CS F364

Design & Analysis of Algorithms

Complexity – Optimization Problems

Approximation Algorithms

- Relative Approximation
- Design Techniques
  - Greedy Method
  - Sequencing
  - Randomization

4/30/2015 Sundar B. CSIS, BITS, Pilani

Relative Approximation

- Given an optimization problem π, for any input instance x and for any feasible solution y, the performance ratio of y is defined as:
  - R(x,y) = max(m(x,y)/m*(x), m*(x)/m(x,y))
- Given an optimization problem π, an algorithm A is said to be an r-approximate algorithm if there exists a constant r such that
  - i.e. for any x in I_π R(x,A(x)) <= r
- Example:
  - Greedy_Vertex_Cover is a 2-approximation algorithm

PROBLEM – BIN PACKING

- Problem Definition: BIN PACKING
  - Given a set of N items each with values S1, S2, … Sn, distribute them into equal-sized bins such that the number of bins required is minimum.
- Application:
  - Memory allocation problem (e.g.):
    - Memory is available as fixed-size blocks (i.e. bins)
    - Memory requests come in different sizes
- Note:
  - As the bins are equal-sized, we can assume that they are of unit size and scale the values S1, S2, … Sn accordingly.

PROBLEM – BIN PACKING [2]

- Formal Problem Definition: BIN PACKING
  - I = { S | S is a finite multi-set of n rational numbers in (0,1] }
  - F(S) = a partition $\{ B_1, B_2, \ldots B_k \}$ of S

    s.t. $\Sigma_{a \text{ in } Bj}\ a <= 1$ for each j
  - $m(S, \{B_1, B_2, \ldots B_k\}) = k$
  - goal = min

- Definition: Partition
  - $\{ B_1, B_2, \ldots B_k \}$ is said to be a partition of S if
  - $\cup B_j = S$ and $\cap B_j = \{\}$

Relative Approximation – Example – Bin Packing

- Lower Bound (on the optimal solution):
  - Given input instance S, let A = $\Sigma_{a \in S}$ a
  - Claim:
    - m*(x) >= ceil(A)
  - Proof:
    - Trivial (Perfect packing)

- Algorithm Next_Fit (S)
  1. i=0;
  2. for each item a in S:

     if ((a + $\Sigma_{b \in B_i}$ b) <= 1) { assign a to B_i }

     else { assign a to B_{i+1} ; i = i+1; }

Relative Approximation – Example – Bin Packing [2]

- Comments:
  - The technique used by NEXT_FIT is referred to as *Sequencing.*
  - NEXT_FIT is an online algorithm.
- Theorem:
  - NEXT_FIT is a polynomial time 2-approximate algorithm for BIN PACKING.
  - Proof:
    - For each pair of consecutive bins
      - the sum of values assigned to these two bins is > 1
    - i.e. # bins used / 2 <= A where $A = \sum_{a \in S} a$
    - $m_{NEXT\_FIT}(S) <= 2 * ceil(A) <= 2 * m^*(S)$

Relative Approximation – Example – Bin Packing [3]

- Claim:
  - The approximation ratio of 2 is asymptotically tight for NEXT_FIT
  - Proof:
    - Given any integer n, consider the instance with 4n items

      S = { 1/2, 1/2n, 1/2, 1/2n, … , 1/2, 1/2n }
    - Optimal solution would require: n+1 bins
    - NEXT_FIT would require: 2*n bins
- Question:
  - What is the weakness of NEXT_FIT ?

RELATIVE APPROXIMATION – BIN PACKING - FIRST FIT

- Algorithm FIRST_FIT (S):
  1. j=0; // j+1 is the number of bins; initially there is 1 bin
  2. for each item a in S {
     1. i = 0;
     2. while (i<=j) {
        1. if ((a + $\sum_{b \in B_i}$ b) <= 1) { assign a to B_i ; break; }
        2. else { i = i+1; }

        }
     3. if (i>j) { j=j+1; assign a to B_j ; }

     }

Exercise:
Compare the running time of NEXT_FIT w. that of FIRST_FIT.

