Revista Integración
Escuela de Matemáticas
Universidad Industrial de Santander
Vol. 32, No. 2, 2014, pág. 153–168

The work of Leslie Valiant: alle die Strassen führen nach Strassen

J. Andrés Montoya
Universidad Nacional de Colombia, Departamento de Matemáticas, Bogotá, Colombia.

Abstract. This year Leslie VALIANT becomes sixty five years old; we celebrate his fest with this work in which we analyze some of his major achievements. We focus our attention on those of his works for which a strong influence of Volker Strassen can be easily detected. Strassen's work has had a strong and lasting influence on Valiant. It does not means, as the title could suggest, that the manyfaced, relevant and complex work of Leslie Valiant can be understood as a corollary to Strassen.
Keywords: Complexity, algorithms, parsing, matrix algorithms.
MSC2010: 01AXX, 03D15.

La obra de Leslie Valiant

Resumen. Este año Leslie VALIANT cumple 65 años y nosotros queremos celebrar este importante aniversario con este trabajo en el que se analiza su obra. Centramos nuestra atención en aquellos de sus trabajos en los que una clara influencia de Volker STRASSEN puede ser detectada. Es patente la influencia de Strassen en la obra de Valiant, pero esto no quiere decir que el trabajo de Valiant, complejo y multifacético, sea un simple corolario a la obra del primero.
Palabras clave: Complejidad, algoritmos, análisis sintáctico, algoritmos para matrices.

Valiant is sixty five

We survey the work of Leslie Gabriel VALIANT (born 28 March 1949), one of the most influential theorist within the computer science milieu. Valiant got the Nevalina Medal in 1986 because of his many (early) contributions to computer science. The Nevalina Medal is the Field Medal of computer science. One important feature of Valiant's work, besides its depth, is its visionary character:

[*] E-mail: jamontoyaa@unal.edu.co
Received: 03 February 2014, Accepted: 07 July 2014.
To cite this article: J. Andrés Montoya, The work of Leslie Valiant: alle die Strassen führen nach Strassen, *Rev. Integr. Temas Mat.* 32 (2014), no. 2, 153-168.

- He created, from scratch, *Counting Complexity*: He introduced the class #P and the notion of #P-completeness [37, 38] which allows us to explain, to some extent, why most enumeration problems are intractable.
- He was one of the pioneers of *randomized complexity*, publishing an influential paper in the field as early as 1977 [1].
- He was one of the creators of *machine learning*, a theory that found its first formal ground on Valiant's *probably approximately correct* (PAC) model [24].
- He introduced one of the first sound theoretical models of *parallel computing machine* [40].
- He introduced the notions of *superconcentrator* and *matrix rigidity*, and the concept of *holographic algorithms*.
- He discovered, while a Ph.D. student, an algorithm for context-free language recognition, which is still the asymptotically fastest known [34].

Leslie Valiant received the Nevalina Prize in 1986, the Knuth Prize in 1997, the EATCS Award in 2008 and the ACM Turing Award in 2010 (the Nobel Prize of computer science). He is one of the most influential computer scientists, his work spans several areas of theory and it makes it hard to present a panoramic view of that work. We focus our attention on some few works of Valiant. The choice of the topics is related to the title of this work: We focus our attention on those works of Valiant that are closely related to Strassen's work.

Volker STRASSEN (1936, Düsseldorf-Gerresheim) is an influential German computer scientist. He was awarded the Cantor Medal, the Paris Kanellakis Award and the Knuth Prize (which is aimed to honor the whole work of a computer scientist because of its deep and lasting influence on the development of the field). In despite of this we will not study the work of Volker Strassen; instead, we will concentrate on the influence of Strassen's work on the mathematical development of Leslie Valiant.

Strassen career is crowned by many important achievements:

- Strassen discovered, jointly with Arnold Schönhage [27], an algorithm for integer multiplication based on the fast Fourier transform, which was the fastest known till the discovering of *Fürer's algorithm* in 2007 [10].
- Strassen discovered the first polynomial time randomized algorithm for the recognition of prime numbers (jointly with Robert Solovay) [28]; this achievement of Strassen gained the attention of the algorithm-community towards the revolutionary concept (shaped by his work) of *probabilistic algorithm*.
- He discovered the first subcubic algorithm for the multiplication of integer matrices [26]; this work of Strassen (one of his earliest works) has had, as we will see, a strong influence on Valiant's work.

Those three breakthroughs launched the development of *algebraic complexity,* a subfield of complexity theory aimed to investigate the algorithmic hardness of algebraic computations.

Volker Strassen got his Ph.D. from Göttingen University in 1962 working under the supervision of Konrad Jakobs. His first works are concerned with probability theory and statistics (see [29]). He moved in the early sixties to Berkeley University and went back to Germany in 1966, where he obtained the *Venia Legendi* from Erlangen University, working again under the supervision of Jakobs. Then, he accepted a call from Zürich University, where he switched to algebraic complexity. In 1969 he published his first major result in this new field, a short but very influential paper [26], where he was able to prove that, in despite of the conventional wisdom, gaussian elimination is not optimal: He discovered an algorithm for matrix inversion which is asymptotically faster than gaussian elimination.

Outline of the paper. This paper is organized into four sections: The first one surveys the work of Volker Strassen. The other three sections are related to three stages of Valiant's career: the beginning (the doctoral thesis), the summit (represented by the Nevalina prize), and his later works.

