CSC2420 Spring 2015: Lecture 9

Allan Borodin

March 12, 2015

Announcements and todays agenda

- Announcements
- The first 4 questions of the final assignment are now posted.
- Todays agenda
  1. Review discusion of naive randomized algorithm for Max Sat and its derandomization (i.e. Johnson's algorithm).
  2. Getting past $\frac{2}{3}$ for Max Sat by greedy-like algorithms.
  3. The Buchbinder double sided algorithm for the Unconstrained Non-Monotone Submodular Maximzation and application to Max Sat.
  4. Randomized rounding of LP and SDP for Max-2-Sat.
  5. 2-SAT and Random walks

The naive randomized algorithm for exact Max-k-Sat

We continue our discussion of randomized algorthms by considering the use of randomization for improving approximation algorithms. In this context, randomization can be (and is) combined with any type of algorithm. **Warning**: For the following discussion of Max-Sat, we will follow the prevailing convention by stating approximation ratios as fractions $c < 1$.

- Consider the exact Max-k-Sat problem where we are given a CNF propositional formula in which every clause has exactly k literals. We consider the weighted case in which clauses have weights. The goal is to find a satisfying assignment that maximizes the size (or weight) of clauses that are satisfied.
- Since exact Max-k-Sat generalizes the exact k- SAT decision problem, it is clearly an NP hard problem for $k \geq 3$. It is interesting to note that while 2-SAT is polynomial time computable, Max-2-Sat is still NP hard.
- The naive randomized (online) algorithm for Max-k-Sat is to randomly set each variable to *true* or *false* with equal probability.

Analysis of naive Max-k-Sat algorithm continued

- Since the expectation of a sum is the sum of the expectations, we just have to consider the probability that a clause is satisfied to determine the expected weight of a clause.
- Since each clause C_i has k variables, the probability that a random assignment of the literals in C_i will set the clause to be satisfied is exactly $\frac{2^k-1}{2^k}$. Hence $\mathbf{E}$ [weight of satisfied clauses] $= \frac{2^k-1}{2^k} \sum_i w_i$
- Of course, this probability only improves if some clauses have more than k literals. It is the small clauses that are the limiting factor in this analysis.
- This is not only an approxination ratio but moreover a "totality ratio" in that the algorithms expected value is a factor $\frac{2^k-1}{2^k}$ of the sum of all clause weights whether satisfied or not.
- We can hope that when measuring againt an optimal solution (and not the sum of all clause weights), small clauses might not be as problematic as they are in the above analysis of the naive algorithm.

Derandomizing the naive algorithm

We can derandomize the naive algorithm by what is called the method of conditional expectations. Let $F[x_1, \ldots, x_n]$ be an exact k CNF formula over n propositional variables $\{x_i\}$. For notational simplicity let $true = 1$ and $false = 0$ and let $w(F)|\tau$ denote the weighted sum of satisfied clauses given truth assignment τ.

- Let x_j be any variable. We express $\mathbf{E}[w(F)|_{x_i \in_u \{0,1\}}]$ as $\mathbf{E}[w(F)|_{x_i \in_u \{0,1\}} | x_j = 1] \cdot (1/2) + \mathbf{E}[w(F)|_{x_i \in_u \{0,1\}} | x_j = 0] \cdot (1/2)$
- This implies that one of the choices for x_j will yield an expectation at least as large as the overall expectation.
- It is easy to determine how to set x_j since we can calculate the expectation clause by clause.
- We can continue to do this for each variable and thus obtain adeterministic solution whose weight is at least the overall expected value of the naive randomized algorithm.
- NOTE: The derandomization can be done so as to achieve an online algorithm (i.e. within the priority model with adversarial order).

