

Discrete Optimization

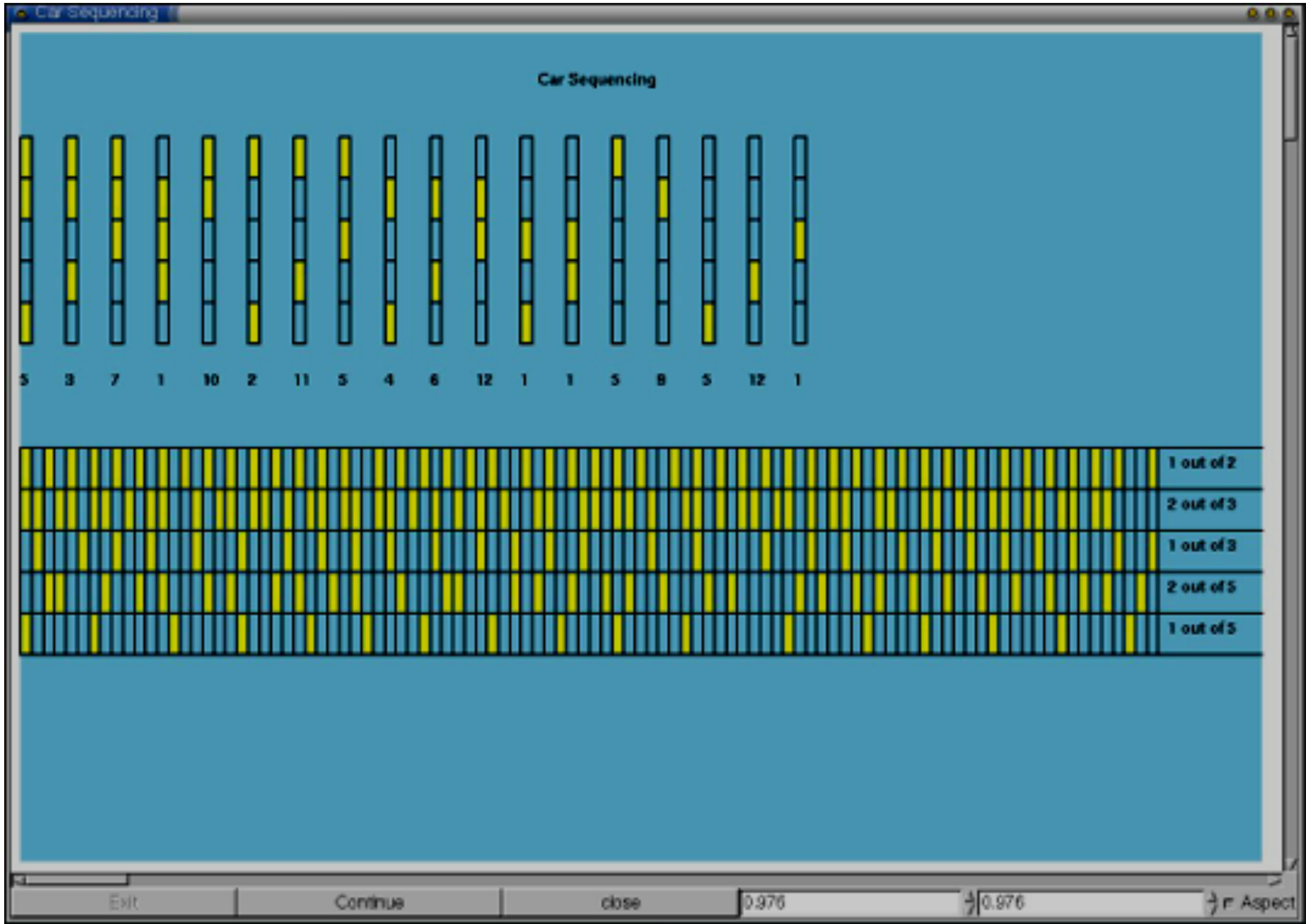
Constraint Programming: Part VII

Car Sequencing

- ▶ Cars on an assembly line
- ▶ Cars require specific options
 - leather seats, moonroof
- ▶ Capacity constraints on the production units
 - at most 2 out of 5 successive cars can require a moonroof
- ▶ Sequence all the cars such that the capacity constraints are satisfied



Car Sequencing



Car Sequencing

Options	1	2	3	4	5	Demand
Class 1	yes		yes	yes		1
Class 2				yes		1
Class 3		yes			yes	2
Class 4		yes		yes		2
Class 5	yes		yes			2
Class 6	yes	yes				2
Capacity	1/2	2/3	1/3	2/5	1/5	

