Discrete Optimization

Local Search: Part II

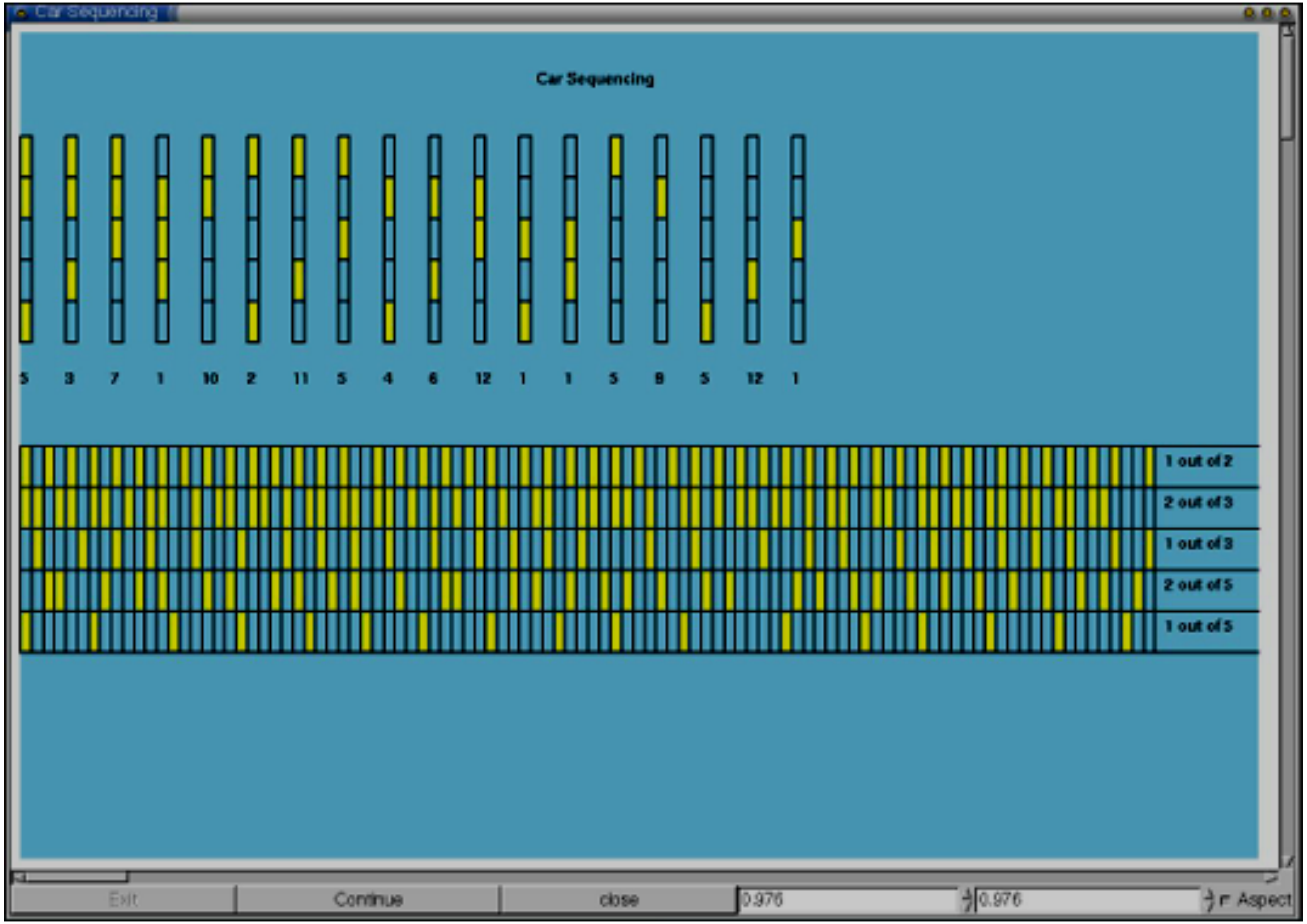
Goals of the Lecture

- Local search
  - swaps

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



Swaps

- Neighborhood
  - swap two configurations on the assembly line
- Search strategy
  - find a configuration that appears in violations
  - swap that configuration with another configuration to minimize the number of violations

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	1 1 1 3
Option 2											2/3	2
Option 3											1/3	2
Option 4											2/5	2
Option 5											1/5	3

