Parallel Computation – classes **NC & RNC**,

Algorithms and Complexity II

6/3/2014

Galenianou Myrto

Until now

- Sequential computations
- Bounded amount of computational activity at each instant

  -TM

  -multistring TM

  - RAM

- Exception : NDTM

Parallel models of computation -Uniform family of circuits

- A Boolean circuit C is defined as a finite directed acyclic graph.
- Each vertex corresponds to a gate
- The size of C is the total number of gates
- The depth of C is the total number of nodes in the longest path in C

Parallel models of computation -Uniform family of circuits

- A uniform family of boolean circuits is a family of circuits $C = (C_1, C_2, \ldots)$ where all circuits in the family are connected to the same algorithmic idea

- For functions *f(n),g(n)* : $N \rightarrow N$
- The parallel time of C is *O(f(n))*
- *The total work of C is O(g(n))*

Parallel models of computation -Uniform family of circuits

- def. Class PT/WK(*f(n), g(n)*) : the class of all languages $L \subseteq \{0,1\}^*$ such that there is a uniform family of circuits C deciding L with

  O(f(n)) parallel time and O(g(n)) total work

Parallel models of computation -Parallel Random Access Machines

- A RAM program is a finite sequence $\Pi = (\pi_1, \pi_2, ..., \pi_m)$ of instructions that edit memory cells
- A PRAM is a sequence of RAMs $P = (\Pi_1, \Pi_2, ..., \Pi_q)$
- Each RAM program works in parallel with the others
- q is a function; q=q(m,n), where m is the number of integers in the input and n the total length of the integers

Parallel models of computation -Parallel Random Access Machines

- Uniform family $P = (P_{m,n} : m, n \geq 0)$

- function F mapping finite sequences of integers to finite sequences of integers

- Then P computes F in parallel time f(n) , using g(n) processors.

- TH : If $L \subseteq \{0,1\}^*$ is in PT/WK(*f(n), g(n)*), then there is a uniform PRAM that computes the corresponding function F mapping $\{0,1\}^*$ to {0,1} in parallel time O(f(n)), using $O(\frac{g(n)}{f(n)})$ processors.

- TH :Suppose that a function F can be computed by a uniform PRAM in parallel time f(n) with g(n) processors, where f(n) and g(n) can be computed from 1^n in logarithmic space. Then there is a uniform family of circuits of depth O(f(n)(f(n)+logn) and size $O(g(n)f(n)(n^k f(n)+g(n))$ which computes the binary representation of F.

The Class NC

$$NC = PT\ /WK\,(\log^k n, n^k)$$

- Problems solvable in polylogarithmic parallel time with polynomial amount of total work.
- Subclasses : $NC_j = PT\ /WK(\log^j n, n^k)$
- NC closed under reductions

- Open problem : NC=P?

