Approximation Basics (2)

Design Techniques

Xiaofeng Gao

Department of Computer Science and Engineering
Shanghai Jiao Tong University, P.R.China

Spring 2015

Procedure

Given:

- An instance x of the problem and a feasible solution y (found using some other algorithm)

Goal:

- Improve the current solution by moving to a better "neighbor" solution

Steps:

- Given a feasible solution y and its neighborhood structure
- Look for a neighbor solution with an improved value of the measure function
- Repeat the steps until no improvement is possible
- The algorithm stops in a "local optimum" solution.

Comments

Main issues for neighborhood structure involve

- The quality of the solution obtained (how close is the value of the local optimum to the global optimal value);
- The order in which the neighborhood is searched;
- The complexity of verifying that the neighborhood does not contain any better solution;
- The number of solutions generated before a local optimum is found.

The behavior of local search algorithm depends on the following parameters:

- The neighborhood function $\mathcal{N}$.
- The starting solution s_0.
- The strategy of selection of new solutions.

Parallel Job Scheduling Problem

Problem

Instance: *Given n jobs each with p_j executing time, and m machines, each of which can process at most one job at a time.*

Solution: *Assign each job to a machine sequentially.*

Measure: *Complete all jobs as soon as possible. Say, if job j completes at time C_j, then the target is to minimize $C_{\max} = \max_{1 \le j \le n} C_j$ (called makespan).*

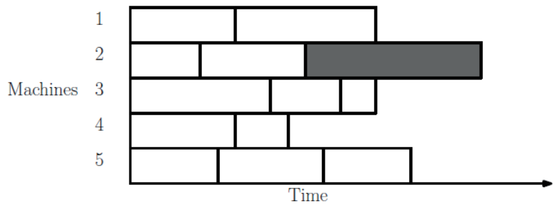
Local Search Algorithm

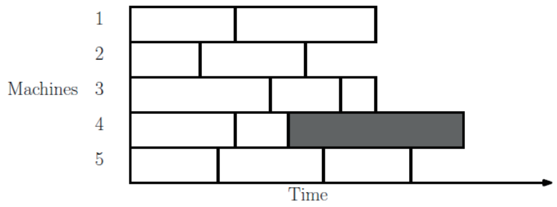
Algorithm 1 Local Scheduling

Input: n jobs each with p_j, m.
Output: A schedule on m machines.

```
Let S be an arbitrary schedule.
repeat
    Consider the job ℓ that finishes last.
    if ∃m_i whose finishing time is earlier than C_ℓ − p_ℓ then
        transfer job ℓ to this machine m_i.
    end if
until The last job to complete cannot be transferred
Return S
```


Illustration





Approximation Ratio

Theorem: Local Scheduling is a 2-Approximation.

Proof: Let $C^*_{\max}$ be the optimal schedule. Since each job must be processed, $C^*_{\max} \geq \max_{1 \leq j \leq n} p_j$.

Next $P = \sum_{j=1}^{n} p_j$ is the total time units to accomplish, and only m machines are available, a machine will be assigned $\frac{P}{m}$ average units of works. Consequently, there must exist one machine that is assigned at least that much work.

$$C^*_{\max} \geq \frac{\sum_{j=1}^{n} p_j}{m}$$

Proof (2)

Consider the solution of Local Scheduling. Let ℓ be a job that completes last in the final schedule, then $C_\ell = C_g$. Since algorithm terminates at this stage, every other machine must be busy from time 0 till the start of ℓ at $S_\ell = C_\ell - p_\ell$.

Partition the schedule into two disjoint time intervals by S_ℓ. Since every job must be processed, the latter interval has length at most $C^*_{\max}$.

Proof (3)

Now consider the former interval, the total amount of work being processed in this interval is mS_ℓ which is no more than the total work to be done. Thus

$$S_\ell \leq \sum_{j=1}^{n} p_j/m.$$

Clearly $S_\ell \leq C^*_{\max}$. We thereby get a 2-approximation. □

Time Complexity

Theorem: The time complexity of Local Scheduling is $O(n)$.