(Exact) Max-k-Sat

- For exact Max-2-Sat (resp. Max-3-Sat), the approximation (and totality) ratio is $\frac{3}{4}$ (resp. $\frac{7}{8}$).
- For $k \geq 3$, using PCPs (probabilistically checkable proofs), Hastad proves that it is NP-hard to improve upon the $\frac{2^k-1}{2^k}$ approximation ratio for Max-k-Sat.
- For Max-2-Sat, the $\frac{3}{4}$ ratio can be improved (as we will see) by the use of semi-definite programming (SDP).
- The analysis for exact Max-k-Sat clearly needed the fact that all clauses have at least k clauses. What bound does the naive online randomized algorithm or its derandomztion obtain for (not exact) Max-2-Sat or arbitrary Max-Sat (when there can be unit clauses)?

Johnson's Max-Sat Algorithm

Johnson's [1974] algorithm

```
For all clauses $C_i$, $w'_i := w_i/(2^{|C_i|})$
Let $L$ be the set of clauses in formula $F$ and $X$ the set of variables
**For** $x \in X$ (or until $L$ empty)
   Let $P = \{C_i \in L$ such that $x$ occurs positively$\}$
   Let $N = \{C_j \in L$ such that $x$ occurs negatively$\}$
   **If** $\sum_{C_i \in P} w'_i \geq \sum_{C_j \in N} w'_j$
     $x := true; L := L \setminus P$
     **For all** $C_r \in N, \quad w'_r := 2w'_r$   **End For**
   **Else**
     $x := false; L := L \setminus N$
     **For all** $C_r \in P, \quad w'_r := 2w'_r$   **End For**
   **End If**
   Delete $x$ from $X$
**End For**
```

Johnson's algorithm is the derandomized algorithm

- Twenty years after Johnson's algorithm, Yannakakis [1994] presented the naive algorithm and showed that Johnson's algorithm is the derandomized naive algorithm.
- Yannakakis also observed that for arbitrary Max-Sat, the approximation of Johnson's algorithm is at best $\frac{2}{3}$. For example, consider the 2-CNF $F = (x \vee \bar{y}) \wedge (\bar{x} \vee y) \wedge \bar{y}$ when variable x is first set to true.
- Chen, Friesen, Zheng [1999] showed that Johnson's algorithm achieves approximation ratio $\frac{2}{3}$ for arbitrary weighted Max-Sat.
- For arbitrary Max-Sat (resp. Max-2-Sat), the current best approximation ratio is .797 (resp. .931) using semi-definite programming and randomized rounding.

Modifying Johnson's algorithm for Max-Sat

- In proving the (2/3) approximation ratio for Johnson's Max-Sat algorithm, Chen et al asked whether or not the ratio could be improved by using a random ordering of the propositional variables (i.e. the input items). We can study online and ROM problems within the priority algorithm framework.
- To precisely model the Max-Sat problem within the priority framework, we need to specify the input model.
- In increasing order of providing more information (and possibly better approximation ratios), the following input models can be considered:
  1. Each propositional variable x is represented by the length of each clause C_i in which x appears positively, and for each clause C_j in which it appears negatively. This is sufficient for the naive algorithm and its derandomization (Johnson's algorithm).
  2. In addition, for each C_i and C_j, a list of the other variables in that clause is specified.
  3. The variable x is reprsented by a complete specification of each clause it which it appears.

Improving on Johnson's algorithm

- The question asked by Chen et al was answered by Costello, Shapira and Tetali [2011] who showed that in the ROM model, Johnson's algorithm achieves approximation $(2/3 + \epsilon)$ for $\epsilon \approx .003653$
- Poloczek and Schnitger [same SODA 2011 conference] show that the approximation ratio for Johnsons algorithm in the online model is at most $2\sqrt{157} \approx .746 < 3/4$, where the 3/4 ratio is obtained by Yannakakis' IP/LP approximation that we will soon present.
- Poloczek and Schnitger first consider a "canonical randomization" of Johnson's algorithm"; namely, the canonical randomization sets a variable $x_i = \textit{true}$ with probability $\frac{w_i'(P)}{w_i'(P)+w_i'(N)}$ where $w_i'(P)$ (resp. $w_i'(N)$) is the current combined weight of clauses in which x_i occurs positively (resp. negatively). Their substantial additional idea is to adjust the random setting so as to better account for the weight of unit clauses in which a variable occurs.

