Optimizing Multi-level Combinational Circuits for Generating Random Bits

Chen Wang and **Weikang Qian**
University of Michigan-SJTU Joint Institute
Shanghai Jiao Tong University

Random Bits

- Also known as a Bernoulli random variable
  - A special discrete random variable takes 1 with probability p and 0 with probability $1-p$.

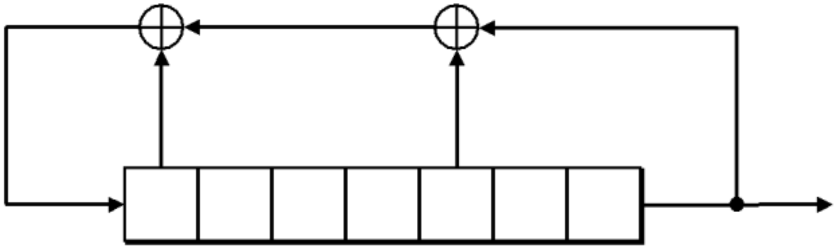
$$P(X) = \begin{cases} p, & \text{if } X = 1 \\ 1-p, & \text{if } X = 0 \end{cases}$$

- Use of random bits
  - Cryptography
  - Monte Carlo simulation
  - Testing of IC chips
    - Weighted random testing

Random Bits Generation

- Pseudorandom number generators (PRNGs)
  - Produce deterministic sequences
  - Require a large amount of hardware

Linear Feedback Shift Register



Random Bits Generation

- Truly random source, such as thermal noise
  - Hard to control the output probability
  - A post-processing unit is needed

- One post-processing method is to synthesize combinational logic to transform source probabilities into target probabilities.

Basic Problem

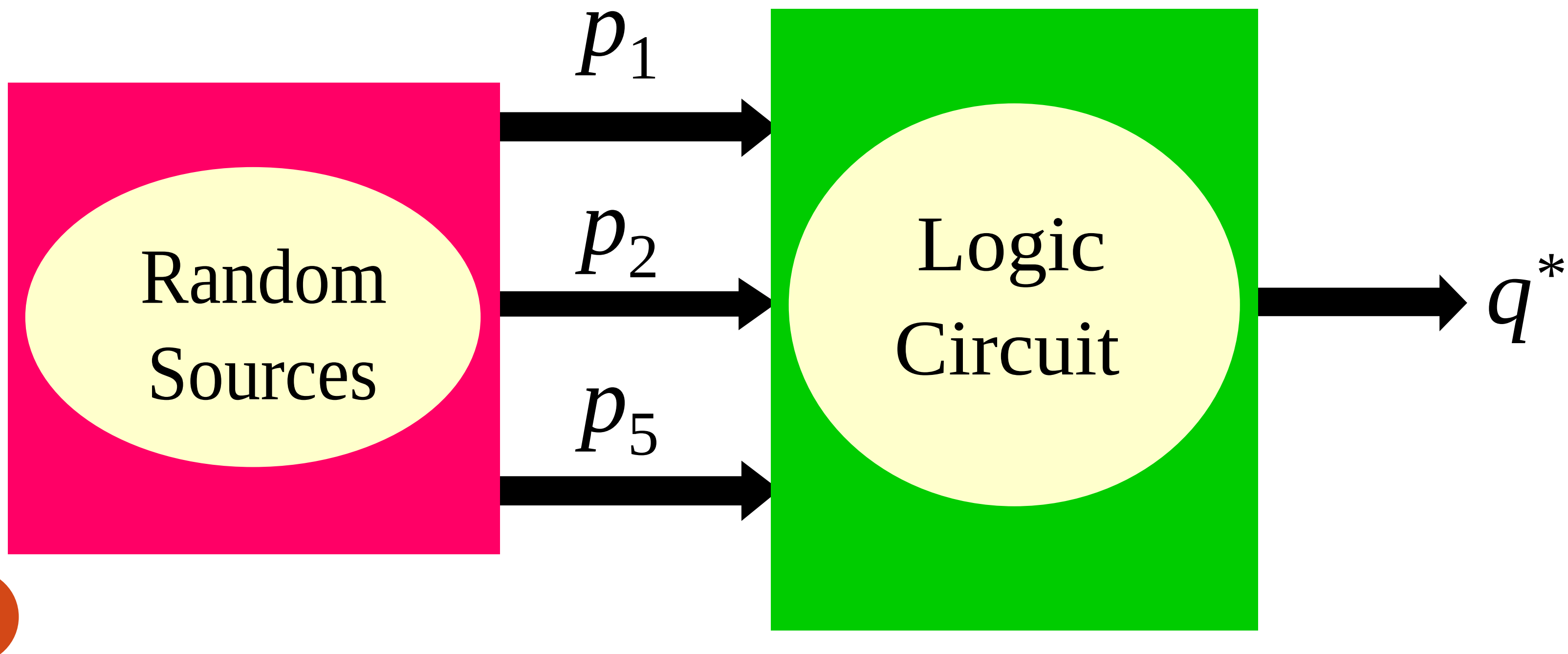
Source Probability Set

$S = \{p_1, p_2, p_3, p_4, p_5, p_6\}$

Target Probability

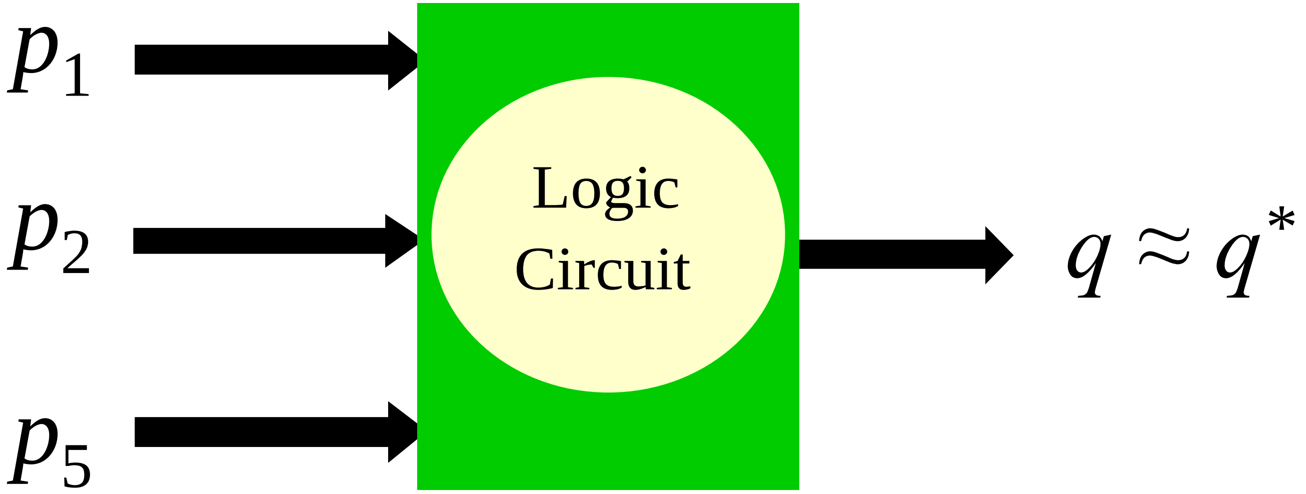
q^*

Synthesize a logic circuit that takes input probabilities from S and outputs the target probability.



