A Reconfigurable Analog Substrate for Highly Efficient Maximum Flow Computation

Gai Liu and Zhiru Zhang
School of Electrical and Computer Engineering, Cornell University, Ithaca, NY
{gl387, zhiruz}@cornell.edu

Abstract

We present the design and analysis of a novel analog reconfigurable substrate that enables fast and efficient computation of maximum flow on directed graphs. The substrate is composed of memristors and standard analog circuit components, where the on/off states of the crossbar switches encode the graph topology. We show that upon convergence, the steady-state voltages in the circuit capture the solution to the maximum flow problem. We also propose techniques to minimize the impacts of variability and non-ideal circuit components on the solution quality, enabling practical implementation of the proposed substrate. Our performance evaluation indicates two to three orders of magnitude improvements in speed and energy efficiency compared to a standard CPU implementation. In the last part of this report, we also discuss the major limitations of the current design, and suggest promising research directions.

1. Introduction

Overcoming the performance and efficiency limitations of traditional von Neumann architecture has been one of the major themes in the recent computing research. Exploring novel architectures and computing paradigms becomes even more imperative as power-limited technology scaling fails to offer the same performance improvement that we have seen in the past few decades [14]. This trend is motivating researchers to investigate new technologies and alternative models of computations to provide the next wave of major improvements in computing. An active direction of research seeks to improve performance and energy efficiency through domain-/application-specific hardware customization using specialized accelerators such as ASICs, FPGAs, and GPGPUs [9]. Parallel to this direction, more disruptive computing paradigms are being examined, including approximate computing [20, 29], neuromorphic computing [33], and quantum computing [26]. These disruptive computing paradigms are expected to achieve significant improvements in performance and energy efficiency for certain domains of applications compared to the traditional approaches.

In this work, we investigate using a physics-based analog computing system as an unconventional approach to solving an important optimization problem — the maximum flow (max-flow) problem. The max-flow problem is one of the most pervasive optimization problems in applications such as transportation [38], Internet traffic scheduling [40], VLSI CAD [10], and many other emerging applications including computer vision [6] and data mining [15]. These emerging applications often deal with large-scale graphs, which need to be analyzed under ever-demanding performance and energy efficiency requirements. Needless to say, a new method that solves the max-flow problem

efficiently in both time and energy consumption would be of great interest to the CAD community as well as the field of computing in general.

However, the max-flow problem has been shown to be computationally demanding [19] and difficult to parallelize [18]. We take a complementary approach to solving the max-flow problem by constructing carefully engineered electrical circuit and mapping the steady-state behavior of the circuit to the solution of the corresponding max-flow problem. We propose a reconfigurable analog substrate composed of traditional analog devices including resistors, diodes and operational amplifiers (op-amps), as well as an emerging device called memristor. The substrate encodes graph topology using the on/off states of the memristor switches, and leverages Kirchhoff's circuit laws to enforce flow constraints, which are represented in the form of circuit node voltage values. The max-flow objective, defined as the net flow out of the source vertex, is then driven to its maximum value. Upon convergence, the steady-state node voltages represent the solution to the max-flow problem.

This work is inspired by recent proposals on solving optimization problems using analog circuits [42, 41, 24] and approximate algorithms for the max-flow problem [8, 3, 32]. In particular, several key circuit structures of our substrate build on Vichik and Borrelli's seminal work on constructing problem-specific analog circuits to solve linear and quadratic programs [42]. Our work differs from [42] by specializing the analog circuits for the max-flow problem, which essentially is a more restricted linear program. In addition, our analog substrate is programmable and capable of solving different instances of the max-flow problem by reconfiguring the electrical properties of the underlying circuit components. More specifically, our main technical contributions are as follows:

1. We present a reconfigurable analog substrate that efficiently solves max-flow problems, and show that the proposed substrate optimally solves the max-flow problem under ideal assumptions.
2. We study the convergence time and power consumption of the reconfigurable analog substrate, and demonstrate promising results in performance and energy efficiency.
3. We analyze the influence of non-ideal circuit components and process variation on the solution quality, and propose techniques to mitigate the impact of these non-ideal factors.

This report extends our previous paper in [30] by including more elaborate discussions on the comparison with related work, major limitations of our current proposal, as well as the potential research directions for addressing these limitations. The remainder of the report is structured as follows: Section 2 describes the circuit mechanism of our substrate; Section 3 introduces the reconfigurable crossbar architecture; Section 4 provides the implementation details of the substrate; Section 5 compares the performance of our proposed analog substrate with the software implementation on a CPU; Section 6 further discusses the circuit dynamic behaviors and points out the current limitations of this work and future directions; Related work is discussed in Section 7 followed by conclusions in Section 8.

2. Computing Max-flow using Analog Circuit

The max-flow problem we are interested in solving is stated as follows: Given a directed graph[1] $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges, we distinguish two vertices, a source vertex s and sink vertex t, and assign each edge e a nonzero integral capacity c_e. A $s - t$ flow is a function $f : E \rightarrow \mathbb{R}$ such that for every vertex n_i other than s and t, the flow coming into n_i is equal to the flow going out of n_i, and $0 \leq f(e) \leq c_e$ for all $e \in E$. The value $|f|$ of a $s - t$ flow is defined to be

[1] Max-flow problem on an undirected graph can be converted to an equivalent problem on a directed graph by allocating two opposite edges with the same capacity as the edge in the original undirected graph.

the net flow out of the source vertex. The max-flow problem asks for finding a feasible $s - t$ flow in G of the maximum value.

The circuit used to compute max-flow is constructed in a direct mapping manner, i.e., for every edge, there is a circuit component to ensure edge capacity constraint; and for every node, there is another circuit component to enforce flow conservation constraint; there is also a circuit component for realizing the functionality of max-flow objective function. It is worth noting that several important circuit structures are based on the proposal in [42]. While the analog circuits in [42] are designed for generic linear and quadratic programming, ours are customized to fit the max-flow problem for enabling reconfigurability and reduced circuit complexity.

2.1 Edge Capacity Constraint

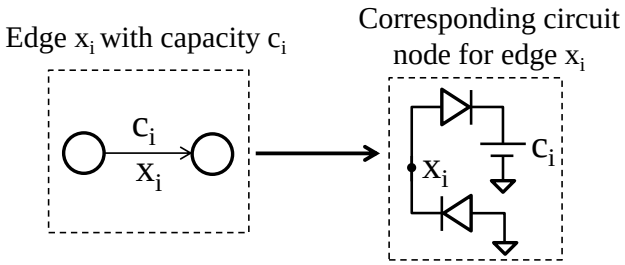


Figure 1: Circuit component for edge capacity constraint.

For each edge capacity constraints on edge x_i, we create a circuit component composed of two diodes and one voltage source, as shown in Figure 1, where the capacity on edge x_i is c_i. One of the two diodes is directly connected to the circuit node x_i, with its anode connected to ground. Ideally, the diode will turn on whenever the voltage on circuit node x_i is below zero[2]. Since the ideal diode in on-state behaves like an ideal wire, the voltage on circuit node x_i will always be non-negative. Similarly, the other diode and a voltage source of value c_i enforce that the circuit node voltage at x_i never exceeds c_i. Thus we have the following relation

$$0 \leq V_{x_i} \leq c_i \tag{1}$$

for every node x_i that corresponds to an edge x_i in the original graph, where V_{x_i} is the voltage value on circuit node x_i, and c_i is the corresponding edge capacity on edge x_i.

2.2 Flow Conservation Constraint

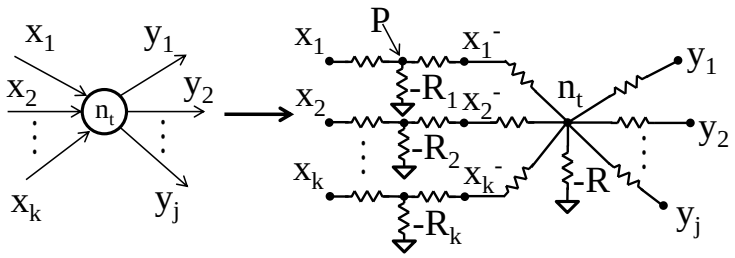


Figure 2: Circuit component for flow conservation constraint.

[2] Note that in practice we need to account for the turn-on voltage of the diode by adjusting the values of the voltage sources.

Flow conservation constraint is enforced using the circuit in Figure 2. For each outgoing edge y_i ($i = 1, 2, \ldots, j$) of a vertex n_t in the original graph, the corresponding circuit node y_i with voltage value V_{y_i} is connected to node n_t via a resistor; while for each incoming edge $x_i (i = 1, 2, \ldots, k)$ of the vertex n_t, the corresponding node of negated voltage value $V_{x_i^-}$ is connected to node n_t. All positive resistors have the same resistance value r. Negative resistors $-R_1, \ldots, -R_k$ all have the same resistance value of $-r/2$, and the negative resistor connected to n_t has resistance value $R = \frac{r}{N}$, where $N = j + k$. Compared to the equality constraint circuit in [42], the main simplification is that we set all the positive resistances to the same value r, since the flow conservation constraints all have 1 or -1 coefficients.

