Approximation Algorithms
Lecture Notes
Lan Guo

Minimum Partition

Minimum Partition
Instance: Finite set X of items, for each $x_i \in X$ a weight $a_i \in Z^+$.
Solution: A partition of the items into two sets Y_1 and Y_2.
Measure: max $\{\sum_{x_i \in Y_1} a_i, \sum_{x_i \in Y_2} a_i\}$.

The measurement of the minimum partition represents the unfairness of the actual partition. We want to minimize the unfairness. Generally, we are interested to find the min (max $\{\sum_{x_i \in Y_1} a_i, \sum_{x_i \in Y_2} a_i\}$). It is known to be a weakly NP-complete problem. We can find the optimal solution in $O(2^n)$ time, since each item has two possibilities: either belonging to Y_1 or Y_2; for n items, there are total of 2^n possible combinations. By using dynamic programming algorithm, we can solve the problem in $O(n\sum a_i)$-time, which is left as an exercise.

We define S(Y) to be the total weight of all items in the set Y.

Strategy 1

1. Initialize $S(Y_1) = S(Y_2) = 0$
2. For (i = 1 to n)
    If $S(Y_1) \geq S(Y_2)$
        Put a_i into Y_2
        $S(Y_2)$ += a_i
    Else
        Put a_i into Y_1
        $S(Y_1)$ += a_i

Analysis: This algorithm is not always optimal. For example, {ε, ε, M} (ε is a small number and M is a very big number) will be partitioned as Y_1 = {ε, M} and Y_2 = {ε}, while the optimal is $Y_1 = \{\varepsilon, \varepsilon\}$ and $Y_2 = \{M\}$. However, we can prove that it is 2-approximation.

First we know that OPT $\geq \sum a_i /2$. If you put everything in one partition, it is still 2-approximation, i.e. algo $< \sum a_i \leq$ 2OPT. Therefore, Strategy 1 is a 2-approximation algorithm.

Strategy 2

0. Sort in non-increasing order.
1. Initialize $S(Y_1) = S(Y_2) = 0$
2. For (i = 1 to n)
    If $S(Y_1) \geq S(Y_2)$
        Put a_i into Y_2
        $S(Y_2)$ += a_i
    Else
        Put a_i into Y_1
        $S(Y_1)$ += a_i

Analysis: We can find an example that this algorithm is not optimal, such as {10, 10, 9, 9, 2}. According to this algorithm, the partition is $Y_1 = \{10, 9, 2\}$ and $Y_2 = \{10, 9\}$, while the optimal partition is $Y_1 = \{10, 10\}$ and $Y_2 = \{9, 9, 2\}$.

Claim: $m_H \leq 7/6$ OPT

This problem can be viewed as Job Scheduling with 2 processors. Observe that in *Longest Processing Time* algorithm, we got $C_{LPT} \leq (4/3 - 1/3m)$ OPT. Substitute m=2 into the formula, we have $C_{LPT} \leq (4/3 - 1/6)$ OPT = 7/6 OPT.

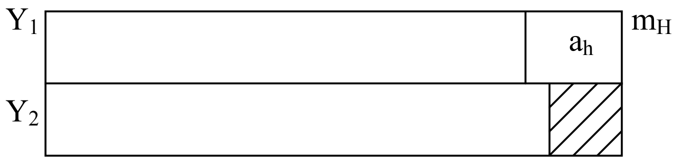

Figure 1. Minimum 2-Partition

Partition PTAS (Polynomial Time Approximation Scheme)

1. Input (X, γ)
2. If $\gamma \geq 2$ return (X, Φ)
3. Else

    Sort items in non-increasing order with respect to their wight;

    (*Let ($x_1,\ldots,x_n$) be the obtained sequence*)

    $k(r) = \left\lceil \frac{2-\gamma}{\gamma-1} \right\rceil$;

    (*First phase*)

    Find an optimal partition Y_1, Y_2 of $x_1,\ldots, x_{k(r)}$;

    (*Second phase*)

    for j:= k(γ) + 1 to n do

    if $\sum_{x_i \in Y_1} a_i \leq \sum_{x_i \in Y_2} a_i$ then

    $Y_1 := Y_1 \cup \{x_j\}$;

    Else

    $Y_2 := Y_2 \cup \{x_j\}$;

4. Return Y_1, Y_2

Theorem: *Partition PTAS is a polynomial-time approximation scheme for Minimum Partition.*

Proof: Let us first prove that, given an instance x of Minimum Partition and a rational γ>1, the algorithm provides an approximation solution (Y_1, Y_2) whose performance ratio is bounded by γ. If $\gamma \geq 2$, then the solution (X, Φ) is clearly an γ-approximate solution since any feasible solution has measure at least equal to half of the total weight w(X) = $\sum a_i$. Let us then assume that $\gamma < 2$ and let $w(Y_i) = \sum_{X_i \in Y_i} a_j$, for i=1,2, and L = w(X)/2. Without loss of generality, we may assume that $w(Y_1) \geq w(Y_2)$ and that a_h is the last item that has been added to Y_1 (see Figure 1.). This implies that $w(Y_1) - a_h \leq w(Y_2)$. By adding $w(Y_1)$ to both sides and dividing by 2 we obtain that

$$w(Y_1) - L \leq a_h/2.$$

If a_h has been inserted in Y_1 during the first phase of the algorithm, then it is easy to see that the obtained solution is indeed an optimal solution. Otherwise (that is, a_h has been inserted during the second phase), we have that $a_h \leq a_j$, for any j with $1 \leq j \leq k(\gamma)$, and that $2L \geq a_h(k(\gamma) + 1)$. Since $w(Y_1) \geq L \geq w(Y_2)$ and $OPT \geq L$, the performance ratio of the computed solution is

$$\frac{w(Y_1)}{OPT} \leq \frac{w(Y_1)}{L} \leq 1 + \frac{a_h}{2L} \leq 1 + \frac{1}{k(r)+1} \leq 1 + \frac{1}{\frac{2-r}{r-1}+1} = \gamma.$$

Finally, we prove that the algorithm works in time $O(n\log n + n^{k(r)})$. In fact, we need time $O(n\log n)$ to sort the n items. Subsequently, the first phase of the algorithm requires time exponential in $k(\gamma)$ in order to perform an exhaustive search for the optimal solution over the $k(\gamma)$ heaviest items $x_1, \ldots, x_{k(\gamma)}$ and all other steps have a smaller cost. Since $k(\gamma)$ is $O(1/(\gamma - 1))$, the theorem follows. [1]

Reference:

[1] G. Ausiello etc. Complexity and Approximation—Combinatorial Optimization Problems and Their Approximability Properties, Springer-Verlag, New York, 1999