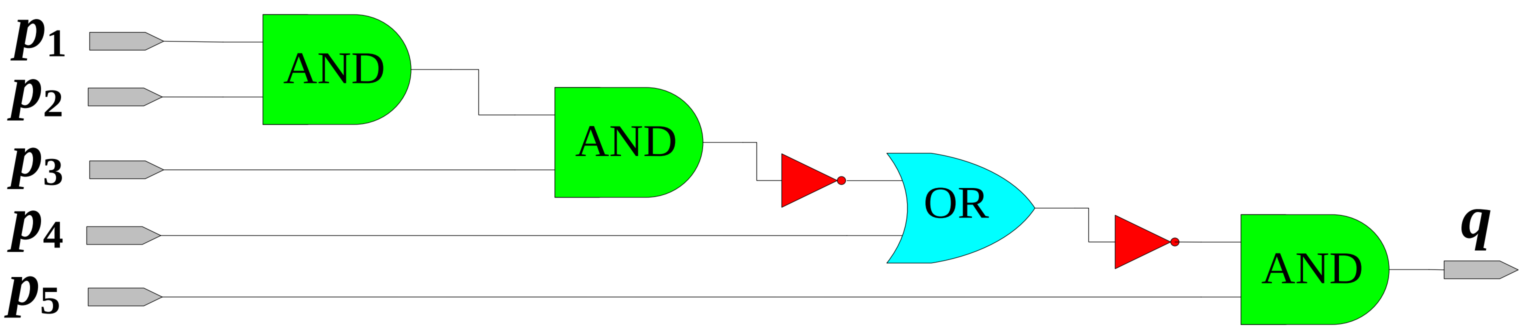
Characteristics of the Problem

- All the inputs from set S are independent.
- Each probability in the set S is used at most once.
- The output probability usually cannot be realized exactly .
  - We find the closest implementation.



Previous Method

- Qian et al. "Transforming probabilities with combinational logic."
  - A greedy method is applied to incrementally build the circuit.
  - However, the synthesized circuit is in the form of a gate chain, not satisfactory in depth and output accuracy.

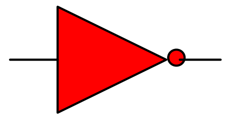


Our Contribution

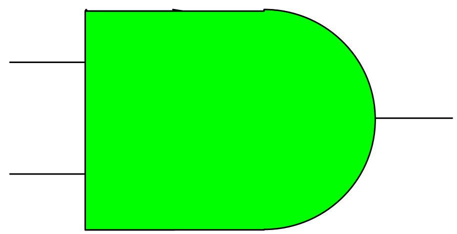
- A new algorithm for synthesizing combinational circuits to generate random bits from some input probabilities:
  - Expand the search space to include tree-style circuits
    - The circuits synthesized by our algorithm have much smaller depths and output errors.
  - Apply linearity property to simplify the problem.
  - Apply iterative search method to find the best solution.

Probabilistic Computation with Basic Gates

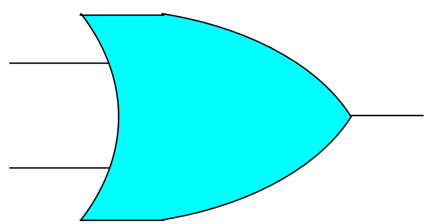
Inverter

p  $1 - p$

AND gate

p_1, p_2  $p_1 \cdot p_2$

OR gate

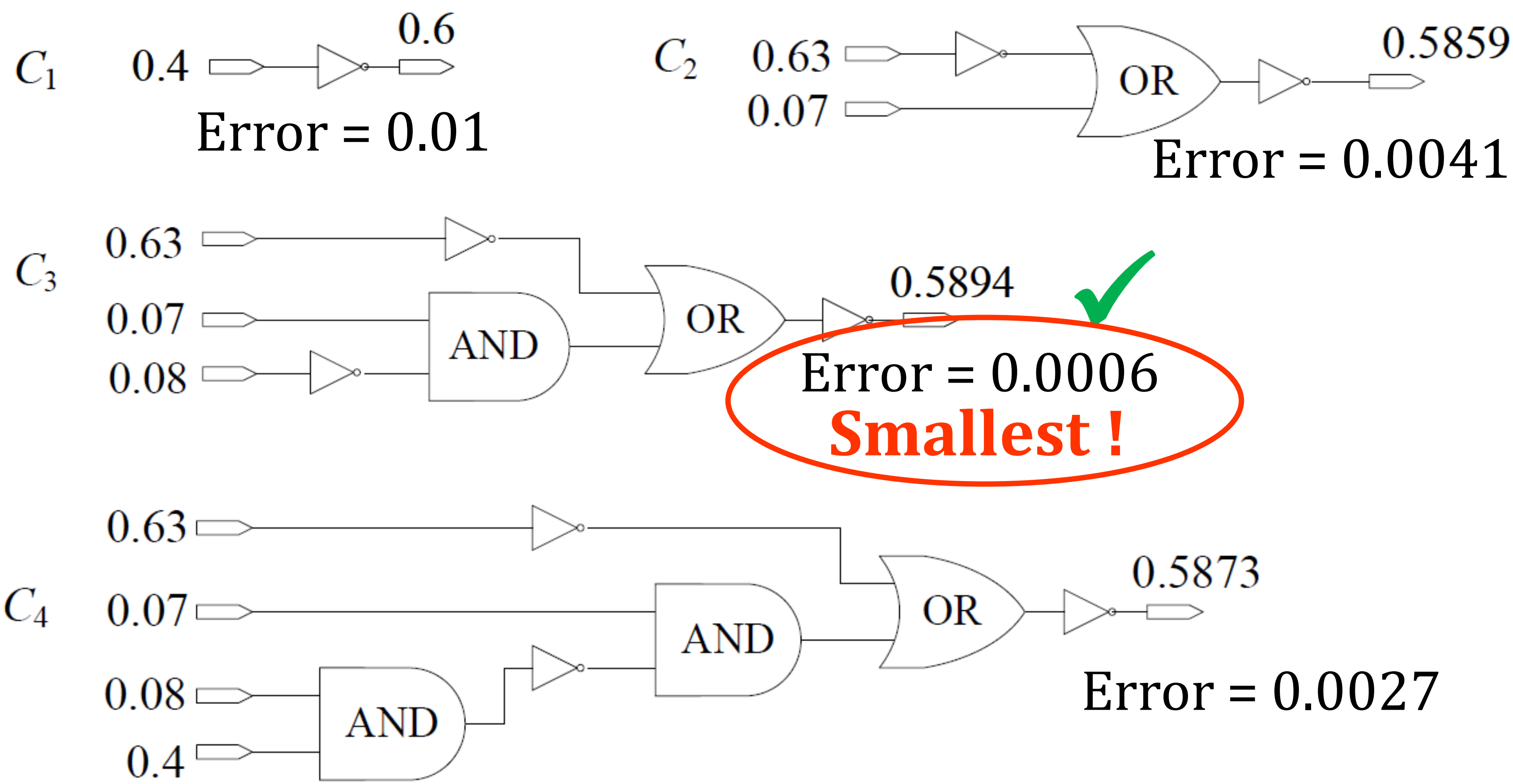
p_1, p_2  $p_1 + p_2 - p_1 \cdot p_2$

Main Procedure of Our Method

- Suppose $|S| = n$. Synthesize a series of circuits C_1 ,..., C_n.
  - Circuit C_k has k inputs.
- The best one of $C_1,\ldots,C_n$ is chosen.

