Discrete Optimization

LDS and LNS

Goals of the Lecture

- Searching
  - Limited discrepancy search (LDS)
    - how to explore the search tree more effectively?
  - Large neighborhood search (LNS)
    - hybridization of local and CP/MIP search

Limited Discrepancy Search

- Assume that a good heuristic is available
  - it makes very few mistakes
  - the search tree is binary
    - typical in disjunctive scheduling
  - following the heuristic means branching **left**
  - branching **right** means the heuristic was wrong

Limited Discrepancy Search

- Assume that a good heuristic is available
  - it makes very few mistakes
  - the search tree is binary
    - typical in disjunctive scheduling
  - following the heuristic means branching **left**
  - branching **right** means the heuristic was wrong
- Limited Discrepancy Search (LDS)
  - explore the search space in increasing order of mistakes
  - trusting the heuristic less and less

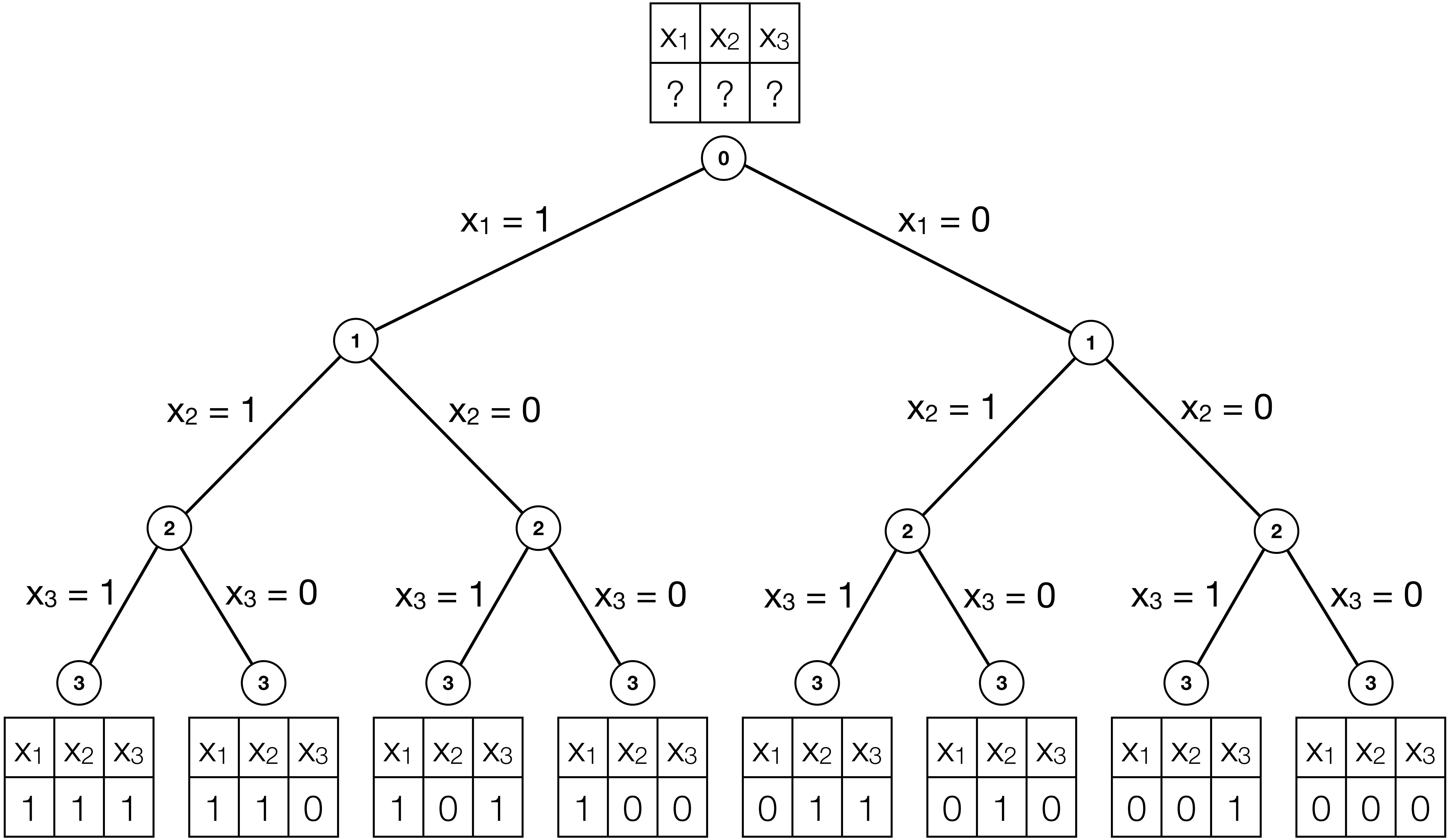
Limited Discrepancy Search

- Limited Discrepancy Search (LDS)
  - explore the search space in increasing order of mistakes
  - trusting the heuristic less and less

Limited Discrepancy Search

- Limited Discrepancy Search (LDS)
  - explore the search space in increasing order of mistakes
  - trusting the heuristic less and less
- Explores the search space in waves
  - no mistake
  - one mistake
  - two mistakes
  - ....

