COMPUTATIONAL GEOMETRY ON ANALOG NEURAL CIRCUITS

Frank Dehne, Boris Flach,
Jörg-Rüdiger Sack, Natana Valiveti

SCS-TR-201, JANUARY 1992

School of Computer Science, Carleton University
Ottawa, Canada, KIS 5B6

Computational Geometry on Analog Neural Circuits

Extended Abstract

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

1 Introduction

In this paper we investigate how *analog circuits* can be utilized for solving *Computational Geometry* problems. The main feature of analog devices is that they manipulate analog data in continuous time, as opposed to standard digital (sequential or parallel) machines which manipulate digital data in discrete time steps. The circuits considered in this paper belong to a class of analog neural circuits known as *Analog Hopfield Nets* [3, 6], originally designed as an electronic model of biological neurons. These circuits consist of extremely simple analog processing elements which are essentially analog amplifiers. The amplifiers 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).

The main contribution of this paper is to demonstrate that analog circuits are useful computing devices for Computational Geometry. In particular, we present 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*, finding the smallest ε so that two given point sets are *ε-congruent via translation*, and determining for a given line segment set a *subset of non-intersecting line segments of maximum total length*. In the standard digital model these problems either have high polynomial time solutions or are conjectured/known to be NP-hard/complete.

In Section 3 we give, for the minimum weight triangulation problem, a detailed description of the circuit design, derive sufficient conditions on the circuit's parameters to always produce a feasible solution, prove the correctness of the circuit, and experimentally demonstrate the quality of the solution. Due to space limitations, the design and analysis of analog circuits for the other geometric problems listed above can only be outlined; this is done in Section 4.

The architectural simplicity of analog circuits allows the building of Analog Hopfield Nets with large numbers of neurons. Hardware realizations exist in VLSI (CCD and NMOS) and fiber optics technologies; see e.g. [14, 16-19]. The dynamics of an analog circuit, without the constraints of enforced discrete time steps, can provide extremely fast solutions even to hard computational problems. Such behavior has been observed, for example, for the traveling salesman and other NP-hard optimization problems; see e.g. [6-8, 18]. Efficient solutions to computationally hard problems are by nature heuristics; in contrast to standard (digital) heuristic methods, however, analog neural circuits produce, in general, virtually instantaneous results (no learning involved!). For the design of an analog circuit, the additional

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

challenge is to express the heuristic method within the framework of the system of differential equations describing the dynamic behavior of the circuit. Our solutions illustrate design techniques for solving geometric problems within that framework. Furthermore, in the circuit analysis, we give rigorous proofs of the feasibility of the solution; we have not found any such circuit analysis in the existing literature describing analog circuits for other problem areas. The circuits described in the previous literature are pure heuristics evaluated through experiments only. For example, Hopfield's solution for the traveling salesman problem [6-8] neither guarantees that the reported tour is optimal nor that it is even valid (and sometimes the circuit does actually report invalid "tours"). In our analysis we prove that, e.g., 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 stable state, is proportional to the total length of the selected edges (plus a fixed constant).

Any analog circuit must be verified experimentally, since the circuit's energy can settle in either a global or a local minimum. (Due to the high dimensionality of the system of differential equations, an analytical evaluation is extremely hard. We have not found any such analysis in the previous literature.) We have simulated and tested our circuits on a SPARC workstation and on a Transputer Network. The experimental results, included in this paper, show that our circuits do indeed produce solutions of very good quality. Next, in Section 2, we will describe Analog Hopfield Nets.