P-completeness

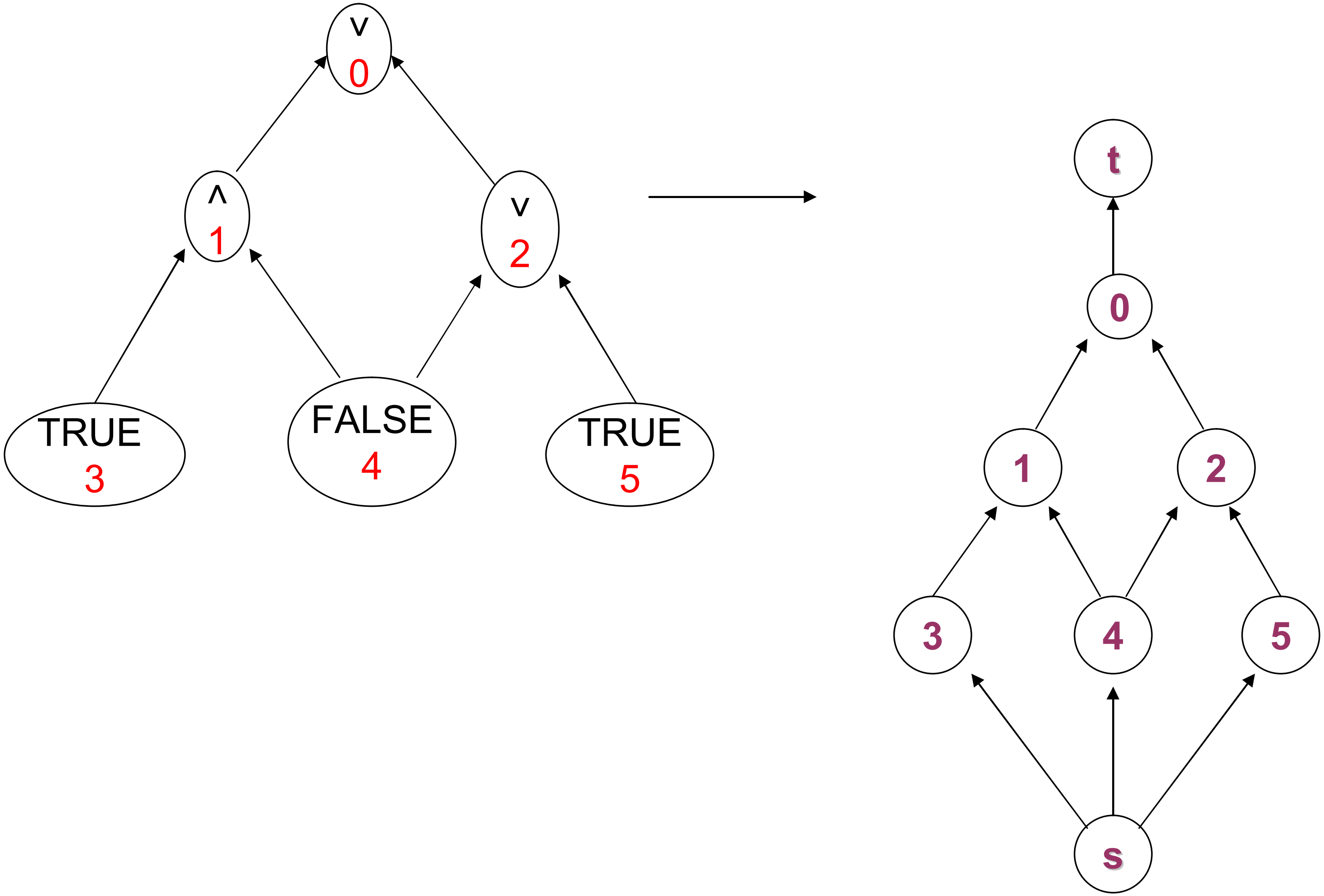
- Odd Max Flow is not in NC
- Odd Max Flow is P-complete

