Analog Parallel Computational Geometry

Frank Dehne*§, Boris Flach†, Jörg-Rüdiger Sack*§, Natana Valiveti*¶

* *School of Computer Science*
Carleton University
Ottawa, Canada K1S 5B6

† *Nuclear Energy Research Institute*
Rossendorf, Postfach 10
O-8061 Dresden, Germany

Abstract

This paper presents a novel approach to *Parallel Computational Geometry* by using networks of *analog* components (referred to as *analog networks* or *analog circuits*). Massively parallel analog circuits with large numbers of processing elements exist in hardware and have proven to be efficient architectures for important problems (e.g. constructing an associative memory). In this paper it is demonstrated how Parallel Computational Geometry problems can be solved by exploiting the features of such analog parallel architectures. Using massively parallel analog circuits requires a radically different approach to geometric problem solving because (1) time is continuous instead of the standard discretized stepwise parallel processing, and (2) geometric data is represented by analog components (e.g. voltages at certain positions of the circuit) instead of the usual digital representation.
The paper presents *analog* parallel algorithms for the following geometrical problems: minimum weight triangulation of planar point sets or of polygons with holes, minimum rectangular partitions of rectilinear polygons with holes, and determining for a given line segment set a subset of non-intersecting line segments of maximum total length. The proofs given in this paper provide ranges for the circuit parameters for which the circuits are guaranteed to produce a feasible solution. Such analysis has previously been unavailable for Hopfield-net circuits. The paper also includes experimental results which demonstrate that, in practice, our analog parallel circuits produce high quality outputs.

1 Introduction

Parallel (as well as sequential) computational geometry has thus far focussed on processing digital data in discrete time steps. *Analog Parallel Computational Geometry*, as introduced in this paper, manipulates *analog* geometric data in *continuous time*. The two approaches are radically different and require both different architectures and different problem solving techniques.

The manipulation of analog geometric data in continuous time is how the brain is believed to solve geometric problems. May be it is in this distinction that we will find the explanation to why many geometric problems require considerable algorithmic effort but are solved by humans with surprising speed and ease. Consider, for example, the problems of determining whether two planar point sets are linearly separable, or determining a collision-free motion of an object in the presence of obstacles.

The introduction of analog computational geometry arises out of the necessity to bridge the software-hardware gap for processing analog geometric data. Hardware is readily available (see below) while little, if any, work has been done on the design of computational geometry algorithms for such architectures.

---

§ Research partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).
¶ Supported by an NSERC Postgraduate Scholarship.

We found analog parallel computational geometry to be particularly attractive when solving problems with a sequential high time complexity.

As architecture we consider a class of analog circuits known as *Analog Hopfield Nets* [2, 5]. These analog parallel architectures are called artificial *analog neural nets* because they exploit the massively parallel analog local processing and distributed representation properties that are believed to exist in the brain. The analog network (also referred to as *analog circuit*) consists of a large number of extremely simple analog processing elements which are essentially analog amplifiers. These amplifiers (also referred to as *neurons*) are connected via feedback circuits consisting of wires, resistors, and capacitors. A schematic diagram of such an analog circuit is shown in Figure 1 (see Section 2 for more details). Analog Hopfield Nets are not just theoretical models of biological neurons but are also attractive architectures from a practical perspective. They have hardware realizations in VLSI (CCD and NMOS) and fiber optics technologies [13, 15-18]. The architectural simplicity of analog circuits allows to build circuits with very large numbers of neurons. Applications of great practical value include, for example, analog associative memory cells ([17] pp.58-72). The dynamics of an analog circuit, without the constraints of enforced discrete time steps, can provide for virtually instantaneous outputs even to some hard computational problem; see e.g. [5-7, 17]. The main challenge is to exploit the dynamic behavior of the feedback loops connecting the amplifiers in such a way that the system actually solves the given problem.

The contribution of this paper is to introduce Parallel *Analog* Computing to the field of Parallel Computational Geometry by demonstrating how *geometric* problems can be solved by exploiting the features of existing analog architectures. The techniques developed are completely different from those developed for solving computational geometry problems in standard digital parallel models of computation. In particular, we present analog circuit designs for the following geometrical problems: *minimum weight triangulation* of *planar point sets* or of *polygons with holes*, *minimum rectangular partitions* of *rectilinear polygons with holes*, and *determining for a given line segment set a subset of non-intersecting line segments of maximum total length*. In the standard digital model most of these problems either have high polynomial time solutions or are conjectured/known to be NP-hard/complete.

