The Quantum Approximate Optimization Algorithm

Ryan Hoque

May 2019

1 Overview

This paper will be covering the development and current state of the Quantum Approximate Optimization Algorithm (QAOA), a promising heuristic algorithm originally presented by Edward Farhi et al. in 2014 [3]. We will discuss the algorithm itself, its application to Max-Cut and other combinatorial optimization problems, its connection to adiabatic quantum computation [5], how it compares to classical methods, how useful it may be in the future, and recent extensions of the algorithm such as the Quantum Alternating Operator Ansatz [7].

2 Background

In order to understand and properly motivate QAOA, we need to review some key concepts.

2.1 Adiabatic Quantum Computation

Adiabatic quantum computation (AQC) is a theoretical framework for a quantum computer. A competing paradigm is the circuit model, in which quantum algorithms are

implemented as circuits composed of quantum gates representing unitary operations. Interestingly enough, Edward Farhi, the MIT professor who created QAOA, also was the first to propose adiabatic quantum computation in 2000 [5].

In AQC we are interested in two entities: the driver Hamiltonian (H_D) and the problem Hamiltonian (H_P). A Hamiltonian represents the energy state of a quantum system. In AQC, the problem Hamiltonian encodes a quantum state we are interested in (the end result of some algorithm) as its ground state, and the driver Hamiltonian encodes some quantum state that is easy to prepare as its ground state. In other words, we start with a ground state that is easy to prepare (i.e. the ground state of H_D) and wish to end up with the quantum state we are interested in (i.e. the ground state of H_P), which is usually significantly more difficult to prepare.

This transition is accomplished via the Adiabatic Theorem, which states that a system in the ground state of some Hamiltonian will remain in the ground state if the Hamiltonian is changed slowly enough. More formally, we have some $s(t)$, a smooth function on $[0, T]$ where $s(0) = 0$ and $s(T) = 1$ known as the *schedule*. T is a value of time set high enough for the Adiabatic Theorem to hold. We define the Hamiltonian

$$H(t) = (1 - s(t))H_D + s(t)H_P$$

and let our quantum system evolve under it. By the Adiabatic Theorem, given an appropriate $s(t)$, we will stay in the ground state of this $H(t)$ during the entire interval $[0, T]$. Therefore, we can see that at time $t = 0$ we are in the ground state of H_D and by time $t = T$ we are in the desired ground state of H_P.

Unfortunately, time evolution under this time-dependent Hamiltonian involves a very messy integral that is hard to evaluate:

$$U(t) = \tau \exp\{\frac{-i}{\hbar}\int_0^t H(T)dT\}.$$

We can mitigate this with the Trotterization technique [9]. We discretize $U(T)$ into intervals of Δt small enough that the Hamiltonian is approximately constant over each interval. This allows us to use the much cleaner formula for the time-independent Hamiltonian. If we let $U(b, a)$ represent time evolution from time a to time b,

$$\begin{aligned} U(T,0) &= U(T, T-\Delta t)U(T-\Delta t, T-2\Delta t)\cdots U(\Delta t, 0) \\ &= \prod_{j=1}^{p} U(j\Delta t, (j-1)\Delta t) \\ &\approx \prod_{j=1}^{p} e^{-iH(j\Delta t)\Delta t} \end{aligned}$$

where the approximation improves as p gets larger (or, equivalently, as Δt gets smaller) and we have chosen Δt to be in units of $\hbar$. Finally, using the approximation $e^{i(A+B)x} = e^{iAx}e^{iBx} + \mathcal{O}(x^2)$ and plugging in our Hamiltonian $H(j\Delta t) = (1 - s(j\Delta t))H_D + s(j\Delta t)H_P$,

$$U(T,0) \approx \prod_{j=1}^{p} \exp\{-i(1-s(j\Delta t))H_D\Delta t\} \exp\{-is(j\Delta t)H_P\Delta t\}. \tag{1}$$

Thus we can approximate AQC by repeatedly letting the system evolve under H_P for some small $s(j\Delta t)\Delta t$ and then H_D for some small $(1 - s(j\Delta t))\Delta t$. We can efficiently construct the unitaries for these operations as $U = e^{-i\alpha H\Delta t}$, where α is some number in [0,1] that incorporates the scaling due to $s(j\Delta t)$.