Remark 1. *Some of the material included in this work is conjectural.*

1. Volker STRASSEN: The birth of Algebraic Complexity

Strassen said once (see [30]), that he found lower bounds more interesting than algorithms (influenced as he was by Gödel's theorem). Strassen began, after the publication of [26], a systematic search for lower bounds related to algebraic computations. By the time complexity theory did not exist (we have to wait until 1971, when Cook published his historical paper [3]), given that there was not, among other things, a clear answer to the following question: How can one measure the complexity of general computations? The specific subfield of algebraic complexity had experienced an earlier development, given that there was (and there is) a natural measure of complexity for algebraic computations over rings: the number of algebraic operations that must be performed along the computation. Thus, there did not exist a general framework to deal with lower bounds and because of this Strassen had to focus his attention on *practical complexity* [30], that is, Algebraic complexity.

Strassen is considered the creator of algebraic complexity, but it should be clear that there were some antecedents. Perhaps, the first published work aimed to analyze the number of operations that should be performed in order to solve a given algebraic problem is a paper of Alexander Ostrowski [20]. Ostrowski proved, in that paper, that *Horner's rule* is optimal. To this end he had to introduce the computational model of *straight-line programs*, which has became the main tool in the analysis of algebraic computations over rings.

Let R be a ring and let $f : R^n \to R$ be a function. A straight-line program for $f(X_1, ..., X_n)$ is a sequence of instructions, say $R_1, ..., R_t$, such that for all $i \leq t$, R_i the i^{th} instruction, it has the form

$$v_i \leftarrow y_1^i \circ y_2^i,$$

where, for given $k = 1, 2$ we have that either $y_k^i = v_j$ for some $j \lneqq i$ or $y_k^i = X_s$ for some $s \leq n$, or y_k^i is a scalar (that is: y_k^i is an element of R). Furthermore, we have that $\circ$ belongs to $\{+, -, \times\}$. The *output of the straight-line program* $R_1, ..., R_t$ is the value of v_t.

Straight-line programs constitute a nonuniform model of computation (it means that we need a different program for each input size) which can be used to measure the amount of algebraic operations that must be performed in order to solve a given algebraic problem. Consider the following problem:

Problem 1.1. ($UNIEVAL$: Evaluation of univariate polynomials).

- Input: $\left(\sum\limits_{i \leq n} a_i X^i, c\right)$, where $a_0, ..., a_n, c \in \mathbb{R}$.
- Problem: compute $\sum\limits_{i \leq n} a_i c^i$.

The size of the input $\left(\sum\limits_{i \leq n} a_i X^i, c\right)$ is equal to n, which is the degree of the polynomial $\sum\limits_{i \leq n} a_i X^i$. If we fix a positive integer n, we can use a straight-line program $\mathcal{R}_n = R_1^{(n)}, ..., R_{t_n}^{(n)}$ to evaluate any pair $\left(\sum\limits_{i \leq n} a_i X^i, c\right)$. But given $n \neq m$ and given $\left(\sum\limits_{i \leq n} a_i X^i, c\right)$, $\left(\sum\limits_{i \leq m} b_i X^i, d\right)$, two instances of $UNIEVAL$ of different size, we need two different straight-line programs to evaluate those inputs. Also, if we want to analyze the algebraic complexity of the problem $UNIEVAL$ by means of the model of straight-line programs, we have to consider sequences $(\mathcal{R}_n)_{n \geq 1}$ where, for each $n \geq 1$, $\mathcal{R}_n$ is a straight-line program. Let $(\mathcal{R}_n)_{n \geq 1}$ be one of such sequences, and suppose that $\mathcal{R}_n = R_1^{(n)}, ..., R_{t_n}^{(n)}$. The *running time* of the sequence (of the algorithm encoded by this sequence) is given by the function

$$n \mapsto t_n.$$

Employing such an algorithm on an input of size n requires the computation of t_n algebraic operations. Let $\mathcal{R}_n = R_1^{(n)}, ..., R_{t_n}^{(n)}$ and let $\mathcal{R}_n^+$ and $\mathcal{R}_n^\times$ be the sets

$$\begin{aligned}
\mathcal{R}_n^+ &= \left\{ i \leq t_n : R_i^{(n)} \text{ is equal to } v_i \leftarrow y_1^i \circ y_2^i \text{ and } \circ \text{ belongs to } \{+, -\} \right\}, \\
\mathcal{R}_n^\times &= \left\{ i \leq t_n : R_i^{(n)} \text{ is equal to } v_i \leftarrow y_1^i \circ y_2^i \text{ and } \circ \text{ is equal to } \times \right\}.
\end{aligned}$$

If one wants to distinguish between the number of multiplications required by the algorithm and the number of additions, one can define two different *running time functions.* Let $t_n^+ = |\mathcal{R}_n^+|$ and let $t_n^\times = |\mathcal{R}_n^\times|$. Notice that $t_n \geq t_n^+ + t_n^\times$. We define the $+$-running time of $(\mathcal{R}_n)_{n \geq 1}$ as

$$n \mapsto t_n^+.$$

and the $\times$-running time accordingly.