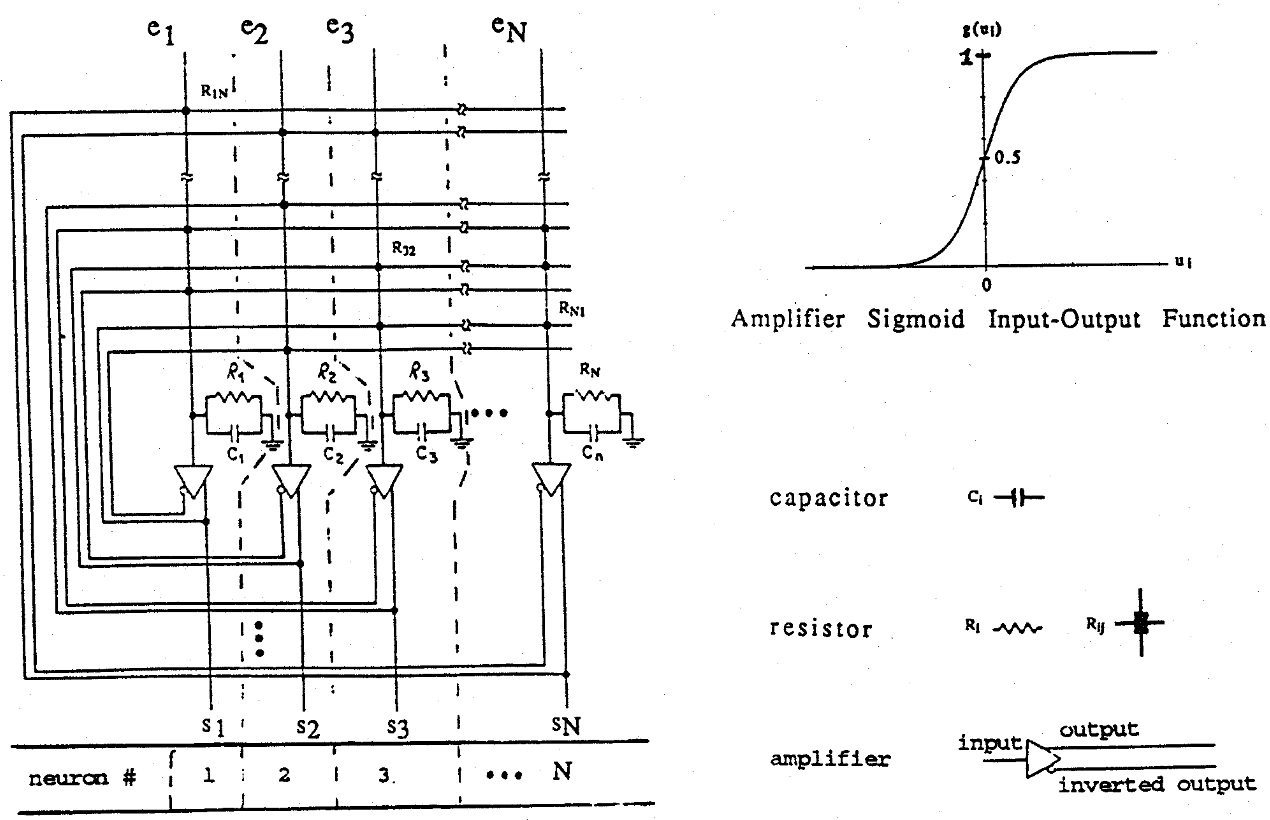


Figure 1. An Analog Neural Circuit With *N* Neurons.

2 Analog Hopfield Nets

The following discussion of the *Analog Hopfield Net* [3, 6] 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 simple 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 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:

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

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 convergeces 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 $\underline{u} = (u_1, \ldots, u_n)^t$, $\underline{s} = (s_1, \ldots, s_n)^t$, $0 \le s_i \le 1$, $\underline{e} = (e_1, \ldots, e_n)^t$, 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/β)) + 1), $0 < \beta < 1$ (3); see Figure 1. For the remainder of this paper let $\Delta \ge 19$ be a "large " value for which $\tanh(\Delta) \cong 1$ and, hence, $g(\Delta\,\beta) \cong 1$. For a symmetric T matrix, $T_{ij} = T_{ji}$, with 0 diagonal elements, $T_{ii} = 0$, it has been shown [6-8] 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 [2] 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 $\underline{s} = (s_1, \ldots, s_n)^t$ 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 [6, 7, 20] 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 $S = \{p_1, \ldots, p_n\}$ 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. [15]. It is an open problem whether the minimum weight triangulation problem for point sets in the plane is NP-complete [4, 5, 13]. 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 [10].

