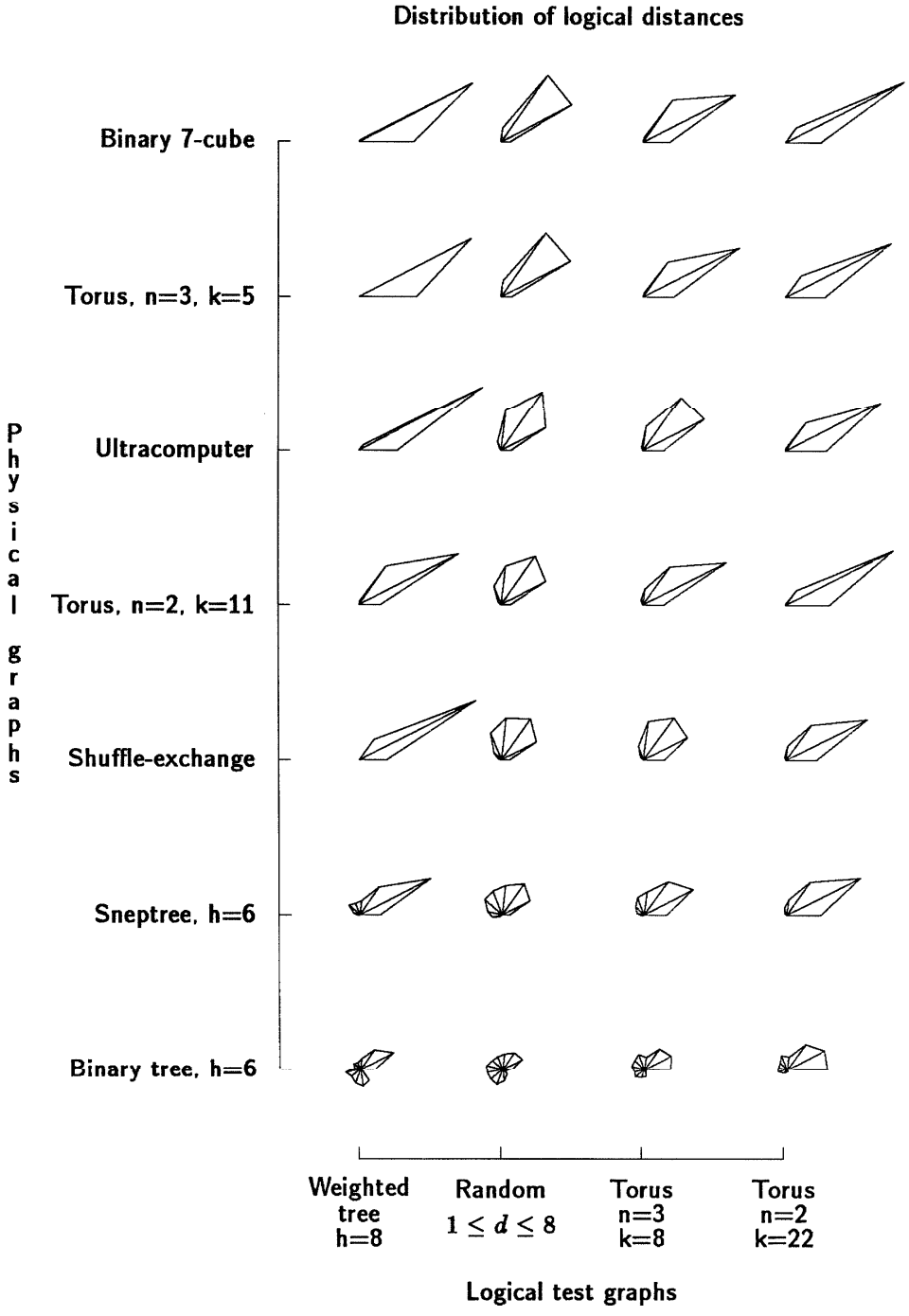

Figure 4.13: Distance distributions for $4 \rightarrow 1$ series

energy function. As an example, consider two identical trees mapped onto each other. If the logical root is placed one level beneath the physical graph root, a relatively good mapping is obtained. All vertices of the tree are perfectly mapped onto one subtree except for the leaves which are forced onto the other subtree. Moving individual logical vertices in any direction will give an equal or higher cost. Without extremely slow cooling, the simulated annealing process will be caught in such a local minimum. This sort of result is likely when the logical graph is weighted more heavily at the root. Evidence for this is seen in a bimodal distance distribution; most logical connections are fairly short, but leaf nodes have strikingly longer connections. A multimodal distance distribution can indicate low suitability of the method for a problem.

Not all mappings involving trees will produce exceptionally poor results, only those where the logical and physical graphs are similar.

4.4 Conclusion

The two mapping studies demonstrate the robustness of this simulated annealing technique for graph embedding. These comparisons are unbiased by manual intervention, and valid comparisons may be made using these data.

Some of the resultant mappings can clearly be bettered by hand. On the other hand, these studies are done with idealized graphs; many real problems can be expected to be less orderly and less amenable to human analysis.

Chapter 5

Analysis of results

... a point whereon little rivers of tabular statements periodically flowed into the howling ocean of tabular statements, which no diver ever got to any depth in and came up sane.

Charles Dickens

5.1 Introduction

Some of the questions naturally raised by the studies of the last chapter are addressed in this chapter, i.e. 'how good are the results?', 'how reproducible and stable are the results?', 'which physical interconnect is best?', and 'can large problems be attacked with NET?'

5.2 The quality of resultant mappings

How good are NET's results? The mappings construct upper bounds for average logical distance. It is desirable to be able to estimate a lower bound without extensive computation.

5.2.1 Estimating a lower bound for average distance

An easily computed estimator should use only local graph information, such as vertex degrees. NET computes a *star lower bound* for the average logical distance by considering only the degree of the logical graph vertices while using somewhat more connectivity information from the physical graph.

Each vertex of the logical graph is considered to be the center of a *ball* with its logical neighbors as the ball's surface. The ball contains the central vertex and its n neighbors (where n is the vertex degree) for a total *volume* of $n + 1$ vertices. Since only radial connections from the central vertex to its neighbors are accounted, this logical subgraph is a star.

In a mapping, the average logical distance is defined as the sum over all logical vertices of the physical distances of edges to logical neighbors. We can see that this is equivalent to the sum of radial distances over all logical stars. (Recall that these 'logical distances' are the physical length of the logical edges for a given embedding.) Indeed, NET computes the distance energy term in this process-oriented manner, using adjacency lists of neighbors.

The problem of finding a lower bound may be rephrased as 'what is the smallest ball in the physical graph containing $n+1$ logical vertices?' Consider the one-process-to-one-node mapping of a two-dimensional torus into a ring. The torus has degree 4, so the relevant logical ball contains a vertex and its 4 neighbors, 5 vertices total. Since each physical vertex is here assumed to contain only one logical vertex, we must find the smallest ring subset containing 5 vertices. For each physical vertex we must find the 4 nearest neighbors. This can be done by scanning the distance table. For the ring, the smallest physical ball will be the central vertex and two vertices on each side. Two of the logical star's four edges will be mapped to adjacent physical vertices; two will be to vertices at physical distance 2. The average logical distance of the star is then $(1+1+2+2)/4 = 1.5$. This is a lower bound since this physical ball is the smallest possible.

Both physical and logical graphs may be irregular. For irregular physical graphs, NET computes minimal balls for all vertices and uses the smallest-radius ball to ensure that the estimate is a valid lower bound. The treatment of irregular logical graphs is more complicated. Consider one of the logical test graphs, a 120-ary tree of height 1 (a star). Each edge is attached to one vertex of degree 120 and another of degree 1. The strongest lower bound will be found if the edge is assumed to be part of the logical ball containing 121 vertices rather than one of 2 vertices. Therefore each edge is assigned an *edge degree* equal to the maximum of its endpoints. For example, in binary trees all edges will be of degree 3.

The average distances for balls of size $n+1$ are weighted by the overall number of edges of degree n, to give the star lower bound for average logical distance.

The tightness of the star bound depends on the graphs involved. For the height 1 trees, the star estimate is, of course, an exact bound. One-to-one mappings of logical graphs of degree lower than the target physical graph will always yield an estimate of 1. Many-to-one mappings tend to have looser bounds than one-to-one cases, since for logical graphs of small degree most of the ball vertices may fit on one node. For example, consider the mapping of the n=2, k=22 torus onto the n=2, k=11 torus, discussed in section 4.3.5. The best possible embedding would yield an average distance of .5; the star lower bound is .25.

The minimum radius of the highest degree ball is a lower bound for the maximum logical distance.