Relative Approximation – BIN Packing – First Fit [2]

- Claim (w/o Proof):
  - $m_{FIRST\_FIT}(S) <= 1.7 * m^*(S) + 2$
    - Question: When is this better than NEXT_FIT?
- Comments:
  - FIRST_FIT is also an online algorithm

- Algorithm FIRST_FIT_DEC(S) // An offline algorithm
  1. Sort S in non-increasing order
  2. FIRST_FIT(S)
- How does FIRST_FIT_DEC perform?

Relative Approximation – BIN Packing –First Fit [3]

- Claim (w.o. proof):
  - $m_{FIRST\_FIT\_DEC}(S) <= (11 * m^*(S))/9 + 4$
- Claim:
  - The bound given above is tight.
- Proof:
  - For any +ve integer n, define an instance x_n of 5*n items:
    a. n items of size $1/2 + \epsilon$
    b. n items of size $1/4 + 2*\epsilon$
    c. n items of size $1/4 + \epsilon$
    d. 2*n items of size $1/4 - 2*\epsilon$

Optimal Packing: **($\lceil 3* n/2 \rceil$ bins)**
- n bins each w. one of a, c, and d
- $\lceil n/2 \rceil$ bins each w. two of b and d

FIRST_FIT_DEC Packing:
- n bins each w. one of a and b
- $\lceil n/3 \rceil$ bins each w. three of c
- $\lceil n/2 \rceil$ bins each w. four of d

$m_{FIRST\_FIT\_DEC}(x_n) = 11*n/6$

$m^*(x) = 3*n/2$

Relative Approximation – BIN Packing – First Fit [4]

- Question:
  - What is the weakness of FIRST_FIT (and that of FIRST_FIT_DEC)?

- BEST_FIT attempts to reduce fragmentation
- Algorithm BEST_FIT_DEC(S) // An offline algorithm
  1. Sort S in non-increasing order
  2. BEST_FIT(S)

Relative Approximation – BIN Packing – Best Fit

- Algorithm BEST_FIT (S):
  1. j=0;
  2. for each item a in S {
     1. i = 0; f = -1; minGap = MAX_INT;
     2. while (i<=j) {
        1. w = (a + $\sum_{b \in B_i}$ b)
        2. if (w <= 1) { if (minGap > 1-w) { minGap = 1-w; f=i; }}
        3. i = i+1;

        }

     if (f=-1) { j=j+1; assign a to B_j ; } else { assign a to B_f ; }

  }

Compare the running time of BEST_FIT_DEC with that of FIRST_FIT_DEC.

Relative Approximation – BIN Packing – Best Fit [2]

- Exercise:
  - Compare the running time of BEST_FIT_DEC with that of FIRST_FIT_DEC.
- Claim :
  - For any instance S of BIN_PACKING,

    $m_{FIRST\_FIT\_DEC}(S) >= m_{BEST\_FIT\_DEC}(S)$
  - Exercise: Prove this claim!

Relative Approximation – BIN Packing – Best Fit [3]

- Claim :
  - There exists instances S of BIN_PACKING such that $m_{FIRST\_FIT\_DEC}(S) > m_{BEST\_FIT\_DEC}(S) = m^*(S)$:
    - Construct such instances!
- Claim:
  - The approximation ratio of BEST_FIT_DEC is the same as that of FIRST_FIT_DEC.
    - The tight example works for this as well!
    - Combine this with the claim on the measures of the two solutions (*see previous slide*)

Approximation By Randomization : Example - Maximum Satisfiability

- Definition:
  - Instances: I = { C | C is a set of disjunctive clauses on a set of variables V }
  - SOL(C) = f, s.t. f : V --> { 0,1 }
  - m(C,f) = # clauses satisfied by f
  - Goal = max
- Algorithm Random_Satisfiability (RS)
  - for each v in V,
    - independently set f(v) = 1 with probability ½
  - return f;

Approximation By Randomization : Example - Maximum Satisfiability

- Algorithm Random_Satisfiability (RS)
  - for each v in V,
    - independently set f(v) = 1 with probability ½
  - return f;
- Measure of the Otpimal Solution:
  - The expected measure of a solution is given by

  $E[m_{RS}(C)] = (1-(1/2^k))*|C|$

  where each clause in C has at least k literals.
- Exercises:
  - Prove the above claim.
  - Find an upper bound for $m^*(x) / E[m_{RS}(C)]$
    - A trivial upper bound for the optimal measure is |C|