On the Complexity of MAX/MIN/AVRG Circuits

Manuel Blum Rachel Rue Ke Yang

March 29, 2002
CMU-CS-02-110

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We study the complexity of a class of circuits, namely, the MAX/MIN/AVRG circuits. On the wires of these circuits are real values between 0 and 1; the functions each gate performs are MAX, MIN, and AVERAGE of fan-in 2; there can be feed-backs in the circuit. It can be shown that every such circuit has at least a "stable" solution, meaning that there is a way to set each wire to a particular value such that each gate is satisfied. However, finding a stable solution in polynomial time seems to be a tricky problem and remains unsolved. We discuss some results concern this computation model, as well as its applications.

Partially supported by the NSF Aladdin center.

Keywords: circuits, complexity, NP, co-NP, two-person game

1 Introduction

We consider a family of circuits which differ from conventional boolean circuits in three ways: (1) We allow feed back; (2) instead of logic gates we have fan-in 2 MAX, MIN, and average gates; (3) we allow real values between 0 and 1 on all wires, although we assume that the inputs to the circuit are either 0 or 1. All circuits we consider will be in this family. This model was first introduced by Condon [C92, C93].

Without loss of generality, we may assume that our circuits also have gates setting their outputs to constants between 0 and 1.

Definition 1 *A gate of a circuit is* **satisfied** *if its output is correct, given its inputs. An* **assignment** *to a circuit is a mapping from the set of all the gates to the interval* $[0, 1]$*, s.t. each gate is assigned a real value. An assignment is* **stable** *if every gate is satisfied. A* **solution** *to a circuit, given fixed boolean inputs, is any stable assignment of values to its output wires. For any circuit* C*, let* X_C *be the set of output wires in* C*.*

Next, a (natural) definition of sub-circuits.

Definition 2 (Sub-circuits) *Given a circuit* C *with an assignment and a subset* S *of all the gates, the* **sub-circuit** *defined by* S *consists all the gates in* S *and all the wires within* S*. For wires whose outputs are in* S *and inputs not in* S*, we replace its input by a constant value.*

Now, the distance.

Definition 3 (Distance) *If we view each assignment as a vector in* R^n*, we define the* **distance** *between any two assignments to be the* L_∞ *norm, namely, the distance between two assignments is the maximum value difference between them at each gate.*

Domination and comparisons.

Definition 4 (Domination) *Given two assignments* A_1*,* A_2 *to the same circuit, and we represent them as vectors:* $A_1 = \langle a_1^1, a_1^2, ..., a_1 \rangle$ *and* $A_2 = \langle a_2^1, a_2^2, ..., a_2^n \rangle$*. We say* A_1 **dominates** A_2*, or* A_2 **is dominated by** A_1*, if* $a_1^i \geq a_2^i$ *for all* $i = 1, 2, ..., n$*, and we denote that by* $A_1 \geq A_2$*.*

Definition 5 (Monotone sequence) *Given a sequence of assignments* $A_1, A_2, ..., A_n$*, we say the sequence is* **monotone***, if* $A_1 \geq A_2 \geq ...A_n$*, or* $A_1 \leq A_2 \leq ... \leq A_n$*. We can be more specific by saying the sequence is* **increasing** *or* **decreasing***.*

2 How to Converge

A central question for the circuit is: does it have a solution? It turns out that the answer is always YES:

Theorem 1 *For any MMA-circuit, it always has at least one solution.* ∎

Then the next question is: how do we find it? A natural intuition is: if we somehow "run" the circuit, will it converge to a solution? Intuitively, suppose we give a circuit an arbitrary assignment, which might not be stable, if we let the circuit "run" for a while, it should be able to "converge" to a stable one, or a solution. This question is much more subtle than we first thought.

2.1 Synchronous Update

If we are not careful, things might get tricky: observe the following "flip-flop" type circuit. If we let both gate update simultaneously, the assignment might not converge.

So, "synchronous" transition is bad, and we want a way to run the circuit very asynchronously, so that it will bound to converge. The intuition is we want to change one gate at a time, and we want to do that in a pre-defined order.

2.2 Gate-by-gate update

If we update the circuits gate-by-gate, we might be able to converge if we are careful about the starting-point.

One easy observation is that if you start from **0**, or the all-zero assignment, simply running the circuit in any arbitrary order will lead to a solution: actually this is the *minimal solution*. Also if we start from all-one assignment, we will end up with the *maximal* solution:

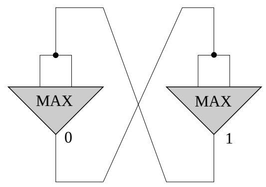

Figure 1: Bad case: circuit might not converge

Lemma 1 (Convergence for gate-by-gate update) *For any MMA-circuit, if we start the assignment with* **0** *(resp.* **1***),and update the gates (namely, change the output of a gate according to its current input) in any arbitrary order, the assignments we are getting converge to a solution, and the solution is the minimal (resp. maximal) solution for the circuit.*

Proof's sketch: Notice that all the gates (MAX, MIN, AVG) are *monotone* gates, and thus if we start from all-zero, no gate will decrease its output. Therefore when we are updating the circuit, we get monotonically increasing assignment, and since we have the apparent upper bound for the assignments, we know the assignments will converge to a solution — denote this solution by S_0.

To see that S_0 is the minimal one, think of the following experiment: we take two copies of the circuit, give the all-zero assignment to one copy — call it C_1, and give an arbitrary solution, S' to the other — call itC_2. Then we run the two circuits simultaneously in the same order. We know that C_1 will converge to the solution S_0, while C_2 will remain to be S' since all the gates are already satisfied. But again since the circuit is monotone, we know at any time the assignment at C_2 dominates the assignment at C_1 and therefore S' dominates S_0. This is true for any S' and thus S_0 is truly the minimal solution.

The same argument shows that the solution we get from the all-one assignment is the maximal solution. ■

2.3 Converging the hard way

We describe a complicated way that converges any original assignment to a solution.

Operation 1 (Converging an arbitrary assignment to MMA-circuit to a solution) *We describe how to operate on the circuit in the certain way to make sure it converges*[1]*.*

- **CONVENTION** *We arbitrarily order the gates by $G_1, G_2, ..., G_n$, and name their output values by $X_1, X_2, ..., X_n$.*
- **INPUT** *The input is an assignment $\langle a_1, a_2, ..., a_n \rangle$, where $0 \le a_i \le 1, \forall i = 1, 2, ..., n$, meaning $X_i = a_i, i = 1, 2, ...n$.*
- **OUTPUT** *After the operation, the circuit will stay in a solution $s_1, s_2, ..., s_n$. The solution is uniquely determined by the input assignment.*
- **ACTION** *The actual actions are hierarchical, and we will define them in terms of "level"s.*
  - **Level 1**
    The action of level 1 is very simple: given an arbitrary assignment, we fix the values at the outputs of gate $G_2, G_3, ..., G_n$, and change the output of gate G_1 according to its input.
  - **Level** i $(i = 2, 3, ..., n)$
    For action of level i, it is recursively defined as the following: we fix the outputs of gates $G_i, G_{i+2}, ..., G_n$, do actions of level $i-1$ continuously, until the sub-circuit consisting of $G_1, G_2, ..., G_{i-1}$ converges to a solution, assign the outputs of gates $G_1, ..., G_{i-1}$ to the solution, and then change the output of gate G_i according to its input.

  The operation is just continuously doing action of level n, until the assignment converges to a solution.

Intuitively, the operation we perform is: each time we run gate G_i for one step, we will let all gates $G_1, G_2, ..., G_{i-1}$ run for many (maybe infinitely) steps to make sure they converge, and then run G_i for one step, then $G_1, ..., G_{i-1}$ for many steps, and so on, and so to let G_i converge also.

The reason our operation is valid is guaranteed by the following lemma:

[1] We don't use the term "algorithm" here, since the operation is only mathematically (and physically) well-defined, and might take infinite amount of time to simulate on a digital computer.

Lemma 2 *At each level of action, the following properties are preserved:*

1. *The operation is "shrinking", namely, given any two assignments A_1 and A_2 for the same circuit, and then after one step of level i, suppose the values become B_1 and B_2, respectively. Then we have $\mathsf{dist}(A_1, A_2) \geq \mathsf{dist}(B_1, B_2)$.*

2. *The operation is monotone, i.e., if we perform a sequence of actions of level i, and get a sequence of assignments $A_1, A_2, ..., A_n, ...$, then the sequence is monotone.*

3. *At each level i, the sequence of actions will converge to s solution for the sub-circuit of gates $G_1, G_2, ..., G_i$.*

Proof: We prove lemma 2 by induction.

Base case: $i = 1$. Notice each individual gate (MAX, MIN, AVRG) is a "shrinking" function, i.e., for any x_1, x_2, y_1, y_2, we have

$$|g(x_1, y_1) - g(x_2, y_2)| \leq \mathsf{MAX}\{|x_1, x_2|, |y_1, y_2|\}$$

where g is MAX, MIN, or AVRG. For level 1, we only need one step, and then the sub-circuit (of only the gate G_1) is satisfied. So the operation is shrinking, monotone, and it converges trivially.

Inductive case: suppose we have proved the lemma for up to level $(i-1)$, now we look at level i. Since for actions of up to level i, the gates $G_{i+1}, ...G_n$ are all fixed, we only need to look at the sub-circuit of $G_1, G_2, ..., G_i$. Suppose we start with an arbitrary, we first converge the gates $G_1, G_2, ..., G_{i-1}$ to a satisfied assignment — this is guaranteed to succeed by the induction hypothesis. Say now the assignment is $A = \langle a_1, a_2, ..., a_i \rangle$. Then we change the output of G_i according to its input: say we change it from a_i to b_i. Then we let gates $G_1, ..., G_{i-1}$ run again to converge to solution $b_1, b_2, ..., b_{i-1}$.

Here is the observation: in the process, we are changing one gate at a time, and thus it is easy to see the "shrinking" property is preserved. A bit formal: given two different original assignments, if we perform actions of level $(i-1)$, then by induction hypothesis, their distance will not increase. Notice we might do level $(i-1)$ for infinitely many times, but the point is when both converges to a solution, then the distance between the two solutions is still bounded by the distance of the original assignments. So however many steps, and however many levels of actions we perform, the operation is always a "shrinking" one.

Next is to show the monotonicity. WLOG we assume that $b_i > a_i$, i.e., the output of G_i increases in this step. Then gates $G_1, G_2, ..., G_{i_1}$ start to change *only* because of the change of G_i (in other words, if G_i doesn't change, $G_1, G_2, ..., G_{i-1}$ wouldn't change either, since they are already stable). Then observe all the gates are monotone gates, so any gate that takes in G_i as input will also increase since G_i increased, and any other gate will also increase (or stay put). Since in this step, G_i is the only source that makes the change[2], the overall changes for all gates are all non-negative. In other words, the new solution for gates $G_1, ..., G_{i-1}$, $b_1, b_2, ..., b_{i-1}$, will dominate the old one, namely, $a_1, a_2, ..., a_{i-1}$. Thus overall, the assignment after the step dominates the assignment before the step. Then again by the monotonicity of each gate, the next step will bring the assignment to an even higher position, and the assignment sequence is monotone.

Now the converging is obvious: since we have an monotone assignment sequence, and the assignment is clearly bounded (the value of any gate is always between 0 and 1), it must converge.

■

Now it is easy to see the solution we are converging to is indeed unique and well-defined.

2.4 Time-average convergence

We discuss another way to converge the circuit from ANY assignment to a solution. The idea came from Avrim Blum and a large part of the proof comes from Adam Kalai.

The operation is very simple: we do synchronous update, i.e., we update all the gates simultaneously, but the trick is: we don't update the gates "all the way to the new value", rather we take a convex combination of the old value and the new value. To be more precise, we can use a function $f : [0, 1]^n \to [0, 1]^n$ to denote the mapping that "update the output of each gate according to its input", then our update scheme is:

$$g(A_n) := \epsilon A_{n-1} + (1 - \epsilon) f(A_{n-1})$$

We will always use $f(\cdot)$ and $g(\cdot)$ to denote the "direct update" operation and the time-average operation, respectively.

Notice that when ϵ is a rational, the convergence can be simulated by a MMA-circuit that has some more AVG gates attached to the original gates. In this case, we call the original gates the *primary* gates and the added AVG gate the *secondary* gates. Figure 2 shows an example for the case $\epsilon = 1/2$. Also notice the new circuit and the old circuit have *exactly* the same solution set.

[2] This is actually the core property of our operation that guarantees the convergence — the reason the flip-flop example fails is exactly some gates changes simultaneously, and some goes up, and some goes down.

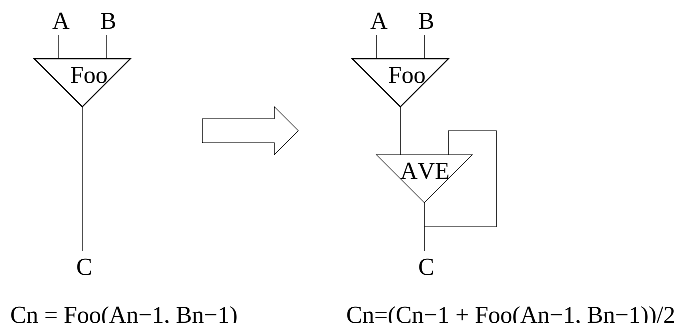

Figure 2: Time-average convergence: case $\epsilon = 1/2$

We show that this update scheme will guarantee to converge regardless where is starts from. We first make some observations.

Observation 1 *The operation $g(\cdot)$ is a shrinking, monotone, and continuous mapping.*

Proof's sketch: Since $g(X) = \epsilon \cdot X + (1-\epsilon) \cdot f(X)$, and $f(\cdot)$ is shrinking, monotone, and continuous. ■

For this subsection, we will use X_0 to denote an arbitrary assignment and define

$$X_n = g(X_{n-1}, \ n > 0$$

namely, X_n are the assignments we get after applying g repeated. We will show that $\{X_n\}$ converges to a solution.