5.2.2 Improvement over a random mapping

The *physical graph average* distance is the average length of all possible paths between vertices in the physical graph (including a zero-length path to self). The physical graph average is a 'probabilistic' upper bound on the average logical distance of a mapping, since it is the approximate value of a random placement of processes. One measure of the quality of NET's computed mapping is how far the mapping's average distance has moved from the random upper bound toward the star lower bound :

$$improvement\ from\ randomness = 1 - \frac{average - star\ lower\ bound}{physical\ graph\ average - star\ lower\ bound}$$

A value of 1 indicates both a tight lower bound and an optimal mapping; 0 a mapping no better than random. This statistic can be expected to remain fairly stable as problems are scaled, and is insensitive to small variation in the tightness of the star lower bound. If the 'star' estimate of lower bound is not tight, the degree of improvement will be un-

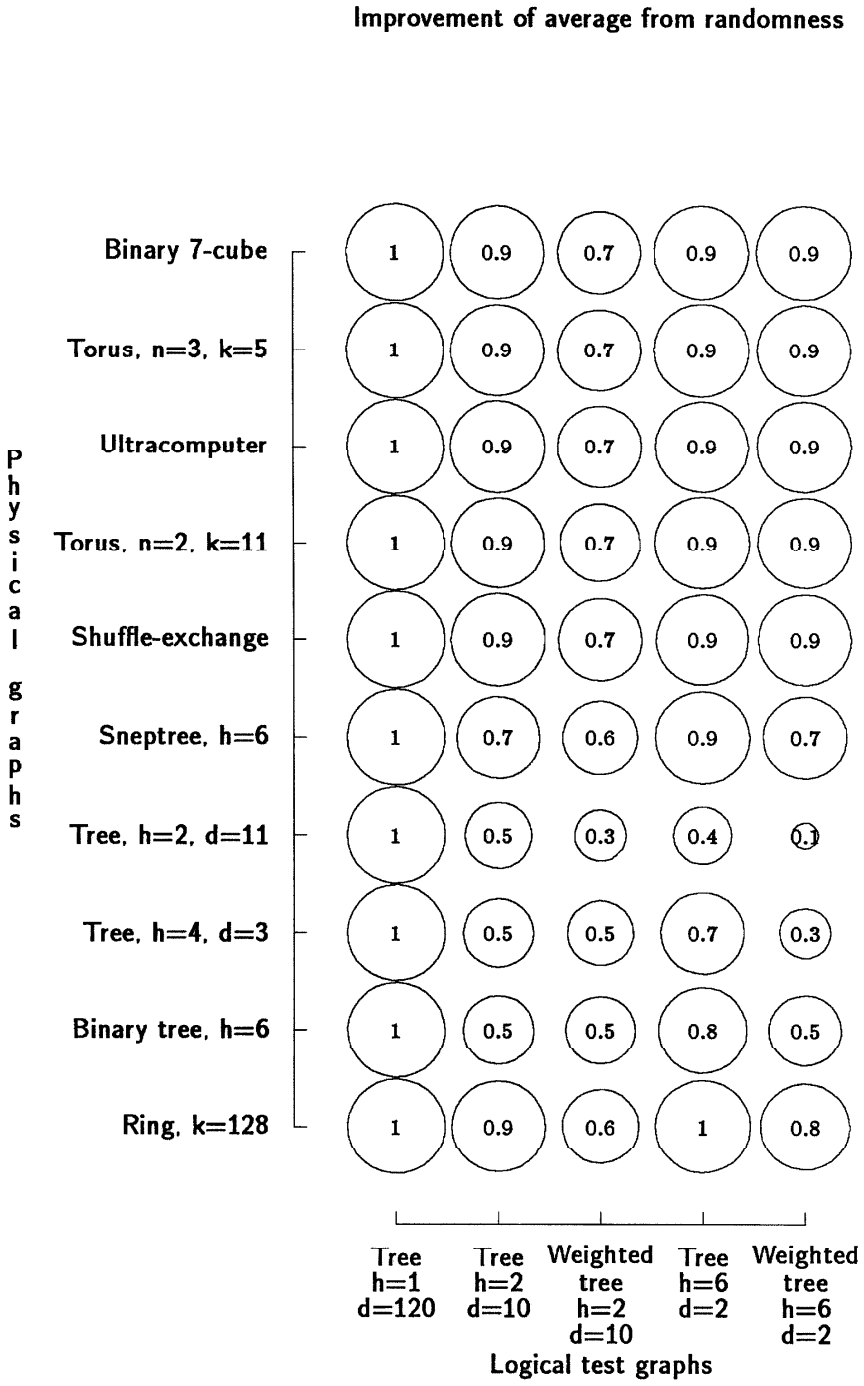

Figure 5.1: Improvement of average distances from random placement to 'star' lower bound estimate, for $1 \rightarrow 1, E = \sum distance^2$ series

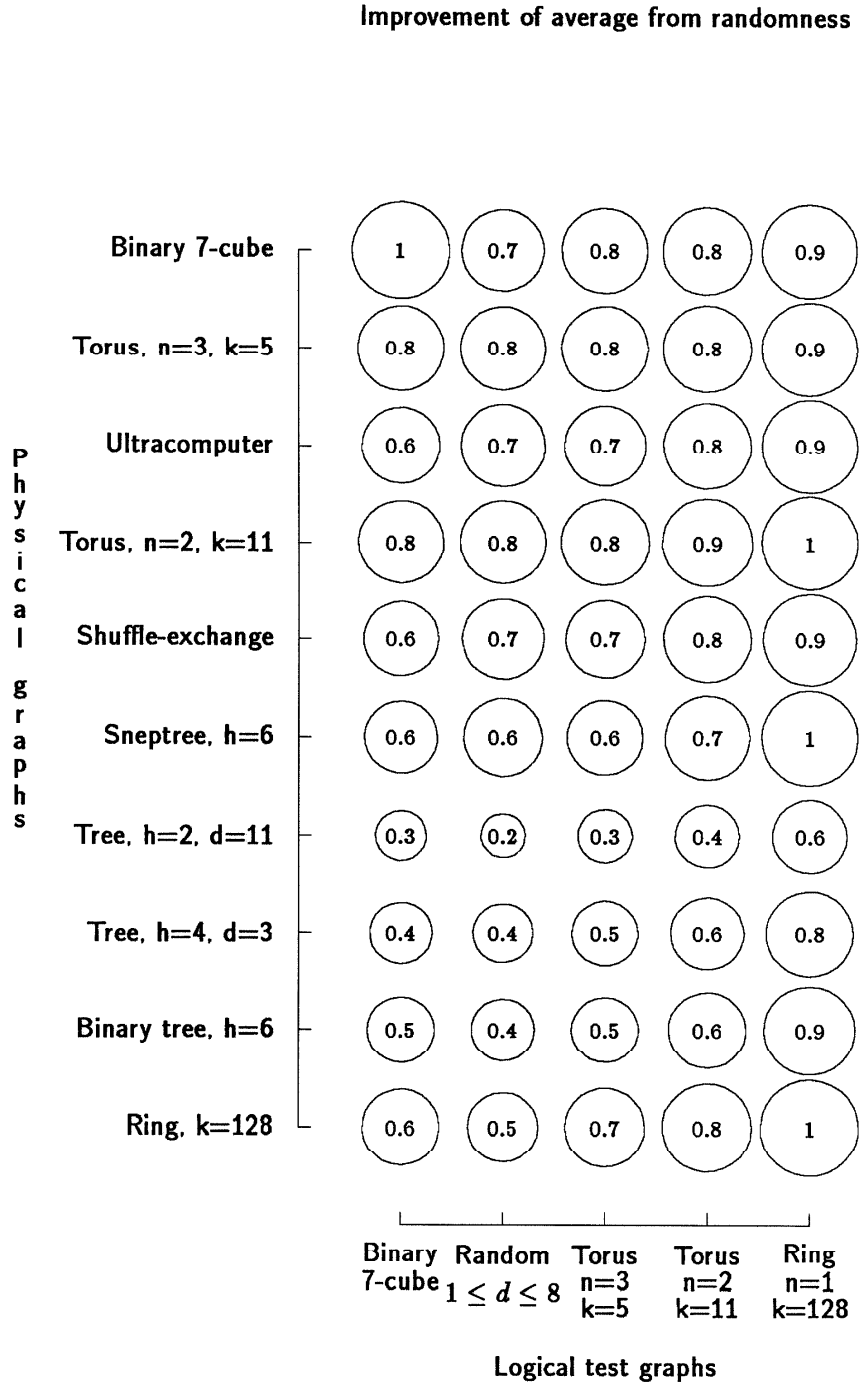

Figure 5.2: Improvement of average distances from random placement to 'star' lower bound estimate, for $1 \rightarrow 1, E = \sum distance^2$ series