Proof: We prove it by showing that each job can be rescheduled only once. Let $C_{\min}$ be the completion time of a machine that completes earliest. Then $C_{\min}$ never decreases.

Assume a job j can be rescheduled twice, from machine i to i' then to i^*. When j is reassigned to i', it then starts at $C_{\min}$ for the current schedule. Similarly, When j is assigned to i^*, it then starts at $C'_{\min}$.

Proof (2)

No change occurred to the schedule on machine i' in between these two moves for job j.

Hence, $C'_{\min}$ must be strictly smaller than $C_{\min}$, which contradicts our claim that $C_{\min}$ is nondecreasing over the iterations of the Local Scheduling.

Thus, each job should only be considered once, and the time complexity of Local Scheduling is $O(n)$. □

Maximum Cut Problem

Problem

Instance: *Given $G = (V, E)$.*

Solution: *Partition of V into disjoint sets V_1 and V_2.*

Measure: *The cardinality of the cut, i.e., the number of edges with one endpoint in V_1 and one endpoint in V_2.*

Local Search Algorithm

Algorithm 2 Local Cut

Input: $G = (V, E)$

Output: Local optimal cut (V_1, V_2).

$s = s_0 = (\emptyset, V)$. ▷ Initial Feasible Solution

$\mathcal{N}(V_1, V_2)$ includes all (V_{1k}, V_{2k}) for $k = 1, \cdots, |V|$ s.t.

$$\begin{cases} \text{If } v_k \in V_1, \text{ then } V_{1k} = V_1 - \{v_k\}, V_{2k} = V_2 + \{v_k\} \\ \text{If } v_k \in V_2, \text{ then } V_{1k} = V_1 + \{v_k\}, V_{2k} = V_2 - \{v_k\} \end{cases}$$

repeat

  Select any $s' \in \mathcal{N}(s)$ not yet considered;

  if $m(s) < m(s')$ **then**

    $s = s'$;

  end if

until All solutions in $\mathcal{N}(s)$ have been visited

Return s

Approximation Ratio

Theorem: Given an instance G of Maximum Cut, let (V_1, V_2) be a local optimum w.r.t. neighborhood structure $\mathcal{N}$ and let $m_{\mathcal{N}}(G)$ be its measure. Then

$$\frac{m^*(G)}{m_{\mathcal{N}}(G)} \leq 2.$$

Proof:

- Let m be the number of edges of the graph G.
- Then we have $m^*(G) \leq m$.
- It is sufficient to prove that $m_{\mathcal{N}}(G) \geq \frac{m}{2}$.

Proof (2)

We denote by m_1 and m_2 the number of edges connecting vertices inside V_1 and V_2 respectively. Then,

$$m = m_1 + m_2 + m_{\mathcal{N}}(G).$$

Given any vertex v_i, we define

$$m_{1i} = \{v | v \in V_1 \ \& \ (v, v_i) \in E\}, m_{2i} = \{v | v \in V_2 \ \& \ (v, v_i) \in E\}.$$

If (V_1, V_2) is a local optimum, $\forall v_k$, $m(V_{1k}, V_{2k}) \leq m_{\mathcal{N}}(G)$. Thus

$$\forall v_i \in V_1, |m_{1i}| - |m_{2i}| \leq 0;$$

$$\forall v_j \in V_2, |m_{2j}| - |m_{1j}| \leq 0;$$

Proof (3)

By summing over all vertices in V_1 and V_2, we obtain

$$\sum_{v_i \in V_1} (|m_{1i}| - |m_{2i}|) = 2m_1 - m_{\mathcal{N}}(G) \leq 0$$

$$\sum_{v_j \in V_2} (|m_{2j}| - |m_{1j}|) = 2m_2 - m_{\mathcal{N}}(G) \leq 0$$

Sum two inequalities together, we have

$$m_1 + m_2 - m_{\mathcal{N}}(G) \leq 0$$

Recall that $m_1 + m_2 = m - m_{\mathcal{N}}(G)$, we have $m - 2m_{\mathcal{N}}(G) \leq 0$, thus $m_{\mathcal{N}}(G) \geq \frac{m}{2}$, and

$$\frac{m^*(G)}{m_{\mathcal{N}}(G)} \leq \frac{m}{m_{\mathcal{N}}(G)} \leq 2.$$

□

Overview

An overview of LP relaxation and rounding method is as follows:

- Formulate an optimization problem as an integer program (IP).
- Relax the integral constraints to turn the IP to an LP.
- Solve LP to obtain an optimal solution x^*;
- Construct a feasible solution x^I to IP by rounding x^* to integers.

