Physics and the New Computation

Paul Vitányi*

CWI and University of Amsterdam

Abstract. New computation devices increasingly depend on particular physical properties rather than on logical organization alone as used to be the case in conventional technologies. This has impact on the synthesis and analysis of algorithms and the computation models on which they are to run. Therefore, scientists working in these areas will have to understand and apply physical law in their considerations. We discuss three cases in some detail: interconnect length and communication in massive multicomputers which depend on the geometry of space and speed of light; energy dissipation and reversible (adiabatic) computation which depend on thermodynamics; and quantum coherent parallel computing which depends on quantum mechanics.

1 Introduction

In a sequential computation such as performed by a Turing machine or a von Neumann architecture computer, one can safely ignore many physical aspects of the underlying computer system and analyse the computational complexity of a program in a purely logical fashion. In the realm of nonsequential computation one cannot ignore the reality of the physical world we live in to such an extent. The appropriateness of the analysis may stand or fall with the account taken of physical reality. Moreover, nonclassical or nonstandard physical realizations of computers may have totally unexpected properties.

A popular model to analyse parallel algorithms is the parallel random access machine (PRAM) were many processors can read and write a single shared memory in unit time per operation. Typically, for n inputs we have $p(n)$ processors and clever algorithms are designed which, say, add n numbers of n^{ϵ} bits in $O(\log n)$ parallel time (the longest chain of operations executed by any single processor in the lot). However, something is wrong here. Since $p(n)$ processors are necessary and sufficient for the algorithm, we cannot dispense with any one of them, and hence the results of the calculations of each pair of processors must interact somewhere. This means that we have to signal between each pair of processors, and, taking the outermost ones, the distance between them is

* Partially supported by the European Union through NeuroCOLT ESPRIT Working Group Nr. 8556, and by NWO through NFI Project ALADDIN under Contract number NF 62-376 and NSERC under International Scientific Exchange Award ISE0125663. Address: CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands. Email: paulv@cwi.nl

$\Omega(p(n)^{1/3})$ because of the geometry of space. Hence the time required for interaction is $\Omega(p(n)^{1/3})$ by the bounded speed of light. That is, what we called 'parallel time' is in fact a series of 'consecutive steps' were the length of each step depends on physical considerations. A similar problem of relation between the theoretical model and physical realization occurs with NC networks (polynomial number of processors and polylogarithmic depth).

In fact, optimality of PRAM algorithms may be misleading, because in any physically realizable machine architecture it may be the case that a much simpler and unsophisticated algorithm outperforms the optimal PRAM algorithm. Do networks help with this problem? We can simulate PRAMs fast by networks of processors communicating by message passing at the cost of a multiplicative slowdown square logarithmic in the number of processors n for simulation on a $\log n$-dimensional hypercube, [Upfal and Wigderson, 1987]. This doesn't solve the problem mentioned above, since the hypercube nodes need to be order $n^{1/3}$ apart for the majority of pairs (see below). Together it turns out that rather than saving time, the simulation costs at least a logarithmic in n factor more time than the original. Rolf Landauer, [Landauer, 1991] has emphasized "information is physical". So is communication.

At the outset of high density electronic chip technology (VLSI = Very Large Scale Integration), a flurry of activity in analyzing computational complexity focussed on the AT^2 measure, where A is the total (two dimensional) chip area and T is the time (maximal number of transitions or steps of any component on the chip). Typically, up to a polylogarithmic factor the results say $AT^2 = \Omega(n^2)$ for many problems (for example input n bits and determine whether or not they sum to $n/2$). It seems difficult to reach significantly higher lower bounds. Superficially, it seems that this measure is nice since it gives a lower bound trade-off for time versus area. However, it does not say much about *physical* chips. For n input bits, assume that at the start of the computation we have them on chip. Since each bit physically takes $\Omega(1)$ area we have that $A = \Omega(n)$ outright (for example for the Kolmogorov random inputs which are the overwhelming majority). That means that most input bits are $\Omega(n^{1/2})$ distant from most other input bits by the geometry of space argument. In any computation where none of the input bits can be ignored, each pair of bits needs to interact somewhere, and hence information must be exchanged across $\Omega(n^{1/2})$ distance. This means, by the bounded speed of light, that $T = \Omega(n^{1/2})$. Together this trivially shows $AT^2 = \Omega(n^2)$.

Even if we assume that $A = n/f(n)$ ($f(n)$ unbounded) then the chip can contain at most $n/f(n)$ input bits and the computation needs to proceed through entering about n input bits. Since the circumference of the chip is $\Omega(\sqrt{n/f(n)})$ this takes at least $\Omega(\sqrt{n \cdot f(n)})$ time, resulting in $AT^2 = \Omega(n^2)$ again. If we account for the bounded 'pinability', bounded number of pins through which the input can be entered, we find $AT^2 = \Omega(n^3/f(n))$. All these estimates are gross underestimates because they ignore actual computing time on chip.

The AT^2 measure was widely studied [Thompson, 1979, Ullman, 1984] perhaps due to the fact that the argument used is to bisect the postulated but unknown embedded communication network (divide them into two parts whith approximately equal number of nodes by a cut of the layout), and express both A and T in terms of the unknown *minimum bisection width* of the network (minimum number of edges and nodes on the cut). Fortuitiously, using AT^2, the unknown minimum bisection width gets divided out. According to [Mead & Conway, 1980], a measure like AT has physical significance because it is related to the maximal energy consumption and energy disipation of a chip. If the gates constitute a constant fraction of A, and if all gates switch at each clock cycle, conventional technologies dissipate $\Omega(AT)$ energy in the form of heat. Because of overheating and meltdown this is a main factor which determines viability. Related measures were defined and first investigated in [Kissin, 1982–1991].

Physics has a treasure trove of nonconventional technologies which may yield computation opportunities. We cite three novel items. The first one is quantum cryptography, [Bennett, *et al.*, 1992]. Viewed first as science fiction, after a working prototype had been demonstrated this idea has now been taken on by commercial developers. British Telecom recently announced a working setup using optical fiber communication in excess of 10 kilometer. A second new development is quantum coherent computation. Because new developments in quantum coherent computation (if physically realizable) allow breaking most commonly used cryptosystems, see Section 4, quantum cryptography may be the only safe principle for public cryptography currently known, [Brassard, 1994]. In contrast with other systems, whose safety rests (or rested) on unproven cryptographic assumptions, the safety of quantum cryptography rests on the validity of quantum mechanics.

A third new principle is computation using DNA. Recently a small instance of the 'Hamiltonian path problem' was encoded in molecules of DNA and solved inside of a test tube using standard methods of molecular biology, [Adleman, 1994]. This has raised excitement about the following questions: Can practical molecular computers actually be built? Might they be as much as a billion times faster that current super computers? According to [Adleman, 1994], "To some, a computer is a physical device in the real world. But being a computer is something that we externally impose on an object. There might be a lot of computers out there, and I suspect there are".

Acknowledgements Section 3 is based on joint work with Ming Li. Discussions and help from André Berthiaume, Harry Buhrman, Tao Jiang, and John Tromp are gratefully acknowledged.

2 Geometry of Space

Models of parallel computation that allow processors to randomly access a large shared memory, such as PRAMs, or rapidly access a member of a large number of other processors, will necessarily have large latency. If we use n processing elements of, say, unit size each, then the tightest they can be packed is in a 3-dimensional sphere of volume n. Assuming that the units have no "funny" shapes, e.g., are spherical themselves, no unit in the enveloping sphere can be closer to all other units than a distance of radius R,

$$R = \left(\frac{3 \cdot n}{4\pi} \right)^{1/3} \tag{1}$$

Because of the bounded speed of light, it is impossible to transport signals over n^α ($\alpha > 0$) distance in $o(n)$ time. In fact, the assumption of the bounded speed of light says that the lower time bound on *any* computation using n processing elements is $\Omega(n^{1/3})$ outright.