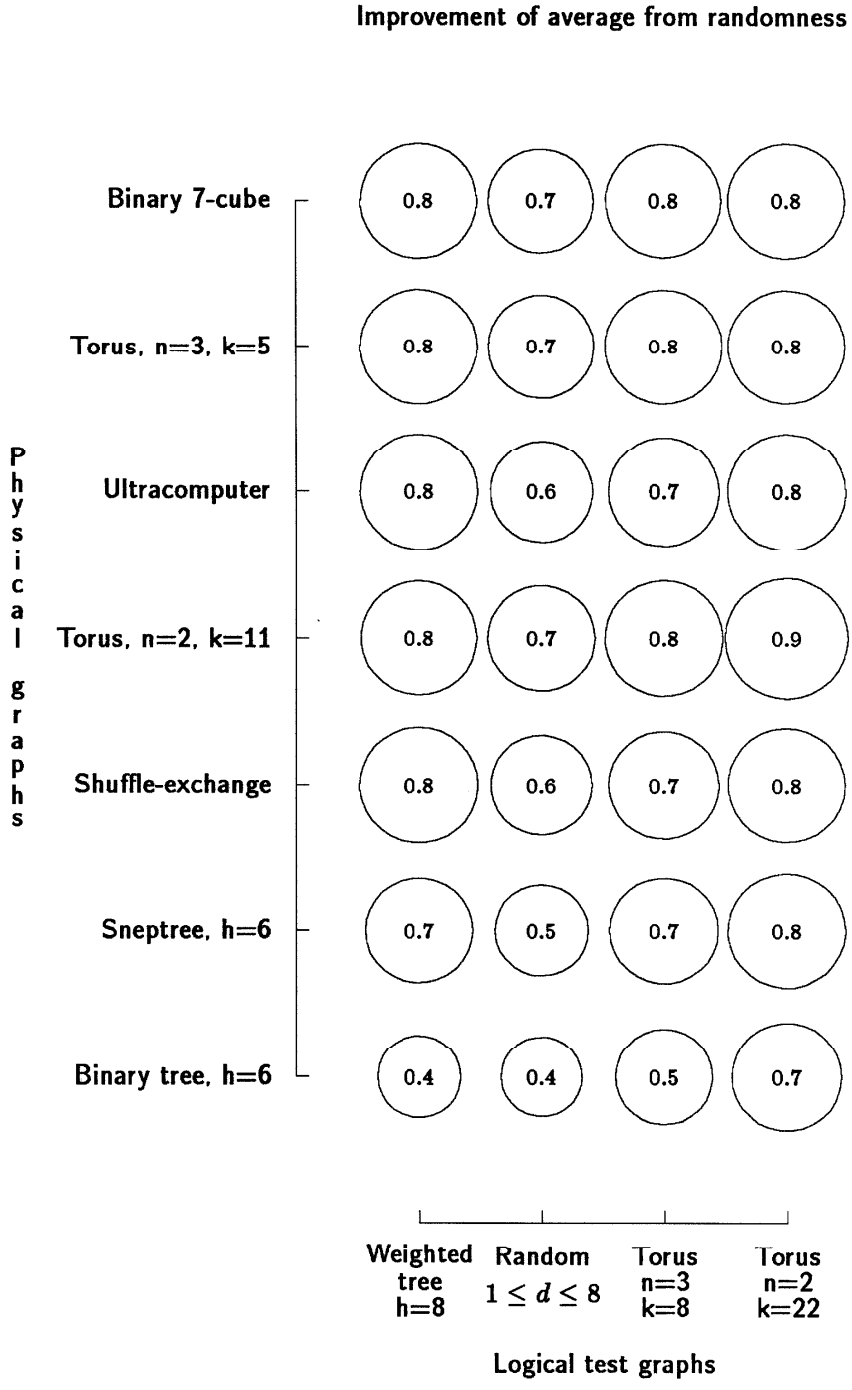

Figure 5.3: Improvement of average distances from random placement to 'star' lower bound estimate, for $4 \rightarrow 1$ series

derestimated, never overstated. The other end of the interval, the physical graph average distance, is exact.

Figures 5.1 and 5.2 show the improvement of the average logical distance from a random placement for the $1 \rightarrow 1$ study. For the 120-ary tree logical graph all mappings find the (exact) lower bound. For physical graphs which are trees the lower bound is generally not tight, since the problem of accommodating logical graph cycles is neglected by the star estimate. In particular, the 11-ary tree of height 2 has a star lower bound of one for for all logical graphs except the 120-ary tree; the low indicated improvement is in part due to the loose lower bound. As noted previously, the use of an unweighted average logical distance is somewhat misleading for weighted logical graphs, since the strongly weighted connections near the root are preferentially optimized to the detriment of other connections. Figure 5.3 shows the improvement for the $4 \rightarrow 1$ study of the last chapter.

The three figures show that NET produces good mappings for most cases, though results for physical tree graphs are by and large mediocre.

5.3 Reproducibility of results

The reproducibility of NET results can be assessed in two ways.

5.3.1 With identical parameters

With identical input (but a different random number sequence), the quality of the results varies by only a few percent unless the annealing is grossly misparameterized. For example, four different mappings of the random graph of size 512 onto an Ultracomputer graph give identical maximum logical distances and only .9% standard deviation of the average even though runtimes differ by as much as 30%.

5.3.2 With different parameters

Linear and quadratic distance cost exponents

Section 4.2.5 examined the effect of changing the distance cost term exponent from 1 to 2, finding that the latter results in a reduction in maximum logical distances, as one might expect. Average values are only slightly changed.

$4 \rightarrow 1$ mappings with soft and hard constraints

The $4 \rightarrow 1$ study of section 4.3 uses a memory utilization penalty to enforce a 'soft' constraint of at most 4 processes per physical node. Since the processes in this study are of equal size, an initial legal mapping is easy to produce. Therefore the constraint can be enforced, without the necessity of considering a memory utilization term, by starting from an initial legal configuration and using only process exchanges as allowed moves. This hard constraint method is demonstrated in the following $4 \rightarrow 1$ study, with all parameters except memory utilization costs the same as the previous (soft constraint) study.

Figure 5.4 compares average logical distances of the two studies. The table shows the fractional improvement of the hard constraint versus the soft constraint result. Negative values indicate the soft constraint result is better; for example the mapping of the weighted tree onto the 7-cube is 20% better using a memory utilization cost term. The hard-constraint

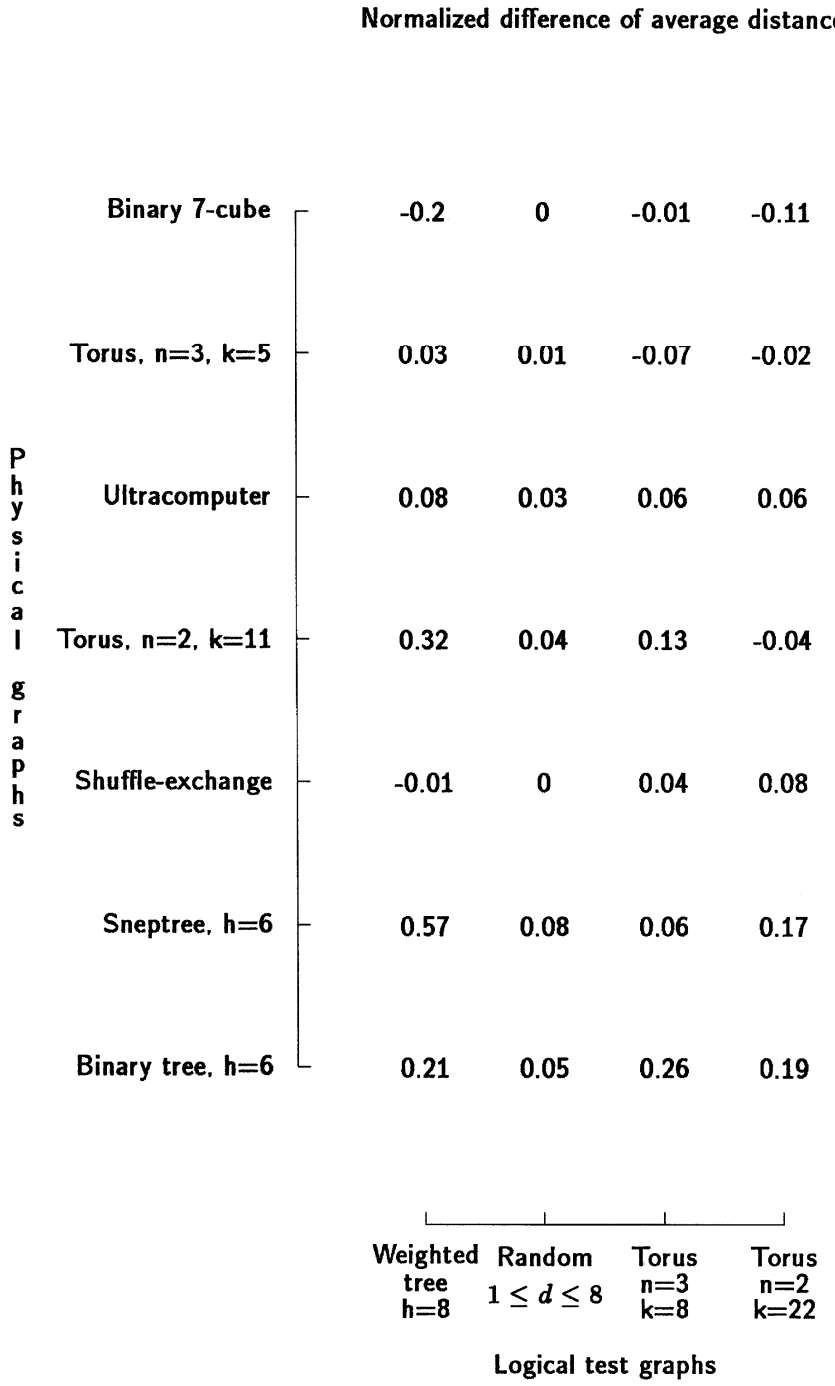

Figure 5.4: Normalized differences in the average logical distance for $4 \rightarrow 1$ mappings, for hard and soft constraints

exchange-only method produced results 7.2% better on the average, with a relatively large 14.5% standard deviation.

Surprisingly, the soft constraint study used only 17% more total CPU time, despite having a much slower move set. The run time for a mapping is largely determined by the estimated relaxation time of the system. However NET looks only for approximate equilibrium of the aggregate energy, which is dominated by the memory utilization term for the soft constraint case. Memory utilization can vary more rapidly than the distance term; therefore the estimate of system relaxation time tends to be less than the true relaxation time of the distance energy term. This effect is most pronounced in relatively poorly connected, low degree physical graphs. For example, the mapping of the weighted tree onto the two-dimensional torus has an estimated equilibration time 4 times greater for the case of the hard constraint, resulting in evaluation of almost 3 times as many configurations and 32% improvement in average logical distance, seen in Figure 5.4. The problem could be ameliorated by considering each cost function term individually to avoid ignoring slowly relaxing terms in favor of quickly equilibrating terms of greater weight.