Observation 2 *For all $n \geq 1$,*

$$|X_{n+1} - X_n| \leq |X_n - X_{n-1}|$$

Proof's sketch: Notice g is shrinking. ■

Now we will give a proof that $\{X_n\}$ converges.

Notice that if the MMA circuit doesn't have any MAX/MIN gates, things are much easier: we are only left with AVG gates, and then both f and g are non-negative linear functions. We love linear functions. However, f and g not far from linear functions — they are *piece-wise linear.*

Observation 3 *Suppose an MMA-circuit contains k MAX/MIN gates, then we can partition the whole space $[0,1]^n$ into 2^k polytopes, such that f when restricted on each of these polytopes is a linear function.*

Proof's sketch: We can view each MAX/MIN gate as a LEFT/RIGHT gate, according to their input values. Notice that LEFT and RIGHT are both linear functions. There are totally k MAX/MIN gates, and therefore totally 2^k possible LEFT/RIGHT settings of these gates, which correspond to 2^k different linear functions. Also notice each setting gives a set of linear constrains (e.g. if $A = \mathsf{MAX}(B, C)$ and this MAX gate is set to LEFT, then the constraint is $B \geq C$). We include the constant gates 0 and 1 in our circuit so as to make sure the function is indeed a linear function, instead of an affine one. ■

Then we can write down all these 2^k linear functions as $f_1, f_2, ..., f_{2^k}$ and then we know that

$$\forall X \in [0,1]^n, \ \exists i \in \{1, 2, ..., 2^k\}, \ s.t. \ f(X) = f_i(X)$$

We can write down the matrices for each of these linear functions: we use F_i to denote the matrix for the i-th linear function f_i. Then we have:

Observation 4 *For each $i \in \{1, 2, ..., 2^k\}$, the matrix F_i satisfies:*

1. *Each entry of F_i is non-negative.*
2. *Each row of F_i adds up to 1.*

Then notice that we can express the G_i's in terms of F_i's:

$$G_i = \epsilon I + (1-\epsilon) F_i$$

and thus we have

Observation 5 *For each $i \in \{1, 2, ..., 2^k\}$, the matrix G_i satisfies:*

1. *Each entry of G_i is non-negative.*
2. *Each row of G_i adds up to 1.*
3. *Each entry on the diagonal of G_i is at least ϵ, and if it is not 1, it is at most $(1+\epsilon)/2$.*

Since when we apply g, we are applying the same linear function on each polytope — we use P_i to denote the i-th polytope. Then we can ask what happens to a polytope as a whole under the function g? Notice it is a linear function and thus the result is another polytope.

Then we will prove this nice lemma:

Lemma 3 *For each polytope P_i under the operation g, either one of the following events is true:*

1. *P_i remains unchanged, in which case all the points in P_i are solutions.*
2. *P_i is mapped to a new polytope Q_i, s.t.*
$$||Q_i|| \leq \delta \cdot ||P_i||$$
for some constant δ. Where we use $||X||$ to denote the measure of polytope X.

Proof: We start by stating some facts about determinants.

Fact 1 *Suppose a polytope P is mapped to another polytope Q under the matrix M, then*

$$||Q|| = |\det(M)| \cdot ||P||$$

■

Fact 2 *A matrix A has determinant at most 1, if for each row, the absolute values of the entries add up to be at most 1. Moreover, if there is a row whose entries' absolute values add up to be $1-\epsilon$, then the determinant of A is at most $1-\epsilon$. More formally (and more concisely), we have*

$$\det(A) \leq \min \left\{ \sum_{j=1}^{n} |a_{ij}| \right\}_i$$

Proof's sketch: By induction on n, break the determinant down into sums of smaller determinants. ■

Now we look at the polytope $\bar{P}$ and the matrix $\bar{G}$ that operates on $\bar{P}$. If $\bar{G}$ is the identity matrix, then apparently $\bar{P}$ remains unchanged under $\bar{G}$ and thus it remains under the operation g, which means all points in $\bar{P}$ is a solution — this is the case 1 in the lemma.

If $\bar{G}$ is not the identity matrix, then we use A_{ij} to denote the (i,j)-th entry of $\bar{G}$. WLOG we can assume that

$$0 < a_{11} < 1$$

Then from the observation, we know that $a_{11} \leq (1+\epsilon)/2$.

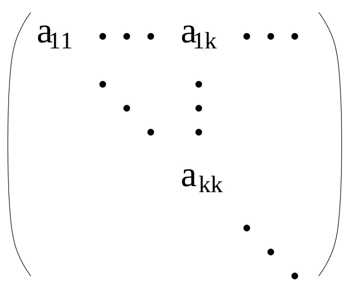

Figure 3: Determinants

Then we can compute the determinant of $\bar{G}$ by expanding it in the first row, and the crucial observation is that a_{kk} is guaranteed to be at least ϵ for each k, and thus for the determinant we get by removing the 1st row and the k-th column, it has determinant at most $1 - \epsilon$, since a_{kk} is removed from the sub-matrix. Now we have

$$\begin{aligned}
\det(\bar{G}) &= \sum_{i=1}^{n} \pm a_{1i} \cdot \det(G_{1i}) \\
&\leq a_{11} \cdot \det(G_{11}) + \sum_{i=2}^{n} a_{1i} \cdot \det(G_{1i}) \\
&\leq a_{11} + \sum_{i=2}^{n} a_{1i} \cdot (1-\epsilon) \\
&= a_{11} + (1 - a_{11})(1-\epsilon) \\
&= 1 - (1 - a_{11}) \cdot \epsilon \\
&= 1 - (1-\epsilon)\epsilon/2
\end{aligned}$$

and $1 - a_{1k} \cdot \epsilon$ is a constant also. So in either way, when $\bar{G}$ is not identity, its determinant is strictly smaller than 1 — which translates to P_i's measure gets reduced by a constant factor. This is the case 2 in the lemma. ■

Intuitively, what the lemma says is: the polytope that's not the solution will keep strictly shrinking. One immediate results from the lemma is:

Corollary 1 *We call all an assignment X "bad", if $g^n(X)$ doesn't converge as $n \to \infty$. Then the set of bad assignments has measure 0.*

Proof: Notice if X got mapped to itself, then it is a solution. So each bad point is in a shrinking polytope. ■

But we really something stronger: we want to say the bad set is the empty — a set of measure zero might be bad, since all the points we can operate on, namely all the rational points, have measure zero.

It turns out it is not hard to amplify the Corollary 1 to the following theorem:

Theorem 2 *From any assignment X, $g^n(X)$ converges as $n \to \infty$.*

Proof: Suppose to the contrary, there exists an X_0, such that $\{X_n = g^n(X_0)\}$ doesn't converge. But we know some sub-sequence of $\{X_n\}$ converges — say there is a subsequence $\{X_{i_n}\}$ converging to A. Take an $\epsilon > 0$ and look at all the X_n's that's at least ϵ away from A — if there are only finite left, we pick a smaller ϵ. Since $\{X_n\}$ doesn't converge, there exists an ϵ such that there are infinitely many assignments in $\{X_n\}$ that are ϵ away from A. There these points have a subsequence that converges to another point B. We know that $||A - B|| > \epsilon$. We also know that for any $\delta > 0$, there exists N_1 and N_2, such that $||X_{N_1} - A|| \leq \delta$ and $||X_{N_2} - B|| \leq \delta$.

Now since all points that don't converge have measure zero, we know there exists a Y which converges and $||X - Y|| \leq \epsilon/5$ — otherwise the ball centered at X with radius $\epsilon/5$ contains all the bad points: that's positive measure. We use $\{Y_n\}$ to denote the sequence of points when applying g. Now $\{Y_n\}$ converges, and thus there exists an N, such that for all $m > n > N$, $||Y_n - Y_m|| \leq \epsilon/5$.

Then take $M_1 > N$ and $M_2 > N$, such that $||X_{M_1} - A|| \leq \epsilon/5$ and $||X_{M_2} - B|| \leq \epsilon/5$. Then we have

$$\begin{aligned}
\epsilon &< ||A - B|| \\
&\leq ||A - X_{M_1}|| + ||X_{M_1} - Y_{M_1}|| + ||Y_{M_1} - Y_{M_2}|| + ||Y_{M_2} - X_{M_2}|| + ||X_{M_2} - B|| \\
&\leq \epsilon/5 + \epsilon/5 + \epsilon/5 + \epsilon/5 + \epsilon/5 \\
&= \epsilon
\end{aligned}$$

Contradiction.

■

Actually this gives another way to converge the circuit, and this also gives a continuous mapping that's as good as the one we got in the last section.

3 Topology of the Solution Space

We discuss the topological properties of the set of the solutions to a circuit.

3.1 Continuous Mapping

Now we have a way to start from any initial assignment, and converges to a unique solution. We can view this as a mapping, from the space of initial assignments, i.e., $\{0,1\}^n$, to the space of solutions – call it Σ. We write this as

$$\phi: \{0,1\}^n \mapsto \Sigma$$

Observation: it is a "shrinking" mapping, namely,

$$\mathsf{dist}(\phi(A), \phi(B)) \leq \mathsf{dist}(A, B)$$

for any $A, B \in \{0,1\}^n$. It is a monotone mapping, namely, if $A \leq B$, then $\phi(A) \leq \phi(B)$, for any $A, B \in \{0,1\}^n$. It derives directly from the lemma in last section.

We use $\mathbf{0}$ and $\mathbf{1}$ to denotes the points $\langle 0, 0, ..., 0\rangle$ and $\langle 1, 1, ..., 1\rangle$. Remember Σ is a lattice[3], and it has a minimal solution and a maximal one. Use **MIN** and **MAX** to denote them. We have the following observations:

Lemma 4 *For any $s \in \Sigma$, $\phi(s) = s$.*

It is intuitively obvious, since the converging won't change any satisfied gate.

Lemma 5 $\phi(\mathbf{0}) = \mathbf{MIN}$ *and* $\phi(\mathbf{1}) = \mathbf{MAX}$

Proof's sketch: Notice **MIN** dominates $\mathbf{0}$, we have $\phi(\mathbf{0}) \leq \phi(\mathbf{MIN}) = \mathbf{MIN}$. But **MIN** is the minimal point in Σ, and thus $\mathbf{MIN} \leq \phi(\mathbf{0})$. Therefore we have $\phi(\mathbf{0}) = \mathbf{MIN}$.

The same proof goes for $\phi(\mathbf{1}) = \mathbf{MAX}$. ■

The next one is important:

Lemma 6 *ϕ is continuous*[4].

Proof: The shrinking property of ϕ already guarantees the continuity: take any $\epsilon > 0$ small enough and any $P, Q \in \{0,1\}^n$, $\mathsf{dist}(P, Q) < \epsilon$, we have $\mathsf{dist}(\phi(P), \phi(Q)) \leq \mathsf{dist}(P, Q) < \epsilon$. That means: any neighbor of point P will be mapped into the neighbor of $\phi(P)$. ■

3.2 Path-connectness and One-Or-Infinite Theorem

Now, ϕ is a continuous and onto mapping from $\{0,1\}^n$ to Σ, and by the well-known fact (say, page 265 of [C66], or page 16 of [J84]), we know Σ is path-connected, since $\{0,1\}^n$ is path-connected. Intuitively, that means if you arbitrarily pick two points P, Q from Σ, then there is a continuous path that connected P and Q, and the path is completely in Σ.

So, we have

Theorem 3 *Σ is path-connected.*

This turns out to be pretty powerful, since one trivial corollary is:

Corollary 2 (One-Or-Infinite Theorem) *If Σ contains more than one point, then it contains uncountably many.*

Proof: If Σ contains two distinct points P and Q, then there is a path connecting them, and all points on the path are in Σ — this is a continuum. ■

4 Much Ado About Rationals

In this section we are interested in rational values, and especially the "simple" rational numbers — those with small denominators.

[3] This is from Manuel.

[4] Pedantically, or more strongly, it is uniformly continuous.

4.1 Complexity of Circuit vs. Complexity of Rational

Notice that given a circuit, both its max and min solutions are rationals, and from [C92], we know all values are rationals with denominator less than or equal to 4^n.

Now we prove a converse result: for any rational number p/q, where $0 < p < q < 2^n$, we are build a circuit using $O(n)$ gates, and the output of one gate is p/q. Actually we will just play with a little algebra and we only need AVG gates.

First, let's prove some lemmas:

Lemma 7 *For any integer n and integer x, $0 < x < 2^n$, one can compute $x/2^n$ using no more than n gates.*

Proof: By induction on n. The base case is $n = 1$, trivial.

Now if the lemma is true for $n-1$, then we look at the case for n. If $x < 2^{n-1}$, then $x/2^n = \mathsf{AVG}[x/2^{n-1}, 0]$, otherwise $x/2^n = \mathsf{AVG}[(x - 2^{n-1})/2^{n-1}, 1]$. ∎

Okay, given one can compute $x/2^n$, we can actually do $x/2^n \cdot A$, given any input A.

Next we look at the linear combination of 1 and A:

Lemma 8 *For any integer n and integer $x > 0, y > 0$, $x + y < 2^n$, one can compute $(x + y \cdot A)/2^n$ using no more than $2n$ gates, when given the input A.*

Proof: Again by induction, base case ($n = 1$) is trivial.

Now if the lemma is true for $n - 1$, we look at the case for n.

- If $x > 2^{n-1}$, then $(x + y \cdot A)/2^n = \mathsf{AVG}[1, ((x - 2^{n-1}) + y \cdot A)/2^{n-1}]$.
- If $y > 2^{n-1}$, then $(x + y \cdot A)/2^n = \mathsf{AVG}[(x + (y - 2^{n-1}) \cdot a)/2^{n-1}, y]$.
- Otherwise both x and y are less than 2^{n-1} and thus $(x + y \cdot A)/2^n = \mathsf{AVG}[x/2^{n-1}, y \cdot A/2^{n-1}]$. by Lemma 7, we can compute each part using $n - 1$ gates. So that's $2n - 1$ gates in total.