We study the following problem. Let $G = (V, E)$ be a finite undirected graph, without loops or multiple edges, *embedded* in 3-dimensional Euclidean space. Let each embedded node have unit *volume*. For convenience of the argument, each node is embedded as a sphere, and is *represented* by the single point in the center. The *distance* between a pair of nodes is the Euclidean distance between the points representing them. The *length* of the embedding of an edge between two nodes is the distance between the nodes. How large does the *average* edge length need to be?

We illustrate the approach with a popular architecture, say the *binary d-cube*. Recall, that this is the network with $n = 2^d$ nodes, each of which is identified by a d-bit name. There is a two-way communication link between two nodes if their identifiers differ by a single bit. The network is represented by an undirected graph $C = (V, E)$, with V the set of nodes and $E \subseteq V \times V$ the set of edges, each edge corresponding with a communication link. There are $d2^{d-1}$ edges in C. Let C be embedded in 3-dimensional Euclidean space, each node as a sphere with unit volume. The distance between two nodes is the Euclidean distance between their centers. Let x be any node of C. There are at most $2^d/8$ nodes within Euclidean distance $R/2$ of x, with R as in Equation 1. Then, there are $\geq 7 \cdot 2^d/8$ nodes at Euclidean distance $\geq R/2$ from x. Construct a spanning tree T_x in C of depth d with node x as the root. Since the binary d-cube has diameter d, such a shallow tree exists. There are n nodes in T_x, and $n - 1$ paths from root x to another node in T_x. Let P be such a path, and let $|P|$ be the *number of edges* in P. Then $|P| \leq d$. Let $length(P)$ denote the Euclidean length of the embedding of P. Since 7/8th of all nodes are at Euclidean distance at least $R/2$ of root x, the average of $length(P)$ satisfies

$$(n-1)^{-1} \sum_{P \in T_x} length(P) \geq \frac{7R}{16}$$

The average Euclidean length of an embedded edge *in a path P* is bounded below as follows:

$$(n-1)^{-1} \sum_{P \in T_x} \left(|P|^{-1} \sum_{e \in P} length(e) \right) \geq \frac{7R}{16d}. \tag{2}$$

This does *not yet* give a lower bound on the average Euclidean length of an edge, the average taken *over all edges* in T_x. To see this, note that if the edges incident with x have Euclidean length $7R/16$, then the average edge length *in* *each path* from the root x to a node in T_x is $\geq 7R/16d$, even if all edges not incident with x have length 0. However, the average edge length *in the tree* is dominated by the many short edges near the leaves, rather than the few long edges near the root. In contrast, in the case of the binary d-cube, because of its symmetry, if we squeeze a subset of nodes together to decrease local edge length, then other nodes are pushed farther apart increasing edge length again. We can make this intuition precise, [Vitányi, 1986, Vitányi, 1988].

Lemma 1. *The average Euclidean length of the edges in the 3-space embedding of* C *is at least* $7R/(16d)$.

The symmetry property yielding such huge edge length is 'edge-symmetry.' To formulate the generalization of Lemma 1 for arbitrary graphs, we need some mathematical machinery. Let $G = (V, E)$ be a simple undirected graph, and let Γ be the automorphism group of G. Two edges $e_1 = (u_1, v_1)$ and $e_2 = (u_2, v_2)$ of G are *similar* if there is an automorphism γ of G such that $\gamma(\{u_1, v_1\}) = \{u_2, v_2\}$. We consider only connected graphs. The relation 'similar' is an equivalence relation, and partitions E into nonempty equivalence classes, called *orbits*, $E_1, ..., E_m$. We say that Γ *acts transitively* on each E_i, $i = 1, ..., m$. A graph is *edge-symmetric* if every pair of edges are similar ($m = 1$).

Additionally, we need the following notions. If x and y are nodes, then $d(x, y)$ denotes the number of edges in a *shortest path* between them. Let D denote the *diameter* of G defined by D is the maximum over all node pairs x, y of $d(x, y)$. For $i = 1, ..., m$, define $d_i(x, y)$ as follows. For edges $\{x, y\} \in E$, if $\{x, y\} \in E_i$ then $d_i(x, y) = 1$, else $d_i(x, y) = 0$. Let Π be the set of shortest paths between x and y along edges in E. If x and y are not incident on the same edge ($\{x, y\} \notin E$), then $d_i(x, y) = |\Pi|^{-1} \sum_{P \in \Pi} \sum_{e \in P} d_i(e)$. Clearly,

$$d_1(x, y) + \cdots + d_m(x, y) = d(x, y) \leq D$$

Denote $|V|$ by n. The ith *orbit frequency* is

$$\delta_i = n^{-2} \sum_{x, y \in V} \frac{d_i(x, y)}{d(x, y)},$$

$i = 1, ..., m$. Finally, define the *orbit skew coefficient* of G as $M = \min\{|E_i|/|E| : 1 \leq i \leq m\}$. Consider a d-dimensional Euclidean space embedding of G, with embedded nodes, distance between nodes, and edge length as above. Let R be the *radius* of a d-space sphere with volume n, corresponding to Equation 1 for $\mathrm{d} = 3$. We are now ready to state the main result.

Theorem 2. *Let graph* G *be embedded in d-space with the parameters above, and let* $C = (2^{\mathrm{d}} - 1)/2^{\mathrm{d}+1}$.

(i) Let $l_i = |E_i|^{-1} \sum_{e \in E_i} l(e)$ *be the average length of the edges in orbit* E_i, $i = 1, ..., m$. *Then,* $\sum_{1 \leq i \leq m} l_i \geq \sum_{1 \leq i \leq m} \delta_i l_i \geq CRD^{-1}$.

(ii) Let $l = |E|^{-1} \sum_{e \in E} l(e)$ *be the average length of an edge in* E. *Then,* $l \geq CRMD^{-1}$.

For the proof we refer to [Vitányi, 1988], where the theorem is applied to binary d-Cube, Cube-Connected Cycles, edge-symmetric graphs (including complete graph, star graph, δ-dimensional meshes with wrap-around), and complete binary tree. The lower bound is optimal in the sense of being within a constant multiplicative factor of an upper bound for several example graphs of various diameters, [Vitányi, 1988]. An extension of the argument shows the same for related networks like the Bruijn networks, shuffle-exchange graphs, and so on, [Koppelman, 1995].

2.1 Irregular Networks

Since low-diameter symmetric network topologies lead to high average interconnect length, it is natural to ask what happens with irregular topologies. In fact, it is sometimes proposed that since symmetric networks of low diameter lead to high interconnect length, one should use random networks where the presence or absence of a connection is determined by a coin flip. We report on some work in [Vitányi, 1994] that such networks will also have impossibly high average interconnect length.

Kolmogorov complexity The Kolmogorov complexity, [Kolmogorov, 1965], of x is the length of the *shortest* effective description of x. That is, the *Kolmogorov complexity* $C(x)$ of a finite string x is simply the length of the shortest program, say in FORTRAN[2] encoded in binary, which prints x without any input. A similar definition holds conditionally, in the sense that $C(x|y)$ is the length of the shortest binary program which computes x given y as input. It can be shown that the Kolmogorov complexity is absolute in the sense of being independent of the programming language, up to a fixed additional constant term which depends on the programming language but not on x. We now fix one canonical programming language once and for all as reference programming language, and thereby we fix $C()$ uniquely.

For the theory and applications, see [Li & Vitányi, 1993]. Let $x, y, z \in \mathcal{N}$, where $\mathcal{N}$ denotes the natural numbers and we identify $\mathcal{N}$ and $\{0,1\}^*$ according to the correspondence $(0, \epsilon), (1, 0), (2, 1), (3, 00), (4, 01), \ldots$. Hence, the length $|x|$ of x is the number of bits in the binary string x. Let $T_1, T_2, \ldots$ be a standard enumeration of all Turing machines. Let $\langle \cdot, \cdot \rangle$ be a standard invertible effective bijection from $\mathcal{N} \times \mathcal{N}$ to $\mathcal{N}$. This can iterated to $\langle \langle \cdot, \cdot \rangle, \cdot \rangle$.