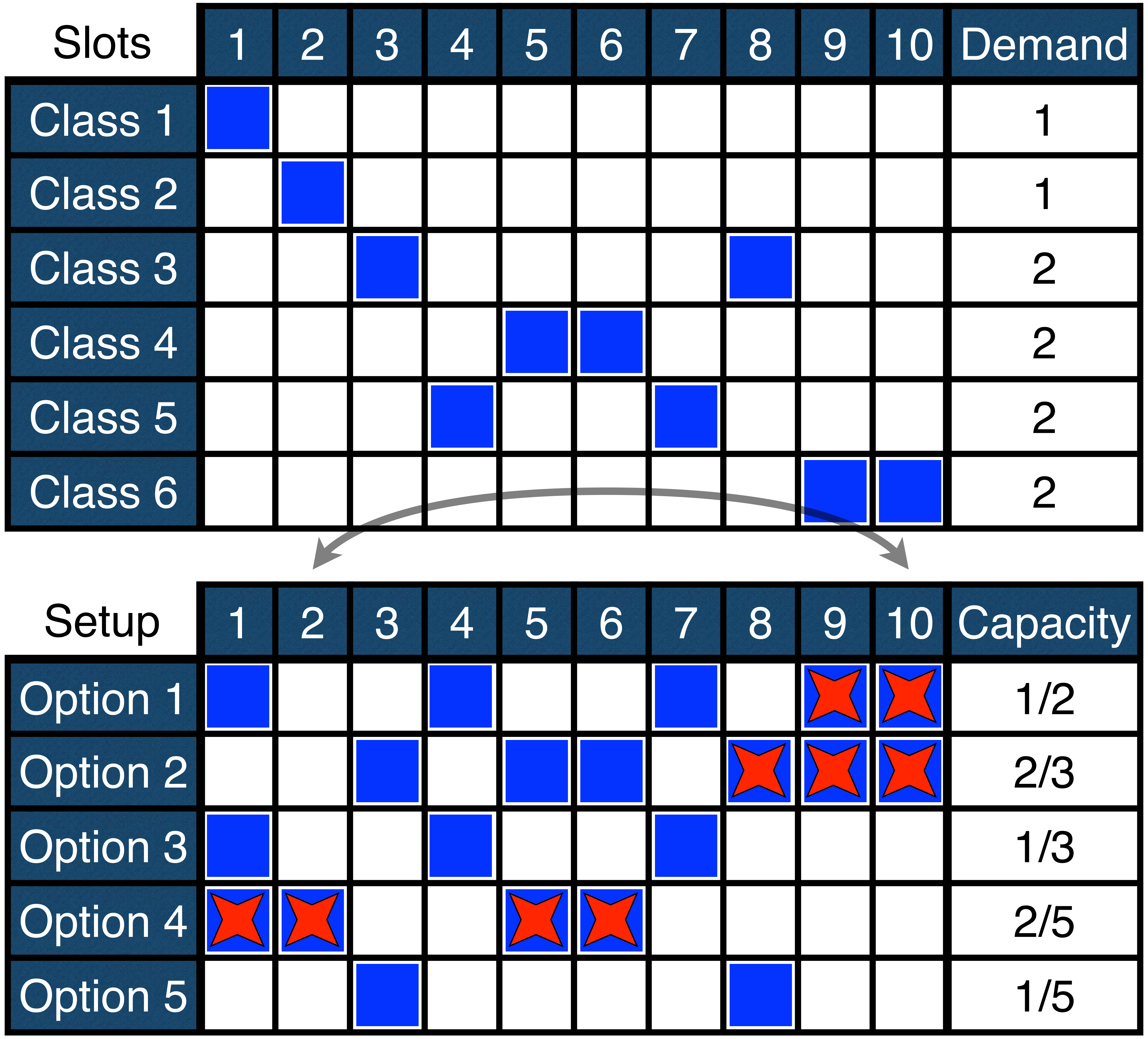
Car Sequencing

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	3
Option 2											2/3	2
Option 3											1/3	2
Option 4											2/5	2
Option 5											1/5	3