2.2 Combinatorial Optimization

A classical combinatorial optimization problem is the task of finding an optimal object from a finite set of objects. We can pose the problem in terms of constraints we wish to satisfy so that the "optimal object" is the one that satisfies the most constraints. Usually the set of objects is too large to exhaustively search (i.e. the problem is NP-hard), and we resort to approximating the optimal solution. (Since we will try to do this with QAOA,

this is where the "Approximate Optimization" in "Quantum Approximate Optimization Algorithm" comes from.)

Formally, we can define an approximate combinatorial optimization problem as finding the n-bit string z that approximately satisfies the maximal amount of m constraints C_α, each of which takes the form

$$C_\alpha(z) = \begin{cases} 1 \text{ if } z \text{ satisfies the constraint} \\ 0 \text{ otherwise.} \end{cases} .$$

We wish to find a z that approximately maximizes the objective function

$$C(z) = \sum_{\alpha=1}^{m} C_\alpha(z). \tag{2}$$

The quantum analogue of this problem defines $\hat{C}$ as a diagonal operator on the 2^n-dimensional Hilbert space where each bitstring z is a basis vector $|z\rangle$ [3]. $\hat{C}$ acts on $|z\rangle$ as follows, where $C(z)$ is as defined in (2):

$$\hat{C}\,|z\rangle = C(z)\,|z\rangle\,. \tag{3}$$

Since $C(z)$ is a scalar, we can see that each $|z\rangle$ is an eigenstate of $\hat{C}$. If we view $\hat{C}$ as a Hamiltonian (since $\hat{C}$ is diagonal, it is Hermitian), the highest energy eigenstate $|z\rangle$ is the solution to the combinatorial optimization problem, as it gives the highest value of $C(z)$.

A well-known example of a combinatorial optimization problem is the Traveling Salesman Problem (TSP), which, given a list of cities, the pairwise distances between them, and the starting city, asks for the shortest path that visits each city exactly once and returns to the starting city. TSP has *global constraints* in that we must compute distance on all the cities in our attempts to find the shortest path. QAOA, as we will see, is well-suited for problems with *local constraints*. One such problem is the Max-Cut problem,

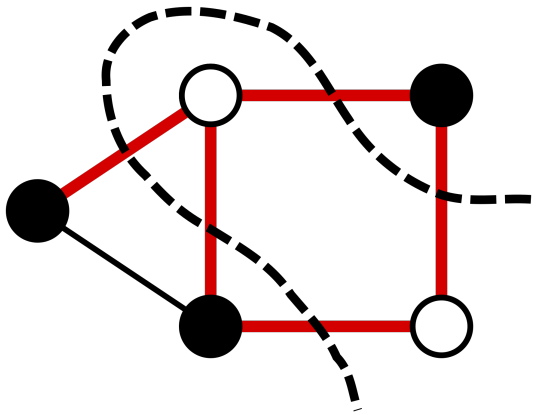

Figure 1: An instance of Max-Cut. The white vertices form one set and the black vertices form the other, yielding the given maximum partition with 5 crossing edges.

the application given in the original paper and most of the QAOA literature. Given a graph, Max-Cut attempts to partition the vertices into two sets such that the number of crossing edges between the sets is maximized. See Figure 1 for an illustration. Formally, we can label each vertex z_i as 1 or -1 to represent which set it belongs to and maximize

$$C(z) = \frac{1}{2} \sum_{(i,j) \in E} 1 - z_i z_j$$

where E is the set of edges. By observing $C(z)$ we can see that the problem has *local* constraints because each individual constraint only involves z_i and z_j.