Definition 3. Let U be an appropriate universal Turing machine such that $U(\langle\langle i, p\rangle, y\rangle) = T_i(\langle p, y\rangle)$ for all i and $\langle p, y\rangle$. The *Kolmogorov complexity* of x given y (for free) is

$$C(x|y) = \min\{|p| : U(\langle p, y\rangle) = x, p \in \{0,1\}^*, i \in \mathcal{N}\}.$$

[2] Or in Turing machine codes.

One way to express irregularity or *randomness* of an individual network topology is by a modern notion of randomness like Kolmogorov complexity. A simple counting argument shows that for each y in the condition and each length n there exists at least one x of length n which is *incompressible* in the sense of $C(x|y) \geq n$, 50% of all x's of length n is incompressible but for 1 bit ($C(x|y) \geq n-1$), 75% of all x's is incompressible but for 2 bits ($C(x|y) \geq n-2$) and in general a fraction of $1-2^{-c}$ of all strings cannot be compressed by more than c bits. (This is because there are 2^n strings of length n and only $2^{n-c}-1$ binary programs of length at most $n-c$, [Li & Vitányi, 1993]. A more sophisticated argument shows that there are a large number of strings of length n with complexity at least n.)

Random Graphs Each graph $G=(V,E)$ on n nodes $V=\{0,\ldots,n-1\}$ can be coded (up to isomorphism) by a binary string of length $n(n-1)/2$. We enumerate the $n(n-1)/2$ possible edges in a graph on n nodes in standard order and set the ith bit in the string to 1 if the edge is present and to 0 otherwise. Conversely, each binary string of length $n(n-1)/2$ encodes a graph on n nodes. Hence we can identify each such graph with its corresponding binary string.

We shall call a graph G on n nodes *random* if it satisfies

$$C(G|n) \geq n(n-1)/2 - cn, \tag{3}$$

where c is an appropriate constant ($c = 0.09$ suffices for n large enough). Elementary counting shows that *a fraction* of at least

$$1 - 1/2^{cn}$$

of all graphs on n nodes has that high complexity.

Lemma 4. *The degree d of each node of a random graph satisfies $|d-(n-1)/2| < n/4$.* [3]

Proof. Assume that the deviation of the degree d of a node v in G from $(n-1)/2$ is at least k. From the lower bound on $C(G|n)$ corresponding to the assumption that G is random, we can estimate an upper bound on k, as follows.

Describe G given n as follows. We can indicate which edges are incident on node v by giving the index of the connection pattern in the ensemble of

$$m = \sum_{|d-(n-1)/2| \geq k} \binom{n}{d} \leq 2^n e^{-k^2/(n-1)} \tag{4}$$

[3] One can replace cn in Equation 3 by $o(n)$. Then in Lemma 4 we can replace $n/4$ by $o(n)$. The random graphs under this definition contain only nodes with vertex degree about $n/2$. With $n/\log n$ substituted for $o(n)$, they constitute a slightly smaller fraction $1-1/2^{n/\log n}$ of all graphs on n nodes, but still a fraction which goes to 1 fast with n. With Harry Buhrman we have proved in a forthcoming paper that there are $2^n/d$ graphs G satisfying $C(G|n) \geq n(n-1)/2$, where d is a constant. For such complex G we have $|d-(n-1)/2| = O(\sqrt{n})$. The fraction of such G among all graphs on n nodes is at least $1/d$.

possibilities. The last inequality follows from a general estimate of the tail probability of the binomial distribution, Chernoff's bounds, [Li & Vitányi, 1993], pp. 127-130. To describe G it then suffices to modify the old code of G by prefixing it with

- the identity of the node concerned in $\lceil \log n \rceil$ bits,
- the value of d in $\lceil \log n \rceil$ bits, possibly adding nonsignificant 0's to pad up to this amount,
- the index of the interconnection pattern in $\log m + 2 \log \log m$ bits in self-delimiting form (this form requirement allows the concatenated binary sub-descriptions to be parsed and unpacked into the individual items: it encodes a separation delimiter in $< 2 \log \log m$ bits, [Li & Vitányi, 1993]),

followed by the old code for G with the bits in the code denoting the presence or absence of the possible edges which are incident on the node v deleted.

Clearly, given n we can reconstruct the graph G from the new description. The total description we have achieved is an effective program of

$$\log m + 2 \log \log m + O(\log n) + n(n-1)/2 - (n-1)$$

bits. This must be at least the length of the shortest effective binary program, which is $C(G|n)$ satisfying Equation 3. Therefore,

$$\log m + 2 \log \log m \geq n - 1 - O(\log n) - cn.$$

Since we have estimated in Equation 4 that

$$\log m \leq n - (k^2/(n-1)) \log e,$$

it follows that, with $c = 0.09$,

$$k < n/4.$$

The lemma shows that each node is connected by an edge with about 25% of all nodes in G. Hence G contains a subgraph on about 25 % of its nodes of diameter 1. This is all we need. [4]

Theorem 5. *A fraction of at least $1 - 1/2^{cn}$ ($c = 0.09$) of all graphs on n nodes (the incompressible, random, graphs) have total interconnect length of $\Omega(n^{7/3})$ in each 3-dimensional Euclidean space embedding (or $\Omega(n^{5/2})$ in each 2-dimensional Euclidean space embedding).*

Proof. By lemma 4 we know that in a random graph G each node x is at distance 1 of $(n-1)/2 \pm n/4$ other nodes y, and 7/8th of these nodes y (in 3 dimensions) is at distance $\Omega(n^{1/3})$ of x by Equation 1. The argument for 2 dimensions is analogous. By Lemma 4 we know that a random graph G on n nodes has $\Omega(n^2)$ edges since each node has at least $n/4$ incident edges.

[4] Using another standard incompressibility argument, as suggested by Harry Buhrman, one can show that all graphs which are random in the sense above have diameter precisely 2.

Since both the very regular symmetric low diameter graphs and the random graphs have high average interconnect length which sharply rises with n, the only graphs which will scale feasibly up are symmetric fairly high diameter topologies like the mesh—which therefore will most likely be the interconnection pattern of the future massive multiprocessor systems.

2.2 Interpretation of the Results

An effect that becomes increasingly important at the present time is that most space in the device executing the computation is taken up by the wires. Under very conservative estimates that the unit length of a wire has a volume which is a constant fraction of that of a component it connects, we have shown in [Vitányi, 1988] that in 3-dimensional layouts for binary d-cubes, the volume of the $n = 2^d$ components (nodes) performing the actual computation operations is an asymptotic fastly vanishing fraction of the volume of the wires needed for communication:

$$\frac{\text{volume computing components}}{\text{volume communication wires}} = o(n^{-1/3})$$

If we charge a constant fraction of the unit volume for a unit wire length, and add the volume of the wires to the volume of the nodes, then the volume necessary to embed the binary d-cube is $\Omega(n^{4/3})$. However, this lower bound ignores the fact that the added volume of the wires pushes the nodes further apart, thus necessitating longer wires again. How far does this go? A rigorous analysis is complicated, and not important here. The following intuitive argument indicates what we can expect well enough. Denote the volume taken by the nodes as V_n, and the volume taken by the wires as V_w. The total volume taken by the embedding of the cube is $V_t = V_n + V_w$. The total wire length required to lay out a binary d-cube as a function of the volume taken by the embedding is, substituting radius R obtained from $V_t = 4\pi R^3/3$ in the formula for the total wire length obtained in [Vitányi, 1988],

$$L(V_t) \geq \frac{7n}{32}\left(\frac{3V_t}{4\pi}\right)^{1/3}$$

Since $\lim_{n\to\infty} V_n/V_w \to 0$, assuming unit wire length of unit volume, we set the total interconnect length $L(V_t)$ at $L(V_t) \approx V_t$. This results in a better estimate of $\Omega(n^{3/2})$ for the volume needed to embed the binary d-cube. When we want to investigate an upper bound to embed the binary d-cube under the current assumption, we have a problem with the unbounded degree of unit volume nodes. There is no room for the wires to come together at a node. For comparison, therefore, consider the fixed degree version of the binary d-cube, the CCC (see above), with $n = d2^d$ trivalent nodes and $3n/2$ edges. The same argument yields $\Omega(n^{3/2} \log^{-3/2} n)$ for the volume required to embed CCC with unit volume per unit length wire. It is known, that every small degree n-vertex graph, e.g., CCC, can be laid out in a 3-dimensional grid with volume $O(n^{3/2})$ using a unit volume per unit wire length assumption, [Mead & Conway, 1980, Ullman, 1984]. This neatly matches the lower bound.