Car Sequencing

```
range Slots = ...;
range Configs = ...;
range Options = ...;
int demand[Configs] = ...;
int nbCars = sum(c in Configs) demand[c];
int lb[Options] = ...;
int ub[Options] = ...;
int requires[Options,Config] = ...;
var{int} line[Slots] in Configs;
var{int} setup[Options,Slots] in 0..1;

solve {
    forall(c in Configs)
        sum(s in Slots) (line[s] = c) = demand[c];

    forall(s in Slots,o in Options)
        setup[o,s] = requires[o,line[s]];

    forall(o in Options, s in 1..nbCars-ub[o]+1)
        sum(j in s..s+ub[o]-1) setup[o,s] <= lb[o];
}
```

line[s] denotes the
type of car sequenced
on slot s

setup[o,s]=1
if slot[s] has a car
requiring option o

demand
constraints

defines the setup
variables

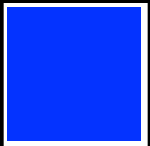

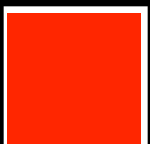
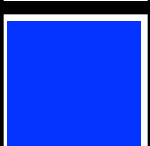


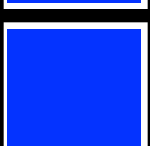
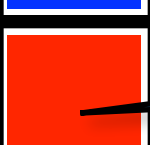
capacity
constraints

Car Sequencing

Slots	1	2	3	4	5	6	7	8	9	10	Demand
Class 1											1
Class 2											1
Class 3											2
Class 4											2
Class 5											2
Class 6											2

Car Sequencing

Options	1	2	3	4	5	Demand
Class 1	yes		yes	yes		1
Class 2				yes		1
Class 3		yes			yes	2
Class 4		yes		yes		2
Class 5	yes		yes			2
Class 6	yes	yes				2
Capacity	1/2	2/3	1/3	2/5	1/5	

Setup	1	2	3	4	5	6	7	8	9	10	Capacity
Option 1											1/2
Option 2											2/3
Option 3											1/3
Option 4											2/5
Option 5											1/5

capacity constraint

element constraint

Car Sequencing

```
range Slots = ...;
range Configs = ...;
range Options = ...;
int demand[Configs] = ...;
int lb[Options] = ...;
int ub[Options] = ...;
int requires[Options,Config] = ...;
var{int} line[Slots] in Configs;
var{int} setup[Options,Slots] in 0..1;

solve {
    forall(c in Configs)
        sum(s in Slots) (line[s] = c) = demand[c];

    forall(s in Slots,o in Options)
        [setup[o,s] = requires[o,line[s]]];

    forall(o in Options, s in 1..nbCars-ub[o]+1)
        [sum(j in s..s+ub[o]-1) setup[o,j] <= lb[o]];
}
```


Car Sequencing

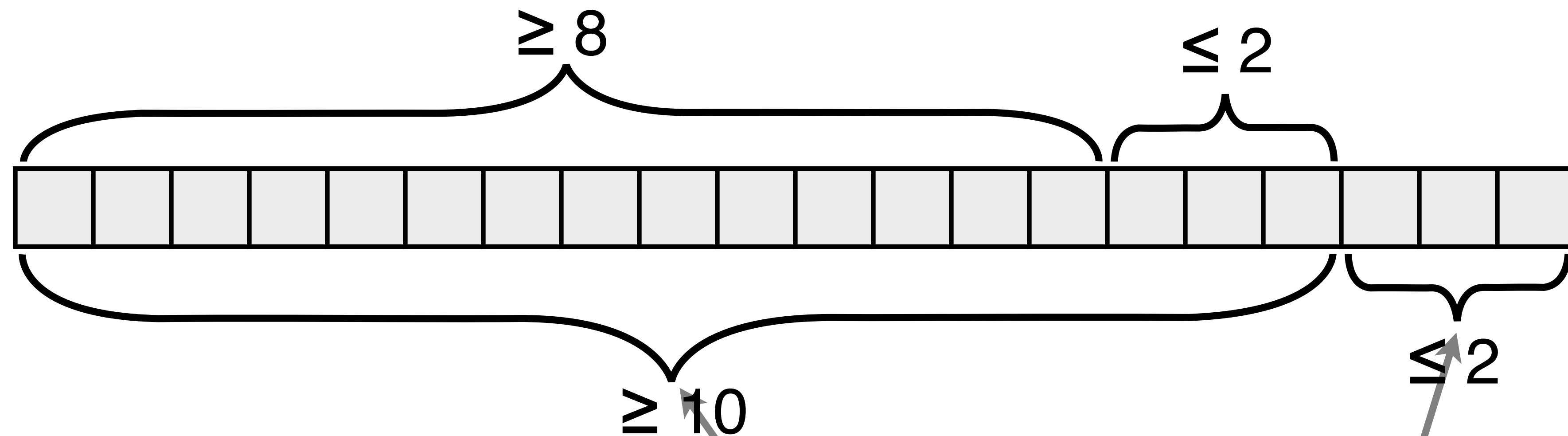
Setup	1	2	3	4	5	6	7	8	9	10	Capacity
Option 1	<div></div>	<div></div>									1/2
Option 2	<div></div>										2/3
Option 3	<div></div>	<div></div>	<div></div>								1/3
Option 4	<div></div>										2/5
Option 5	<div></div>										1/5