3 The Algorithm

We can now discuss the steps of the actual algorithm. Because it leverages both classical and quantum techniques, it is a hybrid quantum-classical algorithm. QAOA leverages approximate adiabatic quantum computation as discussed in Section 2.1. We want to get from an eigenstate of the driver Hamiltonian to one of the problem Hamiltonian via adiabatic evolution. Our problem Hamiltonian is $\hat{C}$ as defined in (3), repeated here and

relabeled as C for convenience:

$$C\left|z\right\rangle = \sum_{\alpha=1}^{m} C_\alpha(z)\left|z\right\rangle.$$

Recall that the maximum energy eigenstate of C is the solution to the combinatorial optimization problem. For our driver Hamiltonian we choose

$$B = \sum_{j=1}^{n} \sigma_j^x$$

where σ_j^x is the σ^x Pauli operator on bit z_j. This is known as a "mixing operator" and has the easily-prepared maximum energy eigenstate

$$W\left|0\right\rangle = \left|+\right\rangle_1 \left|+\right\rangle_2 \cdots \left|+\right\rangle_n$$

where W is the Walsh-Hadamard gate on n qubits. Also, we define $U(C,\gamma) = e^{-i\gamma C}$ and $U(B,\beta) = e^{-i\beta B}$, which is letting the system evolve under C for some γ amount of time and under B for some β amount of time respectively.

The Quantum Approximate Optimization Algorithm.

Step 1. Find angles $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_p)$ and $\beta = (\beta_1, \beta_2, \ldots, \beta_p)$ to maximize the expression

$$F_p(\gamma,\beta) = \left\langle \gamma,\beta\right| C\left|\gamma,\beta\right\rangle$$

where $\left|\gamma,\beta\right\rangle$ is a quantum state parameterized by the angles γ and β to be defined in Step 2. We do this step classically. Since each $\gamma_i \in [0, 2\pi]$ and each $\beta_i \in [0, \pi]$, given an efficient way of calculating $F_p(\cdot)$ we can use a technique like grid search, which discretizes the range of each γ_i and β_i and evaluates $F_p(\cdot)$ on all possible combinations. Notice that if $\left|\gamma,\beta\right\rangle$ is an eigenstate of C, $\left\langle \gamma,\beta\right| C\left|\gamma,\beta\right\rangle$ gives the corresponding eigenvalue. Therefore,

maximizing $F_p(\cdot)$ corresponds to approximating the highest energy eigenstate of C (the best we can do with the value of p we choose).

Step 2. We now turn to our quantum computer to prepare the quantum state $|s\rangle = |+\rangle_1 |+\rangle_2 \cdots |+\rangle_n$ and compute

$$|\gamma, \beta\rangle = U(B, \beta_p)U(C, \gamma_p) \cdots U(B, \beta_1)U(C, \gamma_1) |s\rangle .$$

If we compare this with Equation (1), we can see that this is just approximate adiabatic quantum computation! We begin in an eigenstate of H_D and then repeatedly let the system evolve under C and B, alternating between the two. (Small caveat: since we want the *highest energy* eigenstate instead of the *lowest energy* eigenstate, we substitute $-C$ for H_P and $-B$ for H_D in (1).) Our approximation improves as p increases, and as $p \to \infty$ we perfectly replicate AQC as we can set Δt arbitrarily small in our Trotterization.

Step 3. Measure $|\gamma, \beta\rangle$ in the computational basis to get some bitstring z, and evaluate $C(z)$.

Step 4. Repeat Steps 2 and 3 $\mathcal{O}(m \log m)$ times (where m is the number of constraints) to obtain z such that $C(z) \geq F_p(\gamma, \beta) - 1$ with high probability ($1 - 1/m$, to be exact). Output z as the approximate solution to our combinatorial optimization problem. See [3] for how Farhi et al. obtained these values.

4 QAOA for Max-Cut

Recall that the classical Max-Cut problem maximizes

$$C(z) = \frac{1}{2} \sum_{(i,j) \in E} 1 - z_i z_j$$

The quantum analogue is

$$C = \frac{1}{2} \sum_{(i,j) \in E} I - \sigma_i^z \sigma_j^z$$

because σ^z has eigenvalues 1 and -1 and qubits z_i have state $|0\rangle$ or $|1\rangle$ to represent which set the vertex belongs to in the cut. To apply QAOA to Max-Cut we simply pass this C to QAOA as detailed in Section 3.

