

Time-Space Trade-Offs in Parallel and Neural Computing

Razvan Andonie

Computer Science Department, Wayne State University
431 State Hall, Detroit, MI 48202, USA
andonie@cs.wayne.edu

on leave of absence from:
Department of Electronics and Computers
Transylvania University, 2200 Brasov, Romania

Abstract

To which extend can we speak about time-space trade-offs in the sense that, for a given problem, a faster algorithm requires more space than a slower algorithm? Furthermore, to which extend can we speak about such trade-offs in the context of parallel or neural computing? This paper aims to give an insight to these problems. The parallel computation thesis, first proposed in print in [Ch.S.76], is a relation between parallel time and sequential space. The thesis holds for the parallel model P if *parallel time is polynomially related to sequential space*. How should we state a “neural computation thesis”? Considering the VLSI implementation of a neural network, we suggest here that a “neural computation thesis” has to relate sequential space to parallel time and fan-ins. Meanwhile, we state some open problems and assumptions concerning the representation of logical functions and circuits by neural networks. We focus on how to use time-space trade-offs and polynomial complexity theory in designing VLSI- and size- optimal neural networks.

1. Preliminaries

Our reference computational model is the Turing machine. Let us recall a few basic concepts concerning the computational complexity of Turing machines. It is known that each algorithm is Turing solvable. In the context of function computability, the Church-Turing thesis states that each intuitively computable function is Turing computable. The languages accepted by Turing machines form the recursively enumerable language family L_0 and, according to the Church-Turing thesis, L_0 is also the class of algorithmic computable sets. In spite of its generality, the Turing model can not solve any problem. Recall, for example, that the halting problem is Turing unsolvable: it is algorithmic

undecidable if an arbitrary Turing machine will eventually halt when given some specified, but arbitrary, input.

Complexity theory has developed mainly on the basis of two measures of complexity: space and time. In order to distinguish between work-space and input (or output) space, we will use Turing machines having distinct work-tapes. We measure space as the maximum, over all inputs x , each of size n , of the number of work-tape cells used in processing x . Nondeterminism is defined in the usual way, with space defined to be the maximum, over all inputs x of size n , of the number of work-tape cells used in the accepting computation. We shall consider several classes of languages which can be recognized by Turing machines. $DSPACE(s(n))$ is the class of languages which can be recognized in space $O(s(n))$ by a deterministic Turing machine. $NSPACE(s(n))$ is the class of languages which can be recognized in space $O(s(n))$ by a nondeterministic Turing machine.

Similarly, $DTIME(t(n))$ is the class of languages which can be recognized in time $O(t(n))$ by a deterministic Turing machine and $NTIME(t(n))$ is the class of languages which can be recognized in time $O(t(n))$ by a nondeterministic Turing machine.

We introduce also the classes with their common notations:

$$\begin{aligned} P &= \cup \{DTIME(n^k) \mid k \geq 1\} \\ NP &= \cup \{NTIME(n^k) \mid k \geq 1\} \\ PSPACE &= \cup \{DSPACE(n^k) \mid k \geq 1\} \end{aligned}$$

and with the following relations between them [Bov.C.94], [Gr.H.R.95]:

$$P \subseteq NP \subseteq PSPACE$$

It is not known whether any of these inclusions are proper.

2. Relations between space and time

Clearly, the time is generally not bounded by the same function bounding the space since memory cells can repeatedly be used by the computation. Is it possible to derive other relations between time-complexity and space-complexity classes?

We do know that any deterministic or nondeterministic Turing machine working in space $O(s(n))$ runs in time $O(2^{ks(n)})$ with k constant [Bov.C.94]. This means that an upper bound on space implies an upper bound on time. The consequence is interesting: *the halting problem becomes decidable whenever we have an upper bound on space.*

Conversely, it is clear that an upper bound on time implies an equal upper bound on space: in t steps, at most t tape cells can be scanned! Can we do better? Other important relationships between time and space have been found. Denote by T (respectively, NT) a deterministic (respectively, nondeterministic) Turing machine running in time $O(f(n))$, with $f(n) \geq n^2$, and by T' (respectively, NT') a space-bounded machine that simulates it. Then:

- If T is a one-tape machine, then T' is a one-tape machine requiring space $O(f(n)^{1/2})$ [Hop.U.68].
- If T is a k -tape machine, then T' is a k -tape machine requiring space $O(f(n)/\log(n))$ [Hop.P.V.68].
- If T is a one-tape machine, then T' is a one-tape machine running in time $O(f(n)^{3/2})$ but requiring space $O(f(n)^{1/2})$ [Lis.L.89].
- If NT is a one-tape machine, then NT' is a one-tape machine running in the same time but requiring space $O(f(n)^{1/2})$ [Lor.L.88].