In the following, we show that the circuit in Figure 2 indeed enforces flow conservation constraint on node n_t using Kirchhoff's current law (KCL). If we denote the voltage value on node P in Figure 2 as V_P, we have the following relation according to KCL:

$$\frac{V_{x_1} - V_P}{r} + \frac{V_{x_1^-} - V_P}{r} = \frac{V_P}{-R_1} \tag{2}$$

Since $-R_1 = -r/2$, we have

$$V_{x_1} = -V_{x_1^-} \tag{3}$$

Similarly, we have $V_{x_i} = -V_{x_i^-}$ for all $i = 1, 2, \ldots, k$.

Furthermore, based on KCL we also have the following relation on node n_t:

$$\sum_{i=1}^{k} \frac{V_{x_i^-} - V_{n_t}}{r} + \sum_{i=1}^{j} \frac{V_{y_i} - V_{n_t}}{r} = \frac{V_{n_t}}{-R} \tag{4}$$

Plugging $R = \frac{r}{k+j}$ into Equation (4):

$$\sum_{i=1}^{k} V_{x_i^-} + \sum_{i=1}^{j} V_{y_i} = 0 \tag{5}$$

Using the result from Equation (3), we obtain the following relation that satisfies flow conservation constraint: $\sum_{i=1}^{k} V_{x_i} = \sum_{i=1}^{j} V_{y_i}$.

2.3 Max-flow Objective

The objective function is implemented using the circuit in Figure 3, which is also similar to the one used in [42], but with the simplification that all positive resistors have the same resistance value of r. Intuitively, the voltage source V_{flow} will drive the node voltages as large as possible, subject to the capacity and conservation constraints. As a result, the steady-state voltages of the circuit nodes reflect the maximum possible flow through the network that does not violate the requirements of the constraints.

In the following, we derive the steady-state solution of the circuit using basic circuit theorems, and show that the steady-state node voltages represent the solution for the max-flow problem. We will first prove that the value of $s - t$ flow will strictly increase as V_{flow} increases by showing that the constructed resistor network is passive. Combining this fact with the flow conservation constraint and

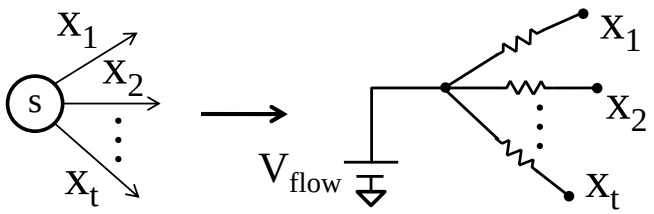
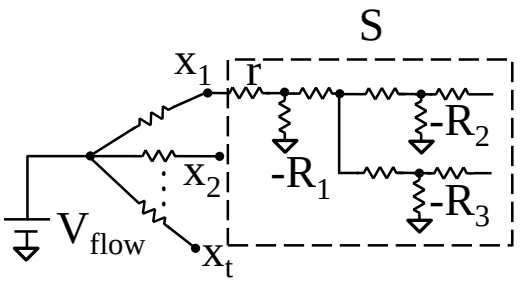
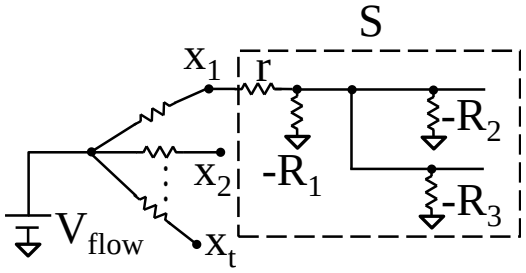


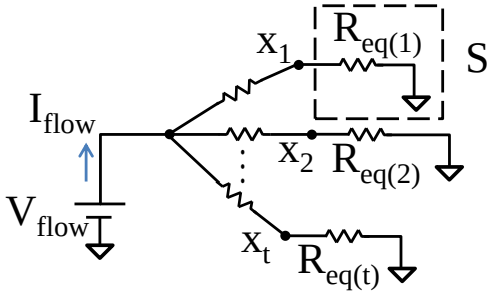
Figure 3: Circuit component for implementing max-flow objective.



(a) Original resistor network.



(b) Resistor network after reducing the resistance of certain positive resistors to zero.



(c) Lumped resistor network with equivalent resistance.

Figure 4: Replacing resistor network with equivalent resistance.

edge capacity constraint derived in Sections 2.1 and 2.2, we can show the optimality of the circuit solution.

A formal proof of optimality for general LP and QP analog circuit has been proposed in [42]. Their proof of optimality also applies to our max-flow circuit. However, since the max-flow problem is a

restricted form of linear programs (i.e., all the coefficients in the objective function and the linear constraints are one, and all the inequality constraints only involve a single unknown variable), we can devise a more concise and intuitive proof of optimality for the max-flow problem, as detailed below.

We first show that the equivalent resistance of the resistor network in the max-flow circuit is positive. Figure 4a shows a generic circuit for solving the max-flow problem, where nodes x_1 to x_t are the subset of nodes that are directly connected to V_{flow}. In Figure 4a, we mark the resistor that directly connects to node x_1 with its resistance value r. We also list all the negative resistors in the network as $-R_1, -R_2, \ldots, -R_k$. Note that according to the construction of the circuit, we have $|-R_i| < r$ for all the negative resistors. Now we find the equivalent resistance of the resistor subnetwork S to the right of node x_1, shown in the dashed box in Figure 4a. If we set all the resistance of the positive resistors except r in S to be zero, the resulting resistor network consists only of the resistors shown in Figure 4b. Since each of the resistance $|R_i|$ is strictly less than r, all the parallel negative resistors together also have an equivalent resistance that is less than r. As a result, the equivalent resistance of S, which is the sum of r and the equivalent resistance of these negative resistors, is always positive. Figure 4c shows the equivalent circuit of Figure 4a, where $R_{eq(i)} > 0$ for all $i = 1, 2, \ldots, t$.

Each circuit branch in Figure 4c is a basic voltage divider. Since all the remaining positive resistors have a resistance value of r, we have $V_{x_i} = V_{flow}\frac{R_{eq(i)}}{R_{eq(i)}+r}$ for all $i = 1, 2, \ldots, t$. Note that we have shown $R_{eq(i)} > 0$ for all $i = 1, 2, \ldots, t$, thus the steady-state solution V_{x_i} will strictly increase as V_{flow} increases, as long as edge capacity constraints and flow conservation constraints are satisfied. Summing up all the node voltages, we have

$$\begin{aligned}\sum_{i=1}^{t} V_{x_i} &= V_{flow}\sum_{i=1}^{t}\frac{R_{eq(i)}}{R_{eq(i)}+r} = V_{flow}\sum_{i=1}^{t}(1-\frac{r}{R_{eq(i)}+r}) \\ &= tV_{flow} - rI_{flow} \triangleq J\end{aligned} \tag{6}$$

J in Equation (6) plays the role of objective function in the max-flow problem. V_{flow} in the circuit is set to a high voltage value, so that V_{flow} will try to maximize the node voltages in the circuit. Combining this result with the derived edge capacity constraint and flow conservation constraint, we have:

$$J = tV_{flow} - rI_{flow} = \sum_{i=1}^{t} V_{x_i} \tag{7a}$$

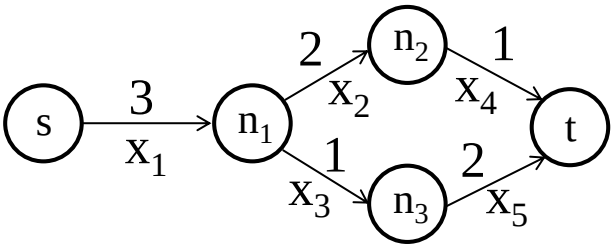
$$\sum_{i=1}^{k} V_{x_i} = \sum_{i=1}^{j} V_{y_i} \tag{7b}$$

$$0 \le V_{x_i} \le c_i \tag{7c}$$

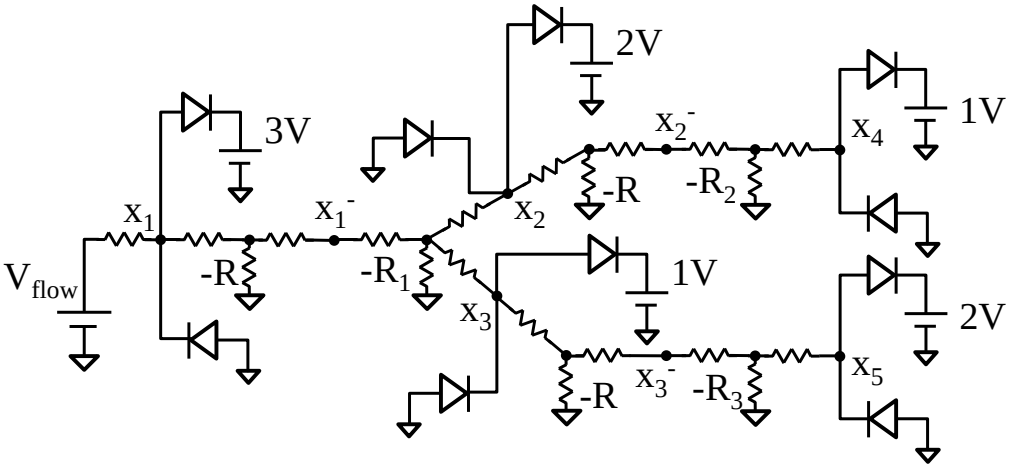
where Equation (7b) holds for all vertices with incoming edges x_i and outgoing edges y_i, and Equation (7c) holds for all edges in the graph. Equation (7) is identical to the original max-flow problem. Thus we conclude that the circuit solution is equivalent to the solution of the corresponding max-flow problem.