Options	1	2	3	4	5	Demand
Class 1	yes		yes	yes		1
Class 2				yes		1
Class 3		yes			yes	2
Class 4		yes		yes		2
Class 5	yes		yes			2
Class 6	yes	yes				2
Capacity	1/2	2/3	1/3	2/5	1/5	

Slots	1	2	3	4	5	6	7	8	9	10	Demand
Class 1	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	<div></div>	1
Class 2	<div></div>										1
Class 3	<div></div>										2
Class 4	<div></div>										2
Class 5	<div></div>	<div></div>	<div></div>								2
Class 6	<div></div>	<div></div>									2

Car Sequencing: Redundant Constraints

- Consider an option o with
 - Capacity: 2 out of 3
 - Demand: 12 cars



- How many cars with option o can be in the last 3 slots?
- How many cars with option o must be in these slots?

Car Sequencing: Redundant Constraints

```
range Slots = ...;
range Configs = ...;
range Options = ...;
int demand[Configs] = ...;
int lb[Options] = ...;
int ub[Options] = ...;
int requires[Options,Config] = ...;
var{int} line[Cars] in Configs;
var{int} setup[Options,Slots] in 0..1;

solve {
    forall(c in Configs)
        sum(s in Slots) (line[s] = c) = demand[c];
    forall(s in Slots,o in Options)
        setup[o,s] = requires[o,line[s]];
    forall(o in Options, s in 1..nbCars-ub[o]+1)
        sum(j in s..s+ub[o]-1) setup[o,s] <= lb[o];

    forall(o in Options, i in 1..demand[o])
        sum(s in 1..nbCars-i*ub[o]) setup[o,s] >= demand[o] - i*lb[o];
}
```

Car Sequencing

Setup	1	2	3	4	5	6	7	8	9	10	Capacity
Option 1											1/2
Option 2											2/3
Option 3											1/3
Option 4											2/5
Option 5											1/5

Options	1	2	3	4	5	Demand
Class 1	yes		yes	yes		1
Class 2				yes		1
Class 3		yes			yes	2
Class 4		yes		yes		2
Class 5	yes		yes			2
Class 6	yes	yes				2
Capacity	1/2	2/3	1/3	2/5	1/5	

Slots	1	2	3	4	5	6	7	8	9	10	Demand
Class 1											1
Class 2											1
Class 3											2
Class 4											2
Class 5											2
Class 6											2

Car Sequencing

Slots	1	2	3	4	5	6	7	8	9	10	Demand
Class 1											1
Class 2											1
Class 3											2
Class 4											2
Class 5											2
Class 6											2

Options	1	2	3	4	5	Demand
Class 1	yes		yes			1
Class 2						1
Class 3		yes			yes	2
Class 4		yes				2
Class 5	yes		yes			2
Class 6	yes					2
Capacity	1/2	2/3	1/3	2/5	1/5	

Setup	1	2	3	4	5	6	7	8	9	10	Capacity
Option 1											1/2
Option 2											2/3
Option 3											1/3
Option 4											2/5
Option 5											1/5

Car Sequencing

Slots	1	2	3	4	5	6	7	8	9	10	Demand
Class 1											1
Class 2											1
Class 3											2
Class 4											2
Class 5											2
Class 6											2

Setup	1	2	3	4	5	6	7	8	9	10	Capacity
Option 1											1/2
Option 2											2/3
Option 3											1/3
Option 4											2/5
Option 5											1/5

Options	1	2	3	4	5	Demand
Class 1	yes		yes	yes		1
Class 2				yes		1
Class 3		yes			yes	2
Class 4		yes		yes		2
Class 5	yes		yes			2
Class 6	yes	yes				2
Capacity	1/2	2/3	1/3	2/5	1/5	

Car Sequencing

Slots	1	2	3	4	5	6	7	8	9	10	Demand
Class 1	■	■	■	■	■	■	■	■	■	■	1
Class 2	■	■	■	■	■	■	■	■	■	■	1
Class 3	■	■			■			■			2
Class 4	■	■	■	■	■			■			2
Class 5	■	■	■	■	■	■	■	■	■	■	2
Class 6	■	■			■			■			2