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 neuron 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)	$C_2 >> 1$	(12)
$C_1 > C_2 + 2\Delta\beta$	(10)	$\beta << 1 \ (\beta \to 0)$	(13)
$r << 0.5$	(11)	$\gamma << 1/2$	(14)

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

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

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

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

For each neuron i we set the **bias** e_i to: $$e_i = C_1 - C_2 \frac{l_i}{l_{max}} . \qquad (17)$$

The **initial state** of the system is set to: $$s_i = 0.5 + random \qquad (18)$$

where *random* is a random number with the property : $$-r \le random \le r . \qquad (19)$$

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

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

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

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

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 $l_i / l_{max} \leq 1$, it follows from (10) that $u_i \geq C_1 - C_2 \geq \Delta\, \beta$. Thus, $s_i = g(u_i) \geq g(\Delta\beta)$, 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* $D_1 + D_2 *$ $(\sum_{i \text{ selected}} l_i)$ *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. $\beta \to 0$, the third term also converges to zero [6]. 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

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

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.

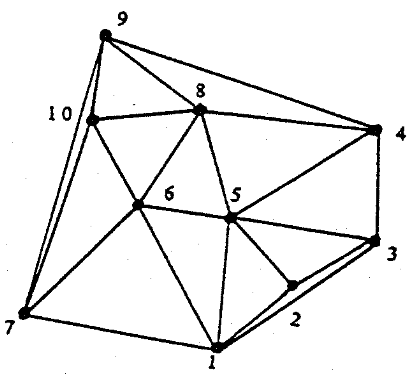
3.2 Experimental Results

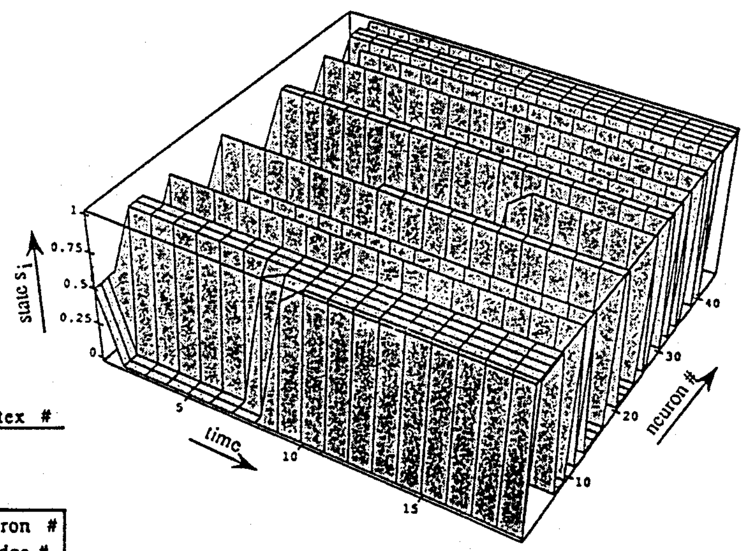
An analog circuit always migrates towards a state that minimizes its energy. An unfortunate property of analog Hopfield Nets (and analog circuits in general) is that the circuit's energy might stabilize in a local minimum which is not necessarily the global minimum. Due to the high dimensionality of the problem, an analytical evaluation is extremely hard. We have not found any such analysis in the previous literature. 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 workstations 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, $C_2 = 8000$. 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. It turned out that for all our experiment β=0.1 was already sufficient. (Note that β must 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.

(a)





(c)

vertex #	1	2	3	4	5	6	7	8	9	10
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
5						31	32	33	34	35
6							36	37	38	39
7								40	41	42
8									43	44
9										45

