1 May 2019

Randomized Approximation for Max-Cut

Upcoming schedule:

Wed	MAX CUT
Fri	PRIMALITY
Mon	MIN CUT

Randomization in algorithms: why does it help?

Which of these sorting algorithms improves its worst-case running time if you use randomness?

(A) Mergesort (B) Heapsort $\boxed{\text{(C) Quicksort}}$

Why does randomness help?

(1) Guard against pathological inputs by randomizing behavior to reduce probability of a "worst-case interaction between input & algorithm."

(2) Sub-sample the input data to reduce its size while (approximately) preserving the property you want to analyze.

(3) Reduce the complexity of implementing operations inside the algorithm by substituting a "randomized approximation" that's cheaper to compute.
E.g. SGD versus GD.

This lecture: Example of (1) illustrated using MAX CUT.
Given undirected $G=(V,E)$ and weights $w(e) \geq 0$ $\forall e \in E$, partition V into A, B to maximize

$$w(A,B) = \sum_{e \in E(A,B)} w(e) \qquad E(A,B) \triangleq \{\text{edges from } A \text{ to } B\}.$$

This is NP-complete. (Exercise. Warning: challenging!)

Embarrassingly simple randomized algorithm (ESRA).

- output a uniformly random partition of V.

Analysis. We will show $\mathbb{E}[w(A,B)] = \frac{1}{2} \sum_{e \in E} w(e) \geq \frac{1}{2} \cdot OPT$.

Therefore ERSA is a 2-approx in expectation.

(Repeat with indep. randomness and take best solution found if you want to boost success probability.)

Lemma: For every edge $e = (u,v)$, $\Pr(e \in E(A,B)) = \frac{1}{2}$.

Proof. The following 4 events each have probability $\frac{1}{4}$.

$$u \in A, \quad v \in A$$

$$\left.\begin{array}{ll} u \in A, & v \in B \\ u \in B, & v \in A \end{array}\right\} (u,v) \in E(A,B)$$

$$u \in B, \quad v \in B$$

$$\Pr\big((u,v) \in E(A,B) \big) = \frac{2}{4} = \frac{1}{2}.$$

Linearity of expectation

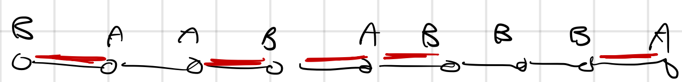
$$\mathbb{E}[w(A,B)] = \mathbb{E}\Big[\sum_{e \in E(A,B)} w(e) \Big]$$

$$= \sum_{e \in E} w(e) \cdot \Pr[e \in E(A,B)]$$

$$= \frac{1}{2} \sum_{e \in E} w(e) \qquad \text{QED.}$$

Conclusion ESRA gets $\geq \frac{1}{2} \cdot OPT$ in expectation.

E.g. G is a path



DERANDOMIZATION: Modifying a randomized algorithm to simulate the same (or qualitatively as good) behavior deterministically.

METHOD OF CONDITIONAL EXPECTATIONS. Fix the algorithm's random bits, one at a time, substituting a deterministic value for the random one, greedily optimizing the conditional expected outcome.

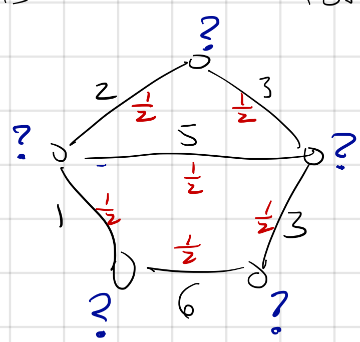
In step t, when deciding whether t^{th} rand bit should be $\emptyset$ or 1, calculate

$$E[\text{solution quality} \mid \text{set bit to } \emptyset]$$
$$E[\text{———} \mid \text{set bit to } 1]$$

Take better of these two alternatives.

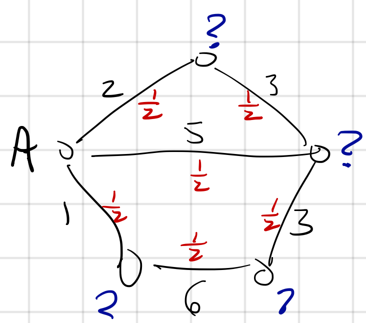
How this looks for MAX CUT:

$t=0$

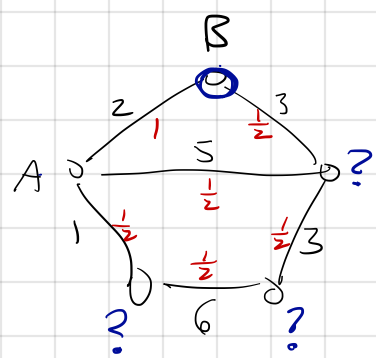


? means random
A means vertex $\in A$
B means vertex $\in B$

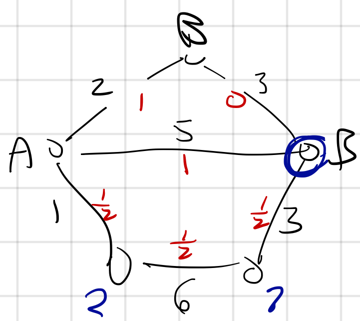
$t=1$



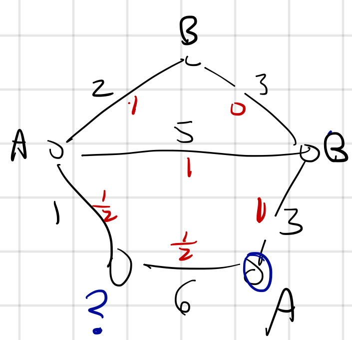
$t=2$



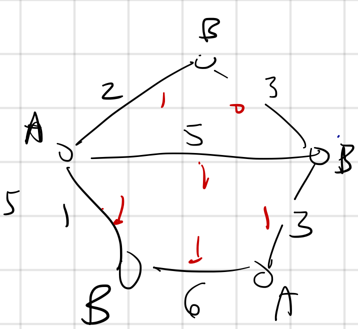
$t=3$



$t=4$



$t=5$



Simpler description of the algorithm: iterate through the vertices, labeling each one to maximize the number of neighbors already assigned the opposite label.