It is easy to figure out a sequence (*scheme*) $(\mathcal{R}_n)_{n\geq 1}$ of straight-line programs, computing the problem $UNIEVAL$ and such that for all n the equations

$$t_n^+ = n \text{ and } t_n^\times = \frac{n\,(n+1)}{2}$$

hold. Such an scheme corresponds to the naive evaluation of polynomials (the $n+1$ monomials are computed independently and then they are added from left to right). Can we do better? Naive algorithms are, most of the time, not optimal. Also, there must exist much better algorithms (*straight-line schemes*) solving the problem $UNIEVAL$. In 1819 William George Horner [13] described an algorithm (a straight-line scheme) which can be used to solve the problem $UNIEVAL$. The $+$-running time and the $\times$-running time of Horner's scheme are equal to n. Horner's Scheme corresponds to write the polynomial $\sum_{i\leq n} a_i X^i$ (the polynomial to be evaluated) as

$$a_0 + X\,(a_1 + X\,(a_2 + ...X\,(a_{n-1} + Xa_n)))\,.$$

Computation starts with the innermost parentheses using the coefficients of the highest degree monomials and works outward, each time multiplying the previous result by and adding the coefficient of the monomial of the next lower degree. Alexander Ostrowski posed the question of whether any other method of computing $p\,(X)$ could use fewer operations. He proved that n additions are necessary [20]. V. Pan proved, in 1964, that n multiplications are necessary [22]. Thus, Horner's scheme is optimal. Notice that the elementary model of straight-line programs allowed us to analyze the algorithmic complexity of a typical (an ubiquitous) algebraic task: the evaluation of real polynomials. Furthermore, an optimality result was obtained (which is very rare).

The paper [20] could be considered the first published paper in algebraic complexity. Soon after that Strassen began to publish his works on those topics, giving the theory a definitive form.

We trace the influence of Strassen's work on some of the theoretical developments achieved by Valiant. We will observe, along the paper, that Strassen's matrix multiplication algorithm had a depth and lasting influence on Valiant. We will also observe that a theoretical development (like the one by Strassen) targeted to deal with a very specific problem, can have a strong influence on the development of areas of theory that are not linked to it.

2. The Beginning: A doctoral thesis

Leslie G. Valiant made his doctoral studies at Warwick under the supervision of Mike Paterson. His dissertation studies a special class of context-free languages that properly contains the class of deterministic context-free languages. He was interested in the development of decision algorithms for Deterministic Finite-Turn Pushdown Automata [33].

It was around 1971, the year at which Steven Cook published his historical paper [3] (where the class NP and the notion of NP completeness were introduced), when Valiant entered the doctoral program in Mathematics and Computer Science at Warwick. Valiant was interested in context-free languages, parsing problems and decision procedures for formal languages. There was not, by this time, a theoretical framework for the complexity analysis of this kind of problems. The model of straight-line programs and the theoretical developments contained in the already published work of Strassen were tailor-made to deal with a very different class of problems. One can identify the birth of Complexity Theory with the publishing of Cook's paper. Off course, there were many antecedents. In 1936 Alan Turing introduced the model of Turing machines [32], which allowed the formalization of the notion of algorithm. In the sixties, Hartmanis and Stearns (see [11]) settled the basis of algorithm analysis: they showed that the model of Turing machines could be used to measure the efficiency of general algorithms. Then, people began to analyze algorithms and, as consequence, to look for efficient algorithms solving important computational tasks; the concept of lower bound was in the air.

Noam Chomsky introduced, in the fifties, context-free grammars and context free languages as toy models of generative grammars [6]. Chomsky was a linguist interested in the study of natural languages, but his concept of context-free grammar became functional in the development of programming languages. Programming languages are artificial languages with a syntax and a grammar, most of those grammars are context-free. If a language is context-free it holds a context-free grammar which is a succinct code of the whole language and which can be employed to recognize it. Given a context-free grammar G, the recognition problem for G is the problem defined by:

Problem 2.1. $RL(G)$: Recognizing $L(G)$.

- Input: α, where α is a expression (a string of characters).
- Problem: decide if α belongs to $L(G)$.

Remark 2. *We use the symbol $L(G)$ to denote the language generated by G.*

The problem $RL(G)$ can be solved employing a suitable pushdown automaton [12]. It is a sound algorithmic solution, but it could be an *unfeasible* one given that there is not an uniform upperbound on the running time of nondeterministic pushdown automata. People working in the theory of programming languages was interested in the development of efficient algorithms for the recognition of context-free languages. The first important achievements were obtained, in the early sixties, by Earley [9] and Kasami [17], who discovered recognition algorithms that run in time $O(n^3)$. The study of context-free recognition algorithms and context-free parsers was a hot topic in those days and it is not hard to understand why Valiant, being Ph.D student, was interested in those issues.

Valiant's advisor, Mike Paterson, got his Ph.D in 1967 from Cambridge University; his thesis was entitled *Equivalence Problems in a Model of Computation.* Notice the similarity between the topics studied in Paterson's thesis and the topics studied in Valiant's thesis. The advisor of Paterson, David Michael Ritchie Park, was an student of Hartley Rogers and Alonso Church. Thus, he was a typical exponent of *computability (recursion) theory*, the mathematics of computation studied in England by the time. After obtaining his Ph.D. Paterson went to MIT, where he spent three years developing join work with Ian Munro and Larry Stockmeyer. This work was related to the efficient evaluation of polynomials and other algebraic functions [19, 21]. Thus, Paterson discovered at MIT the flourishing field of algebraic complexity. It can be conjectured that Paterson discovered Strassen's algorithm for the multiplication of square matrices with integer entries around this time [26]. And it can also be conjectured that Paterson, when he was back at Warwick, pointed out to Valiant the relevance of Strassen's work. Although Strassen's algorithm was not clearly linked to the work of Valiant, he discovered an unexpected connection of Strassen's work with his own work: Valiant discovered that he could employ Strassen's algorithm to design a subcubic context-free recognition algorithm [34]. The discovery of the first subcubic context-free recognition algorithm, indebted to Valiant, can be considered as the first breakthrough in his career.