We may conclude that the Turing computational model respects a restricted time-space trade-off. In the following two sections we shall see to which extent we can speak about such a trade-off in the context of parallel and neural computing.

3. The parallel computation thesis

Assume that a time-complexity class of languages has been defined by making use of a parallel model of computation. Is it possible to characterize the same class by a deterministic Turing machine? By putting suitable limitations on both the number and the power of parallel processors, a general rule seems to hold which is known as the *parallel computation thesis* [Ch.S.76]. Denote by Q a parallel model of computation. The thesis holds for the model Q if two polynomials p and q exist such that [Bov.C.94]:

$$P\text{-TIME}(f(n)) \subseteq DSPACE(p(f(n))) \text{ and} \\ DSPACE(g(n)) \subseteq P\text{-TIME}(q(g(n)))$$

where $P\text{-TIME}(f(n))$ denotes the class of languages which can be recognized by model Q in time $O(f(n))$. This is a relation between parallel time and sequential space. In other words, the thesis holds for the model P if *parallel time is polynomially related to sequential space.* The parallel computation thesis was successively verified for a parallel model called vector machine [Pra.S.76], for circuits [Bor.77], and for different types of parallel random access shared memory models [Gold.82].

The parallel computation thesis is a concise and beautiful expression of a time-space relationship. Restating a case where the parallel computation thesis has been proven, we have the following theorem [Quinn94]:

The class of problems solvable in polynomial time by a parallel random access machine (PRAM) is equal to the class of problems solvable in polynomial space by a random access machine (RAM).

On a RAM, the class of problems solvable in polynomial space is PSPACE and the set of problems solvable in polynomial time is P [Bov.C.94]. Since PSPACE is thought to be a much larger class of problems than P, this theorem quantifies the effective improvement made possible by parallelism. A consequence of this theorem is that a PRAM can solve NP-complete problems in polynomial time. For example, the graph coloring problem is NP-complete, yet there is a quadratic time algorithm to solve it [Quinn94]. Nevertheless, an exponential number of processors are used. A time-space trade-off is working: *The exponential time on the RAM is transformed into an exponential space (number of processors) on the PRAM, whereas the polynomial space on the RAM is transformed into a polynomial time on the PRAM.*

This trade-off is easier to understand if we try to restrict both parallel time and parallel space: *If the PRAM has a polynomial number of processors, then the class of problems solvable in parallel polynomial time is P.*

It is easy to define parallel models of computation which do not obey the parallel computation thesis. For instance, if we allow the PRAM to be nondeterministic, the model becomes too powerful and the thesis is not true [Bov.C.94]. Compliance with the parallel computation thesis should be viewed not as a general rule, but rather as an insurance that the parallel model considered is “reasonable”. A number of restrictions have been used in the literature to define “reasonable” machines, including restrictions on the instruction-set and bounds upon processors and time [Par.87].

The parallel computation thesis provides us with a powerful theoretical tool:

- i) We find here some “old” limits of computation: if the number of processors in a PRAM is restricted to some polynomial function of the size of the input, then the problems solvable in parallel polynomial time is P, the set of problems solvable in sequential polynomial time.
- ii) Suppose that we are interested in those problems from P which can be solved in polylogarithmic time by a “reasonable” parallel machine. If a “reasonable” machine is one which satisfies the parallel computation thesis, then these are exactly the members of P which can be solved in polylogarithmic space by a deterministic Turing machine.