The error in relaxation time estimation is significant for mappings which tend to induce excessive concentration of processes in nodes. Mappings where both logical and physical graphs are homogeneous have smallest error and show only minor variation in Figure 5.4. Inhomogeneous logical graphs tend to produce strongly bound 'clumps' of processes, but not at any special physical location. Inhomogeneous physical graphs tend to overload some special nodes. When both conditions are true, as in the case of the weighted tree mapped onto the Sneptree, the error is significant.

Though the causes of the variation seen in this comparative table are of technical interest, in general the differences are not large and are fairly predictable.

5.4 Relative merits of physical interconnection schemes

Which physical graphs are better? The scale of the problem is significant. The medium-sized physical graphs of the previous chapter did not evidence dramatic difference in the quality of mappings except for trees. The average physical distances of these graphs are comparable.

5.4.1 Distance distribution of the physical graphs

The ranking of physical graphs by quality of (previous) results agrees with the relative average physical distance for graphs of approximately 128 nodes. However, scaling the graphs to larger sizes changes the order of average physical distances as characteristic scaling laws come to dominate.

Physical graph distances				
Physical graph	$\|V\| \approx 128$		$\|V\| \approx 1024$	
	Avg.	Max.	Avg.	Max.
Binary n-cube	3.5	7	5.0	10
Torus, n = 3	3.6	6	7.5	15
Ultracomputer	4.3	9	7.3	13
Torus, n = 2	5.5	10	16.0	32
Shuffle-exchange	5.5	13	9.0	19
Sneptree	6.1	12	11.6	18
Binary tree	8.3	12	14.1	18

The graphs in the table are those used in the previous $4 \rightarrow 1$ studies and the scaling study to be presented in section 5.5.3.

A minor curiousity is seen in the scaling of the three-dimensional torus and the binary n-cube. Both are members of the family of toroidal networks defined as k-coordinate periodic n-dimensional cubes [12], having a maximal shortest path between nodes of $\lfloor k/2 \rfloor n$. It is interesting to note that a torus of odd k has the same maximum distance as a $k-1$ torus. (Tori with k even have two maximal shortest paths in each dimension, tori with k odd only one.) Thus the $k = 2$, $n = 6$ torus (binary 6-cube) has the same maximum distance as the $k = 4$, $n = 3$ torus (4x4x4 three-dimensional cube) to which it is isomorphic. An approximated doubling of the size of both produces a binary 7-cube with maximum distance of 7 on the one hand, and a 5x5x5 three-dimensional torus of maximum distance 6 on the other. This one-time advantage is seen in the above table; however all larger $n = 3$ tori compare unfavorably to the binary n-cube.

5.4.2 Relative ranking of results

Figure 5.5 shows the now familiar results of the previous chapter's $4 \rightarrow 1$ study, with each result normalized by the best result for that logical graph on any physical graph, i.e. by columns. A value of 1 indicates that the physical graph is as good as any other for that particular test graph, a value of .5 that the average logical distance for that physical graph is twice that of the best physical graph. We see that the physical graphs are listed in descending approximate overall rank. For this scale ($|V| \approx 128$) and logical test set, it is fair to infer (for example) that the three-dimensional torus is twice as good as the Sneptree, and quite comparable to the 7-cube. Figures 5.6 and 5.7 show the similarly normalized distances for the $1 \rightarrow 1$ study of section 4.2.

As the size of the physical graph is scaled, we would expect graphs with logarithmic growth in distances to surpass those with square or cube root characteristics, as indicated by the above table. However, trees cannot be expected to improve for the cyclic logical graphs. Section 5.5.3 examines the effects of scaling on these comparisons.

5.5 Scaling

Does this approach to graph embedding using simulated annealing scale well to larger problems?