∎

Now we are ready to prove the main thing:

Theorem 4 *For any integer n, and rational p/q, where $0 < p < q < 2^n$, we can construct a circuit with $O(n)$ gates, which have a unique solution. In the unique solution, one of the outputs of the circuit is p/q.*

Proof: WLOG assume $q > 2^{n-1}$, since otherwise we can find a smaller n. Let $X = p/q$ and $k = 2^n - q$. Then notice x is the only solution to the linear equation

$$k \cdot X + p = 2^n \cdot X$$

or

$$\frac{k \cdot X + p}{2^n} = X$$

But notice $k+p = 2^n - q + p < 2^n$, and thus by Lemma 8, given X, one can construct a circuit that outputs $(k \cdot X + p)/2^n$. Then we feed this output back to X, and we are done. The number of gates is still bounded by $2n$.

Here is a figure illustrating the construction.

∎

So we can relate the complexity of rational numbers (defined as number of bit required to describe such a number) to the complexity of the circuit: basically, all the values on the gates of a size n circuit are of complexity $O(n)$, and all rational numbers between 0 and 1, and of complexity n, can be contracted by a circuit of size $O(n)$.

Next, we look at a more complicated theorem:

Theorem 5 *Suppose we have a circuit, C, of size n, and a rational number $x = p/q$, where $1 < p < q < 2^m$. If there is a solution to the circuit (not necessarily the minimal/maximal solution), and one gate g has value x, then there must exist a solution where g still has the value x, and all other gates have rational values of complexity $O(m + n)$.*

This is actually saying the circuit is "inherently rational" — even when we fix a gate with a specified value, we can still have a "reasonably simple" rational solution, whenever there is a solution.

Proof's sketch: The proof is actually easy: notice if we fix each of the MAX and MIN gates to one direction (namely, let each MAX or MIN gate output of specified inputs), we get a circuit consists of only AVG gates. But then this is a linear system, and if it has a solution, then it must have a rational solution.

Since we know the circuit has a solution with the value of g be x, we just fix the MAX and MIN gates as in the solution. Now we get a linear system that has a solution even if we add one more constraint: $g = x$. Since it has a solution, it must has a rational solution, and each number in this rational solution has complexity at most $m + 2n$. ∎

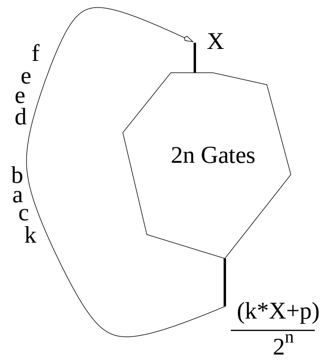

Figure 4: Building a n-bit rational using $O(n)$ gates

5 The Reconfiguration Lemma

We prove a lemma, namely the Reconfiguration Lemma in this section. We feel that this lemma is very useful in proving things about the circuit and it provides a lot of insights to MMA circuits.

Definition 6 (Reconfiguration) *Let g be a gate in an mma circuit C. We may* reconfigure *g by*

- *changing the inputs to g;*
- *changing the function g computes. In particular, we allow the output of g to be set to any constant in the interval* $[0,1]$.

Let C be an MMA circuit; let $R = \{g_1, \ldots, g_k\}$ be a subset of the gates in C; let C^* be a circuit obtained from C by reconfiguring the gates in R. Suppose that the unique minimum solutions of C and $C*$ are S and S^*, respectively. Define $\Delta_{C,C^*} := \mathsf{MAX}_{g \in R}\{|g_S - g_{S^*}|\}$.

Lemma 9 (Reconfiguration Lemma) *Let C be an MMA circuit; let C^* be a circuit obtained from C by reconfiguring a subset R of the gates in C; Assume C and C^* have minimum solutions S, S^*, and let $m \in R$ be such that $|m_S - m_{S^*}| = \Delta_{C,C^*}$. Let G be the set of all gates in C. Then* $\mathsf{MAX}_{g \in G}\{|g_S = g_{S^*}|\} = \Delta_{C,C^*}$.

Proof: Let C^{**} be the circuit obtained from C by reconfiguring each gate $g \in R$ by fixing its ouputs to the constant g_{S_*}. Then S^* is also a solution to C^**. We assume without loss of generality that $C^* = C^{**}$.

Define starting assignments C_{start} and C^*start as follows:

- For all $g \in G \setminus R$, $g_{C_{\text{start}}} = 0$.
- For all $g \in G \setminus R$, $g_{C^*_{\text{start}}} = 0$.
- For all $g \in R$, $g_{C_{\text{start}}} = g_S$.
- For all $g \in R$, $g_{C^*_{\text{start}}} = g_{S^*}$.

Imagine the following experiment: Start C and C^* in their starting assignments C_{start} and C^*start, respectively. Now run the circuits in tandem, in each time step updating the same individual gate in both circuits. Let the value of gate g in time step i be g_i in C and g_i^* in C^*. The sequence $< g_n >$ converges to g_S; $< g_n^* >$ converges to g_S^*. The sequence $< |g_n - g_n^*| >$ converges to Δ_{C,C^*}.

Base case: for $g \in R$, $|g_0^* - g_0| \leq \Delta$ by definition. For $g \in G \setminus R$, $|g_0^* - g_0| = 0 \leq \Delta$. Inductive step: let $z \in G$ compute the function $f(x,y)$, where x, y are the inputs to z, and f is a *max*, *min*, or *average* function.

Claim: For any $p, q \in \Re$, $|f(x,y) - f(x+p, y+q)| \leq \mathsf{MAX}\{|p|, |q|\}$. Suppose that z is the gate to be updated in step $i+1$. By hypothesis, after step i, $\forall g \in G, |g_i^* - g_i| \leq \Delta$. By the Claim and hypothesis, $|f(a_i, b_i) - f(a_i^*, b_i^*)| \leq \Delta$.

Thus we have

$$|z_{i+1} - z_{i+1}^*| = |f(a_i, b_i) - f(a_i^*, b_i^*)| \leq \Delta.$$

We have shown that for all n, $|z_n - z_n^*| \leq \Delta$. In addition, $\forall z \in G, \lim_{n\to\infty} |z_n - z_n^*| = |z_S - Z_{S^*}$. So $|z_S - z_{S^*}| \leq \Delta$.

■

Next, we show how one can use this lemma to prove that the stable circuit problem is in NP $\cap$ co-NP.

Let C be an MMA circuit; let S be a solution to C; let M be the minimum solution to C. Let MAX be the set of max gates in C; let MIN and AVG be the sets of min and average gates, respectively. Let LP be the linear program

$$\mathsf{MIN} \sum_{i \in MAX} i$$

subject to

$$\begin{aligned} x &\geq a \text{for all } x \in MAX, a \text{ an input to } x \\ x &= \frac{a+b}{2} \text{for all } x \in AVG, a, b \text{ inputs to } x \end{aligned}$$

We use the phrase "fix min gate x left or right according to S" to mean, "if in S, x is equal to its left input, reconfigure x by making it equal to its left input, and similarly set X to its right input if in S x is equal to its right input. Let C^* be the circuit obtained from C by fixing the output of each min gate according to S.

Claim 1 *If $S = M$, then LP run on C^* will return S.*

Proof: Let M^* be the minimum solution to C^*. LP returns M^*. S is a solution to C^*, so $M^* \leq S$. Suppose $M = S \neq M^*$. Then M^* cannot be a solution to C. The constraints of LP guarantee that all max and average gates are satisfied, so any unsatisfied gates must be min gates fixed to the higher of their two inputs. But in that case, all unsatisfied gates want to go down, and by monotonicity, there is a solution $M'' < M^* < M$. Contradiction. ■

Claim 2 *If $S \neq M$, then LP run on C^* may return S. In that case, there must be a min gate x with inputs a, b such that $x_S - x_M = \Delta_{C,C^*}$ and $S(a) = S(b)$.*

Proof: Since S is a solution to C^*, every min gate y is less than or equal to both of its inputs in S. Suppose that y is a min gate with an input w such that $y_S < w_S$, and $y_M = w_M$. (There must exist such a y; otherwise all min gates would be set as in M, and LP would have returned M.) Then

$$\Delta \geq w_S - w_M > y_S - y_M$$

The Claim follows, by the Reconfiguration Lemma. ■

Tweaking

Let D be any set of min gates known to contain the set of min gates $\{x : |x_S - x_M| = \Delta\}$. The algorithm TWEAK applied to the pair (C^*, D) does two things:

1. Reconfigures every gate $x \in D$ by fixing x to the constant $x_S - \epsilon$, for some $\epsilon < 4^{-n-1}$;
2. Uses LP to find the minimum solution for the reconfigured circuit.

Let C^T and S^T be the circuit and minimum solution obtained by applying TWEAK to (C^*, D).

Claim 3 *If $S = M$, then for some $x \in D$ with inputs a, b, $x_{S^T} < \mathsf{MIN}\{a_{S^T}, b_{S^T}\}$.*

Proof: By the Reconfiguration Lemma, $y_{S^*} - y_{S^T} \leq \epsilon$ for every gate y. It follows that for every tweaked gate $x \in D$ with inputs a, b, $x_{S^T} \leq \mathsf{MIN}\{a_{S^T}, b_{S^T}\}$. Suppose this were satisfied at equality for every $x \in D$. Then every $x \in D$ would be satisfied by S^T. By monotonicity, any unsatisfied gate in C would be too high; again by monotonicity, it follows that there must be a solution to C lower than $S = M$. Contradiction. ■

Claim 4 *if $S \neq M$, then for any x such that $x_S - x_M = \Delta$, $x_{S^T} \geq \mathsf{MIN}\{a_{S^T}, b_{S^T}\}$, where a, b are the inputs to x.*

Proof: By definition, $x_M = \mathsf{MIN}\{a_M, b_M\}$. By the Reconfiguration Lemma, $a_{S^T} \leq a_M + \Delta$ and $b_{S^T} \leq b_M + \Delta$. The Claim follows. ■

The Claims above establish that the following algorithm correctly decides whether or not $S = M$.

Iterated Tweaking

1. Run LP on C^*. If $S^* \neq S$, halt and output $S \neq M$.
2. Set $D := \{x_i \in MIN : x_{i\,S^*} = a_{i\,S^*} = b_{i\,S^*}$, where a_i, b_i are inputs to x_i.

3. apply TWEAK to (C^*, D).
   - If $\forall x \in D x_{ST} \geq \mathsf{MIN}\{a_{ST}, b_{ST}\}$, halt and output $S \neq M$.
   - If $\forall x \in D x_{ST} < \mathsf{MIN}\{a_{ST}, b_{ST}\}$, halt and output $S = M$.
4. Set $D := D \setminus \{x \in D : x_{ST} < \mathsf{MIN}\{a_{ST}, b_{ST}\}$. Go to Step 3.

Analysis. By Claims 1, 3, and 4, the output of Iterated Tweaking is always correct. Since D shrinks by at least one element in each iteration, the algorithm must halt after calling the TWEAK subroutine at most $|MIN|$ times. So iterated tweaking decides whether $S = M$ after solving $O(n)$ linear programs, each of size linear in n.

Corollary 3 *Fix a canonical way of expressing a (circuit, solution) pair in binary. Given a circuit C whose minimum (possibly unknown) solution is M, the decision problem, "Is the first bit of the canonical description of (C, M) a 1?" is in NP$\cap$co-NP.*

Proof: Guess M. Use Iterated Tweaking to prove that M is the minimum solution to C. Look at the first bit of (C, M) and decide whether or not it is a 1. ■

6 Multiplication

This section is to investigate the computing power of the MAX/MIN/AVG circuit. We will show 3 results on doing multiplication — two negative and one positive, all concerning different models of doing multiplication.

There are two ways to look at the multiplication: one is the traditional digital way: we encode each number in binary (or wherever), and do the multiplication in binary, like $101 \times 11 = 1111$ — this is the computer science way; the other is the analogous (probably more traditional) way: take two real numbers a and y, and output their product xy — notice our circuit can have real numbers on the wires, so it make sense to look at multiplication in this way. Sadly we will see neither way would work here.

6.1 Binary Multiplication

Let's first define the binary multiplication model:

Definition 7 (Binary Multiplication Circuit) *a* **Binary Multiplication Circuit of size** *n is a MAX/MIN/AVG circuit with $4n$ special wires, denoted by $A_0, A_1, ..., A_{n-1}$, $B_0, B_1, ..., B_{n-1}$, and $C_0, C_1, ..., C_{2n-1}$, along with 2 distinct real numbers $0 \leq a < b \leq 1$ — think of a and b as encodings of 0 and 1. If we set each A_i and each B_i to be either a or b, then the circuit has a unique solution with each C_i also set to be either a or b. What's more, then we read these wires as binary encodings of integers, and get two n-bit numbers $A := \overline{A_{n-1} \cdots A_1 A_0}, B := \overline{B_{n-1} \cdots B_1 B_0}$, and a $2n$-bit number $C := \overline{C_{2n-1} \cdots C_1 C_0}$. We should have $C = A \cdot B$.*

The figure below is an illustration.

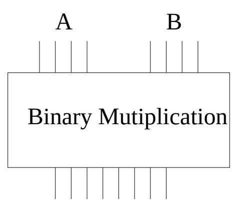

Figure 5: A binary multiplication circuit

Unfortunately, we have the following negative result:

Theorem 6 *Binary multiplication circuit of size $n \geq 2$ doesn't exist.*

Proof: Assume to the contrary there exists such a circuit with m gates. Set wires $A_0 = A_1 = B_0 = b$, and the rest A_i and B_i to be a. Then we are computing $11 \times 01 = 0011$, and we should have $C_0 = C_1 = b$. Denote the current setting by P. Since P is a solution, $\phi(P) = P$.