A few comments on the Poloczek and Schnitger algorithm

- The Poloczek and Schnitger algorithm is called Slack and has approximation ratio $= 3/4$.
- In terms of priority algorithms this is a randomized online algorithm (i.e. adversary chooses the ordering) where the variables are represented in the second (middle power) input model.
- This approximation ratio is in contrast to Azar et al [2011] who prove that no randomized online algorithm can achieve approximation better than $2/3$ when the input model is the weakest of the input models.
- Finally (in this regard), Poloczek [2011] shows that no deterministic priority algorithm can achieve a $3/4$ approximation within the second (middle) input model. This provides a strong sense in which to claim the Poloczek and Schnitger Slack algorithm "cannot be derandomized".
- The best deterministic priority algorithm in the third (most powerful) model remains an open problem as does the best randomized priority algorithm.

Submodular maximization problems; A small diversion before returning to MaxSat

- A set function $f : 2^U \rightarrow \Re$ is submodular if $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$ for all $S, T \subseteq U$.
- Equivalently, f is submodular if it satisfies decreasing marginal gains; that is, $f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T)$ for all $S \subseteq T \subseteq U$ and $x \in U$
- We will always assume that f is *normalized* in that $f(\varnothing) = 0$.
- Submodular functions arise naturally in many applications and has been a topic of much recent activity.
- Probably the most frequent application of (and papers about) submodular functions is when the function is also monotone (non-decreasing) in that $f(S) \leq f(T)$ for $S \subseteq T$.
- Note that linear functions (also called modular) functions are a special case of monotone submodular functions.

Submodular maximization continued

In the submodular maximization problem, we want to compute S so as to maximize $f(S)$.

- For monotone functions, we are maximizing $f(S)$ subject to some constraint (otherwise just choose $S = U$).
- For the non monotone case, the problem is already interesting in the unconstrained case. Perhaps the most prominent example of such a problem is Max-Cut (and Max-Di-Cut).
- Max-Cut is an NP-hard problem. Using an SDP approach just as we will see for the Max-2-Sat problem yields the approximation ratio $\alpha = \frac{2}{\pi} \min_{\{0 \leq \theta \leq \pi\}} \frac{\theta}{(1-cos(\theta)}$ $\approx .87856$. Assuming UGC, this is optimal.
- For a submodular function, we may be given an explicit representation (when a succinct representation is possible as in Max-Cut) or we access the function by an oracle such as the *value oracle* which given S, outputs the value $f(S)$ and such an oracle call is considered to have $O(1)$ cost. Other oracles are possible (e.g. given S, output the element x of U that maximizes $f(S \cup \{x\}) - f(S)$).

Unconstrained (non monotone) submodular maximization

- Feige, Mirrokni and Vondrak [2007] began the study of approximation algorithms for the unconstrained non monotone submodular maximization (USM) problem establishing several results:
    1. Choosing S uniformly at random provides a $1/4$ approximation.
    2. An oblivious local search algorithm results in a $1/3$ approximation.
    3. A non-oblivious local search algorithm results in a $2/5$ approximation.
    4. Any algorithm using only value oracle calls, must use an exponential number of calls to achieve an approximation $(1/2 + \epsilon)$ for any $\epsilon > 0$.
- The Feige et al paper was followed up by improved local search algorithms by Gharan and Vondrak [2011] and Feldman et al [2012] yielding (respectively) approximation ratios of .41 and .42.
- The $(1/2 + \epsilon)$ inapproximation was augmented by Dobzinski and Vondrak showing the same bound for an explicitly given instance under the assumption that $RP \neq NP$.

The Buchbinder et al (1/3) and (1/2) approximations for USM

In the FOCS [2012] conference, Buchbinder et al gave an elegant linear time deterministic 1/3 approximation and then extend that to a randomized 1/2 approximization. The conceptually simple form of the algorithm is (to me) as interesting as the optimality (subject to the proven inapproximation results) of the result. Let $U = u_1, \dots u_n$ be the elements of U *in any order*.

