A Network-Flow Based Valve-Switching Aware Binding Algorithm for Flow-Based Microfluidic Biochips

18th Asia and South Pacific Design Automation Conference

Kai-Han Tseng

Sheng-Chi You, Wajid Hassan Minhass*, Tsung-Yi Ho, Paul Pop*

Department of Computer Science and Information Engineering

National Cheng Kung University, Taiwan

*Department of Informatics and Mathematical Modeling
Technical University of Denmark, Denmark

DTU

Outline

- Introduction
- Preliminary
- Problem Formulation
- Algorithms
- Experimental Results
- Conclusion

What is Microfluidic?

- **Microfluidics** deals with the behavior, precise control and manipulation of **fluids** that are geometrically constrained to a small, typically sub-millimeter, scale.
- Typically, **micro** means one of the following features:
  - small volumes (*μ*L, *n*L, *p*L, *f*L)
  - small size
  - low energy consumption
  - effects of the micro domain

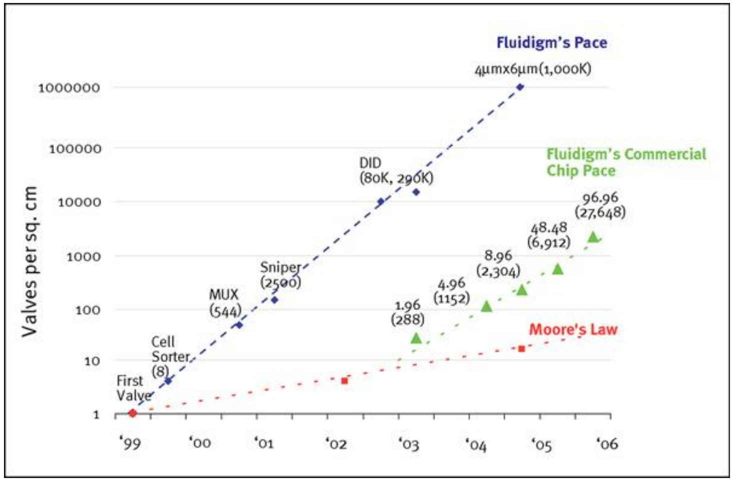
WIKIPEDIA
The Free Encyclopedia

Benefits of Microfluidics

- Economy of Scales
  - Volume reductions by several orders of magnitude over benchtop experiments
  - Extreme cost reduction for biological experiments
  - Rare samples (stem cells) can be studied in more detail
- Integration
  - Thousands of complex experiments can be performed in parallel
  - Integration with solid state optics, MEMS, and NEMS detectors
- Automation
  - All steps can be fully automated, reducing labor costs
- Cheap Mass-production

The Need of CAD Support

- Applications become more complicated
  - Large-scale bioassays
  - Multiple and concurrent assay operations on a biochip
- Design complexity is increased
  - The increasing rate of the valve numbers is four times faster than Moore's Law
- Current methodologies
  - Manual
  - Full-custom



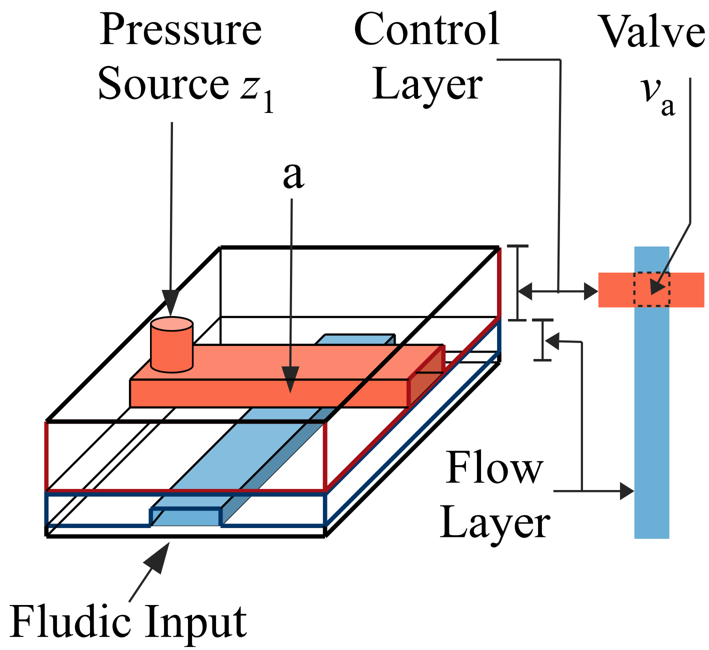
Source: Fluidigm

Outline

- Introduction
- Preliminary
- Problem Formulation
- Algorithms
- Experimental Results
- Conclusion

Valve: The Basic Element of Microfluidics

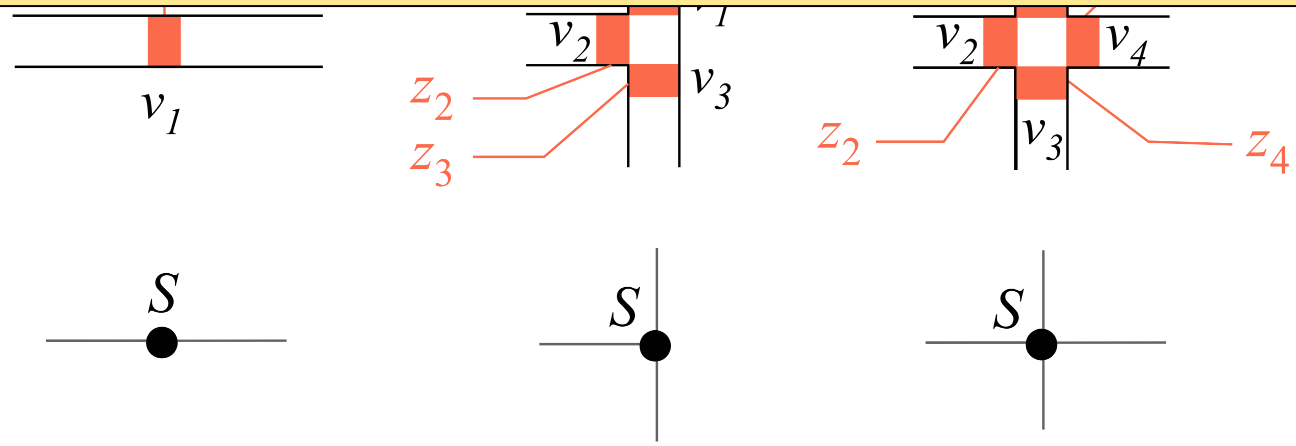
- Technology: multi-layer soft lithography
- Fabrication substrate: elastomers (e.g., PDMS)
  - Good biocompatibility
  - Optical transparency



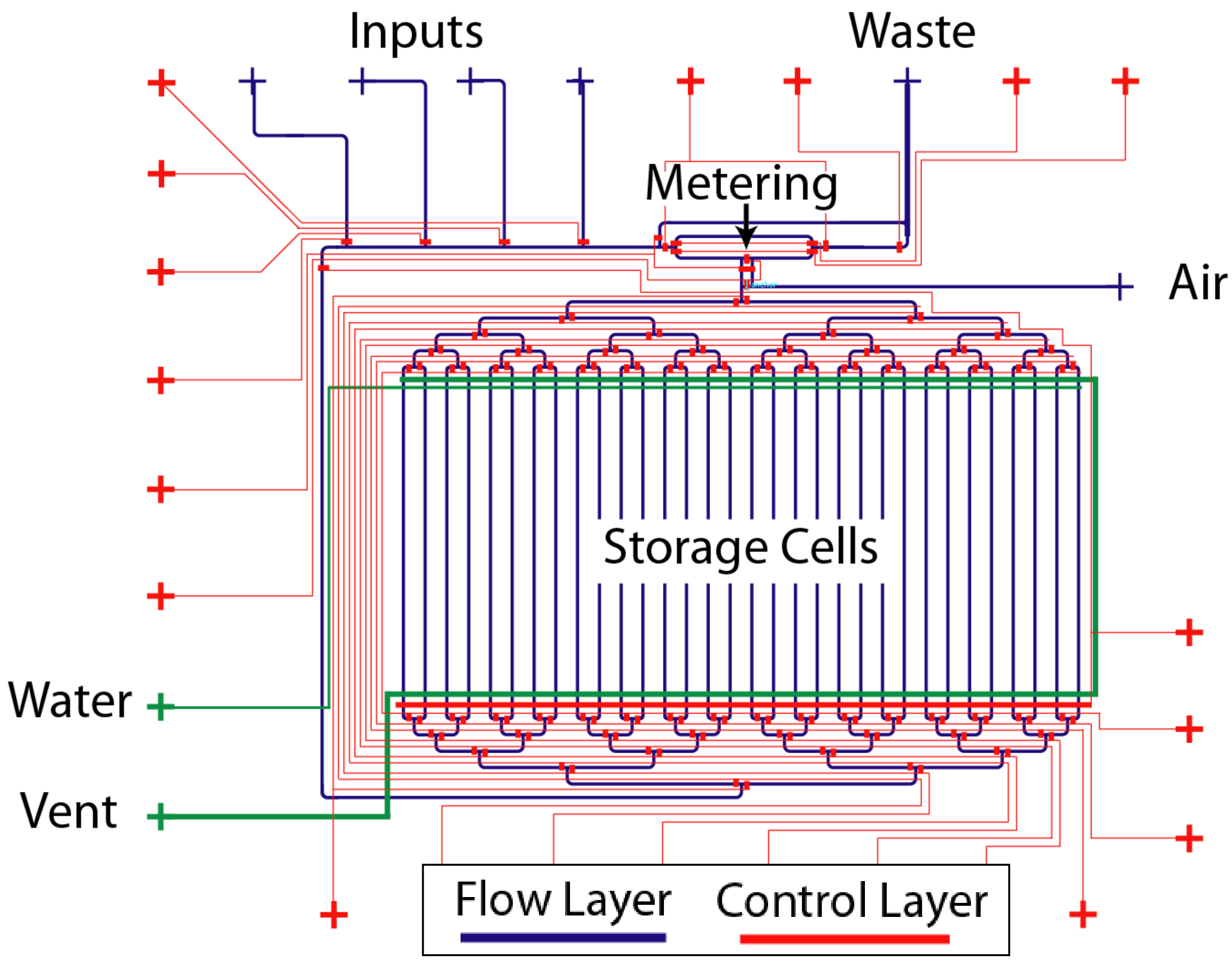
Valve: The Basic Element of Microfluidics

Valves combined to form more complex units,
e.g., latches, switches, mixers, multiplexers, micropumps.