Rounding can be done deterministically or probabilistically (called **randomized rounding**).

Set Cover Problem

Problem

Instance: *Given a universe $U = \{e_1, \cdots, e_n\}$ of n elements, a collection of subsets $\mathbf{S} = \{S_1, \ldots, S_m\}$ of U, and a cost function $c : \mathbf{S} \to \mathbb{Q}^+$.*

Solution: *A subcollection $\mathbf{S}' \subseteq \mathbf{S}$ that covers all elements of U.*

Measure: *Total cost of the chosen subcollection, $\sum_{S_i \in \mathbf{S}'} c(S)$.*

Integer Program for Set Cover

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in \mathbf{S}} c(S) x_S \\ \text{subject to} \quad & \sum_{S: e \in S} x_S \geq 1, e \in U \\ & x_S \in \{0, 1\} \end{aligned}$$

x_S is a variable for each set $S \in \mathbf{S}$, which is allowed 0/1 values, and it is set to 1 iff set S is picked in the set cover.

LP-Relaxation for Set Cover

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in \mathbf{S}} c(S) x_S \\ \text{subject to} \quad & \sum_{S: e \in S} x_S \geq 1, e \in U \\ & x_S \geq 0 \\ & x_S \leq 1 \leftarrow \text{this constraint is redundant} \end{aligned}$$

Deterministic Rounding

Algorithm 3 Set Cover via LP-Rounding (Deterministic)

Input: U with n item; $\mathbf{S}$ with m subsets; cost function $c(S_i)$.
Output: Subset $\mathbf{S}' \subseteq \mathbf{S}$ such that $\bigcup_{e_i \in S_k \in \mathbf{S}'} e_i = U$.

```
Find an optimal solution X_S to the LP-relaxation.
Define f as the frequency of the most frequent element.
for all x_S ∈ X_S do
    if x_S ≥ 1/f then
        round x_S = 1;
    else
        round x_S = 0;
    end if
end for
Return S' = {S | x_S = 1}.
```


Performance Analysis

Theorem: LP-Rounding achieves an approximation factor of f for the set cover problem.

Proof:

- Feasible Solution: For $e \in U$, $\sum_{S: e \in S} x_S \geq 1$. e is at most in f sets, then there must exist a set S such that $e \in S$ and $x_S \geq 1/f$. Thus e is covered by this algorithm.
- Approximation Ratio: For $S \in \mathbf{S}'$, x_S is increased by a factor of at most f. Thus,

$$cost(\mathbf{S}') \leq f \cdot OPT_f \leq f \cdot OPT,$$

where OPT_f is the optimal solution of LP, and OPT is the optimal solution for the original problem. □

Randomized Rounding (Step 1)

Algorithm 4 Set Cover via LP-Rounding (Randomized, Step 1)

Input: U with n item; $\mathbf{S}$ with m subsets; cost function $c(S_i)$.
Output: Subset $\mathbf{S}' \subseteq \mathbf{S}$ such that $\bigcup_{e_i \in S_k \in \mathbf{S}'} e_i = U$.

```
Find an optimal solution X_S to the LP-relaxation.
for all S ∈ S do
    Pick S into S' with probability x_S;
end for
Return S'.
```


Expected Cost of Step 1

If $\mathbf{S}'$ is the collection of the sets picked, then the cost expectation of our solution in Step 1 is:

$$\begin{aligned} E[cost(\mathbf{S}')] &= \sum_{S \in \mathbf{S}} Pr[S \text{ is picked}] \cdot c_S \\ &= \sum_{S \in \mathbf{S}} x_S \cdot c_S \\ &= OPT_f \end{aligned}$$

which means the expected cost of Step 1 is equal to the optimal solution of LP.