Setup	1	2	3	4	5	6	7	8	9	10	Capacity
Option 1	■	■			■			■			1/2
Option 2	■	■	■	■	■	■	■	■	■	■	2/3
Option 3	■	■	■		■			■			1/3
Option 4	■	■	■	■	■			■			2/5
Option 5	■	■			■			■			1/5

Options	1	2	3	4	5	Demand
Class 1	yes		yes	yes		1
Class 2				yes		1
Class 3		yes			yes	2
Class 4		yes		yes		2
Class 5	yes		yes			2
Class 6	yes	yes				2
Capacity	1/2	2/3	1/3	2/5	1/5	

Car Sequencing

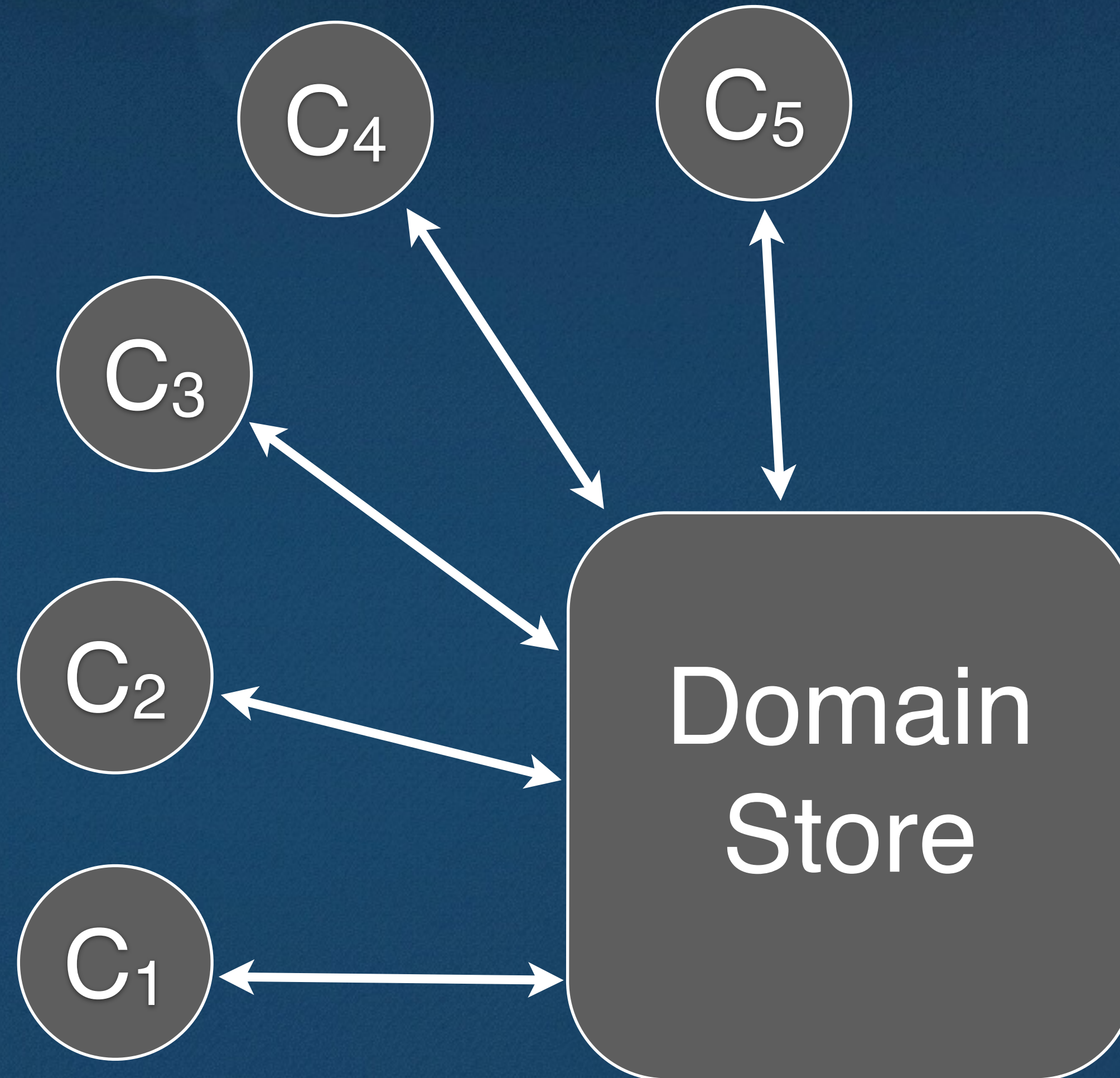
Slots	1	2	3	4	5	6	7	8	9	10	Demand
Class 1	█	█	█	█	█	█	█	█	█	█	1
Class 2	█	█	█	█	█	█	█	█	█	█	1
Class 3	█	█			█			█			2
Class 4	█	█	█	█	█			█			2
Class 5	█	█	█	█	█	█	█	█	█	█	2
Class 6	█	█			█			█			2