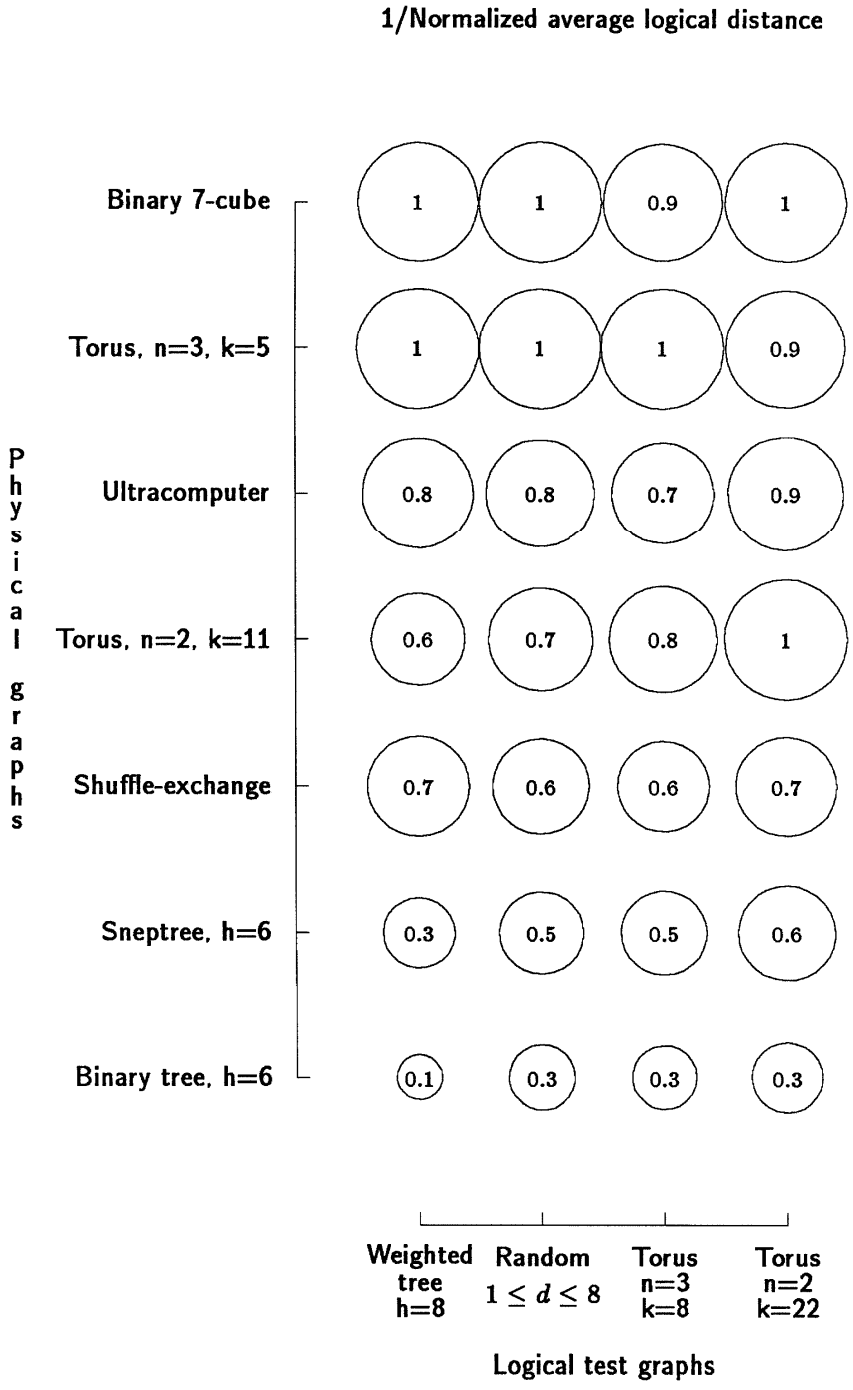

Figure 5.5: Inverse normalized average distances for $4 \rightarrow 1$ series

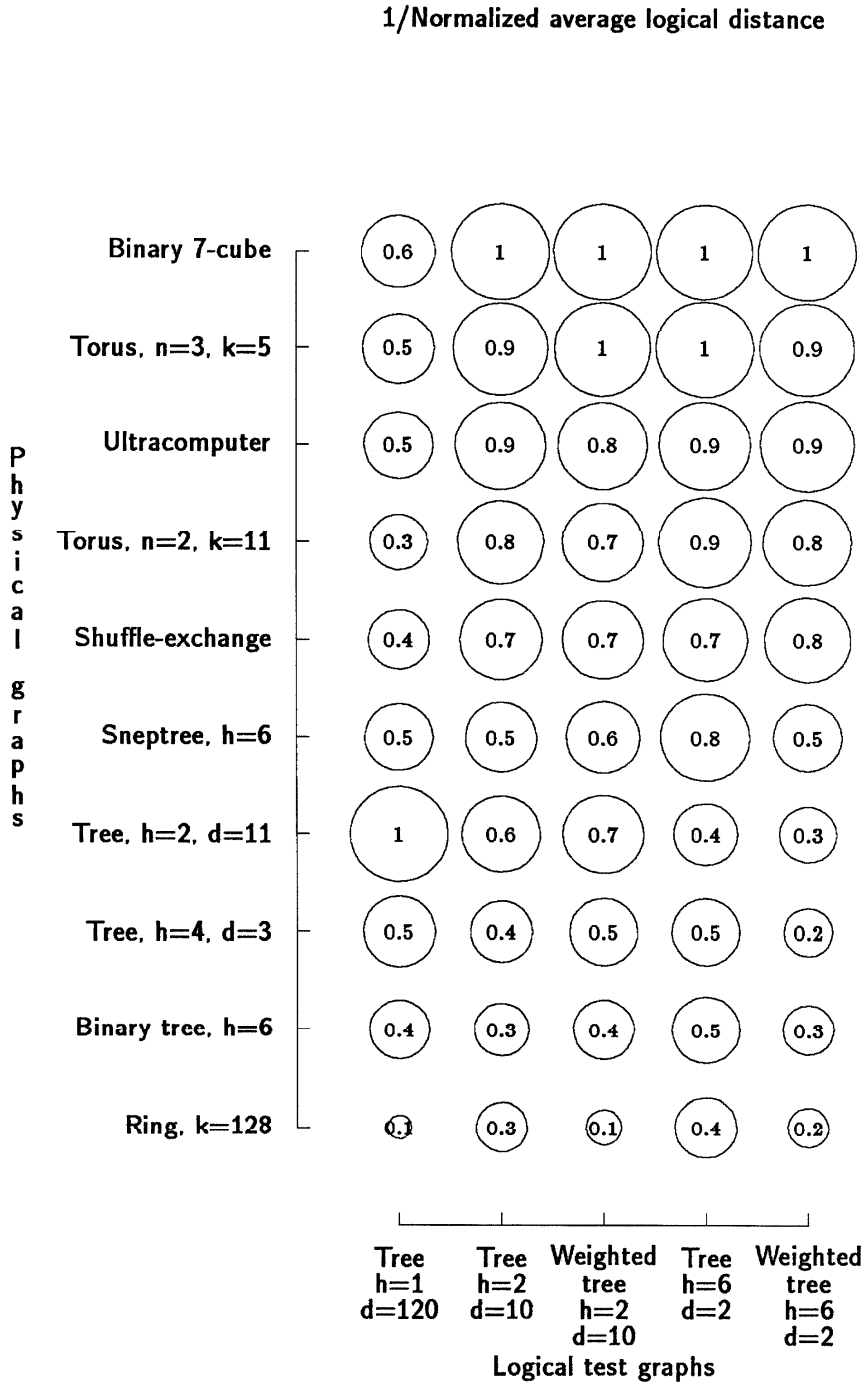

Figure 5.6: Inverse normalized average distances for $1 \rightarrow 1, E = \sum distance^2$ series

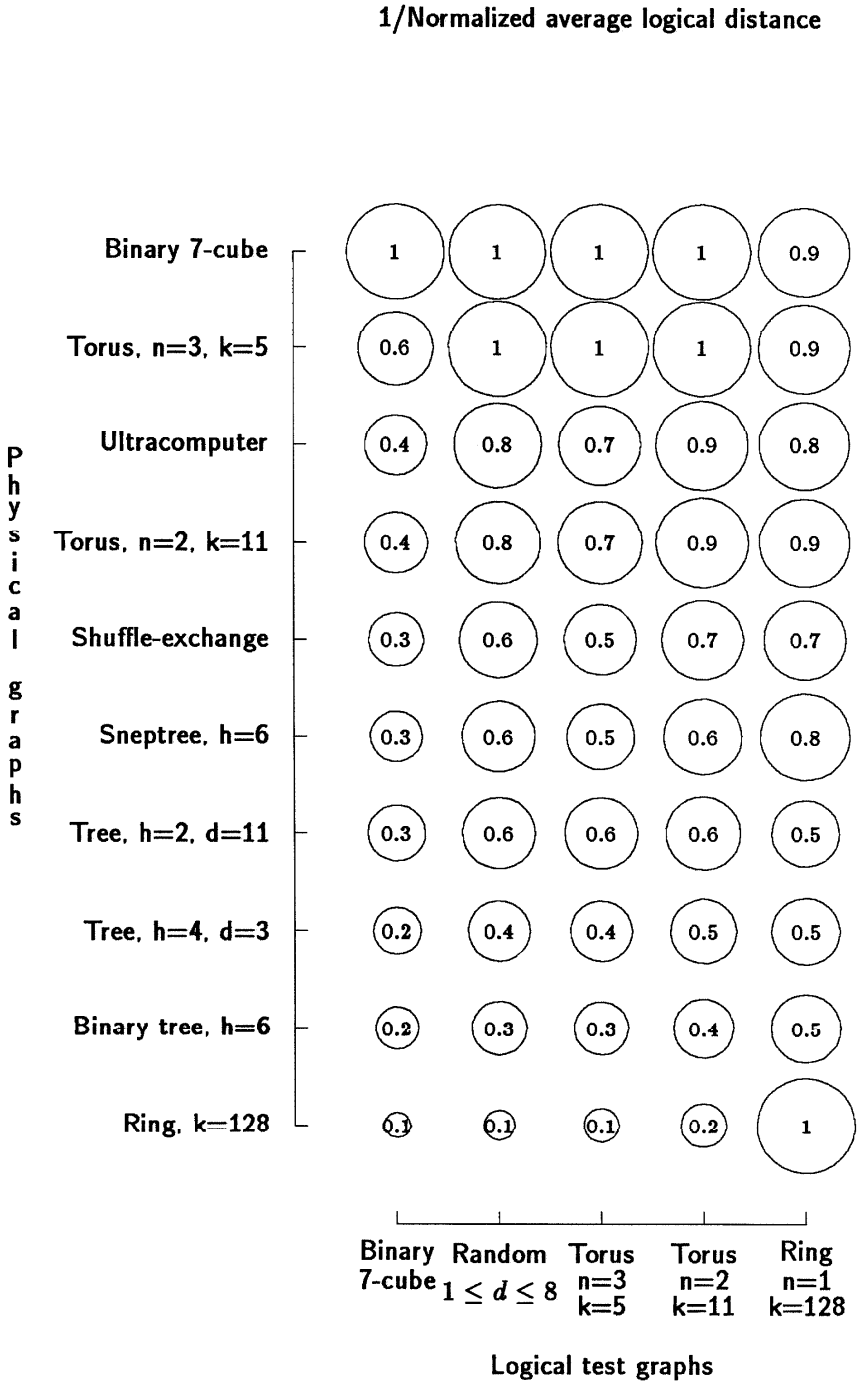

Figure 5.7: Inverse normalized average distances for $1 \rightarrow 1, E = \sum distance^2$ series

5.5.1 Space

As discussed in section 3.11.2, the current implementation limits the size of the physical graph to approximately 2000 vertices due to the size of the physical graph distance table.

5.5.2 Time

With fixed parameters, NET's runtime grows less than linearly for the range ($|V| \approx$ 128, 1024) of systems studied. The factor of 8 size increase produces only a factor of 5.2 increase in runtime. The runtime is determined by the high and low estimated temperatures and the approximate relaxation time. The temperature scale is defined by local interactions, (single move energies), which become relatively constant for a fixed topology as 'edge effects' diminish. The coarsely estimated relaxation time seems to follow the typical system distance, which grows sublinearly except for rings. (The bias in relaxation time estimation discussed in section 5.3.2 does not apply here since only one energy term is used.)

5.5.3 Quality

Increasing problem size with fixed NET parameters yields an increase in average distances, as might be expected given the disproportionately small increase in runtime. To assess the degree of deterioration a large one-process-per-node study was done. Graph sizes approximate 1024 vertices, and are suitably scaled versions of graph types of the $4 \rightarrow 1$ studies.

Figure 5.8 shows the inverse of the average logical distances, which are slightly worse than comparable values of the smaller $1 \rightarrow 1$ study. For comparison purposes, the ratio of the average distance found in each study is shown is Figure 5.9, where a value less than 1 indicates that the smaller study has a better result. Overall, increasing the graph size by a factor of 8 results in a 43% increase in the average logical distance.

Absolutely, the result quality suffers; relative to the average distance of the physical graph (Figure 5.10) it is nearly constant. Figure 5.11 shows the ratio, for the large and small study, of the improvement of the average logical distance from a random placement to the estimated lower bound. Overall, this improvement statistic dropped by only 4% from the smaller to the larger study.

In general, the larger scale hurts the relative ranking of the tree-based physical graphs, while improving the n-cube's status. (The 10-cube is of course more richly connected than a 7-cube.) Figure 5.12 shows the normalized averages which indicate the relative quality of the mappings. The two and three dimensional toroids still compare well with the Ultracomputer and shuffle-exchange graphs; the logarithmic advantage has not materialized for the logical graphs of this test set.