4. Time-space trade-offs and the parallel computation thesis

The parallel computation thesis relates parallel time to sequential space. We observed that, when solving the graph coloring problem, a time-space compensation is working. Hence, it seems reasonable to state a parallel computation thesis where *parallel space is polynomially related to sequential time*. Moreover, for the same model of parallel computation, we may relate both: i) parallel time to sequential space, ii) parallel space to sequential time. The question of sharpening the parallel computation thesis to consider simultaneous resource bounds was raised by Borodin [Bor.77]. For example, is the class of problems that simultaneously use polynomial time and

polylogarithmic space equivalent to the class that simultaneously use polynomial numbers of processors and polylogarithmic parallel time? Without answering this question directly, Pippenger [Pip.79] showed that similar simultaneous resource bounds did in fact hold when Turing machine time and reversals were compared to circuit size and depth, and when Turing machine time and space were compared to circuit size and width. Ruzzo [Ruz.79] proved that simultaneous size and depth of circuits is very close related to simultaneous space and time on alternating Turing machines. Stockmeyer and Vishkin [Sto.V.84] proved that size and depth of a uniform unbounded fan-in circuit is equivalent to processors and time of a shared-memory machine (provided that, for the latter, the running-time is at most polynomial in the number of processors), the first pair of resources by a constant multiple, and the second pair by a polynomial.

Other results concerning a parallel computation thesis with simultaneous resource bounds were obtained by Dymond [Dym.80], and Parberry and Schnitger [Par.S.86].

5. Neural networks and Turing machines

McCulloch and Pitts [McC.P.43] asserted that neural networks are computationally universal. A neural network implementation of a Turing machine was provided by Franklin and Garzon [Fra.G.90], [Fra.G.94]. The consequence is, of course, that any algorithmic problem that is Turing solvable can be encoded as a problem solvable by a neural network: *neural networks are at least as powerful as Turing machines*.

The converse has been widely presumed true [Hart.S.87], since computability has become synonymous with Turing computability. Nonetheless, Franklin and Garzon [Fra.G.90], [Fra.G.94] proved that the halting problem, when suitably encoded on a neural network, is solvable by an infinite neural network. The consequence of this property has a fundamental importance: *infinite neural networks are more powerful than Turing machines*. Despite its appearance, this result is not a counterexample to the Church-Turing thesis since the thesis concerns only *algorithmic problems solvable by finite means* [And.96], [And.97].

Meanwhile, the result of Franklin and Garzon is the expression of a time-space trade-off: the infinite time complexity of the halting problem on a Turing machine has been transferred into the infinite number of neurons. In this case, *sequential time is related to parallel space*.

6. Toward a “neural computation thesis”

Verifying the parallel computation thesis for some neural computing models is an open problem. Should we state a “neural computation thesis”? In this case, new time-space relations may be discovered. For instance, in the case of the infinite neural network considered by Franklin and Garzon, do we have a polynomial relation between parallel time and sequential space? Connected to this, can we define a “reasonable” neural model of computation?

Considering feedforward neural networks made of linear threshold gates, the first idea would be to use a trade-off between the size and the depth of such circuits. An optimal hardware implementation of a circuit would then correspond to an optimal value of the size-depth trade-off. In this case, a “neural computation thesis” would be a restatement of the simultaneous resource bounds as considered by Borodin [Bor.77], Ruzzo [Ruz.79], Stockmeyer and Vishkin [Sto.V.84] and, more recently, Impagliazzo *et al.* [Imp.P.S.97]. In all these papers, logical circuits have unbounded fan-in.

However, the size and the depth of a circuit are not the best criteria for ranking different solutions when going for silicon [Bei.D.M.97]. Beside size (number of neurons, or linear threshold gates) and depth (number of layers) several other measures are very important, like: the total number of connections, the total number-of-bits needed to represent the weights and thresholds, the maximum number of the fan-in. Beiu *et al.* [Bei.D.M.97] showed that VLSI- and size- optimal neural networks can be obtained for small (i.e., lower than linear) fan-in values. A large class of logical functions, the $F_{n,m}$ functions, can be implemented in VLSI- optimal (i.e., minimising AT^2 , where A is the area and T is the delay) neural networks of small constant fan-ins. Meanwhile, size-optimal neural implementations of arbitrary logical functions can be obtained for small fan-ins. The last result is based on the following property [Bei.D.M.97]:

Arbitrary logical functions can be implemented in a neural network restricted to fan-in d in $O(n / \log d)$ layers.