Why do local constraints help us? Recall that

$$F_p(\gamma, \beta) = \langle \gamma, \beta | \, C \, | \gamma, \beta \rangle = \langle \gamma, \beta | \sum_{<ij>} C_{<ij>} \, | \gamma, \beta \rangle = \sum_{<ij>} \langle \gamma, \beta | \, C_{<ij>} \, | \gamma, \beta \rangle.$$

Consider the case where $p = 1$. If we plug in our formula for $|\gamma, \beta\rangle$, each term in the summation is of the form

$$\langle s | \, U^\dagger(C, \gamma_1) U^\dagger(B, \beta_1) C_{<ij>} U(B, \beta_1) U(C, \gamma_1) \, | s \rangle.$$

Since $U(B, \beta_1) = e^{-i\beta_1 B} = e^{-i\beta_1 \sum_j \sigma_j^x}$ and $C_{<ij>}$ only involves qubits i and j, any σ^x that does not involve qubits i or j commutes through $C_{<ij>}$ and we are left with

$$\langle s | \, U^\dagger(C, \gamma_1) e^{i\beta_1(\sigma_i^x + \sigma_j^x)} C_{<ij>} e^{-i\beta_1(\sigma_i^x + \sigma_j^x)} U(C, \gamma_1) \, | s \rangle.$$

Similarly, terms in $U(C, \gamma_1)$ that do not depend on qubits i or j will commute through and cancel out. Our final expression, then, only involves qubits on edge $< ij >$ and edges adjacent to $< ij >$. For general p we can use a similar argument to show that the final expression only depends on qubits at most p edges away from edge $< ij >$. Thus our $F_p(\gamma, \beta)$ calculation only depends on p and does not grow with number of qubits n, allowing us to efficiently find γ and β with classical methods for Step 1 of the algorithm.

5 Key Results and Comparison with Classical Methods

The key result found in [3] was that QAOA with $p = 1$ achieves an approximation ratio of 0.6924 when performing Max-Cut on 3-regular graphs (in which every vertex has degree 3). The approximation ratio here is $C(z)/C_{max}$, where z is the output of QAOA and $C_{max} = \max_z C(z)$. Does this outperform the best classical algorithm? Unfortunately it does not. In 2002 Halperin et al. achieved a significantly better approximation ratio of 0.9326 with a semidefinite programming based approach [8]. There may still be *specific instances* of 3-regular graphs on which QAOA outperforms its classical counterparts, however.

Since QAOA was not the best algorithm for Max-Cut, Farhi looked for a combinatorial optimization problem on which it would be. He settled on the Max-3-XOR problem, in which we want to find the n-bitstring z that satisfies the maximal amount of m clauses of the form

$$z_i + z_j + z_k = c \mod 2$$

where $c \in \{0, 1\}$. In 2014 he succeeded in improving upon the best-known classical result by showing that QAOA with $p = 1$ could satisfy at least

$$\left(\frac{1}{2} + \frac{\mathcal{O}(1)}{d^{3/4}}\right) m$$

clauses, where d is the maximum number of clauses a single variable is allowed to appear in [4]. In response, in 2015 a team of some of the best classical theorists in the world including Berkeley's own Luca Trevisan and Prasad Raghavendra found a classical algorithm that satisfied at least

$$\left(\frac{1}{2} + \frac{\mathcal{O}(1)}{d^{1/2}}\right) m$$

clauses [1]. Furthermore, in 2001 Trevisan had showed that achieving a better approxi-

mation ratio than this is NP-hard [10]. Farhi soon after improved his analysis to show that QAOA with $p = 1$ satisfied

$$\left(\frac{1}{2} + \frac{\mathcal{O}(1)}{d^{1/2}\ln(d)}\right) m$$

clauses, but he was not able to match the classical result.

6 Utility of QAOA

Despite being thus far unable to improve upon classical results, will QAOA be a useful algorithm on the Noisy Intermediate-Scale Quantum (NISQ) computers of the near future? After benchmarking QAOA, Gavin Crooks from Rigetti Computing is optimistic that it will be [2]. Crooks used a quantum virtual machine (classical simulation of a quantum computer) to run QAOA on instances of Max-Cut with randomly generated graphs with 10 vertices. He optimized the gate parameters γ and β with stochastic gradient descent and compared QAOA with the Goemans-Williamson algorithm, a widely used polynomial-time classical algorithm for approximately solving Max-Cut. The results are in Figure 2, in which we can see that QAOA with $p = 5$ matches Goemans-Williamson's approximation ratio of about 0.94 and that QAOA with higher p values clearly outperforms Goemans-Williamson, approaching a perfect ratio of 1.0. Crooks also details how to construct the QAOA circuit for a linear array of n qubits with only $\mathcal{O}(n^2 p)$ gates, a feasible circuit size for NISQ computers.