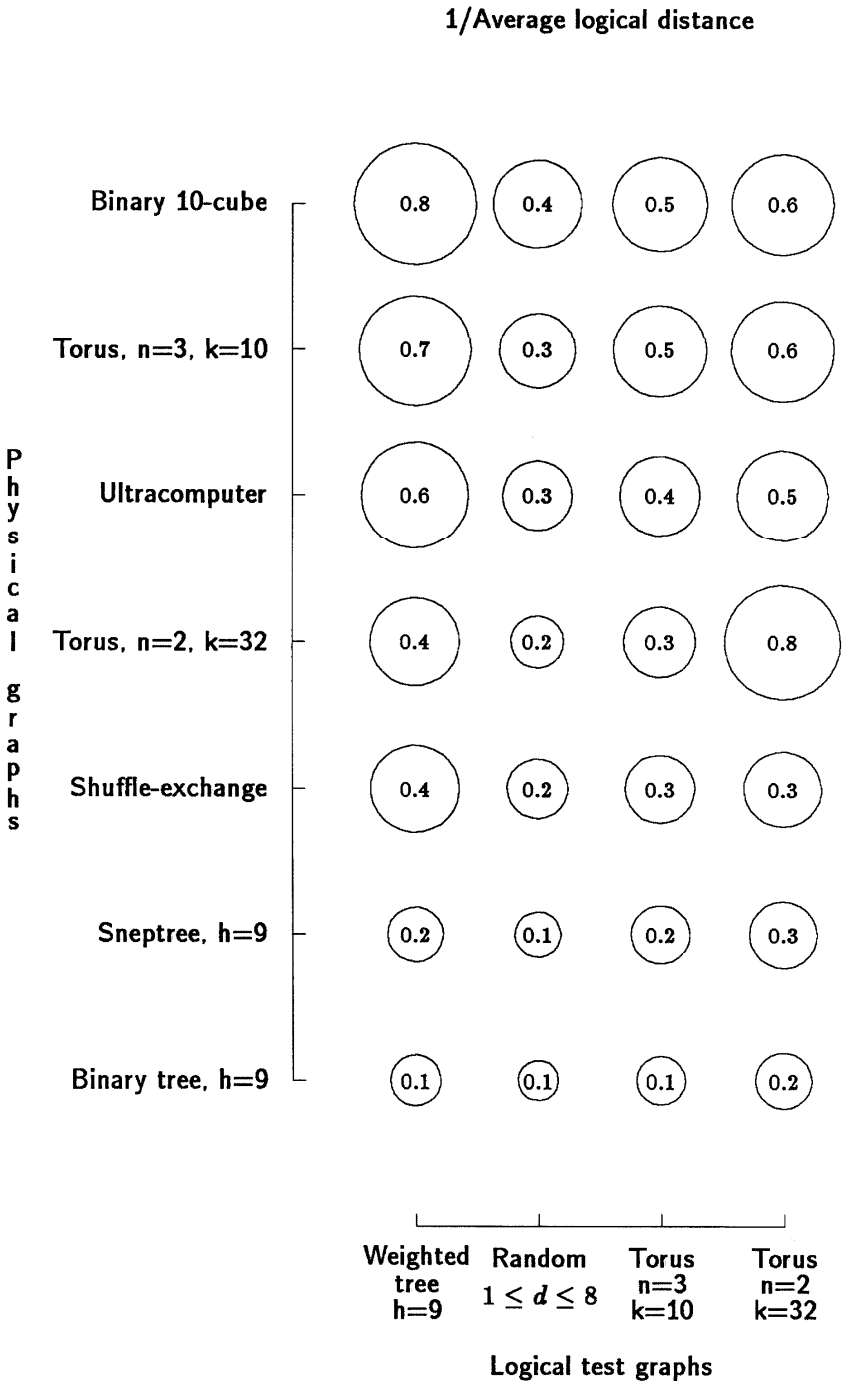

Figure 5.8: Inverse average distances for $1 \rightarrow 1$, $|V| \approx 1024$, $E = \sum distance^2$ series

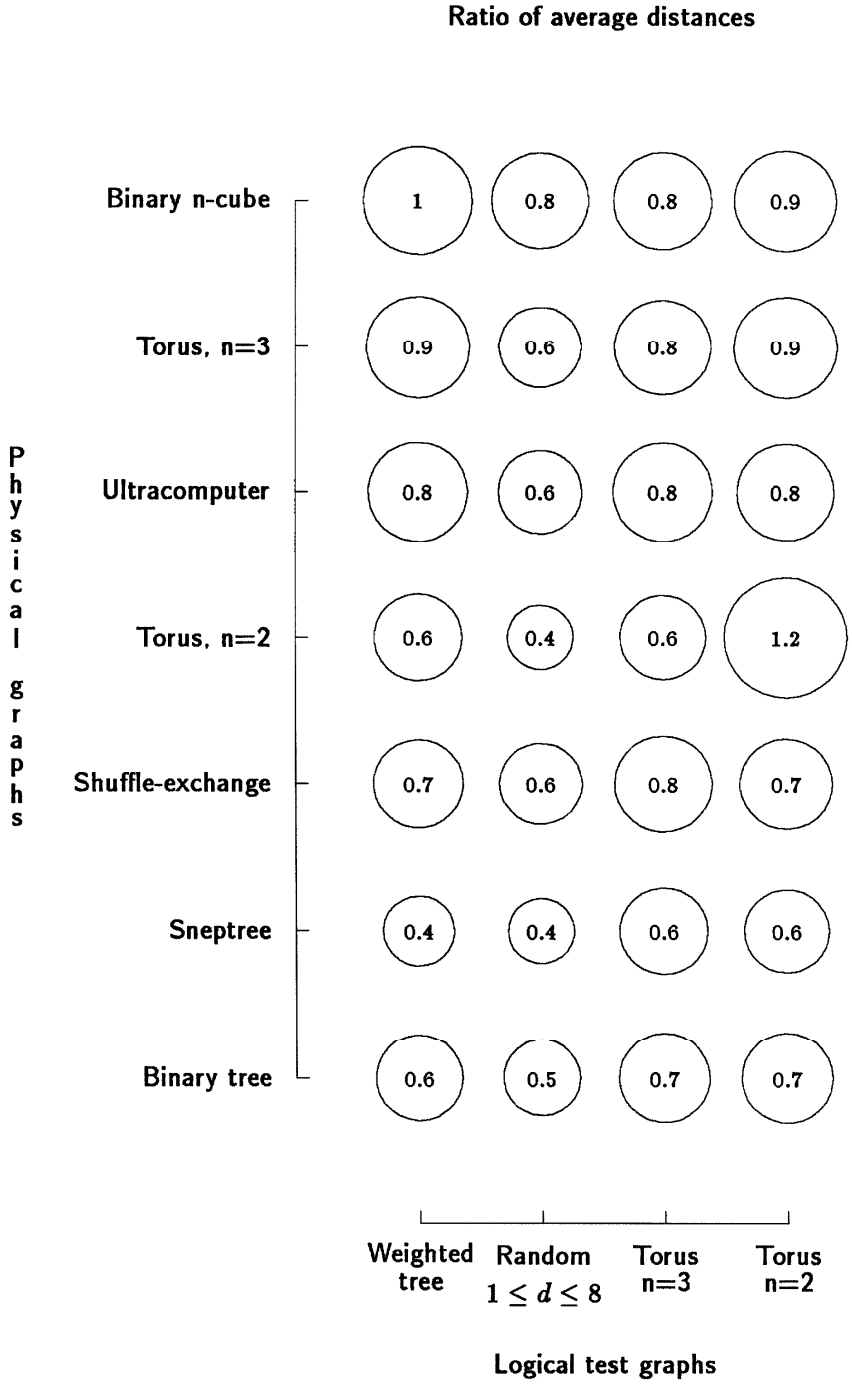

Figure 5.9: Ratio of average distances for $1 \rightarrow 1$, $|V| \approx 128$ and 1024, $E = \sum distance^2$ studies