2.4 An Example

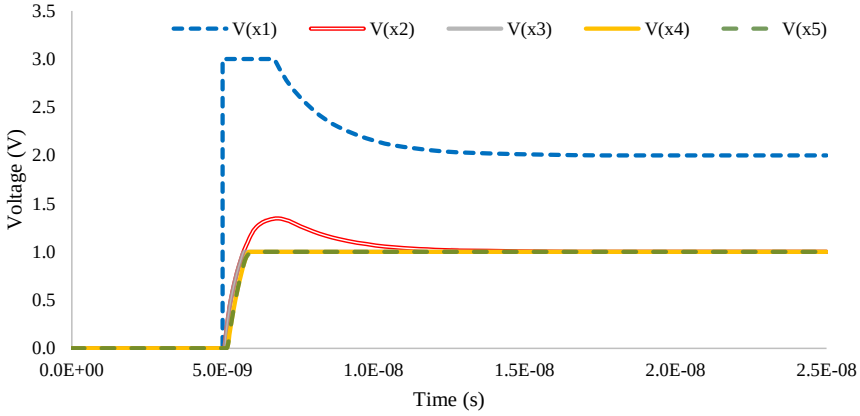
Figure 5b shows an example circuit that solves a max-flow problem instance in Figure 5a. A step function is first applied to V_{flow} that drives V_{x_1} to its maximum value $3V$. Then the circuit components



(a) An example max-flow problem.



(b) The corresponding circuit.



(c) Waveform of the node voltages.

Figure 5: An example of solving max-flow using analog circuit.

implementing flow conservation constraints start working together to bring the other nodes to steady state. V_{x_3} and V_{x_4} will reach their maximum values of $1V$, corresponding to the case that flows on edge e_{x_3} and e_{x_4} reach their maximum value of 1. Then V_{x_1} and V_{x_2} in return bring V_{x_1} to its final value of $2V$. Figure 5c shows the waveform of the node voltages as a function of time.

3. Reconfigurable Crossbar Architecture

Constructing a custom circuit for each graph instance is generally impractical as fabricating a custom circuit is both expensive and time consuming. We propose to construct a reconfigurable substrate

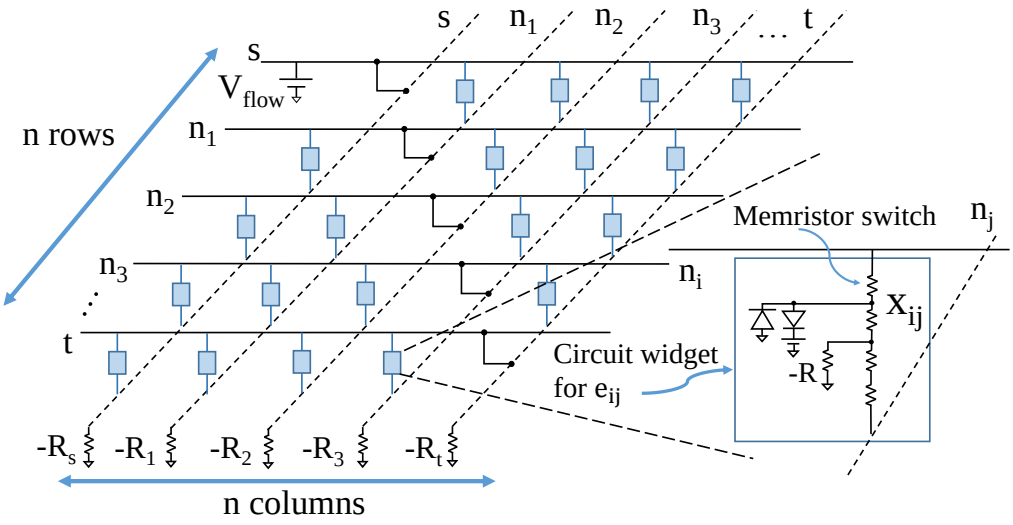


Figure 6: Reconfigurable crossbar architecture.

that can be configured to encode different graph topologies, enabling the solution of various problem instances using a single substrate. We propose to use memristors as the reconfigurable switches in our substrate. The memristor [22] is a two terminal device with a metal-insulator-metal sandwiched structure whose resistance (called memristance) can be modulated by external stimulus. Memristor can be switched between the high-resistance state (HRS) and the low-resistance state (LRS) by external voltage stimulus, and retains its resistance value when external stimulus is removed or below a certain threshold. Our reason for choosing memristors as switches is twofold: (1) memristor-based switches are more area-efficient than the traditional SRAM-based switches; (2) since resistors are widely used on our substrate, we can directly replace the resistors with memristors so that one single memristor acts both as a switch and as a resistor. Namely, A memristor in HRS represents a disconnected switch; A memristor in LRS represents both a connected switch and a resistor whose resistance is equal to the LRS memristance. Additionally, using LRS memristors as normal resistors enables the fine-tuning of memristance in LRS, which helps mitigate the impact of process variation and parasitic resistance, as will be discussed in Section 4.3.2.

Our substrate is composed of an $n \times n$ crossbar as shown in Figure 6. At each intersection (n_i, n_j), there is a small circuit widget connecting into the crossbar through a memristor switch. This circuit widget represents an edge x_{ij}. Essentially, we can view the crossbar as a physical representation of the adjacency matrix of the graph. If edge (i, j) is present in the graph, the memristor switch at position (n_i, n_j) will be set to LRS. Otherwise, the memristor switch at position (n_i, n_j) will be set to HRS. When properly configured, the crossbar architecture implements the circuit described in Section 2. We implement the max-flow objective using the first row of the crossbar, where the memristor switch at position (s, n_i) will be turned on if and only if edge (s, i) is present in the graph. We enforce the edge capacity constraint using two diodes at each intersection, and honor the flow conservation constraint by connecting column n_i with all the edges incident to n_i. Figure 7 shows the mapping of the example circuit to the crossbar. By examining the circuit widgets at the intersections, we can verify that the crossbar correctly implement the circuit in Figure 5b.[3]

[3] Since there is no outgoing edge from sink t, the negative resistors in the circuit widgets of column t can be omitted without affecting the correctness of the circuit.

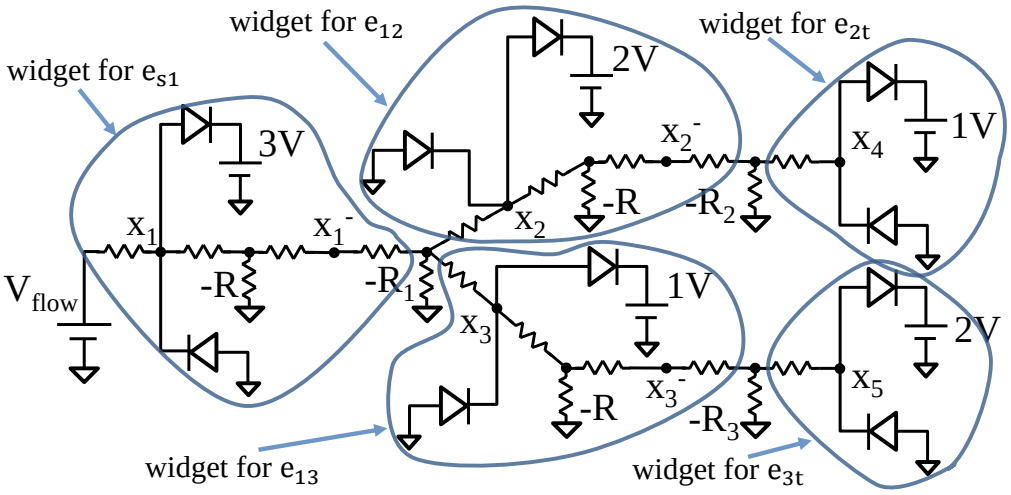


Figure 7: Mapping the example circuit to the crossbar.

3.1 Configuring the Crossbar

In the configuration stage, we program the memristors in the crossbar to the desired resistance state using voltage pulses. V_{flow} is set to zero in this stage. The programming stage takes n cycles to complete, one cycle for each row. At the i^{th} programming cycle, row i is activated by setting the row wire to a low voltage V_{low}, while keeping all the other rows at 0V. For every memristor that needs to be set to LRS, the corresponding column will be set to a high voltage V_{high}, while keeping all the other columns at 0V. By setting the voltage difference $(V_{high} - V_{low})$ greater than the memristor threshold voltage, the desired memristors will be set to LRS, while all the other memristors stay in HRS. After n cycles of configuration, all the memristor switches are set to the desired state.

3.2 Computing Max-flow on the Crossbar

At the beginning of the computing stage, a step function is applied to V_{flow}, and the circuit starts converging to steady state. To avoid the complexity of detecting steady-state condition, we run the circuit for a period of time t. t should be long enough to ensure that the circuit converges, which can be determined by profiling the worst-case execution time of the substrate. After convergence, we measure the current through the voltage source V_{flow} and calculate the objective value (i.e. value of max-flow) based on Equation (7a).

4. Implementation

We have shown that the proposed substrate generates optimal solution for max-flow problems under several assumptions: (1) one voltage source for each edge with exact voltage level; (2) ideal negative resistors and diodes; (3) no process variations or parasitic resistance. In this section, we analyze the impact of these assumptions on the solution quality, and propose techniques that enable a practical implementation of the proposed substrate.