neuron # = edge #

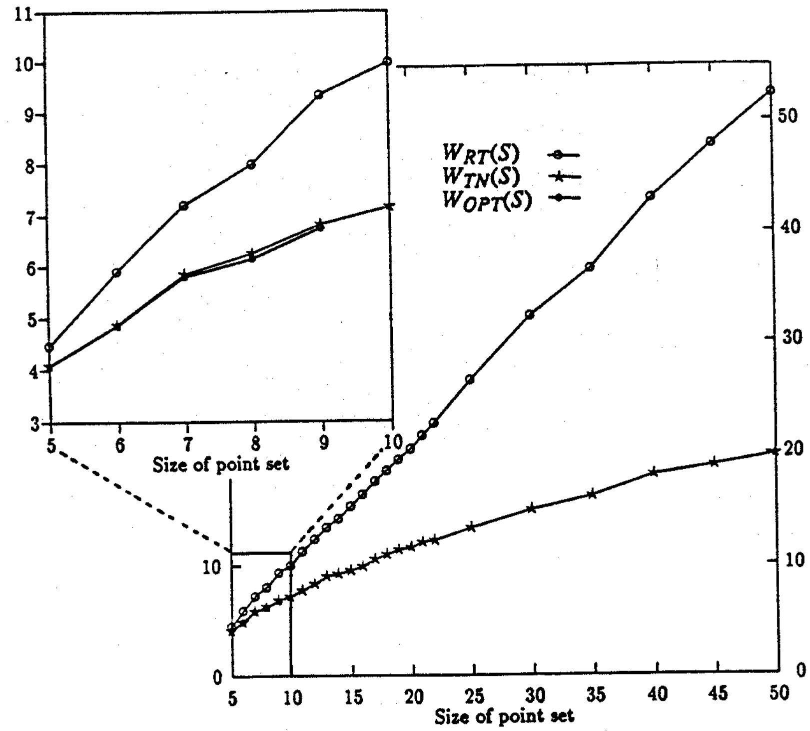
(edges of min. length triangulation highlighted)

(b)

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 The Final Stable State.

The 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 weight of the triangulation produced by our triangulation circuit (W_{TN}) with the weight of the actual minimum weight triangulation (W_{OPT}) and, in addition, with the weight of a random triangulation (W_{RT}). Our main problem here is that the actual minimum weight can only be determined for small problem sizes, because there is no known sequential polynomial time algorithm for determining the minimum weight triangulation. In Figure 3, column K_1 shows the number of samples tested for a given size of point sets (n). Column K_2 shows the average difference between W_{TN} and W_{OPT} in %, and column K_3 shows the variance of K_2 in %. Column K_4 shows the average difference between W_{TN} and W_{RT} in %. The W_{OPT},W_{TN}, W_{RT} values shown are of course averaged as well (variance ≈ 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 produced by our circuit is always less than 2% (variance <2.5%), and for $n \geq 10$ the weight produced by our circuit is at least 38% better than a random triangulation (with steadily increasing difference). The fact that for $n \geq 10$ the W_{TN} value increases only very slowly leads us to conjecture that it will stay very close to the optimum weight. Further testing (particularly for larger point sets) is in progress, but because of the problem indicated above very time consuming. An Intel iPSC/860 hypercube has been purchased (by Carleton University) and will we available in Jan. 1992. This will allow us to get exact minimum length triangulations for larger point sets, to compare them with the results reported by our circuit. Additional testing results will be incorporated in the final version of this paper.



n	K_1	W_{OPT}	W_{TN}	W_{RT}	K_2	K_3	K_4
5	25	4.09	4.10	4.48	0.27	1.37	9.69
6	25	4.85	4.87	5.91	0.55	1.63	21.64
7	25	5.81	5.86	7.22	0.97	2.29	24.08
8	25	6.15	6.26	8.01	1.88	2.47	28.39
9	25	6.76	6.83	9.35	1.05	1.40	37.16
10	50		7.18	10.00			39.52
11	50		7.74	11.29			46.10
12	50		8.31	12.38			49.26
13	50		9.01	13.47			49.68
14	50		9.22	14.22			54.34
15	50		9.51	15.31			61.02
16	50		9.91	16.38			65.25
17	50		10.54	17.57			66.70
18	50		10.96	18.53			68.86
19	50		11.38	19.51			71.51
20	50		11.60	20.45			76.41
21	50		12.07	21.66			79.63
22	50		12.24	22.75			85.84
25	50		13.38	26.62			99.06
30	10		14.96	32.33			116.44
35	5		16.19	36.61			126.31
40	5		18.05	43.00			137.86
45	5		18.92	47.78			152.81
50	2		19.86	52.41			163.95

Figure 3. The Triangulation Circuit compared to the Optimum and Random Triangulations.

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.

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 [9].) 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 [12]. It is interesting to observe that minimizing the number of rectangles is different from minimizing the total edge length [11].

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.

