Solving Vertex Cover via Ising Model on a Neuromorphic Processor

Kevin Corder
University of Delaware
kcorder@udel.edu

John V. Monaco
U.S. Army Research Laboratory
john.v.monaco2.civ@mail.mil

Manuel M. Vindiola
U.S. Army Research Laboratory
manuel.m.vindiola.civ@mail.mil

***Abstract*—Neuromorphic architectures are characterized by a network of asynchronous computation units that resemble the behavior of neurons in the brain. With much interest devoted to how these architectures can be utilized in learning and intelligent systems, we note that the massive parallelism of neuromorphic architectures can also be leveraged to solve some NP problems more efficiently than on a von Neumann architecture. In this work, we demonstrate how a neuromorphic processor can be used to solve the classic vertex cover problem via an Ising spin model. Mapping the Ising model to a neural architecture itself requires solving two NP-hard problems at initialization, for which we use approximate solutions. The maximum fan-in and fan-out constraint, common on many neuromorphic processors, requires a graph partitioning, and solving the time-dependency constraints in updating the graph amounts to a graph coloring problem. As a result, space and time efficiency is decreased only by a constant factor without degrading solution quality.**

I. Introduction

Neuromorphic chips have recently gained popularity due to their ability to operate within extreme size-, weight-, and power-constrained environments. A number of new neuromorphic processors have been developed—including SpiNNaker, TrueNorth, NeuroGrid, and BrainScales—each with their own advantages [1]. This has naturally led to much interest invested in how these devices can be leveraged for machine learning and intelligent systems, such as to efficiently perform the inference task of a deep neural network. Less explored is how these massively-parellel processors can be used for solving computationally-intensive problems, particularly those for which the von Neumann bottleneck is a limitation to scaling.

Recent work has explored using materials innate with the properties of the Ising model [2]. The complexity of an Ising model simulation could be dramatically reduced by using a material such as a magnetic-tunnel-junction which already contains the quantum properties of stochastic annealing and particle majority voting. Other recent work has leveraged the massive parallelism and constant-time synaptic integration of neuromorphic processors to perform integer factorization [3], solve constraint satisfaction [4] and quadratic unconstrained binary optimization problems [5], or used reinforcement learning and neural networks to solve combinatorial problems [6].

In this paper we give a neuromorphic implementation for solving the classic vertex cover problem. In a graph $G = (V, E)$, a vertex cover is a subset of the vertices $V' \subseteq V$ such that every edge $(u, v) \in E$ has an endpoint in V'. Let the number of vertices $|V| = n$. We use the Ising model formulation of vertex cover [7] and adapt it for a Metropolis-Hastings simulation in a neuromorphic architecture. This involves distributing the Hamiltonian function among nodes, determining how to best fit nodes onto cores, and minimizing the time for updating all nodes on each iteration.

The Metropolis-Hastings algorithm is a well-known and popular Markov chain Monte Carlo (MCMC) method to approximate high-dimensional distributions by iterative sampling. Our formulation uses the Metropolis-Hastings algorithm to simulate an Ising model with an energy function that is minimized by valid a vertex cover. The Ising model is mapped to a network of leaky integrate and fire (LIF) neurons, where the behavior of each node in the Ising model is achieved by a small fixed-size network of neurons.

To implement the Ising model on a neuromorphic architecture, we needed to know which nodes (1) can be simultaneously updated, (2) should be placed together on cores, and (3) should share probability inputs. The first issue corresponds to a graph coloring problem, the second to a graph partitioning problem, and the third is a maximal matching of degrees of distinct color classes for each new graph partition. These hard subproblems are solved at initialization time with greedy, polynomial solutions. Non-exact solutions for these subproblems increase the space used in hardware and iteration time, but do not affect solution quality.

We implemented the Ising model on TrueNorth, a digital neuromorphic processor with 1 kHz update rate, and examined the number of ticks (i.e., clock cycles) needed to reach a solution. Solution quality depends on several hyperparameters, including the number of ticks to run for and the initial temperature, both of which are dependent on the graph size and density. Although not always valid vertex cover solutions, those discovered by our implementation generally use about the same number of nodes as the common vertex cover 2-approximation method. For dense graphs, our vertex covers require more than the minimum number of nodes, a fundamental issue with the distributed, asynchronous updates used in our mapping.