4.1 Voltage Level Quantization

In practice, it is too expensive to use one distinct voltage source on the substrate for each edge. Also, the output voltage level of a voltage source cannot be arbitrarily large. To address these two issues, we propose the following quantization scheme that maps the edge capacities to a set of discrete voltage levels. Given a supply voltage of value V_{dd}, we uniformly divide the voltage interval $[0, V_{dd}]$ into N

voltage levels: $\mathbf{V} = \{\frac{1}{N}V_{dd}, \frac{2}{N}V_{dd}, \ldots, \frac{N-1}{N}V_{dd}, V_{dd}\}$. For each voltage level in $\mathbf{V}$, we construct one voltage source with the corresponding voltage value. One voltage source will be used for multiple edges if these edges share the same voltage level after quantization. The quantization function $\mathbf{Q}$ uniformly maps the edge capacities of a max-flow instance into the voltage levels in set $\mathbf{V}$. Specifically, given an edge capacity x, $\mathbf{Q}$ maps x to one of the voltage levels in $\mathbf{V}$: $\mathbf{Q}(\mathbf{x}) = \frac{\lfloor (x/C)N \rfloor}{N} V_{dd}$, where C is the largest edge capacity of the max-flow instance. The solution $\mathbf{Y} = \{V_{x_1}, V_{x_2}, \ldots, V_{x_m}\}$ obtained from the circuit is then mapped back to the interval $[0, C]$: $\tilde{\mathbf{Y}} = \mathbf{Y} \times \frac{V_{dd}}{C}$. $\tilde{\mathbf{Y}}$ is an approximate solution of the original problem with quantization errors. The worst-case quantization error e of the flow on one edge is bounded by the quantization step: $e = \frac{C}{N}$ with C being the largest edge capacity and N being the number of voltage levels. Note that the choice of N provides a trade-off between accuracy and cost— increasing N reduces the worst-case quantization error, but also increases circuit complexity and cost. Figure 8 illustrates the application of voltage quantization on the same graph in Figure 5a with $N = 20$ and $V_{dd} = 1V$, the result shows a 5% deviation in the final flow value $|f|$.

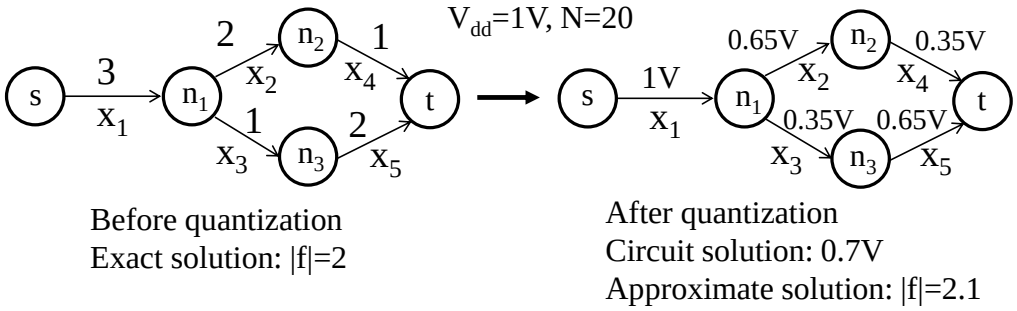


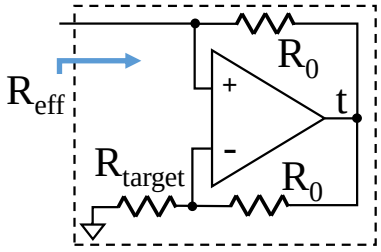
Figure 8: An example of voltage level quantization.

4.2 Negative Resistors

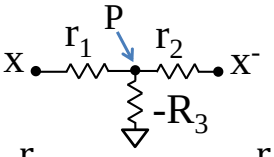
We can view a negative resistor as a current-controlled voltage source with the transresistance equal to $-R$, which can be implemented using an op-amp, as shown in Figure 9a. The precision of the negative resistor is determined by the open loop gain A of the op-amp. To derive the precision of the negative resistor, we calculate the effective resistance R_{eff} of the negative resistor as $R_{eff} = -(1 + \frac{1}{A}\frac{R_0}{R_{target}})R_{target}$, where $\frac{R_0}{R_{target}}$ is a constant that is close to 1. Thus the precision of the negative resistor is inversely proportional to the gain A of the op-amp. Since modern op-amps can easily achieve gain $A > 1000$, indicating a precision of $\pm 0.1\%$, we conclude that the implementation of op-amp has negligible impact on the precision of the negative resistor.

4.3 Process Variation and Parasitic Resistance

The actual resistance of the resistors deviates from their nominal values due to unavoidable process variation and parasitic effects, which degrade the solution quality. In this section, we first make an important observation that the solution quality depends only on the ratio of the resistance values instead of the absolution resistance values. This enables the use of layout matching techniques to minimize variations. We will also discuss post-fabrication resistance tuning to further reduce the impact of resistance variation.



(a) Negative resistor using operational amplifier.



$$V_{x^-} = -\frac{r_2}{r_1}V_x + V_P(1 + \frac{r_2}{r_1} - \frac{r_2}{R_3})$$

(b) Resistance tuning circuit.

Figure 9: Negative resistor and the resistance tuning circuit.

4.3.1 Resistance Matching

Although most integrated resistors have tolerances of ± 20 to 30%, the tolerance of the ratio between two resistors can be controlled to be better than $\pm 1\%$, and in many cases, to be within $\pm 0.1\%$ [21]. We have shown in Section 2 that the circuit node voltages are only functions of ratios of resistances. Since the circuit node voltages in steady state represent the flow value, we are encouraged to discover that the solution quality is also only a function of resistance ratios, independent of the absolute resistance values. Consequently, the substrate is expected to be highly insensitive to process variation. However, resistance compensation techniques are still needed to counter the other sources of variations such as parasitic resistance.

4.3.2 Resistance Tuning

The fact that all the resistances in the circuit are realized by memristors makes it possible to conduct post-fabrication fine-grained resistance tuning [28]. This is especially useful to minimize the influence of parasitic resistance. We outline the tuning process as follows.

To begin with, we configure the substrate to implement the tuning circuit shown in Figure 9b, which is a simple circuit that enforces $V_x = -V_{x^-}$. For each of the tuning circuit, we first set $V_x = 0$ and modulate R_3 until $V_{x^-} = 0$. This step ensures $\frac{1}{R_3} = \frac{1}{r_1} + \frac{1}{r_2}$. We then set $V_x = 1V$, and collectively change the values of r_1 and r_2 until $V_{x^-} = -1V$. These two steps can be iterated for a few times for better tuning precision. We note that in principle the tuning process has to be done only once right after fabrication, thus no performance overhead is incurred afterwards. However, since the memristance may slowly drift over a long period of time, the tuning process can be repeated based on the endurance of the memristors.

5. Performance Evaluation

We construct a 1000×1000 substrate in circuit-level netlist and simulate the substrate in SPICE [35]. The detailed parameters used in the simulation are listed in Table 1. The parameters for memristors

and op-amps are set to typical values from recent literature [5, 13]. We will first evaluate the execution time for various benchmarks, then estimate the power consumption of the substrate using an analytical method.

Table 1: Design parameters for the max-flow computing substrate.

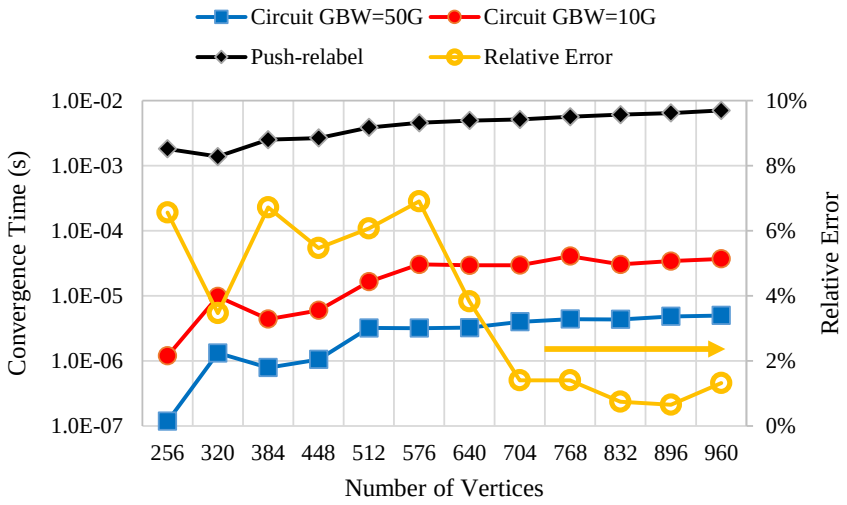
Memristor LRS resistance ($k\Omega$)	10
Memristor HRS resistance ($k\Omega$)	1000
Objective function voltage V_{flow} (V)	3
Open loop gain of op-amp	1×10^4
Gain-bandwidth product of op-amp (GHz)	10 to 50
Number of columns in the crossbar	1000
Number of rows in the crossbar	1000
Number of voltage levels	20