Next we change wire B_1 from a to b — denote the new setting by Q, and let the circuit converge to another solution – denoted by $\phi(Q)$.

Notice now we are computing $11 \times 11 = 1001$, and we should have $C_0 = C_3 = b$ and $C_1 = C_2 = a$ in $\phi(Q)$.

Notice we raise the value of wire B_1 from a to b to obtain Q from P, and thus Q dominates P. By the monotonicity of ϕ, we have $\phi(Q) \geq \phi(P) = P$. But in P, the wire $C_1 = b$, while in $\phi(Q)$, the wire $C_1 = a < b$. This is a contradiction. ■

Notice we have proved a much more general statement: in general, any boolean function that is not bit-wise monotone, cannot by computed by the MAX/MIN/AVG circuit. Multiplication is a monotone function in general, but not monotone in each bit in binary representation.

It also should be noticed that we are pretty strict in defining the binary multiplication circuit. There are many ways we can loosen the definition, for example: the length of the two inputs need not to be the same, the length of the output need not to be twice of the input length; we can use different encodings for different wires; we can even compute multiplication in Z_n for some integer n instead of multiplication in Z... But in general, it is always impossible to do non-trivial binary multiplication in the MAX/MIN/AVG circuit.

6.2 Real-valued Multiplication

The impossibility of the binary multiplication motivates us to look at the alternative: since each wire can have real values, is that possible that we construct a circuit that directly computes multiplication in real values? Multiplication, as a whole, is a monotone function anyway.

Let's be precise by what we mean:

Definition 8 (Real-valued Multiplication Circuit) *A* **Real-valued Multiplication Circuit** *is a circuit with two special input wires A and B, and a special output wire C, along with a constant λ and a sub-rectangle $\Pi := [a, b] \times [c, d] \in [0, 1] \times [0, 1]$ with positive measure. For any $(x, y) \in \Pi$, if we set wire A to x, and B to y, then the circuit will have a unique solution, with $\lambda \cdot xy$ on the wire C.*

The figure below is an illustration.

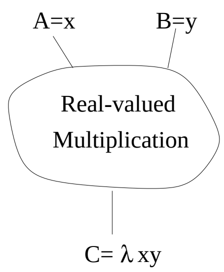

Figure 6: A real-valued multiplication circuit

Unfortunately, this real-valued multiplication circuit is still impossible.

Theorem 7 *There doesn't exist a real-valued multiplication circuit*

Proof: Assume to the contrary, there exists a circuit with n gates, and a rectangle Π that computes real-valued multiplication.

Notice we can view each MAX gate and each MIN gate as a "multiplexer" that outputs one of its inputs. Each gate has two choices, and so there are 2^n possible settings in the whole circuit. We can thus partition Π into 2^n partitions, each one corresponds to a setting of all the MAX and MIN gates.

But since Π has positive measure, one of its partitions must have positive measure, too. Denote this partition by Ω.

Take two points in Ω,(x_1, y_1) and (x_2, y_2). So they should cause the same setting for the circuit, and the circuit should compute the correct product for both points.

Now we claim the circuit would be linear to the input, i.e, if we feed $(\rho \cdot x_1 + (1-\rho) \cdot x_2, \rho \cdot y_1 + (1-\rho) \cdot y_2)$ to the circuit, where ρ is a real number between 0 and 1, the circuit will remain in the same setting, and each wire will also be the linear combination of the value when we feed (x_1, y_1) and the value when we feed (x_2, y_2). This is easily verified by the following facts:

1. If both $a_1 > b_1$ and $a_2 > b_2$, or both $a_1 < b_1$ and $a_2 < b_2$, then $\rho \cdot a_1 + (1-\rho) \cdot a_2 > \rho \cdot b_1 + (1-\rho) \cdot b_2$, and

$$\mathsf{MAX}(\rho \cdot a_1 + (1-\rho) \cdot a_2,\ \rho \cdot b_1 + (1-\rho) \cdot b_2) = \rho \cdot \mathsf{MAX}(a_1, b_1) + (1-\rho) \cdot \mathsf{MAX}(a_2, b_2)$$

2. If both $a_1 > b_1$ and $a_2 > b_2$, or both $a_1 < b_1$ and $a_2 < b_2$, then $\rho \cdot a_1 + (1-\rho) \cdot a_2 > \rho \cdot b_1 + (1-\rho) \cdot b_2$, and

$$\mathsf{MIN}(\rho \cdot a_1 + (1-\rho) \cdot a_2,\ \rho \cdot b_1 + (1-\rho) \cdot b_2) = \rho \cdot \mathsf{MIN}(a_1, b_1) + (1-\rho) \cdot \mathsf{MIN}(a_2, b_2)$$

3. For any a_1, b_1, a_2, b_2,

$$\mathsf{AVG}(\rho \cdot a_1 + (1-\rho) \cdot a_2,\ \rho \cdot b_1 + (1-\rho) \cdot b_2) = \rho \cdot \mathsf{AVG}(a_1, b_1) + (1-\rho) \cdot \mathsf{AVG}(a_2, b_2)$$

So if we feed in $(\rho \cdot x_1 + (1-\rho) \cdot x_2, \rho \cdot y_1 + (1-\rho) \cdot y_2)$ in wires A and B, we will get the $\rho \cdot \lambda \cdot x_1 y_1 + (1-\rho) \cdot \lambda \cdot x_2 y_2)$ at wire C.

But when does that equal the product of the input? i.e., when does

$$\rho \cdot \lambda \cdot x_1 y_1 + (1-\rho) \cdot \lambda \cdot x_2 y_2 = (\rho \cdot x_1 + (1-\rho) \cdot x_2) \times (\rho \cdot y_1 + (1-\rho) \cdot y_2)$$

This is true only when $x_1 = x_2$, or $y_1 = y_2$.

So that means whenever there are two points $P, Q \in \Omega$ that don't have the same x- or y- coordinates, then any point on the line interval PQ (i.e., the points taking the form $\rho \cdot P + (1-\rho) \cdot Q$) is not in Ω. Then Ω only has measure zero[5].
■

6.3 Hybrid Multiplication

It seems we are pretty dead here: in general there is no way to do multiplication in either binary or real value. But, it turns out we can do the hybrid: if one input is a real, while the other is a binary, then we can do the multiplication, and pretty easily. This is due to Manuel Blum.

Definition 9 (Hybrid Multiplication Circuit) *A* **Hybrid Multiplication Circuit** *is a circuit that has $n+1$ input wires: one wire A takes in a real value x, while the other n input wires, denoted by $B_0, B_1, ..., B_{n-1}$ takes in either 0 or 1. And it has an output wire C. For any input to A and B_i, if we view the B_i's as the encoding of an n-bit number, y, then the value of C is $\lambda \cdot x \cdot y/c$, where λ is a constant independent to the input.*

And we have the following positive result:

Theorem 8 (Blum's hybrid) *For any n, there exists a hybrid multiplication circuit of size n.*

Proof's sketch: We only show an example for $n = 4$.

The key observation is that when a is either 0 or 1, and x is between 0 and 1, $\mathsf{MIN}(a, x) = a \cdot x$.

So in the construction, we can use a sequence of AVG gates to get all the "shifts" of x, and then use a sequence of MIN gates to do the multiplication on each line. Finally we use $\log n$ AVG gates to add the partial multiplications up and get the result.

We only use $(n + \log n)$ AVG gates and n MIN gates, and the constant $\lambda = 2^{-\lceil \log n \rceil}$.

■

So it turns out the hybrid model works for this kind of MAX/MIN/AVG circuit.

7 New angels – Simulating boolean circuits

7.1 P-hardness: simulating any circuit

Notice if we only allow binary values, then MAX and MIN are like OR and AND gates. Therefore we could use the MAX/MIN/AVG circuit to simulate a boolean circuit. The problem is no NOT gates are allowed here, and therefore we can only simulate monotone boolean circuit, which isn't very fun.

However, there is a way to use monotone circuits to simulate a normal, non-monotone circuit. The idea is: for each input, we take the complement of that input also, and for each output, we also output its complement. So by introducing the non-monotonicity at the input level, we can eliminate all the NOT gates in the computation.

Let's do an example: suppose we want to compute $\neg(x \wedge y)$, so we take 4 input wires: x, $\bar{x}$, y, and $\bar{y}$. When computing $x \wedge y$, we also compute $\bar{x} \vee \bar{y}$, which is the `NOT` of $x \wedge y$. As shown in the figure.

Actually we just keep track of all gates *along with all their complements.*

In this way, we can simulate all polynomial-size circuits. And also we can do multiplication and all sorts of things. Even more, it is easy to verify that transforming a normal circuit to such a monotone circuit can be done in log space. Therefore we have

Theorem 9 *MAX/MIN/AVG circuit evaluation is P-hard.*

[5] I am pretty sure that it is the case, but I still don't have an easy topological proof for that.

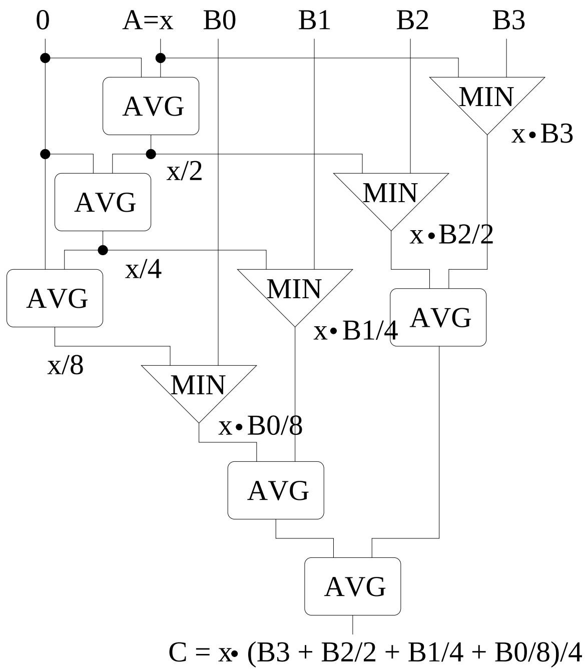

Figure 7: A construction for hybrid multiplication of size 4

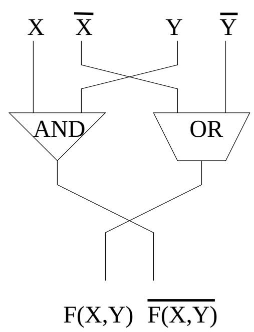

Figure 8: Using monotone circuit to simulate any circuit

7.2 NP-completeness: different formulation of the problem

Since we have the circuit, we can now encode some hard problems, like SAT.

Theorem 10 *The following problem is NP-complete: Given a MAX/MIN/AVG circuit, two gates in the circuit, G_1, G_2, and two real (or rational) numbers x_1, x_2, whether there is a solution to the circuit where the output of G_1 and G_2 are x_1 and x_2.*

The proof is not hard: we will reduce SAT to this problem.

Proof's sketch: Given any instance of SAT, i.e., a boolean formula Φ, we will construct a MAX/MIN/AVG circuit such that Φ is satisfiable, iff the circuit has a solution with two gates having the specified value.

So what we do is we simulate the formula Φ, that is, we start with all inputs and their complements, and for every output, we compute its complement also. So our circuit would consists $2n$ inputs (suppose that Φ has n variables), denoted by x_1, $\bar{x_1}$, x_2, $\bar{x_2}$, ..., x_n, $\bar{x_n}$, and 2 outputs, denoted by S and $\bar{S}$. Denote this circuit by C.

Then the SAT problem is transformed to the problem: does the circuit C has a solution, where for each $i = 1, 2, ..., n$, the set $\{x_i, \bar{x_i}\} = \{0, 1\}$, $S = 1$, and $\bar{S} = 0$.

But the requirements above can be transformed into:

$$\mathsf{MAX}\ \{\mathsf{MIN}\{x_1, \bar{x_1}\}, \mathsf{MIN}\{x_2, \bar{x_2}\}, ...\mathsf{MIN}\{x_n, \bar{x_n}\}, \bar{S}\} = 0$$

$$\mathsf{MIN}\ \{\mathsf{MAX}\{x_1, \bar{x_1}\}, \mathsf{MAX}\{x_2, \bar{x_2}\}, ...\mathsf{MAX}\{x_n, \bar{x_n}\}, S\} = 1$$

We can add more MAX and MIN gates to compute the left hand sides of the above two equations — denote these two results (represented by two gates) G_1 and G_2. Then we know Φ is satisfiable, iff there is a solution the circuit such that $G_1 = 0$ and $G_1 = 1$. ∎

7.3 NP $\cap$ co-NP: another way to look at the problem

So if we fix two gates with their desired values, the problem of the existence of a solution is NP-complete. However, we will see if we just fix one gate with its desired value, and problem is not as hard: actually we will show it is in NP $\cap$ co-NP.

Theorem 11 *The following problem is in NP $\cap$ co-NP: Given a MAX/MIN/AVG circuit C, one gate G in the circuit C, and a real (or rational) number x, whether there is a solution to the circuit where the output of G is x.*

Proof: One direction is easy: that if there exists a solution with the particular gate at the particular value, then simply showing the solution suffices the proof.

By Theorem 5, a concise solution (i.e., a solution where each rational number only needs polynomials many bit to represent) can be shown.

The other direction is a little bit harder. First we utilize the fact that the set of all solutions, Σ, is path-connected (Theorem 3). So the projection of Σ on any of its coordinates is continuous. That means when we look at an arbitrary gate g, and if its values in the maximal and minimal solutions are v_{max} and v_{min}, respectively, then we know for any $v \in [\, v_{\mathsf{min}}, v_{\mathsf{max}}\,]$, there is a solution with value v at gate g.

So if we are given an x, and there is no solution to the circuit that the value of gate g is x, then x must be either greater than the maximal value or less than the minimal value at this gate. We can proof this fact efficiently.