The valves have the problem of reliability

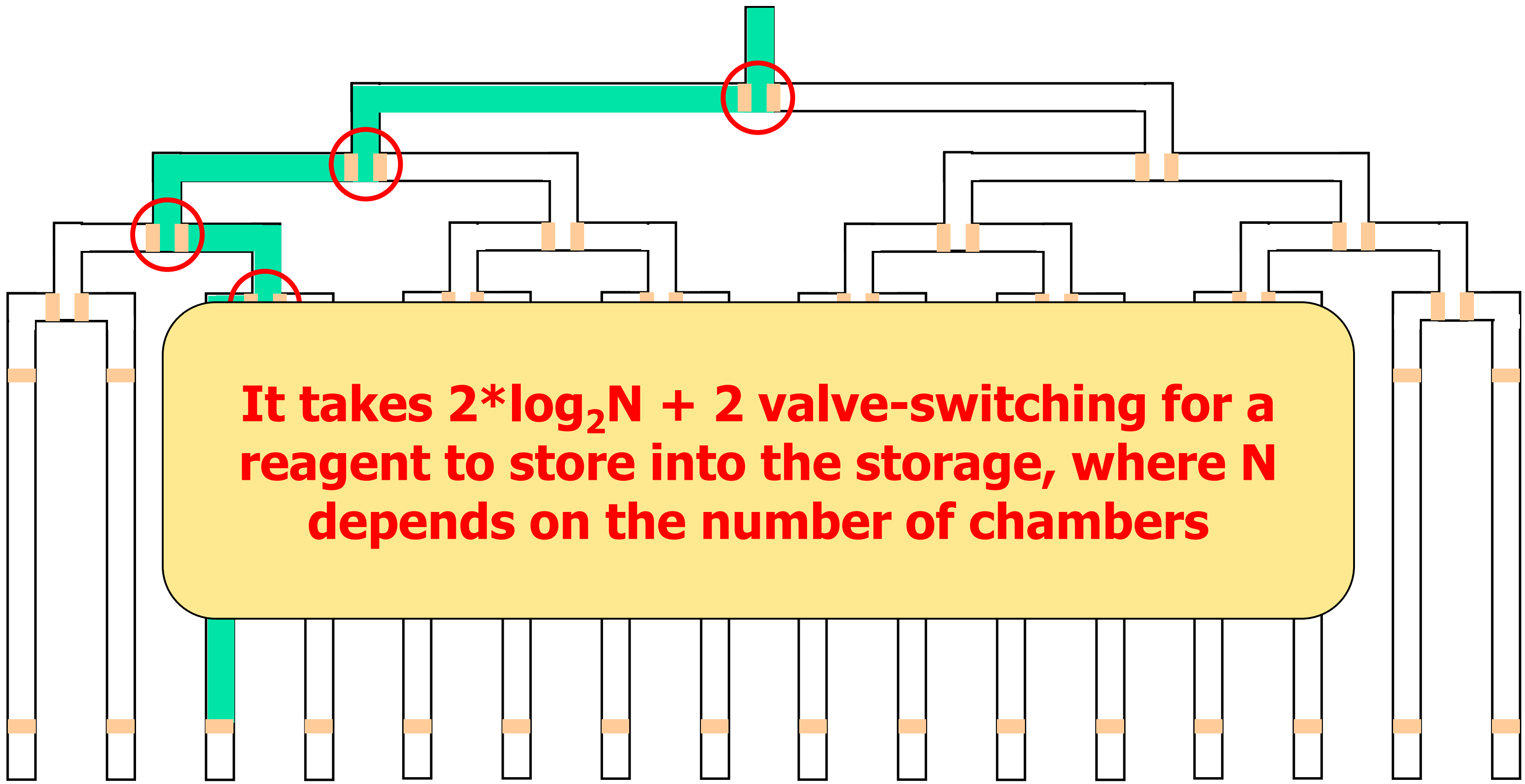


Component Model: Storage



Component Model

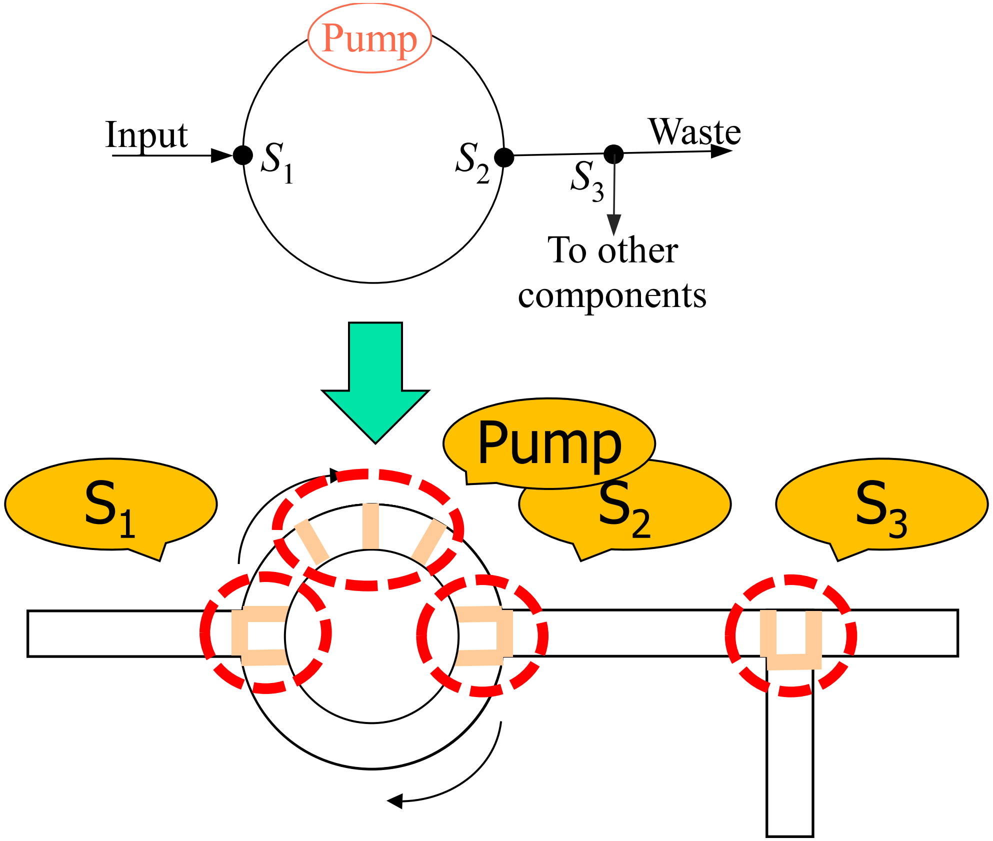
Storage



It takes $2*\log_2 N + 2$ valve-switching for a reagent to store into the storage, where N depends on the number of chambers

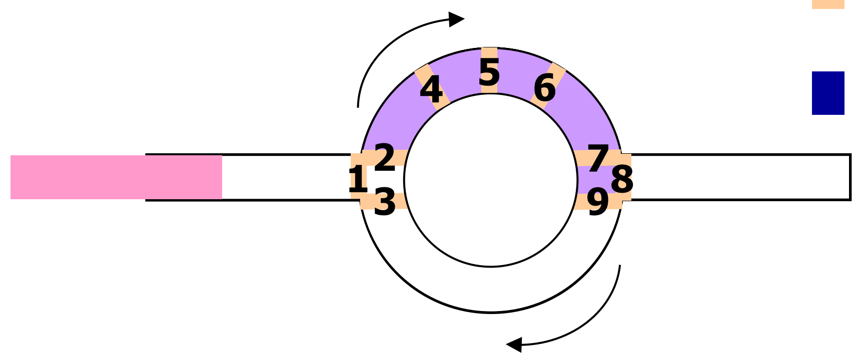
Component Model: Mixer

Microfluidic mixer



Valve-Switching for Mixing Operation

Microfluidic mixer



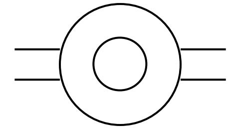
Open the valve

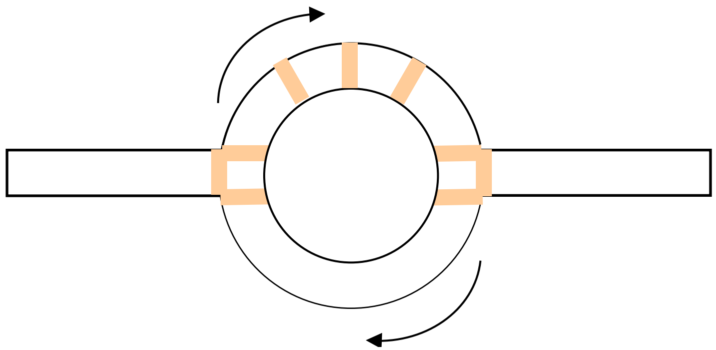
Close the valve

1: Valve-switching 20 + 3r Times

Phase	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
1. Ip1	0	0 / 1	0	0	0	0	0 / 1	0	0
2. Ip2	0	0	0 / 1	0	0	0	0	0	0 / 1
3. Mix	0 / 1	0	0	0 / Mix	0 / Mix	0 / Mix	0	0 / 1	0
4. Op1	0	0 / 1	0	0	0	0	0 / 1	0	0
5. Op2	0	0	0 / 1	0	0	0	0	0	0 / 1

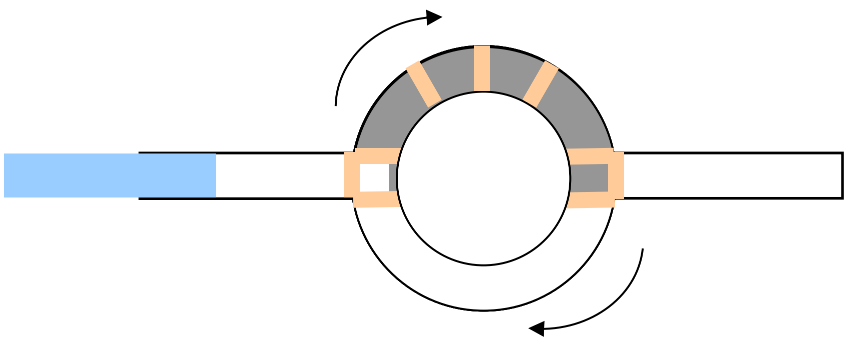
Motivation

Valves reagents  Mixer



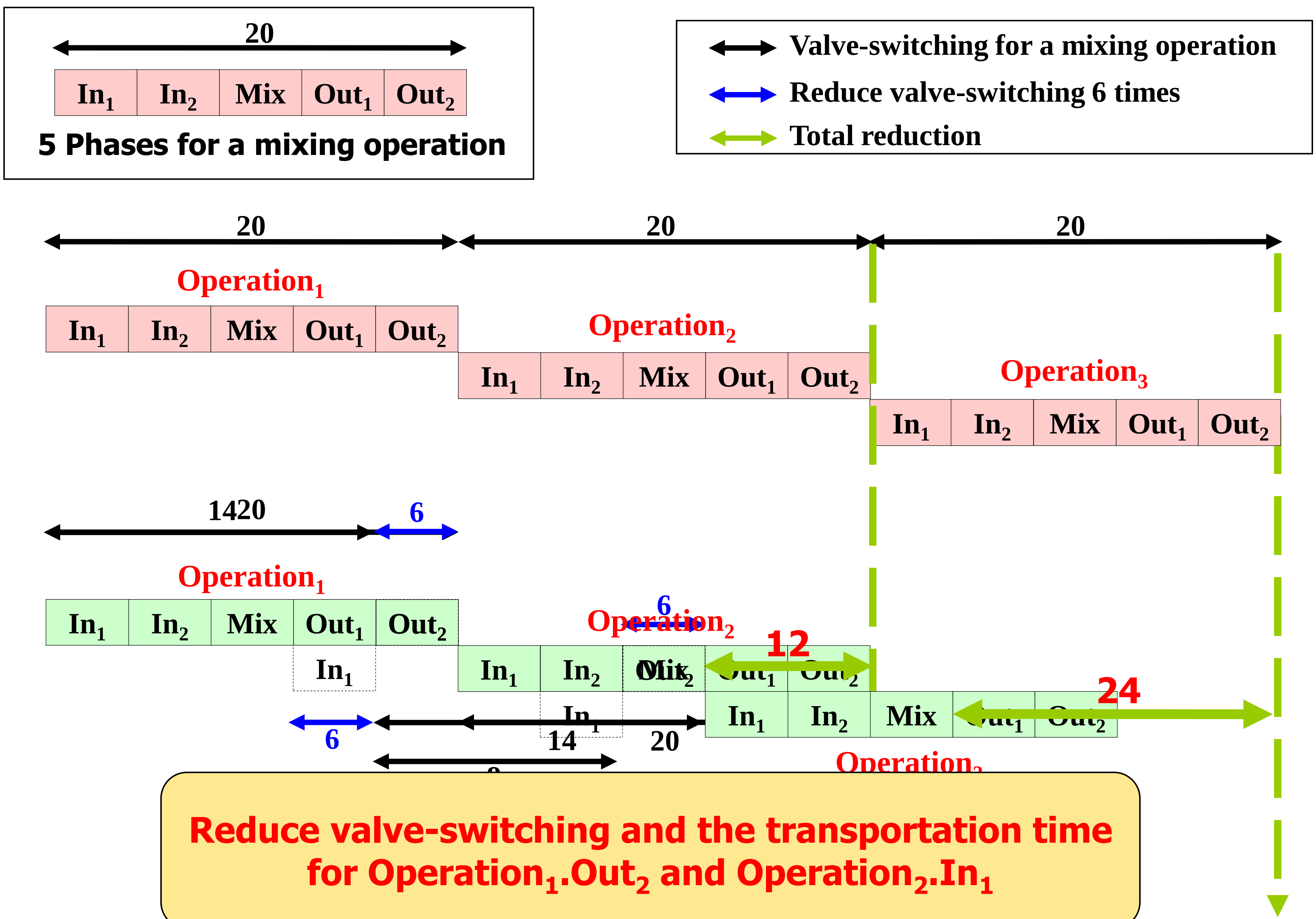
1: Valve-switching 20 + 3r Times

Case1:
When two reagents mix,
it takes **20+3r** valve-switching to finish the mixing operation



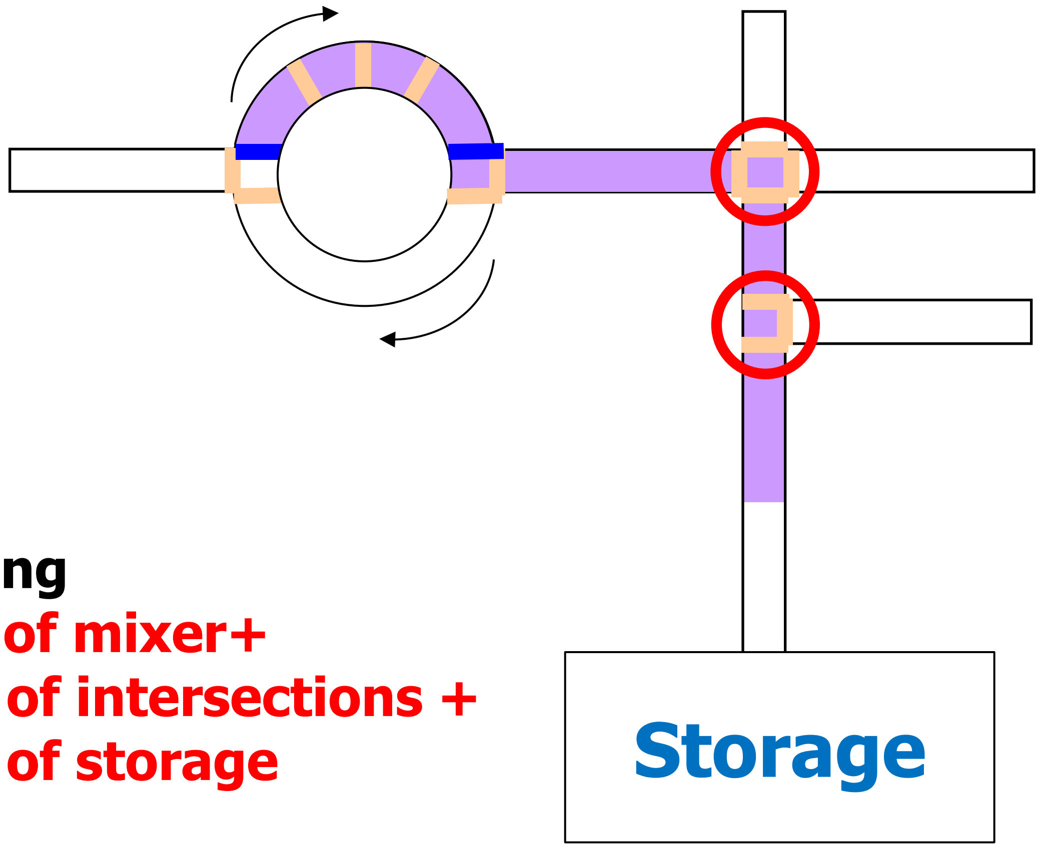
2: Valve-switching 14 + 3r Times

Caes2:
If we leave one reagent in the same component, we can reduce **6** times of valve-switching