The rest of the paper is organized as follows. In Section II, we describe the vertex cover problem and general Ising model. Section III discusses how the vertex cover problem is encoded in a modified Ising model formulation and the space required for complete graphs. In Section IV we detail the TrueNorth model and constraints, how each graph node is represented as a

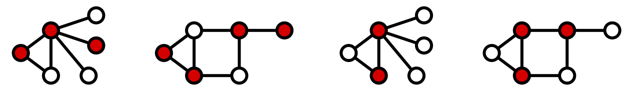
(a) A vertex cover (not minimum) (b) A minimum vertex cover

Fig. 1. Vertex covers shown on the same underlying graphs, where a highlighted node indicates that it is contained in the cover.

circuit on chip, and how the network is processed in order to fit onto cores and run in proper timing on the TrueNorth. Section V shows our numerical results of solution quality for various random graphs. Finally, Section VI contains a discussion and describes future work.

II. Background

A. Vertex Cover

Finding a minimum vertex cover of a graph is a classical optimization problem. In an undirected graph $G = (V, E)$, a vertex cover is a subset of the vertices $V' \subseteq V$ such that every edge $(u, v) \in E$ has at least one of u, v in V'. A minimum vertex cover is a vertex cover with the fewest possible number of nodes included in the cover. Every graph G trivially has a vertex cover where $V' = V$. Figure 1a shows a vertex cover, and Figure 1b shows a minimum vertex cover for the same graph.

The vertex cover problem is NP-complete, and the minimum vertex cover problem is NP-hard. A simple approximation method can find 2-approximate vertex covers (i.e., those that are not larger than twice the minimal solution), but others have slightly improved on this bound. We compare our solutions against the 2-approximation and a mixed-integer linear programming reduction of the problem for ground truth.

B. Ising Model

The Ising spin model is a statistical mechanics model of ferromagnetism that considers interactions between magnetic particle spins. The general Ising model consists of a k-D lattice of particles, each with an atomic spin of either +1 or -1, interacting with their 2^k immediately adjacent particles. The energy associated with each particle interaction depends on their spins, and the interaction strength between spins σ_i, σ_j is J_{ij}. Additionally, each spin σ_i has an interaction h_i with an external magnetic field. Intuitively, this follows the Pauli exclusion principle: two particles in a quantum system cannot simultaneously occupy the same quantum state, so having opposite spins has lower energy (more stable). The system Hamiltonian function H, the nonnegative sum of local spin interaction energies of a spin configuration σ, is given by:

$$H(\sigma) = -\sum_{\langle i,j \rangle} J_{ij}\sigma_i\sigma_j - \mu \sum_i h_i \sigma_i \tag{1}$$

Several versions of the Ising model have been shown to be NP-complete, including those with nonplanar connections, a 3D lattice, a 2-D planar model with an external field, or the *spin glasses model* where spins take continuous values [8], [9]. An arbitrary graph with varying degree between nodes, such as our model, could easily be nonplanar and may thus be NP-complete to simulate on a traditional computer.

Once the Ising model is embedded in an environment with a temperature T, which corresponds with the chance of exploration, the probability of any configuration of spins is given by the Boltzmann distribution

$$P_\beta(\sigma) = \frac{e^{-\beta H(\sigma)}}{\sum_\sigma e^{-\beta H(\sigma)}} \tag{2}$$

where $\beta = \frac{1}{k_B T}$ is the inverse temperature. There is a fixed number of configuration probabilities for any lattice, and more importantly for our problem, even fewer distinct probabilities for each spin site σ_i.

III. Vertex Cover as Ising Model

In the Metropolis-Hastings MCMC method, states change stochastically such that lower energy states are always accepted and higher energy states are accepted with some probability (given by the Boltzmann distribution in our case). An Ising model encoding of the vertex cover problem should then give a valid vertex cover when its Hamiltonian function is minimized. A +1 spin indicates that a node is contained in the vertex cover, and a -1 spin node is excluded.