Just like for the complete graph, the situation for the random graph which we consider here, is far worse. For a random graph we have, under the assumption that the wires have unit volume per unit length, that the total wire length in 3 dimensional embeddings is $\Omega(n^{7/3})$ by Theorem 5, and that

$$\frac{\text{volume communication wires}}{\text{volume computing components}} = \Omega(n^{4/3})$$

The proof of Theorem 5 actually shows that the total interconnect length of an embedded random graph is $L(V_t) = \Omega(n^2 V_t^{1/3})$, where the radius of an as tight as possibly packed 3-dimensional sphere of the total volume V_t of nodes and wires together is $\Omega(V_t^{1/3})$. Considering that the larger volume will cause the average interconnect length to increase, as above for the binary d-cube, setting the total interconnect length $L(V_t) \approx V_t$ since the volume of the computing nodes add a negligible term, we find for a random graph that on n nodes that the total volume satisfies

$$V_t = \Omega(n^3).$$

Here we have not yet taken into account that longer wires need larger drivers and have a larger diameter, that the larger volume will again cause the average interconnect length to increase, and so on, which explosion may make embedding altogether impossible with finite length interconnects as exhibited in related contexts in [Vitányi, 1985].

3 Adiabatic Computation and Thermodynamics

All computations can be performed logically reversibly, [Bennett, 1973], at the cost of eventually filling up the memory with unwanted garbage information. This means that reversible computers with bounded memories require in the long run irreversible bit operations, for example, to erase records irreversibly to create free memory space. The minimal possible number of irreversibly erased bits to do so is believed to determine the ultimate limit of heat dissipation of the computation by Landauer's principle, [Landauer, 1961, Bennett, 1973, Bennett, 1982, Proc. PhysComp, 1981, 1992, 1994]. In reference [Bennett *et al.,* 1993] we and others developed a mathematical theory for the unavoidable number of irreversible bit operations in an otherwise reversible computation.

Methods to implement (almost) reversible and dissipationless computation using conventional technologies appear in [Proc. PhysComp, 1981, 1992, 1994], often designated by the catch phrase 'adiabatic switching'. Many currently proposed physical schemes implementing adiabatic computation reduce irreversibility by using longer switching times. This is done typically by switching over equal voltage gates after voltage has been equalized slowly. This type of switching does not dissipate energy, [Proc. PhysComp, 1981, 1992, 1994], the only energy dissipation is incurred by pulling voltage up and down: the slower it goes the less energy is dissipated. If the computation goes infinitely slow, zero energy is dissipated. Clearly, this counteracts the purpose of low energy dissipation which is faster computation.

In [Li & Vitányi, 1994] it is demonstrated that even if adiabatic computation technology advances to switching with no time loss, a similar phenomenon arises when we try to approach the ultimate limits of minimal irreversibility of an otherwise reversible computation, and hence minimal energy dissipation. This time the effect is due to the logical method of reducing the number of irreversible bit erasures in the computation irrespective of individual switching times. By computing longer and longer (in the sense of using more computation steps), the amount of dissipated energy gets closer to ultimate limits. Moreover, one can trade-off time (number of steps) for energy: there is a new time-irreversibility (time-energy) trade-off hierarchy. The bounds we derive are also relevant for quantum computations which are reversible except for the irreversible observation steps, [Deutsch, 1985–1992, Benioff, 1980–1986, Benioff, 1995].

3.1 Background

The ultimate limits of miniaturization of computing devices, and therefore the speed of computation, are governed by unavoidable heating up attending rising energy dissipation caused by increasing density of switching elements in the device. On a basically two dimensional device, linear speed up by shortening interconnects is essentially attended by squaring the dissipated energy per area unit per second because we square the number of switching elements per area unit, [Mead & Conway, 1980].

Therefore, the question of how to reduce the energy dissipation of computation determines future advances in computing power. Around 1940 a computing device dissipated about 10^{-2} Joule per bit operation at room temperature. Since that time the dissipated energy per bit operation has roughly decreased by one order of magnitude (tenfold) every five years. Currently, a bit operation dissipates[5] about 10^{-17} Joule. Extrapolations of current trends show that the energy dissipation per binary logic operation needs to be reduced below kT (thermal noise) within 20 years. Here k is Boltzmann's constant and T the absolute temperature in degrees Kelvin, so that $kT \approx 3 \times 10^{-21}$ Joule at room temperature. Even at kT level, a future laptop containing 10^{18} gates in a cubic centimeter operating at a gigahertz dissipates 3 million watts/second. For thermodynamic reasons, cooling the operating temperature of such a computing device to almost absolute zero (to get kT down) must dissipate at least as much energy in the cooling as it saves for the computing.

Considerations of thermodynamics of computing started in the early fifties. J. von Neumann [Burks, 1966] reputedly thought that a computer operating at temperature T must dissipate at least $kT \ln 2$ Joule per elementary bit operation (about 3×10^{-21} J at room temperature).

[5] After R.W. Keyes, IBM Research.

Around 1960, R. Landauer [Landauer, 1961] more thoroughly analyzed this question and concluded that it is only 'logically irreversible' operations that dissipate energy. An operation is *logically reversible* if its inputs can always be deduced from the outputs. Erasure of information in a way such that it cannot be retrieved is not reversible. Erasing each bit costs $kT \ln 2$ energy, when computer operates at temperature T.

One should sharply distinguish between the issue of logical reversibility and the issue of energy dissipation freeness. The fact that some computers operates in a logically reversible manner says nothing about whether they dissipate heat. It only says is that the laws of physics do not preclude that one can invent a technology in which to implement a logically similar computer to operate physically in a dissipationless manner. Computers built from reversible circuits, or the reversible Turing machine, [Bennett, 1973, Bennett, 1982, Fredkin & Toffoli, 1982], implemented with current technology will presumably dissipate energy but may conceivably be implemented by future technology in an adiabatic fashion. For nonreversible computers adiabatic implementation is widely considered impossible.

Thought experiments can exhibit a computer that is both logically and physically perfectly reversible and hence perfectly dissipationless. An example is the billiard ball computer, [Fredkin & Toffoli, 1982], and similarly the possibility of a coherent quantum computer, [Feynman, 1982—1987, Deutsch, 1985–1992]. Our purpose is to determine the theoretical ultimate limits to which the irreversible actions in an otherwise reversible computation can be reduced.

3.2 Irreversibility Cost of Computation

The ultimate limits of energy dissipation by computation will be expressed in number of irreversibly erased bits. Hence we consider compactification of records. In analogy of garbage collection by a garbage truck, the cost is less if we compact the garbage before we throw it away. The ultimate compactification which can be effectively exploited is expressed in terms of Kolmogorov complexity.

Let $\mathbf{R} = R_1, R_2, \ldots$ be a standard enumeration of reversible Turing machines, [Bennett, 1973]. We define $E(\cdot, \cdot)$ as in [Bennett *et al.,* 1993] (where it is denoted as $E_3(\cdot, \cdot)$).

Definition 6. The *irreversibility cost* $E_R(x, y)$ of computing y from x by a reversible Turing machine R is is

$$E_R(x, y) = \min\{|p| + |q| : R(\langle x, p\rangle) = \langle y, q\rangle\}.$$

We denote the class of all such cost functions by $\mathcal{E}$.

We call an element E_Q of $\mathcal{E}$ a *universal irreversibility cost function*, if $Q \in \mathbf{R}$, and for all R in $\mathbf{R}$

$$E_Q(x, y) \leq E_R(x, y) + c_R,$$

for all x and y, where c_R is a constant which depends on R but not on x or y. Standard arguments from the theory of Turing machines show the following.

Lemma 7. *There is a universal irreversibility cost function in $\mathcal{E}$. Denote it by E_{UR}.*

Proof. In [Bennett, 1973] a universal reversible Turing machine UR is constructed which satisfies the optimality requirement.