5.1 Convergence Time

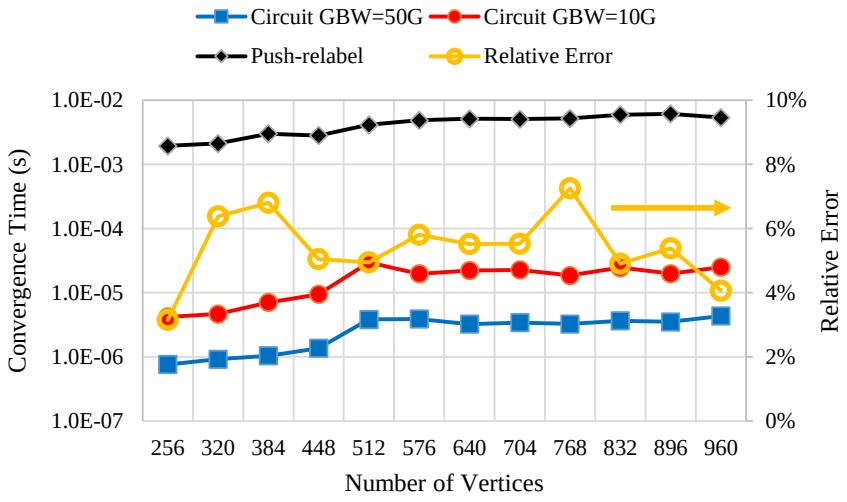
The convergence time of the substrate is measured as the time interval between the rising edge of V_{flow} and the timestamp when the flow value is within 0.1% of the final value. To model the effect of parasitic capacitances, we add a parasitic capacitance of 20fF to each circuit net. We compare the convergence time against the execution time of the widely used push-relabel algorithm running on a server with 3GHz Intel Xeon processor and 16GB of RAM. The push-relabel algorithm is compiled using GCC 4.4.7 with -$O3$ optimization option. When measuring execution time on CPU, we exclude the time taken to read input file and the time taken to generate internal data structure for fair comparison. We use R-MAT synthetic graph generator [7] to generate both dense ($|E| \propto |V|^2$) and sparse ($|E| \propto |V|$) graphs. The number of nodes in the generated graph ranges from 200 to 1000, and the number of edges ranges from 500 to 8000. We plot the convergence time of the substrate and the execution time of the push-relabel algorithm in Figure 10. Using op-amps with 10G gain-bandwidth product (GBW), our substrate achieves a range of speedups between $150X$ and $1500X$ compared to the software implementation. If we further increase the GBW to 50G, we can achieve an additional $10X$ speedup. Such dramatic speedup is not too surprising as the circuit operates vastly different and more efficient than the CPU — instead of sequentially executing millions of instructions, the circuit converges to the optimal solution in a massively concurrent fashion, with physics being the driving force, e.g., pulling up and pulling down of the node voltages. We also plot the relative error of the circuit solution compared to the optimal solution in Figure 10. The deviations from the optimal solutions are due to the quantized voltage levels and non-ideal circuit components. We can observe that in all cases the relative error is within 8%, with an average relative error of 3.7% for dense graphs and 5.4% relative error for sparse graphs.

5.2 Analysis of Power and Energy Consumption

We analytically model the power consumption of the substrate with the goal of showing the general trend of power consumption as a function of circuit size. The model provides insights on how a given power budget impacts the overall design. Here we only consider the power consumption of the substrate itself, thus the power consumption of the analog-digital conversion between the digital computer and the analog substrate is not included. The majority of power is consumed by op-amps



(a) Dense graph: $|V| \propto |E|^2$.



(b) Sparse graph: $|V| \propto |E|$.

Figure 10: Convergence time of synthetic benchmarks.

and resistors. For resistors, we observe that it is possible to proportionally scale up the resistor values without influencing the correctness of the solution (Section 4.3.1). Scaling up the resistance reduces the average current through the resistors, thus reducing the power consumption of the resistors. Consequently, the op-amps are the major source of power consumption. According to the circuit structure, one op-amp is needed for each edge that is present in the graph.[4] Additionally, one op-amp is needed for each node to enforce flow conservation constraint. Thus the power consumption for solving a graph of $|V|$ nodes and $|E|$ edges is roughly $(|E| + |V|)P_{amp}$, where P_{amp} is the average power consumption of an op-amp.

The maximum graph size that a substrate can support is limited by the power budget. Given a power budget of P_{tot} and assuming $|V| \ll |E|$, then the substrate can support roughly up to P_{tot}/P_{amp} number of edges. Assuming a 1V supply voltage and an average current of 500μA for the op-amp, which is typical for the 32nm technology node, P_{amp} evaluates to 500μW. Given a power budget of

[4] Note that if an edge is not present in the graph, the corresponding op-amp can be power gated, thus consuming no power.

$P_{tot} = 5$W, which is typical for an embedded system, we can accommodate up to 10^4 active edges on the substrate. Increasing the power budget to $P_{tot} = 150$W, a typical value for high-end servers, the number of active edges that the substrate can support boosts to 3×10^5. This estimation shows that the substrate is indeed promising in solving large-scale max-flow problems. Although in this case power consumption of the substrate is comparable to CPUs, the energy efficiency is drastically higher because our substrate is around $150X$ to $1500X$ faster than the CPUs.

6. Discussions and Future Directions

In this section, we shall discuss the major limitations of our current work, as well as the potential directions to mitigate some of these limitation. Specifically, we propose techniques to handle large graphs, including clustered architectures, the dual min-cut formulation, and graph decomposition. We also discuss the dynamic behavior of the circuit and propose to investigate its asymptotic rate of convergence.

6.1 Limitations of Current Work

While our simulation results show that the proposed analog substrate can achieve substantial speedup over traditional CPU execution, several major challenges still remain that limit the scalability and accuracy of the current proposal:

1. Handling large graphs – Our current implementation cannot handle large graphs whose corresponding circuits exceed the area constraint of the analog circuit. We propose techniques that could improve the scalability of our approach through clustered architectures (Section 6.2) and graph decomposition (Section 6.4).
2. Asymptotic behavior of the circuit – Since our current performance evaluation is based on relatively small graphs, the asymptotic behavior of the circuit remains to be explored. To this end, we provide insights and initial studies on the dynamic circuit behavior in Section 6.5.
3. Reading out individual flow values – The current implementation can only output the final flow value. Reading out individual flow values on the edges require sensing the voltage values at the internal nodes in the substrate. Additional circuit-level techniques need to be developed to address this limitation.
4. Consideration of wire resistances and temperature effects of the resistors – The wire resistances pose significant challenge to the accuracy of the solution. We envision that techniques in Section 4 can be extended to account for wire resistances. In addition, the analog substrate relies on the precise resistance values of the resistor to generate highly accurate solutions. However, resistance of the resistors is temperature dependent. Compensation techniques need to proposed to mitigate variations due to temperature effects.
5. Digital-analog interface – In this work, we did not consider the interface between digital computer and the analog substrate. ADC and DAC modules need to be carefully designed to maximize system accuracy, as the precision of these modules will have an impact on the final solution quality.

6.2 Clustered Island-Style Architectures

Because of its simple and regular nature, the mesh-based architecture permits very efficient graph to substrate mapping. However, the utilization of the circuit components in the mesh will be low for sparse graphs. To exploit graph sparsity and further improve the scalability of the substrate, we propose to explore clustered architectures, which are structured in a similar fashion to the popular