Reduce valve-switching and the transportation time for Operation$_1$.Out$_2$ and Operation$_2$.In$_1$

Another Advantage



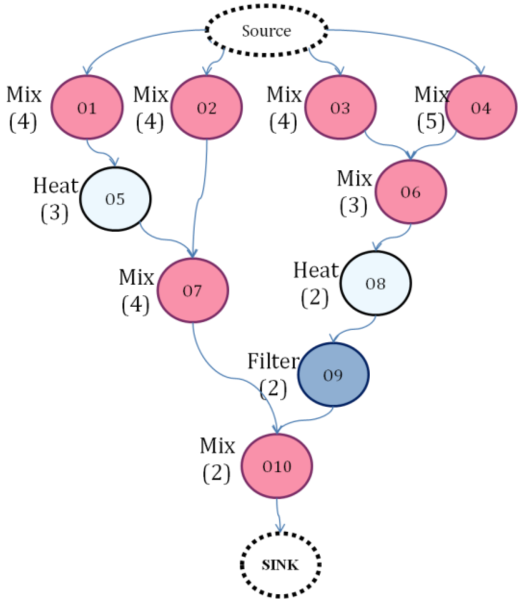
# of valve-switching
= Valve-switching of mixer+
Valve-switching of intersections +
Valve-switching of storage

Outline

- Introduction
- Preliminary
- Problem Formulation
- Algorithms
- Experimental Results
- Conclusion

Problem Formulation

- **Input:** A biochemical application modeled as a sequential graph and a component library
- **Objective:** Obtain a resource binding result such that the total valve-switching amount and the application complete time is minimized
- **Constraint:** Resource constraint

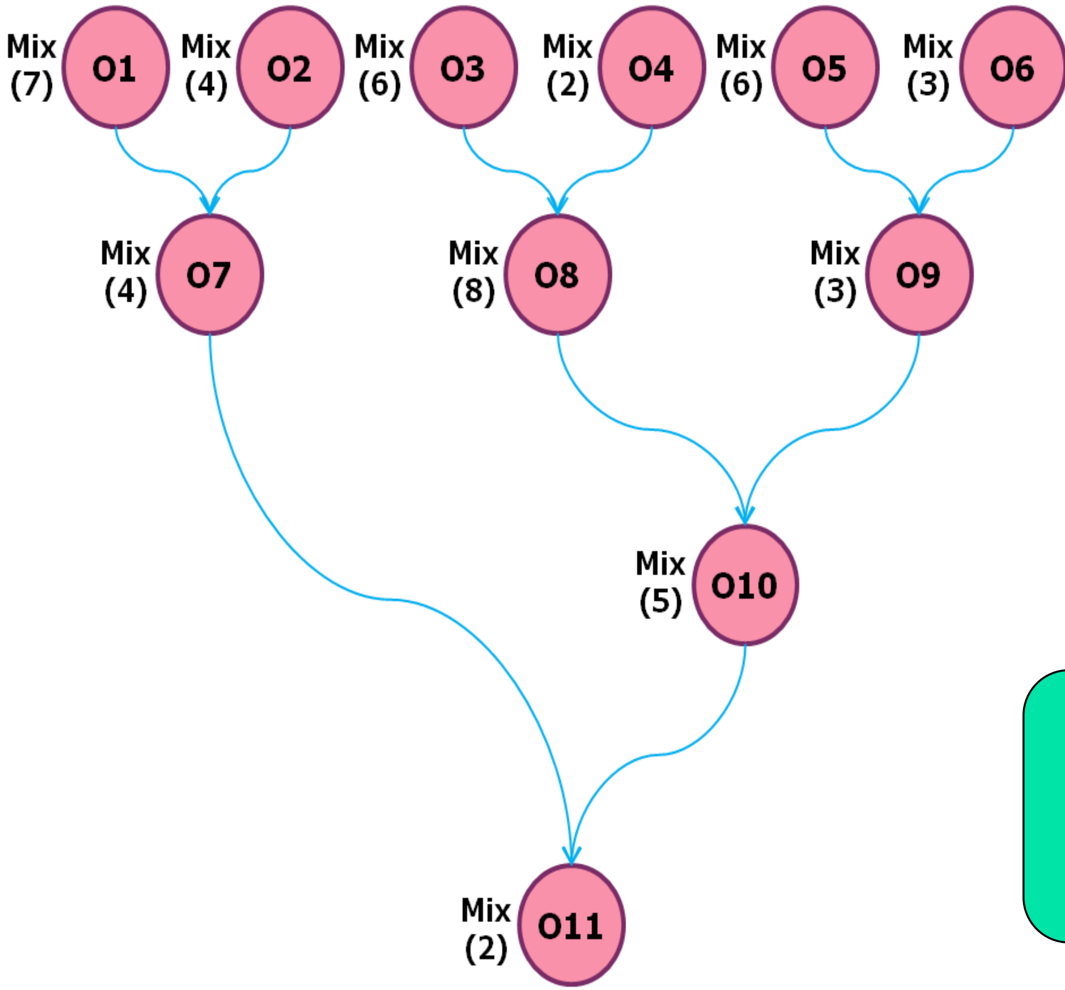


Component	Phases
Mixer	Ip_1 / Ip_2 / Mix / Op_1 / Op_2
Filter	Ip / Filter / Op_1 / Op_2
Detector	Ip / Detect / Op
Separator	Ip_1 / Ip_2 / Separat/ Op_1 / Op_2
Heater	Ip / Heat / Op

Outline

- Introduction
- Preliminary
- Problem Formulation
- Algorithms
- Experimental Results
- Conclusion

Input



Application Graph

Component	Phases
Mixer	Ip_1 / Ip_2 / Mix / Op_1 / Op_2
Filter	Ip / Filter / Op_1 / Op_2
Detector	Ip / Detect / Op
Separator	Ip_1 / Ip_2 / Separat/ Op_1 / Op_2
Heater	Ip / Heat / Op

Component Library

# of Mixers : 3

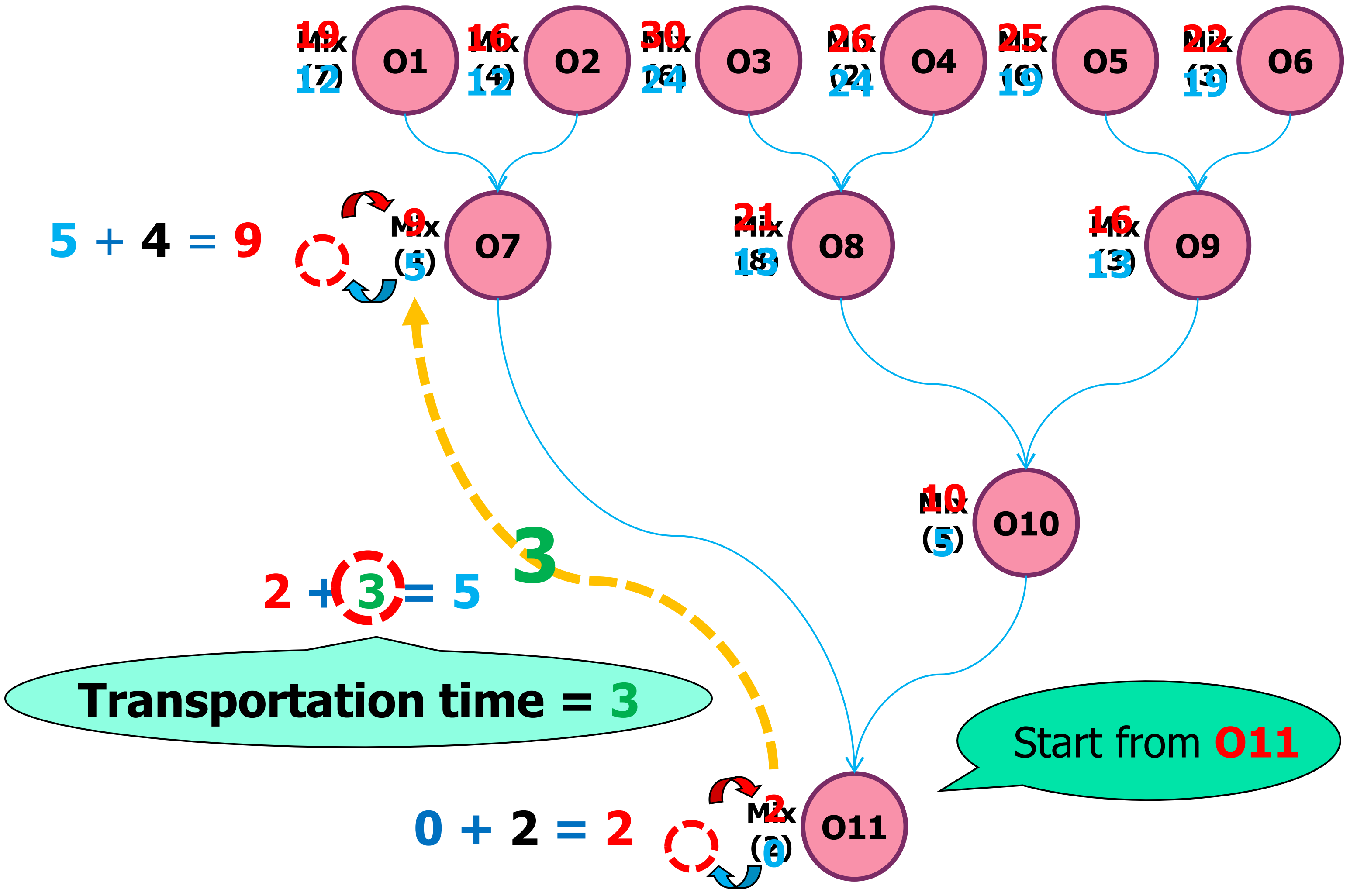
Given Components

Baseline Method : List Scheduling Based Binding Algorithm

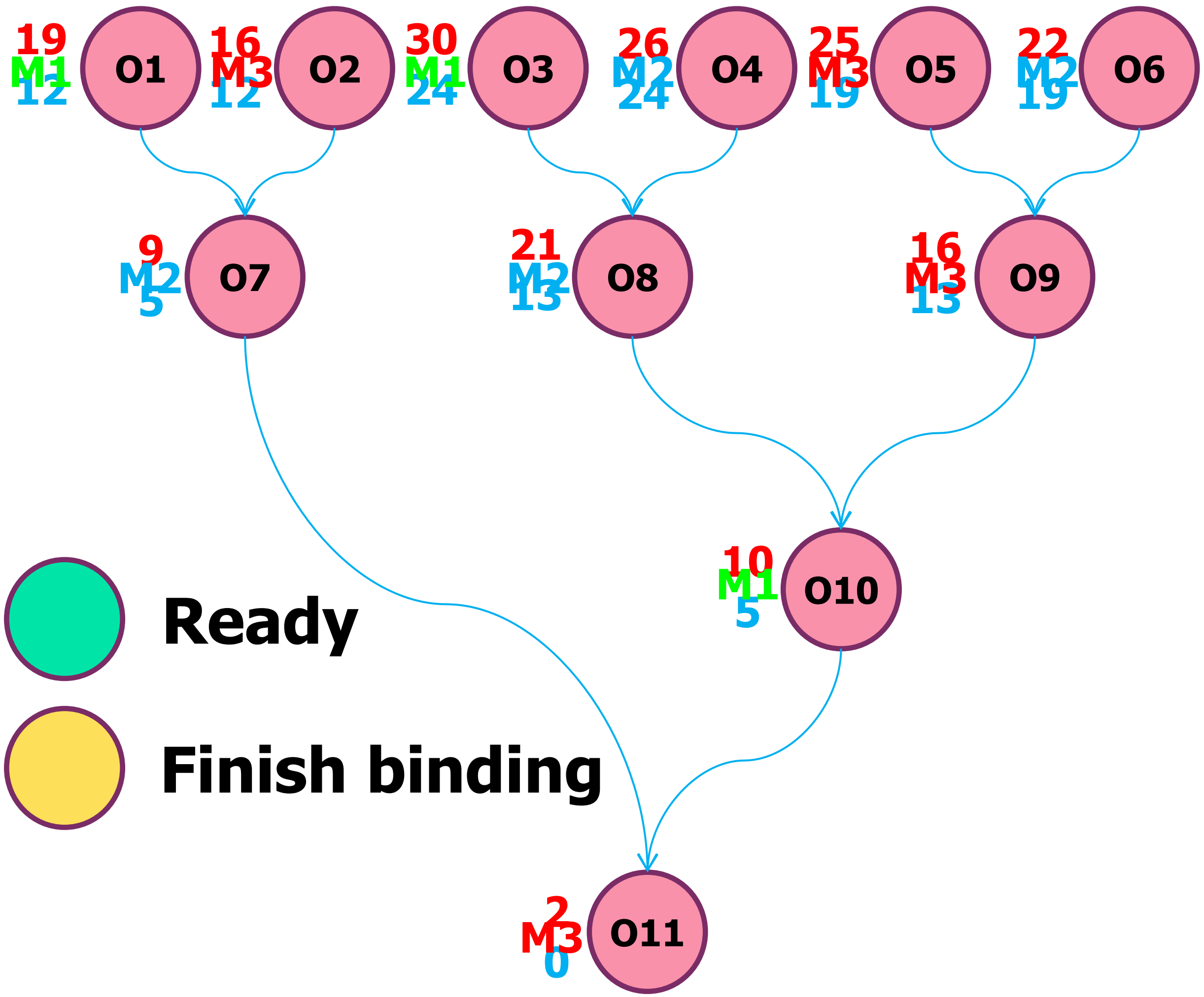
- Topological Sort
  - Apply topological sort for the application graph to **compute the urgency criteria** for the operations