2.1. Strassen Algorithm

Strassen algorithm allows one to compute the product of two integer matrices of order n in time $O\left(n^{\log_2(7)}\right)$. The running time of the naive integer matrix multiplication algorithm is $O\left(n^3\right)$. Strassen discovered the first subcubic integer matrix multiplication algorithm [26]. After Strassen many other subcubic algorithms have been discovered, some of them exhibiting a better performance than Strassen algorithm. The current $O\left(n^k\right)$ integer matrix multiplication algorithm with the lowest known exponent k is the Coppersmith–Winograd algorithm[1]. It was presented by Don Coppersmith and Shmuel Winograd in 1990, and it has an asymptotic complexity of $O(n^{2.376})$ [5].

Remark 3. *It can be easily shown that there not exists a $O\left(n^k\right)$ integer matrix multiplication algorithm with k strictly lesser than* 2.

If one tries to multiply two integer matrices of order n using the naive multiplication algorithm (using the definition of matrix product), one has to compute n^3 integer multiplications. The computation of integer multiplications is the most expensive operation to be made, and then, it makes sense to measure the running time of an integer matrix multiplication algorithm in terms of the number of integer multiplications computed by it. Strassen discovered a procedure which allows one to compute the product of two integer matrices of order n, computing at most $O\left(n^{\log_2(7)}\right)$ integer multiplications. Strassen algorithm is based on the following fact:

Let $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$, $B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$ be two 2×2 matrices and let $C =$

[1]Very recently (November 2011), Virginia Vassilevska and Andrew Stothers announced the discovery of matrix multiplication algorithms beating the Coopersmith-Winograd Algorithm. Those works have not been published, although most of the work of Stothers is included in his Ph.D thesis published in 2010; the work of V. Vassilevska was presented at STOC 2012 [46].

$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$ be its product. Strassen wrote down seven equations:

$$\begin{aligned}
\mathrm{I} &= (A_{11}+A_{22})(B_{11}+B_{22}), \\
\mathrm{II} &= (A_{21}+A_{22})B_{11}, \\
\mathrm{III} &= A_{11}(B_{12}-B_{22}), \\
\mathrm{IV} &= A_{22}(-B_{11}+B_{21}), \\
\mathrm{V} &= (A_{11}+A_{12})B_{22}, \\
\mathrm{VI} &= (-A_{11}+A_{21})(B_{11}+B_{22}), \\
\mathrm{VII} &= (A_{12}-A_{22})(B_{21}+B_{22}),
\end{aligned}$$

and he noted that

$$\begin{aligned}
C_{11} &= \mathrm{I}+\mathrm{IV}-\mathrm{V}+\mathrm{VII}, \\
C_{21} &= \mathrm{II}+\mathrm{IV}, \\
C_{12} &= \mathrm{III}+\mathrm{V}, \\
C_{22} &= \mathrm{I}+\mathrm{III}-\mathrm{II}+\mathrm{VI}.
\end{aligned}$$

Notice that the above equations allow one to compute the matrix C computing no more than 7 matrix multiplications. And notice that the above equalities hold if the entries of A and B are square matrices. Given $n \geq 1$, given A, B two matrices of order n and given m such that $n \leq 2^m \leq 2n$ one can deal with those two matrices as if they were matrices of order 2^m. Then, one can reduce the multiplication of A and B to the multiplication of a series of 2×2 matrices whose entries are blocks (submatrices). If one uses Strassen's equations at each stage, one can reduce the number of integer multiplications to $O\left(n^{\log_2(7)}\right)$. Thus, Strassen algorithm can be used to compute in time $O\left(n^{\log_2(7)}\right)$ the product of two $n \times n$ matrices.

It should be noted that Strassen's algorithm can be adapted to obtain subcubic algorithms for matrix inversion and for the computation of determinants. Therefore, we have that Strassen algorithm beats *The Gaussian Elimination Algorithm* which is a cubic algorithm.

Where do Strassen equations come from? How could Strassen discover those striking equations? Legend says that Strassen was trying to prove that Gaussian elimination is optimal, then he began to work hard on the order-two case, and while he was trying to prove that the multiplication of 2×2 integer matrices requires eight integer multiplications he discovered, by chance, his famous equations. This legend could be right, but in despite of this we should be able to answer the following question: How could a mathematician discover those equations? Although this question is related to the analysis of mathematical creativity, which is by no means our theme, we would like to say some things related to those questions, but we prefer to point out to the reader the very interesting entry of wikipedia related to Strassen's algorithm which tries to explain, from a combinatorial point of view, the origin of Strassen's equations [47].

Remark 4. *Strassen's algorithm can be employed for computing the product of two square matrices with entries in an arbitrary ring; it includes the case of real and complex matrices. Unfortunately it is not numerically well-behaved. It is weakly stable but it is not strongly stable* [22].