The deterministic 1/3 approximation for USM

```
$X_0 := \emptyset$; $Y_0 := U$
For $i := 1 \dots n$
  $a_i := f(X_{i-1} \cup \{u_i\}) - f(X_{i-1})$; $b_i := f(Y_{i-1} \setminus \{u_i\}) - f(Y_{i-1})$
  **If** $a_i \geq b_i$
    **then** $X_i := X_{i-1} \cup \{u_i\}$; $Y_i := Y_{i-1}$
    **else** $X_i := X_{i-1}$; $Y_i := Y_{i-1} \setminus \{u_i\}$
  **End If**
**End For**
```

The randomized 1/2 approximation for USM

- Buchbinder et al show that the “natural randomization” of the previous deterministic algorithm achieves approximation ratio $1/2$.
- That is, the algorithm chooses to either add $\{u_i\}$ to X_{i-1} with probability $\frac{a'_i}{a'_i+b'_i}$ or to delete $\{u_i\}$ from Y_{i-1} with probability $\frac{b'_i}{a'_i+b'_i}$ where $a'_i = \max\{a_i, 0\}$ and $b'_i = \max\{b_i, 0\}$.
- If $a_i = b_i = 0$ then add $\{u_i\}$ to X_{i-1}.
- Note: Part of the proof for both the deterministic and randomized algorithms is the fact that $a_i + b_i \geq 0$.

Applying the algorithmic idea to Max-Sat

Buchbinder et al are able to adapt their randomized algorithm to the Max-Sat problem (and even to the Submodular Max-Sat problem). So assume we have a monotone normalized submodular function f (or just a linear function as in the usual Max-Sat). The adaption to Submodular Max-Sat is as follows:

- Let $\phi : X \rightarrow \{0\} \cup \{1\} \cup \varnothing$ be a standard partial truth assignment. That is, each variable is assigned exactly one of two truth values or not assigned.
- Let $\mathcal{C}$ be the set of clauses in formula Ψ. Then the goal is to maximize $f(\mathcal{C}(\phi))$ where $\mathcal{C}(\phi)$ is the sat of formulas satisfied by ϕ.
- An extended assignment is a function $\phi' : X \rightarrow 2^{\{0,1\}}$. That is, each variable can be given one, two or no values. (Equivalently $\phi' \subseteq X \times \{0,1\}$ is a relation.) A clause can then be satisfied if it contains a positive literal (resp. negative literal) and the corresponding variable has value $\{1\}$ or $\{0,1\}$ (resp. has value $\{0\}$ or $\{0,1\}$.
- $g(\phi') = f(\mathcal{C}(\phi'))$ is a monotone normalized submodular function. ‘

Buchbinder et al Submodular Max-Sat

Now starting with $X_0 = X \times \varnothing$ and $Y_0 = Y \times \{0,1\}$, each variable is considered and set to either 0 or to 1 (i.e. a standard assignment of precisely one truth value) depending on the marginals as in USM problem.

Algorithm 3: RandomizedSSAT(f, Ψ)

```
1  $X_0 \leftarrow \emptyset$, $Y_0 \leftarrow \mathcal{N} \times \{0,1\}$.
2  for $i$ = 1 to $n$ do
3      $a_{i,0} \leftarrow g(X_{i-1} \cup \{u_i, 0\}) - g(X_{i-1})$.
4      $a_{i,1} \leftarrow g(X_{i-1} \cup \{u_i, 1\}) - g(X_{i-1})$.
5      $b_{i,0} \leftarrow g(Y_{i-1} \setminus \{u_i, 0\}) - g(Y_{i-1})$.
6      $b_{i,1} \leftarrow g(Y_{i-1} \setminus \{u_i, 1\}) - g(Y_{i-1})$.
7      $s_{i,0} \leftarrow \max\{a_{i,0} + b_{i,1}, 0\}$.
8      $s_{i,1} \leftarrow \max\{a_{i,1} + b_{i,0}, 0\}$.
9      with probability $s_{i,0}/(s_{i,0} + s_{i,1})^*$ do:
       $X_i \leftarrow X_{i-1} \cup \{u_i, 0\}$, $Y_i \leftarrow Y_{i-1} \setminus \{u_i, 1\}$.
10     else (with the compliment probability
       $s_{i,1}/(s_{i,0} + s_{i,1})$) do:
11     $X_i \leftarrow X_{i-1} \cup \{u_i, 1\}$, $Y_i \leftarrow Y_{i-1} \setminus \{u_i, 0\}$.
12 return $X_n$ (or equivalently $Y_n$).
```

* If $s_{i,0} = s_{i,1} = 0$, we assume $s_{i,0}/(s_{i,0} + s_{i,1}) = 1$.