- Binding Strategy
  - An operation is seen as ready only if it's previous operations were already bound to the components
  - **Bind the ready operations to the components based on their urgency criteria**, the operations having bigger urgency criteria will have higher priority

Topological Graph / Application Graph



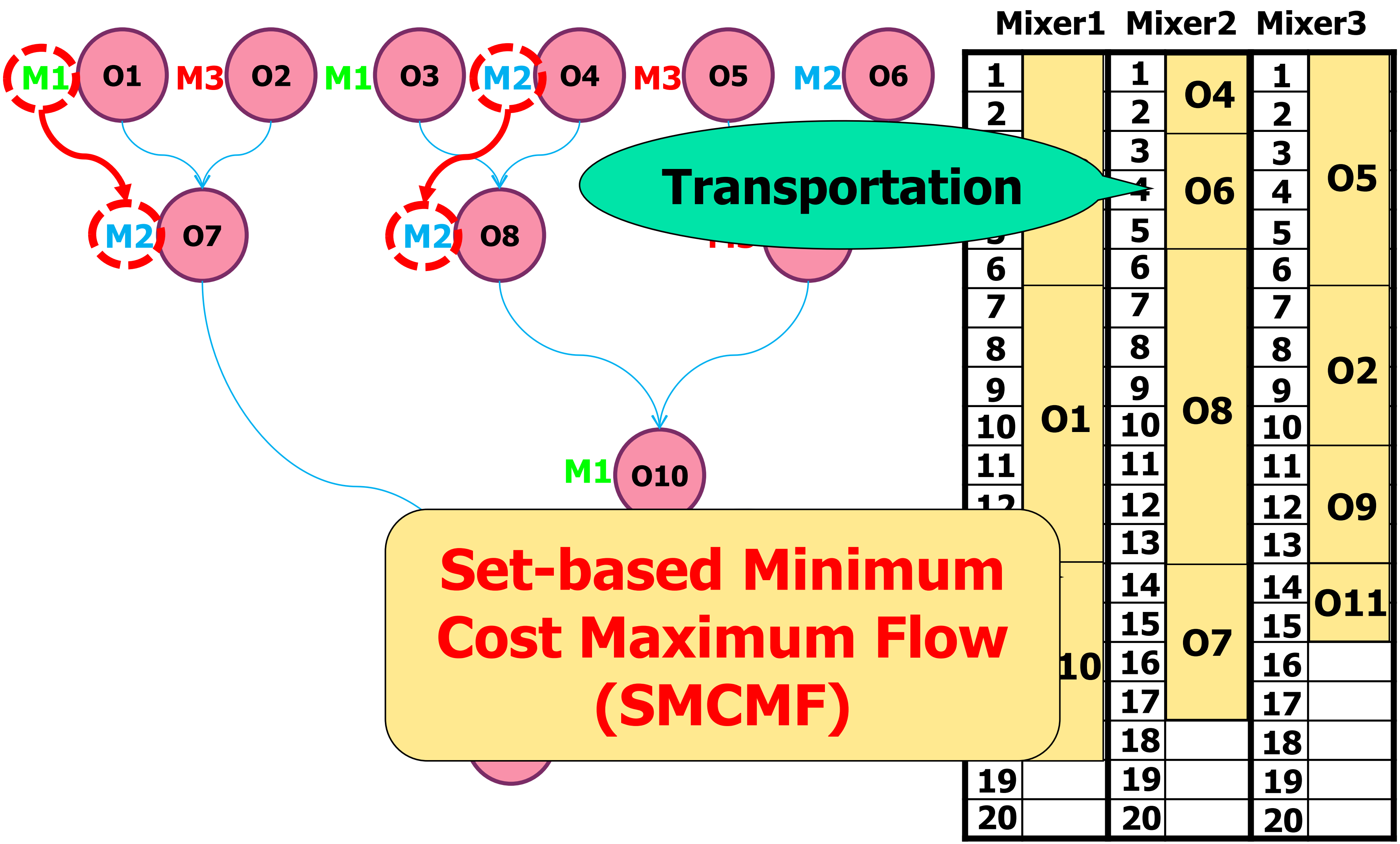
Binding by the urgency criteria



<table>
<tr><th></th><th>Mixer1</th><th></th><th>Mixer2</th><th></th><th>Mixer3</th></tr>
<tr><td>1</td><td rowspan="6">O3</td><td>1</td><td rowspan="2">O4</td><td>1</td><td rowspan="6">O5</td></tr>
<tr><td>2</td><td>2</td><td>2</td></tr>
<tr><td>3</td><td>3</td><td rowspan="3">O6</td><td>3</td></tr>
<tr><td>4</td><td>4</td><td>4</td></tr>
<tr><td>5</td><td>5</td><td>5</td></tr>
<tr><td>6</td><td>6</td><td rowspan="8">O8</td><td>6</td></tr>
<tr><td>7</td><td rowspan="7">O1</td><td>7</td><td>7</td><td rowspan="4">O2</td></tr>
<tr><td>8</td><td>8</td><td>8</td></tr>
<tr><td>9</td><td>9</td><td>9</td></tr>
<tr><td>10</td><td>10</td><td>10</td></tr>
<tr><td>11</td><td>11</td><td>11</td><td rowspan="3">O9</td></tr>
<tr><td>12</td><td>12</td><td>12</td></tr>
<tr><td>13</td><td>13</td><td>13</td></tr>
<tr><td>14</td><td rowspan="5">O10</td><td>14</td><td rowspan="4">O7</td><td>14</td><td rowspan="2">O11</td></tr>
<tr><td>15</td><td>15</td><td>15</td></tr>
<tr><td>16</td><td>16</td><td>16</td><td></td></tr>
<tr><td>17</td><td>17</td><td>17</td><td></td></tr>
<tr><td>18</td><td>18</td><td></td><td>18</td><td></td></tr>
<tr><td>19</td><td></td><td>19</td><td></td><td>19</td><td></td></tr>
<tr><td>20</td><td></td><td>20</td><td></td><td>20</td><td></td></tr>
</table>

Set-based Minimum Cost Maximum Flow

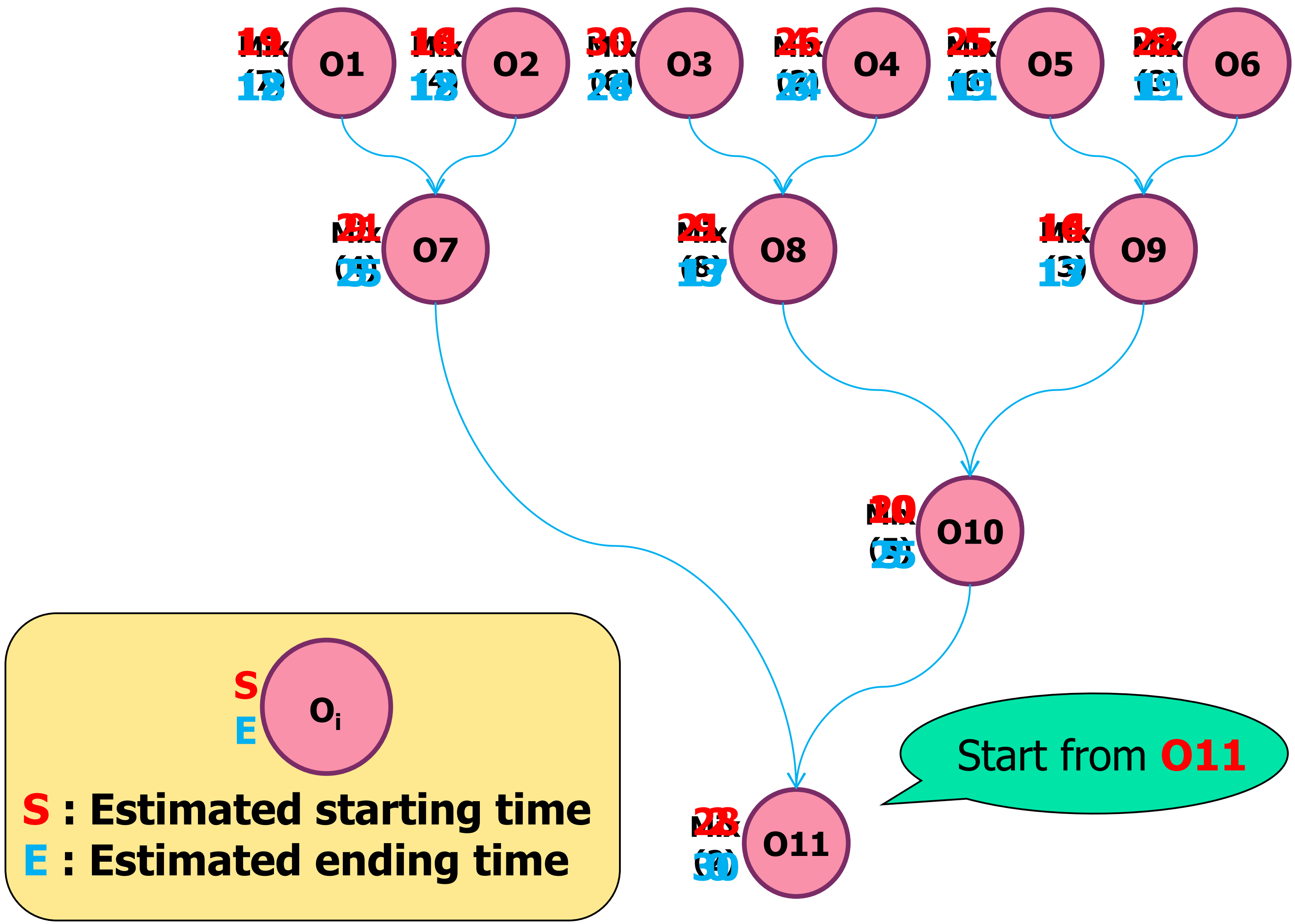
Transportation Issue



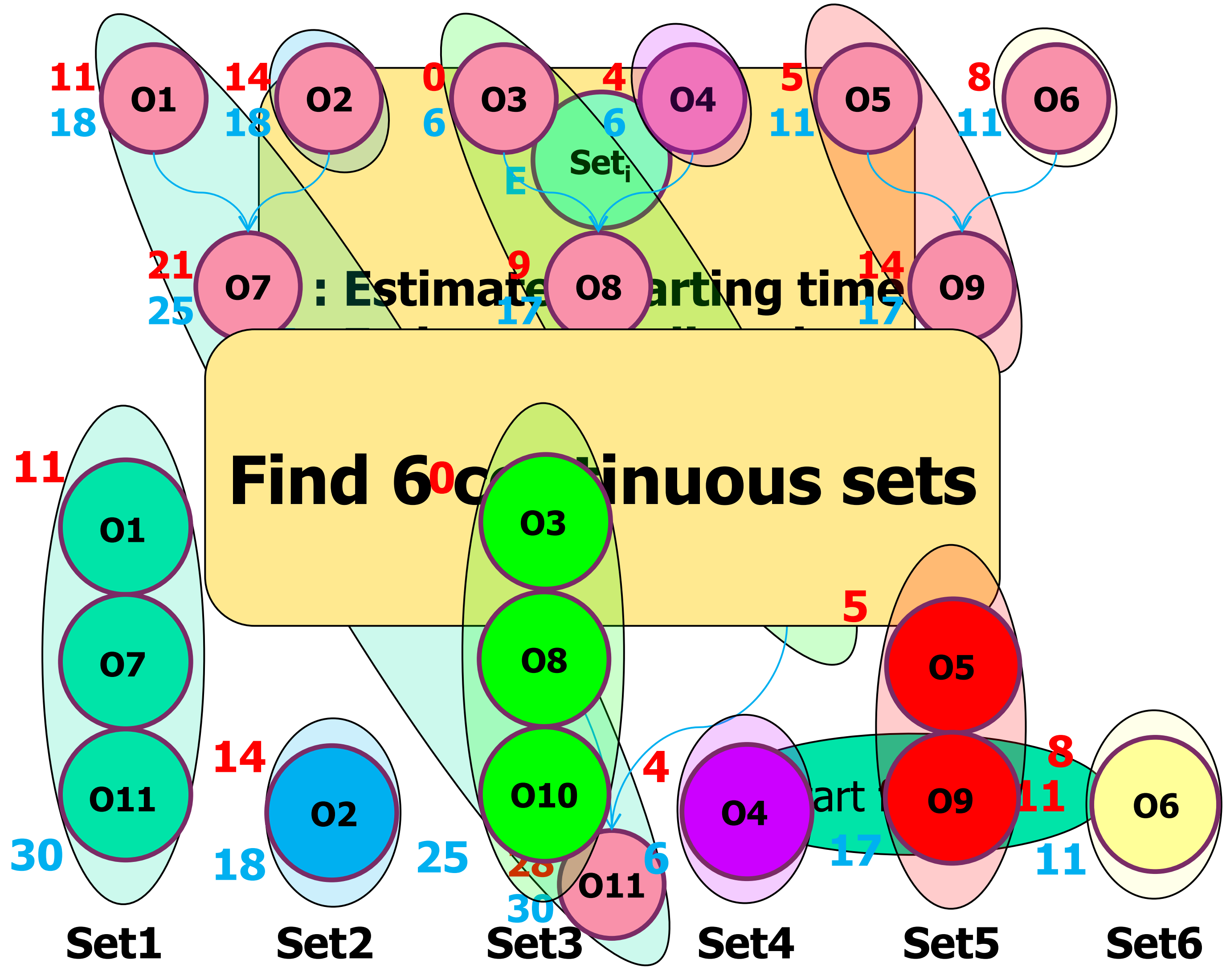
Set-based Minimum Cost Maximum Flow

- Set-based
  - Because binding continuous operations to the same components can reduce the total valve-switching amount, we first **group continuous operations in a set**
- Maximum Flow
  - In our SMCMF algorithm, each flow path represents a component. So, our goal here is to **maximize the component parallelization**
- Minimum Cost
  - We are interested to find a way that not only satisfies the parallelization but also **minimize the application complete time**