2.2. A technical aside: Valiant's context-free recognition algorithm

Valiant discovered the first context-free recognition algorithm that runs in subcubic time. It can be considered as his first major result.

Let L be a context-free language; there exists a pushdown automaton $\mathcal{M}$ recognizing L [12]. Can we use $\mathcal{M}$ as a recognition algorithm for L? It depends on $\mathcal{M}$. If $\mathcal{M}$ is deterministic then $\mathcal{M}$ yields a *real-time* recognition algorithm for L, while if $\mathcal{M}$ is a nondeterministic pushdown automaton then $\mathcal{M}$ yields an *unfeasible* exponential time recognition algorithm for L. Can language L be recognized in polynomial time? One can use a polynomial time *parser* to recognize the language L. The history of polynomial time parsers is long and rich. The first polynomial time parsers were discovered by Kasami [17] and Earley [9]; those two parsers are cubic algorithms. The algorithm of Valiant is obtained by cleverly reducing the context-free recognition problem to integer matrix multiplication. This type of ingenious algorithmic reductions from combinatorial to algebraic problems became, after studying Strassen's work, one of the main themes of his work.

Let us discuss the main ideas of Valiant's algorithm. Let L be a context free language. One can compute in subcubic time a *Chomsky grammar*[2] $\mathcal{G}$ such that $L(\mathcal{G}) = L$. Given $\mathcal{G} = (N, \Sigma, S, P)$, one can define a *ring* $A(\mathcal{G}) = (\mathcal{P}(N), \cdot, \cup)$, where $\cdot$ is the operation defined by: given $R, H \subset N$ we have that $R \cdot H$ is equal to

$$\{A \in N : \exists (B \in R) \exists (C \in H) ((A \rightarrow BC) \in P)\}.$$

Given $m \geq 3$ and given $R_1, R_2, ..., R_m \in \mathcal{P}(N)$, we set (in order to enforce associativity)

$$R_1 \cdot ... \cdot R_m = \bigcup_{i=1}^{m-1} ((R_1 \cdot ... \cdot R_i) \cdot (R_{i+1} \cdot ... \cdot R_m)).$$

And given $a \in \Sigma$, we define A_a as $\{A \in N : (A \rightarrow a) \in P\}$.

Fact. *Let $x \in \Sigma^n$. It is easy to check (see reference [33]) that $x \in L(\mathcal{M})$ if and only if $S \in A_{x_1} \cdot A_{x_2} \cdot ... \cdot A_{x_n}$.*

Last fact is the core idea of Valiant's algorithm. Valiant used this idea to design a subcubic recognition algorithm for context-free languages. We fix a context-free language L and we use the symbol $\mathcal{V}(L)$ to denote Valiant's algorithm for L. Algorithm $\mathcal{V}(L)$ is based on the following additional fact:

[2]Recall that a context-free grammar is a tuple $\mathcal{G} = (V, T, S, R)$, where V is a finite set of variables (nonterminal symbols), T is a finite set of constants (terminal symbols), $S \in V$ is the initial variable and R is a finite set of production rules which have the form $X \rightarrow Y$, where $X \in V$ and $Y \in (V \cup T)^*$. Production rules can be employed to *produce* strings. Given $w \in T^*$, if w can be obtained from the initial symbol S by the application of a suitable sequence of production rules, then we say that w belongs to the language $L(\mathcal{G})$, which is the context-free language generated by the context-free grammar $\mathcal{G}$. Given a context-free grammar $\mathcal{G}$ it is possible to compute a context-free grammar $\mathcal{H}$ such that $L(\mathcal{G}) = L(\mathcal{H})$ and such that $\mathcal{H}$ is a Chomsky grammar [12]. A Chomsky grammar is a context-free grammar (V, T, S, R) such that if $X \rightarrow Y$ is a production rule in R, then there exist $R, T \in V$ and $a \in T$ such that either $Y = RT$ or $Y = a$.

Fact. *Let $M_x = [m_{ij}]_{i,j \leq n+1}$ be the square matrix defined by*

$$m_{ij} = \begin{cases} 0, & \text{if } j \neq i+1, \\ A_{x_i}, & \text{if } j = i+1; \end{cases}$$

then $S \in A_{x_1} \cdot A_{x_2} \cdot \ldots \cdot A_{x_n}$ *if and only if* $S \in m_{1n+1}^{(n)}$, *where* $m_{1n+1}^{(n)}$ *is the* $(1, n+1)$ *entry of the matrix* $(M_x)^n$ *(the arithmetic operations are computed in the ring* $A(\mathcal{G})$*).*

Algorithm $\mathcal{V}(L)$ works, on input x, as follows:

1. Compute the matrix M_x.
2. Compute, **using Strassen's algorithm** and *fast exponentiation*, the matrix $(M_x)^n = \left[m_{ij}^{(n)}\right]_{i,j \leq n+1}$.
3. If $S \in m_{1n+1}^{(n)}$ accept the input, otherwise reject.

The above description of the algorithm $\mathcal{V}(L)$ is a very brief description which highlights the role played by Strassen's algorithm. It can be argued that the knowledge of Strassen's algorithm was essential in the development of $\mathcal{V}(L)$, but it should be clear that this work of Valiant cannot be understood as a simple corollary to Strassen's results.

