CS 4980: Capstone Research Notes

Professor Mark Floryan,
compiled by Grady Hollar

Fall 2025

1 Basic Definitions

What is an approximation algorithm?

2 Basic Examples

Vertex cover, set cover?

3 The Traveling Salesperson Problem

3.1 Metric TSP

Optimization Problem (Metric-TSP) Given a complete undirected graph $G = (V, E)$ and cost function $c : E \to \mathbb{Q}^+$ such that the triangle inequality holds for all $u, v, w \in V$:

$$c(u, v) \leq c(u, w) + c(w, v),$$

find a minimum cost cycle in G visiting every vertex exactly once.

3.2 Inapproximability of General TSP

4 Randomization and LP Methods

4.1 A simple MAX-3SAT algorithm

Let's recall the most fundamental NP-complete problem: the *satisfiability* (or *SAT*) *problem.* Recall that a Boolean formula in k-conjunctive normal form (k-CNF) consists of clauses connected by logical and's, where each clause consists of exactly k distinct literals connected by logical or's. For example,

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee \neg x_4 \vee x_5) \wedge (\neg x_3 \vee x_4 \vee x_6) \wedge (x_1 \vee x_3 \vee x_5)$$

is a 3-CNF formula with 4 clauses and 6 variables. Given a CNF formula, the SAT problem asks whether we can find an assignment for its variables such that the formula evaluates to true.

In the optimization version of this decision problem, we'll instead focus on trying to satisfy as many clauses in the formula as possible. We'll also concern ourselves only with 3-CNF formulas, since the general SAT problem can be reduced to 3-SAT (source?).

Aside: In classical literature, the definition of a k-CNF formula only requires each clause to contain *at most* k literals, not exactly k. Adding in this extra condition as we have is formally known as MAX-EkSAT, with the E standing for "exactly". It turns out that the usual formal version of the MAX-3SAT problem also has a . The interested reader may wish to look up the "Karloff-Zwick algorithm".

Optimization Problem (MAX-3SAT) Given a 3-CNF formula ϕ, find an assignment of ϕ that satisfies the largest number of clauses.

How good of a solution could we get to the above problem by setting each variable in the given formula completely randomly? We'll soon see that, on average, this simple approach will surprisingly get us close to an optimal solution—quite close, in fact! But what do we mean by "on average"? We need to formalize what it means to have an approximation factor when our algorithm involves an element of randomization.

Definition (Randomized Approximation Algorithm). We say that a randomized algorithm for an optimization problem has an approximation ratio of $\rho(n)$ if, for any input of size n, the *expected cost* of the solution produced by the randomized algorithm is within a factor of $\rho(n)$ of the cost of an optimal solution.

To see how we calculate this expected cost in practice, let's prove that our proposed algorithm to the 3-SAT problem is, interestingly, an 8/7-approximation for the MAX-3SAT problem.

```
Algorithm APPROX-MAX-3SAT($\phi$)
  for $x_i \in \phi$ do
      Assign $x_i$ randomly to be 0 or 1 with equal probabilities 1/2
  end for
  return the assignment
```

Theorem 2.1. Approx-MAX-3SAT is an 8/7-approximation for MAX-3SAT.

Proof. Let ϕ be a 3-CNF formula with n variables $x_1, x_2, \ldots, x_n$ and m clauses. For $1 \leq i \leq m$, define the random variable

$$Y_i = \begin{cases} 1, & \text{clause } i \text{ is satisfied} \\ 0, & \text{otherwise} \end{cases}.$$

Then, the number of clauses satisfied overall is modeled by the random variable defined by

$$Y = \sum_{i=1}^{m} Y_i.$$

So, the expected cost of the algorithm will be exactly $E[Y]$. Since each literal is set to 1 with probability 1/2 and 0 with probability 1/2, and a clause is not satisfied only if all three of its literals are set to 0, we have

$$P[Y_i = 0] = (1/2)^3 = 1/8 \quad \Longrightarrow \quad P[Y_i = 1] = 1 - 1/8 = 7/8.$$

Computing $E[Y_i]$ then gives us

$$E[Y_i] = 1 \cdot 7/8 + 0 \cdot 1/8 = 7/8.$$

Now we can use the familiar properties of expected values along with the above calculations to see that

$$\begin{aligned} E[Y] &= E\left[\sum_{i=1}^{m} Y_i\right] \\ &= \sum_{i=1}^{m} E[Y_i] \\ &= \sum_{i=1}^{m} 7/8 \\ &= 7m/8. \end{aligned}$$

Since the maximum amount of clauses that can be satisfied in ϕ is m, the approximation ratio is at most $m/(7m/8) = 8/7$. □

The keen reader may have realized that in our proof we have not addressed an important possibility: what if a clause contains both a variable and its negation? Surely this will change the probabilities we calculated, and in turn the approximation factor. Let's show that, in fact, there is no need for such worries.

Proposition 2.2. Without assuming no clause contains a variable and its negation, Algorithm 1 is *still* an 8/7-approximation for MAX-3SAT.

Proof. Suppose that k of ϕ's clauses contain both a variable and its negation, and m clauses do not. For each of these remaining m clauses, we'll define the same random variable Y_i as we did before. Now, notice that for each of the k clauses containing a variable and its negation, the clause will be satisfied *no matter what assignment* for ϕ is chosen. So, the total number of satisfied clauses in ϕ is exactly

$$Y = Y_1 + Y_2 + \cdots + Y_m + k.$$

The calculation of each $E[Y_i]$ is the same as before, and so

$$E[Y] = E\left[\sum_{i=1}^{m} Y_i + k\right] = E\left[\sum_{i=1}^{m} Y_i\right] + k = 7m/8 + k.$$

Since $m + k$ is the maximum number of clauses that can be satisfied, the approximation ratio is at most

$$\frac{m+k}{7m/8+k} \le \frac{m+k}{7m/8+7k/8} = \frac{8}{7}.$$

□

4.2 Weighted vertex cover and LP rounding

Optimization Problem (Min-Weight Vertex Cover) Given an undirected graph $G = (V, E)$ and a weight function on vertices $w : V \to \mathbb{Q}^+$, find a vertex cover $C \subseteq V$ of minimum weight $w(C) = \sum_{v \in C} w(v)$.

Consider the following linear program:

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{v \in V} w(v) \cdot x_v & \\ \text{subject to} & x_u + x_v \ge 1 & \forall (u,v) \in E \\ & x_v \le 1 & \forall v \in V \\ & x_v \ge 0 & \forall v \in V \end{array}$$

Algorithm APPROX-MIN-WEIGHT-VC(G, w)

```
$C = \emptyset$
Compute an optimal solution $\overline{x}$ to $G$'s associate linear program.
for $v \in G.V$ do
    if $\overline{x}_v \ge 1/2$ then
        $C = C \cup \{v\}$
    end if
end for
return $C$
```

Theorem 2.2. APPROX-MIN-WEIGHT-VC is a 2-approximation algorithm for the minimum weight vertex cover problem.

Proof. Want to show:

Why is C a cover?

Compare an optimal cover C^* to the objective function value for optimal solution of the LP z^*. Obtain $z^* \le w(C^*)$. (C^* is a feasible solution to the LP)

Obtain $z^* \geq w(C)/2$. Key step:

$$z^* \geq \sum_{v \in V : \overline{x}_v \geq 1/2} w(v) \cdot \overline{x}_v$$

Combine inequalities to get $w(C) \leq 2w(C^*)$.

□

5 A Fully Polynomial Time Approximation Scheme