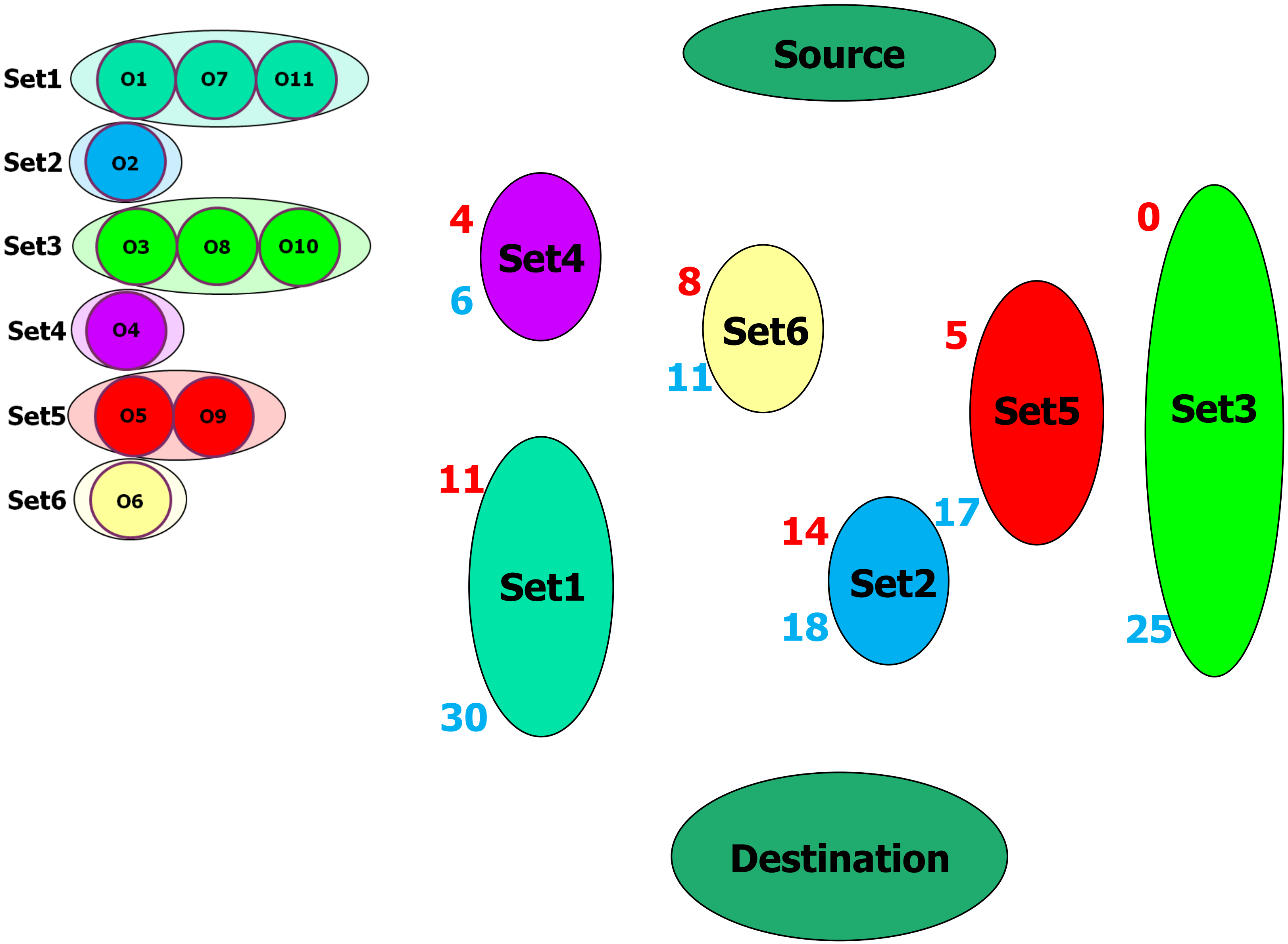
Application Graph
Topological Sort



Depth-first Search

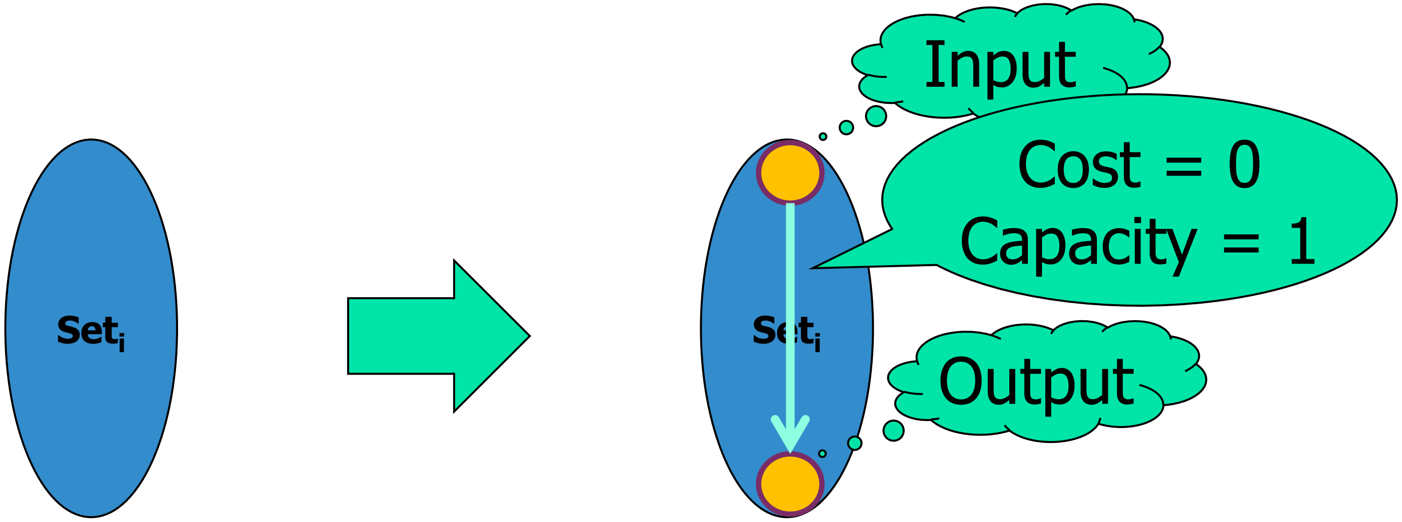


Build the Flow Network for the Sets



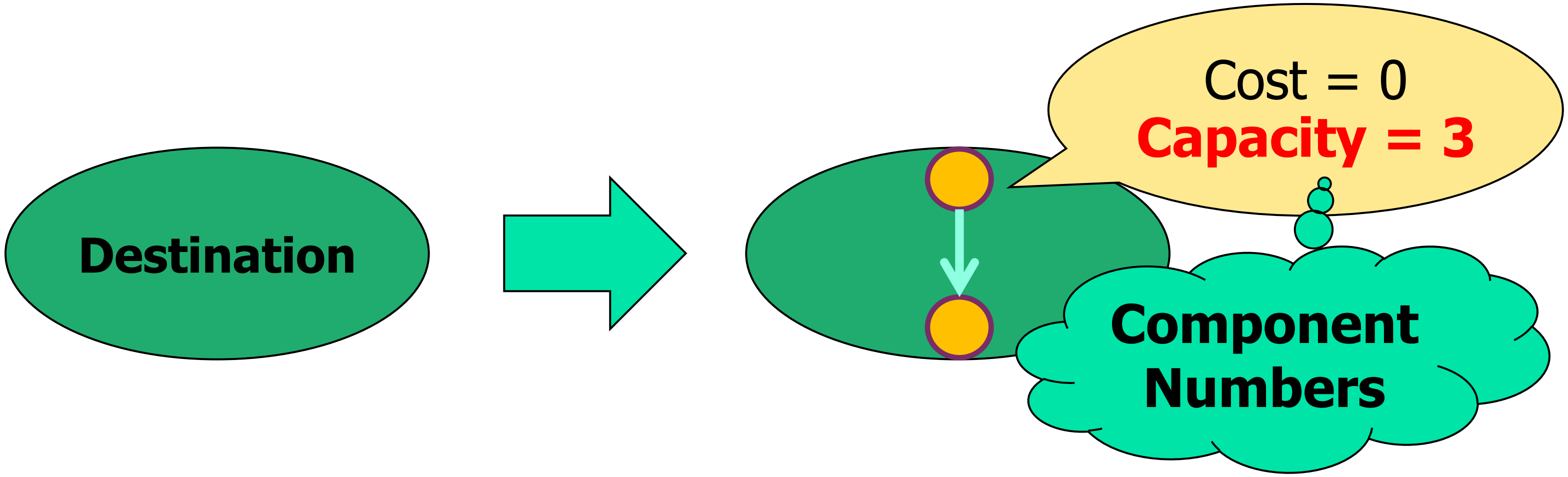
Separate Each Set to Two Nodes

Separate each set into two nodes, one for the input and the other for the output
Create an edge from input to output

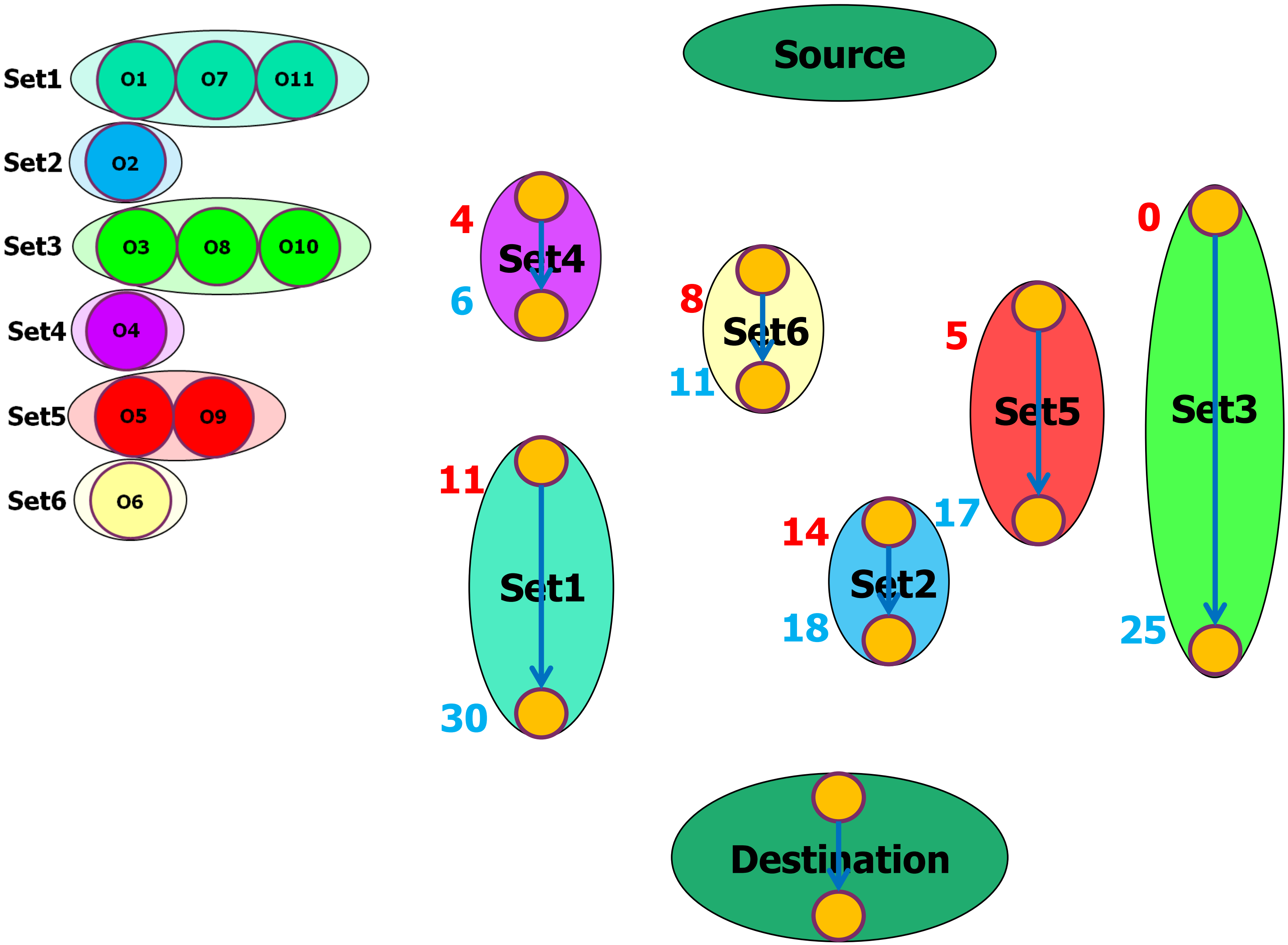


Separate the Destination Node

Separate the destination node into two node, one for the input and the other for the output
Cost = 0
Capacity = Given component number