3. The Summit: A medal and a discourse

Leslie Valiant received the Nevalina Medal in 1986. The Nevalina Medal is awarded each four years along the international congress of mathematicians. Thus, one could think that the Nevalina medal is the Field Medal of computer science and then it should be considered as one of the highest honors achievable within this research community.

The awarding of this medal is one of the most important moments in Valiant's career. It is an interesting coincidence that Volker Strassen was the speaker in charge of presenting Valiant's main work. Thus, Strassen is connected to (at least) two of the most important moments in Valiant's career: the beginning and the summit.

After receiving his Ph.D degree Valiant quickly abandoned his first research field: Context-free languages and parsing. The discovery of Strassen algorithm left an indelible mark in Valiant. He switched to algebraic complexity, where he made many important contributions. Some of those contributions explain the choice of Valiant as the second (in the history) Nevalina prize holder; some of those contributions were analyzed by Strassen in his discourse. We claim that the mathematical evolution of Valiant (from parsing to algebraic complexity, from algebraic complexity to counting complexity, from counting complexity to randomization, from randomization to the theory of parallel computation) is determined, to some large extent, by his early discovery of Strassen' s work. Let us give a panoramical (and very succinct) review of Valiant's work in the period 1974-1986.

1. Valiant introduced in [35] the fundamental concept of *superconcentrator*. Superconcentrators are graphs of very high connectivity closely related to *expander graphs*

[7]. It is important to remark that the main goal of Valiant, when he introduced this new concept, was proving that *The Fast Fourier Transform Algorithm* of Cooley and Tukey [4] is optimal.

Let m be a positive integer. An m-superconcentrator is a directed graph with m input and m output nodes, such that for every $r < m$ any r input nodes may be connected to any r output nodes in some order by r disjoint directed paths. Are there superconcentrators with a linear number of edges? If there are not superconcentrators of linear size then the discrete Fourier transform cannot be computed in linear time, proving this would be a first major step towards an optimality proof for the algorithm of Cooley and Tukey. Valiant observed that any straight-line algorithm for computing the Discrete Fourier Transform of order m yields an m-superconcentrator of a size proportional to the length of the algorithm. Then we have that The Fast Fourier Transform Algorithm yields m-superconcentrators of size $O\left(mlogm\right)$ and any improvement of the Fast Fourier Transform would lead to m-superconcentrators of still smaller size. Does the minimal size of m-superconcentrators grows like $mlogm$? By the previous remarks a positive answer would yield an optimality proof for the Fast Fourier Transform. It was the main goal of Valiant, but he proved the opposite: he proved that there exist m-superconcentrators of size linear in m. Superconcentrators of linear size have became useful tools in the information and communication sciences far beyond their original purpose. We would like to mention that superconcentrators were the technical basis for the first construction of error-correcting codes that are optimal to within constant factors in all aspects: linear distance, constant rate, and linear-time encoding and decoding [25]. Moreover, they lead to the *expander revolution* in computer science which has been crucial in the research on derandomization and in the development of distributed algorithms for sorting, searching, and routing, among many other applications.

2. Valiant realized that most counting problems[3] can be understood as evaluation problems: Most counting problems reduce to the computation of algebraic objects such as determinants, permanents or multivariate polynomials. Valiant also realized that most evaluation problems are counting problems. Then, he developed (almost simultaneously) *Counting Complexity* and his own theory of algebraic computation, establishing a framework for understanding which algebraic formulas can be evaluated efficiently. In analogy with the Boolean complexity classes P and NP, his theory characterizes the difficulty of computing fundamental functions in

[3]Let L be a problem in NP. We know that there exists a polynomial function $p\left(X\right)$ and a relation R such that given x, an instance of L, we have that

$$x \in L \text{ if and only if there exists } y \text{ such that } \left|y\right| = p\left(\left|x\right|\right) \text{ and } \left(x, y\right) \in R.$$

The relation R allows one to associate to L the counting problem $\#L$ defined by

- *Input:* x, *where* x *is an instance of* L.
- *Problem: compute* $\left|\left\{y : \left|y\right| = p\left(\left|x\right|\right) \text{ and } \left(x, y\right) \in R\right\}\right|$.

The problem $\#L$ is at least as hard as the problem L (if we can count the number of solutions (*certificates*) then we can decide if there exists at least one solution) and one could conjecture that those two problems have the same complexity. Interesting enough it is not the case; Valiant proved that there exist decision problems in NP such that their associated counting problems are very much harder [38].

linear algebra, namely the Determinant and the Permanent. Those two lines of research, opened by Valiant, set the stage for some of the most exciting subsequent developments in computational complexity, such as the development of interactive proofs for problems beyond NP: his definition of the class #P [37], his proof that the Permanent is complete for this class [38], and the special properties of the Permanent (such as random-self-reducibility) played a key role in unveiling the power of interactive proofs, PCPs, program checking and more.

Finally, we would like to mention that the research on counting problems leads Valiant to study randomized algorithms. He discovered (together with Vijay Vazirani and Mark Jerrum) an important connection between the existence of sampling algorithms and the existence of approximation schemes [16]. This connection is the ground basis of *The Markov Chain Monte Carlo Counting Method*, which is the most robust technique discovered up to the date for the design of randomized algorithms for approximate counting [14].