Example

S = {0.07, 0.08, 0.4, 0.63}; Target q^* = 0.59



Constructing C_k

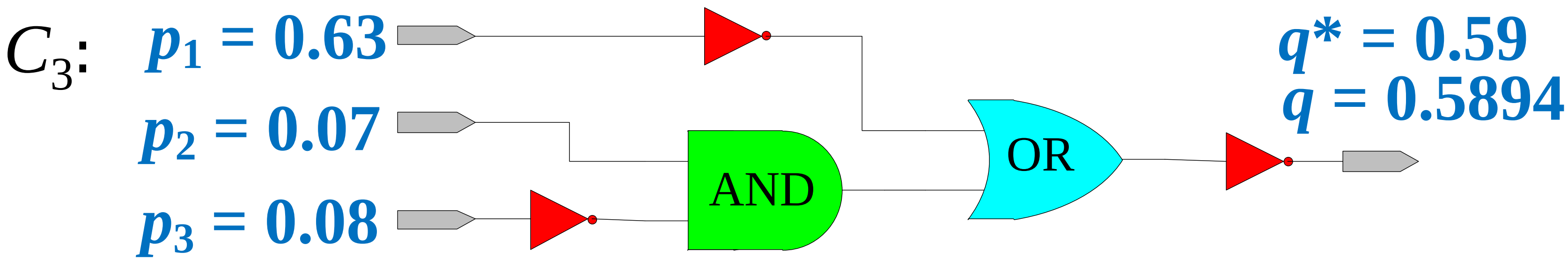
- Circuit C_k is built from $C_{k\text{-}1}$ by replacing one of $C_{k\text{-}1}$'s inputs by a new gate with two input probabilities.
  - Other part of $C_{k\text{-}1}$ is kept unchanged.

- All inputs of $C_{k\text{-}1}$ are examined to find the best place to replace.

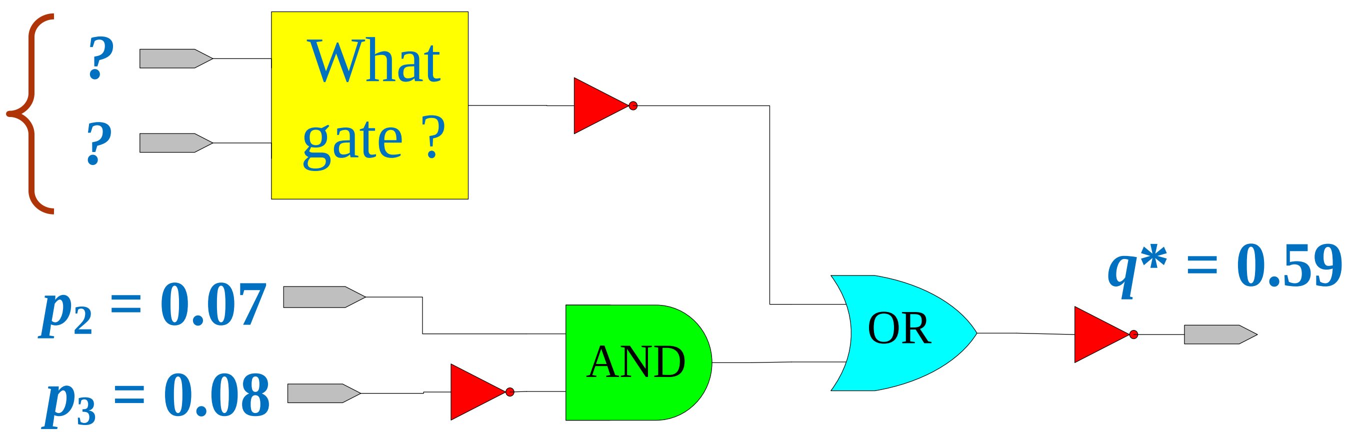
Example of Building C_4 from C_3

Source probability set

S = {0.07, 0.08, 0.3, 0.4, 0.63}; q^* = 0.59



Chosen from S' = {0.3, 0.4, 0.63}

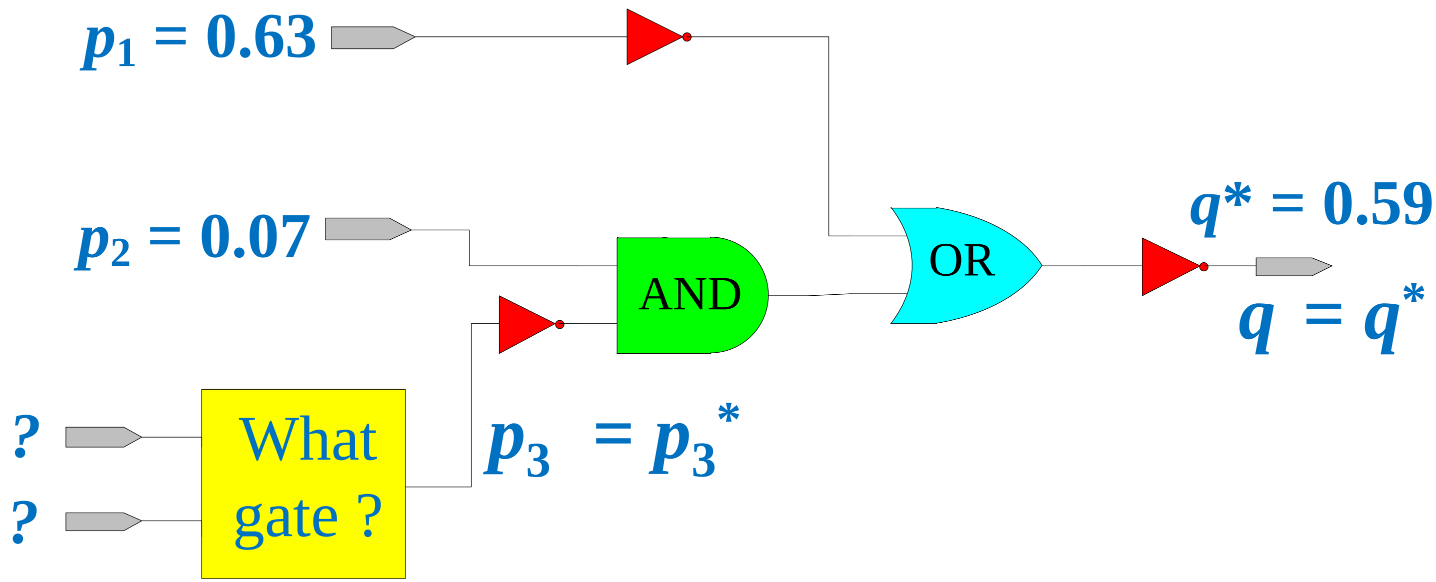


Determine the Added Gate Type and its Inputs

- For each input, determine the optimal gate type and its input probabilities.

- Achieved by two steps:
  1. Calculate the **ideal probability** for the input.
  2. Choose the gate type and its two inputs to make the output of the gate be closest to the ideal probability.