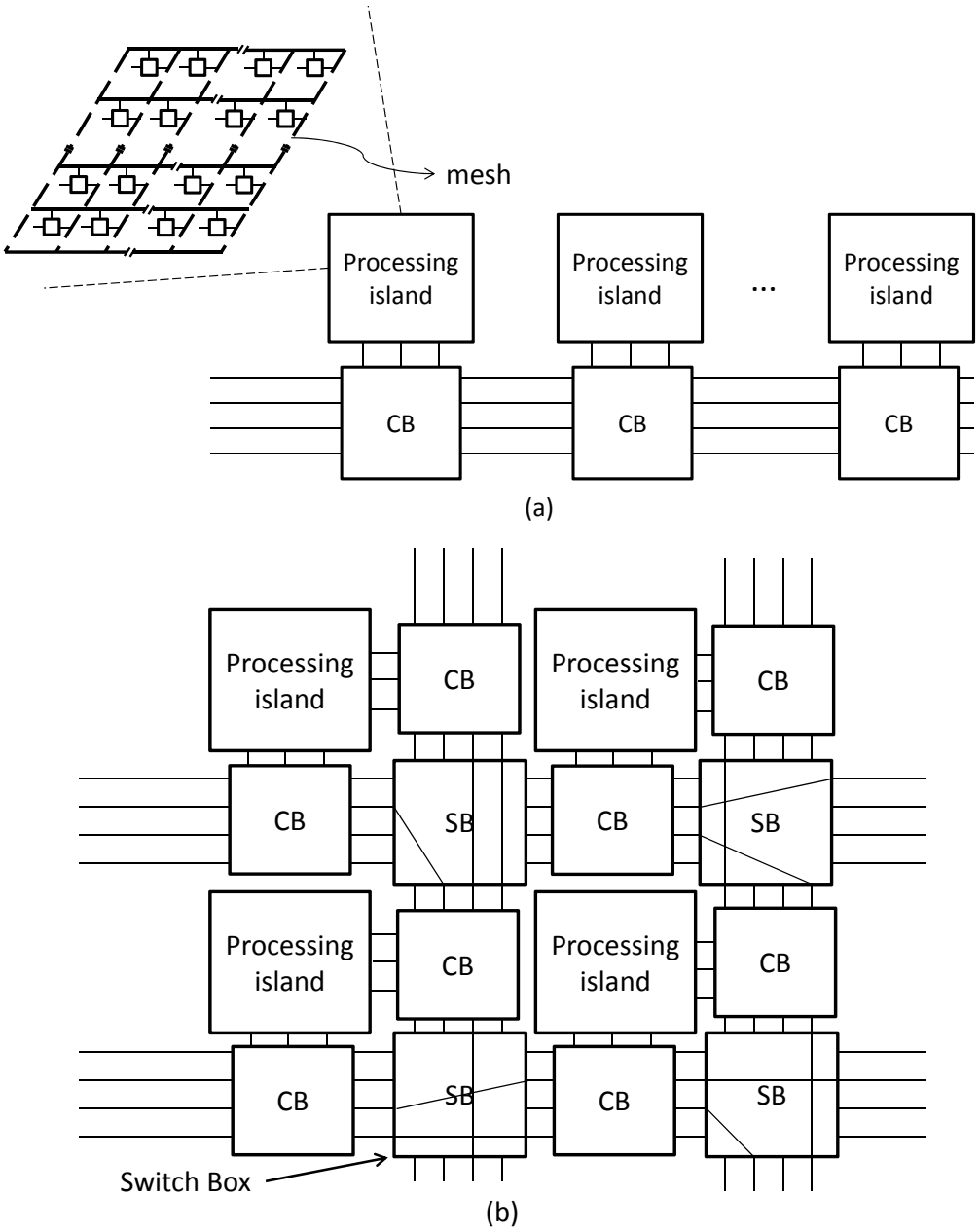


Figure 11: Clustered architectures. (a) one-dimensional structure (CB = connection box); (b) two-dimensional structure (SB = switch box).

island-style field-programmable gate arrays (FPGAs) [4]. More specifically, a clustered architecture includes a collection of mesh-based processing islands which are interconnected by a routing network. For a sparse graph, the highly connected subgraphs will map to separate processing islands, while the connections between these subgraphs will use the routing network.

Figure 11 illustrates two interesting kinds of clustered architectures with one- and two-dimensional routing structure, respectively. These two architectures exhibit different characteristics in flexibility and complexity. We hypothesize that the one-dimensional (1-D) architecture in Figure 11a is more amenable for fast graph mapping but less scalable for large-scale graphs due to limited routing

$$
\begin{aligned}
&\textit{minimize} \sum_{i,j \in V} c_{ij} d_{ij} \\
\text{Subject to: } & d_{ij} - p_i + p_j \geq 0 \;\; (i,j) \in E \\
& p_s - p_t \geq 1 \\
& p_i \geq 0 \quad i \in V \\
& d_{ij} \geq 0 \;\; (i,j) \in E
\end{aligned}
$$

Figure 12: Linear program of the min-cut problem. p_i indicates the partition that vertex i belongs to, d_{ij} indicates if edge (i, j) is in the cut set, and c_{ij} captures the edge capacities.

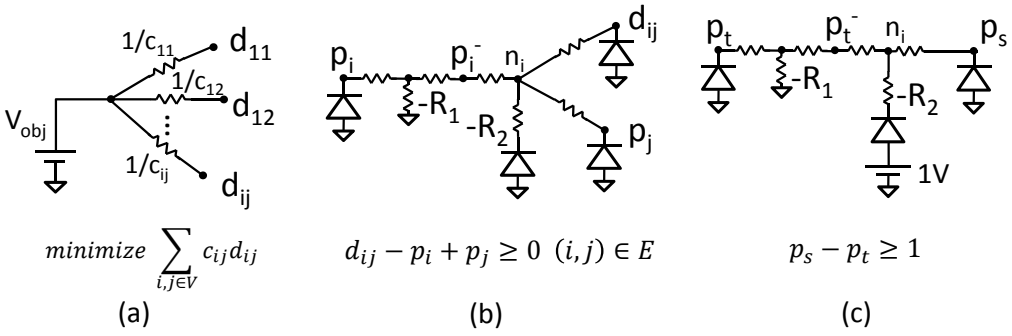


Figure 13: Circuit components for min-cut problem. (a) Objective function. (b) Graph cut constraint. (c) Source/sink constraint.

flexibility. On the other hand, the two-dimensional (2-D) architecture in Figure 11b is potentially more flexible but requires more complex circuits with switch boxes and more sophisticated CAD algorithms for efficient clustering, placement, and routing.

We plan to study these interesting yet intricate trade-offs by exploring various 1-D and 2-D architectures. Furthermore, we propose to explore a set of important architectural parameters to carefully navigate the tension between architectural flexibility and complexity. Examples include the granularity of each island (i.e., local mesh size), homogeneity versus heterogeneity in the island configurations, and hierarchical versus two-level clustered organization.

6.3 The Dual Formulation

Another interesting direction is to investigate the min-cut problem, which is the dual formulation of max-flow. Studying the min-cut problem enables us to utilize existing works on dual decomposition to handle large graphs, as detailed in Section 6.4. Figure 12 shows the linear program of the min-cut problem, which partitions the nodes into two sets such that the sum of the edge capacities across these two sets, is minimized. Figure 13 illustrates how to map a min-cut problem into analog circuits. The underlying mechanism is similar to the one for max-flow discussed in the previous section. The objective function in Figure 13a tries to drive down the node voltages while satisfying the constraints enforced by circuits in Figures 13b and 13c. It is worth noting that we no longer require additional voltage sources for enforcing the edge capacity constraints. Instead, the capacity values are reflected by the objective function in the dual formulation.

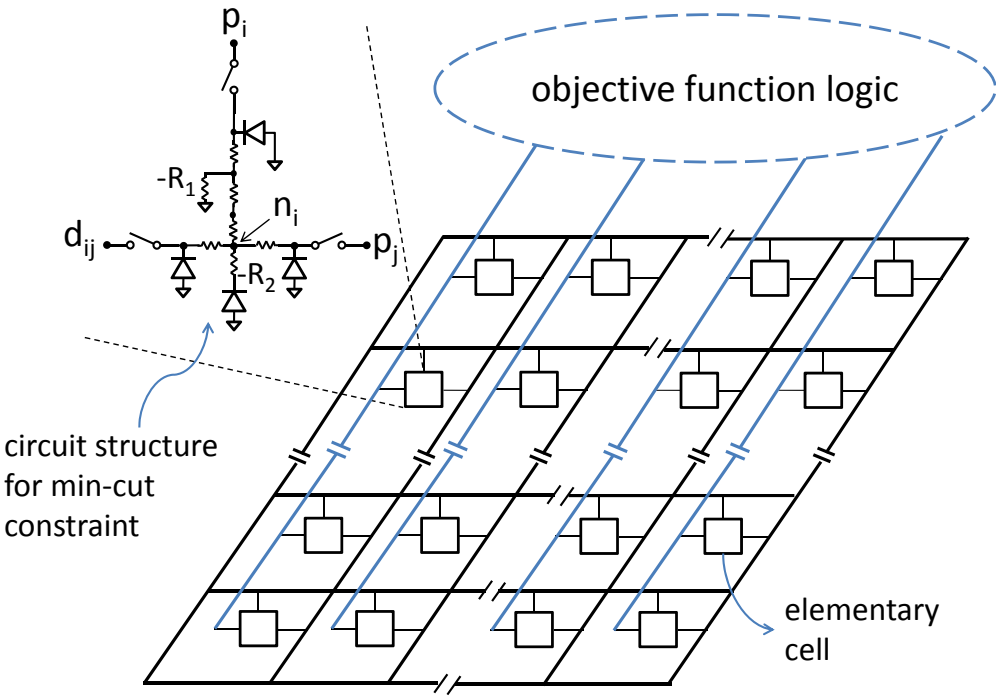


Figure 14: The mesh-based architecture exploiting the dual formulation – Switches in cell (i, j) are closed if the corresponding edge (i, j) is present in the graph.

Leveraging the dual formulation, we propose a mesh-based architecture similar to the one in Section 3, which is shown in Figure 14. This architecture forms an $n \times n$ mesh where each elementary cell in the mesh implements one constraint of the min-cut problem and uses three switches to achieve reconfigurability. Additional circuit is also introduced to implement the objective function using the mechanism shown in Figure 13a. We note that this mesh-based architecture essentially encodes the adjacency matrix of the input graph, which requires $O(n^2)$ elementary cells for any graph with up to n nodes.