Two such universal (or optimal) machines UR and UR' will assign the same irreversibility cost to a computation apart from an additive constant term c which is *independent* of x and y (but does depend on UR and UR'). We select a reference universal function UR and define the *irreversibility cost* $E(x,y)$ of computing y from x as

$$E(x,y) \equiv E_{UR}(x,y).$$

Because of the expression for $E(x,y)$ in Theorem 8 below it is called the *sum cost* measure in [Bennett *et al.,* 1993].

In physical terms this cost is in units of $kT \ln 2$, where k is Boltzmann's constant, T is the absolute temperature in degrees Kelvin, and ln is the natural logarithm.

Because the computation is reversible, this definition is *symmetric*: we have $E(x,y) = E(y,x)$.

In our definitions we have pushed all bits to be irreversibly provided to the start of the computation and all bits to be erased to the end of the computation. It is easy to see that this is no restriction. If we have a computation where irreversible acts happen throughout the computation, then we can always mark the bits to be erased, waiting with actual erasure until the end of the computation. Similarly, the bits to be provided can be provided (marked) at the start of the computation while the actual reading of them (simultaneously unmarking them) takes place throughout the computation).

Now let us consider a general computation which outputs string y from input string x. We want to know the minimum irreversibility cost for such computation. This leads to the following theorem, for two different proofs see [Bennett *et al.,* 1993, Li & Vitányi, 1994].

Theorem 8 Fundamental theorem. *Up to an additive logarithmic term*

$$E(x,y) = C(x|y) + C(y|x).$$

Erasing a record x is actually a computation from x to the empty string ϵ. Hence its irreversibility cost is $E(x,\epsilon)$.

Corollary 9. *Up to a logarithmic additive term, the irreversibility cost of erasure is $E(x,\epsilon) = C(x)$.*

3.3 Trading Time for Energy

Because now the time bounds are important we consider the universal Turing machine U to be the machine with two work tapes which can simulate t steps of a multitape Turing machine T in $O(t \log t)$ steps (the Hennie-Stearns simulation). If some multitape Turing machine T computes x in time t from a program p, then U computes x in time $O(t \log t)$ from p plus a description of T.

Definition 10. Let $C^t(x|y)$ be the *minimal length* of binary program (not necessarily reversibly) for the two work tape universal Turing machine U computing x given y (for free) *in time* t. Formally,

$$C^t(x|y) = \min_{p \in \mathcal{N}} \{|p| : U(\langle p, y \rangle) = x \text{ in } \leq t(|x|) \text{ steps}\}.$$

$C^t(x|y)$ is called the *t-time-limited conditional Kolmogorov complexity* of x given y. The unconditional version is defined as $C^t(x) := C^t(x, \epsilon)$. A program p such that $U(p) = x$ in $\leq t(|x|)$ steps and $|p| = C^t(x)$ is denoted as x_t^*.

Note that with $C_T^t(x|y)$ the conditional t-time-limited Kolmogorov complexity with respect to Turing machine T, for all x, y, $C^{t'}(x|y) \leq C_T^t(x|y) + c_T$, where $t' = O(t \log t)$ and c_T is a constant depending on T but not on x and y.

This $C^t(\cdot)$ is the standard definition of time-limited Kolmogorov complexity, [Li & Vitányi, 1993]. However, in the remainder of the paper we always need to use reversible computations. Fortunately, in [Bennett, 1989] it is shown that for any $\epsilon > 0$, ordinary multitape Turing machines using T time and S space can be simulated by reversible ones using time $O(T)$ and space $O(ST^{\epsilon})$.

To do effective erasure of compacted information, we must at the start of the computation provide a time bound t. Typically, t is a recursive function and the complexity of its description is small, say $O(1)$. However, in Theorem 11 we allow for very large running times in order to obtain smaller $C^t(\cdot)$ values.

Theorem 11 Irreversibility cost of effective erasure. *If $t(|x|) \geq |x|$ is a time bound which is provided at the start of the computation, then erasing an n bit record x by an otherwise reversible computation can be done in time (number of steps) $O(2^{|x|} t(|x|))$ at irreversibility cost (hence also thermodynamic cost) $C^t(x) + 2C^t(t|x) + 4 \log C^t(t|x)$ bits. (Typically we consider t as some standard explicit time bound and the last two terms adding up to $O(1)$.)*

Proof. Initially we have in memory input x and a program p of length $C^t(t, x)$ to compute reversibly t from x. To separate binary x and binary p we need to encode a delimiter in at most $2 \log C^t(t|x)$ bits.

1. Use x and p to reversibly compute t. Copy t and reverse the computation. Now we have x, p and t.
2. Use t to reversibly dovetail the running of all programs of length less than x to find the shortest one halting in time t with output x. This is x_t^*. The computation has produced garbage bits $g(x, x_t^*)$. Copy x_t^*, and reverse the computation to obtain x erasing all garbage bits $g(x, x_t^*)$. Now we have x, p, x_t^*, t in memory.
3. Reversibly compute t from x by p, cancel one copy of t, and reverse the computation. Now we have x, p, x_t^* in memory.
4. Reversibly cancel x using x_t^* by the standard method, and then erase x_t^* and p irreversibly.

Corollary 12. *$E(x, \epsilon) \geq \lim_{t \to \infty} C^t(x) = C(x)$, and by Theorem 8 up to an additional logarithmic term, $E(x, \epsilon) = C(x)$.*

Essentially, by spending more time we can reduce the thermodynamic cost of erasure of x_t^* to its absolute minimum. In the limit we spend the optimal value $C(x)$ by erasing x^*, since $\lim_{t \to \infty} x_t^* = x^*$. This suggests the existence of a trade-off hierarchy between time and energy. The longer one reversibly computes to perform final irreversible erasures, the less bits are erased and energy is dissipated. This intuitive assertion will be formally stated and rigourously proved below.

Definition 13. Let UR be the reversible version of the two worktape universal Turing machine, simulating the latter in linear time by Bennett's result mentioned above. Let $E^t(x, y)$ be the *minimum irreversibility cost* of an otherwise reversible computation from x to y *in time* t. Formally,

$$E^t(x,y) = \min_{p,q \in \mathcal{N}} \{|p| + |q| : UR(\langle x, p \rangle) = \langle y, q \rangle \text{ in } \leq t(|x|) \text{ steps}\}.$$

Because of the similarity with Corollary 12 ($E(x, \epsilon)$ is about $C(x)$) one is erroneously led to believe that $E^t(x, \epsilon) = C^t(x)$ up to a log additive term. However, the time-bounds introduce many differences. To reversibly compute x_t^* we may require (because of the halting problem) at least $O(2^{|x|} t(|x|))$ steps after having decoded t, as indeed is the case in the proof of Theorem 11. In contrast, $E^t(x, \epsilon)$ is about the number of bits erased in an otherwise reversible computation which uses at most t steps. Therefore, as far as we know possibly $C^t(x) \geq E^{t'}(x, \epsilon)$ implies $t' = \Omega(2^{|x|} t(|x|))$. More concretely, it is easy to see that for each x and $t(|x|) \geq |x|$,

$$E^t(x, \epsilon) \geq C^t(x) \geq E^{t'}(x, \epsilon)/2, \tag{5}$$

with $t'(|x|) = O(t(|x|)$. Theorem 11 can be restated in terms of $E^t(\cdot)$ as

$$E^{t'}(x, \epsilon) \leq C^t(x) + 2C^t(t|x) + 4 \log C^t(t|x),$$

with $t'(|x|) = O(2^{|x|} t(|x|))$. Comparing this to the righthand inequality of Equation 5 we have improved the upper bound on erasure cost at the expense of increasing erasure time. However, these bounds only suggest but do not actually prove that we can exchange irreversibility for time. The following result definitely establishes the existence of a trade-off, [Li & Vitányi, 1994].

