Approximation Algorithms

Martin Böhm (University of Bremen)

May 9, 2019

Sum of completion times, randomization

$1|r_j|\sum C_j$

Input: n jobs with processing time p_j and release date r_j

Task: Find non-preemptive schedule of the jobs (no job j scheduled before r_j)

Objective: Minimize sum of completion times $\sum_{j\in[n]} C_j$

Relaxation

Preemptive schedules: we are allowed to stop (preempt) the processing of jobs and resume them later.

Optimal preemptive solution

Schedule according to shortest remaining processing time (SRPT) rule.

$1|r_j|\sum C_j$

C_j^P: preemptive optimal completion time of job j.

Lemma

$$\sum_{j=1}^{n} C_j^P \leq \mathrm{OPT}$$

Idea

Schedule non-preemptively in order of C_j^P.

C_j^N: non-preemptive optimal completion time of job j.

Theorem

$\sum_{j=1}^{n} C_j^N \leq 2\,\mathrm{OPT}$.

$1|r_j|\sum w_j C_j$

Weighted version: jobs have weights w_j.

Objective: $\sum_{j=1}^{n} w_j C_j$

NP-hard to solve preemptive problem. $\Rightarrow$ Use a different relaxation.

LP

$$\begin{aligned}
\min \quad & \sum_{j=1}^{n} w_j C_j & \\
\text{s.t.} \quad & C_j \geq r_j + p_j & \forall j \\
& \sum_{j \in S} p_j C_j \geq \frac{1}{2}\left(\sum_{j \in S} p_j\right)^2 & \forall S \subseteq [n] \\
& C_j \geq 0 & \forall j
\end{aligned}$$

Theorem

Let C^* be an optimal LP solution, then scheduling in order of completion times C_j^* is a 3-approximation for minimizing weighted sum of completion times with release dates on a single machine.

Randomization for approximation algorithms

Randomization models

Monte Carlo:

- Algorithm always finishes in polynomial time.
- We compare the expected value of the solution to the optimum – for minimimization, $E[ALG] \geq \alpha OPT$.

Las Vegas:

- Our algorithm always returns an α-approximate solution.
- The expected runtime of our algorithm is polynomial, but can be longer in some cases.

- For approximations, we usually care about *Monte Carlo* algorithms.
- Both models make sense for decision problems.
- Fairly big open question: **Is randomization inside** P**?**

Satisfiability

Input: A list of n variables and a Boolean formula C on those variables (e.g. $(x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3)$).

Decide: Can you set the variables so that every clause is satisfied?

- Clause: $(x_1 \vee \neg x_2 \vee x_4)$.
- Literal: $x_1, x_2, \neg x_2, \neg x_4$.
- Variable: x_1, x_3.

Sat and Max-Sat

SAT:

Input: A list of n variables and a Boolean formula C on those variables (e.g. $(x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3)$).

Decide: Can you set the variables so that every clause is satisfied?

MAX-SAT:

Input: A list of n variables and a Boolean formula C on those variables (e.g. $(x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3)$).

Output: Any assigment, possibly evaluating some clauses to `False`.

Goal: Maximize the number of the clauses that are satisfied.

Weighted Max-Sat

MAX-SAT:

Input: A list of n variables and a Boolean formula C on those variables (e.g. $(x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3)$).

Output: Any assigment, possibly evaluating some clauses to `False`.

Goal: Maximize the number of the clauses that are satisfied.

WEIGHTED MAX-SAT: each clause has a weight $w_c \geq 0$, maximize the weighted sum.

- K-SAT – all clauses are of size at most k.
  3-SAT – $(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3)$.
- EK-SAT – all clauses are of size exactly k.
- **Theorem:** 2-SAT is polynomial-time solvable.
- **Theorem:** MAX-2-SAT (decision version) is NP-complete.

Approximations for Weighted Max-Sat

Trivial deterministic algorithm:

1. Try setting all variables to 0.
2. Try setting all variables to 1.
3. Pick better of the two solutions.

Theorem

This is a $1/2$*-approximation algorithm for* Weighted Max-Sat.

The same algorithm for:

- Max-E2-Sat: $1/2$-approximation.
- Max-E3-Sat: $1/2$-approximation.

Flipping coins for Max-Sat

Simple randomized Monte Carlo algorithm:

1. Set all variables uniformly randomly to 1 with probability $1/2$.
2. Return the final solution.

Theorem

The above is a randomized $1/2$*-approximation algorithm for* Weighted Max-Sat.

- Max-E2-Sat: 3/4-approximation.
- Max-E3-Sat: 7/8-approximation.

Derandomization

Converting the previous algorithm into a deterministic one with the same guarantee (value $E[ALG]$).

Method of conditional expectations: For x_1:

- Set $x_1 = 0 \Rightarrow$ value is $E[\mathrm{Sol}|x_1 = 0]$.
- Set $x_1 = 1 \Rightarrow$ value is $E[\mathrm{Sol}|x_1 = 1]$.
- $E[\mathrm{ALG}] = \frac{1}{2}E[\mathrm{Sol}|x_1 = 0] + \frac{1}{2}E[\mathrm{Sol}|x_1 = 1] \Rightarrow$ one of them gives a better value, pick that one.

How to compute $E[\mathrm{Sol}|x_1 = 0, x_2 = 1, \ldots, x_k = 0]$?

$$E[\mathrm{Sol}|x_1 = 0, x_2 = 1, \ldots, x_k = 0] = \sum_{c=1}^{m} w_c E[\text{clause is satisfied}] =$$

$$\sum_{c=1}^{m} w_c Pr[\text{clause is satisfied after fixing } x_1, \ldots, x_k].$$