CS F364
Design & Analysis of Algorithms

Algorithm Design - Approximation

Approximation Schemes

- Polynomial Time Approximation Scheme (PTAS)

- Design and Example.

5/2/2015 Sundar B. CSIS, BITS, Pilani

Approximation Ratio vs. Time Complexity

- Given an $\mathbb{NP}$-complete problem, is there a way to trade-off approximation ratio for time complexity?
  - In practice, even 2 may not be a "good" approximation ratio.
  - In particular, can we find algorithms with an approximation ratio of $1+\epsilon$ where ϵ can be made arbitrarily small?
    - Note that ϵ must be positive:
      - $\epsilon = 0$ would imply an exact solution for a $\mathbb{NP}$ hard problem.

Approximation Schemes

- Polynomial Time Approximation Scheme (PTAS):
  - An $\mathbb{NP}$-complete problem π is said to admit a PTAS if there exists an algorithm A such that
  - for any input instance x of π, and for any r > 1, A(x,r) returns an r-approximate solution i.e.
    - m(x,A(x,r)) <= r * m*(x) (assuming minimization)
- Typically:
  - As r decreases (and approaches 1) the time complexity of the algorithm increases.
    - Of course, when r=1 we expect the algorithm to be have exponential time complexity (unless $\mathbb{P}=\mathbb{NP}$)

PROBLEM - PARTITION

- Problem Definition:
  - Given a multi-set X of rational numbers, partition the numbers into two sets Y1 and Y2, so that sums of their values are as close as possible.
- Formal Problem Definition:
  - I = { X | X is a multi-set of numbers in Z^+ }
  - F(X) = (Y1, Y2) a partition of X
  - m(X, (Y1,Y2)) = max(w(Y1), w(Y2)),
    - where for any set Y, w(Y) = $\Sigma_{a \in Y}$ a
  - goal: min

Partition – Relative Approximation

- Lower Bound (on optimal measure)
  - Lemma : m*(X) >= w(X) / 2
  - Proof:
    - For (Y1,Y2) in F(X) m(X,(Y1,Y2)) >= w(X)/2
      - (by definition of m and partition)
- Corollary:
  - Consider the trivial solution (X,{}) for any input instance X:
    - m(X, (X, {})) = w(X) <= 2 * m*(x)
  - i.e. we only need to consider r-approximation algorithms for values of r < 2

PARTITION – R-APPROXIMATION ALGORITHM

- Greedy_PART(X)
  1. Sort items in X in decreasing order to get x1, x2, … xn
  2. Y1 = Y2 = { }
  3. for j = 1 to n do

     if $((\sum_{a \in Y1} a) >= (\sum_{a \in Y2} a))$

     then Y2 = Y2 U { xj }

     else Y1 = Y1 U { xj }
  4. return (Y1, Y2)

- Claim:
  - Greedy_PART(X) does not yield an optimal solution
- (Counter)Example:
  - Consider the input instance

  { 50, 49, 41, 39, 19 }
  - Optimal solution:

  ({ 50, 49}, { 41, 39, 19})
  - Greedy_PART(X) will return

  ({ 50, 39, 19}, {49, 41})

Partition – r-Approximation Algorithm - Analysis

- Lemma 1:
  - If Greedy_PART(X) returns (Y1, Y2) such that |Y1| = 1 and w(Y1) >= w(Y2) then (Y1, Y2) is the optimal solution for X.
  - Proof: Trivial
- Lemma 2:
  - Let Greedy_PART(X) returns (Y1, Y2) such that |Y1| >= 2 and w(Y1) >= w(Y2). If the last item added to Y1 is xh and the rank of xh is k then xh <= w(X)/k
  - Proof:
  - If rank of xh was k then at least k items each >= xh have been added
  - i.e. X includes at least k items each >= xh.
  - i.e. w(X) >= k*xh
  - i.e. xh <= w(X)/k

PARTITION – r-APPROXIMATION ALGORITHM – ANALYSIS [2]

- Lemma 3: If Greedy_PART(X) returns (Y1,Y2)

  then m(X, (Y1,Y2)) <= (3/2) * m*(X)

- Proof: Assume without loss of generality:
  - w(Y1) >= w(Y2) and xh is the last item added to Y1
- Then: w(Y1) – xh <= w(Y2)
  - i.e. 2*w(Y1) – xh <= w(X)
  - i.e. 2*w(Y1) <= xh + w(X) [E1]

1. If |Y1| = 1, then m(X, (Y1,Y2))/m*(X) = 1 by Lemma 1.
2. If |Y1|>=2 then
   - m(X, (Y1,Y2))/m*(X) <= w(Y1) / m*(X)
   - <= 2 * w(Y1) / w(X) [by lower bound]
   - <= 1 + xh / w(X) [by E1]
   - <= 1 + 1/2 [by Lemma 2]

Tighter result: 1+1/k, where k is rank of last item added to larger set.

PARTITION – r-APPROXIMATION ALGORITHM – ANALYSIS [3]

- Lemma 4:
  - Greedy_PART is a polynomial time 3/2 – approximate algorithm for PARTITION.
- Proof:
  - m(X, Greedy_PART(X)) / m*(X) <= 3/2 for any X by Lemma 3
  - Time complexity of Greedy_PART is O(N*A(N) + N*logN)
    - where A(N) is the time taken to add an element to a set of size N

PTAS – Example – Partition

- Assume EXACT_PART is an exact algorithm for PARTITION
  - For instance, it may obtain all subsets Y1 of the given input set X and find the one with minimum measure among all (Y1, X – Y1)
- PTAS_PART(X, r)
  1. If r>=2 then return (X, { })
  2. Sort items in X in decreasing order to get x1, x2, … xn
  3. k = fpart(r, n) // fpart returns an integer in the range 1..n
  4. (Y1, Y2) = EXACT_PART({x1,…,xk});
  5. for j = k+1 to n do

     if $((\sum_{a \in Y1} a) >= (\sum_{a \in Y2} a))$ then Y2 = Y2 U { xj }

     else Y1 = Y1 U { xj }
  6. return (Y1, Y2)

PTAS – Example – Partition [2]

- Lemma 5: If PTAS_PART(X) returns (Y1,Y2)

  then m(X, (Y1,Y2)) <= (1+(1/(1+fpart(r,n)))) * m*(X)
- Proof:
  - Assume without loss of generality:
    - w(Y1) >= w(Y2) and xh is the last item added to Y1
  - Then:

1. If xh was added to |Y1| by EXACT_PART then the solution is optimal.
2. Otherwise: xh * (fpart(r,n)+1) <= w(X) [E3]
   1. Modify Lemma 3 as follows:

      In the last step (application of Lemma 2), use E3.
   2. Then m(X, (Y1,Y2)) / m*(X) <= 1 + (1/(1+fpart(r,n)))

PTAS – Example – Partition [10]

- Theorem: PTAS_PART is a Polynomial Time Approximation Scheme for PARTITION
- Proof:
  - By lemma 5,
    - m(X, PTAS_PART(X)) <= (1+(1/(1+fpart(r)))) * m*(X)
  - Then let 1 + (1/(1+fpart(r))) = r
    - i.e. define fpart(r) = (1/(r-1)) – 1
  - i.e. PTAS_PART(X,r) with fpart(r) = (1/(r-1))-1 will be an r-approximate algorithm for PARTITION.
  - Time Complexity of PTAS_PART:
    - O(nlogn) for sorting + $O(2^{f(r)})$ for EXACT_PART + O(n*A(n)) for greedy part
    - i.e. $O(n\log n + 2^{f(r)})$ i.e. $O(n\log n + 2^{O(1/(r-1))})$