The analog circuits described in the previous literature are pure heuristics evaluated through experiments only. Circuit parameters are determined by experiment, making statements about the correctness of these solutions impossible. For example, Hopfield's solution for the traveling salesman problem [5-7] neither guarantees that the reported tour is optimal nor that it is even valid (and sometimes the circuit does actually report invalid "tours"). In contrast to the previous literature we give, in our circuit analysis, rigorous proofs of the feasibility of the solution. For example, we show that our minimum weight triangulation circuit always produces a valid triangulation and that the energy of the circuit, which is known to be minimized when the circuit reaches a global minimum in a stable state, is proportional to the total length of the selected edges (plus a fixed constant). We have not found any such circuit analysis in the existing literature describing analog circuits for other problem areas.

Efficient solutions to computationally hard problems are by nature heuristics. The main advantage of analog circuits, in contrast to standard digital heuristic methods, is that analog circuits produce (in general) virtually instantaneous results. The disadvantage for the designer is that the solution must be expressed within the particular framework of the system of differential equations describing the dynamic

behavior of such a circuit. This poses an additional challenge compared to using standard heuristics or optimization techniques.

The remainder of this paper is organized as follows. Next, in Section 2, we give a brief review of Analog Hopfield Nets. In Section 3 we present for the minimum weight triangulation problem a detailed description of the circuit design, analytically derive sufficient conditions on the circuit's parameters to always produce a feasible solution, and prove the correctness of the circuit. The design and analysis of analog circuits for the other geometric problems listed above is discussed in Section 4. We have simulated and tested our circuits on a SPARC workstation and on a Transputer Network. The experimental results, also included in this paper, show that our circuits produce high-quality outputs.

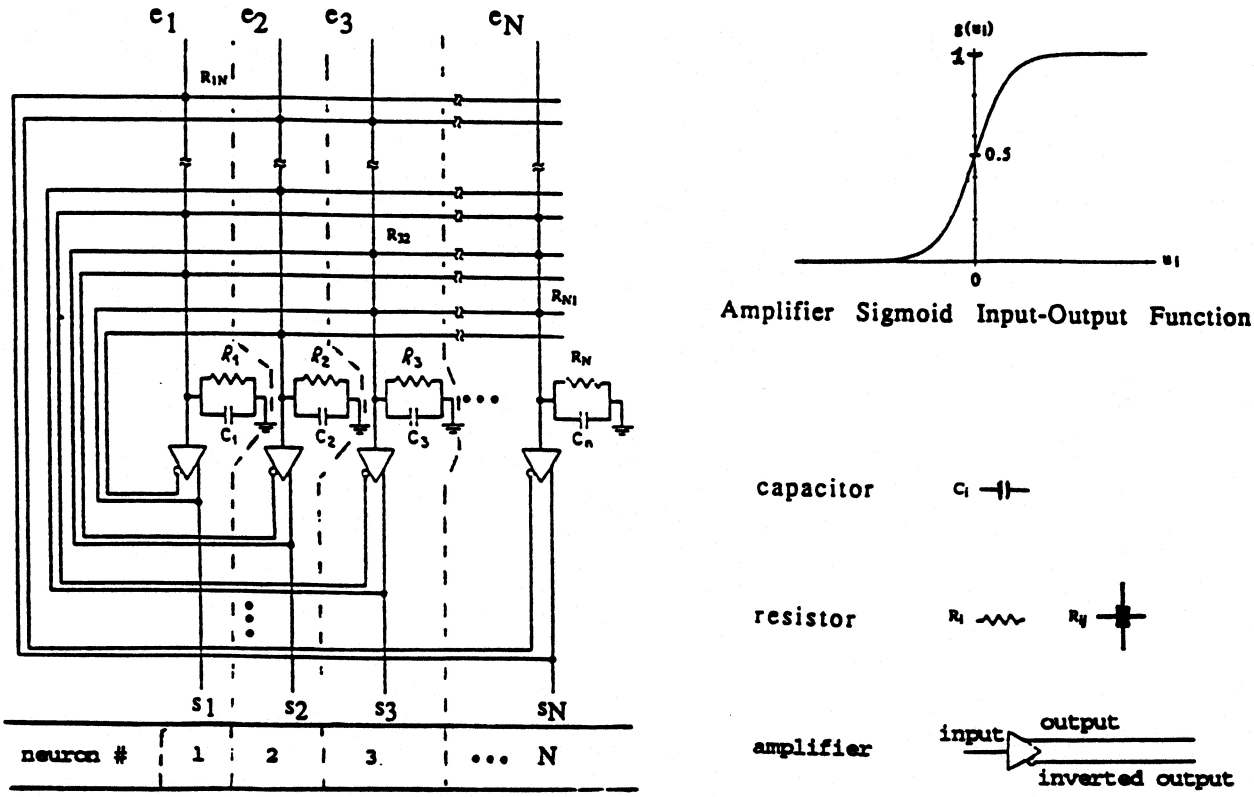


Figure 1. An Analog Neural Circuit With N Neurons.