Concluding discussion of Submodular Max-Sat

- The algorithm is shown to have a $\frac{3}{4}$ approximation ratio for Monotone Submodular Max-Sat.
- In the paper, they claim that for the standard Max-Sat (i.e. when the function f is a linear function), that the algorithm can be made to run in linear time.
- Poloczek et al show that the algorithm turns out to be equivalent to a previous Max-Sat algorithm by van Zuylen.
- In fact, as observed by Poloczek, the algorithm can be de-randomized to a two pass determinstic algorithm (running in linear time).
  1. The first pass sets a probability for each variable rather than making an irrevocable assignment.
  2. The second pass then uses the same order of the variables derandomize the algorithm using the method of conditional probabilities.
- We can view this as a "2-pass online algorithm" in the priority framework using the middle strength input model.

Yannakakis' IP/LP *randomized rounding* algorithm for Max-Sat

- We will formulate the weighted Max-Sat problem as a $\{0,1\}$ IP.
- Relaxing the variables to be in $[0,1]$, we will treat some of these variables as probabilities and then round these variables to 1 with that probability.
- Let F be a CNF formula with n variables $\{x_i\}$ and m clauses $\{C_j\}$. The Max-Sat formulation is :
  maximize $\sum_j w_j z_j$
  subject to $\sum_{\{x_i \text{ is in } C_j\}} y_i + \sum_{\{\bar{x}_i \text{ is in } C_j\}} (1-y_i) \geq z_j$
  $y_i \in \{0,1\}$; $z_j \in \{0,1\}$
- The y_i variables correspond to the propositional variables and the z_j correspond to clauses.
- The relaxation to an LP is $y_i \geq 0$; $z_j \in [0,1]$. Note that here we cannot simply say $z_j \geq 0$.

Randomized rounding of the y_i variables

- Let $\{y_i^*\}, \{z_j^*\}$ be the optimal LP solution,
- Set $\tilde{y}_i = 1$ with probability y_i^*.

Theorem

Let C_j be a clause with k literals and let $b_k = 1 - (1 - \frac{1}{k})^k$. Then $Prob[C_j$ is satisifed $]$ is at least $b_k z_j^*$.

- The theorem shows that the contribution of the j^{th} clause C_j to the expected value of the rounded solution is at least $b_k w_j$.
- Note that b_k converges to (and is always greater than) $1 - \frac{1}{e}$ as k increases. It follows that the expected value of the rounded solution is at least $(1 - \frac{1}{e})$ LP-OPT $\approx .632$ LP-OPT.
- Taking the max of this IP/LP and the naive randomized algorithm results in a $\frac{3}{4}$ approximation algorithm that can be derandomized.

The quadratic program for Max-2-Sat

- We introduce {-1,1} variables y_i corresponding to the propositional variables. We also introduce a homogenizing variable y_0 which will correspond to a constant truth value. That is, when $y_i = y_0$, the intended meaning is that x_i is set *true* and *false* otherwise.
- We want to express the $\{-1, 1\}$ truth value $val(C)$ of each clause C in terms of these $\{-1, 1\}$ variables.
    1. $val(x_i) = (1 + y_i y_0)/2$
       $val(\bar{x}_i) = (1 - y_i y_0)/2$
    2. If $C = (x_i \vee x_j)$, then $val(C) = 1 - val(\bar{x}_i \wedge \bar{x}_j) = 1 - (\frac{1-y_i y_0}{2})(\frac{1-y_j y_0}{2}) = (3 + y_i y_0 + y_j y_0 - y_i y_j)/4 = \frac{1+y_0 y_i}{4} + \frac{1+y_0 y_j}{4} + \frac{1-y_i y_j}{4}$
    3. If $C = (\bar{x}_i \vee x_j)$ then $val(C) = (3 - y_i y_0 + y_j y_0 + y_i y_j)/4$
    4. If $C = (\bar{x}_i \vee \bar{x}_j)$ then $val(C) = (3 - y_i y_0 - y_j y_0 - y_i y_j)/4$