We will use several facts from [C92]:

For any circuit C with n gates and any integer m, there exists a correspondent circuit C_{MIN}, such that:

- The size of C_{MIN} is polynomial in $\max\{m, n\}$.
- C_{MIN} has a unique solution.
- For any gate g in C, C_{MIN} contains a correspondent gate g' of the same type.
- In the minimal solution of C and C_{MIN}, we have $g - 1/2^m \le g' \le g$.

So to prove x is smaller than the minimal value at gate g, one simply construct a circuit C_{MIN}, with proper m, and shows the (unique) solution to C_{MIN}, where the value at gate g' i greater than x, and therefore we know the minimal value of g is also smaller than x

We can do the same thing to show x is greater than the maximal value.

Therefore the problem is also in co-NP. ∎

8 Extended model: MMAN-circuit

In this section, we focus on an extended model of the MMA-circuit, namely the MMAN-circuit: we add a new type of gate, the NOT gate. A NOT gate is a gate with a single input, and the output is 1 minus the input — kind of the "extension" of the NOT in boolean logic.

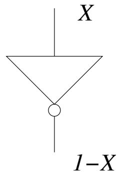

Figure 9: A NOT gate

Apparently the MMAN-circuit should have more computation power than the MMA-circuit since the NOT gate breaks the monotonicity of the MMA-circuit. And it is even not obvious whether the circuit now has a stable solution at all.

We will show the following results in this section:

1. Any MMAN-circuit has at least one stable solution.
2. There exists a continuous mapping that maps any assignment to a solution. This implies that the One-Or-Infinite Theorem is still true for the MMAN-circuit.
3. For any MMAN-circuit C , there exists a corresponding MMA-circuit C', s.t. every stable solution in C corresponds to a solution in C'.
4. The following problem is NP-complete: given a MMAN-circuit C, one gate in the circuit G and a rational number x, whether there exists a stable solution to C where the output of G is x. (Notice the corresponding problem in MMA-circuit is in NP∩ co-NP).

8.1 Converging the MMAN-circuit

We discuss an operation on the MMAN-circuit that will take any assignment to a solution. As in the MMA-circuit, this "operation" is *not* an algorithm since it might take infinitely many steps to finish. However, it can be defined mathematically.

Operation 2 (Converging an arbitrary assignment to MMAN-circuit to a solution) *We describe how to operate on the circuit in the certain way to make sure it converges.*

- **CONVENTION** *We arbitrarily order the gates with the only restriction that the* NOT *gates are all at the beginning: suppose there are n gates in total and there are m* NOT *gates, there the gates $G_1, G_2, ..., G_m$ are the* NOT *gates and the rest gates are ordered as $G_{m+1}, ..., G_n$.*
- **INPUT** *The input is an assignment $\langle a_1, a_2, ..., a_n\rangle$, where a_i is the assigned output value for gate G_i, and $0 \leq a_i \leq 1$ for $i = 1, 2, ..., n$.*
- **OUTPUT** *After the operation, the circuit will stay in the solution $\angle s_1, s_2, ..., s_n\rangle$, such that each gate is satisfied if s_i is the output value of gate G_i. The solution is uniquely determined by the input assignment.*
- **ACTION** *The actual actions are hierarchical, and we will define them in terms of "levels"s.*
  - **Level 0**
    The action of level 0 is: we fix the values of all the NOT *gates and let the rest circuit (which is now an MMA-circuit) converge to a solution.*
  - **Level** i **(i= 1, 2, ... m)**.
    The action of level i is recursively defined as follows: fix the gates $G_i, G_{i+1}, ..., G_m$, and let the rest of the circuit (containing $i-1$ NOT *gates) converge to a solution (we will prove later that the rest of the circuit will actually converge to a unique solution). Now if gate G_i is satisfied, namely, the input to G_i equals 1 minus the output of G_i, we are done; otherwise we will continuously change the output of G_i while fixing $G_{i+1}, ..., G_m$ — each time G_i changes, the rest of the circuit might change and so is the input to G_i. We first*

move the output of G_i up until 1 and then move it down from 1 to 0 (see figure 10) — during the move, the gates that are not fixed are changed as well so as to make sure they are always satisfied. We will prove later that in the process of changing G_i, there is at least a moment that G_i is also satisfied. We stop at the first time G_i is satisfied and we are done.

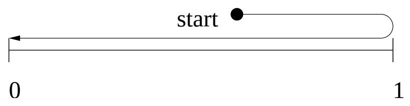

Figure 10: The way to change the output value of G_i

Intuitively, the operation is to rank all the NOT gates and then inductively "turn the knob" for each NOT gate to find a solution — but to turn a knob at a higher level would require all the knobs at lower levels to be turned continuously. This operation is more complicated than the one for MMA-circuit.

The reason our operation is valid is guaranteed by the following lemma:

Lemma 10 *At each level i of action, the following properties are preserved:*

1. *The operation is deterministic, i.e., the value of the gates after one step of level i is uniquely defined, and that naturally defines a function for mapping an assignment before one step of level i to an assignment after the step of level i. We denote this function by $\phi : [0,1]^n \mapsto [0,1]^n$.*
2. *The function ϕ is "shrinking", namely, given any two assignments A_1 and A_2 for the same circuit to start with, then after one step of level i, suppose the values become B_1 and B_2, then we have $\mathsf{dist}(A_1, A_2) \geq \mathsf{dist}(B_2, B_2)$.*
3. *The function ϕ is (uniformly) continuous.*
4. *The function is piece-wise affine. In other words, one can break the domain into finitely many sub-domains, such that in each domain, function is affine, i.e., it can be written as*

$$\phi(X) = M \cdot X + C$$

where M is a constant matrix and C is a constant vector.

Proof: We prove lemma 10 by induction on the number of the NOT gates.

Base case: $i = 0$. The circuit is actually an MMA-circuit and the lemma follows directly from lemma 2 and lemma 6. The affine-ness needs proof, but this is can be done by straight-forward induction.

Inductive case: Suppose we have proved the lemma for all the MMAN-circuits with no more than $(i-1)$ NOT gates, now we look at an MMAN-circuit of i NOT gates.

We first show that the operation will deterministically map an arbitrary assignment to a solution.

The i NOT gates are, by the convention, ordered as $G_1, G_2, ..., G_i$. Now if we "break" the gate G_i, namely, we fix the output of G_i to an arbitrary value X, and let the input to G_i be a "floating wire" (see figure 11), we get an MMAN-circuit of $(i-1)$ NOT gates.

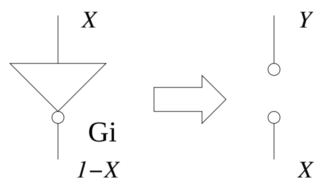

Figure 11: Breaking the gate G_i

By inductive hypothesis, the operation will take any assignment and map the assignment to a solution, and the mapping is shrinking and continuous. No consider the following experiment: we give the operation different assignments as input — all these assignments only differ on the value of X (i.e., the value of the original output of G_i) and are the same else-where. The operation will give back a solution for each of these assignments, and if we only focus on the value of Y, (i.e., the value of the "floating wire" which was the input of the original gate G_i), we get a function

$$Y = f(X)$$

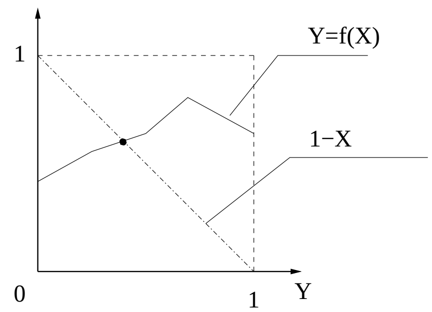

Figure 12: The input to gate G_i as the function of the output from gate G_i

The function f is a "restricted version" of the mapping given by the inductive hypothesis: we fix all other inputs except X, and we only look at Y of the output. This restricted function is still shrinking and continuous. Both the domain of the range of f are the interval $[0, 1]$: see figure 12.

Notice we have $f(0) \leq 1$ and $f(1) \geq 0$, and since f is continuous, it must intersect with the line $Y = 1 - X$, i.e., there must exist at least a point a, such that if we assignment X to be a, then $Y = 1 - a$ and thus we "put back" the gate G_i, it will be satisfied. In the operation, one finds such a point a by moving X back and forth (first all the way up to 1 and then all the way down to 0), and therefore guarantees this will produce a unique solution.

Now we show the mapping defined by the operation on MMAN-circuits of i NOT gates is shrinking and continuous. In other words, we show that given two assignments A_1 and A_2 to the same MMAN-circuit of i NOT gates, the solution we end up getting, B_1, B_2, satisfy:

$$\mathsf{dist}(B_1, B_2) \leq \mathsf{dist}(A_1, A_2)$$

We write A_1 and A_2 as:

$$A_1 = \langle a_1^1, a_2^1, ..., a_n^1 \rangle$$

and

$$A_2 = \langle a_1^2, a_2^2, ..., a_n^2 \rangle$$

Then after one step of level $i - 1$ operation, we should get two assignments:

$$C_1 = \langle c_1^1, c_2^1, ..., c_n^1 \rangle$$

and

$$C_2 = \langle c_1^2, c_2^2, ..., c_n^1 \rangle$$

where the gates $G_1, G_2, ..., G_{i-1}, G_{m+1}, ..., G_n$ are all satisfied and we have $c_k^1 = a_k^1$, $c_k^2 = a_k^2$, $k = 1, 2, ..., i$. Also by inductive hypothesis, we know that

$$\mathsf{dist}(C_1, C_2) \leq \mathsf{dist}(A_1, A_2)$$

and we will show that

$$\mathsf{dist}(B_1, B_2) \leq \mathsf{dist}(C_1, C_2)$$

For C_1 and C_2, each of them determines a function from the output of gate G_i to the input of gate G_i — we use f_1 and f_2 to denote the two functions, as shown in figure 13. Suppose that X_1 and X_2 are the intersection points.

We use D to denote $\mathsf{dist}(C_1, C_2)$. Then by inductive hypothesis, we know that for any X, $|f_1(X) - f_2(X)| \leq D$ — think of the two assignments where they agree on gates $G_1, G_2, ..., G_{i-1}, G_{m+1}, ..., G_n$ (value doesn't matter), they both have X at gate G_i and their values on $G_{i+1}, ..., G_m$ are those of C_1 and C_2, respectively. They will converge so that gate G_i has value $f_1(X)$ and $f_2(X)$, respectively, while the original distance is bounded by D.

Now let's prove that $|X_1 - X_2| \leq D$. Otherwise we assume that $X_1 < X_2$, as shown in figure 13. We pick another point X' such that $X_1 + D < X' < X_2$. Then notice that since both f_1 and f_2 are "shrinking" by the inductive hypothesis and thus we know

$$f_2(X') \leq f_2(X_2) + (X_2 - X'') = 1 - X'$$

and since $f_1(X_1) = 1 - X_1$, we have

$$|f_1(X_1) - f_2(X')| > |X_1 - X'|$$

But this is bad — now we pick two assignments E_1 and E_2: Let E_1 and E_2 agree on gates $G_1, ..., G_{i-1}, G_{m+1}, ..., G_n$, let the G_i of E_1 be X_1 and the G_i of E_2 be X', and let the gates $G_{i+1}, ..., G_m$ of E_1 be the same as C_1 and the gates $G_{i+1}, ..., G_m$ of E_2 be the same as C_2. Then after one operation of level $i - 1$, the gate G_i should turn to $f_1(X_1$ and $F_2(X')$, respectively, violating the shrinking property.

Therefore $|X_1 - X_2| \leq D$ and thus the function is shrinking and thus continuous. The inductive case is done.

∎

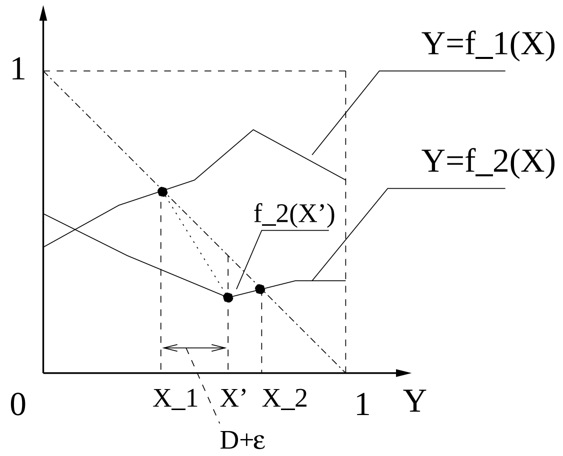

Figure 13: The two functions

8.2 Another way to converge

It turns out that the "time-average" operation for the MMA-circuits can also be used here and also converges. The proof needs minor changes:

Theorem 12 *The time-average operation g also converges in the MMAN-circuits.*

Proof: Notice that g is still shrinking and continuous. But it is not monotone any more. We follow the proof 2, the linear algebra proof. The partition still holds, and we need to modify the proof about the determinant a little, namely, lemma 3. Notice if we don't include the constants in the vectors, it is still true that for each of the linear (should be affine, to be more precise) function, each row of the matrix adds up to be at most 1. But it is no longer true that each entry along the diagonal is at least ϵ: think of what happens if a NOT gets its input from itself. However, we can still bound the determinant — all we want to show is: If there is a row that adds up to be less than 1, then it is at most $(1+\epsilon)/2$ and the whole determinant is at most $(1+\epsilon)/2$. Otherwise we still have each entry in the diagonal is at least ϵ and at most $(1+\epsilon)/2$ if not 1, and all the rest proof follows.

■

8.3 One-or-Infinite Theorem

Now that we have a way to converge the circuit, we got the following theorems:

Continuous:

Lemma 11 *There exists a mapping ϕ, that takes an arbitrary assignment to any MMAN circuit and outputs a solution. The mapping is uniformly continuous.*

Theorem 13 *The set of all the solutions, denoted by Σ, is path-connected.*

Corollary 4 (One-Or-Infinite Theorem, MMAN circuit) *If Σ contains more than one point, then it contains uncountably many.*

8.4 Relations between MMA-circuits and the MMAN-circuits

Here we show that there is a relation between the MMA-circuits and the MMAN-circuits. Or to be exact, we show how to construct an MMA-circuit C' from any MMAN-circuit C, such that any solution in C corresponds to a solution in C'.