In the Ising formulation of the vertex cover problem [7], the system Hamiltonian H is defined as $H = H_A + H_B$, where H_A encodes each edge having an endpoint in the cover and H_B encodes minimizing the number of colored vertices:

$$H_A = A \sum_{u,v \in E} (1 - x_v)(1 - x_u) \tag{3}$$

$$H_B = B \sum_{v \in V} x_v \tag{4}$$

for positive constants A, B and binary variable x_i, where $x_i = 0$ represents a -1 spin and $x_i = 1$ represents a +1 spin. It is required that $B < A$ so that decreasing the number of colored nodes never lowers energy if the result is not a valid vertex cover. We additionally require A to be even to satisfy integer constraints on weights (we use $A = 2, B = 1$). The full Hamiltonian is:

$$H = B \sum_{v \in V} x_v + A \sum_{u,v \in E} (1 - x_v)(1 - x_u) \tag{5}$$

$$= \sum_i^V \left[Bx_i + \frac{A}{2}(x_i - 1) \sum_{u:(i,u) \in E} (x_u - 1) \right] \tag{6}$$

We will call the expression within the brackets of Equation 6 the *local Hamiltonian* for a vertex v_i because it represents that node's marginal energy contribution to H.

Take the second graph in Figure 1b as an example. All edges have one endpoint in the cover so $H_A = A \cdot 0 = 0$, and $H_B = 3B$ because 3 nodes are colored.

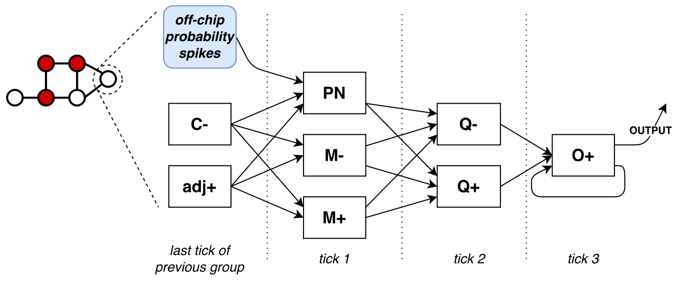

Fig. 2. A circuit representing a single node in the graph. Each neuron's spikes are sent to an axon and received by potentially multiple neurons. Each node group (nodes with same color in graph coloring) update takes 3 ticks.

IV. TrueNorth Implementation

A. TrueNorth Constraints

The TrueNorth architecture simulates a LIF neuron model with membrane potential given by

$$V(t) = V(t-1) + \sum_i a_i w_i + \lambda \tag{7}$$

at time t with the i-th neuron input a_i ,synapse weight w_i, and leak λ [10]. Due to precision constraints, all of these values must be integers. The neuron first integrates the potential of its weighted inputs and leak, spikes or fires when the potential $V(t)$ exceeds the threshold α, and then resets to a specified rest potential V_{rst}. Time advances in discrete *ticks*, where all spikes are delivered to their destinations within each tick.

Each TrueNorth chip consists of 4096 cores, each with 256 neurons and 256 axons arranged in a crossbar. A neuron can send its spike to a single axon on any core, and all neurons on that core can receive the axon's signal. Therefore, synaptic connectivity on TrueNorth supports dense core crossbar connections and sparse inter-core connections.

Each core has 4 available axon types, and each neuron can assign a unique weight to each axon type. Thus, the synapses connected to a single neuron must share a set of 4 unique weights, and all the neurons on a core share the same permutation of axon types. All of these parameters are fixed when the model is built and cannot be changed during execution. For this reason, the environment temperature is adjusted off-chip through a series of *probability spike* inputs.

B. Node circuit

We implement each node of the graph by a *node circuit*, which maintains and updates the node state dependent on its neighbors in G, shown in Figure 2. The node circuit must be able to (1) maintain the state of node over time, and (2) change state if its local Hamiltonian lowers (shown in Equation 6).

For a given node with degree d, let adj_+ be the number of adjacent nodes with a +1 spin (those contained in the vertex cover). There are $d+1$ distinct acceptance probabilities which cause the local energy to be lowered.