6.4 Graph Decomposition

The clustered architecture with min-cut formulation can potentially allow the analog substrate to host reasonably large graphs with tens of thousands nodes and edges. To further extend beyond this limit, we propose to investigate advanced decomposition techniques to handle very-large-scale network flow problems in an iterative manner where the substrate is reconfigured and reused to efficiently solve the intermediate subproblems. In particular, we plan to explore dual decomposition techniques such as [39], which partition the dual problem of max-flow into overlapping subproblems and iteratively solve the subproblems to arrive at a global optimal solution with agreement on the overlapping variables. For example, given a min-cut objective $minimize \sum_{i,j \in V} c_{ij} d_{ij}$ and a partition of variables into two sets M and N, the objective function can be reformulated as $minimize\ (E_M(x) + E_N(y))$, subject to $x_i = y_i, i \in M \cap N$, where $E_M(x) = \sum_{i,j \in M} c_{ij} x_{ij} - \frac{1}{2} \sum_{i,j \in M \cap N} c_{ij} x_{ij}$, and $E_N(y) = \sum_{i,j \in N} c_{ij} y_{ij} - \frac{1}{2} \sum_{i,j \in M \cap N} c_{ij} y_{ij}$. Here variable x_i belongs to set M and y_i belongs to set N. The variables in the overlap are constrained to be equal. The Lagrange dual function of this optimization

problem is: $g(\lambda) = minimize\ (E_M(x) + E_N(y) + \sum_{i \in M \cap N} \lambda_i(x_i - y_i)) = minimize\ (E_M(x) + \sum_{i \in M \cap N} \lambda_i x_i) + minimize\ (E_M(y) - \sum_{i \in M \cap N} \lambda_i y_i)$. Evaluating this function g is thus equivalent to solving two independent min-cut problems, and strong duality ensures $x_i = y_i, i \in M \cap N$ when g is maximized.

6.5 Circuit Dynamic and Asymptotic Behavior

We have shown that the steady-state voltage values of the circuit converge to the solution of the max-flow problem. We are also interested in learning the dynamic behavior of the circuit, namely, how the node voltages evolve as a function of time. The exact voltage trajectory depends on the drive voltage V_{flow}. The step function used in the previous sections induces complex transient behavior that require detailed numerical analysis. To gain intuition for circuit convergence, we examine a simpler scenario where we restrict V_{flow} to be a slow-varying function, so that at any given time step it is valid to approximate the circuit behavior using the corresponding steady-state solution, also known as the quasi-static approximation.

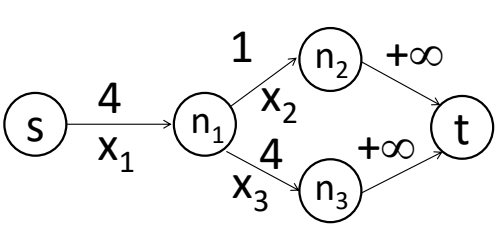
Consider the max-flow problem shown in Figure 15a, with the objective function and constraints shown in Equation (8). To simply the discussion, two of the edges have infinite edge capacity so that these two edges will not restrain the final flow value. Figure 15b shows the corresponding circuit for the max-flow problem, where the diodes and voltage sources due to edge capacity constraints are omitted for brevity. Figure 15c shows a graphical illustration of the solution space and how the circuit reaches the final voltage values, with the feasible region shown in green, and the voltage trajectory shown in red.

$$maximize \quad x_1 \tag{8a}$$
$$x_1 = x_2 + x_3 \tag{8b}$$
$$0 \leq x_1 \leq 4 \tag{8c}$$
$$0 \leq x_2 \leq 1 \tag{8d}$$
$$0 \leq x_3 \leq 4 \tag{8e}$$

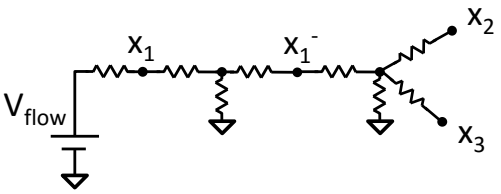
Assuming initially $V_{flow} = 0$, we solve for the voltage values of this circuit and show that $V_{x_1} = \frac{2}{9}V_{flow}$ and $V_{x_2} = V_{x_3} = \frac{1}{9}V_{flow}$. We slowly increase the value of V_{flow} until V_{x_2} reaches its maximum $V_{x_2} = 1V$, corresponding to point D in Figure 15c. At this point $V_{flow} = 9V$ and the diode at x_2 (not shown in the circuit diagram) comes into play to hard-wire V_{x_2} to 1V.

We then add the constraint $V_{x_2} = 1V$ to the circuit, and solve it again to obtain $V_{x_1} = 1 + V_{x_3}$, $V_{x_2} = 1V$, and $V_{x_3} = \frac{V_{flow}-4}{5}$. We increase V_{flow} all the way to $19V$. At this point $V_{x_1} = 4V$ and $V_{x_3} = 3V$. The circuit obtains the final solution $V_{x_1} = 4V$, $V_{x_2} = 1V$, and $V_{x_3} = 3V$, corresponding to point B in Figure 15c. At this point, the objective function cannot be further improved, thus the circuit reaches the final state, corresponding to the optimal solution of the max-flow problem.

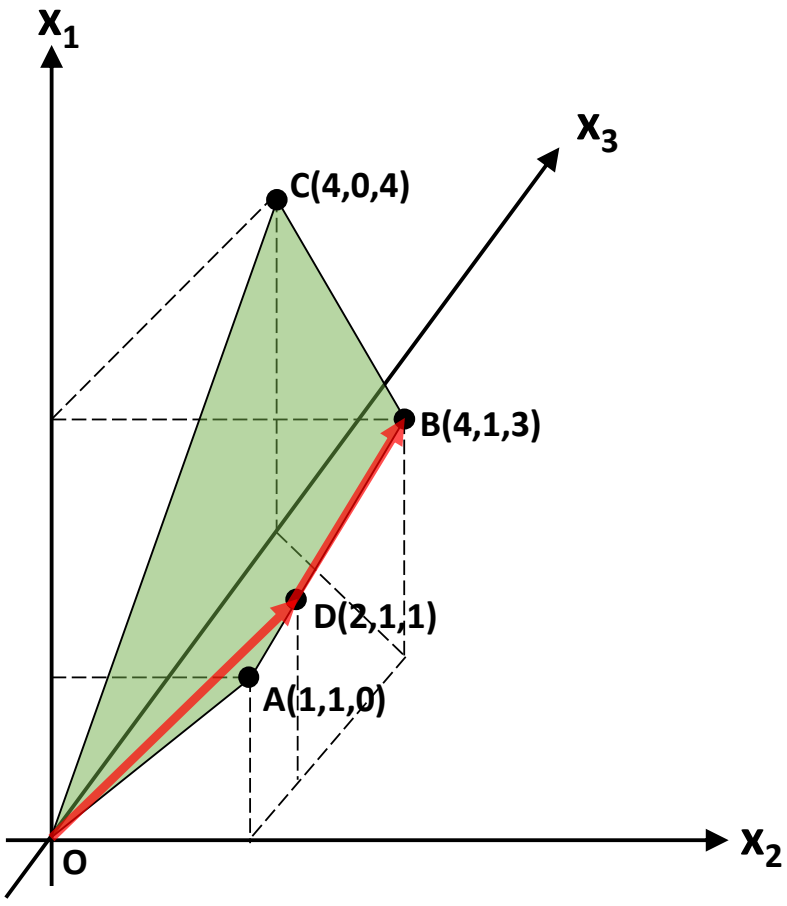
The trajectory in Figure 15c shows that instead of moving across the extreme points of the feasible region, the circuit actually drives the solution through the interior of the feasible region. We conjecture there is a loose connection between the analog circuit behavior and the algorithmic behavior of the interior-point method [36], and plan to further investigate the convergence rate of the analog circuit.



(a) Example max-flow problem.



(b) The corresponding circuit with diodes omitted for brevity.



(c) Trajectory of node voltages.

Figure 15: Example circuit dynamic behavior.

7. Related Work

There has been a large body of work seeking to break the limit of the traditional von Neumann architecture and to achieve higher performance and efficiency. Silicon implementation of artificial neural network is one of the most well-known efforts in this field. Recently, a digital neuromorphic chip with one million spiking neurons was built with a power consumption of less than 70mW [34]. Analog implementation of the neuron network is also studied to achieve speedup and power reduction for specific applications [1].

Novel devices such as memristors are utilized to construct unconventional computing substrates. Memristor-based substrates are demonstrated to solve maze [37], conduct matrix multiplication [23], as well as general approximate computation [27]. Hierarchical memristor crossbar design is proposed in [31] to improve the efficiency and flexibility of the neuromorphic computing accelerator. Our proposed clustered island-style architecture shares the same spirit of their hierarchical crossbar design. A generalized concept of universal memcomputing machine is also proposed as an alternative to the von Neumann architecture [11]. In an effort to achieve reconfigurability in analog circuit, field-programmable analog arrays (FPAAs) integrate dedicated analog components with floating-gate transistors and programmable resistor/capacitor networks, enabling reconfigurable computing in the analog domain [2].

Most relevant to our work, Vichik and Borrelli design an analog circuit to solve linear and quadratic programming problems, where unknown variables are directly modeled by node voltages and each constraint is specified by a dedicated circuit component [42]. The node voltages representing unknown variables are driven towards their optimal value subjected to the circuit constraints. However, their approach is problem specific, i.e., different instances of the analog circuit need to be built for solving different linear or quadratic programs. The structure of circuit components in our paper are built upon their work, but tailored to exploit the specific properties of max-flow problems. In addition, our substrate is reconfigurable, which is capable of solving multiple instances of the max-flow problem using the same substrate.

In the field of discrete algorithms, decades of effort has aimed to develop efficient max-flow algorithm. Apart from formulating the flow constraints and solving for the maximum flow as a linear programming problem, the Ford-Fulkerson algorithm based on augmenting path sets the foundation as a specialized max-flow algorithm [16]. Among the classical approaches, more efficient algorithms include Dinitz's blocking flow algorithm [12] and Goldberg and Tarjan's push-relabel algorithm [17]. More recently, Christiano et al. show that near-linear time approximate algorithm for the max-flow problem is possible, where the algorithm converts the max-flow problem to solving current flow in electrical circuit [8]. By modeling the behavior of the electrical circuit, efficient approximate solution to the original max-flow problem can be obtained. This result also inspired our study.