2 Analog Hopfield Nets

The following discussion of the *Analog Hopfield Net* [2, 5] refers to the net's circuit diagram given in Figure 1. The subcircuits separated by adjacent dashed lines are called *neurons* (the circuit was originally designed as a simplified electronic model for biological neurons). Each neuron i has an output s_i, an inverted output $-s_i$, and an input line. For each pair (i,j) of neurons, either s_j or $-s_j$ is connected to the input line of neuron i via a resistor of resistance R_{ij}. Define T_{ij} to be $1/R_{ij}$ if s_j is connected to the input line of neuron i and to be $-(1/R_{ij})$ otherwise. All inputs to a neuron i, weighted by the respective inverse

resistances T_{ij}, as well as an external input e_i, are added on its input line and create an amplitude referred to as u_i. The neuron's output s_i is the output of an analog amplifier with input u_i. The input/output behavior of the amplifier is described by $s_i = g(u_i)$, where $g(u_i)$ is a strictly monotone increasing sigmoid function as shown in Figure 1. For neuron i, the values e_i, u_i and s_i are referred to as its *bias*, *internal state* and *state*, respectively; T_{ij} is called the *weight* of the connection between neuron j and neuron i.

An analog neural circuit is "programmed" by encoding the problem input into a set of resistance, capacitance, and voltage values for the resistors, capacitors and external input lines, as well as defining the initial internal state of the circuit (in terms of voltage levels at specific internal positions). Starting at this initial state, the internal feed-back loops cause a dynamic change of the system which can be described by a system of differential equations. Under certain circumstances, the dynamics are such that a final stable state will be reached, which is called the *equilibrium*. The dynamic behavior of the Hopfield Net is described by the following system of differential equations:

$$C_i \dot{u}_i = \sum_{j=1}^{n} T_{ij} s_j - \frac{u_i}{R_i} + e_i \qquad , i=1, ..., N. \tag{1}$$

The main challenge is to encode the problem in such a way that the circuit's behavior is predictable and that it actually solves the given problem, i.e. it converges to an equilibrium state which encodes a solution to the problem. The advantage of this approach is that, even with many neurons, the time for an analog circuit to settle into a stable state is typically extremely small.

In order to facilitate the analysis of the dynamic behavior of the circuit, the differential equations (1) are normalized to $\dot{\underline{u}} = - \underline{u} + T \underline{s} + \underline{e}$ (2) where $\underline{u} = (u_1, ..., u_n)^t$, $\underline{s} = (s_1, ..., s_n)^t$, $0 \le s_i \le 1$, $\underline{e} = (e_1, ..., e_n)^t$, T is the $n \times n$ matrix of T_{ij} values, and $s_i = g(u_i)$ for $g(u_i) = \frac{1}{2} (\tanh (u_i / \beta)) + 1)$, $0 < \beta < 1$ (3); see Figure 1. For the remainder of this paper let $\Delta \ge 19$ be a "large " value for which $\tanh(\Delta) \equiv 1$ and, hence, $g(\Delta \beta) \equiv 1$. For a symmetric T matrix, $T_{ij} = T_{ji}$, with 0 diagonal elements, $T_{ii} = 0$, it has been shown [5-7] that

$$E(\underline{s}) = - \frac{1}{2} \underline{s} T \underline{s} - \underline{e} \underline{s} + \sum_i \int_0^{s_i} g^{-1}(s) \, ds \tag{4}$$

is a Lyapunov function [1] for (2). That is, $\dot{E}(\underline{s}) \le 0$ for all $\underline{s}$ (5) and the Hopfield Net migrates to a stable state $\underline{s}$ with $\dot{\underline{s}} = \underline{0}$ (6). The function $E(\underline{s})$ is equivalent to the circuit's energy when it is in state $\underline{s}$.The state space of all possible states $\underline{s} = (s_1, ..., s_n)^t$ over which the circuit operates is the interior of the n-dimensional (real-valued) hypercube $[0,1]^n$. The case when the amplifier gain curve $g(u_i)$ is narrow, more precisely when β converges to 0, is called the *high gain limit*. It has been shown in [5, 6, 19] that, in the high gain limit, for non-degenerate T matrices, every stable state $\underline{s}$ has the property that $E(\underline{s})$ is a local minimum (7) and $\underline{s}$ converges to a corner of the n-dimensional hypercube (8). That is, every s_i converges to either 1 or 0. In the first case, neuron i is called *selected*, otherwise it is called *unselected*.

3 Minimum Weight Triangulation

Let $S = \{p_1, ..., p_n\}$ be a planar set of n distinct points p_i in general position. Consider a weight function assigning a positive weight to every possible edge connecting two points of S (in many cases, the weight of an edge is defined as its length). A *minimum weight triangulation* of S is a maximal set of non-intersecting straight-line segments (edges), whose endpoints are in S, such that the total weight of all

selected edges is minimized; see e.g. [14]. It is an open problem whether the minimum weight triangulation problem for point sets in the plane is NP-complete [3, 4, 12]. The minimum weight triangulation problem has several applications. Recently it was also shown that the minimum weight triangulation problem for a class of extremely flat convex polygons is dual to the problem of constructing optimal binary search trees with zero key access probabilities [9]. Thus this work may also have applications to data structuring.