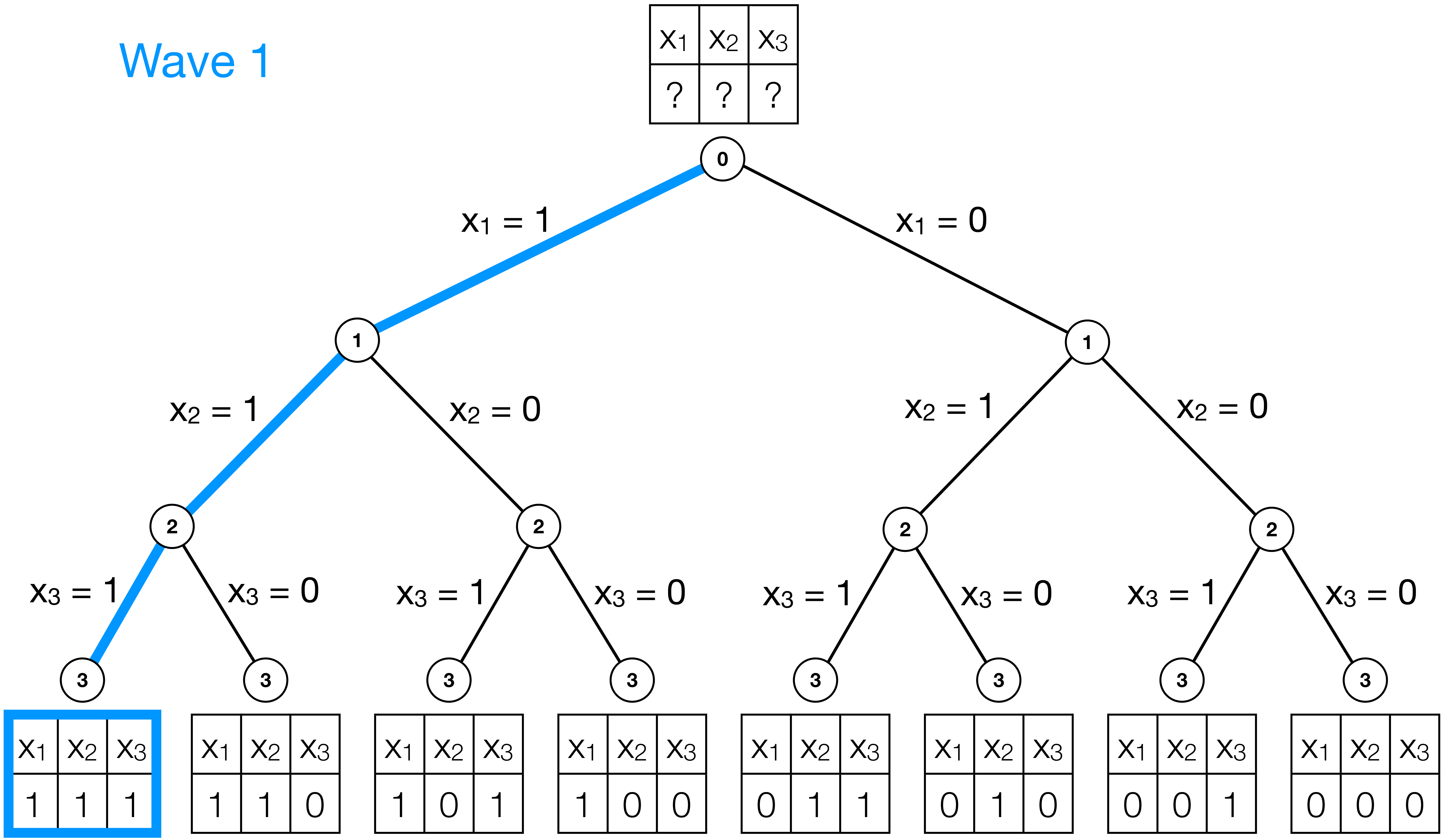
Waves



x1 x2 x3
? ? ?
0
x1 = 1
x1 = 0
1
1
x2 = 1
x2 = 0
x2 = 1
x2 = 0
2
2
2
2
x3 = 1
x3 = 0
x3 = 1
x3 = 0
x3 = 1
x3 = 0
x3 = 1
x3 = 0
3
3
3
3
3
3
3
3
x1 x2 x3
1 1 1
x1 x2 x3
1 1 0
x1 x2 x3
1 0 1
x1 x2 x3
1 0 0
x1 x2 x3
0 1 1
x1 x2 x3
0 1 0
x1 x2 x3
0 0 1
x1 x2 x3
0 0 0