These three important developments are not a full account of the work developed by Valiant in the period 1974-1986, but they are three highlights of this work. We would like to notice that all those three lines of research are related to Strassen's work, given that they originated in the study of algebraic-numerical problems. We conjecture that Valiant arrived to the study of randomized algorithms via counting complexity, and that he arrived to the study of counting problems via algebraic complexity, which in turn gained his attention thanks to the study of Strassen's work. Thus, we claim that most of the breakthroughs achieved by Valiant in this period are originated, to some extent, in his deep knowledge of Strassen's work.

4. Hopefully, it is not the end: going back to the roots.

Valiant developed a fruitful an manyfaced work in complexity theory in the period 1986-2000 (after the nevalina prize):

- He introduced the PAC model of machine learning [24]. This model has had enormous influence on artificial intelligence and many areas of computing practice, such as natural language processing, handwriting recognition, and computer vision.
- He studied the different models of parallel and distributed computing and he made important contributions in this field. He discovered a randomized routing algorithm which can be efficiently used to minimize congestion effects in communication networks [40]. Randomized routing provides a way of avoiding congestion in sparse networks, something unavoidable for any deterministic algorithm. This algorithm and the beauty subsequent analysis of it exposed how randomization can (should be) used in the design, analysis and administration of communication networks. His novel ideas and the challenges he posed for parallel computing [39] shaped the direction of the field.

And then, with the beginning of the century, he switched to computational neurosciences focusing on understanding memory and learning. [41, 42]. Although most of his later work is not directly related to complexity theory, he introduced in this period an important concept that belongs to this area of computer science: the concept of *holographic algorithm* [44, 45].

Holographic algorithms are counting algorithms which resemble the classical *Kasteleyn algorithm* [18] for the counting of perfect matchings in planar graphs[4]. It is so, and it should be so, given that holographic algorithms (also called *holographic reductions*) are used to reduce a given (and seemingly intractable) counting problem to the counting of perfect matchings in planar graphs. However, the reductions are holographic, namely they are many-to-many (as opposed to the standard many-to-one reductions which are commonly used), and seem to use cancellation in a mysterious way. We would like to discuss an important aspect of this theory that is related to the main theme of this article: This last theory of Valiant owes a big debt to Strassen.

Valiant arrived to the concept of holographic algorithms via quantum complexity. He began studying classical circuits which can be employed to simulate quantum circuits [43]. Let us quote Valiant [44]:

Holographic algorithms are inspired by the quantum computational model [Deutsch, 1985; Bernstein and Vazirani, 1997]. However, they are executable on classical computers and do not need quantum computers. They can be understood best, perhaps, in terms of cancellations in classical computation. ***Strassen's algorithm for matrix multiplication [Strassen, 1969] offers an early striking example of the power of computations that compute extraneous terms only to cancel them later.*** *It is known that cancellations can provide exponential speedups in computations, and in the several cases that have been analyzed, linear algebra algorithms for computing the determinant play a major role [Valiant, 1980; Jerrum and Snir, 1982; Tardos, 1987].*

Also, as Valiant says, Strassen's algorithm was an important source of inspiration for the discovery of holographic algorithms, the last (up to the date) work of Valiant on complexity theory (and hopefully not the very last).

5. Concluding remarks

It has been said that we are dwarfs standing on the shoulders of giants. Leslie Valiant is a gigantic dwarf who found in Strassen's shoulders a very fine standpoint. We have briefly reviewed three important moments in Valiant career: the beginning, the summit and his last works on complexity theory. We have observed that the influence of Strassen's work is ubiquitous and that this influence can be easily traced.

Acknowledgement. Thanks to Universidad Nacional de Colombia, thanks to DIF Ciencias R.P. 16860.

[4]Kasteleyn algorithm reduces the counting of perfect matchings to the computation of Pfaffian determinants. In the planar case Pfaffian orientations can be computed in polynomial time, but it is not longer true in the nonplanar case: Recall that Valiant proved that the counting of perfect matchings is $\#P$ complete [38]

References

[1] Angluin D. and Valiant L., "Fast probabilistic algorithms for Hamiltonian circuits and matching", *Proceedings of STOC*, (1977), 30-41

[2] Bernstein E. and Vazirani U., "Quantum complexity theory", *SIAM J. Comput.* 26 (1997), no. 5, 1411-1473.

[3] S. Cook., "The complexity of theorem-proving procedures", *Proceedings of STOC*, (1971), 151-158.

[4] Cooley J. and Tukey J., "An algorithm for the machine calculation of complex Fourier series", *Math. Comput.* 19 (1965), 297-301.

[5] Coppersmith D. and Winograd S., "On the asymptotic complexity of matrix multiplication", *SIAM J. Comput.* 11 (1982), no. 3, 472-492.

[6] Chomsky N., "On certain formal properties of grammars", *Information and Control* 2 (1959), no. 3, 137-167.

[7] Chung F., "Spectral graph theory", in *CBMS Regional Conference Series in Mathematics*, 92, American Mathematical Society, (1997).

[8] Deutsch D., "Quantum theory, the Church-Turing principle, and the universal quantum computer", *Proc. Roy. Soc. London Ser. A* 400 (1985), 97-117.

[9] Earley J., "An Efficient Context-Free Parsing Algorithm", *Commun. ACM* 13 (1970), no. 2, 94-102.

[10] Fürer M., "Faster integer multiplication", *SIAM Journal on Computing* 39 (2009), no. 3, 979-1005.

[11] Hartmanis J. and Stearns R., "The complexity of recursive sequences", *Proceedings of FOCS* (1964), 82-90.