In Section 3.1 we describe the construction of an analog circuit for solving the minimum weight triangulation problem. Our system always converges to an equilibrium state. We prove that every stable state of the circuit corresponds to a triangulation of the given point set. Furthermore, we show that minimizing the weight of the triangulation is equivalent to minimizing the energy of the circuit.

In Section 3.2 we present results of an experimental study illustrating the performance of the circuit.

3.1 Analog Circuit Design and Analysis

Our analog circuit, called $TN(S)$, for the minimum weight triangulation problem is an analog neural circuit with $N = \frac{n(n-1)}{2}$ neurons, referred to as neurons 1, 2, ..., N. Each edge connecting two points of S is assigned to a unique neuron; the edge assigned to neuron i will be referred to as $edge_i$.

In an equilibrium state reached by the circuit, an edge $edge_i$ is called *selected* if and only if the corresponding neuron i is selected. The set of selected edges is the *output* produced by the circuit. It will be shown that the output of the circuit is always a (valid) triangulation of the point set. We will also prove that minimizing the energy of our circuit is equivalent to minimizing the weight of the triangulation.

In a preprocessing phase we compute the weights, l_i, of all edges $edge_i$, and the maximum weight, l_{max}. Furthermore, we determine for each pair of edges whether or not they *intersect properly* (i.e. they intersect at a point which is not a vertex). We now define the circuit by setting T, $\underline{e}$, and an initial state $\underline{s}$.

Circuit $TN(S)$ for Minimum Weight Triangulation of a Point Set S:

Select constants $\beta > 0$, $\gamma > 0$, $r > 0$, $B > 0$, $C_1 > 0$, and $C_2 > 0$ with the following six properties:

$B \geq (C_1 + \Delta\beta) / (1-\gamma)$ (9)

$C_1 > C_2 + 2\Delta\beta$ (10)

$r << 0.5$ (11)

$C_2 >> 1$ (12)

$\beta << 1$ ($\beta \to 0$) (13)

$\gamma << 1/2$ (14)

We define the **T-matrix** as follows:

$$T_{ij} = -B\, X_{ij}\, (1-\delta_{ij}) \tag{15}$$

where $\delta_{i,j}$ is the Kronecker symbol and

$$X_{i,j} = \begin{cases} 1 & \text{if the edes } edge_i \text{ and } edge_j \text{ intersect properly} \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

For each neuron i we set the **bias** e_i to:

$$e_i = C_1 - C_2 \frac{l_i}{l_{max}}. \tag{17}$$

The **initial state** of the system is set to:

$$s_i = 0.5 + random \tag{18}$$

where *random* is a random number with the property :

$$-r \leq random \leq r. \tag{19}$$

Practical choices of β, γ, r, B, C_1, and C_2 are discussed in Section 3.3. We now study the dynamic behavior of the circuit $TN(S)$. First, we state the following observation:

Observation 1.

(1) T is a symmetric matrix, i.e. $T_{ij} = T_{ji}$ for all $1 \le i,j \le N$.

(2) $T_{ii} = 0$ for all $1 \le i \le N$.

Thus, $TN(S)$ will reach an equilibrium state; see (4) to (6). Since (13) ensures that we operate the circuit in the high gain limit, every equilibrium state has the property that every s_i converges to either 0 or 1, and $E(\underline{s})$ is a local minimum; see (7) and (8). The remainder of this section discusses the dynamic behavior of $TN(S)$ with respect to our goal of computing a minimum weight triangulation of S.

Let $\underline{s}$ be an equilibrium state, $u_i = g^{-1}(s_i)$ for all i, then from (2) and (6) follows

$$\forall\, i \qquad u_i = \sum_j T_{ij}\, s_j + e_i \ . \tag{20}$$

Lemma 2 *Let $edge_i$ and $edge_k$ be two edges that are selected in an equilibrium state $\underline{s}$ (for the high gain limit). Then $edge_i$ and $edge_k$ do not intersect properly.*

Proof: Assume, by contradiction, that there is a stable state $\underline{s}$ where two selected edges $edge_i$ and $edge_k$ are intersecting properly, i.e. $X_{ik} = 1$. Since neurons i and j are selected, s_i and s_k converge to 1; hence, $s_i \ge 1-\gamma$ and $s_k \ge 1-\gamma$. From (20), together with (9) and (17), we obtain

$$\begin{aligned} u_i &= \sum_j T_{ij}\, s_j + e_i = -Bs_k - \sum_{j \ne k} BX_{ij}\,(1-\delta_{ij})s_j + e_i \\ &\le -Bs_k + e_i \le -B\,(1-\gamma) + e_i \\ &\le -(C_1 + \Delta\beta) + \left(C_1 - C_2 \frac{l_i}{l_{max}}\right) \\ &\le -\Delta\beta. \end{aligned}$$