A node circuit consists of 6 unique neuron types. The M- and M+ neurons will fire if the local energy is lowered by changing state to -1 or +1, respectively; at most one M neuron will fire. The Q- and Q+ neurons multiplex the M neurons by changing state to -1 or +1 if the associated M neuron fires, or if the probability neuron PN fires; at most one Q neuron will fire. PN integrates the off-chip probability spikes and fires if the number of spikes $\geq adj_+$. The recurrent output neuron O+ maintains the node's state and switches state if either of Q- and Q+ fired. Three ticks are needed to propagate the signal through a node circuit.

Besides node circuit neuron types, there are also the C+ (excitatory) and C- (inhibitory) *clock neurons* which regulate the firing of the rest of the network. The C+ neuron "counts" and signals when a particular group of nodes should activate to potentially change state. Due to neuron constraints, however, we use C- neurons to invert the signal of their associated C+ neurons and suppress all node circuits from firing out of turn. These clock neurons reside on a dedicated core(s).

Describing the neuron parameters is out of the scope of this paper, but in general we use $\alpha = 1$ and $V_{rst} = 0$ for all neuron types so there is no residual potential on each tick, and all neurons in a node circuit receive its current O+ state.

C. Network Preprocessing

After replacing each graph node with its node circuit implementing the local Hamiltonian behavior, to implement on neuromorphic hardware we needed to determine which nodes (1) can be simultaneously updated, (2) should be placed together on cores, and (3) should share input probability axons.

To distribute the Ising model simulation we would like to update nonadjacent nodes in the graph simultaneously. This translates to a graph coloring problem where all nodes of the same color class can be concurrently updated (and will share the same C+ neuron). We solve this NP-hard problem with a greedy solution at initialization. The number of color classes from this step determines the number of iterations for a full update of all nodes. The lower bound for a full update is the chromatic number $\chi(G)$ of G. Let $\gamma(G)$ be the number of color classes from our greedy solution; the time for a full update is then $3\gamma(G)$.

Due to fan-in, fan-out constraints of the neuromorphic hardware, distributing the node circuits optimally among cores translates to the NP-hard graph partitioning problem such that the subgraphs will reside on different cores. Each inter-core edge to a distinct core requires an additional sending neuron for each node and a receiving axon for each distinct inter-core node. Since dense local connections are most space efficient on the TrueNorth architecture, we greedily partition problem G into dense components with recursive spectral bisection: bisect G using the minimum cut of the graph according to the Fiedler eigenvector, and recurse on the pieces induced by this cut [11]. This recursion is repeated until each subgraph $G' \subseteq G$ fits onto a core, keeping track of the number of neurons and axons required by G' (including the extras for inter-core edges, probability spikes, and C- spikes).

Note that these greedy solutions for graph partitioning and graph coloring only increase the number of allocated cores and number of ticks in a full node update, respectively—neither will affect the solution quality.

All nodes with degree d on each core need at least $d+1$ probability spikes for their PN neuron but not all nodes will execute each iteration. Therefore, we can save space by sharing these probability axons between nodes of different colors and we need only $\gamma(c_i)$ sets of probability axons, where $\gamma(c_i)$ is the number of color classes on core c_i. Determining which nodes should share probability axons is then a maximal matching of node degrees with different color classes from the graph coloring step. We do a simple sort of each color class's nodes' degrees, then iteratively select the maximum degree as the number of axons for those nodes to share.

D. Space Usage

As can seen in Figure 2, each node circuit requires 8 neurons: M+, M-, PN, Q+, Q-, and 3 copies of O+ for the recurrent state, adjacent nodes on the same core, and the off-chip output state. The circuit requires an additional O+ neuron for each inter-core edge to a unique core because all neurons on the receiving core share that axon; similarly, only one additional axon is required for each adjacent node on another core.

Each node circuit requires 7 axons and may share axons for receiving inter-core edges, probability spikes, and C- clock signals. The number of C- axons is exactly $\gamma(c_i)$ because these are shared between all nodes of each color class. As we saw in the previous section, $\gamma(c_i)$ probability spike sets are needed and each set need only be as large as the color's highest degree node $+1$.