The idea is like the one of simulating any boolean circuit using the MMA-circuits we mentioned before. Below is an outline of what we do:

Given an MMAN-circuit C, we do the following:

1. **Break all the NOT gates** First we break all the NOT gates as in figure 11. Thus we get a "broken" MMA-circuit — denote it my C_1.

2. **Create a mirror circuit** Then we create a "mirror" circuit for C_1 — we first create a clone of C_1, then replace all the MAX gates by the MIN gates while replacing all the MIN gates by the MAX gates in the clone, and replace all the 0's by 1's and replace 1's by 0's. Name the resultant circuit by C_2.

3. **Cross-link** Now we want to connect the two "mirrored" circuits: for each NOT gate in C, it corresponds to one input wire and an output wire in both C_1 and C_2. We connect the input wire in C_1 to the output wire in C_2, and connect the input wire in C_2 to the input wire in C_1.

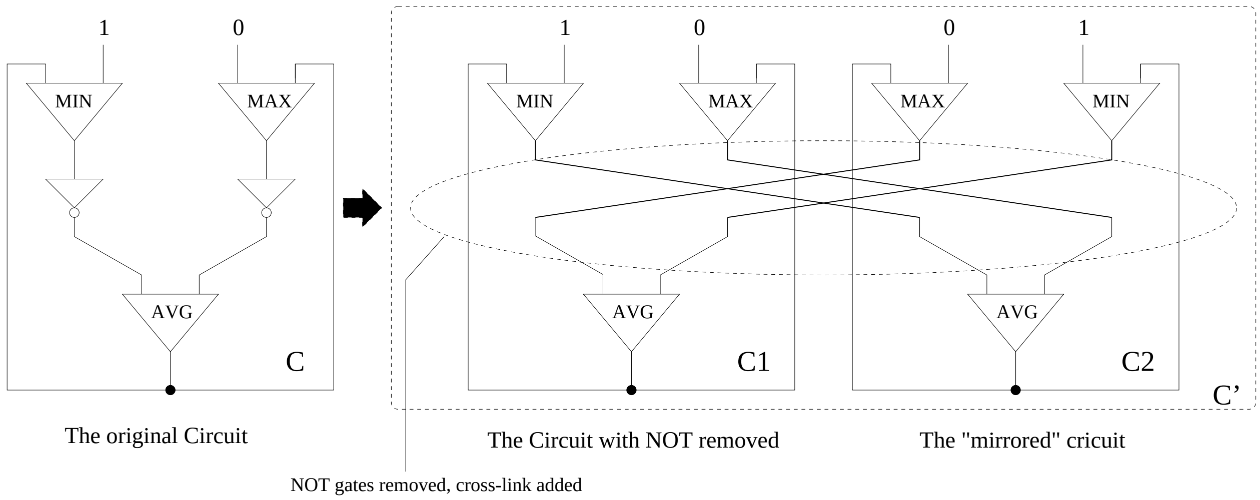

Figure 14: How to convert an MMAN-circuit to an MMA-circuit

Figure 14 shows an example of such construction.

Now we show that any solution in C have a corresponding solution in C'. Actually this is not hard: for each gate G_i in the original circuit, C, we use G_i^1 and G_i^2 to denote the corresponding gates in circuit C' — G_i^1 is in the circuit with NOT removed and G_i^2 is in the "mirrored" circuit. Then we look at a solution S for circuit C — we use s_i to denote the value of the output of gate G_i. Then we claim that if we assign s_i to gate C_i^1 and $(1-s_i)$ to gate C_i^2, we get a solution for the corresponding circuit C'.

The proof is omitted. ■

Notice this construction gives a one-way mapping from a solution for a MMAN-circuit to a solution for the MMA-circuit and we don't know if it is true in general.

8.5 Unique-solution Circuit

Circuits with unique solutions are always interesting. One common trick to convert an arbitrary circuit to a circuit with a unique solution is to insert a $(1-\epsilon)$ "gate" (actually a sequence of gates) after each of the original gates. In the MMA-circuit, it is shown that it will always make a unique-solution circuit and the solution in the new circuit is very close (can be arbitrarily close if you make ϵ very small) to the *minimal* solution in the original circuit. Now the question is what happens if we play the same trick to an MMAN-circuit?

We will show that we still get a unique-solution circuit back, and the solution to the new circuit is close to *a* solution in the original one — it is not clear what solution it looks like to us now.

Actually the uniqueness is not hard to show, if not trivial. One can use Ann Condon's original proof — one just "runs" the circuit and it is easy to see that the "change" one makes in each round is shrinking and thus the "running" will converge to a unique solution. But my favorite proof is the one by Manuel and Rachel: If we fix a MIN/MAX setting, we get a linear equation like $X = A \cdot X + B$, which has a unique solution since $||A|| \leq 1-\epsilon < 1$ and thus there is at most one solution for each MIN/MAX setting. Therefore the total number of solutions is bounded by the total number of MIN/MAX settings, which is finite. By the one-or-infinite theorem, there can only be a unique solution.

Now we can have another interesting observation (this is due to Rachel):

Lemma 12 *Given a oracle that can solve the unique-solution MMA-circuit, we can solve any unique-solution MMAN-circuit that is generated by the $(1-\epsilon)$ trick (aka. "stopping game").*

Proof's sketch: Suppose we are given an MMAN-circuit C, we can construct its corresponding MMA-circuit C'. The crucial observation is that C' is also a unique-solution circuit since still every gate is followed by a $(1-\epsilon)$ gate. Now we

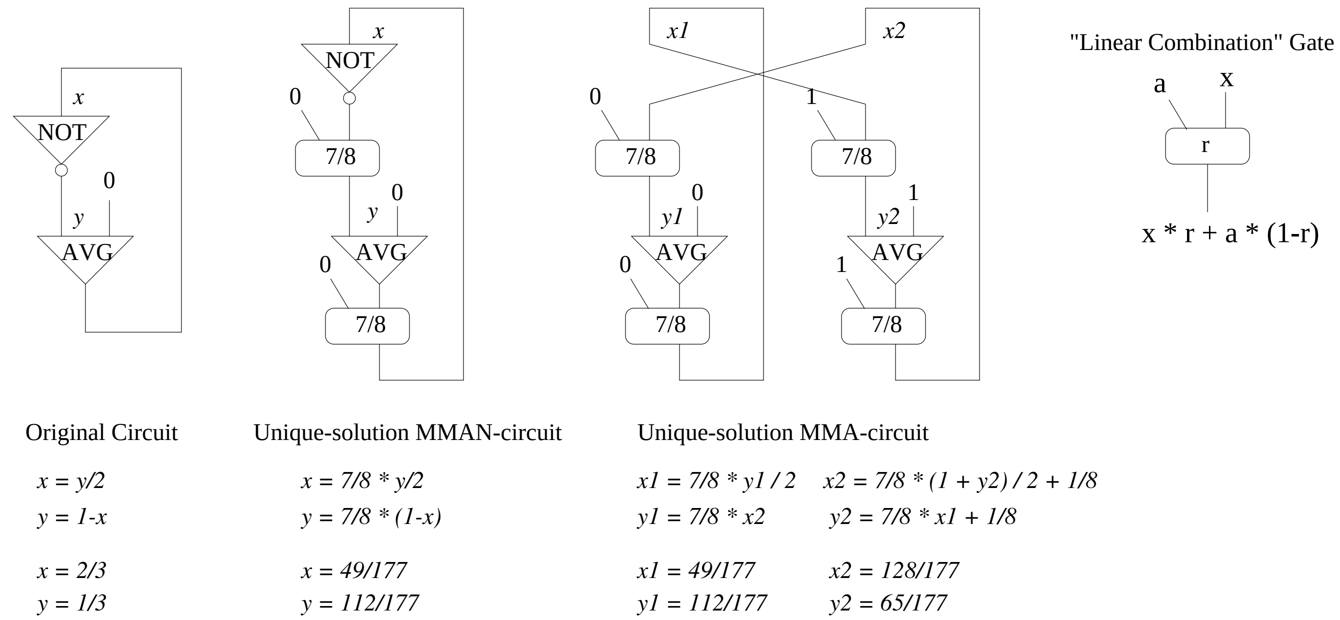

Figure 15: Finding the unique solution for an MMAN-circuit

can use the oracle to find the solution for C'. Notice C' must have a solution that's corresponding to the solution for C, and since C' only has one solution, this solution is the corresponding solution for C! And we can just "read off" the solution for C. ∎

Figure 15 shows an example — how one starts with an MMAN-circuit, transform it into a "stopping-game" circuit and further transfer that to a unique-solution MMA-circuit. One can see that we can "read off" the solution for the "stopping-game' MMAN-circuit from the solution of the MMA-circuit.

Notice the following distinction between MMA-circuits and MMAN-circuits:

- For MMA-circuits, if a circuit has a unique solution, then it is a "stopping-game", i.e., for every gate, there exists a path from a constant input (either 0 or 1) to this gate. The reason being that otherwise we can find a closed "sub-circuit" that doesn't link to any constant gates. Then for any real number between 0 and 1, assigning every gate in the sub-circuit that value would give us a solution back — notice that $\mathsf{MAX}(x,x) = \mathsf{MIN}(x,x) = \mathsf{AVG}(x,x)$.
- For MMAN-circuit, it is possible for a circuit to have unique solution yet not being stopping. The simplest example is $x = 1 - x$. Actually the NOT gate does introduce a constant into the circuit.

We actually have the following theorem:

Theorem 14 *Given an oracle S that can solve any unique-solution MMA-circuit, there exists a polynomial-time algorithm A, such that A^S solves any MMAN-circuit. Namely, A^O takes an MMAN-circuit and outputs one solution.*

Proof: The algorithm A first converts the MMAN-circuit (denoted by C) to a stopping game C' and uses Lemma 12 to find the unique solution for C'. Now what remains to be shown is that there exists a solution for C that is very close to the solution for C' (we use $S(C')$ to denote the solution for C'), and thus we can simply "read off" the solution for C from $S(C')$, if we choose C' to be very close to C.

So the idea is: we do a series of experiments for C': in the k-th experiment, we pick the ϵ to be 2^{-k}, and we get a (unique) solution S_k. Now look at the sequence $\{S_k\}$, and we know a sub-sequence of $\{S_k\}$ will converge, say, to $\bar{S}$. Now we show that $\bar{S}$ is the solution for the original circuit C.

It is actually basic analysis: We look at the converging sequence $\{S_{k_n}\}$. We know that

$$\forall \delta > 0\ \exists N\ s.t. \forall n > N, ||S_{k_n} - \bar{S}|| < \delta$$

Now take an $n > N$ such that $2^{-k_n} < \delta$. We now that $g_k(S_{k_n}) = S_{k_n}$, where g_k denotes the function for the stopping game when $\epsilon = 2^{-k}$. We also know that $||g_k(X) - g(X)|| \leq 2^k$ for any X. Therefore we know that

$$\begin{aligned} ||g(\bar{S}) - S|| &\leq ||g(\bar{S}) - g(S_{k_n})|| + ||g(S_{k_n}) - g_k(S_{k_n})|| + ||g_k(S_{k_n}) - S_{k_n}|| + ||S_{k_n} - \bar{S}|| \\ &\leq \delta + 2^{-k} + 0 + \delta \\ &\leq 3\delta \end{aligned}$$

This is true for any δ, and thus we have $g(\bar{S}) = \bar{S}$. Therefore we know a subsequence of $\{S_k\}$ converges to a solution for C, and then we can show the whole sequence actually converges. This is a simple exercise in analysis and is left to the reader :-).

■

So in this sense, adding a NOT gate doesn't give you a lot of computation power, since an MMA-circuit oracle can solve any MMAN-circuits.

9 Another Extended Model: MMCC

We study another extended model for the MMA circuit, which also turned out to be useful. This section is self-contained and be read independent from the rest of the paper.

9.1 Definitions and Notations

An *MMA circuit* is a circuit with fan-in two max, min, and average gates. An input to a gate g in an MMA circuit may be 1, 0, or the output of any gate in the circuit, including g itself. To avoid excessive notation, we often use the same variable to denote both a gate and its output.

An *assignment* to a circuit is a function from its gate set to the interval $[0, 1]$, specifying a value for the output of each gate. Suppose g is a gate with inputs a, b and output c, that computes a function f . Then g is *satisfied* under assignment A if $f(A(a), A(b)) = A(c)$. An assignment which satisfies all gates in a circuit C is a *solution* to C.

An *MMCC circuit* is a circuit with fan-in two max and min gates, and *convex combination* (cc) gates. Convex combination gates compute a rational convex combination of 1, 0, and the outputs of all gates in the circuit. MMA circuits are clearly a special case of MMCC circuits; we prove elsewhere that if the coefficients in the convex combinations in an MMCC circuit C are all of the form p/q, with $p, q \leq 2^n$, then C can be simulated by an MMA circuit with $O(n)$ gates.

The *simple stochastic game* (ssg) corresponding to an MMCC circuit is a directed graph with one node for each gate, and a node each for 0 and 1. There is an arc in the graph from x to y iff y is an input to x in the circuit. Each node in the graph is labelled as max, min, or convex combination, according to the gate it represents.

A *stopping game* is an ssg with the property that, for any subgraph G in which each max and min node has outdegree one, and for any node $v \in G$, there is a directed path in G from v to either 1 or 0 (or both). If the ssg corresponding to an MMCC circuit C is a stopping game, we also call C a stopping game.

A gate g in an MMA or MMCC circuit may be *reconfigured* by (possibly) changing the function g computes, and adding or deleting inputs to g. The function the reconfigured gate computes may be max, min, or any convex combination; note that any rational constant c between 0 and 1 may be computed by a convex combination gate.