This property is a fan-in – depth trade-off.

A “neural computation thesis” has to relate polynomially the Turing machine space complexity to the number of layers in the neural network. Considering the hardware implementation of a neural network, we have to include here also the fan-in number. Therefore, such a thesis has to relate

sequential space to parallel time and fan-ins. Simultaneous space and time on Turing machines may be related to simultaneous size and numbers of layers of neural networks, considering again a limited fan-in.

From the hardware implementation point of view, considering only the main parameters, a “reasonable” neural model of computation is a neural network with limited size, fan-ins and number of layers. A “neural computation thesis” may be respected only for such neural models.

7. Is the efficient neural network implementation of a circuit a P-complete problem?

The class of *P-complete problems* contains polynomial sequential time problems that appear not to admit linear size – polylogarithmic depth parallel implementations [Gr.93], [Gr.H.R.95]. These problems are highly sequential and do not admit fast parallel algorithms. The most fundamental P-complete problem is the Circuit Value Problem (CVP), which consists in computing the output of a circuit for a given input. Versions of the CVP are also P-complete [Gr.93]:

- i) monotone circuit value problem
- ii) monotone, alternating circuit value problem
- iii) monotone, alternating, fan-in and fan-out 2 circuit problem
- iv) monotone, alternating, synchronous, fan-in and fan-out 2 circuit problem
- v) NAND circuit problem
- vi) planar circuit problem
- vii) arithmetic circuit problem

A circuit is *monotone* if it consists of only AND and OR gates. A monotone circuit is *alternating* if on any path from an input to the output gate or to a gate whose outputs are unconnected, the gates alternate between OR and AND gates. Problem iv) is close to a feedforward neural network implementation since “synchronous” means here that each level of gates in the circuit receive inputs only from gates on the preceding level.

What can one say about the VLSI- and size- optimal neural networks with small fan-in values? Defining classes of logical functions for which the VLSI and size optimal neural implementation is not efficient, in the sense that the CVP for these functions is P-complete, is an open problem. From this point of

view, the results obtained by Beiu *et al.* [Bei.D.M.97] are not a surprise. Can we do better? Polynomial completeness theory can give us the lower limits for speedups achieved by neural implementations.

8. Conclusions and future work

We tried to relate results from such different areas as parallel complexity theory, neural networks, circuits and VLSI implementations. Some of these results are classical, like the parallel computation thesis, others are very new, like the optimal neural VLSI implementation concepts. We found out that new theoretical statements are possible if applying parallel complexity theory to neural hardware implementations.

What are the challenges of such results?

- a) We can define new classes of circuits which can be efficiently implemented as neural networks, considering a size – fan-in – depth trade-off.
- b) We can relate open problems in Turing machines and neural complexity.
- c) We can verify the parallel computation thesis for some neural computing models, stating a “neural computation thesis” which establishes a polynomial relation between sequential space and parallel time and fan-ins.
- d) We can define classes of circuits for which the VLSI- and size- optimal neural implementation is not efficient in the following sense: the circuit value problem for these circuits is P-complete.