Separate Destination and the Sets



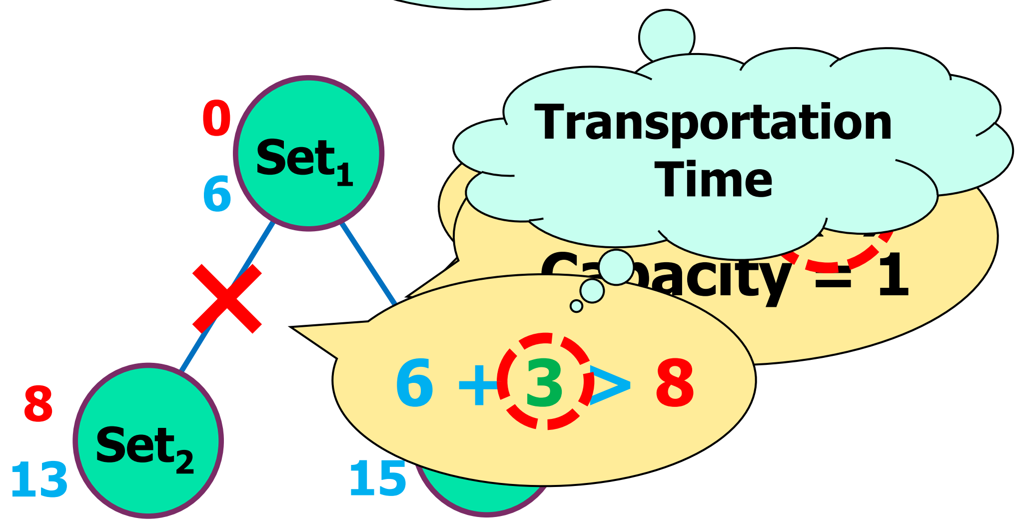
Edge Constraint

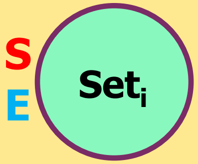
For every two sets Set

$Set_i.S$ + Transport

Create an edg

It is defined as the difference between $Set_2.E$ and $Set_3.S$

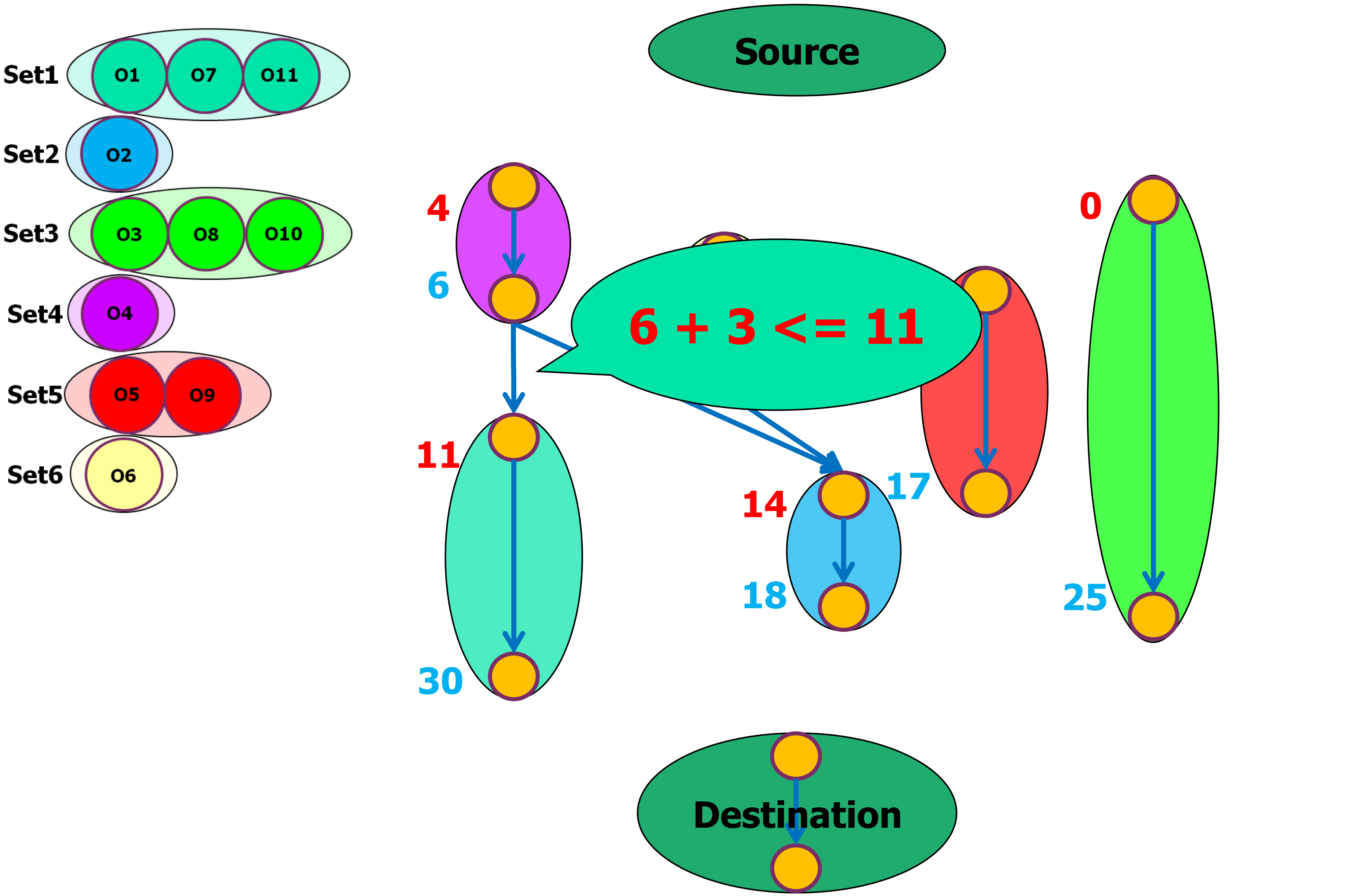




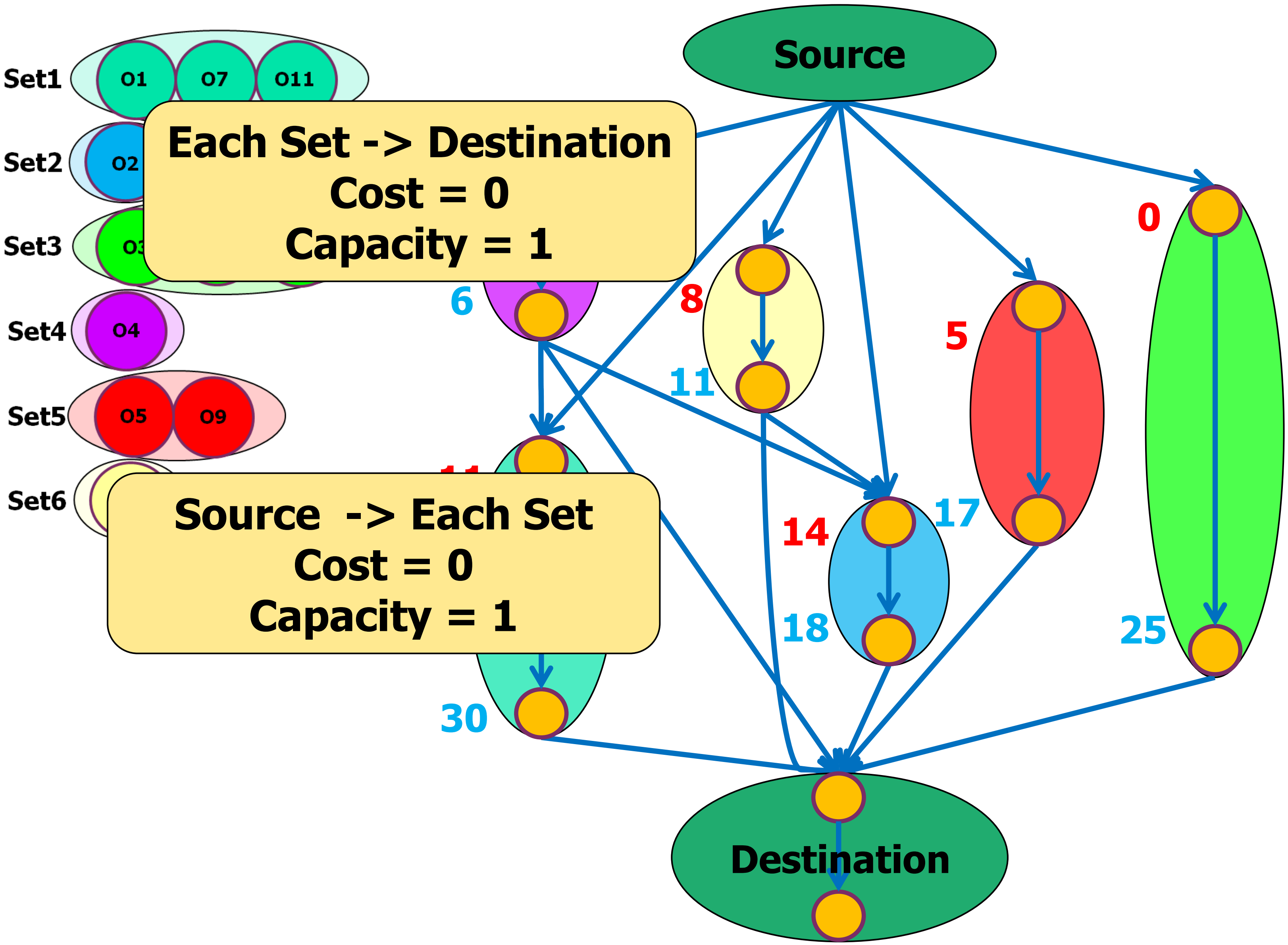
S : Estimated starting time

E : Estimated ending time

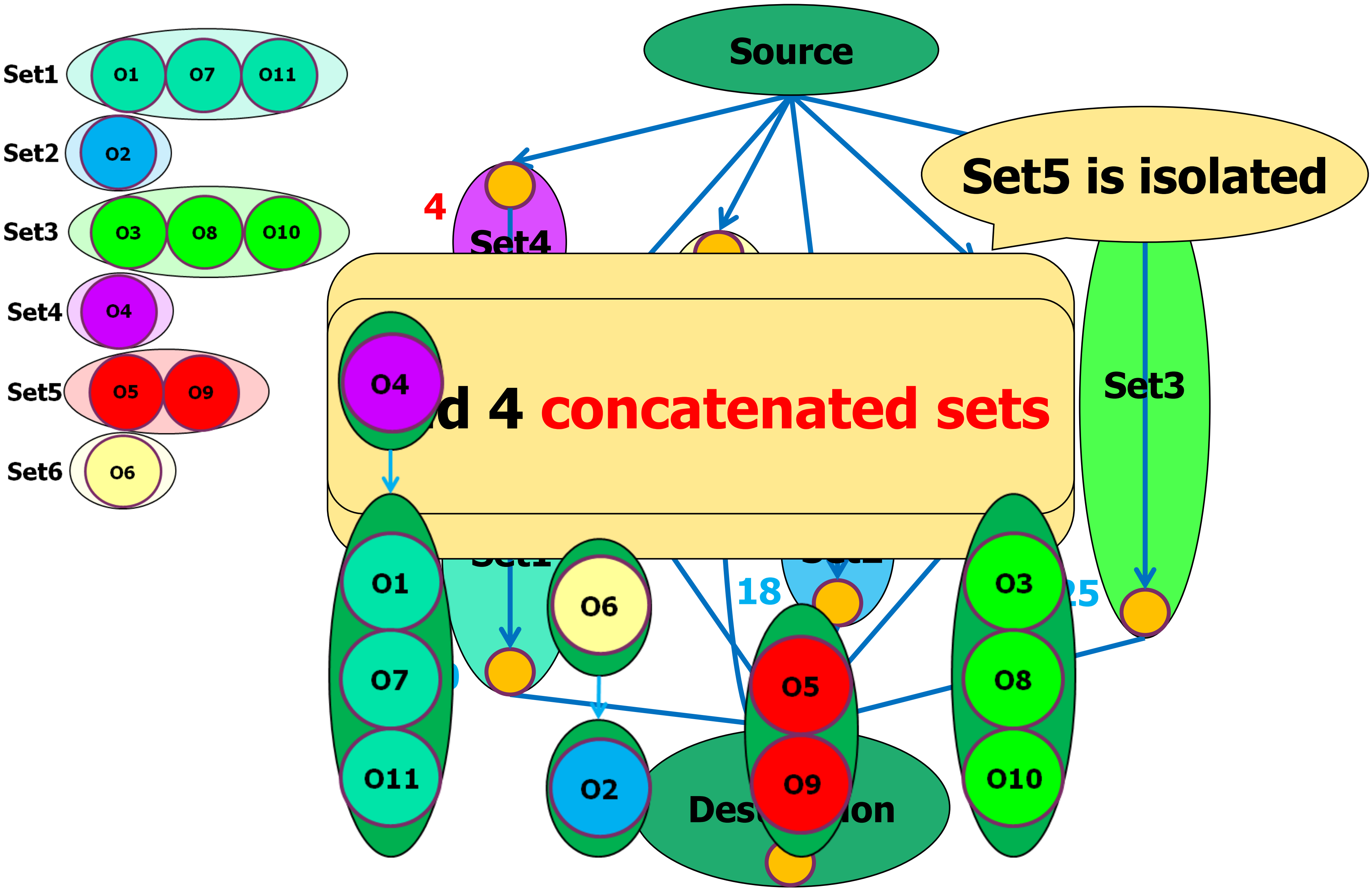
Build Edges for SMCMF



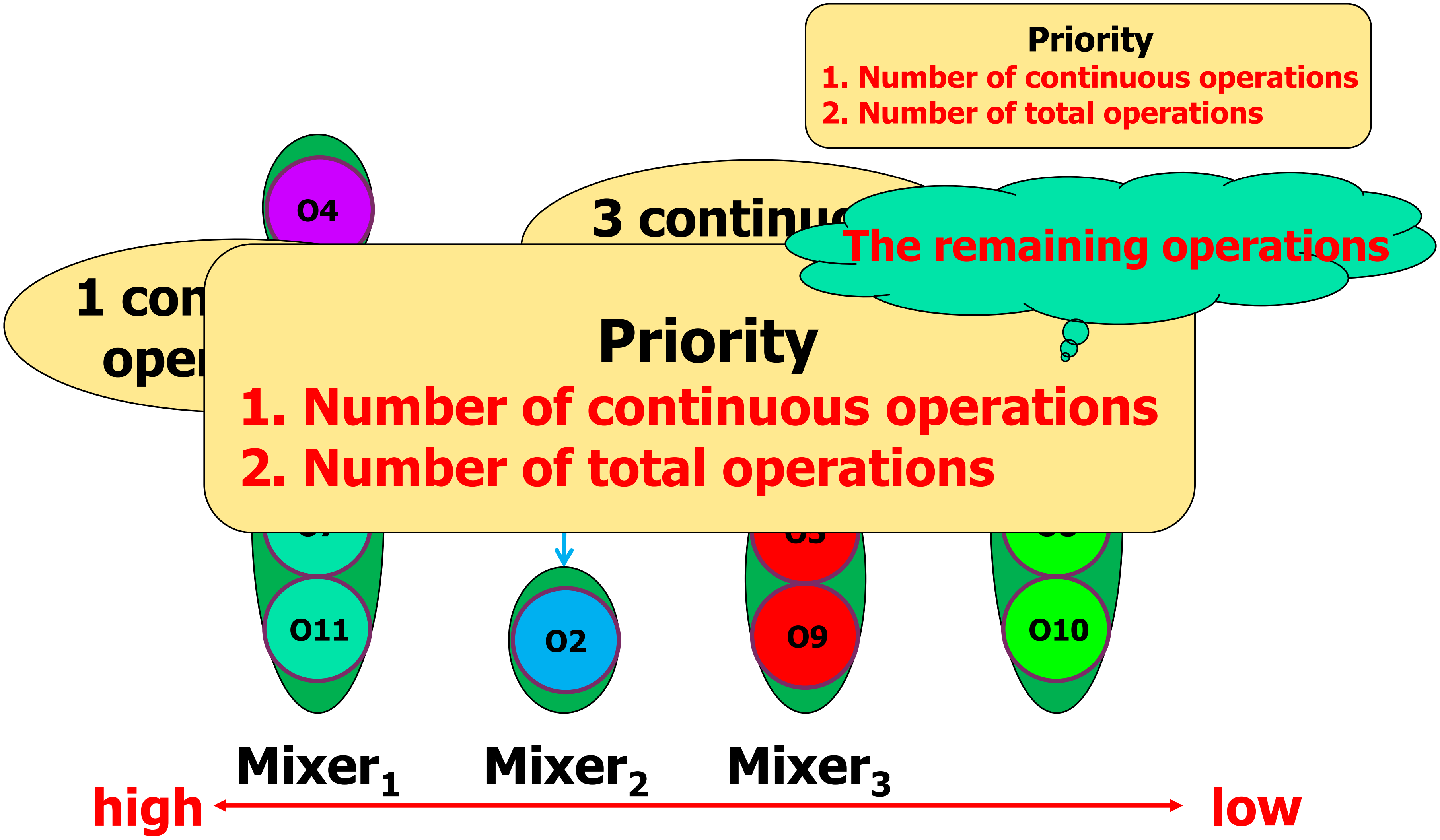
Build Edges for SMCMF



Minimum Cost Maximum Flow

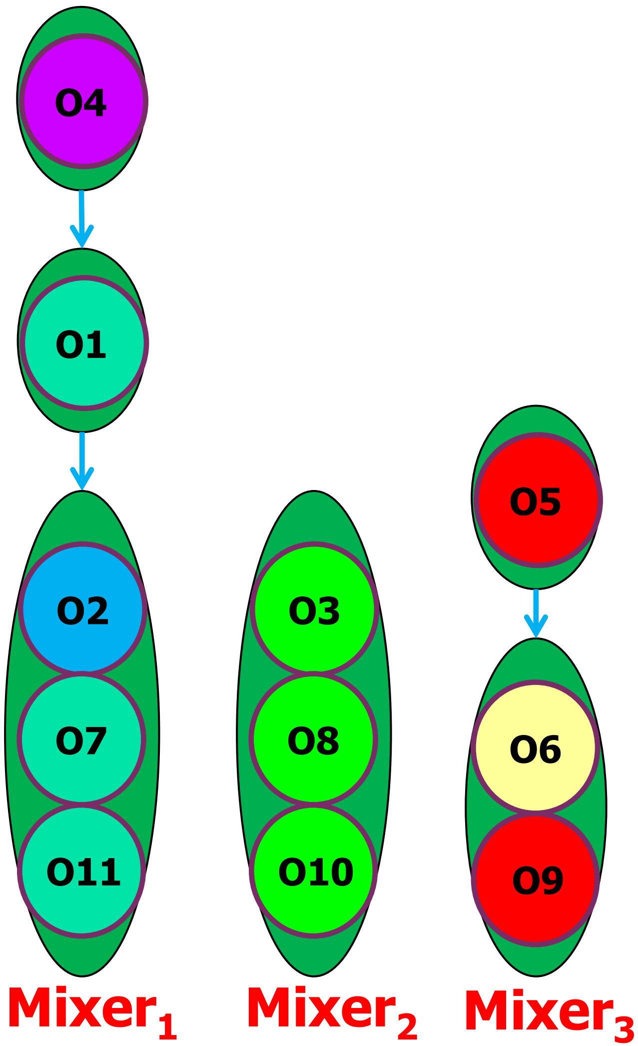


Sort by the Priority



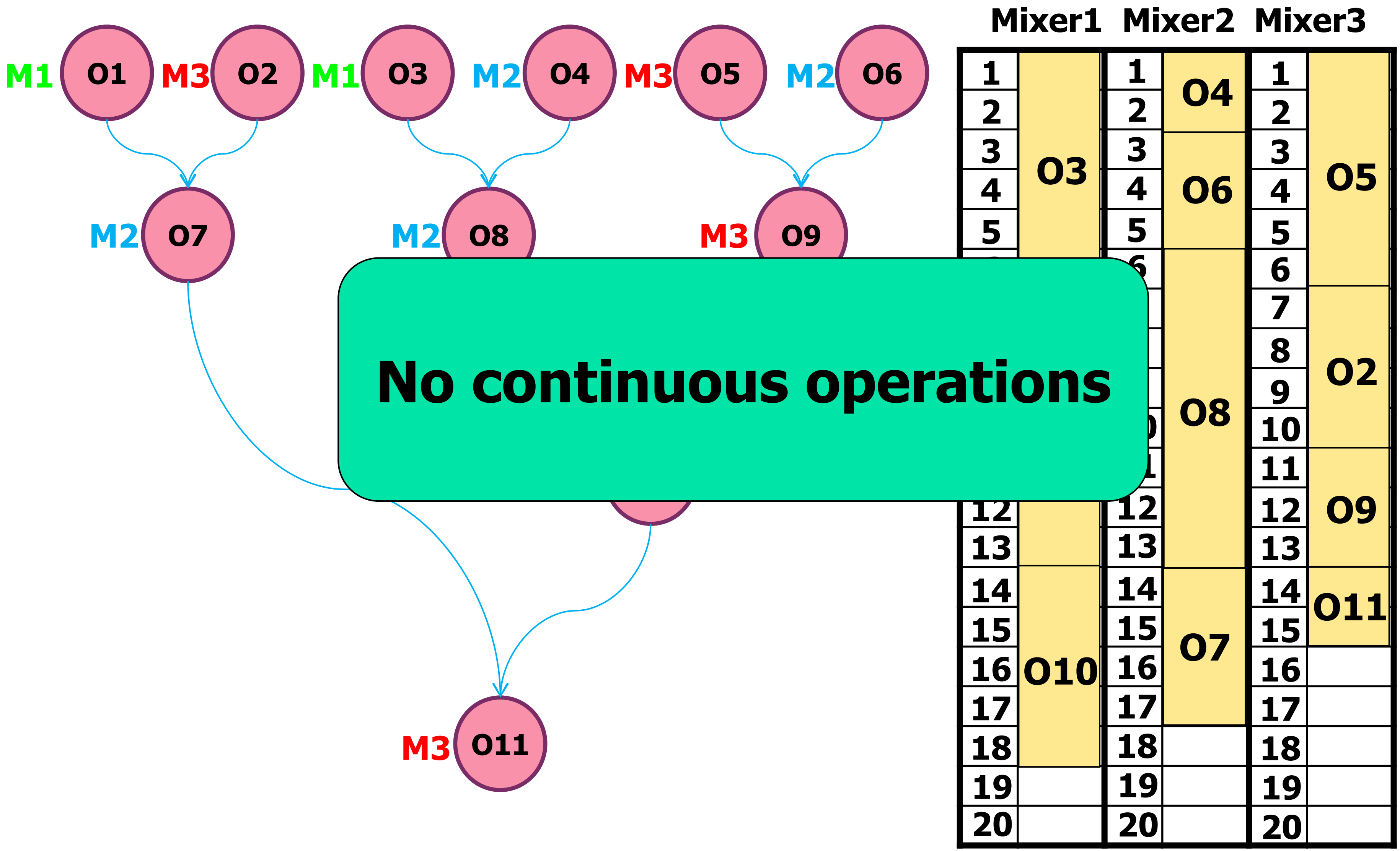
Insert the Remaining Operation

Resource Binding Result

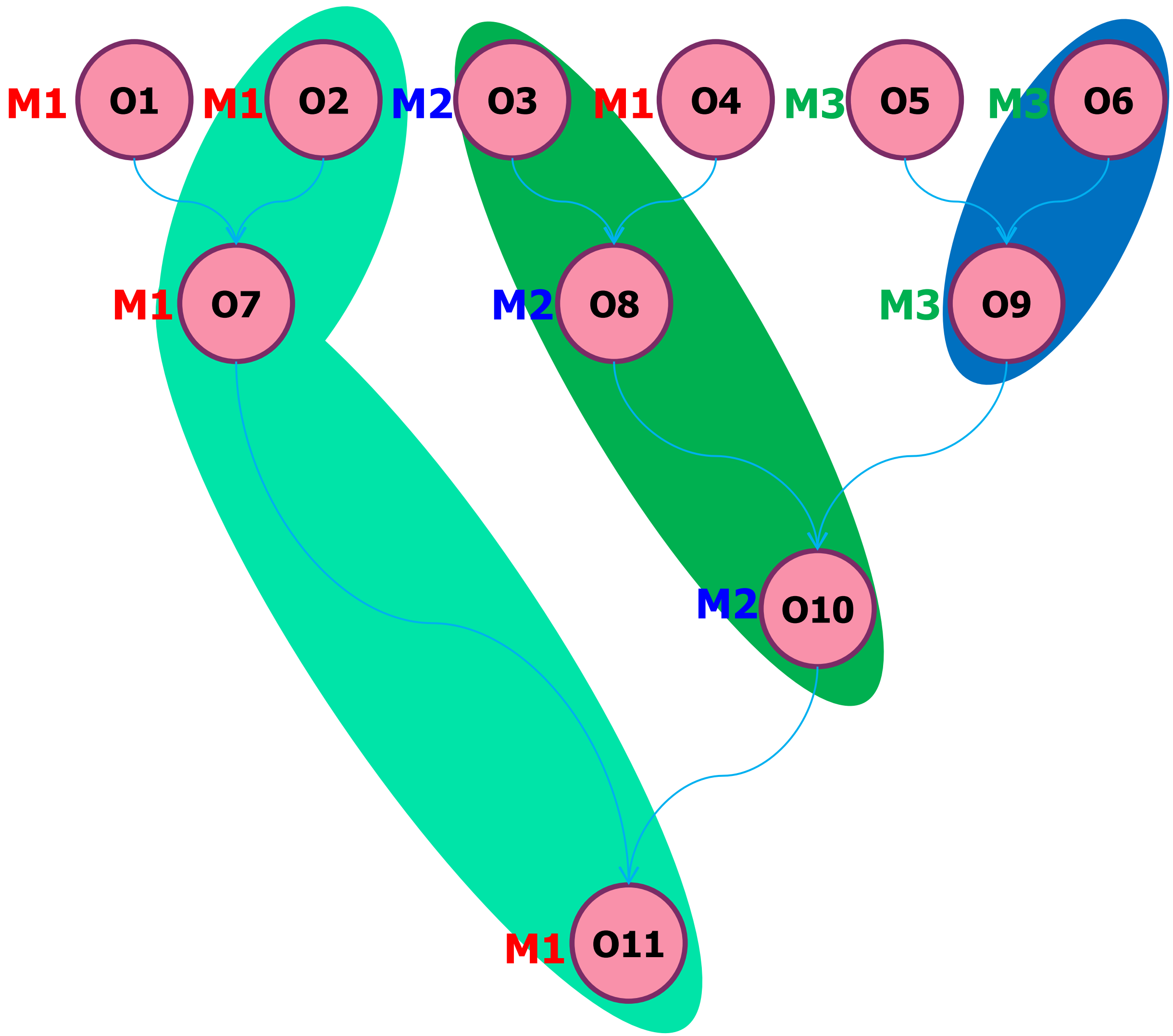


<table>
<tr><th></th><th>Mixer1</th><th></th><th>Mixer2</th><th></th><th>Mixer3</th></tr>
<tr><td>1</td><td rowspan="2">O4</td><td>1</td><td rowspan="6">O3</td><td>1</td><td rowspan="6">O5</td></tr>
<tr><td>2</td><td>2</td><td>2</td></tr>
<tr><td>3</td><td rowspan="7">O1</td><td>3</td><td>3</td></tr>
<tr><td>4</td><td>4</td><td>4</td></tr>
<tr><td>5</td><td>5</td><td>5</td></tr>
<tr><td>6</td><td>6</td><td>6</td></tr>
<tr><td>7</td><td>7</td><td rowspan="8">O8</td><td>7</td><td rowspan="3">O6</td></tr>
<tr><td>8</td><td>8</td><td>8</td></tr>
<tr><td>9</td><td>9</td><td>9</td></tr>
<tr><td>10</td><td rowspan="4">O2</td><td>10</td><td>10</td><td rowspan="3">O9</td></tr>