We may think of an assignment to the n gates of a circuit as a vector. The *feasible polytope* for an MMCC circuit C is a subset of the n-dimensional hypercube defined by inequalities of the form $x_j \leq x_k$ for each min gate with input x_k and output x_j; $x_j \geq x_k$ for each max gate with input x_k and output x_j; and $x_j = \alpha_0 + \sum_{i=1}^{n} \alpha_i x_i$ for a convex combination gate g with output x_j, where $\alpha_0 + \sum_{i=1}^{n} \alpha_i x_i$ is the convex combination of 0, 1, and the outputs of gates $x_1, \ldots x_n$ computed by g.

An assignment U is an *upper bound* on a solution S to an MMCC circuit C if, for every gate x, $U(x) \geq S(x)$. A *Hoffman-Karp upper bound* is an upper bound in which all gates other than min gates are satisfied, and each min gate is equal to one of its inputs. A *Hoffman-Karp lower bound* is a lower bound in which all gates other than max gates are satisfied, and each max gate is equal to one of its inputs.

9.2 Geometry

We claim, but do not prove here, that

- without loss of generality, the feasible polytope F has full dimension;
- if a circuit has a unique solution v, then v is a vertex of F;
- every stopping game has a unique solution.

Theorem 15 *Let C be an MMCC stopping game. Let F be the feasible polytope for C. Let $v \in F$ be the vertex of F which is the unique solution to C. Let $k \notin F$ be a Hoffman-Karp lower bound for v. Then for any real p, $0 < p \leq 1$, the point $pk + (1-p)v \notin F$.*

Proof: Assume without loss of generality that in the solution point v, all max gates are set to the left.

Let X_r be the set of max gates $\{x : x_k = r_k \neq l_k\}$, where x_k is the value of x at k, and l and r are the left and right inputs to x, respectively. We partition X_r into two sets $X_{r,\mathsf{MAX}}$ and $X_{r,\mathsf{MIN}}$; a gate x is in $X_{r,\mathsf{MAX}}$ if $x_k = r_k > l_k$; it is in $X_{r,\mathsf{MIN}}$ if $x_k = r_k < l_k$. We consider the solutions to three circuits C, C_1, and C_2. C_1 is obtained from C by fixing all max gates not in X_r to the left, leaving the gates in X_r as max gates. C_2 is obtained from C by fixing the max gates not in X_r to the left, and the gates in $X_{r,\mathsf{MIN}}$ to the right, leaving the gates in $X_{r,\mathsf{MAX}}$ as max gates. Alternatively, C_2 can be obtained from C_1 by setting all the gates in $X_{r,\mathsf{MIN}}$ to the right.

Since C is a stopping game, C_1 and C_2 are also stopping games and so have unique solutions. The solution to C is the vertex v. v is also a solution to C_1, since in v all max gates are set to the left. Let v_2 be the solution to C_2. Notice that $k = v_2$, since k is a solution to C_2. Since $k \notin F$, $X_{r,\mathsf{MIN}} \neq \emptyset$. Let $x \in X_{r,\mathsf{MIN}}$, and let l and r be the left and right inputs to x. Then

$$\begin{aligned} px_k + (1-p)x_v &< pl_k + (1-p)x_v \\ &= pl_k + (1-p)l_v \end{aligned}$$

Since $c = pk + (1-p)v$ has a max gate x which is strictly lower than one of its inputs, c is not in F. Thus no convex combination of a Hoffman-Karp lower bound point and v is in F, other than v itself. ■

We say a point $p \notin F$ can "see" a point $f \in F$ if the line between f and p lies entirely outside F, except for the point f. The above theorem says that every Hoffman-Karp lower and upper bound can see the solution vertex.

Corollary 5 *Let C be an MMCC stopping game with unique solution v, where v is a vertex of the feasible polytope for C. If there is any bounding plane of F which cannot be seen by some Hoffman-Karp lower or upper bound, the dimension of the problem can be reduced by one.*

Proof: If some bounding plane $x \leq y$ cannot be seen by some Hoffman-Karp point, then the solution vertex cannot be on that plane. It follows that, if the inputs to x are y and z, x must equal z at the solution vertex. Thus we can set $x = z$, reducing the dimension by one. ■

9.3 MMA circuits are equivalent to MMCC circuits

An MMA circuit is an MMCC circuit in which all convex combination gates compute the average of just two inputs. The following theorem shows that MMA and MMCC circuits are essentially equivalent.

Theorem 16 *Let C be an MMCC circuit with m gates, n of which are convex combination gates. If the convex combinations in C all have rational coefficients of the form p/q, with $0 \leq p \leq q \leq 2^n$, then C can be simulated be an MMA circuit with $O(m + n^4)$ gates.*

Proof: **Lemma 13** *Let C be an MMA circuit with gates $x_1, \ldots x_n$. and let $k = \alpha_0 + \sum_{i=1}^{n} \alpha_i x_i$, where $\alpha_i \geq 0$, $i = 0, \ldots, n$ and $\sum_{i=0}^{n} \alpha_i \leq 1$, and for each α_i, $\alpha_i = p_i/q_i$, with $0 \leq p_i \leq q_i \leq 2^n$. Then a circuit C' can be constructed from C by adding $O(n^2)$ average gates, such that one of the new average gates will have the convex combination k as its output.*

Proof: Let x, y be any two gates of C. By Theorem 4, it is possible to add O(m) average gates to any MMA circuit C in such a way that one of the new gates computes any convex combination of x and y, provided all coefficients in the convex combination have numerator and denominator bounded by $2^{O(m)}$. We construct the convex combination k one coefficient at a time, producing the following sequence of convex combinations:

$$\begin{array}{rcl} \left(\frac{\alpha_0}{1-\sum_{i=1}^{n}\alpha_i}\right)\cdot 1 & + & \left(\frac{1-\sum_{i=0}^{n}\alpha_i}{1-\sum_{i=1}^{n}\alpha_i}\right)\cdot 0 \\ \left(\frac{1-\sum_{i=1}^{n}\alpha_i}{1-\sum_{i=2}^{n}\alpha_i}\right)\left(\frac{\alpha_0}{1-\sum_{i=1}^{n}\alpha_i}\right) & + & \left(\frac{\alpha_1}{1-\sum_{i=2}^{n}\alpha_i}\right)\cdot x_1 \\ \left(\frac{1-\sum_{i=2}^{n}\alpha_i}{1-\sum_{i=3}^{n}\alpha_i}\right)\left(\frac{\alpha_0+\alpha_1 x_1}{1-\sum_{i=2}^{n}\alpha_i}\right) & + & \left(\frac{\alpha_2}{1-\sum_{i=3}^{n}\alpha_i}\right)\cdot x_2 \\ & \cdot & \\ & \cdot & \\ & \cdot & \end{array}$$

$$\left(\frac{1-\sum_{i=j}^{n}\alpha_i}{1-\sum_{i=j+1}^{n}\alpha_i}\right)\left(\frac{\alpha_0+\alpha_1x_1+\cdots+\alpha_{j-1}x_{j-1}}{1-\sum_{i=j}^{n}\alpha_i}\right) \quad + \quad \left(\frac{\alpha_j}{1-\sum_{i=j+1}^{n}\alpha_i}\right)\cdot x_j$$

$$\vdots$$

$$(1-\alpha_n)\left(\frac{\alpha_0+\alpha_1x_1+\cdots+\alpha_{n-1}x_{n-1}}{1-\alpha_n}\right) \quad + \quad (\alpha_n)\cdot x_n$$

The last line is the convex combination k. Each coefficient in each convex combination has numerator and denominator bounded by 2^{2n^2}, so by Theorem 4, each of the $n+1$ convex combinations requires $O(n^2)$ new average gates to be added to C. ■

We construct an MMA circuit C' to simulate C as follows: for each max or min gate in C there is a corresponding max or min gate in C'. Using Ke's method, we add $O(n^4)$ average gates, n of which output the n convex combinations computed by the cc gates in C. The inputs to the max and min gates are the same in C' as in C. It is easy to see that from any solution to C a solution to C' can be computed in linear time, and *vice versa.* ■

9.4 MMCC Circuits and 2-SAT

We now show that, under an unrealistically strong assumption, the stable circuit problem can be reduced to 2-SAT.

Lemma 14 *Let C be an MMCC stopping game with unique solution S. Let x and y be two max gates in C. Let A be an assignment to C in which x and y are both set to the lower of their two inputs, and all other gates are satisfied. Let x_l, x_r, and y_l, y_r be the inputs to x and y, respectively. If $A(x) = A(x_l)$ and $A(y) = A(y_l)$, then either $S(x) = S(x_r)$ or $S(y) = S(y_r)$.*

Proof: Call x_l and y_l the left inputs to x and y. Consider the circuit C' obtained from C by setting both gates x and y equal to their left inputs. C' must be a stopping game, since C is a stopping game. A satisfies all gates of C', so A is the unique solution to C'. If in S both x and y are set to the left, then S is a solution to C', so $S = A$. Thus either A is a solution to C, or at least one of x and y is set to the right in S. ■

Theorem 17 *Suppose we have a polynomial time algorithm TwoGate which takes as input an ordered pair $\{C, K\}$, where C is an MMCC stopping game, and k is a Hoffman-Karp lower bound on the solution to C, and outputs a Hoffman-Karp lower bound K' which dominates K, and in which at most two max gates are unsatisfied. Then there is a polynomial time algorithm to find a solution to C.*

Proof: Using TwoGate as an oracle, we reduce the problem of finding a solution to C to 2-SAT. Number the max gates of C $x_1, \ldots x_k$. Let S be the unique solution to C. We construct a 2-DNF formula F one clause at a time, using variables $x_1, \ldots x_k$, where $x_i = 1$ iff gate x_i is equal to its right input in S. We maintain the property that the the settings of the max gates in S must satisfy F. After at most $4\binom{k}{2}$ rounds, any satisfying assignment to F agrees with S on all max gates. For any assignment A, define x_i^A to be x_i if x_i is equal to its right input in A; otherwise $x_i^A = \neg x_i$. Let S_0 be an arbitrarily chosen Hoffman-Karp lower bound.

Step 0. Set $F := \bigwedge_{i=1}^{k}(x_i \vee \neg x_i)$; set $i := 0$; set $C_0 = C$.

Step 1. Using TwoGate, find an assignment $A_i \geq S_i$, with at most two unsatisfied max gates x_j and x_k. Add the clause $(\neg x_j^{A_i} \vee \neg x_k^{A_i})$ to F.

Step 2. Find a satisfying assignment T to F. T corresponds to a right-left setting of the max gates in C. Reconfigure C by setting each max gate right or left according to T. Set C_i to be the reconfigured circuit. Use linear programming to find the unique solution S_i to C_i. If S_i satisfies C, halt. Otherwise, go to 9.4 Step 3.

Step 3. For every max gate x unsatisfied by S_i, reset x to its other input. Set C_i to be the resulting circuit.Use linear programming to find the unique solution S_i to C_i. If S_i satisfies C, halt. Otherwise, go to 9.4 Step 1.

Proof that the above algorithm halts: Any solution found in 9.4 Step 2 is a Hoffman-Karp lower bound. If the solution found in the j^{th} iteration in 9.4 Step 2 is not a solution to C, then 9.4 Step 3 finds a strictly higher Hoffman-Karp lower bound. In round $j+1$ the TwoGate algorithm then also finds a Hoffman-Karp lower bound which strictly dominates the one found in round j. Since C and all of the reconfigured circuits C_i are stopping games with unique solutions, the circuits $C_0, \ldots C_{j+1}$ must all be distinct. It follows that the clause added to F in round j is distinct from all clauses added to F in earlier rounds. Since there are only $4\binom{k}{2}$ possible clauses, the algorithm must halt in at most $4\binom{k}{2}$ rounds. It halts only when it finds a solution to C, so the algorithm finds a solution to C in time polynomial in the number of max gates in C. ■

10 Application: Real-life games

Wouldn't that be boring if we just talk about something theortically that had nothing to do with our daily life? This section we discuss how the MMA-circuit can be used in real life, and especially, 2-person, stochatic games.

10.1 Two-person, stochastic games

We are interested in the games that are two-person, zero-sum, have randomness (namely, where the players toss coins or dice), and have polynomially many states.

Notice some typical games like GO are proven to be PSPACE-hard and it's hopeless to use our MMA-circuit to solve them. If the game further has randomness, it might be NEXP-hard. One intuition why these games are hard is that they have exponentially many states (possible board configurations) and it takes exponential time just to examine them all (notice the $\alpha - \beta$ pruning is basically enumerating all the states and evaluating the game.

Here we are looking at games that are "easier" — those that only have polynomially many states. Then if the game doesn't have randomness, a simple $\alpha - \beta$ algorithm can solve the problem in polynomial time. However, when the game has randomness in it, it is not clear if there is a polynomial time algorithm to find the value of the game – actually in this case the MMA-circuit becomes a handy model to analysis these games. Typical games like that are Pacheese, Backgammon, Sorry, etc.

10.2 Pachisi, Backgammon, and more

Pachisi is called "The National Game of India" but nobody in our department that comes from India claims that he/she played in India :-). Here is a webpage that describes the game comprehensively: `http://web.ukonline.co.uk/james.masters/TraditionalGames/pachisi.htm`.

We have the following theorem.