Guerreschi and Matsuura from Intel are more pessimistic about QAOA [6]. Instead of focusing on the quality of the approximation like [2], they analyzed how soon we can expect a quantum speedup (i.e. at what number of qubits, if any, will QAOA take less time to solve Max-Cut than a classical solver). Their measure of time taken by QAOA is comprehensive of all tasks required to run QAOA from start to finish on an actual quantum device, including preparing the initial state, error correcting, repeating the experiment a

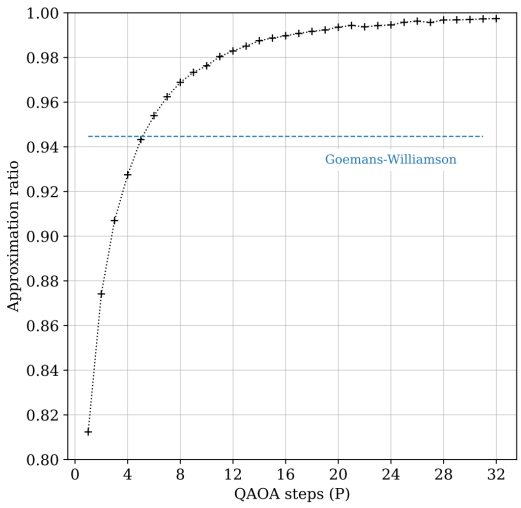

Figure 2: Average approximation ratio of QAOA with various values of p on Max-Cut with 10-vertex graphs.

sufficient amount of times, etc. They ran their experiments on a quantum virtual machine and extrapolated the data to conclude that we will need hundreds if not thousands of qubits to see a quantum speedup with QAOA (see Figure 3), which will be difficult to achieve in the near future.

The results from both papers are, of course, limited and largely speculative, as classical simulation of QAOA is only feasible on small graphs like the ones analyzed. We will only be able to test the validity of their conjectures after significant progress on NISQ devices.

7 Extension of QAOA: Quantum Alternating Operator Ansatz

As detailed in Section 3, the QAOA begins with the $|+ + \cdots +\rangle$ starting state and applies alternating unitaries $U(C, \gamma) = e^{-i\gamma C}$ and $U(B, \beta) = e^{-i\beta B}$, where B and C are the driver and problem Hamiltonians respectively. Hadfield et al. extend this to allow alternation between more general families of unitaries (i.e. not restricted to the form $e^{-i\gamma H}$ or dependent on a Hamiltonian) [7]. With the Quantum Alternating Operator Ansatz framework

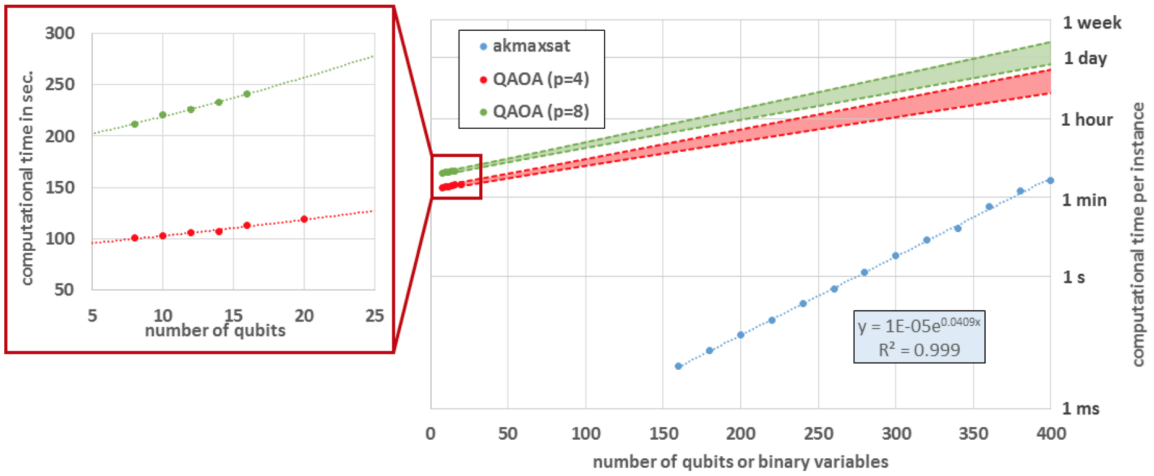

Figure 3: Computational time taken by QAOA (extrapolated from experiments on 8-20 qubits) versus time taken by AKMAXSAT (the chosen classical Max-Cut solver) to solve a single Max-Cut instance on a random 3-regular graph. The shaded areas are 95% confidence intervals based on linear regression. Quantum speedup is predicted to occur at the intersection of the AKMAXSAT line and the QAOA lines.