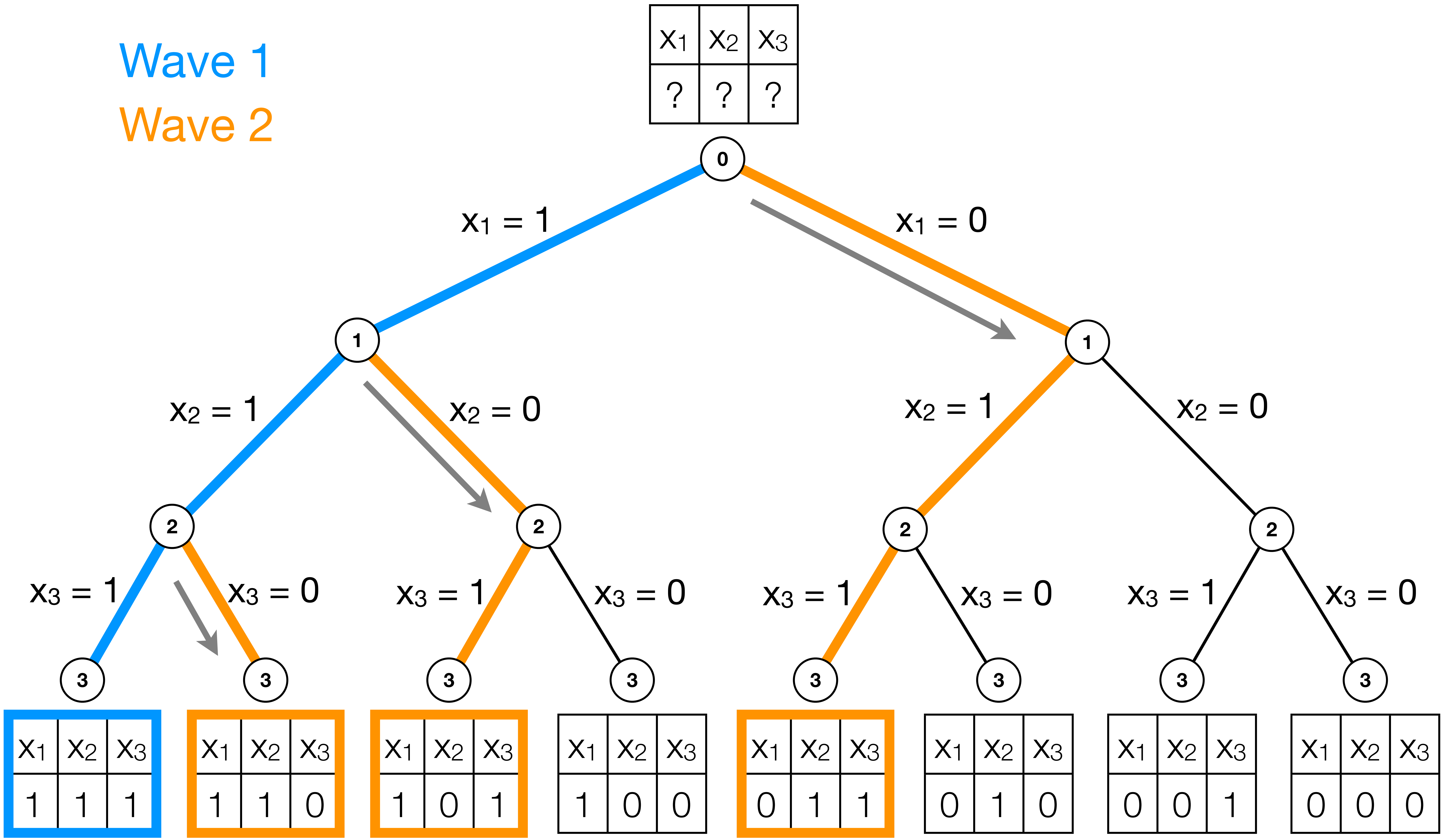
Waves

Wave 1



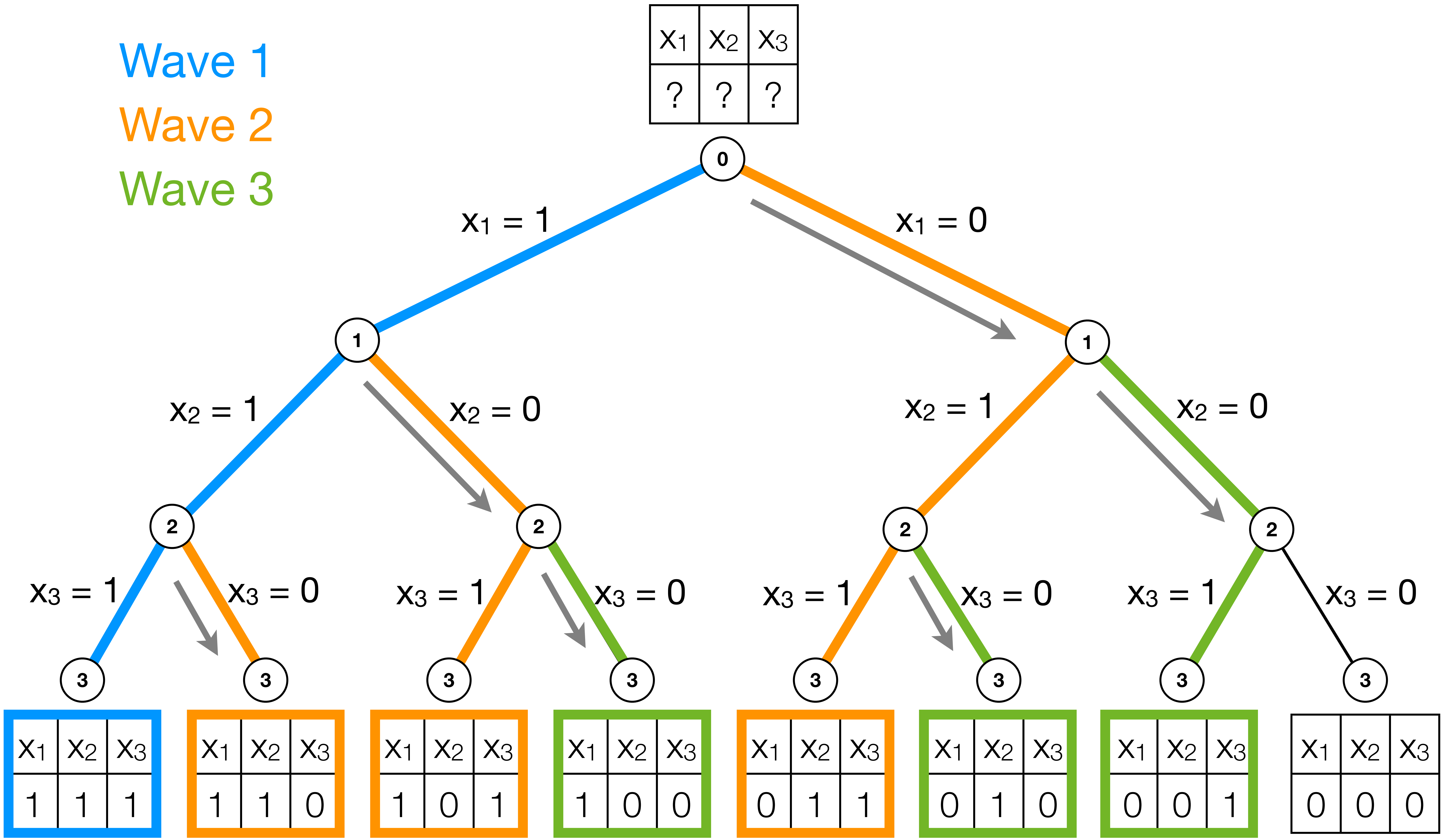
| x_1 | x_2 | x_3 |
| ? | ? | ? |
0
$x_1 = 1$
$x_1 = 0$
1
1
$x_2 = 1$
$x_2 = 0$
$x_2 = 1$
$x_2 = 0$
2
2
2
2
$x_3 = 1$
$x_3 = 0$
$x_3 = 1$
$x_3 = 0$
$x_3 = 1$
$x_3 = 0$
$x_3 = 1$
$x_3 = 0$
3
3
3
3
3
3
3
3
x_1	x_2	x_3
1	1	1
x_1	x_2	x_3
1	1	0
x_1	x_2	x_3
1	0	1
x_1	x_2	x_3
1	0	0
x_1	x_2	x_3
0	1	1
x_1	x_2	x_3
0	1	0
x_1	x_2	x_3
0	0	1
x_1	x_2	x_3
0	0	0