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	

1
1
0
2
0

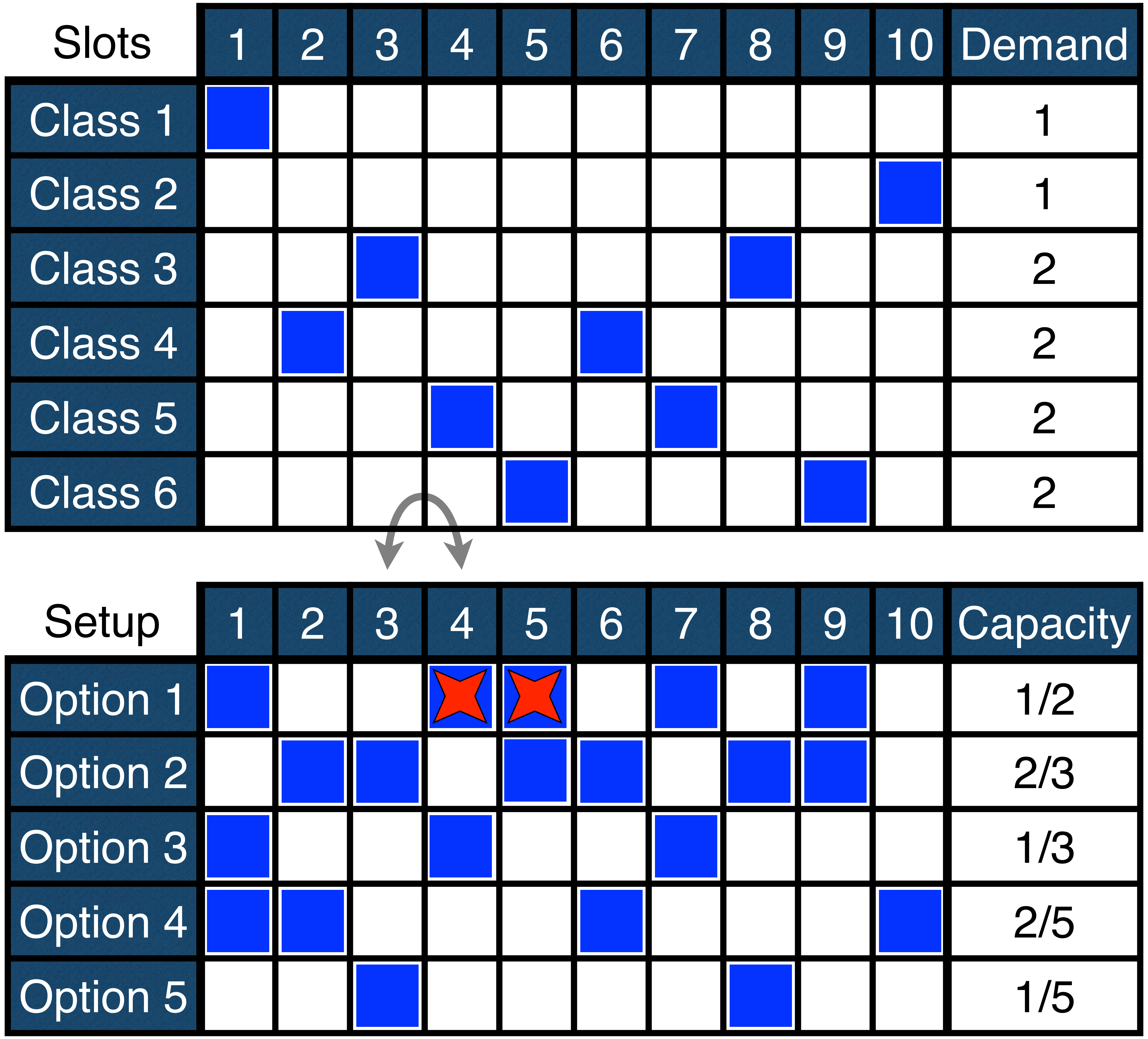
Car Sequencing

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	1
Option 2		■	■		■	■		■	■		2/3	0
Option 3	■			■			■				1/3	0
Option 4	■				■	■				■	2/5	0
Option 5			■					■			1/5	0

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



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	

1
0
0
0
0

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	0
Option 2											2/3	1
Option 3											1/3	1
Option 4											2/5	0
Option 5											1/5	1