Theorem 18 *MMCC circuits are Parchisi-hard.*

Proof: Generalized two-player Parchisi is played on a board with n squares in a cycle, plus a constant number of special start and destination squares for each player. Each player has k tokens, for some constant k. At each turn a player rolls two dice, and has a constant number of legal moves to choose from. The number of possible board positions is $O(n^k)$. Parchisi can be simulated by a simple stochastic game as follows. For each of the $m = 0(n^k)$ possible board positions, there are two convex combination nodes, representing the two players. A convex combination node representing player 1 (the max player) has nonzero coefficients for max nodes representing the 21 possible sets of legal moves available to choose from, depending on which of the 21 possible rolls comes up, with coefficients equal to the probability of each roll. Each max node as represents the pair (board position, roll of the dice). There is a constant number of arcs out of each max node, one to each of board positions to which the max player can legally move, given the current board position and roll of the dice. (Strictly speaking, we should have at two arcs out of each max node, but we can replace any constant number a of arcs with a series of $\log(a)$ max nodes, each with at most two outward arcs.) Similarly, a convex combination representing player 2 (the min player) has nonzero coefficients for min nodes representing sets of legal moves for all possible rolls of the dice. A node representing a board position which is a win for the max player is labelled 1, and a win for the min player is labeled 0. The total number of nodes is $O(n^k)$. Constructing the circuit takes time polynomial in n. The minimum solution to the MMCC circuit corresponding to the ssg represents optimal strategies for the max and min player; thus if a solution to an MMCC circuit can be found in time polynomial in n, then an optimal strategy for each player in a game of Parchisi may be found in time polynomial in n. ■

Backgammon seems to be a much more popular game in the US and there is an entire website devoted to that: `http://www.bkgm.com` and the machine learning people also seem to have a great interest in that.

Some common features shared by Pacheese and Backgammon are:

1. Each player has constant number of tokens (polynomially many states).
2. The players take turns and compete with each other (MAX/MIN gates).
3. The players toss dice (AVG gates).
4. One player can send a token of the other player back and thus progresses can't always be made (feedback).

These properties make the MMA-circuit a good model for the games. There are other similar games like this that can also be solved by the MMA-circuit. Therefore an immediate result says:

Theorem 19 *Parcheese and Backgammon are both in NP $\cap$ co-NP.*

Notice this is NOT a new result — Ann Condon already proved this in her paper [C92].

10.3 Some results about convergence

We did computer simulations on various versions of pacheese and there are some interesting results.

10.3.1 Straight-line parchisi

The simplest version of parchisi we study is the Straight-line Parchisi: There are n cells lined up and we number them from 0 to $n-1$. Each of the both players has two tokens that originally placed at cell 0. The players take turns: in each turn, a player tosses a fair coin and interpret the result as "move forward 1 step" and "move forward 2 steps". Then the player chooses which token to move, and move that token the number of steps as indicated by the coin-toss. A token can't be moved outside the board: that means if a token is at cell $n-2$ and the coin-toss comes out with "move forward 2 steps", then this token can't be moved. If a player's both tokens are at cell $n-2$ and the coin-toss gives a 2, then he simply has no play.

If player A's token lands on a cell which is occupied by 1 or 2 of player B's tokens, the tokens of player B are sent back (there is no "protection" as in the real parcheese) while player A's token remains.

The game terminates when a player moves both of his tokens into cell $n-1$, and the first player that does this wins.

We found that this game is *very easy* to solve, in the sense that the most naïve value-updating algorithm works well, and so does the time-average algorithm. Our computer simulation also shows that algorithms converge very quickly to a solution.

Here is an idea why this is the case (not a formal proof yet): One way to look at the value-update problem is to think of the circuit as a directed graph, which each arc has the reverse direction as in the circuit, and then both players perform random walks on the graphs – they start from the node that corresponds to the initial configuration and try to move to one of the constant nodes (0 and 1). One can prove that this is exactly equivalent to solving the game. Notice still each player has different "policies", namely the setting of the MAX/MIN gates. However, if we can prove that if for each pair of policies, the random walk finishes in polynomial time with reasonably high probability, then the value-updating algorithm converges fast — this can be viewed as a generalization of Rue's convergence lemma.

Lemma 15 *Given an MMA-circuit, if, for any way of setting the* MAX/MIN *gates, a random walk starting from the output node (i.e., the intial state) will hit one of the constant node within $P(n)$ steps with probability as least $1/Q(n)$, where both $P(n)$ and $Q(n)$ are polynomials, then the simple value-updating algorithm will find an ϵ-approximation of the solution in time* $\mathsf{poly}(n, \frac{1}{\epsilon})$.

11 Some Open Problems and interesting ideas

We present some interesting problems and ideas we have collected.

11.1 Reducing natural question to the circuit problem

Say, can we show factorization can be reduced to the circuit problem? Factoring is in NP $\cap$ co-NP anyway. But the intuition is: probably not! Since the circuit is bitwise monotone. But can we still find something interesting?

11.2 Structure of Σ

Remember Σ is the set of the solutions to the circuit. We know it is path-connected, and it is the finite union of simplexes, and it is NOT necessarily convex — thanks to Adam Kalai. Can we say more about that?

11.3 Structure of the pre-images

We know for any point $P \in \Sigma$, $\phi^{-1}(P)$ is closed, since ϕ is a continuous mapping, but is that convex?

The answer is NO!

See the figure above: all three gates are MIN gates, and we converge the circuit in the order A, B, C.

Now set $P_1 = \{A = 0.5, B = 0, C = 1\}$.Obviously the circuit will converge to the minimal solution, or $\langle 0, 0, 0\rangle$.

Then set $P_2 = \{A = 0.5, B = 1, C = 0\}$, and the circuit will also converge to the minimal solution $\langle 0, 0, 0\rangle$.

But if we take the convex combination of P_1 and P_2, say, $P = \frac{1}{2}P_1 + \frac{1}{2}P_2 = \{A = 0.5, B = 0.5, C = 0.5\}$, then the circuit will converge to $\langle 0.5, 0.5, 0.5\rangle$.

Therefore the pre-image of $\langle 0, 0, 0\rangle$ is NOT convex.

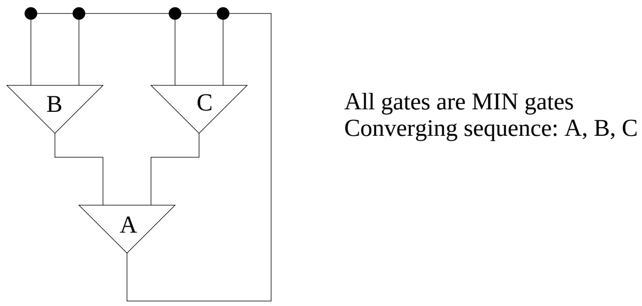

Figure 16: An example for non-convex pre-image

11.4 Flipping MAX/MIN gates

One main constraint to the MAX/MIN/AVG circuit is when you look at the outputs of the gates, they are **monotone** and **piece-wise linear**. So we can't, in general, do a lot of things with them.

However, Rachel has the idea that we can look at the MAX/MIN gates as multiplexers and "read off" the directions as one-bit information: i.e., which input the MAX/MIN gate is outputing. This very precious one-bin information is **not** monotone in general. So the hope is: can we treat the "direction" here as part of the output of the circuit, and can that help us?

So now we are trying to study the "flipping" behavior of the circuit.

Here is one result: we can construct a circuit, and if we let it "run" from **0**, one MIN gate will flip for infinitely many times. Notice the way we "run" the circuit is to update the gates one by one, in a pre-ordered fashion, rather than the converging operation mentioned in section 2. [6] In general this kind of simple "update one gate at a time" method doesn't guarantee convergence, but if we run the circuit from **0**, convergence **is** guaranteed.

Here is the circuit.

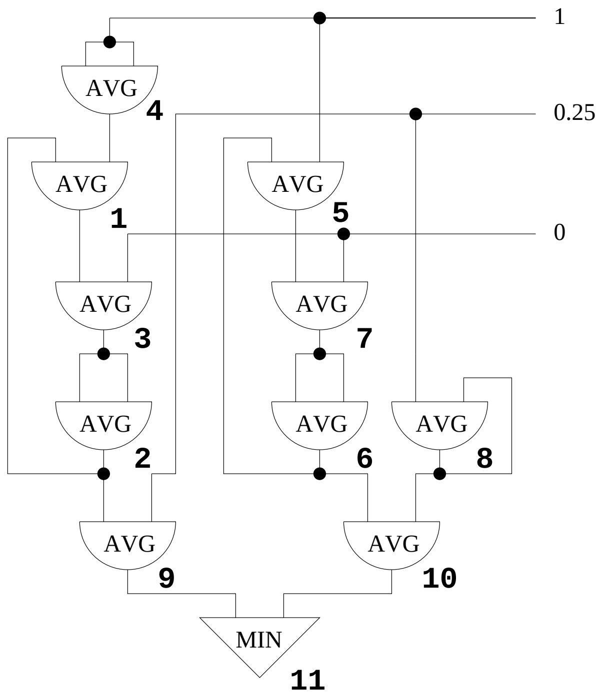

Figure 17: A circuit that flips forever

There is only one MIN gate in the circuit, and all the rest gates are AVG gates. On the right corner of each gate, a number indicates the order the gates get updated. Actually gates 2, 4, and 6 are "delay" gates: they delay the propagation of the values by one step. Gates 1, 2, and 3 is a small unit that will converge to 2/3, 1/3, and 1/3,

[6] I still don't know how to construct a circuit whose MAX or MIN gate flips many times under the converging operation.

respectively. But since there is a delay at gate 2, the converging happens every two rounds. Gates 5, 6, 7 are the same. But gate 4 delays the converging of gates 1, 2, 3 by one step, and thus the output of gate 2 is one step later than gate 6, and thus gate 6 is always greater than or equal to gate 2.

Gate 8 acts as a "perturbation": it starts at 0 and converges to 0.25 gradually. So gate 6 always dominates gate 2, but gate 8 is always smaller than 0.25. Taking the average of gate 2 and 0.25, and the average of gate 6 and gate 8, we got two values that interleaves each other: that are gate 9 and gate 10. So gate 11 flips in every round, and it will flip forever.

We can replace 0.25 by any real number between 0 and 0.5.

Now we have more questions to ask:

- Can we build a circuit that will flip even under the converging operation described in section 2, or under the Blum-converging operation?
- Can we build a clock of n bits with polynomially many gates? A clock of n bits has n MAX (or MIN) gates, and the first MAX flips every round, the second MAX flips every 2 rounds, ... the i-th MAX flips every 2^{i-1} rounds, and the last MAX flips every 2^{n-1} rounds. This is due to M. Blum.
- Can we encode any naturally hard problem into the circuit and use a polynomial-time post-processing algorithm to "read off" the directions of he MAX/MIN gates and give the result? In other words, can the direction of the MAX/MIN gates help?

11.5 Quadratic Programming

Another fantasy towards solving the problem is: since Linear Programming isn't strong enough, can quadratic programming help?

Notice quadratic forms can describe the system perfectly: say we have a gate with x and y as input, and z and output: $z = g(x, y)$. If g is AVG it is trivial. If g is MIN, then was can write as: $z \leq x$, $z \leq y$, and $(z-x)(z-y) = 0$. Notice under the first two inequalities, we know $(z-x)(z-y) \geq 0$. So 0 is the maximum value that $(z-x)(z-y)$ could obtain. Thus we can sum this $(z-x)(z-y)$ over all MIN and MAX gates and let the sum be zero. The the only possible solution to the quadratic form gives a solution to the circuit.

Now the question is: is such quadratic programming solvable?

11.6 Unique-solution vs. non-unique solution circuits

As we have shown in Corollary 2, a circuit has either a unique solution or uncountably many. Also Condon [C92] showed a conversion from any circuit to a circuit with a unique solution which is close to the MIN solution of the original circuit.

We also have the observation that: for a circuit with a unique solution, if somehow we know the value of a gate in the solution, we can simply change the gate to a constant gate with that value, and the remaining circuit still has a unique solution, since otherwise we can "run" the original circuit on another solution of the new circuit and get another solution for the original.

However, this is not true for circuits with infinitely many solutions: one can change a gate to a constant gate with the value that is in one of the solutions and get a new gate. However, the new circuit might as well have many solutions, and it is not clear how one can transform a solution in the new circuits "back" to a solution in the original one.

Now the question is: does it show that unique-solution circuit is the essential computation model?

11.7 Brouwer fixed-Point Theorem and the power of differential topology

This is due to the observation of John Langford. So there is a **Brouwer Fixed-Point Theorem** that's actually very strong in asserting the fixed-point exists. Here is the theorem:

Theorem 20 (Brouwer Fixed-Point Theorem) *Any smooth map f of the closed unit ball $B^n \subset R^n$ into itself must have a fixed point; that is, $f(x) = x$ for some $x \in B^n$.*

This is cited from [GP74], and the proof is believed to carry to continuous functions as well (a smooth mapping is a differentiable function) — actually the exercise 6 at page 66 shows that. So this theorem instantly says that both MMA-circuits and MMAN-circuits have solutions. Thus this is a very powerful theorem. However the proof is non-constructive and doesn't provide any intuition where to find the fixed-point.

Anyway, the big question is: how useful the differential topology is in this circuit problem?

11.8 Reduction, Reduction, Reduction

Here is a reduction result: given an oracle that can answer the question "is the value of this gate (in the solution) higher than 1/2?", we can solve the circuit in polynomial time. Idea: Binary Search.

Here is another one: if we have the guarantee that all the gates have value either 0 or 1, we can solve the circuit in polynomial time.

Acknowledgement

We need to thank lot of people: Avrim Blum, Anne Condon, Alan Frieze, John Langford, Adam Kalai, Steven Rudich, Dana Scott, Wei Xu, Li Zhang.

References

[C66] Eduard Čech, *Topological Spaces*, Revised edition by Zdeněk Frolík and Miroslav Katětov, Publishing House of the Czechslovak Academy of Sciences and Interscience Publishers, a division of John Wiley & son., 1966.

[C92] Anne Condon. *The Complexity of Stochastic Games*, Information and Computation, vol. 96, No. 2, February 1992, page 203-224.

[C93] Anne Condon. *On Algorithms for Simple Stochastic Games*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 13, edited by Jin-Yi Cai, American Mathematical Society, 1993, pages 51-73.

[GP74] Victor Guillemin, Alan Pollack, *Differential Topology*, Prentice-Hall, 1974.

[J84] Klaus Jänich, *Topology*, Translated from German to English by Silvio Levy, Springer-verlag, 1984.