Waves



Wave 1
Wave 2
x1 x2 x3
? ? ?
0
x1 = 1
x1 = 0
1
1
x2 = 1
x2 = 0
x2 = 1
x2 = 0
2
2
2
2
x3 = 1
x3 = 0
x3 = 1
x3 = 0
x3 = 1
x3 = 0
x3 = 1
x3 = 0
3
3
3
3
3
3
3
3
x1 x2 x3
1 1 1
x1 x2 x3
1 1 0
x1 x2 x3
1 0 1
x1 x2 x3
1 0 0
x1 x2 x3
0 1 1
x1 x2 x3
0 1 0
x1 x2 x3
0 0 1
x1 x2 x3
0 0 0

Waves


Wave 1
Wave 2
Wave 3
x1 x2 x3
? ? ?
0
x1 = 1
x1 = 0
1
1
x2 = 1
x2 = 0
x2 = 1
x2 = 0
2
2
2
2
x3 = 1
x3 = 0
x3 = 1
x3 = 0
x3 = 1
x3 = 0
x3 = 1
x3 = 0
3
3
3
3
3
3
3
3
x1 x2 x3
1 1 1
x1 x2 x3
1 1 0
x1 x2 x3
1 0 1
x1 x2 x3
1 0 0
x1 x2 x3
0 1 1
x1 x2 x3
0 1 0
x1 x2 x3
0 0 1
x1 x2 x3
0 0 0

Large Neighborhood Search (LNS)

- Combination of CP and Local Search
  1. start with a feasible solution (CP)
  2. select a neighborhood (LS)
  3. optimize the neighborhood (CP)
  4. repeat

Large Neighborhood Search (LNS)

- Combination of CP and Local Search
  1. start with a feasible solution (CP)
  2. select a neighborhood (LS)
  3. optimize the neighborhood (CP)
  4. repeat
- What is the neighborhood?
  - fix a subset of variables to their values in the best solution found so far
  - which subset?
    - problem-specific
    - exploit the problem structure

Large Neighborhood Search (LNS)

- Combination of CP and Local Search
  1. start with a feasible solution (CP)
  2. select a neighborhood (LS)
  3. optimize the neighborhood (CP)
  4. repeat from step 2
- Why LNS?
  - CP is good to find feasible solutions
  - CP is good in optimizing small combinatorial spaces

Large Neighborhood Search (LNS)

- Combination of CP and Local Search
  1. start with a feasible solution (CP)
  2. select a neighborhood (LS)
  3. optimize the neighborhood (CP)
  4. repeat from step 2
- Why LNS?
  - CP is good to find feasible solutions
  - CP is good in optimizing small combinatorial spaces
- Generalize to MIP directly of course!