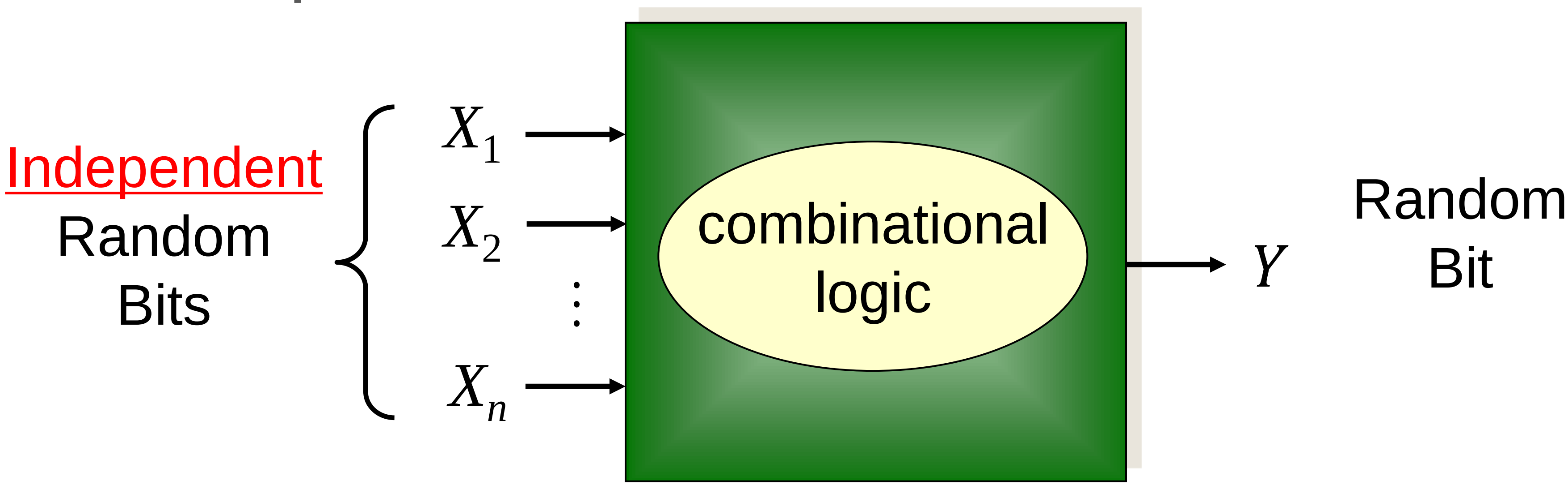
Ideal Probability p_i^*



Ideal probability p_3^*:

If $p_3 = p_3^*$, then the output probability $q = q^*$.

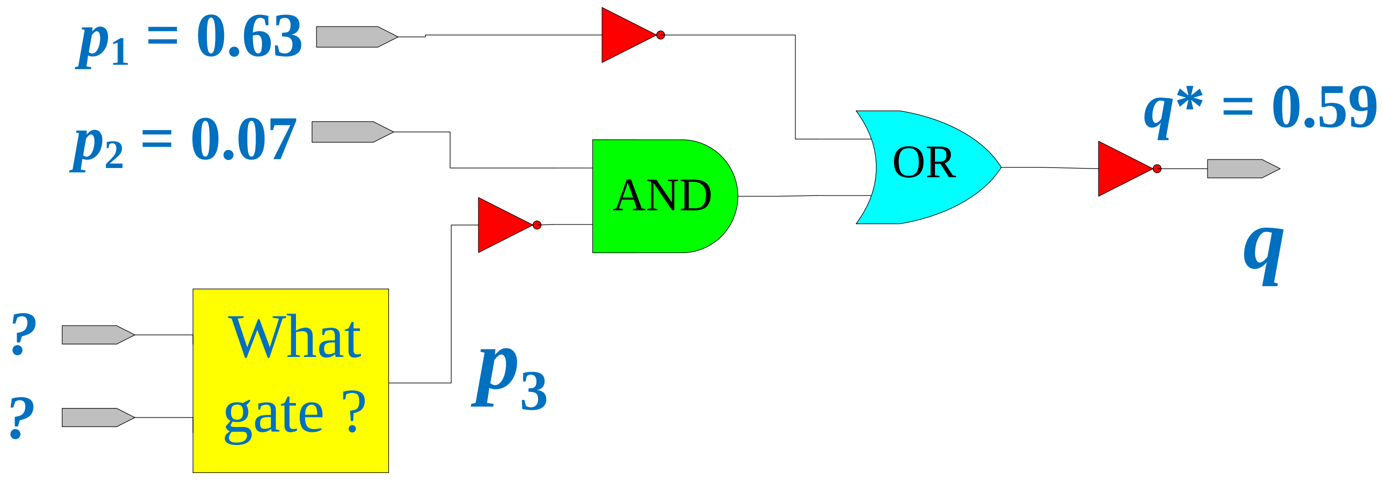
Linearity Property of Probabilistic Logic Computation



$$P(X_i = 1) = p_i \qquad P(Y = 1) = q = F(p_1, \ldots, p_n)$$

F is a multivariate linear polynomial on $p_1, \ldots, p_n$.

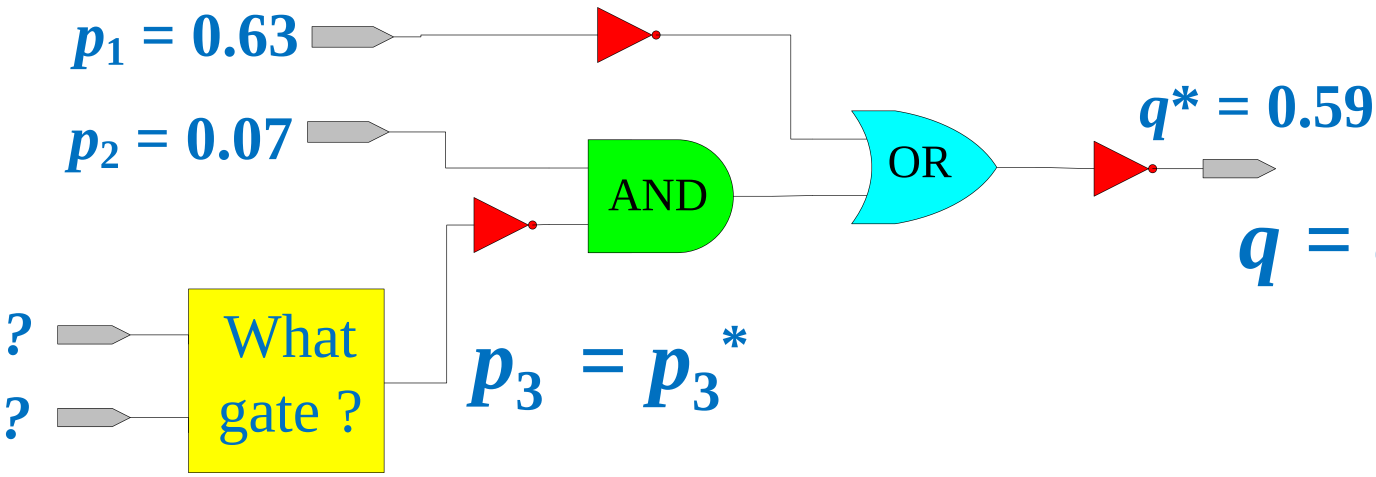
Linearity Property between Input and Output



$$q = a \cdot p_3 + b$$

a and b are constant values related to the values of p_1, p_2 and the structure of the circuit.