Proposition 1. *The largest fully-connected graph that can be used on TrueNorth has 124 nodes.*

Proof. In the largest fully-connected graph K_n with $|V| = n$ nodes, each core c_i contains a single node v_i with $n-1$ off-core neighbors. Every node has degree $n-1$ and is on its own core, so each core needs $n-1$ axons for receiving adj_+ states and $(n-1)+1 = n$ probability axons. Since there will be more axons than neurons, we want to find maximal n such that $7 + (n-1) + (n-1+1) + 1 \leq 256$. $n = 124$ is the maximum. $\square$

V. Experiments

In our experiments, shown in Figure 3, we measure the solution qualities of many random graphs with varying number of nodes n ($n \in \{50, 100, 150, 200\}$), probability p_e of adding an edge between any two nodes ($p_e \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$), and the number of ticks t used in the simulation ($\{31, 61, ..., t, t{+}30, ..., 391\}$). A graph is made for each distinct pair (n, p_e) where a model is built for each maximum tick count. Each of these models is then run over the 10 iterations, where the vertex cover sizes are averaged to stabilize the off-chip probability spikes.

The initial node state is chosen randomly from $\{0, 1\}$. We use the initial temperature $T = \frac{2}{\log(1+\sqrt{2})} \approx 5.22$ and lower it linearly in these experiments. We leave hyperparameter tuning

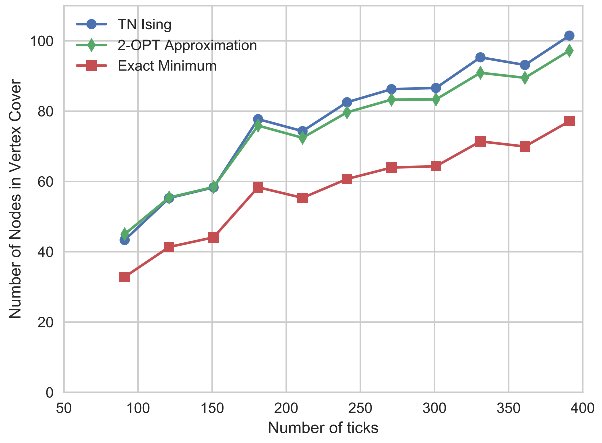

Fig. 3. Number of nodes in output vertex cover of our TrueNorth Ising model reduction and the common 2-approximation method compared against ground truth minimum as a MILP reduction.

of the the initial temperature, lowering function, and speed to future work.

Figure 3 shows that our model finds valid vertex covers that are competitive with the 2-approximation method.

VI. Conclusion & Future Work

In this work we described how to solve the vertex cover problem on a neuromorphic processor by mapping to an Ising model formulation and designing a node circuit to implement the local Hamiltonian function of that model. This work should be instructive for solving other graph problems on neuromorphic systems. In initializing this network on the TrueNorth architecture, we solved graph partitioning and graph coloring NP-hard problems with greedy solutions and optimized for space efficiency using spectral bisection.

Unfortunately, we cannot give a detailed description of the neuron parameters and network preprocessing methods here. There is much more to show in the way of experimental results: future work will present experiments with varying (1) initial temperature, (2) temperature lowering speed or function, and (3) vertex covers in *small-world networks*, whose dense short-range and sparse long-range connectivity is far better suited to the space constraints on neuromorphic processors. We will also show solution results dependent on the graph size and include timing data. The total time ultimately depends on how quickly the temperature T is lowered off-chip, and thus how quickly the exploration probabilities are lowered. The temperature must be lowered slowly enough so that the converged solution is good, and hopefully optimal.

While its operating frequency is much slower, the TrueNorth requires many fewer clock cycles to update the complete graph than a CPU or GPU implementation. With the promise of vastly increased clock rates in future neuromorphic architectures, there exists a potential for scaling up to larger problems and achieving much lower wall clock time than on a von Neumann architecture.

Appendix: Neuron Parameters

TrueNorth neuron parameters and default values include: membrane potential threshold α (1), floor β (0), leak λ (0), reset potential V_{rst} (0), initial potential V_m (0), and synapse weights for each axon type σ ([0,0,0,0]). Table I contains the neuron parameters to configure each neuron type in the node circuit shown in Table 2, omitting those for which the default did not change.