Thus, $s_i = g(u_i) \le g(-\Delta\beta)$, that is, neuron i is unselected; a contradiction. □

Lemma 3 *Let $\underline{s}$ be an equilibrium state (for the high gain limit), and consider an arbitrary $edge_i$. If all edges (properly) intersecting $edge_i$ are unselected, then $edge_i$ is selected.*

Proof: Consider an equilibrium state $\underline{s}$ and a particular neuron i. Assume that all edges which are intersecting the neuron's $edge_i$ are unselected. Then it follows from (15) that any neuron j is either unselected or has value $T_{ij} = 0$. Thus, for the high gain limit, $\Sigma_j T_{ij} s_j = 0$. From (20) and (17), we obtain

$$u_i = \sum_j T_{ij}\, s_j + e_i \ge C_1 - C_2 \frac{l_i}{l_{max}}$$

Since $l_i / l_{max} \le 1$, it follows from (10) that $u_i \ge C_1 - C_2 \ge \Delta\beta$. Thus, $s_i = g(u_i) \ge g(\Delta\beta)$, which implies that neuron i is selected. □

Lemma 4 *Let s be an equilibrium state, then for the high gain limit the energy E converges to $D_1 + D_2 * (\sum_{i\ selected} l_i)$ for some constant D_1 and $D_2 = \frac{C_2}{l_{max}} > 0$.*

Proof: By equation (4) the energy of the circuit is given by:

$$E = -\frac{1}{2} \sum_i \sum_j T_{ij}\, s_i\, s_j \ - \sum_i e_i\, s_i + \sum_i \int_0^{s_i} g^{-1}(s)\ ds \ .$$

From Lemmas 2 and 3 it follows that the first term converges to zero. For the high gain limit, i.e. $\beta \rightarrow 0$, the third term also converges to zero [5]. Hence, for the high gain limit, we obtain from (17)

$$E = -\sum_i e_i s_i = -\sum_i (C_1 - C_2 \frac{l_i}{l_{max}}) s_i$$

$$= -C_1 \sum_i s_i + \frac{C_2}{l_{max}} \sum_i l_i s_i$$

Since Lemmas 2 and 3 show that the selected set of edges is a maximal set of non-intersecting edges, i.e. a triangulation, it follows that the number of selected edges is $3n$-h-3, where h is the number of vertices on the convex hull of the given point set. Hence, using (12), we obtain

$E = D_1 + D_2 (\sum_{i \text{ selected}} l_i)$ with $D_1 = -C_1 (3n\text{-}h\text{-}3)$ and $D_2 = \frac{C_2}{l_{max}} > 0$. □

Theorem 5 *TN(S) will always converge to a stable state. In the high gain limit, a stable state of TN(S) represents a triangulation of the point set S. Minimizing the weight of the reported triangulation is equivalent to minimizing the value of the circuit's energy function.*

Proof: Follows from Observation 1 and Lemmas 2-4. □

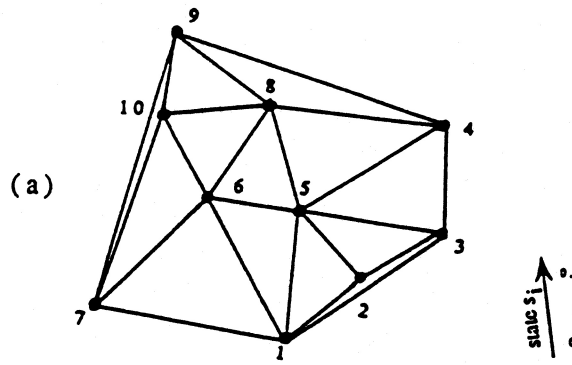
Note that, the above correctness proof for our analog neural circuit is "unusual" compared to the previous literature. We have not found any such circuit analysis in the existing literature on analog circuits for other problem areas. The circuits described in the previous literature are pure heuristics evaluated through experiments only; see Section 1.