Uncovered Rate of Step 1

For any element $e_i \in U$, suppose e_i occurs in k sets of $\mathbf{S}$, say $S_1, S_2, \ldots, S_k$.

Since e_i is fractionally covered, then $x_{S_1} + \cdots + x_{S_k} \geq 1$.

$$\begin{aligned}\mathbf{Pr}[e_i \text{ is not covered by } \mathbf{S}'] &= \prod_{i=1}^{k}(1 - x_{S_i}) \\ &\leq (1 - \frac{1}{k})^k \quad \text{(AM-GM Inequality)} \\ &\leq \frac{1}{e} \quad (e = \sum_{n=0}^{\infty}\frac{1}{n!}, \text{ Euler's number})\end{aligned}$$

AM-GM Inequality: $\sqrt[n]{x_1 x_2 \cdots x_n} \leq \frac{1}{n}(x_1 + x_2 + \cdots + x_n)$.

Randomized Rounding (Step 2)

We need to guarantee a complete set cover. Thus the following algorithm is used to increase the success rate.

Algorithm 5 Set Cover via LP-Rounding (Randomized, Step 2)

1: Pick a constant c such that $\left(\frac{1}{e}\right)^{c\log n} \leq \frac{1}{4n}$.
2: Independently repeat Step 1 for $c\log n$ times to get $c\log n$ subcollections, and compute their union, say $\mathbf{C}'$.
3: Output $\mathbf{C}'$.

Note: c can be set as different constant, resulting different success rate.

Success Rate of Step 2

$Pr[e_i \text{ is not covered by } \mathbf{C}'] \leq \left(\frac{1}{e}\right)^{c\log n} \leq \frac{1}{4n}$;

$\Rightarrow Pr[\mathbf{C}' \text{ is not a valid set cover}] \leq 1 - \left(1 - \frac{1}{4n}\right)^n \leq n \cdot \frac{1}{4n} \leq \frac{1}{4}$;

Clearly, $E[cost(\mathbf{C}')] \leq OPT_f \cdot c\log n$.

$\Rightarrow Pr[cost(\mathbf{C}') \geq OPT_f \cdot 4c\log n] \leq \frac{1}{4}$ (Markov's Inequality)

$\Rightarrow Pr[\mathbf{C}' \text{ is a valid set cover } \& \ cost(\mathbf{C}') \leq OPT_f \cdot 4c\log n] \geq \frac{1}{2}$.

Markov's Inequality: $Pr[X \geq a] \leq \frac{E(X)}{a}$.

Algorithm for LP Randomized Rounding

Algorithm 6 Set Cover via LP-Rounding (Randomized)

Input: U with n item; $\mathbf{S}$ with m subsets; cost function $c(S_i)$.
Output: Subset $\mathbf{S}' \subseteq \mathbf{S}$ such that $\bigcup_{e_i \in S_k \in \mathbf{S}'} e_i = U$.

1: Find an optimal solution $\mathbf{X_S}$ to the LP-relaxation.
2: Pick a constant c such that $\left(\frac{1}{e}\right)^{c\log n} \leq \frac{1}{4n}$.
3: **for** $i = 1$ to $c\log n$ **do**
4: **for all** $S \in \mathbf{S}$ **do**
5: Pick S into $\mathbf{S_i'}$ with probability x_S;
6: **end for**
7: **end for**
8: Return $\mathbf{C}' = \bigcup_{i=1}^{c\log n} \mathbf{S_i'}$.

Performance Analysis

We can verify in polynomial time whether $\mathbf{C}'$ satisfies both these conditions.

If not, we repeat the entire algorithm. The expected number of repetitions needed is at most 2.

Thus, the randomized rounding algorithm achieves an expected approximation ratio of $O(\log n)$. (Log-APX)