Linearity Property between Input and Output



$$q = q^*$$

$$p_3 = p_3^*$$

$$\left.\begin{aligned} q &= a \cdot p_3 + b \\ q^* &= a \cdot p_3^* + b \end{aligned}\right\} \Rightarrow |q - q^*| = |a| \cdot |p_3 - p_3^*|$$

Simplify the Search with Ideal Probability

$$|q - q^*| = |a| \cdot |p_3 - p_3^*|$$

- If we choose p_3 such that $|p_3 - p_3^*|$ is minimal, then the output error $|q - q^*|$ is also minimal.

- Therefore, we can simplify the optimization problem by choosing p_3 closest to p_3^*, rather than calculating q value and comparing it with q^*.
  - "Localize" the search problem.
  - Save a lot of the computation.

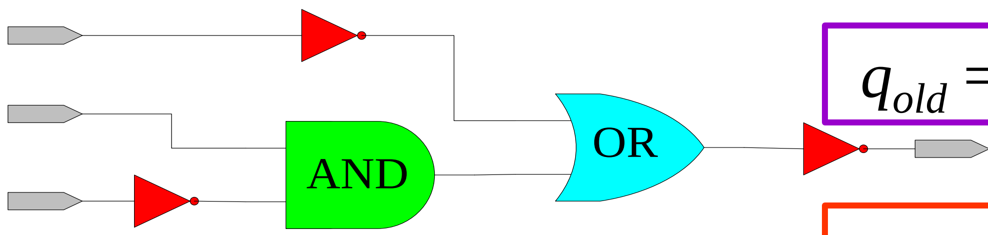
Obtain Ideal Probability

$p_1 = 0.63$

$p_2 = 0.07$

$\boxed{p_{3\_old} = 0.08}$

$\boxed{p_3 = 1}$



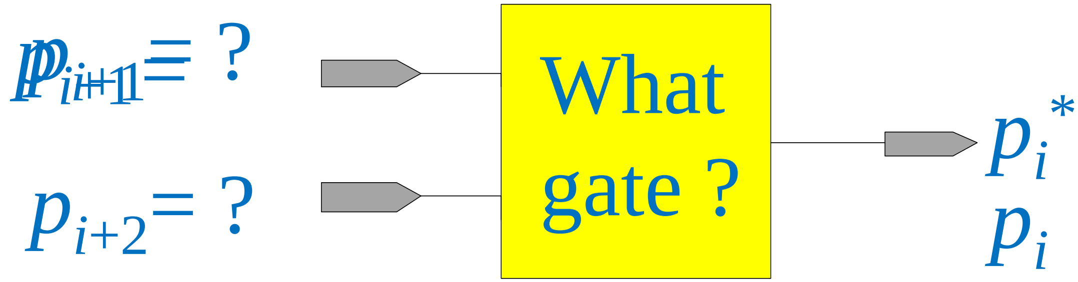
$\boxed{q_{old} = 0.5894}$

$\boxed{q = q' = 0.63}$

$$\left.\begin{aligned} q_{old} &= a \cdot p_{3\_old} + b \\ q' &= a \cdot 1 + b \end{aligned}\right\} \Rightarrow \text{Obtain constants } a \text{ and } b$$

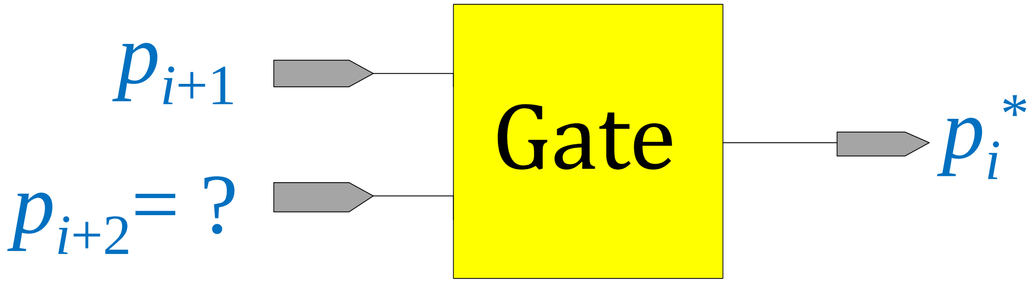
$$q^* = a \cdot p_3^* + b \Rightarrow p_3^* = (q^* - b) / a.$$

Obtain the Best Local Replacement



- Initially, the upper input p_{i+1} is assigned as p_i , the probability value at the input of the previous circuit.
- Then, determine the gate type
  - If $p_{i+1} < p_i^*$, choose *OR* gate. (The output probability of *OR* gate is **larger** than its inputs.)
  - If $p_{i+1} \geq p_i^*$, choose *AND* gate. (The output probability of *AND* gate is **smaller** than its inputs.)

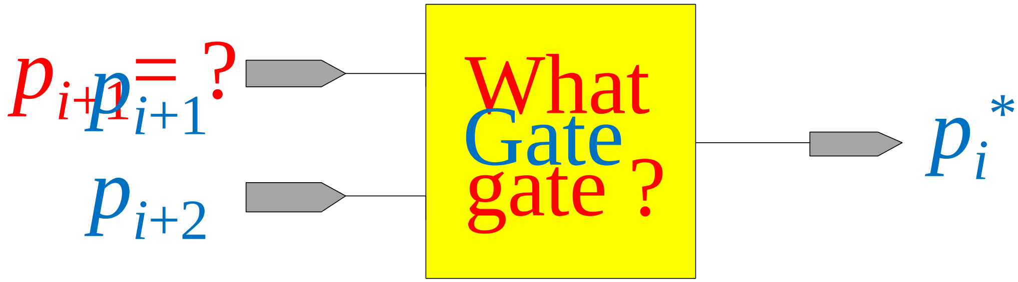
Determine p_{i+2}



- First, determine the ideal probability p_{i+2}^*
  - If the gate is *AND*, $p_{i+2}^* = p_i^* / p_{i+1}$.
  - If the gate is *OR*, $p_{i+2}^* = (p_i^* - p_{i+1}) / (1 - p_{i+1})$.

- Choose p_{i+2} as the closest value to p_{i+2}^* in S' (the set of remaining probabilities).
  - Inverters can be added to the two inputs of the gate.

Obtain the Best Local Replacement



- Obtain the gate type and input values iteratively.
  - When p_{i+2} is chosen, release the value of p_{i+1} and the gate type.
  - Obtain gate type and p_{i+1} in a similar way.

- Terminate when the error of the output $|p_i - p_i^*|$ stops decreasing.