[12] Hopcroft J. and Ullman J., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.

[13] Horner W., "A new method of solving numerical equations of all orders, by continuous approximation", *Philosophical Transactions of the Royal Society of London*, (1819), 308-335.

[14] Jerrum M. and Sinclair A., "The Markov chain Monte Carlo method: an approach to approximate counting and integration", in *Approximation Algorithms for NP-hard Problems* (ed. Dorit Hochbaum), PWS, (1996).

[15] Jerrum M. and Sni M., "Some Exact Complexity Results for Straight-Line Computations over Semirings", *Journal of the ACM* 29 (1982), no. 3, 874-897.

[16] Jerrum M., Valiant L. and Vazirani V., "Random Generation of Combinatorial Structures from a Uniform Distribution", *Theo. Comput. Sci.* 43 (1986), 169-188.

[17] Kasami T., "An efficient recognition and syntax-analysis algorithm for context-free languages", in *Scientific report AFCRL-65-758*, Air Force Cambridge Research Lab, Bedford, MA, (1965).

[18] Kasteleyn P., "The statistics of dimers on a lattice", *Physica* 27 (1961), 1209-1225.

[19] Munro I. and Paterson M., "Optimal Algorithms for Parallel Polynomial Evaluation", *FOCS*, (1971), 132-139.

[20] Ostrowski A., "On two problems in abstract algebra connected with Horner's rule", *Studies in Mathematics and Mechanics Presented to Richard Von Mises*, (1954), 40-48.

[21] Paterson M. and Stockmeyer L., "Bounds on the Evaluation Time for Rational Polynomials", *FOCS*, (1971), 140-143.

[22] Pan V., *How to Multiply Matrices Faster*, New York, Springer-Verlag, 1982.

[23] Pan V., "Some schemes for computation of polynomials with real coefficients", (Russian), *Dokl. Akad. Nauk. SSSR* 127 (1959), 266-269.

[24] Pitt L. and Valiant L., "Computational limitations on learning from examples", *J. ACM* 35 (1988), no. 4, 965-984.

[25] Spielman D., "Linear-time encodable and decodable error-correcting codes", *IEEE Trans. Inform. Theory* 42 (1996), no. 6, 1723-1731.

[26] Strassen V., "Gaussian elimination is not optimal", *Numerische Mathematik* 14 (1969), no. 3, 354-356.

[27] Strassen V. and Schönhage A., "Schnelle Multiplikation Grosser Zahlen", *Computing* 7 (1971), 281-292.

[28] Strassen V. and Solovay R., "A fast Monte-Carlo test for primality", *SIAM Journal on Computing* 6 (1977), 84-85.

[29] Strassen V., "A converse of the law of the iterated logarithm", *Z. Warscheinlichkeitstheorie verw. Geb.* 4 (1966), 265-268.

[30] Strassen V., "My Latest Talk", Slides available at http://cosec.bit.uni-bonn.de/?id=574.

[31] Tardos E., "The gap between monotone and nonmonotone circuit complexity is exponential", *Combinatorica* 7 (1987), 141-142.

[32] Turing A., "On Computable number with an application to the Entscheidung Problem", *Proceedings of the London Mathematical Society* 2 (1937), no. 42, 230-265.

[33] Valiant L., "The Equivalence Problem for Deterministic Finite-Turn Pushdown", *Automata Information and Control* 25 (1974), no. 2, 123-133.

[34] Valiant L., "General Context-Free Recognition in Less than Cubic Time", *Journal of Comput. and Syst. Sci.* 10 (1975), no. 2, 308-315.

[35] Valiant L., “Graph-Theoretic Properties in computational Complexity”, *J. Comput. Syst. Sci.* 13 (1976), no. 3, 278-285.

[36] Valiant L., “Graph-theoretic arguments in low-level complexity”, in *Proc. 6th MFCS*, (1977), 162-176.

[37] Valiant L., “The Complexity of Enumeration and Reliability Problems”, *SIAM J. Comput.* 8 (1979), no. 3, 410-421.

[38] Valiant L., “The Complexity of Computing the Permanent”, *Theor. Comput. Sci.* 8 (1979), 189-201.

[39] Valiant L., “Parallel computation”, in *Proc. 7th IBM Symposium on Mathematical Foundations of Computer Science*, (1982).

[40] Valiant L., “A scheme for fast parallel communication”, *SIAM J. Comput.* 11 (1982), no. 2, 350-361.

[41] Valiant L. “A neuroidal architecture for cognitive computation”, *J. ACM* 47 (2000), no. 5, 854-882.

[42] Valiant L., “Memorization and Association on a Realistic Neural Model”, *Neural Computation*, 17 (2005), no. 3, 527-555.

[43] Valiant L., “Quantum Circuits That Can Be Simulated Classically in Polynomial Time”, *SIAM J. Comput.* 31 (2002), no. 4, 1229-1254.

[44] Valiant L., “Holographic Algorithms (Extended Abstract)”, *FOCS*, (2004), 306-315.

[45] Valiant L., “Holographic Algorithms”, *SIAM J. Comput.* 37 (2008), no. 5, 1565-1594.

[46] Vassilevska V., “Multiplying matrices faster than Coopersmith-Winograd”, *To appear in proceedings of STOC*, (2012).

[47] http://en.wikipedia.org/wiki/Strassen_algorithm [citado el 24 de julio de 2014]