Theorem 14 Irreversibility-time trade-off hierarchy. *For each large enough n there is a string x of length n and a sequence of $m = \frac{1}{2}\sqrt{n}$ time functions $t_1(n) < t_2(n) < \ldots < t_m(n)$, such that*

$$E^{t_1}(x, \epsilon) > E^{t_2}(x, \epsilon) > \ldots > E^{t_m}(x, \epsilon).$$

In the cost measures like $E^t(\cdot, \cdot)$ we have counted both the irreversibly provided and the irreversibly erased bits. But Landauer's principle only charges energy dissipation costs for irreversibly erased bits. It is conceivable that the above results would not hold if one considers as the cost of erasure of a record only the irreversibly erased bits. However, we have show that Theorem 14 also holds for Landauer's dissipation measure, [Li & Vitányi, 1994], in exactly the same form and by almost the same proof.

4 Quantum Coherent Parallel Computation

Classical methods of parallel computation are plagued by wiring problems (Section 2) and heat dissipation problems (Section 3). To counteract such problems attending further miniaturization of parallel computing devices current research considers quantum mechanics based technologies. Classical use of such technologies deals with reducing feature width on chip to below the nanometer level, [Kiehl, 1994], or interacting quantum dots subnanotechnology layouts for cellular automata, [Lent *et al.*, 1994].

This section deals with the prospect of a very *nonclassical* emergent possible computer technology (quantum coherent computing or QCC) which has recently acquired great anticipated economic value. This happened by one of the most fortuitious demonstrations in computing that QCC can break the universally used public key cryptosystems by being able to factor and do the discrete logarithm in polynomial time, [Shor, 1994] with preliminary work in [Deutsch, 1985–1992, Bernstein and Vazirani, 1993, Simon, 1994]. This result opened the vista of a veritable breakthrough in computing. There are apparently formidable obstacles to surmount before a workable technology can be obtained, [Unruh, 1995].

The QCC approach as first advocated in [Benioff, 1980–1986] is currently aimed to exploit the accepted theory that quantum evolution of an appropriate system consists in a superposition of many (potentially infinitely many) simultaneous computation paths. It is theoretically possible that through the specific quantum mechanical rules of interference of the different paths one can boost the probability associated with desirable evolutions and suppress undesirable ones for certain algorithms. Upon observation of the system state one of the states in superposition is realized. By quantum specific algorithmic techniques the desired outcome can theoretically by observed with arbitrary high probability, or the desired outcome can be computed from the observed data with arbitrary high probability.

The QCC approach will partially alleviate the wiring problem (Section 2) because an exploding number of different computation paths will be simultaneously followed (with appropriate probability amplitudes, to be sure) by the same single physical apparatus requiring but a tiny amount of physical space. This is the substance of R. Feynman's dictum "there is room at the bottom" in the context of his proposal of QCC, [Feynman, 1982—1987]. Of course, since the different computation paths of a quantum computation cannot communicate as is often a main feature in a parallel distributed computation, it is only a very special type of room which is available at the bottom. Moreover, since the quantum evolution in a computation if unobstructed by observation and decoherence is reversible, the pure form of QCC, apart from the irreversible observation phase, is energy dissipation free. QCC seems to a very large extent to achieve the optimal adiabatic computation aimed for in Section 3. Although there seems to be agreement that energy gets dissipated in the irreversible observation phase, to the author's knowledge it is not yet clear how much. This seems to require a quantum Kolmogorov complexity based on 'qubits' (quantum bits) as defined in context of quantum information theory by [Schuhmacher, 1994], analogous to the classical bits of information theory of [Shannon, 1948]. Through a sequence of proposals [Benioff, 1980–1986], [Feynman, 1982—1987], [Deutsch, 1985–1992], there has emerged a Turing machine model of quantum coherent computing.

4.1 Background: Probabilistic Turing Machines

The simplest way to describe it seems by way of probabilistic machines. Suppose we consider the well known probabilistic Turing machine which is just like an ordinary Turing machine, except that at each step the machine can make a probabilistic move which consists in flipping a (say fair) coin and depending on the outcome changing its state to either one of two alternatives. This means that at each such probabilistic move the computation of the machine splits into two distinct further computations each with probability $1/2$. Ignoring the deterministic computation steps, a computation involving m coinflips can be viewed as a binary computation tree of depth m with 2^m leaves, where each node at level $t \leq m$ corresponds to a state of the system which after t coinflips occurs with probability $1/2^t$. For convenience, we can label the edges connecting a state x directly with a state y with the probability that a state x changes into state y in a single coin flip (in this example all edges are labeled '1/2').

As an example, given an arbitrary Boolean formula containing n variables, a probabilistic machine can flip its coin n times to check each of the 2^n possible truth assignments to determine wether there exists an assignment to the variables which makes the formula true. If there are m distinct such assignments then the probabilistic machine finds that the formula is satisfyable with probability at least $m/2^n$—since there are m distinct computation paths leading to a satisfyable assignment.

Now suppose the probabilistic machine is hidden in a black box and the computation proceeds without us knowing the outcomes of the coin flips. Suppose that after t coin flips in the computation we open part of the black box and observe the bit at the position of the Turing machine tape which denotes the truth assignment for variable x_5 ($5 \leq t$) which already received its truth assignment. Before we opened the black box all 2^t initial truth assignments to variables $x_1, \ldots, x_t$ were equally possible, each had probability $1/2^t$. After we observed the state of variable x_5, say 0, the probability space of possibilities has collapsed to the truth assignments which consist of all binary vectors with a 0 in the 5th position each of which has probability renormalized at $1/2^{t-1}$.

4.2 Quantum Turing Machines

A quantum Turing machine is related to the probabilistic Turing machine. Consider the same computation tree. However, instead of a probability $p_i \geq 0$ associated with each node i, such that $\sum p_i = 1$, the summation taking place over the states a computation can possible be in at a particular time instant, there is an amplitude α_i associated with each state $|i\rangle$ of an observable of the system (the notation $|\cdot\rangle$ has good reasons in quantum mechanics notation related to the particular matrix mathematics involved). The amplitudes are complex numbers satisfying $\sum ||\alpha_i||^2 = 1$, where if $\alpha_i = a + b\sqrt{-1}$ then $||\alpha_i|| = \sqrt{a^2+b^2}$ and the summation is taken over all distinct states of the observable at a particular instant. The transitions are governed by a matrix U which represents the program executed. This program has to satisfy the following constraints. If the set of possible ID's (complete instantaneous description) of the Turing machine is X, where X is say $\{0,1\}^n$ to simplify the discussion, then U maps the column vector $\underline{\alpha} = (\alpha_x)_{x \in X}$ to $U\underline{\alpha}$. Here $\underline{\alpha}$ is a vector of amplitudes of the quantum superposition of the distinct possible states in X before a step, and $U\underline{\alpha}$ the same after the step concerned. The special property which U needs to satisfy in quantum mechanics is that it is *unitary*, that is, $U \times U^\dagger = I$ where I is the identity matrix and $U^\dagger$ is the conjugate transpose of U ('conjugate' means that all $\sqrt{-1}$'s are replaced by $-\sqrt{-1}$'s and 'transpose' means that the rows and columns are interchanged). In other words, U is unitary iff $U^\dagger = U^{-1}$.

The unitary constraint on the evolution of the computation enforces two facts.

1. If $U^0\underline{\alpha} = \underline{\alpha}$ and $U^t = U \cdot U^{t-1}$ then $\sum_{x \in X} ||(U^t\underline{\alpha})_x||^2 = 1$ for all t (discretizing time for convenience).
2. A quantum computation is reversible (replacing U by $U^\dagger = U^{-1}$).