Although an ASIC or FPGA implementation of max-flow accelerator can improve performance and efficiency on a pure software execution, we note that in practice the demonstrated performance benefit is typically a moderate 3-$5X$ speedup over CPUs using an FPGA-based accelerator [25]. In fact, it has been shown that the max-flow problem is inherently hard to parallelize [18], implying that ASIC or FPGA based hardware acceleration can only achieve limited performance improvements. This motivates us to investigate more disruptive approaches in this research.

8. Conclusions

We present the design and analysis of a reconfigurable analog substrate whose steady-state node voltages represent the solution to a given max-flow problem. We achieve reconfigurability by configuring

the on/off states of the circuit switches that encode the graph topology. Using SPICE simulation and analytical modeling, we demonstrate two to three orders of magnitude improvements in speed and energy efficiency compared to a standard CPU implementation. We further propose techniques to minimize the impacts of variability and non-ideal circuit components on the solution quality, and provide insights on the dynamic behavior of the circuit. We hope this work can provide new insights and spark new research in building programmable analog substrate to solve even more advanced optimization problems.

Acknowledgements

The authors thank Steve Dai and Prof. Eilyan Bitar for fruitful discussions in the early phase of this research project.

References

[1] R. S. Amant, A. Yazdanbakhsh, J. Park, B. Thwaites, H. Esmaeilzadeh, A. Hassibi, L. Ceze, and D. Burger. General-Purpose Code Acceleration with Limited-Precision Analog Computation. *Int'l Symp. on Computer Architecture (ISCA)*, 2014.

[2] A. Basu, S. Brink, C. Schlottmann, S. Ramakrishnan, C. Petre, S. Koziol, F. Baskaya, C. M. Twigg, and P. Hasler. A Floating-Gate-Based Field-Programmable Analog Array. *IEEE Journal of Solid-State Circuits (JSSC)*, 45(9):1781–1794, 2010.

[3] A. A. Benczúr and D. R. Karger. Approximating s-t Minimum Cuts in $\tilde{O}(n^2)$ Time. *ACM Symposium on Theory of computing*, pages 47–55, 1996.

[4] V. Betz and J. Rose. FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density. *Int'l Symp. on Field-Programmable Gate Arrays (FPGA)*, 1999.

[5] D. Biolek, M. Di Ventra, and Y. V. Pershin. Reliable SPICE Simulations of Memristors, Memcapacitors and Meminductors. *Radioengineering*, 22(4), 2013.

[6] Y. Boykov and V. Kolmogorov. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(9):1124–1137, 2004.

[7] D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-MAT: A Recursive Model for Graph Mining. *Int'l Conf. on Data Mining (ICDM)*, 2004.

[8] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S.-H. Teng. Electrical Flows, Laplacian Systems, and Faster Approximation of Maximum Flow in Undirected Graphs. In *ACM Symposium on Theory of computing*, pages 273–282, 2011.

[9] E. S. Chung, P. A. Milder, J. C. Hoe, and K. Mai. Single-Chip Heterogeneous Computing: Does the Future Include Custom Logic, FPGAs, and GPGPUs? *Int'l Symp. on Microarchitecture (MICRO)*, 2010.

[10] J. Cong and Y. Ding. FlowMap: An Optimal Technology Mapping Algorithm For Delay Optimization in Lookup-Table Based FPGA Designs. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 13(1):1–12, 1994.

[11] M. Di Ventra and Y. V. Pershin. The Parallel Approach. *Nature Physics*, 9(4):200–202, 2013.

[12] E. Dinits. Algorithm of Solution to Problem of Maximum Flow In Network with Power Estimates. *Soviet Mathematics Doklady*, 194(4):754, 1970.

[13] A.-T. Do, Z.-H. Kong, K.-S. Yeo, and J. Y. S. Low. Design and Sensitivity Analysis of a New Current-Mode Sense Amplifier for Low-Power SRAM. *IEEE Trans. on Very Large-Scale Integration Systems (TVLSI)*, 19(2):196–204, 2011.

[14] H. Esmaeilzadeh, E. Blem, R. St Amant, K. Sankaralingam, and D. Burger. Dark Silicon and the End of Multicore Scaling. *Int'l Symp. on Computer Architecture (ISCA)*, 2011.

[15] G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee. Self-Organization and Identification of Web Communities. *IEEE Computer*, 35(3):66–70, 2002.

[16] L. R. Ford and D. R. Fulkerson. Maximal Flow Through a Network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.

[17] A. V. Goldberg and R. E. Tarjan. A New Approach to the Maximum-Flow Problem. *Journal of the ACM (JACM)*, 35(4):921–940, 1988.

[18] L. M. Goldschlager, R. A. Shaw, and J. Staples. The Maximum Flow Problem is Log Space Complete for P. *Theoretical Computer Science*, 21(1):105–111, 1982.

[19] F. Halim, R. H. Yap, and Y. Wu. A MapReduce-Based Maximum-Flow Algorithm for Large Small-world Network Graphs. *Int'l Conf. on Distributed Computing Systems (ICDCS)*, 2011.

[20] J. Han and M. Orshansky. Approximate Computing: An Emerging Paradigm for Energy-Efficient Design. *IEEE European Test Symposium (ETS)*, 2013.

[21] A. Hastings. The Art of Analog Layout. 2006.

[22] Y. Ho, G. M. Huang, and P. Li. Nonvolatile Memristor Memory: Device Characteristics and Design Implications. *Int'l Conf. on ComputerAided Design (ICCAD)*, 2009.

[23] M. Hu, H. Li, Q. Wu, and G. S. Rose. Hardware Realization of BSB Recall Function Using Memristor Crossbar Arrays. *Design Automation Conf. (DAC)*, 2012.

[24] M. P. Kennedy and L. O. Chua. Neural Networks for Nonlinear Programming. *IEEE Transactions on Circuits and Systems*, 35(5):554–562, 1988.

[25] D. Kobori and T. Maruyama. An Acceleration of a Graph Cut Segmentation with FPGA. *Int'l Conf. on Field Programmable Logic and Applications (FPL)*, 2012.

[26] M. N. Leuenberger and D. Loss. Quantum Computing in Molecular Magnets. *Nature*, 410(6830):789–793, 2001.

[27] B. Li, Y. Shan, M. Hu, Y. Wang, Y. Chen, and H. Yang. Memristor-Based Approximated Computation. *Int'l Symp. on Low-Power Electronics and Design (ISLPED)*, 2013.

[28] B. Liu, M. Hu, H. Li, Z.-H. Mao, Y. Chen, T. Huang, and W. Zhang. Digital-Assisted Noise-Eliminating Training for Memristor Crossbar-Based Analog Neuromorphic Computing Engine. *Design Automation Conf. (DAC)*, 2013.

[29] G. Liu, Y. Tao, M. Tan, and Z. Zhang. CASA: Correlation-Aware Speculative Adders. *Int'l Symp. on Low-Power Electronics and Design (ISLPED)*, 2014.

[30] G. Liu and Z. Zhang. A Reconfigurable Analog Substrate for Highly Efficient Maximum Flow Computation. *Design Automation Conf. (DAC)*, 2015.

[31] X. Liu, M. Mao, B. Liu, H. Li, Y. Chen, B. Li, Y. Wang, H. Jiang, M. Barnell, Q. Wu, et al. RENO: A High-Efficient Reconfigurable Neuromorphic Computing Accelerator Design. *Design Automation Conf. (DAC)*, 2015.

[32] A. Madry. Fast Approximation Algorithms for Cut-Based Problems in Undirected Graphs. *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 245–254, 2010.

[33] C. Mead. Neuromorphic Electronic Systems. *Proceedings of the IEEE*, 78(10):1629–1636, 1990.

[34] P. A. Merolla et al. A Million Spiking-Neuron Integrated Circuit with A Scalable Communication Network and Interface. *Science*, 345(6197):668–673, 2014.

[35] L. W. Nagel and D. O. Pederson. SPICE: Simulation Program with Integrated Circuit Emphasis. 1973.

[36] Y. Nesterov, A. Nemirovskii, and Y. Ye. *Interior-point Polynomial Algorithms in Convex Programming*. SIAM, 1994.

[37] Y. V. Pershin and M. Di Ventra. Solving Mazes with Memristors: A Massively Parallel Approach. *Physical Review E*, 84(4):046703, 2011.

[38] A. Schrijver. On the History of the Transportation and Maximum Flow Problems. *Mathematical Programming*, 91(3):437–445, 2002.

[39] P. Strandmark and F. Kahl. Parallel and Distributed Graph Cuts by Dual Decomposition. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[40] T. Szymanski. Max-Flow Min-Cost Routing in a Future-Internet with Improved QoS Guarantees. *IEEE Trans. on Communications*, 61(4):1485–1497, 2013.

[41] D. W. Tank and J. J. Hopfield. Simple "Neural" Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit. *IEEE Transactions on Circuits and Systems*, 33(5):533–541, 1986.

[42] S. Vichik and F. Borrelli. Solving Linear and Quadratic Programs with an Analog Circuit. *Computers and Chemical Engineering*, 70(5):160–171, 2014.