<tr><td>11</td><td>11</td><td>11</td></tr>
<tr><td>12</td><td>12</td><td>12</td></tr>
<tr><td>13</td><td>13</td><td>13</td><td></td></tr>
<tr><td>14</td><td rowspan="4">O7</td><td>14</td><td>14</td><td></td></tr>
<tr><td>15</td><td>15</td><td rowspan="5">O10</td><td>15</td><td></td></tr>
<tr><td>16</td><td>16</td><td>16</td><td></td></tr>
<tr><td>17</td><td>17</td><td>17</td><td></td></tr>
<tr><td>18</td><td rowspan="2">O11</td><td>18</td><td>18</td><td></td></tr>
<tr><td>19</td><td>19</td><td>19</td><td></td></tr>
<tr><td>20</td><td></td><td>20</td><td></td><td>20</td><td></td></tr>
</table>

Binding by Baseline Method



Binding by SMCMF



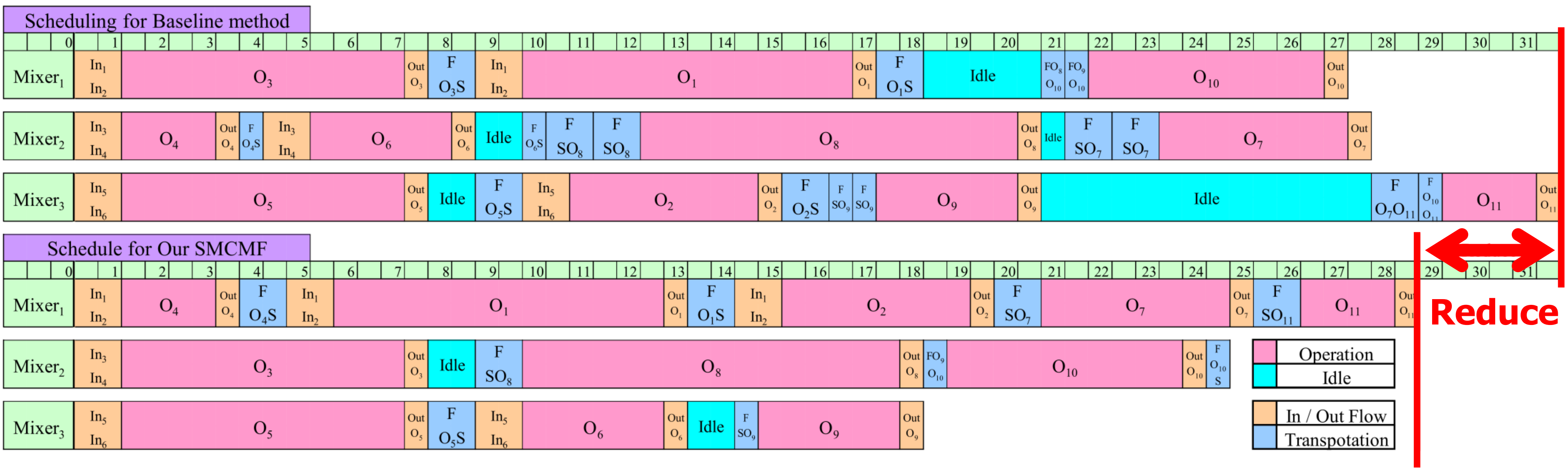
	Mixer1		Mixer2		Mixer3
1	O4	1	O3	1	O5
2		2		2	
3	O1	3		3	
4		4		4	
5		5		5	
6		6		6	
7		7	O8	7	O6
8		8		8	
9		9		9	
10	O2	10		10	O9
11		11		11	
12		12		12	
13		13		13	
14	O7	14		14	
15		15	O10	15	
16		16		16	
17		17		17	
18	O11	18		18	
19		19		19	
20		20		20	

Construction of Biochip Architecture

- Relation-based placement strategy
  - Place the highly related components much closer to reduce the total length and the intersection number of the flow-channels

- Routing by Dijkstra Shortest Path algorithm
  - Make a trade-off between intersection numbers and the total length of the flow-channels

Scheduling



Baseline

Valve-switching : **336** (116/220)

Application complete time : 31.5 (s)

SMCMF

Valve-switching : **228** (68/160)

Application complete time : 28.5 (s)