The quadratic program for Max-2-Sat continued

- The Max-2-Sat problem is then to maximize $\sum w_k val(C_k)$ subject to $(y_i)^2 = 1$ for all i
- By collecting terms of the form $(1 + y_i y_j)$ and $(1 - y_i y_j)$ the max-2-sat objective can be represented as the strict quadratic objective: $\max \sum_{0 \leq i < j \leq n} a_{ij}(1 + y_i y_j) + \sum b_{ij}(1 - y_i y_j)$ for some appropriate a_{ij}, b_{ij}.
- Like an IP this integer quadratic program cannot be solved efficiently.

The vector program relaxation for Max-2-Sat

- We now relax the quadratic program to a vector program where each y_i is now a unit length vector $\mathbf{v}_i$ in $\Re^{n+1}$ and scalar multiplication is replaced by vector dot product. This vector program can be (approximately) efficiently solved (i.e. in polynomial time).
- The randomized rounding (from $\mathbf{v}_i^*$ to y_i) proceeds by choosing a random hyperplane in $\Re^{n+1}$ and then setting $y_i = 1$ iff $\mathbf{v}_i^*$ is on the same side of the hyperplane as $\mathbf{v}_0^*$. That is, if $\mathbf{r}$ is a uniformly random vector in $\Re^{n+1}$, then set $y_i = 1$ iff $\mathbf{r} \cdot \mathbf{v}_i^* \geq 0$.
- The rounded solution then has expected value $2\sum a_{ij} Prob[y_i = y_j] + \sum b_{ij} Prob[y_i \neq y_j]$; $Prob[y_i \neq y_j] = \frac{\theta_{ij}}{\pi}$ where θ_{ij} is the angle between $\mathbf{v}_i^*$ and $\mathbf{v}_j^*$.

The approximation ratio (in expectation) of the rounded solution

Let $\alpha = \frac{2}{\pi} \min_{\{0 \leq \theta \leq \pi\}} \frac{\theta}{(1-cos(\theta))} \approx .87856$ and let OPT_{VP} be the value obtained by an optimal vector program solution.
Then $\mathbf{E}[\text{rounded solution}] \geq \alpha \cdot (OPT_{VP})$.

The random walk algorithm for 2-Sat

- First, here is the idea of the deterministic polynomial time algorithm for 2-Sat: We can first eliminate all unit clauses. We then reduce the problem to the directed $s - t$ path problem. We view each clause $(x \vee y)$ in F as two directed edges $(\bar{x}, y)$ and $(\bar{y}, x)$ in a graph G_F whose nodes are all possible literals x and $\bar{x}$. Then the formula is satisfiable iff there does not exist a variable x such that there are paths from x to $\bar{x}$ and from $\bar{x}$ to x in G_F.
- There is also a randomized algorithm for 2-SAT (due to Papadimitriou [1991]) based on a random walk on the line graph with nodes $\{0, 1, , n\}$. We view being on node i as having a truth assignment τ that is Hamming distance i from some fixed satisfying assignment τ^* if such an assignment exists (i.e. F is satisfiable).
- Start with an arbitrary truth assignment τ and if $F(\tau)$ is true then we are done; else find an arbitrary unsatisfied clause C and randomly choose one of the two variables x_i occurring in C and now change τ to τ' by setting $\tau'(x_i) = 1 - \tau(x_i)$.