3.2 Experimental Results

An analog circuit always migrates towards a state that minimizes its energy. An unfortunate property of analog Hopfield Nets (and analog circuits in general) is that the circuit's energy might stabilize in a local minimum which is not necessarily the global minimum. Due to the high dimensionality of the problem, the current literature considers an analytical evaluation infeasible. Thus, it is necessary that the quality of the results produced by a circuit is verified experimentally. For this, we have implemented an Analog Hopfield Net simulator running on a SUN SPARC workstation as well as a parallelized version running on a Transputer Network. The simulator is essentially a numerical integrator for the system of differential equations (2). An additional front-end program converts a point set into an analog circuit according to (9)-(19), and the final stable state of the circuit back into a set of line segments. We selected the following constants: $\Delta = 19$, $\beta = .1$, $\gamma = .01$, $r = .01$, $B = 8600$, $C_1 = 8500$, $C_2 = 8000$. It is easy to see that these constants are consistent with equations (9) to (14). The variable β determines how sharply the sigmoid function rises. It is important to set β small enough to ensure that the circuit operates in the high gain limit. (For all our experiments β=0.1 was sufficient. Note that β may not be equal to 0, because then the sigmoid function degenerates to a non-differential function, in which case the circuit may not converge at all.) From Lemma 4 it follows that the energy of the system is proportional to $C_2 (\sum_{i \text{ selected}} l_i)$. Hence, C_2 determines the shape of the energy function and should be set to a large value to create steep valleys.

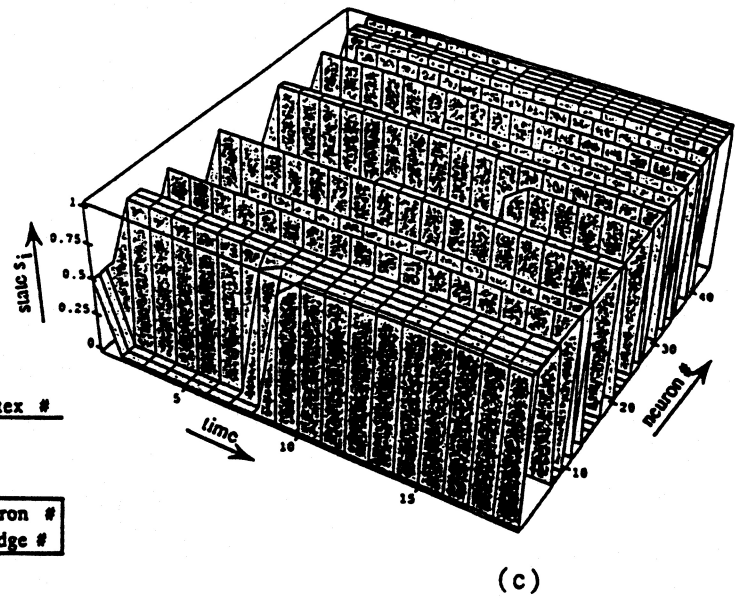
It is instructive to follow the behavior of our triangulation circuit on a particular example as illustrated in Figure 2. Figure 2a shows an example point set S and its minimum weight triangulation. For our circuit $TN(S)$, a neuron i is associated with every pair of points (i.e. a possible edge of a minimum

weight triangulation). The assignment of neuron numbers to vertex pairs is shown in Figure 2b. The dynamic behavior of $TN(S)$, from its initial state to a final stable state, is shown in Figure 2c. For every neuron i, the value of s_i is shown as a function of time. All neurons start at a random value for s_i which is close to 0.5, and in the final stable state each neuron has a value s_i close to either 0 or 1. The state s_1 of neuron 1, for example, first drops from around 0.5 to approx. 0, and then in time steps 8 an 10 it rises to a value close to 1. The final states of all neurons represent the minimum length triangulation in Figure 2a.



	1	2	3	4	5	6	7	8	9	10	vertex #
1		4	5	1	3	2	6	7	8	9	
2			10	11	12	13	14	15	16	17	
3				18	19	20	21	22	23	24	
4					25	26	27	28	29	30	neuron # = edge #
5						31	32	33	34	35	
6							36	37	38	39	
7								40	41	42	
8									43	44	
9										45	
vertex #	(edges of min. length triangulation highlighted)										

(b)



(c)

Figure 2. Circuit $TN(S)$ converging to a Minimum Weight Triangulation.
(a) Point Set and Its Minimum Weight Triangulation. (b) Assignment of Possible Edges To Neurons.
(c) State Vector $\underline{s}$ as a Function of Time, From Its Initial State To A Final Stable State.

Statistical results obtained by executing circuit $TN(S)$ repeatedly on different random point sets of size n are shown in Figure 3. We compare the average weight of the triangulation produced by our triangulation circuit (W_{TN}) with the average weight of the actual minimum weight triangulation (W_{OPT}) and, in addition, with the average weight of a random triangulation (W_{RT}). The values shown have a variance of about 9%. The main result of the data displayed in Figure 3 is that, for our tests, the difference between the optimum weight and the weight of the triangulation produced by our circuit is less than 2% (variance <2.5%). Computing the exact minimum weight W_{OPT} for $n \geq 30$ turned out to be impossible, even with a parallel algorithm running on an Intel iPSC/860 hypercube. The fact that for $n \geq 30$ the W_{TN} value increases only very slowly leads us to conjecture that it will stay very close to the optimum weight. In this context, another interesting observation is that for $n \geq 30$ the weight produced by our circuit is more than 50% better than a random triangulation, with steadily increasing difference.