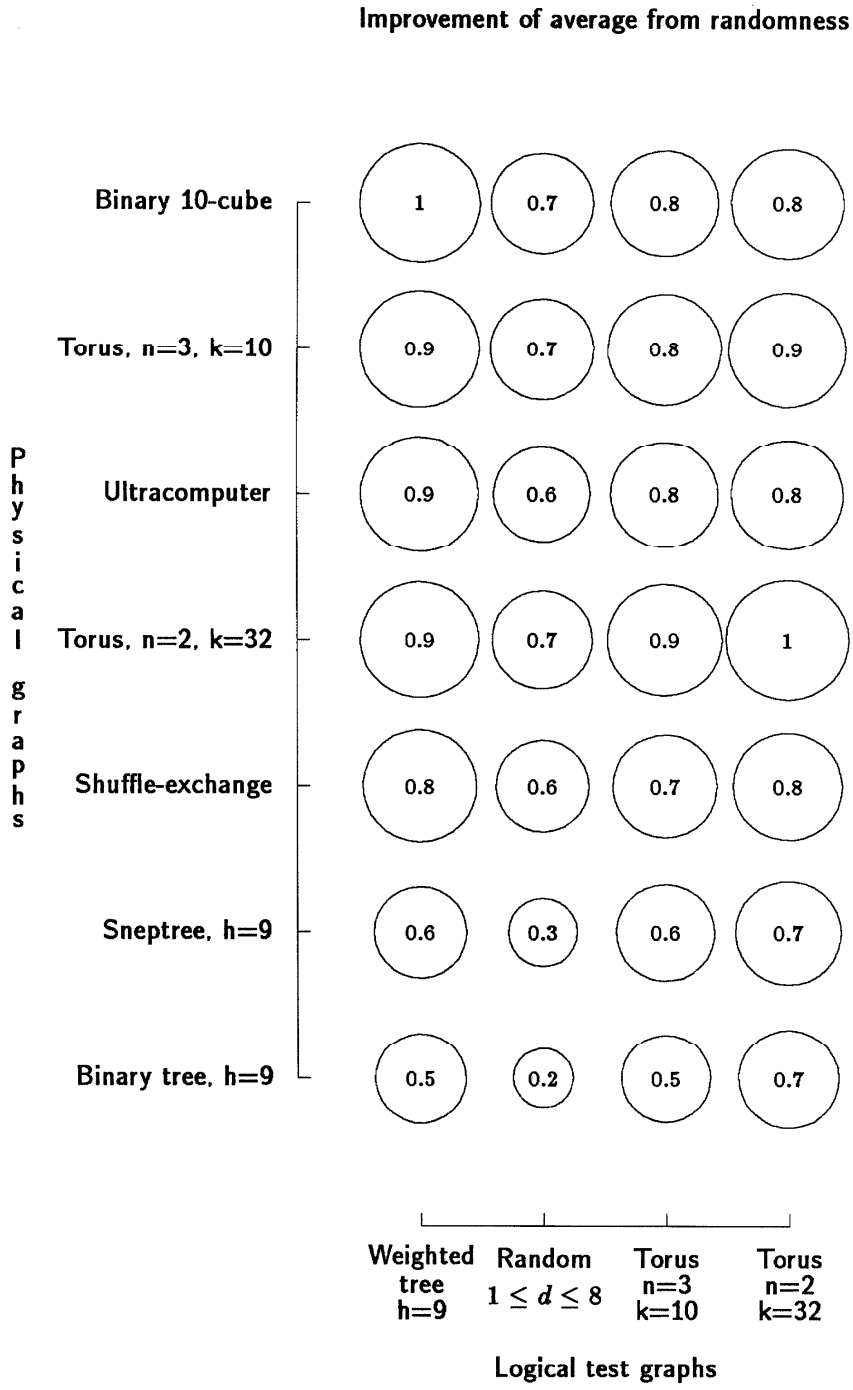

Figure 5.10: Improvement of average distances from random placement to 'star' lower bound estimate, for $1 \rightarrow 1$, $|V| \approx 1024$, $E = \sum distance^2$ series

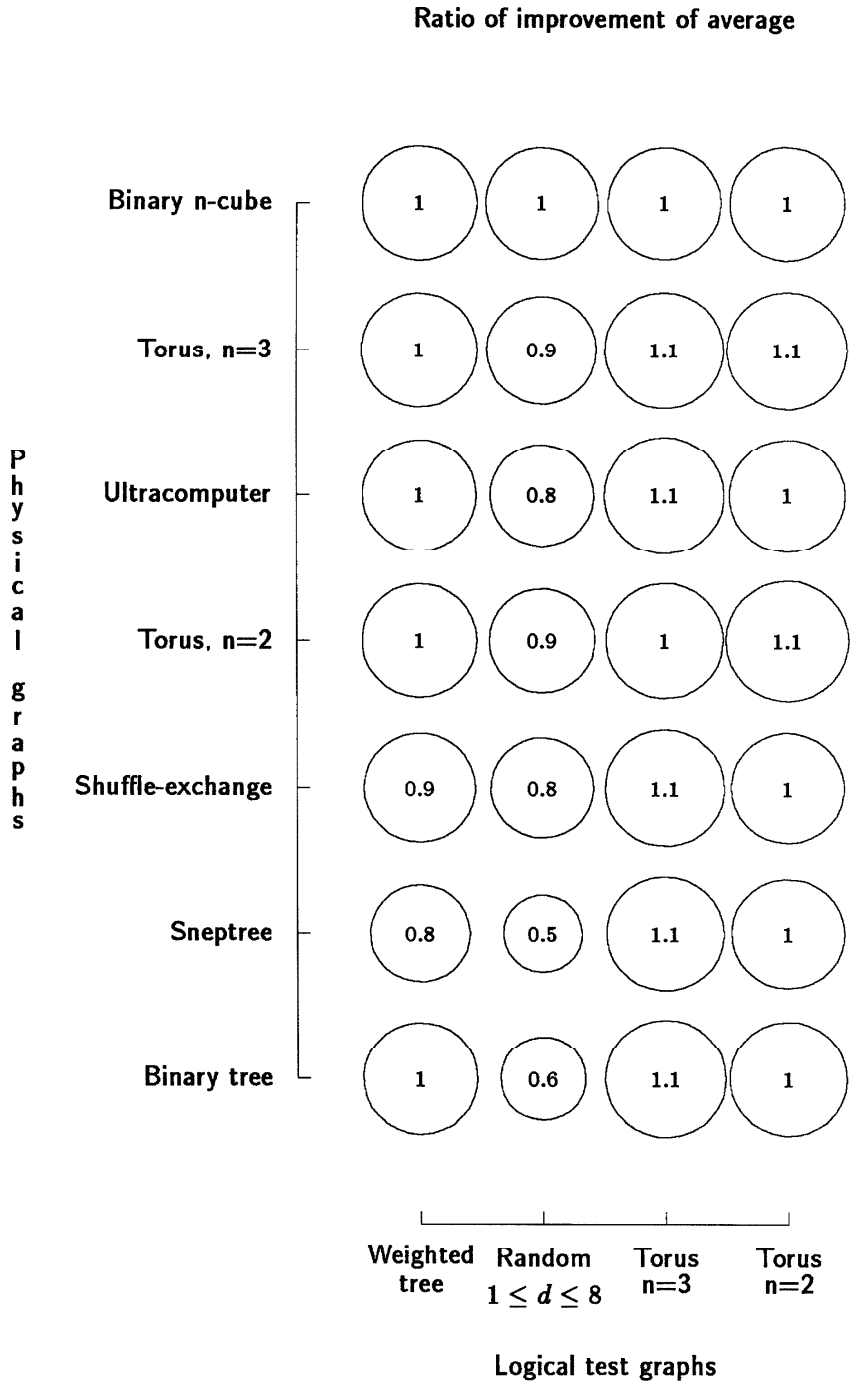

Figure 5.11: Ratio of improvement of average distances from random placement to 'star' lower bound estimate, for $1 \rightarrow 1$, $|V| \approx 128$ and 1024, $E = \sum distance^2$ studies

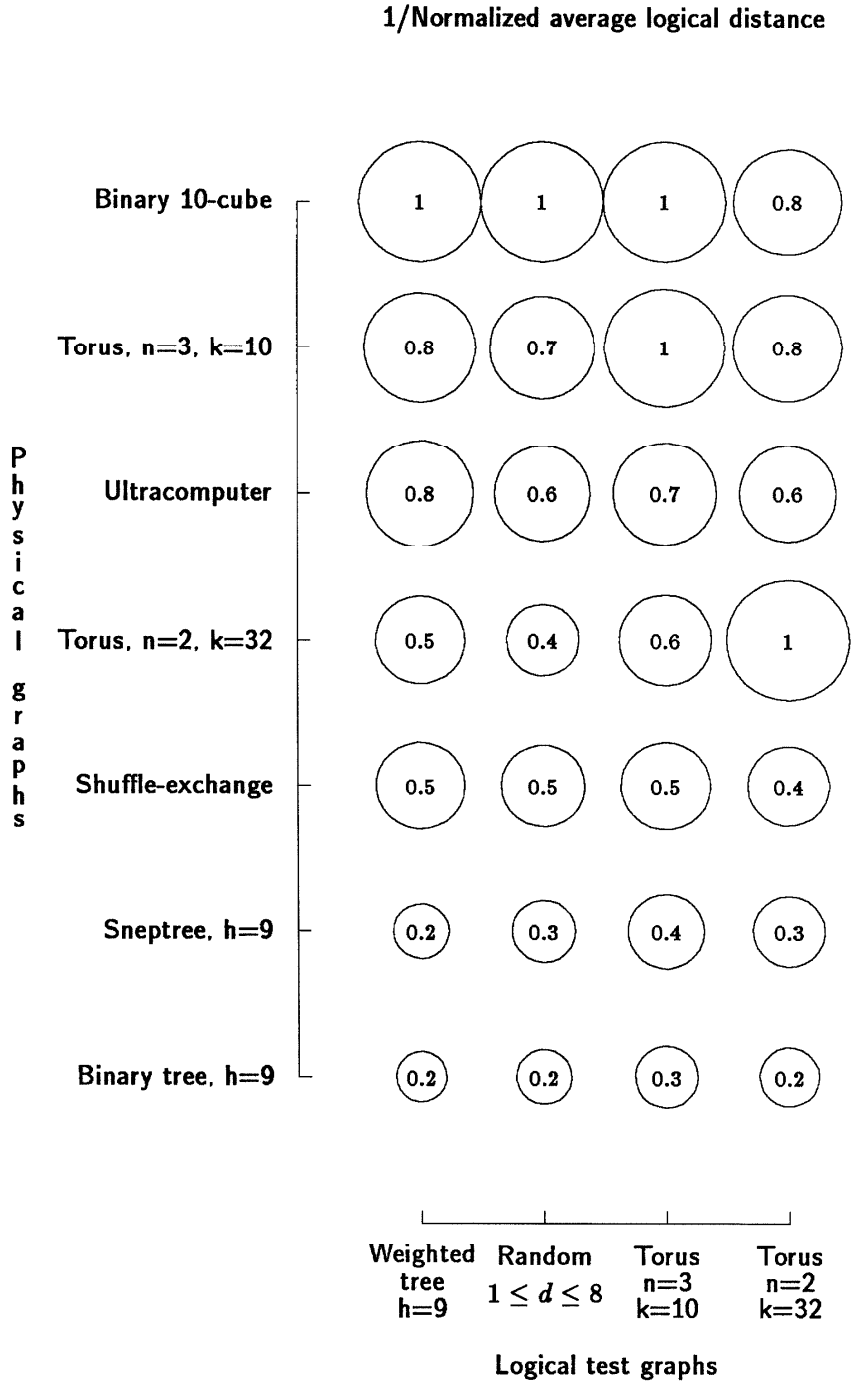

Figure 5.12: Inverse normalized average distances for $1 \rightarrow 1$, $|V| \approx 1024$, $E = \sum distance^2$ series

Chapter 6

Conclusions and future research

Mankind always sets itself only such problems as it can solve ...