Each neuron has a potential threshold α at which it spikes and sends its signal to an axon, which must be of a certain axon type, followed by resetting its potential to V_{rst}. The initial potential is given by V_m. Each neuron also has a negative threshold β, and a leak parameter λ which is added as a bias after every tick.

Synapse weights on TrueNorth are limited in resolution due to there being 4 axon types. All neurons on a core share the same permutation of axon types, i.e., axon types are specified at the core level. Each neuron includes a length-4 vector σ, with integer elements in the range $[-255, 255]$, which specifies the synapse weights for connections along each axon type.

Note that several neuron parameters depend on the input graph and the degree of the current node (that the neuron represents). The approximated graph coloring result, which determines the number of node circuit update groups, results in γ colors. Let $\mathcal{C}$ be the assigned color of the current node ($1 \leq \mathcal{C} \leq \gamma$), and let d be the degree of the current node. Each node group takes $\tau = 3$ ticks to update.

TABLE I
Neuron Parameters

Neuron Type	Parameter	Value
C+	σ	$[0, 0, 0, 0]$
	λ	1
	α	$\gamma\tau$
	V_m	$\alpha - \tau(\mathcal{C} - 1) - 1$
C-	σ	$[-1, 0, 0, 0]$
	λ	1
	V_m	1
PN	σ	$[-255, 0, -1, 1]$
	axon type	3
M+	σ	$[-255, -2d + 1, -2, 0]$
	λ	$1 + 2(d - 1)$
	V_m	$-\lambda$
	axon type	2
M-	σ	$[-255, 2, 1, 0]$
	λ	$-d$
Q+	σ	$[-255, -1, 255, 1]$
	axon type	2
Q-	σ	$[255, 1, -255, 1]$
	λ	-1
O+	σ	$[-2, 1, 2, 0]$
	axon type	1

References

[1] G. Indiveri and S. C. Liu, “Memory and Information Processing in Neuromorphic Systems,” *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1379–1397, Aug 2015.

[2] Y. Shim, A. Jaiswal, and K. Roy, “Ising computation based combinatorial optimization using spin-Hall effect (SHE) induced stochastic magnetization reversal,” *Journal of Applied Physics*, vol. 121, no. 19, p. 193902, 2017. [Online]. Available: https://doi.org/10.1063/1.4983636

[3] J. V. Monaco and M. M. Vindiola, “Integer factorization with a neuromorphic sieve,” in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4.

[4] G. A. Fonseca Guerra and S. B. Furber, “Using Stochastic Spiking Neural Networks on SpiNNaker to Solve Constraint Satisfaction Problems,” *Frontiers in Neuroscience*, vol. 11, p. 714, 2017. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2017.00714

[5] M. Z. Alom, B. V. Essen, A. T. Moody, D. P. Widemann, and T. M. Taha, “Quadratic Unconstrained Binary Optimization (QUBO) on Neuromorphic Computing System,” in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 3922–3929.

[6] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, “Neural Combinatorial Optimization with Reinforcement Learning,” *arXiv preprint arXiv:1611.09940*, 2016.

[7] A. Lucas, “Ising formulations of many NP problems,” *Frontiers in Physics*, vol. 2, p. 5, 2014. [Online]. Available: https://www.frontiersin.org/article/10.3389/fphy.2014.00005

[8] F. Barahona, “On the computational complexity of Ising spin glass models,” *Journal of Physics A: Mathematical and General*, vol. 15, no. 10, p. 3241, 1982. [Online]. Available: http://stacks.iop.org/0305-4470/15/i=10/a=028

[9] B. A. Cipra, “The Ising model is NP-complete,” *SIAM News*, vol. 33, no. 6, pp. 1–3, 2000. [Online]. Available: https://www.siam.org/pdf/news/654.pdf

[10] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science*, vol. 345, no. 6197, pp. 668–673, 2014. [Online]. Available: http://science.sciencemag.org/content/345/6197/668

[11] R. Kannan, S. Vempala, and A. Vetta, “On Clusterings: Good, Bad and Spectral,” *J. ACM*, vol. 51, no. 3, pp. 497–515, May 2004. [Online]. Available: http://doi.acm.org/10.1145/990308.990313