Options	1	2	3	4	5	Demand
Class 1	yes		yes	yes		1
Class 2				yes		1
Class 3		yes			yes	2
Class 4		yes		yes		2
Class 5	yes		yes			2
Class 6	yes	yes				2
Capacity	1/2	2/3	1/3	2/5	1/5	

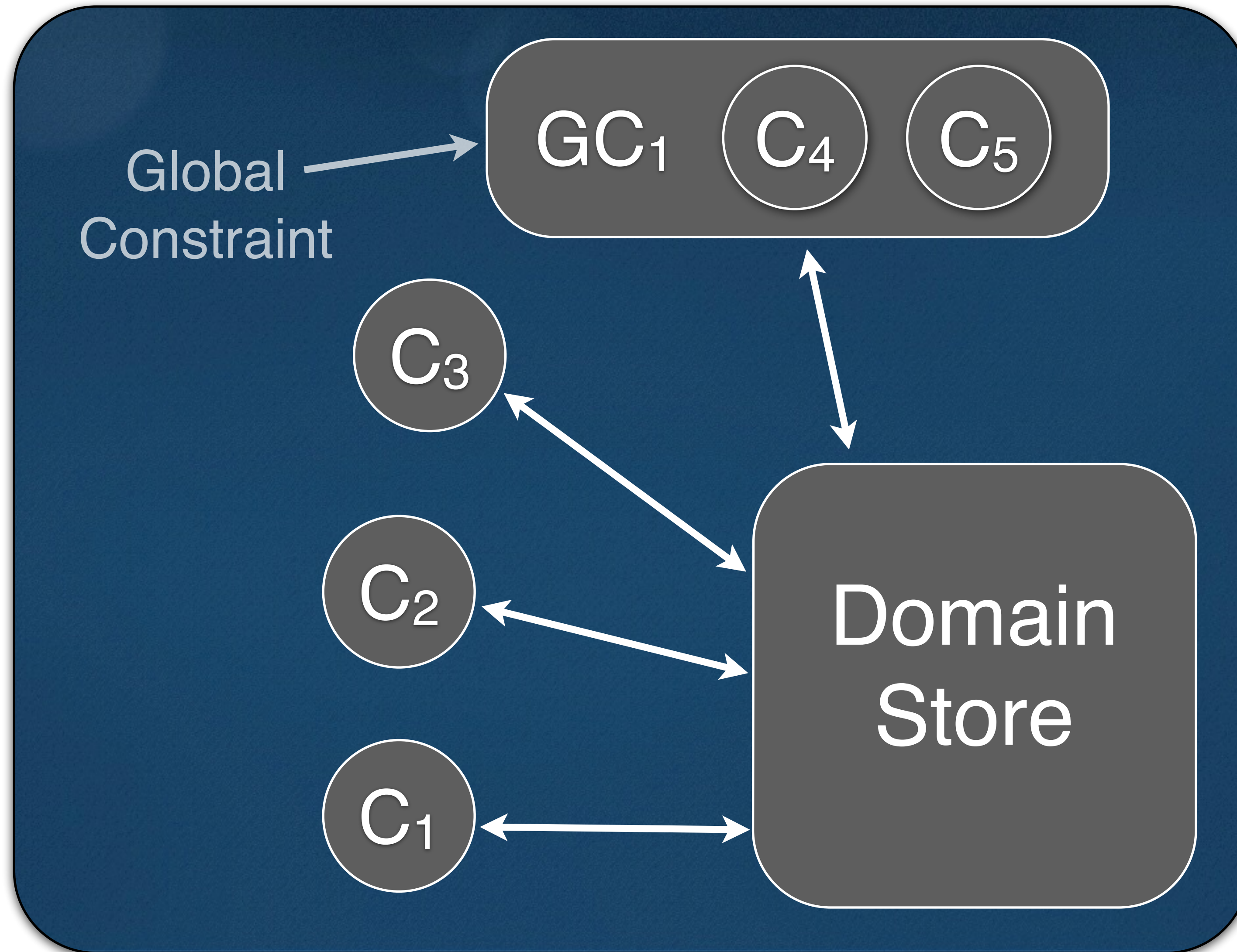
Setup	1	2	3	4	5	6	7	8	9	10	Capacity
Option 1	█	█		█	█	█	█	█	█		1/2
Option 2	█	█	█	█	█	█	█	█	█	█	2/3
Option 3	█	█	█		█			█			1/3
Option 4	█	█	█	█	█			█			2/5
Option 5	█	█			█			█			1/5