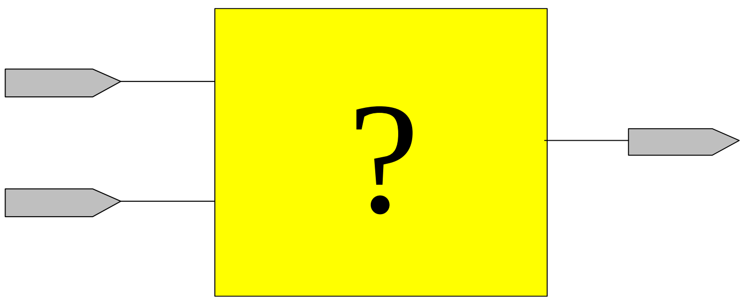
Example

Initial condition:

$S' = \{0.07, 0.15, 0.25\}$

$p_{i+1} = ?$

$p_{i+2} = ?$



$p_i^* = 0.3$

$p_i = 0.8$

Example

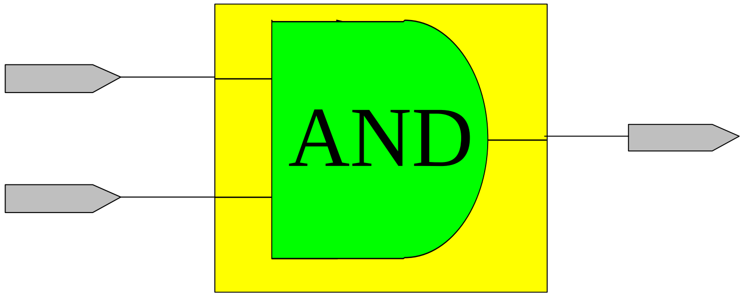
First iteration:

$S' = \{0.07, 0.15, 0.25\}$

$\boxed{0.8 > 0.3}$

$p_{i+1} = ?$

$p_{i+2} = ?$



$p_i^* = 0.3$

$p_i = 0.8$

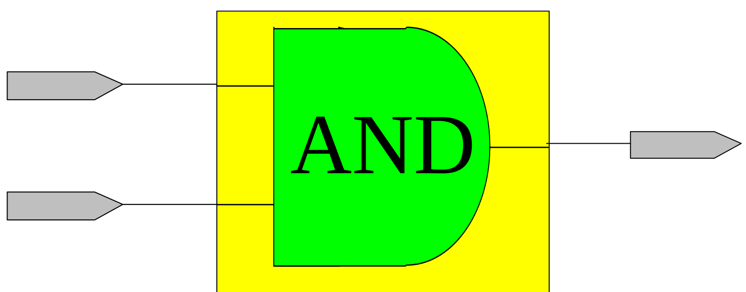
Example

First iteration:

$S' = \{0.07, 0.15, 0.25\}$

$$\boxed{p_{i+2}^{*} = p_i^{*} / p_{i+1}}$$

$p_{i+1} = 0.8$



$p_i^{*} = 0.3$

$p_{i+2} = ?$

$p_{i+2}^{*} = 0.375$

Example

First iteration:

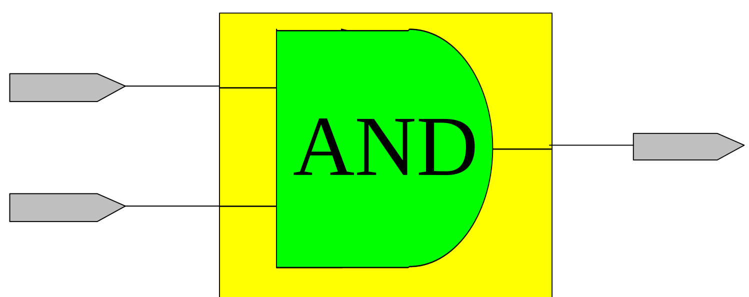
$S' = \{0.07, 0.15, 0.25\}$

$\boxed{\text{Choose } 0.25 \text{ in } S' \text{ for } p_{i+2}}$

$p_{i+1} = 0.8$

$p_{i+2} = 0.25$

$p_{i+2}^{*} = 0.375$



$p_i^{*} = 0.3$

Example

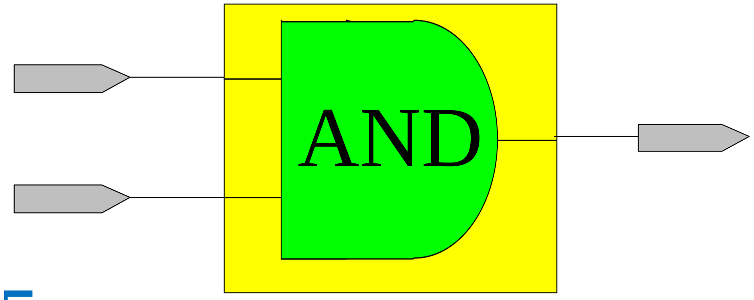
First iteration result:

$S' = \{0.07, 0.15\}$

$p_{i+1} = 0.8$

$p_{i+2} = 0.25$

$p_{i+2}^{*} = 0.375$



$p_i^{*} = 0.3$

$p_i = 0.2$

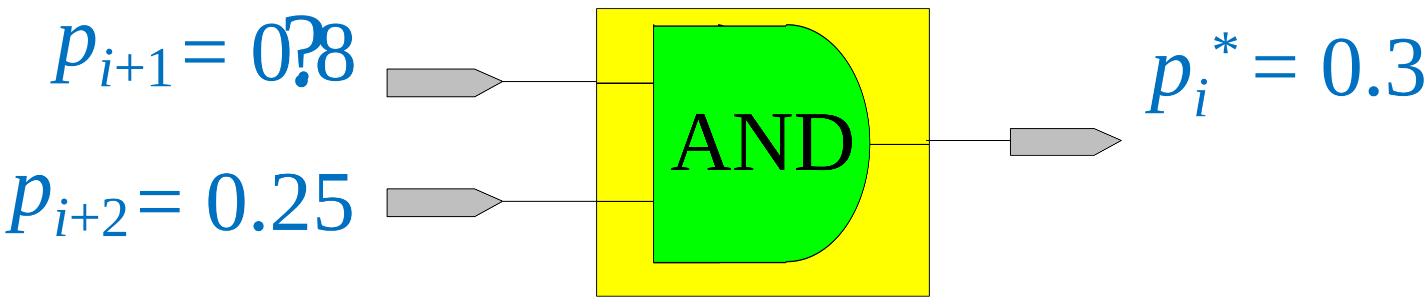
$\text{error} = 0.1$

Example

Second iteration:

$S' = \{0.07, 0.15, 0.8\}$

Release p_{i+1} and gate type, update S' set



Example

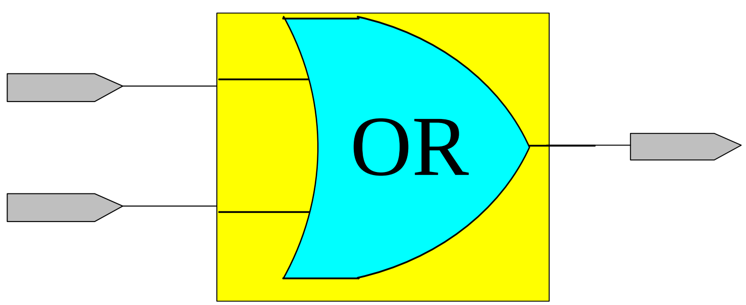
Second iteration:

$S' = \{0.07, 0.15, 0.8\}$

$\boxed{0.25 < 0.3}$

$p_{i+1} = \ ?$

$p_{i+2} = 0.25$



$p_i^* = 0.3$

Example

Second iteration:

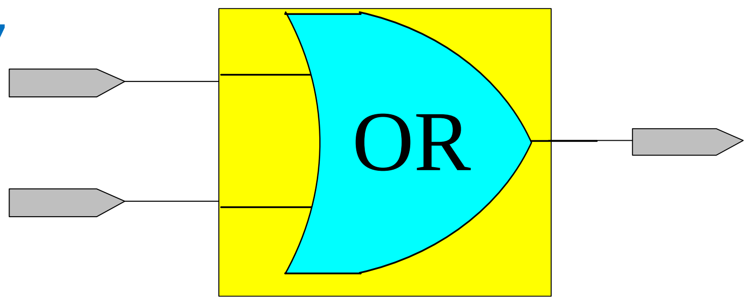
$S' = \{0.07, 0.15, 0.8\}$

Choose 0.07 in S' for p_{i+1}

$p_{i+1}^{*}=0.0667$

$p_{i+1}=0.07$

$p_{i+2}=0.25$



$p_i^{*}=0.3$

Example

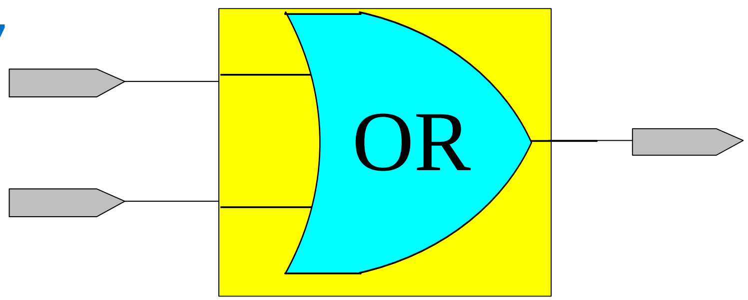
Second iteration result:

$S' = \{0.15, 0.8\}$

$p_{i+1}^{*} = 0.0667$

$p_{i+1} = 0.07$

$p_{i+2} = 0.25$



$p_i^{*} = 0.3$

$p_i = 0.3025$

$\text{error} = 0.0025$

Example

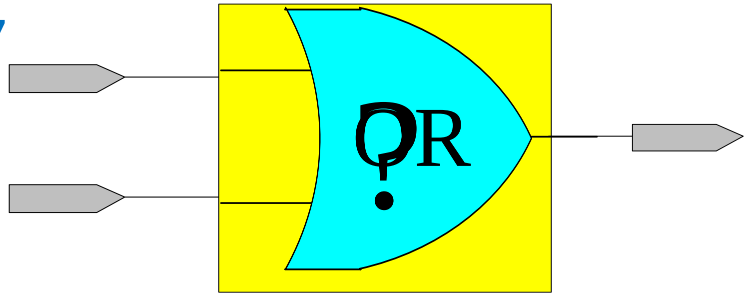
Third iteration:

S' = {0.25, 0.15, 0.8}

Release p_{i+2} and gate type, update S' set

$p_{i+1}=0.07$

$p_{i+2}=0.25$?



$p_i^* = 0.3$

Example

Third iteration result:

$S' = \{0.15, 0.8\}$

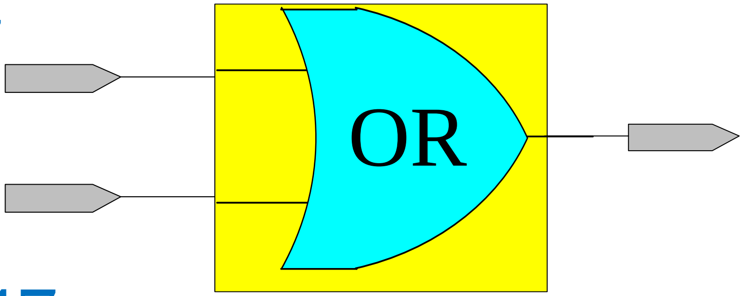
The output error stops decreasing at this iteration.

Terminate !

$p_{i+1} = 0.07$

$p_{i+2} = 0.25$

$p_{i+2}^{*} = 0.247$



$p_i^{*} = 0.3$

$p_i = 0.3025$

$\text{error} = 0.0025$

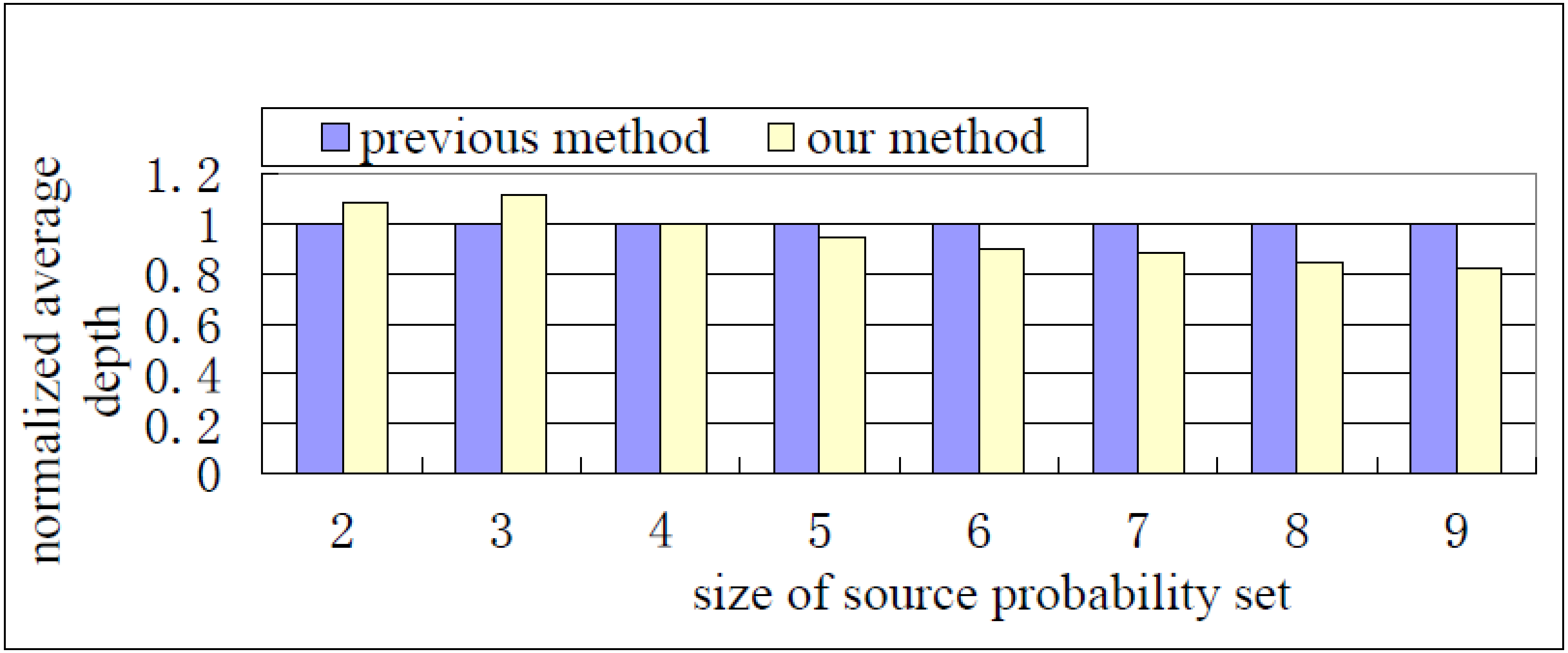
First Set of Experiments

- Objective: Synthesizing a circuit with output probability q such that $|q - q^*|$ is minimal.