The expected time to reach a satisfying assignment

- When we randomly select one the the two literals in C and complement it, we are getting close to τ^* (i.e. moving one edge closer to node 0 on the line) with probability at least $\frac{1}{2}$. (If it turns out that both literal values disagree with τ^*, then we are getting closer to τ^* with probability $= 1$.)
- As we are proceeding in this random walk we might encounter another satisfying assignment which is all the better.
- It remains to bound the expected time to reach node 0 in a random walk on the line where on each random step, the distance to node 0 is reduced by 1 with probability at least $\frac{1}{2}$ and otherwise increased by 1 (but never exceeding distance n). This perhaps biased random walk is at least as good as the case where we randomly increase or decrease the distance by 1 with probability equal to $\frac{1}{2}$.

Claim:

The expected time to hit node 0 is at most $2n^2$.

- To prove the claim one needs some basic facts about Markov chains.

The basics of finite Markov chains

- A finite Markov chain M is a discrete-time random process defined over a set of states S and a matrix $P = \{P_{ij}\}$ of transition probabilities.
- Denote by X_t the state of the Markov chain at time t. It is a memoryless process in that the future behavior of a Markov chain depends only on its current state: $Prob[X_{t+1} = j | X_t = i] = P_{ij}$ and hence $Prob[X_{t+1} = j] = \sum_i Prob[X_{t+1} = j | X_t = i] Prob[X_t = i]$.
- Given an initial state i, denote by r_{ij}^t the probability that the first time the process reaches state j occurs at time t; $r_{ij}^t = Pr[X_t = j \text{ and } X_s \neq j \text{ for } 1 \leq s \leq t-1 | X_0 = i]$
- Let f_{ij} the probability that state j is reachable from initial state i; $f_{ij} = \sum_{t>0} r_{ij}^t$.
- Denote by h_{ij} the expected number of steps to reach state j starting from state i (hitting time); that is, $h_{ij} = \sum_{t>0} t \cdot r_{ij}^t$
- Finally, the *commute time* c_{ij} is the expected number of steps to reach state j starting from state i, and then return to i from j; $c_{ij} = h_{ij} + h_{ji}$

Stationary distributions

- Define $\mathbf{q}^t = (q_1^t, q_2^t, \ldots, q_n^t)$, the state probability vector (the distribution of the chain at time t), as the row vector whose i-th component is the probability that the Markov chain is in state i at time t.
- A distribution π is a stationary distribution for a Markov chain with transition matrix P if $\pi = \pi P$.
- Define the underlying directed graph of a Markov chain as follows: each vertex in the graph corresponds to a state of the Markov chain and there is a directed edge from vertex i to vertex j iff $P_{ij} > 0$. A Markov chain is *irreducible* if its underlying graph consists of a single strongly connected component. We end these preliminary concepts by the following theorem.

Theorem: Existence of a stationary distribution

For any finite, irreducible and aperiodic Markov chain,

(i) There exists a *unique* stationary distribution π.

(ii) For all states i, $h_{ii} < \infty$, and $h_{ii} = 1/\pi_i$.

Back to random walks on graphs

- Let $G = (V, E)$ be a connected, non-bipartite, undirected graph with $|V| = n$ and $|E| = m$. A uniform random walk induces a Markov chain M_G as follows: the states of M_G are the vertices of G; and for any $u, v \in V$, $P_{uv} = 1/deg(u)$ if $(u, v) \in E$, and $P_{uv} = 0$ otherwise.
- Denote by $(d_1, d_2, \ldots, d_n)$ the vertex degrees. M_G has a stationary distribution $(d_1/2m, \ldots, d_n/2m)$.
- Let $C_u(G)$ be the expected time to visit every vertex, starting from u and define $C(G) = \max_u C_u(G)$ to be the *cover time* of G.

Theorem: Aleliunas et al [1979]

Let G be a connected undirected graph. Then

1. For each edge (u, v), $C_{u,v} \leq 2m$,
2. $C(G) \leq 2m(n-1)$.

- It follows that the 2-SAT random walk has expected time at most $2n^2$. to find a satisfying assignment in a satisfiable formula. Can use Markov inequality to obtain probability of not finding satisfying assignment.