Improving Communication

Constraint Store

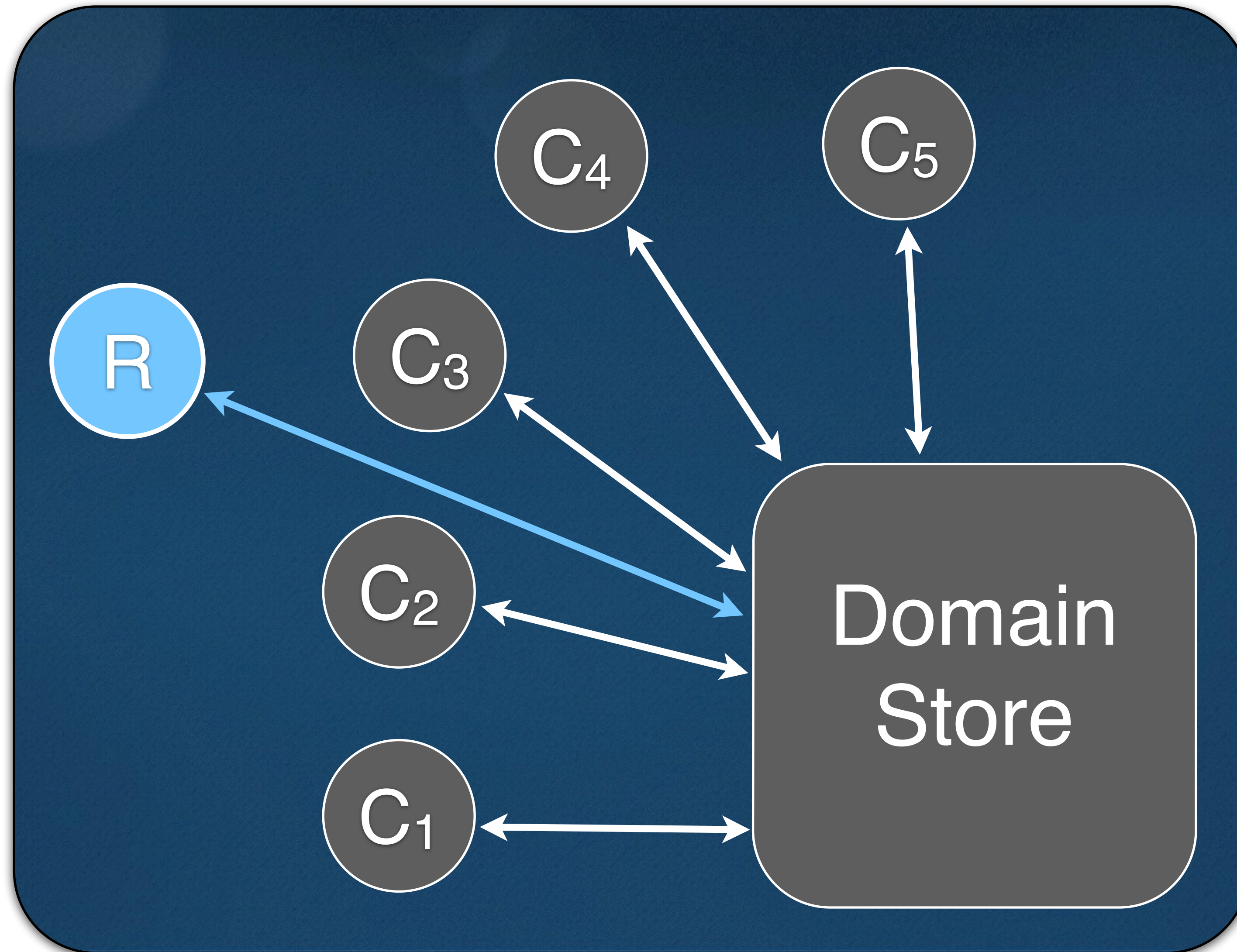


Improving Communication



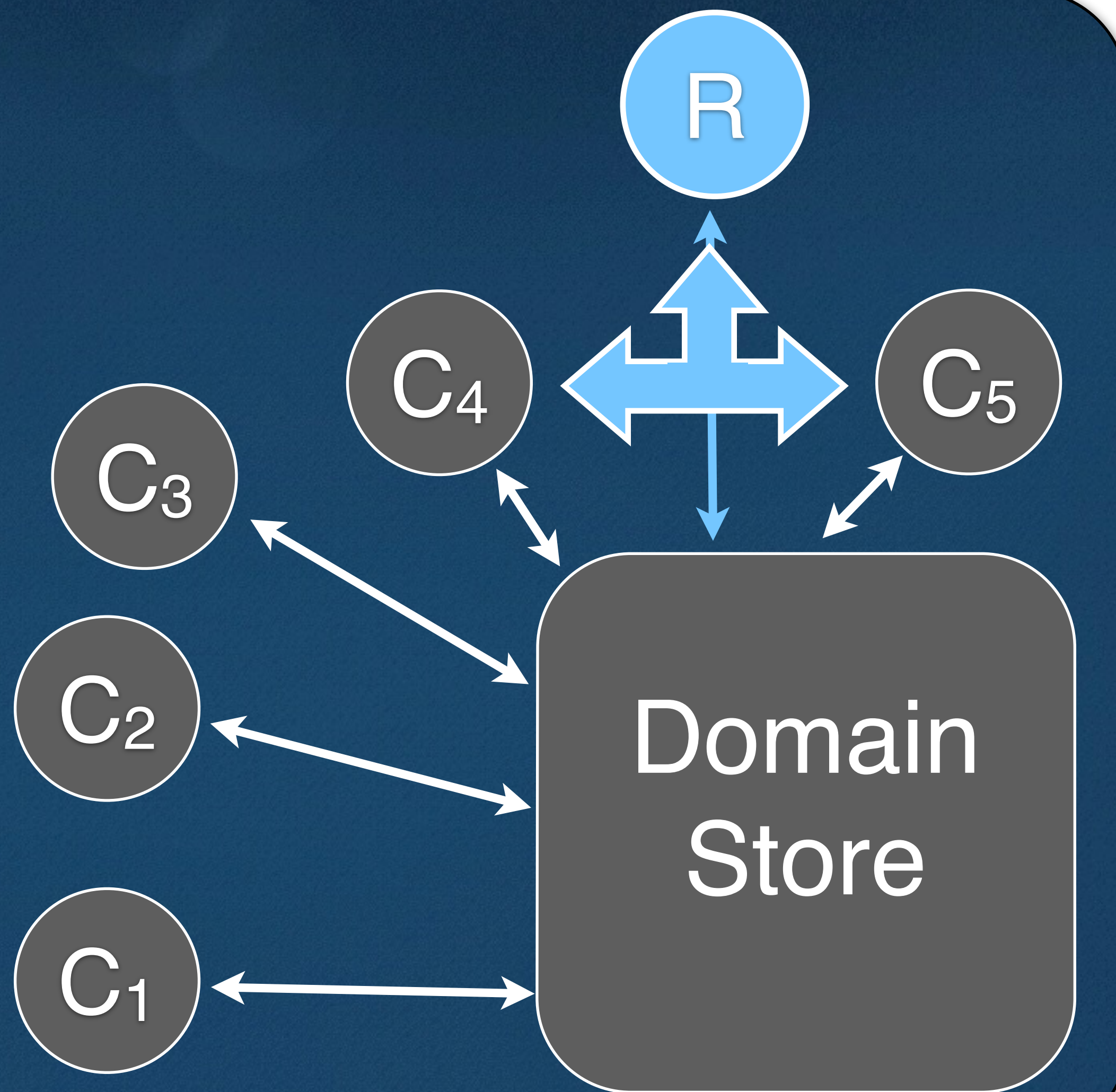
Constraint Store

Improving Communication



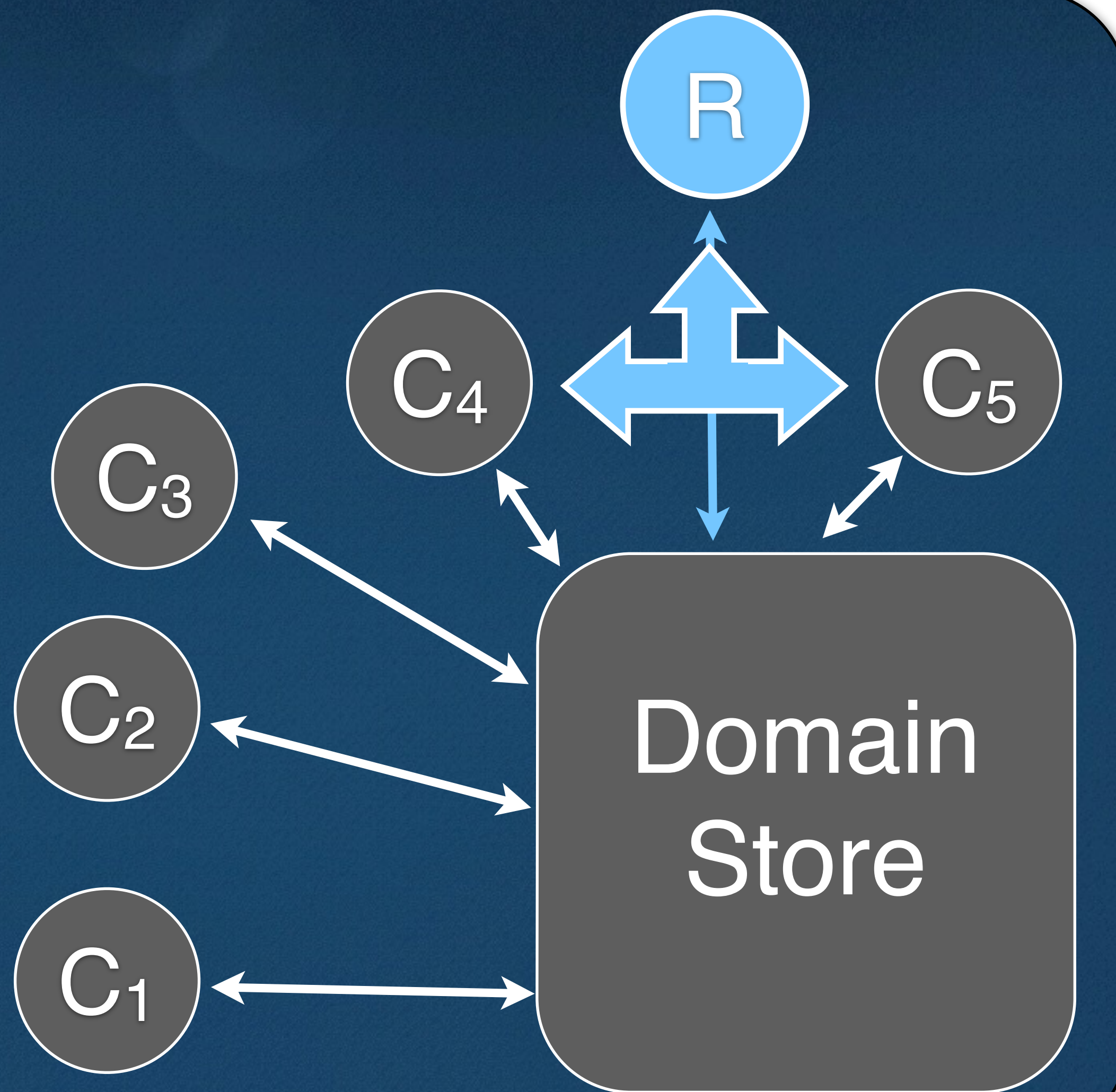
Constraint Store

Improving Communication: Surrogate Constraint



Constraint Store

Improving Communication: Implied Constraint



Constraint Store

Dual Modeling

- ▶ Sometimes there are multiple ways of modeling a problem
 - not the same decision variables
- ▶ The two models may have complementary strengths
 - hard to choose between them
 - some constraints are easier to express in one model and others in the other one
- ▶ Dual modeling
 - the idea of stating multiple models of a problem and linking them with constraints

Back to the 8-Queens Problem

- ▶ What are the decision variables?
 - many possible modelings
 - this is what makes optimization problems interesting :-)