Swaps

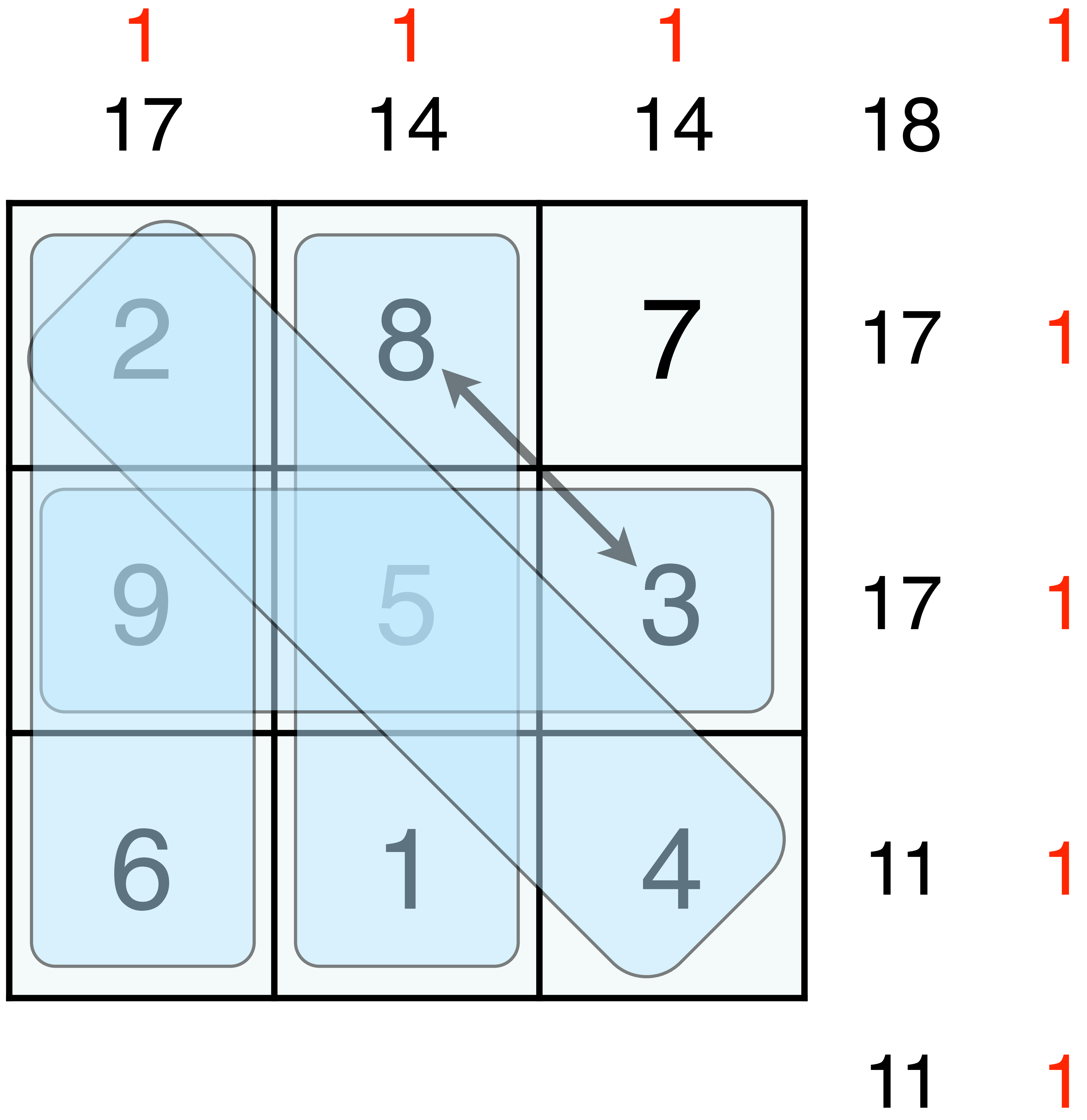
- Neighborhood
  - swap two configurations on the assembly line
- Why swaps, not assignments
  - automatically maintain the demand constraint, that is the correct number of configurations is indeed produced
  - hard constraints
    - always feasible during the search
  - soft constraints
    - may be violated during the search

The Magic Square Problem

```
range R = 1..n;
range D = 1..n^2;
int T = n*(n^2+1)/2;
var{int} s[R,R] in D;
solve {
   forall(i in R) {
      sum(j in R) s[i,j] = T;
      sum(j in R) s[j,i] = T;
   }
   sum(i in R) s[i,i] = T;
   sum(i in R) s[i,n-i+1] = T;
   alldifferent(all(i in R,j in R) s[i,j]);
}
```

The Magic Square Problem

Magic Number: 15



The Magic Square Problem

Magic Number: 15

1	1	1		1
17	9	19	18	
2	3	7	12	1
9	5	8	22	1
6	1	4	11	1
			11	1

Neighborhood for the Magic Square Problem

- Swapping the value of two cells
  - the alldifferent constraint as a hard constraint
    - swaps are used in many permutation problems
  - the inequalities are the soft constraints
- What the violations?
  - 0/1 violations would be pretty useless
  - many moves would not change the violations
  - purely random walk
- For an equation l = r
  - use abs(l - r) as a measure of violations
  - drives the search much more effectively

The Magic Square Problem

Magic Number: 15

2	1	1		3	= 21
17	14	14	18		
2	8	7	17	2	
9	5	3	17	3	
6	1	4	11	4	
			11	4	

The Magic Square Problem

Magic Number: 15

1	0	1		3	= 14
16	15	14	18		
2	9	7	18	3	
8	5	3	16	1	
6	1	4	11	4	
			11	4	

The Magic Square Problem

Magic Number: 15

1	0	1		0	= 5
16	15	14	15		
2	9	4	15	0	
8	5	3	16	1	
6	1	7	14	1	
			14	1	

The Magic Square Problem

Magic Number: 15

0	0	0		0	= 0
15	15	15	15		
2	9	4	15	0	
7	5	3	15	0	
6	1	8	15	0	
			15	0	

Until Next Time

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