The common example here is a simple computation on a one-bit computer. The quantum superposition of states of the computer is denoted by

$$|\Psi\rangle = \alpha \ |0\rangle + \beta \ |1\rangle,$$

where $||\alpha||^2 + ||\beta||^2 = 1$. The different possible states are $|0\rangle = \binom{1}{0}$ and $|1\rangle = \binom{0}{1}$. Our unitary operator will be

$$U \ = \ \frac{\sqrt{2}}{2} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}.$$

It is easy to verify using common matrix calculation that

$$\begin{aligned} U \ |0\rangle &= \sqrt{2}/2 \ |0\rangle - \sqrt{2}/2 \ |1\rangle \\ U \ |1\rangle &= \sqrt{2}/2 \ |0\rangle + \sqrt{2}/2 \ |1\rangle \\ U^2 \ |0\rangle &= 0 \ |0\rangle - 1 \ |1\rangle = - \ |1\rangle \\ U^2 \ |1\rangle &= 1 \ |0\rangle + 0 \ |1\rangle = \ |0\rangle \end{aligned}$$

If we observe the computer in state $U \ |0\rangle$, then the probability of observing state $|0\rangle$ is $(\sqrt{2}/2)^2 = 1/2$ and the probability to observe $|1\rangle$ is $(-\sqrt{2}/2)^2 = 1/2$. However, if we observe the computer in state $U^2 \ |0\rangle$, then the probability of observing state $|0\rangle$ is 0 and the probability to observe $|1\rangle$ is 1. Similarly, if we observe the computer in state $U \ |1\rangle$, then the probability of observing state $|0\rangle$ is $(\sqrt{2}/2)^2 = 1/2$ and the probability to observe $|1\rangle$ is $(\sqrt{2}/2)^2 = 1/2$. But now, if we observe the computer in state $U^2 \ |1\rangle$, then the probability of observing state $|0\rangle$ is 1 and the probability to observe $|1\rangle$ is 0. Therefore, the operator

U inverts a bit when it is applied twice in a row, and hence has acquired the charming name *square root of 'not'*. It is a simple exercise to write U in terms of an if–then–else program:

$$\begin{array}{ll} \textbf{if } |\Psi\rangle = |0\rangle \textbf{ then} & |\Psi\rangle := \sqrt{2}/2 \ |0\rangle - \sqrt{2}/2 \ |1\rangle \\ \qquad\qquad\qquad \textbf{else} & |\Psi\rangle := \sqrt{2}/2 \ |0\rangle + \sqrt{2}/2 \ |1\rangle \end{array}$$

Without mentioning it, and perhaps without the reader even noticing, we have applied as a matter of course an absolutely crucial difference between quantum computation and probabilistic computation.

4.3 Observables

According to quantum mechanics a physical system gives rise to a complex linear vector space $\mathcal{H}$, such that each vector of unit length represents a state of the system $|\Psi\rangle \in \mathcal{H}$.

A *quantum measurement* gives rise to a Hermitian operator $\hat{A}$ (the *observable*) and a decomposition of $\mathcal{H}$ into orthogonal subspaces (different *states* of the observable)

$$\mathcal{H} = A_1 \oplus A_2 \oplus \cdots \oplus A_n,$$

with $\hat{A} = \sum_{i=1}^{n} \alpha_i P_i$ where P_i is the projector of state $|\Psi\rangle$ on A_i (say, $|a_i\rangle$). If we measure observable $\hat{A}$ in system state $|\Psi\rangle$, with $|\Psi\rangle = \sum_{i=1}^{n} c_i \ |a_i\rangle$, then the following happens with probability $||c_k||^2$:

1. The outcome of the measurement α_k is registered.
2. The superposition $|\Psi\rangle$ collapses to superposition $|a_k\rangle \in A_k$.
3. The probability of observing $|a_k\rangle$ is renormalized to 1.

4.4 Interference.

In computing the above amplitudes, subsequent to two applications of U, according to matrix calculus we found that

$$\begin{aligned} U^2 \ |1\rangle &= \sqrt{2}/2 \left(\sqrt{2}/2(\ |0\rangle - \ |1\rangle) + \sqrt{2}/2(\ |0\rangle + \ |1\rangle)\right) \\ &= \tfrac{1}{2}(\ |0\rangle - \ |1\rangle + \ |0\rangle + \ |1\rangle) \ = \ |0\rangle. \end{aligned}$$

In a probabilistic calculation, flipping a coin two times in a row, we would have found that the probability of each computation path in the complete binary computation tree of depth 2 was $1/4$, and the states at the four leaves of the tree were $|0\rangle$, $|1\rangle$, $|0\rangle$, $|1\rangle$, resulting in a total probability of observing $|0\rangle$ being $1/2$ and the total probability of observing $|1\rangle$ being $1/2$ as well.

The principle involved is called *interference*, like with light. If we put a screen with a single small enough hole in between a light source and a target, then we observe a gradually dimming illumination of the target screen, the brightest spot being colinear with the light source and the hole. If we put a screen with two small holes in between, then we observe a diffraction pattern of bright and dark stripes due to interference. Namely, the light hits all of the screen via two different routes (through the two different holes). If the two routes differ by an even number of half wave lengths, then the wave amplitudes at the target are added, resulting in twice the amplitude and a bright spot, and if they differ by an odd number of half wave lengths then the wave amplitudes are in opposite phase and are subtracted resulting in zero and a dark spot. Similarly, with quantum computation, if the quantum state is

$$|\Psi\rangle = \alpha \; |x\rangle + \beta \; |y\rangle,$$

then for $x = y$ we have a probability of observing $|x\rangle$ of $||\alpha + \beta||^2$, rather than $||\alpha||^2 + ||\beta||^2$ which it would have been in a probabilistic fashion. For example, if $\alpha = \sqrt{2}/2$ and $\beta = -\sqrt{2}/2$ then the probability of observing $|x\rangle$ is 0 rather than 1/2, and with the sign of β inverted we observe $|x\rangle$ with probability 1.

4.5 Quantum Parallelism and Realizations

The currently successful trick used in [Shor, 1994, Simon, 1994] is to use a sequence S_n of n unitary operations S (similar to U above) on a register of n bits originally in the all-0 state $|\Psi\rangle = |00\ldots0\rangle$. The result is a superposition of

$$S_n \; |\Psi\rangle = \sum_{x \in \{0,1\}^n} 1/\sqrt{2^n} \;\; |x\rangle$$

of all the 2^n possible states of the register, each with amplitude $1/\sqrt{2^n}$ (and hence probability of being observed of $1/2^n$.) Now the computation proceeds in parallel along the exponentially many computation paths in quantum coherent superposition. A sequence of tricky further unitary operations and observations serves to exploit interference (and so-called entanglement) phenomena to effect a high probability of eventually observing a desired outcome.

Physical realizations of QCC will have to struggle with the fact that the coherent states of the superposition will tend to deteriorate by interaction with each other and the universe, a phenomenon called *decoherence*. In [Unruh, 1995] it is calculated that that QCC calculations using physical realizations based on spin lattices will have to be finished in an extremely short time. For example, factoring a 1000 bit number in square quantum factoring time we have to perform 10^6 steps in less than the thermal time scale $\hbar/kT$ which at 1 K is of order 10^{-9} seconds. Such a QCC computation would need to proceed at optical frequencies. See also [Chuang, *et al.*, 1995].

Another problem is *error correction*: measurements to detect errors will destroy the computation. A novel partial method for error correction has been suggested in [Berthiaume *et al.*, 1994]. A comprehensive discussion on these problems in practically applying QCC is contained in [Landauer, 1995].

References

[Adleman, 1994] L. Adleman, Molecular computation of solutions to combinatorial problems, *Science*, Vol 266, Nov 1994, 1021-1024; A Vat of DNA May Become Fast Computer Of the Future, Gina Kolata in: *The New York Times*, April 11, 1995, Science Times, pp. C1, C10.

[Barenco *et. al*] A. Barenco, C.H. Bennett, R. Cleve, D.P. DiVicenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter, Elementary gates for quantum computation, submitted to *Physical Review A*, March 1995.

[Benioff, 1980–1986] P. Benioff, J. Stat. Phys., 22(1980), 563-591, also *J. Math. Phys.*, 22(1981), 495-507, *Int. J. Theoret. Phys.*, 21(1982), 177-201, *Phys. Rev. Letters*, 48(1982), 1581-1585, *J. Stat. Phys.*, 29(1982), 515-546, *Phys. Rev. Letters*, 53(1984), 1203, *Ann. New York Acad. Sci.*, 480(1986), 475-486.

[Benioff, 1995] P. Benioff, Review of quantum computation, In: *Trends in Statistical Physics*, Council of Scientific Information, Trivandrum, India, To be published.