- ▶ Here is one modeling
 - associate a decision variable with each column
 - the variable denotes the row of the queens placed in this column
- ▶ What are the constraints?
 - the queens cannot be placed on the same row
 - the queens cannot be placed on the same upward diagonal
 - the queens cannot be placed on the same downward diagonal

Back to the 8-Queens Problem

- ▶ What are the decision variables?
 - many possible modelings
 - this is what makes optimization problems interesting :-)
- ▶ Here is another modeling
 - associate a decision variable with each row
 - the variable denotes the column of the queens placed in this row
- ▶ What are the constraints?
 - the queens cannot be placed on the same row
 - the queens cannot be placed on the same upward diagonal
 - the queens cannot be placed on the same downward diagonal

The 8-Queens Problem with Dual Modeling

```
range R = 1..8;
range C = 1..8;
var{int} row[C] in R;
var{int} col[R] in C
solve {
    forall(i in R, j in R: i < j) {
        row[i] ≠ row[j];
        row[i] ≠ row[j] + (j - i);
        row[i] ≠ row[j] - (j - i);
    }
    forall(i in C, j in C: i < j) {
        col[i] ≠ col[j];
        col[i] ≠ col[j] + (j - i);
        col[i] ≠ col[j] - (j - i);
    }
    forall(r in R, c in C)
        (row[c] = r) <=> (col[r] = c);
}
```

Citations

Wolfsburg - Volkswagen Assembly Line (<http://www.flickr.com/photos/24736216@N07/2994043188/>) by Roger Wollstadt (<http://www.flickr.com/photos/24736216@N07/>) CC BY-SA 2.0 (<http://creativecommons.org/licenses/by-sa/2.0/deed.en>)

Geely assembly line in Beilun, Ningbo (http://commons.wikimedia.org/wiki/File:Geely_assembly_line_in_Beilun,_Ningbo.JPG) by Siyuwj (Own work) [CC-BY-SA-3.0 (<http://creativecommons.org/licenses/by-sa/3.0>)], via Wikimedia Commons