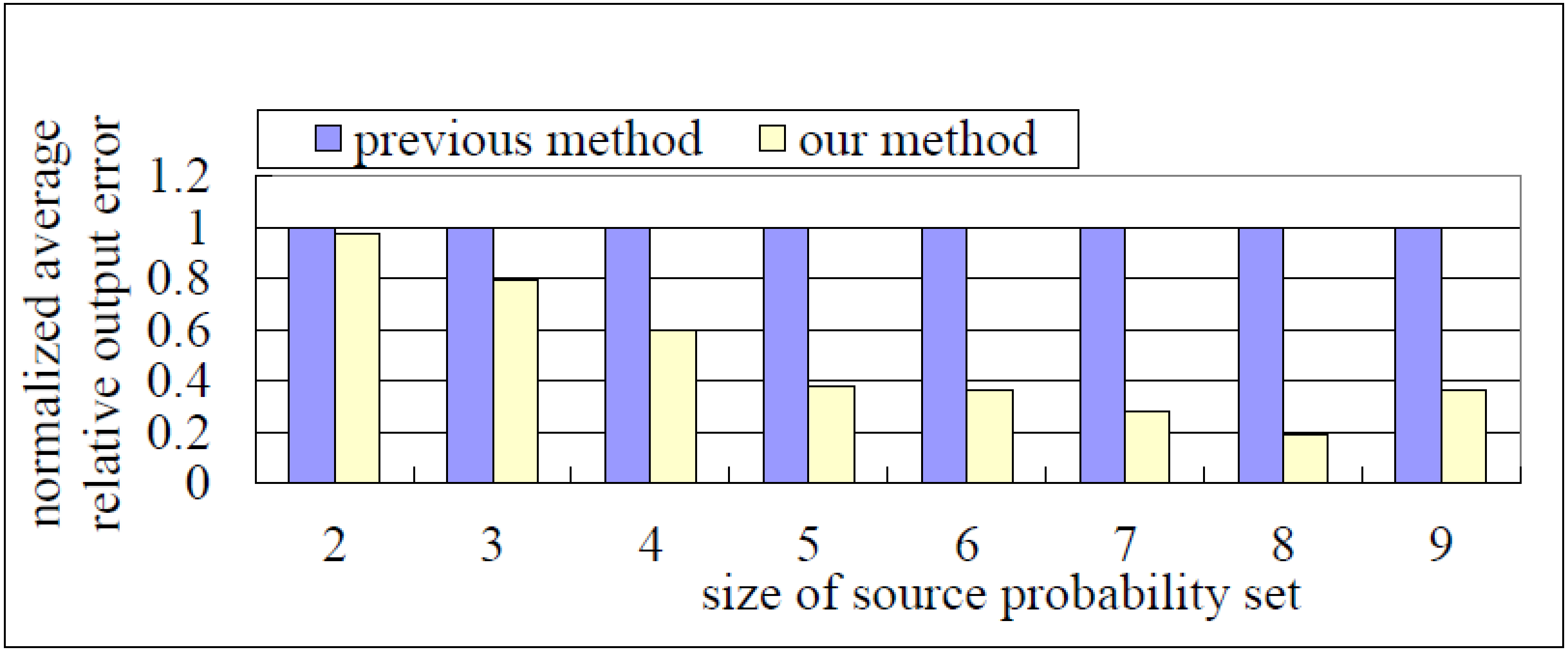
- Test cases
  - Size of source probability set S is from 2 to 9.
  - 800 test cases are generated randomly for each size. Each test case satisfies: when the previous method is applied to it, the output error $|q - q^*| \geq 0.001 \cdot q^*$.

- Apply our algorithm to each of the test cases, obtain the statistic results.

Normalized Average Depth vs. Size of Source Probability Set



Normalized Average Relative Output Error vs. Size of Source Probability Set

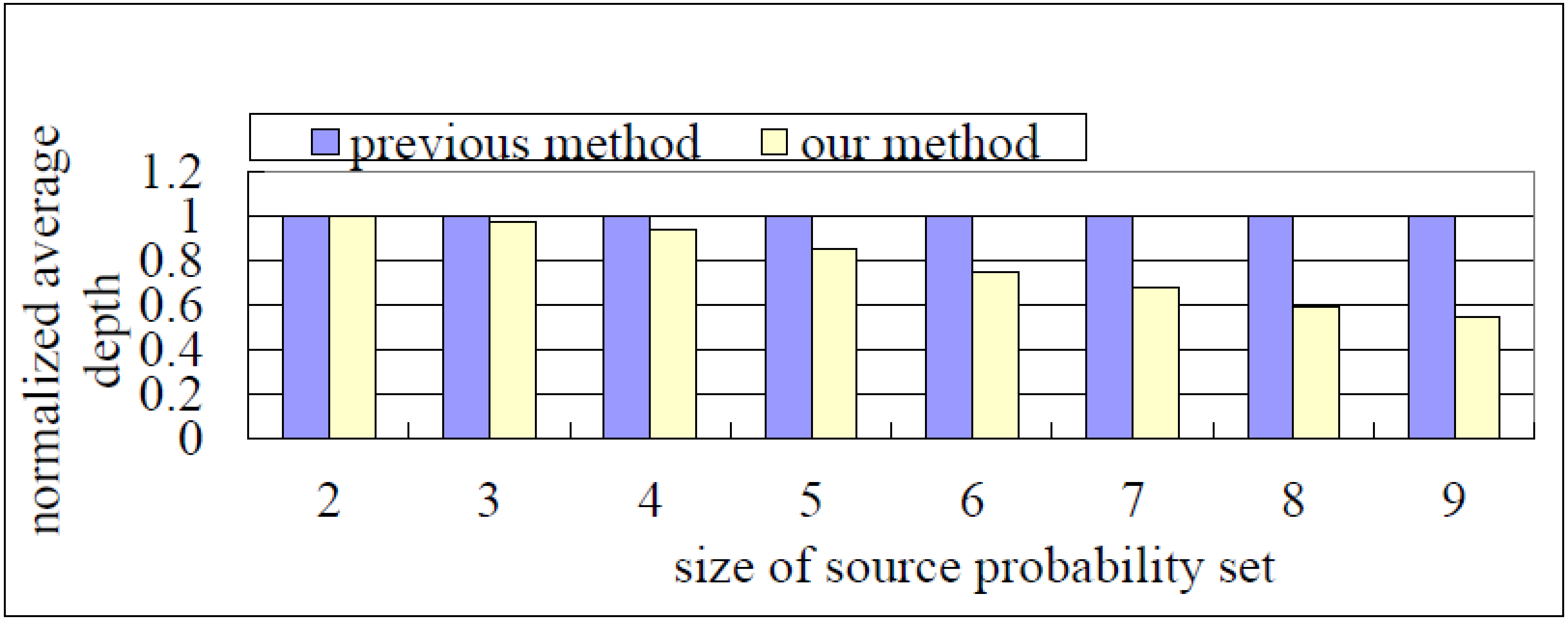


Second Set of Experiments

- Objective: Synthesizing a circuit with minimal depth, whose output probability q satisfies $|q - q^*| \leq e \cdot q^*$, where e is a given error tolerance ratio.

- The test cases are the same as that in the first set of experiments.

Normalized Average Depth vs. Size of Source Probability Set



Conclusion

- We propose a new algorithm for synthesizing combinational circuits to transfer source probabilities into target probabilities.
- We apply a linearity property of probabilistic logic computation and an iterative local search method to increase the efficiency of our program.
- The circuits have much smaller depths and output errors.

Future Work

- Find a global optimization method to improve the solution to the random bit generation problem.

Thank You !

Questions ?