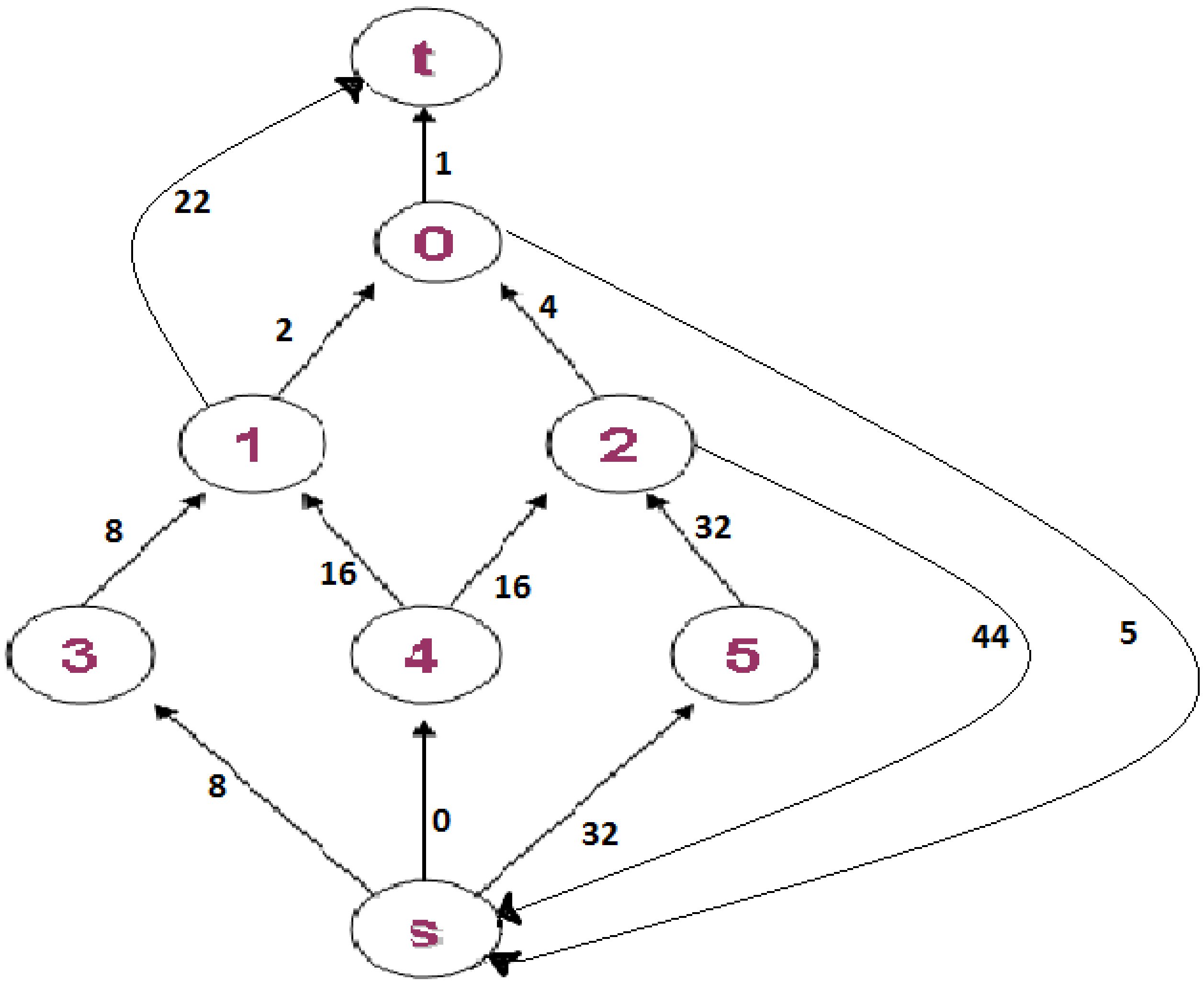
- Odd Max Flow : Given a network N=(V,E,s,t,c) is the maximum flow value odd?
- MCV ≤ Odd Max Flow