Karl Marx

6.1 Conclusions

The program NET is an effective tool for producing reasonably good solutions for graph embedding problems whose exact solution is intractable. For example, averaged over all cases of the largest-scale study ($|V| \approx 1024$), NET achieves 72% of the estimated possible reduction of the average logical distance from random placement.

Good results are obtained for a wide variety of logical and physical graphs. However, poor results are obtained in mapping of trees to similar (and self-similar) trees, due to locally optimal mappings of subtrees. This can be expected of any optimization scheme that does not use global information.

6.1.1 Hard and soft constraints

The method given for constructing a hierarchy of constraints and goals for graph embedding problems allows quite varied applications. Constraints can be implemented effectively both by cost function penalties (soft constraints), and restriction of the configurations generated (hard constraints). Both methods work well for homogeneous logical and physical graphs. Inhomogeneous graphs occasionally require alteration of the relative weight of cost terms when soft constraints are employed. Possible program improvements to address this inconvenience are suggested in section 6.2.4.

6.1.2 Scaling to large problems

The introduction of the 'star' lower bound estimate helps to demonstrate that the method scales reasonably well to larger, more difficult problems. Result quality is fairly constant relative to problem difficulty. In particular, scaling the one-process-per-node study by a factor of 8 results in only a 4% reduction in the improvement of the average logical distance from random placement to the estimated lower bound. In absolute terms, this increase in graph size produces only a 43% increase in average logical distance, with program runtime increased by a factor of 5.

In the present implementation, memory requirements restrict the practical problem size to physical graphs of about 2000 nodes. Logical graphs may be much larger.

6.1.3 Preferred physical interconnection graphs

As judged by the average distance for logical communications paths, the overall quality of n-cube, toroidal mesh, and shuffle-exchange type graphs is surprisingly close for the test problems considered. Even for the most difficult problem (random graph) at the largest scale ($|V| \approx 1024$, section 5.5.3) the average logical distance differs only by a factor of 2.8 from the worst (two-dimensional torus) to the best physical graph (10-cube).

Increasing the physical graph degree beyond that of typical logical graphs may not be cost-effective, as the following table shows. For all physical graphs, the average logical distances are normalized by the the best result for any physical graph (as discussed in section 5.4.2) to indicate relative performance (as previously shown in figures 5.6, 5.7, and 5.12). An average of the normalized results for the four logical graphs of the $|V| \approx 1024$, $1 \rightarrow 1$ study are shown.

Physical graphs of higher degree are more difficult and costly to wire. A reasonable figure of merit for overall attractiveness of a physical interconnection graph is the product of typical degree (maximum number of physical channels per node) and the average of normalized average logical distances.

Wiring-cost degree-distance product			
Physical graph	Typical degree	Normalized distance	Product
Binary n-cube	10	1.07	10.7
Torus, n = 3	6	1.21	7.3
Ultracomputer	4	1.49	5.9
Torus, n = 2	4	1.86	7.4
Shuffle-exchange	3	2.03	6.1
Sneptree	4	3.52	14.1
Binary tree	3	4.72	14.2

For this relatively low dimension logical test set, the richness of 10-cube connections is excessive. The 10-cube and binary tree are comparable in the overall figure of merit, but at opposite ends of the spectrum of cost and performance. The inhomogeneous connection patterns of the Ultracomputer and shuffle-exchange graphs induce uneven communications path loading, unlike the toroidal meshes, a factor not considered here.

6.2 Future work

Brazil is the country of the future, and the future never arrives.

Brazilian folk saying

6.2.1 Parallelism

The current NET program is demanding of both memory space and CPU time. A multiple-process concurrent reformulation is an attractive goal. Process memory size could be reduced to fit small processor nodes by replacing the physical graph distance look-up table by graph-specific procedures.

Even if the processes fit, there may be a subtle tradeoff between interprocess communication costs and the convergence rate of the algorithm. The use of greater than linear distance cost functions is advantageous in reducing maximum distances. However, for a concurrent implementation to maintain a completely consistent data base of process positions limits the possible concurrency. It would be highly desirable for the formulation to tolerate error in the position of physically remote processes. This is highly unlikely with a quadratic or even linear distance energy term. For this reason, it is desirable to conduct experiments with sublinear exponents to determine the impact of this 'decoupling of distant regions' on convergence rates.

6.2.2 Application of NET to a concurrent programming environment

Logical channel labels

Many concurrent programming models are conceptually presented as a regular graph of similar (if not identical) processes. Labelling the logical edges would allow the programmer to use port names for logical communication channels, with an operating system service provided to translate the logical name into the physical address of the corresponding process. Directed edges are naturally required for this purpose, since the *east* port of one process may be the *west* port of another.

Creation of a placement utility

An interface program for NET which generates process loading commands for the Caltech Cube would be useful. The long runtimes of the program in the comparison studies (averaging 4.4 VAX-11/750 CPU-hours for the 1024 vertex $1 \rightarrow 1$ study) are due to conservative parametric choices intended to apply the method beyond any critical point of diminishing return. Shorter runtimes will yield adequate results for many purposes; long runtimes are justifiable for some long-running applications. For common logical structures, standard mappings can be produced and looked up.

6.2.3 Congestion costs

The previous comparative physical graph studies should be redone with node and channel I/O congestion costs calculated. Imbalances in utilization could be important when trying to assess the merit of shuffle-exchange type graphs relative to homogeneous tori.

It would be interesting to evaluate NET's performance when applied to some traditional routing problems, which should be possible with a cost function that includes congestion terms.

6.2.4 Explicit distinction of constraints

Though it is esthetically appealing to treat goals and constraints equivalently in the cost function, some cases require adjustment of NET parameters to enforce the constraint. If

constraints were explicitly identified, the program could alter its initialization procedures to ease the problem by using extremal rather than 'typical' values to estimate relative energy term coefficients.

It might also be advantageous to automatically iterate the annealing process, with new energy coefficients, until the constraint is satisfied. It would be possible to check this when the relevant energy term 'freezes', rather than at the completion of a complete annealing cycle. This iterative procedure may be necessary for completely automatic functioning even if extremal energy terms are used in the coefficient estimation. NET cannot necessarily predict the values needed to satisfy the constraint since the initial mappings of heterogeneous logical and physical graphs may not have a conjunction of 'problem areas'.

6.2.5 Individual energy term relaxation times

As discussed in section 5.3.2, the estimate of system relaxation time should be computed on a term-by-term basis. A related improvement would allow alteration of the basic time interval as individual energy terms are 'frozen'.

6.3 Acknowledgements

Them as counts counts moren them as dont count.

Riddley Walker, Russell Hoban

I would like to thank Scott Kirkpatrick for originating the technique of simulated annealing. Special thanks are due my present advisor, Charles Seitz, for his patient encouragement and textual 'whichhunting', and to John Platt, Ricky Mosteller, Sven Mattisson, Bill Dally, and Roberto Suaya for their helpful conversations and suggestions. Alain Martin deserves credit for developing the graph mapping paradigm. V. A. Teague steered the verbiage toward present tense. Bill Athas looked at all my volumes of graphs; another superhuman task was performed by Hamlet, without whose ponderous aid none of this would have been possible.

Bibliography

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms*, Addison-Wesley, Reading, MA, 1983.

[2] N. Deo, *Graph Theory with Applications to Engineering and Computer Science*, p. 311, Prentice-Hall, Englewood Cliffs, 1974.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[4] S. Geman, D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images", presented at the Workshop on Statistical Physics in Engineering and Biology, IBM Watson Research Center, April 26-27, 1984.

[5] S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi, "Optimization by Simulated Annealing", *Science*, Vol. 220, No. 4598, 13 May 1983.

[6] S. Kirkpatrick, "Optimization by Simulated Annealing: Quantitative Studies", unpublished IBM report, 1983.

[7] M. Lundy and A. Mees,"Convergence of the annealing algorithm", Workshop ...

[8] A. J. Martin & J. L. A. van de Snepscheut, "Networks of Machines for Distributed Recursive Computations", Caltech Computer Science Dept. Technical Report 5147, July 1984.

[9] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, *J. Chem. Phys.* **21**, p. 1087, 1953

[10] D. J. Johnson, "Optimization by simulated annealing: An experimental evaluation", Workshop ...

[11] J. T. Schwartz, "Ultracomputers", *ACM Trans. Programming Languages & Systems*, Vol. 2, No. 4, pp. 484-521, October 1980.

[12] C. L. Seitz, "Concurrent VLSI Architectures", *IEEE Transactions on Computers*, Vol. C-33, No. 12, December 1984.

[13] C. L. Seitz, "The Cosmic Cube", *Communications of the ACM*, Vol. 28, No. 1, January 1985.

[14] S. R. White, "Concepts of scale in simulated annealing", *IEEE Int. Conf. on Computer Design : VLSI in Computers*, p. 646, 1984.