[Bennett, 1973] C.H. Bennett. Logical reversibility of computation. *IBM J. Res. Develop.*, 17:525–532, 1973.

[Bennett, 1982] C.H. Bennett. The thermodynamics of computation—a review. *Int. J. Theoret. Phys.*, 21(1982), 905-940.

[Bennett, 1989] C.H. Bennett. Time-space trade-offs for reversible computation. *SIAM J. Comput.*, 18(1989), 766-776.

[Bennett, *et al.*, 1992] C.H. Bennett, F. Bessette, G. Brassard, L. Salvail and J. Smolin, Experimental quantum cryptography, *J. Cryptology*, 5:1(1992), 3-28; C.H. Bennett, G. Brassard and A. Ekert, Quantum cryptography, *Scientific American*, Oct. 1992, 50-57.

[Bennett *et al.*, 1993] C.H. Bennett, P. Gács, M. Li, P.M.B. Vitányi and W.H Zurek, Thermodynamics of computation and information distance *Proc. 25th ACM Symp. Theory of Computation.* ACM Press, 1993, 21-30.

[Bernstein and Vazirani, 1993] Bernstein, E. and U. Vazirani, "Quantum complexity theory", *Proc. 25th ACM Symposium on Theory of Computing*, 1993, pp. 11 – 20.

[Berthiaume *et al.*, 1994] A. Berthiaume, D. Deutsch and R. Jozsa, The stabilisation of quantum computations, *Proc. 3rd IEEE Workshop on Physics and Computation (PhysComp '94)*, IEEE Computer Society Press, 1994.

[Brassard, 1994] G. Brassard, Cryptology Column—Quantum computing: The end of classical cryptography? *SIGACT News*, 25:4(Dec 1994), 15-21.

[Chuang, *et al.*, 1995] I.L. Chuang, R. Laflamme, P. Shor, and W.H. Zurek, Quantum computers, factoring and decoherence, Report LA-UR-95-241, Los Alamos National Labs, 1995 (quant-ph/9503007).

[Deutsch, 1985–1992] D. Deutsch, Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. Royal Society London.* Vol. A400(1985), 97-117; see also *Proc. Royal Society London*, A425(1989), 73-90; with R. Josza, *Proc. Royal Society London*, A439(1992), 553-558.

[Feynman, 1982—1987] R.P. Feynman, Simulating physics with computers, *Int. J. Theoret. Physics*, 21(1982), 467-488; Quantum mechanical computers. *Foundations of Physics*, 16(1986), 507-531. (Originally published in *Optics News*, February 1985); Tiny Computers Obeying Quantum Mechanical Laws. In: *New Directions in Physics: The Los Alamos 40th Anniversary Volume*,, N. Metropolis and D. M. Kerr and G. Rota, Eds.,Academic Press,, Boston, 1987, 7-25.

[Fredkin & Toffoli, 1982] E. Fredkin and T. Toffoli. Conservative logic. *Int. J. Theoret. Phys.*, 21(1982),219-253.

[Kiehl, 1994] R.A. Kiehl, Research toward Nanoelectronic computing technologies in Japan, In: Proc. 3rd Workshop on Physics and Computation (PhysComp'94), IEEE Computer Society Press, 1994, 1-4.

[Kissin, 1982–1991] G. Kissin, Measuring Energy Consumption in VLSI Circuits: a Foundation, Proc. 14th ACM Symp. Theor. Comp., 1982, 99-104; Lower and Upper Bounds on the Switching Energy Consumed by VLSI Circuits, *J. Assoc. Comp. Mach.,*, 38(1991), pp. 222-254.

[Kolmogorov, 1965] A.N. Kolmogorov, Three approaches to the definition of the concept 'quantity of information', *Problems in Information Transmission*, **1:1**(1965), 1-7.

[Koppelman, 1995] D.M. Koppelman, A lower bound on the average physical length of edges in the physical realization of graphs, Manuscript Dept ECE, Lousiana State Univ. Baton Rouge, 1995.

[Landauer, 1961] R. Landauer. Irreversibility and heat generation in the computing process. *IBM J. Res. Develop.*, 5:183–191, 1961.

[Landauer, 1991] R. Landauer, Information is physical, *Physics Today,* 44:May(1991), 23-29.

[Landauer, 1994] R. Landauer, Zig-zag path to understanding. In: Proc. 3nd Workshop on Physics and Computation (PhysComp'94), IEEE Computer Society Press, 1994, 54-59.

[Landauer, 1995] R. Landauer, Is quantum mechanics useful? *Proc. Roy. Soc. Lond.*, to be published.

[Lent *et al.*, 1994] C.S. Lent, P.D. Tougaw, W. Porod, Quantum cellular automata: The physics of computing with arrays of quantum dot molecules. In: Proc. 3nd Workshop on Physics and Computation (PhysComp'94), IEEE Computer Society Press, 1994, 5-13; also J. Appl. Phys., 74(1993), 3558, 4077, 6227, 75(1994), 1818.

[Li & Vitányi, 1993] M. Li and P.M.B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications.* Springer-Verlag, New York, 1993.

[Li & Vitányi, 1994] M. Li and P.M.B. Vitányi. *Irreversibility and Adiabatic Computation: Trading time for energy*, submitted.

[Mead & Conway, 1980] C. Mead and L. Conway. *Introduction to VLSI Systems.* Addison-Wesley, 1980.

[Proc. PhysComp, 1981, 1992, 1994] Proc. 1981 Physics and Computation Workshop. *Int. J. Theoret. Phys.*, 21(1982). *Proc. 1992 Physics and Computation Workshop.* IEEE Computer Society Press, 1992. *Proc. 1994 Physics and Computation Workshop.* IEEE Computer Society Press, 1994.

[Schuhmacher, 1994] , B.W. Schumacher, On Quantum coding, *Phys. Rev. A*, in press to appear in 1995; (with R. Josza), A new proof of the quantum noiseless coding theorem, *J. Modern Optics*, 41(1994), 2343-2349.

[Shannon, 1948] C.E. Shannon, A mathematical theory of communication, *Bell System Tech. J.*, **27**(1948), 379-423, 623-656.

[Shor, 1994] Shor, P., Algorithms for quantum computation: Discrete log and factoring, *Proc. 35th IEEE Symposium on Foundations of Computer Science*, 1994, 124-134.

[Simon, 1994] Simon, D., On the power of quantum computation, *Proc. 35th IEEE Symposium on Foundations of Computer Science*, 1994.

[Thompson, 1979] C. Thompson, Area-time complexity for VLSI, Proc. 11th ACM Symp. Theor. Comp., 1979, 81-88.

[Ullman, 1984] J. Ullman, *Computational Aspects of VLSI*, Computer Science Press, Rockville, MD, 1984.

[Unruh, 1995] Unruh, W. G., Maintaining coherence in quantum computers, *Physical Review A*, 51(1995), 992-.

[Upfal and Wigderson, 1987] E. Upfal and A. Wigderson, How to share memory on a distributed system, *J. Assoc. Comp. Mach.*, 34(1987), 116-127.

[Vitányi, 1985] Area penalty for sublinear signal propagation delay on chip, *Proceedings 26th IEEE Symposium on Foundations of Computer Science*, 1985, 197-207.

[Vitányi, 1986] P.M.B. Vitányi, Non-sequential computation and Laws of Nature, In: VLSI Algorithms and Architectures (Proceedings Aegean Workshop on Computing, 2nd International Workshop on Parallel Processing and VLSI), *Lecture Notes In Computer Science 227,* Springer Verlag, 1986, 108-120.

[Vitányi, 1988] P.M.B. Vitányi, Locality, communication and interconnect length in multicomputers, *SIAM J. Computing,* 17 (1988), 659-672.

[Vitányi, 1994] P.M.B. Vitányi, Multiprocessor architectures and physical law. In: Proc. 3rd Workshop on Physics and Computation (PhysComp'94), IEEE Computer Society Press, 1994, 24-29.

[Burks, 1966] J. von Neumann. *Theory of Self-Reproducing Automata.* A.W. Burks, Ed., Univ. Illinois Press, Urbana, 1966.

This article was processed using the LaTeX macro package with LLNCS style