Monote Circuit Value

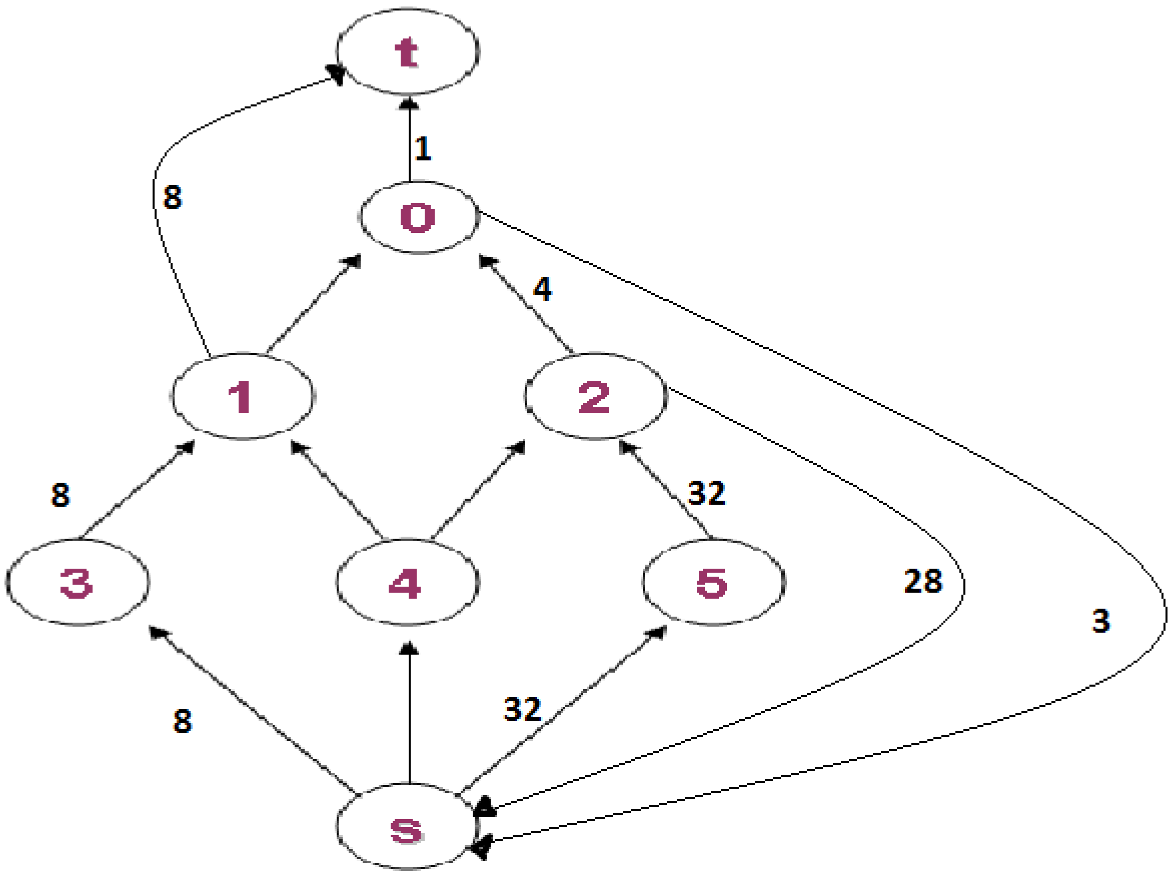
- A monotone boolean circuit's output cannot change from true to false when one input changes from false to true.
- Monotone circuits do not contain : ¬ gates
- Monotone circuit value is circuit value applied to monotone circuits.

C   N




t
1
22
0
2
4
1
2
8
16
16
32
3
4
5
44
5
8
0
32
s

f- standar flow



RNC Algorithms

def. A language L is in RNC if there is a uniform family of NC circuits with the following additional properties :

- C_n specializes in strings of length n and has n+m(n) input gates; m(n) :additional gates, the random bits needed for the algorithm
- If x , of length n is in L then at least half of the $2^{m(n)}$ bit strings y of length m(n) , C_n outputs are TRUE on input x;y
- If $x \notin L$ outputs FALSE on x;y for all y.

An RNC problem

- Does a bipartie graph have a perfect matching?

-Monte Carlo alg. for matching with symbolic determinants

-NC algorithm for arithmetic determinants

- If we know the answer to the decision problem , we compute the perfect matching using dynamic programming techniques.

Minimum-weight perfect-matching

- Each edge has a weight $w_{i,j}$
- We want to minimize $w(\pi) = \sum_{i=1}^{n} w_{i,\pi(i)}$
- There is an algorithm in NC for this problem, which works under 2 conditions

  1) small weights, polynomial in n

  2) π is unique

- We proved that if the minimum weight perfect matching exists and is unique, it can be computed in parallel

- We also know how to test in RNC whether a perfect matching exists

- How can we guarantee that the minimum-weight matching is unique?

The Isolating Lemma

- Suppose that the edges in E are assigned independently and randomly weights between 1 and 2|E|. If a perfect matching exists, then with probability at least ½ the minimum-weight perfect matching is unique.

- Thank you!