Approximate congruence of point sets: Consider the problem of determining whether two sets of points in d-dimensional Euclidean space are congruent via the geometric transformation of translation. From a practical point of view the problem of deciding exact congruence is not very realistic and is furthermore numerically ill-conditioned. Thus the notion of approximate congruence has been introduced [1]: Let A, B be two point sets. Find the smallest ε, and a 1-1 mapping $f{:}B \rightarrow A$ (called *labelling*), such that there exist a translation which moves every $\underline{b} \in B$ into the ε-neighborhood of its assigned point $f(\underline{b}) \in A$. The current sequential sequential time bounds for d=2 are $O(n^6 \log n)$ for the general problem, $O(n)$ when the labelling is known, and $O(n^6)$ when ε is known [1]. The following describes an analog circuit for the general problem and arbitrary d. The circuit consists of $N=n^2$ neurons, where each neuron i is is associated with a pair $(\underline{\mathbf{a}}_i,\underline{\mathbf{b}}_i)$ with $\underline{\mathbf{a}}_i \in A$ and $\underline{\mathbf{b}}_i \in B$. Select constants $\beta > 0$, $e>0$, $A>0$, and $B>0$ such that $A=2(e+\Delta\beta)$ and $e=2n\Delta\beta+nB$. Set every bias $e_i=e$ and the initial state as in (18) and (19), and define the T-matrix as

$$T_{ij} = -\,(1-d_{ij})\,(A\; share_{ij} + B\; value_{ij}) \text{ with } value_{ij} = \frac{|\,(\underline{\mathbf{a}}_i-\underline{\mathbf{b}}_i) - (\underline{\mathbf{a}}_j-\underline{\mathbf{b}}_j)\,|}{max_{kl}\{|\,(\underline{\mathbf{a}}_k-\underline{\mathbf{b}}_l) - (\underline{\mathbf{a}}_k-\underline{\mathbf{b}}_l)\,|\}} \text{ and}$$

$share_{ij}=1$ if $a_i=a_j$ or $b_i=b_j$, otherwise $share_{ij}=0$. We can show (but have to omit this proof here due to space limitations) that in the high-gain limit, the circuit creates a one-to-one mapping between A and B and that the energy approximates the smallest ε for a translation induced by that mapping (with some additive and pos. multiplicative constants). At this point in time, we have only tested the circuit for few random data samples, for which it performed well. Comprehensive testing is in progress, but very time consuming due to the high time complexity of the sequential algorithm. Comprehensive test data will be included in the full version of this paper.

As a final note, we outline another interesting solution, which does however use a more powerful analog circuit model because it includes time variant biases. The circuit consists again of $N=n^2$ neurons, where each neuron i is is associated with a pair $(\underline{\mathbf{a}}_i,\underline{\mathbf{b}}_i)$. The T-matrix is defined as $T_{ij} = -A\,(\delta_{a_i a_j}(1-\delta_{b_i b_j})+\delta_{b_i b_j}(1-\delta_{a_i a_j}))-B\,\delta_{a_i a_j}\delta_{b_i b_j}$ with a time variant bias $e_i(\underline{\mathbf{c}})=\tilde{e}-D(d_i/d_{max})$ where $d_i=|\,\underline{\mathbf{a}}_i-\underline{\mathbf{b}}_i+\underline{\mathbf{c}}\,|^2$ and $d_{max}=max_k\{|\,\underline{\mathbf{a}}_k-\underline{\mathbf{b}}_k+\underline{\mathbf{c}}\,|^2\}$ for a time variant vector $\underline{\mathbf{c}}$. The dynamics of $\underline{\mathbf{c}}$ are described by $\dot{\underline{\mathbf{c}}} = -2\alpha\,\Sigma_i\,[s_i\,(d_i/d_{max})\,(\underline{\mathbf{a}}_i-\underline{\mathbf{b}}_i+\underline{\mathbf{c}})]$ with $\underline{\mathbf{c}}=\underline{\mathbf{0}}$ in the initial state. The neurons' states s_i are initialized according to (18) and (19). Figure 4 shows an example run of the circuit with selected constants $A=400$, $B=700$, $\tilde{e}=750$, $D=600$, $\alpha=100$.