References

- [And.96] Andonie R.: *The new computational power of neural networks*. Neural Network World, **6**, 1996, 469-475.
- [And.97] Andonie R.: *The “Psychological” Limits of Neural Computation*. In: Dealing with Complexity: A Neural Network Approach, M. Karny, K. Warwick, V. Kurkova (eds.), Springer-Verlag, London, 1997, 252-263.
- [Bei.D.M.97] Beiu V., Draghici S., Makaruk H.E.: *On limited fan-in optimal neural networks*. Proc. 4-th Brazilian Symp. on Neural Networks, SBRN, Goiana, Brazil, December 3-5, 1997, also published as Technical Report LA-UR-97-1567, Los Alamos National Laboratory, 1997.
- [Bor.77] Borodin A.: *On relating time and space to size and depth*. SIAM Journal on Computing, **6**, 1977, 733-743.
- [Bov.C.94] Bovet D.P., Crescenzi P.: *Introduction to the theory of complexity*. Prentice Hall, New York, 1994.
- [Ch.S.76] Chandra A.K., Stockmeyer L.J.: *Alternation*. Proc. 17-th Ann. IEEE Symp. on Foundations of Computer Science, Houston, Texas, 1976, 98-108.
- [Dym.80] Dymond P.W.: *Simultaneous resource bounds and parallel computations*. PhD Thesis, Technical Report TR145/80, Dept. of Computer Science, Univ. of Toronto, 1980.
- [Fra.G.90] Franklin S.P., Garzon M.: *Neural computability*. In: Progress in Neural Networks, vol. 1, Omidvar O. (ed.), Ablex, Norwood, NJ, 1990.
- [Fra.G.94] Franklin S.P., Garzon M.: *Neural computability II* (submitted), 1994. Extended abstract in: Proc. 3rd Int. Joint Conf. on Neural Networks, vol I, Washington DC, 1989, 631-637.
- [Gold.82] Goldschlager L.M.: *A universal interconnection pattern for parallel computers*. Journal of ACM, **29**, 1982, 1073-1086.
- [Gr.93] Greenlaw R.: *Polynomial completeness and parallel computation*. In: Synthesis of Parallel Algorithms, J.H. Reif (ed.), Morgan Kaufmann, San Mateo, California, 1993, 901-953.
- [Gr.H.R.95] Greenlaw R., Hoover H.J., Ruzzo W.L.: *Limits to parallel computation: P-completeness theory*. Oxford University Press, New York, 1995.
- [Hart.S.87] Hartley R., Szu H.: *A comparison of the computational power of neural network models*. In: Proc. IEEE 1st Int. Conf. on Neural Networks, vol III, 1987, 17-22.
- [Hop.U.68] Hopcroft J.E., Ullman J.D.: *Relations between time and tape complexities*. Journal of ACM, **15**, 1968, 414-427.
- [Hop.P.V.68] Hopcroft J.E., Paul J.W., Valiant L.: *On time versus space*. Journal of ACM, **15**, 1968, 414-427.
- [Imp.P.S.97] Impagliazzo R., Paturi R., Saks M.E.: *Size-depth tradeoffs for threshold circuits*.

SIAM Journal on Computing, **26**, 1997, 693-707.

- [Lis.L.89] Liskiewicz M., Lorys K.: *Some time-space bounds for one-tape deterministic Turing machines*. Lecture Notes in Computer Science, Vol. 380, Springer-Verlag, 1989, 297-307.
- [Lor.L.88] Lorys K., Liskiewicz M.: *Two applications of Fürer's counter to one-tape nondeterministic TMs*. Lecture Notes in Computer Science, Vol. 324, Springer-Verlag, 1988, 445-453.
- [McC.P.43] McCulloch W., Pitts W.: *A logical calculus of the ideas immanent in nervous activity*. Bull. Math. Biophys., **5**, 1943, 115-133.
- [Par.S.86] Parberry I., Schnitger G.: *Parallel computation with threshold functions*. Proc. Structure in Complexity Conf., Springer-Verlag, Lecture Notes in Computer Science, vol. 223, Berkeley, California, 1986, 272-290.
- [Par.87] Parberry I.: *Parallel complexity theory*. John Wiley, NY, 1987.
- [Pip.79] Pippenger N.J.: *On simultaneous resource bounds*. Proceedings, In: Fifteenth Ann. Allerton Conf. on Communication, Control and Computing, Monticello, IL, 1977, 25-33.
- [Pra.S.76] Pratt V.R., Stockmeyer L.J.: *A characterization of the power of vector machines*. Journal of Computer and System Sciences, **12**, 1976, 198-221.
- [Quinn94] Quinn M.J.: *Parallel computing: theory and practice*. McGraw-Hill, NY, 1994.
- [Ruz.79] Ruzzo W.L.: *On uniform circuit complexity*. Proc. 20-th Ann. Symp. of Foundations of Computer Science, San Juan, Puerto Rico, 1979, 312-318.
- [Sto.V.84] Stockmeyer L., Vishkin U.: *Simulation of parallel random access machines by circuits*. SIAM Journal on Computing, **13**, 1984, 409-422.