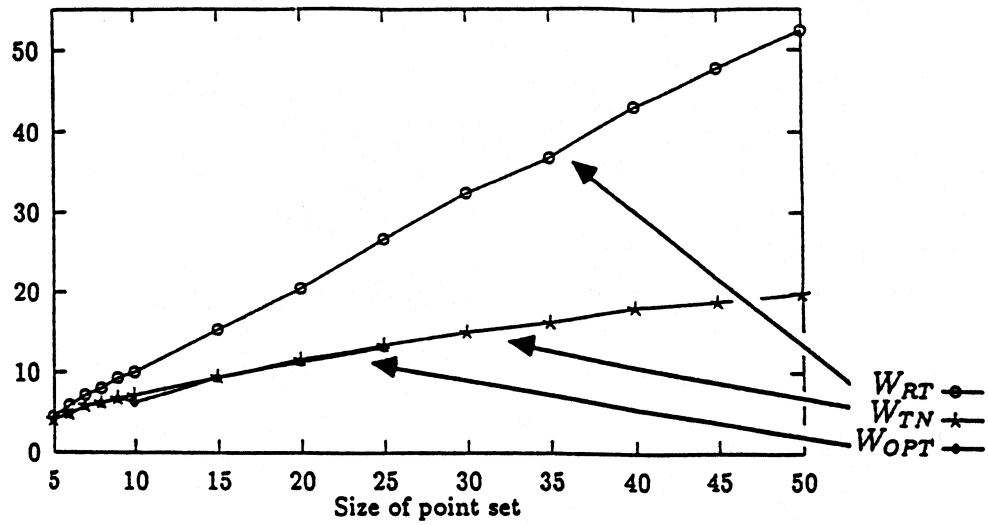


Figure 3. The Triangulation Circuit Compared to the Optimum and to Random Triangulations: Average Weight of the Triangulation Produced by our Triangulation Circuit (W_{TN}), Average Weight of the Minimum Weight Triangulation (W_{OPT}), and Average Weight of a Random Triangulation (W_{RT}).

4 Analog Neural Circuits For Other Geometric Problems

In this section, we outline the design of analog neural circuits for the other geometric problems listed in Section 1. Due to space limitations, we can only give brief expositions of our results. More details will be provided in the final version of this paper.

The analog circuit described in Section 3 can be modified to compute a ***minimum weight triangulation for a simple polygon P with holes***. The holes may be polygonal or simply points. To our knowledge, no polynomial time algorithm is known for this problem. (This is in contrast to the case of polygons without holes for which a polynomial time algorithm exists [8].) Consider the set S of vertices of P and use the circuit (for S) described in Section 3 with the following two modifications: Change equation (15) to $T_{ij} = -B\, X_{ij}\, Y_{ij}\, (1-\delta_{ij})$ where $Y_{ij} = 0$ if at least one of the edges represented by neurons i or j is a boundary edge of P, or lies (partially) outside P, and $Y_{ij} = 1$ otherwise. For the initial state, set $s_i = 0$ for all neurons i corresponding to an edge that lies (partially) outside P, and use (18)-(19) to set the initial state of all other neurons. It follows analogously to the analysis given in Section 3 that such a circuit produces a triangulation where all boundary edges of P are selected and all edges outside P are unselected. The energy of the circuit is proportional to the total weight of the selected edges (plus some constant).

Minimum rectangular partition of a rectilinear polygon with holes: Consider the problem of partitioning a rectilinear polygon into non-overlaping rectangles using the minimum amount of "ink". More precisely, let P be a rectilinear polygon with holes (rectilinear or point holes). We wish to partition

P into rectangles whose interiors are non-intersecting so that the total length of all edges inserted for the partitioning is a minimum. If a given rectilinear polygon is hole-free then polynomial time algorithms exist; but even when point holes are inserted the problem is NP-complete [11]. It is interesting to observe that minimizing the number of rectangles is different from minimizing the total edge length [10].