Outline

- Introduction
- Preliminary
- Problem Formulation
- Algorithms
- Experimental Results
- Conclusion

Experimental Settings

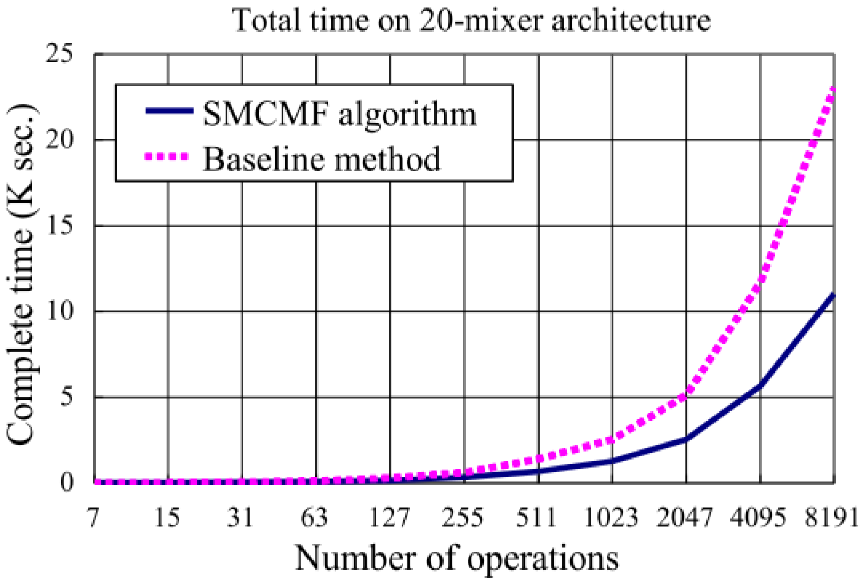
- Implement our algorithm in C++ language on a PC with Core2 Quad processors at 2.66GHz and 3.25GB of RAM

- Compare set-based minimum cost maximum flow binding algorithm with list scheduling based baseline method

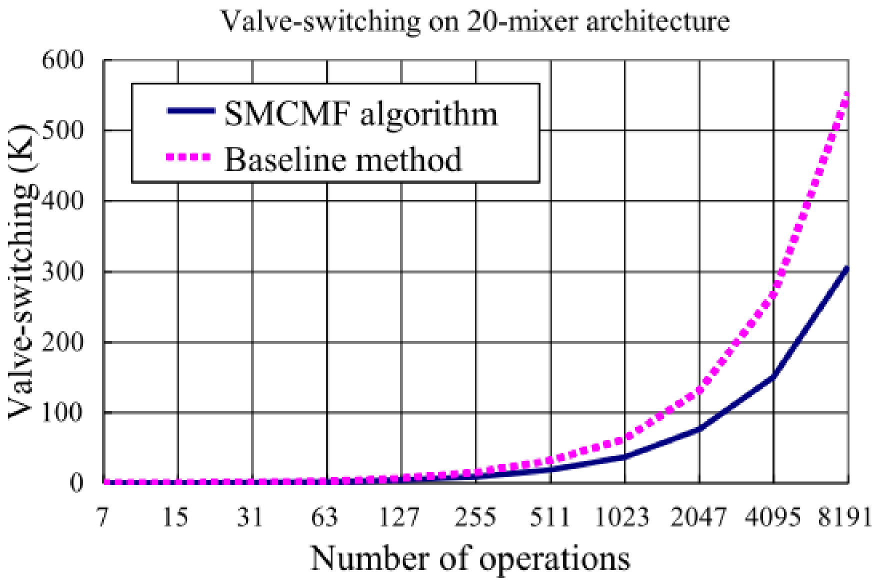
- Test on several synthetic benchmarks
  - Adjust operation numbers from 7 to 8191 and fix resource constraint as 20
  - Adjust resource constraint from 10 ~ 100 and fix operation numbers as 1023

Experimental Result

Operation : **7~8191**
Mixer : 20



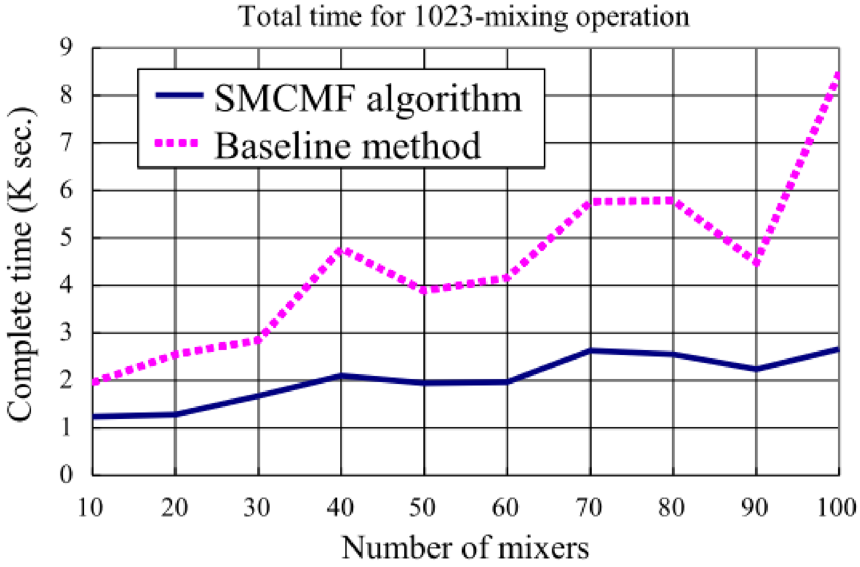
Application Complete Time



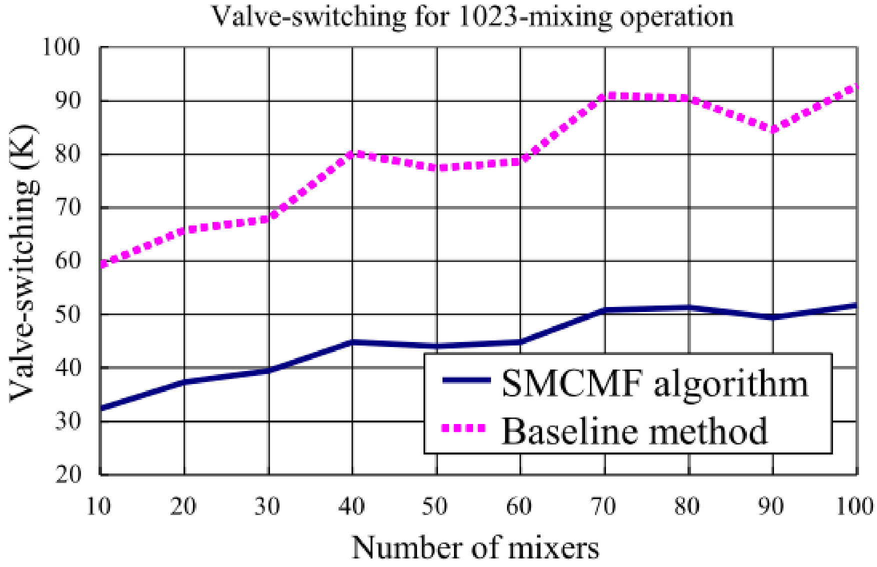
Valve-switching Amount

Experimental Result

Operation : 1023
Mixer : **10 ~ 100**



Application Complete Time



Valve-switching Amount

Outline

- Introduction
- Preliminary
- Problem Formulation
- Algorithms
- Experimental Results
- Conclusion

Conclusion

- The valve-switching activities for the components such as mixer and storage are modeled
- A set-based minimum cost maximum flow (SMCMF) binding algorithm is proposed
- The experimental results shows that set-based minimum cost maximum flow binding algorithm not only minimizes the valve-switching amount but also reduces the application complete time

Thank You for Your Attention!