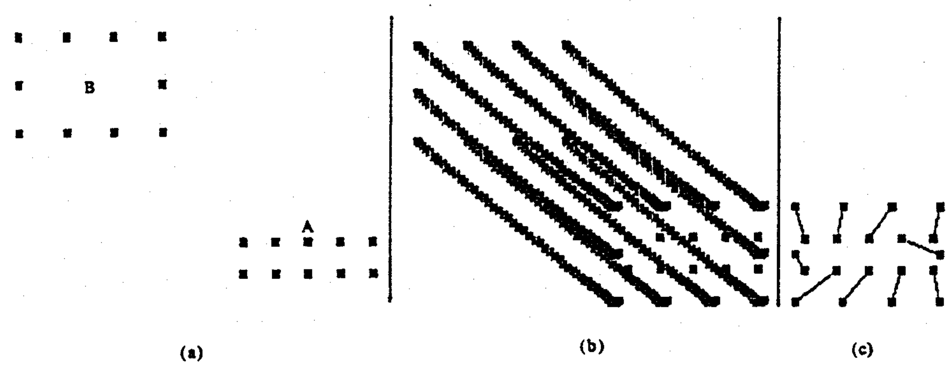


Figure 4. Analog Circuit with Time Variant Biases Converging to a Solution of an Approximate Congruence Problem. (a) The Two Point Sets and Their Locations. (b) Locations of Set B Corresponding to Intermediate States of the Circuit. (c) Finals Location and Point Assignment.

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] H. Alt, K. Mehlhorn, H. Wagner, and E. Welzl, "Congruence, similarity, and symmetries of geometric object," *Discrete and Computational Geometry*, Vol. 3, No. 3, 1988, pp. 237-256.

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

[3] 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.

[4] 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.

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

[6] 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.

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

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

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

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

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

School of Computer Science, Carleton University
Recent Technical Reports

SCS-TR-161 (revised)	**Recognizing Sources of Random Strings** R.S. Valiveti and B.J. Oommen, January 1990 Revised version of SCS-TR-161 "On the Data Analysis of Random Permutations and its Application to Source Recognition", published June 1989
SCS-TR-162	**An Adaptive Learning Solution to the Keyboard Optimization Problem** B.J. Oommen, R.S. Valiveti and J. Zgierski, October 1989
SCS-TR-163	**Finding a Central Link Segment of a Simple Polygon in O(N Log N) Time** L.G. Alexandrov, H.N. Djidjev, J.-R. Sack, October 1989
SCS-TR-164	**A Survey of Algorithms for Handling Permutation Groups** M.D. Atkinson, January 1990
SCS-TR-165	**Key Exchange Using Chebychev Polynomials** M.D. Atkinson and Vincenzo Acciaro, January 1990
SCS-TR-166	**Efficient Concurrency Control Protocols for B-tree Indexes** Ekow J. Otoo, January 1990
SCS-TR-167	**A Hierarchical Stochastic Automaton Solution to the Object Partitioning Problem** B.J. Oommen, January 1990
SCS-TR-168	**Adaptive List Organizing for Non-stationary Query Distributions. Part I: The Move-to-Front Rule** R.S. Valiveti and B.J. Oommen, January 1990
SCS-TR-169	**Trade-Offs in Non-Reversing Diameter** Hans L. Bodlaender, Gerard Tel and Nicola Santoro, February 1990
SCS-TR-170	**A Massively Parallel Knowledge-Base Server using a Hypercube Multiprocessor** Frank Dehne, Afonso Ferreira and Andrew Rau-Chaplin, April 1990
SCS-TR-171	**Parallel Processing of Quad Trees on the Hypercube (and PRAM)** Frank Dehne, Afonso Ferreira and Andrew Rau-Chaplin, April 1990
SCS-TR-172	**A Note on the Load Balancing Problem for Coarse Grained Hypercube Dictionary Machines** Frank Dehne and Michel Gastaldo, May 1990
SCS-TR-173	**Self-Organizing Doubly-Linked Lists** R.S. Valiveti and B.J. Oommen, May 1990
SCS-TR-174	**A Presortedness Metric for Ensembles of Data Sequences** R.S. Valiveti and B.J. Oommen, May 1990
SCS-TR-175	**Separation of Graphs of Bounded Genus** Ljudmil G. Aleksandrov and Hristo N. Djidjev, May 1990
SCS-TR-176	**Edge Separators of Planar and Outerplanar Graphs with Applications** Krzystof Diks, Hristo N. Djidjev, Ondrej Sykora and Imrich Vrto, May 1990
SCS-TR-177	**Representing Partial Orders by Polygons and Circles in the Plane** Jeffrey B. Sidney and Stuart J. Sidney, July 1990
SCS-TR-178	**Determining Stochastic Dependence for Normally Distributed Vectors Using the Chi-squared Metric** R.S. Valiveti and B.J. Oommen, July 1990
SCS-TR-179	**Parallel Algorithms for Determining K-width-Connectivity in Binary Images** Frank Dehne and Susanne E. Hambrusch, September 1990