Next, we describe the construction of an analog circuit for determining a minimum length rectangular partition of a rectilinear polygon with holes. The input to the problem is the description of the polygon P (with its holes) with a total number of vertices equal to n. The output is a set of rectangles which represent a rectangular partition of P. Consider the *grid* induced by P and defined as all horizontal and vertical line segments inside P connecting one vertex of P to the boundary of P. A *grid-point* is the intersection point of a horizontal and vertical line segment of the grid, or of a horizontal (vertical) line segment and the boundary of P. The grid has at most $O(n^2)$ vertices. Each solution rectangle has its vertices on the grid. Thus there are at most $O(n^4)$ possible rectangles. Some rectangles may lie (partially) outside P and will be discarded; all other rectangles are termed *candidate rectangles*. The task of determining whether a rectangle is a candidate rectangle or not is trivial. Furthermore, in a preprocessing phase, we determine for each pair of rectangles whether or not they intersect. The circuit consists of $N = O(n^4)$ neurons. Each neuron is assigned one of the possible candidate rectangles. Select constants $\beta > 0$, $\gamma > 0$, $r > 0$, $B > 0$, $C_1 > 0$, and $C_2 > 0$ subject to (9)-(14). Set $T_{ij} = -B\, Z_{ij}\, (1-\delta_{ij})$ where $Z_{ij} = 1$ if the two candidate rectangles associated with neurons i and j intersect, and $Z_{ij} = 0$ otherwise. The bias is set to $e_i = C_1 - C_2\, (o_i / o_{max})$ where o_i refers to the circumference of the rectangle associated with neurons i, and $o_{max} = max_i\{o_i\}$. The initial state is set as defined in (18) and (19). It is easy to see that this circuit has properties analogous to the ones described in Observation 1, Lemmas 2-4, and Theorem 5. Thus, a valid partitioning is selected, and the energy of the circuit is proportional to the total length of the inserted edges (plus some constant).

Maximum length subset of non-intersecting line segments: Consider the problem of selecting from a given set S of n line segments a subset S' of non-intersecting line segments such that the total length of the selected line segments is maximized. A straight forward adaptation of the circuit presented in Section 3, solves this problem as well.

5 Acknowledgement

The authors would like to thank Michel Gastaldo for helping with the implementation of preliminary versions of the Hopfield Net simulator. We also thank Daryl Graf for helpful discussions.

6 References

[1] N. Chetayav, *The stability of motion*, Pergamon Press, New York, 1961.

[2] M. Cohen and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 13, 1983, pp. 815-825.

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

[4] M.R. Garey and D. S. Johnson, Private communication, November 1991.

[5] J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. of the National Academy of Sciences*, Vol. 81, 1984, pp. 2554-2558.

[6] J. Hopfield and D. Tank, ""Neural" computation of decisions in optimization problems," *Biological Cybernetics*, Vol. 52, 1985, pp. 141-152.

[7] J. Hopfield and D. Tank, "Computing with neural circuits: a model," *Science*, Vol. 233, 1986, pp. 624-633.

[8] G. T. Klincsek, "Minimal triangulations of polygonal domain," *Ann. Discrete Math.*, Vol. 9, 1980, pp. 121-123.

[9] C. Levcopoulos, A. Lingas, and J.-R. Sack, "Nearly optimal heuristics for binary search trees with geometric generalizations," *Theoretical Computer Science*, Vol. 66, No. 2, 1989, pp. 181-203.

[10] A. Lingas, "The power of non-rectilinear holes," in Proc. *9th Int. Colloq. on Automata, Languages and Programming*, 1982, Springer Verlag, Lecture Notes in Computer Science, Vol. 140.

[11] A. Lingas, R. Y. Pinter, R. L. Rivest, and A. Shamir, "Minimum Edge Length Partitioning of Rectilinear Polygons," in Proc. *20th Annual Allerton Conf. on Comm., Control, and Computing*, 1982, pp. 53-63.

[12] E. L. Lloyd, "On triangulations of a set of points in the plane," in Proc. *18th IEEE Conference on Foundations of Computer Science*, 1977, pp. 228-240.

[13] C. Mead, *Analog VLSI and neural systems*, Addison-Wesley, Reading, MA, 1989.

[14] F. P. Preparata and M. I. Shamos, *Computational Geometry: an Introduction*, Springer Verlag, New York, Berlin, Heidelberg, Tokyo, 1985.

[15] J. Sage, "Artificial neural system implementations using CCD and NMOS technologies," in Proc. *Workshop on Artificial Neural Systems*, Jet Propulsion Laboratory, 1987, pp. 72-84.

[16] J. Sage, K. Thompson, and R. Withers, "An artificial neural network integrated circuit based on NMOS/CCD principles," in Proc. *AIP Conference: Neural Networks for Computing*, No. 151, New York, 1986, J. Denker (Ed.), American Insitute of Physics, pp. 381-385.

[17] P. K. Simpson, *Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations*, Pergamon Press, 1990.

[18] M. Sivilotti, M. Emerling, and C. Mead, "VLSI architectures for implementation of neural networks," in Proc. *AIP Conference: Neural Networks for Computing*, No. 151, New York, 1986, J. Denker (Ed.), American Insitute of Physics, pp. 408-413.

[19] D. Tank and J. Hopfield, "Simple "neural" optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Transactions on Circuits and Systems*, Vol. 33, 1986, pp. 533-541.