(which we call QAOA$_2$ for clarity), we begin in some starting state $|s\rangle$ and alternate between *phase-separation operators* $U_P(\gamma)$ and *mixing operators* $U_M(\beta)$. Thus running the QAOA$_2$ algorithm with some p (the number of times we apply the unitaries, as in regular QAOA) gives us the state

$$|\beta, \gamma\rangle = Q_p(\beta, \gamma)\,|s\rangle$$

where

$$Q_p(\beta, \gamma) = U_M(\beta_p)U_P(\gamma_p)\ldots U_M(\beta_1)U_P(\gamma_1).$$

QAOA$_2$ is a powerful generalization that allows us to, for instance, restrict the mixing operator to the feasible subspace of the problem and thereby allow better performance. The paper gives the appropriate $U_P(\gamma)$ and $U_M(\beta)$ for approximately solving a variety of problems including MaxColorableGraph, MaxIndependentSet, and the Traveling Salesman Problem. QAOA$_2$ extends the power of QAOA and is a promising candidate for solving tough problems heuristically as we enter the NISQ era.

8 Conclusion

In this paper we have summarized the Quantum Approximate Optimization Algorithm as originally presented as well as some more recent work on the subject. We broke down the algorithm and how it involves both quantum and classical techniques. We demonstrated how QAOA can be thought of as a discretization of adiabatic quantum computing. We described the type of problem QAOA is well-suited for as a heuristic algorithm, and how we can extend the types of problems we can tackle using the Quantum Alternating Operator Ansatz. We went over important theoretical results for both QAOA and its classical counterparts. Finally we discussed some conflicting predictions for how QAOA will perform on future NISQ devices in terms of quality of approximation and quantum speedup. All things considered, QAOA is a leading candidate for eventually being able to outperform classical computers in approximating solutions to NP-hard problems in areas as diverse as routing, machine scheduling, image recognition, and layout of electronic circuits [6].

References

[1] Boaz Barak et al. *Beating the random assignment on constraint satisfaction problems of bounded degree*. 2015. eprint: `arXiv:1505.03424`.

[2] Gavin E. Crooks. *Performance of the Quantum Approximate Optimization Algorithm on the Maximum Cut Problem*. 2018. eprint: `arXiv:1811.08419`.

[3] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014. eprint: `arXiv:quant-ph/1411.4028`.

[4] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm Applied to a Bounded Occurrence Constraint Problem*. 2014. eprint: `arXiv:1412.6062`.

[5] Edward Farhi et al. *Quantum Computation by Adiabatic Evolution*. 2000. eprint: arXiv:quant-ph/0001106.

[6] G. G. Guerreschi and A. Y. Matsuura. *QAOA for Max-Cut requires hundreds of qubits for quantum speed-up*. 2018. eprint: arXiv:1812.07589.

[7] Stuart Hadfield et al. *From the Quantum Approximate Optimization Algorithm to a Quantum Alternating Operator Ansatz*. 2017. eprint: arXiv:1709.03489.

[8] Eran Halperin, Dror Livnat, and Uri Zwick. *MAX CUT in Cubic Graphs*. 2002. URL: http://dl.acm.org/citation.%20cfm?id=545381.545449.

[9] Yin Sun et al. *Adiabatic Quantum Simulation Using Trotterization*. 2018. eprint: arXiv:1805.11568.

[10] Luca Trevisan. "Non-approximability Results for Optimization Problems on Bounded Degree Instances". In: *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*. STOC '01. Hersonissos, Greece: ACM, 2001, pp. 453–461. ISBN: 1-58113-349-9. DOI: 10.1145/380752.380839. URL: http://doi.acm.org/10.1145/380752.380839.