SCS-TR-180 **A Workbench for Computational Geometry (WOCG)**
P. Epstein, A. Knight, J. May, T. Nguyen, and J.-R. Sack, September 1990

SCS-TR-181 **Adaptive Linear List Reorganization under a Generalized Query System**
R.S. Valiveti, B.J. Oommen and J.R. Zgierski, October 1990

SCS-TR-182 **Breaking Substitution Cyphers using Stochastic Automata**
B.J. Oommen and J.R. Zgierski, October 1990

SCS-TR-183 **A New Algorithm for Testing the Regularity of a Permutation Group**
V. Acciaro and M.D. Atkinson, November 1990

SCS-TR-184 **Generating Binary Trees at Random**
M.D. Atkinson and J.-R. Sack, December 1990

SCS-TR-185 **Uniform Generation of Combinatorial Objects in Parallel**
M.D. Atkinson and J.-R. Sack, January 1991

SCS-TR-186 **Reduced Constants for Simple Cycle Graph Separation**
Hristo N. Djidjev and Shankar M. Venkatesan, February 1991

SCS-TR-187 **Multisearch Techniques for Implementing Data Structures on a Mesh-Connected Computer**
Mikhail J. Atallah, Frank Dehne, Russ Miller, Andrew Rau-Chaplin, and Jyh-Jong Tsay, February 1991

SCS-TR-188 **Generating and Sorting Jordan Sequences**
Out of print
Alan Knight and Jörg-Rüdiger Sack, March 1991

SCS-TR-189 **Probabilistic Estimation of Damage from Fire Spread**
Charles C. Colbourn, Louis D. Nel, T.B. Boffey and D.F. Yates, April 1991

SCS-TR-190 **Coordinators: A Mechanism for Monitoring and Controlling Interactions Between Groups of Objects**
Wilf R. LaLonde, Paul White, and Kevin McGuire, April 1991

SCS-TR-191 **Towards Decomposable, Reusable Smalltalk Windows**
Kevin McGuire, Paul White, and Wilf R. LaLonde, April 1991

SCS-TR-192 **PARASOL: A Simulator for Distributed and/or Parallel Systems**
John E. Neilson, May 1991

SCS-TR-193 **Realizing a Spatial Topological Data Model in a Relational Database Management System**
Ekow J. Otoo and M.M. Allam, August 1991

SCS-TR-194 **String Editing with Substitution, Insertion, Deletion, Squashing and Expansion Operations**
B John Oommen, September 1991

SCS-TR-195 **The Expressiveness of Silence: Optimal Algorithms for Synchronous Communication of Information**
Una-May O'Reilly and Nicola Santoro, October 1991

SCS-TR-196 **Lights, Walls and Bricks**
J. Czyzowicz, E. Rivera-Campo, N. Santoro, J. Urrutia and J. Zaks, October 1991

SCS-TR-197 **A Brief Survey of Art Gallery Problems in Integer Lattice Systems**
Evangelos Kranakis and Michel Pocchiola, November 1991

SCS-TR-198 **On Reconfigurability of Systolic Arrays**
Amiya Nayak, Nicola Santoro, and Richard Tan, November 1991

SCS-TR-199 **Constrained Tree Editing**
B. John Oommen and William Lee, December 1991

SCS-TR-200 **Industry and Academic Links in Local Economic Development: A Tale of Two Cities**
Helen Lawton Smith and Michael Atkinson, January 1992