Asymmetric TSP with Time Windows

- Given
  - a set of locations to visit
  - a service time for each location
  - a time window for each location
  - a possibly asymmetric distance between locations

Asymmetric TSP with Time Windows

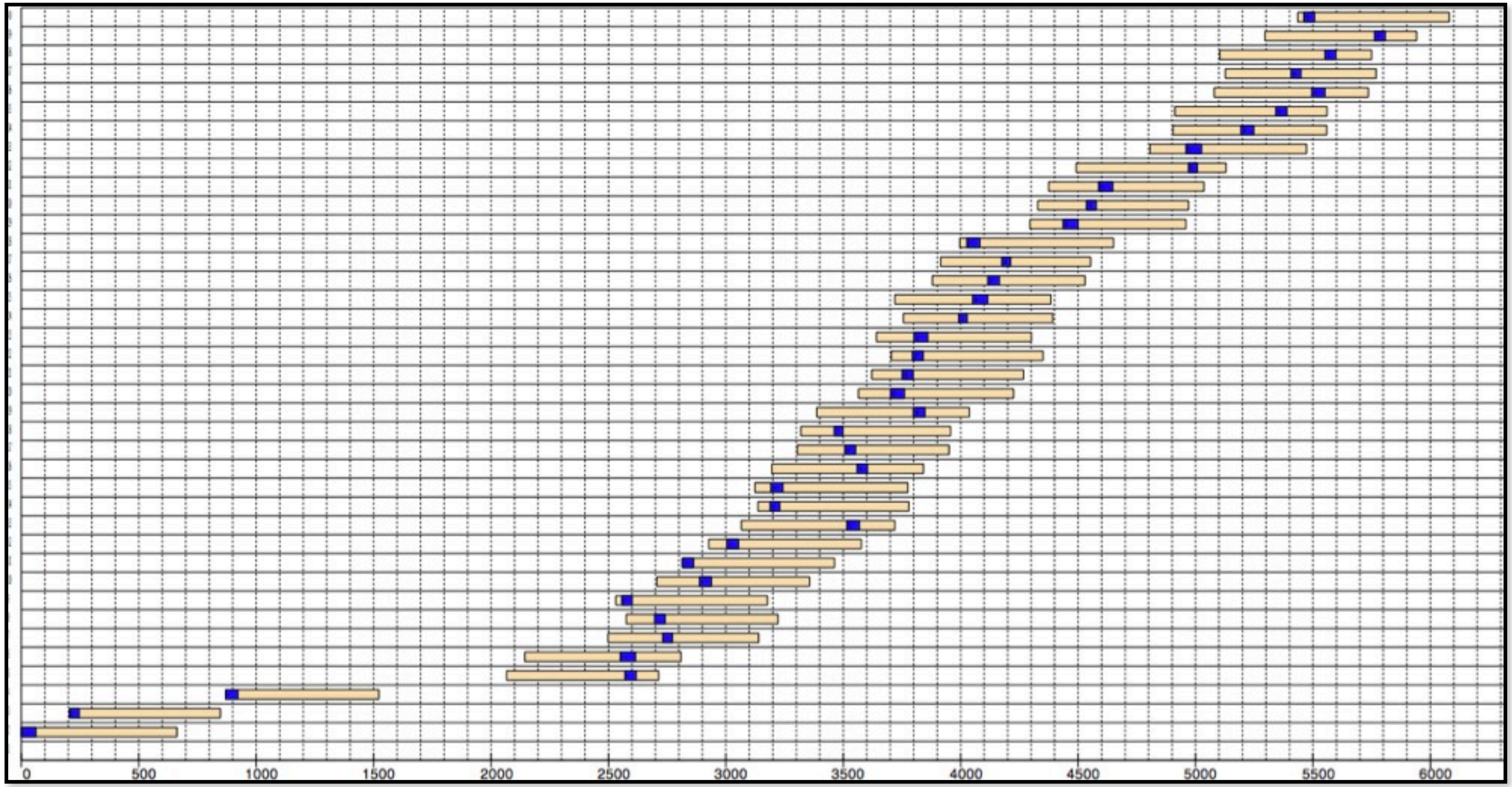
- Given
  - a set of locations to visit
  - a service time for each location
  - a time window for each location
  - a possibly asymmetric distance between locations
- Find a Hamiltonian path
  - satisfying the time windows
  - minimizing the total travel distance

Asymmetric TSP with Time Windows

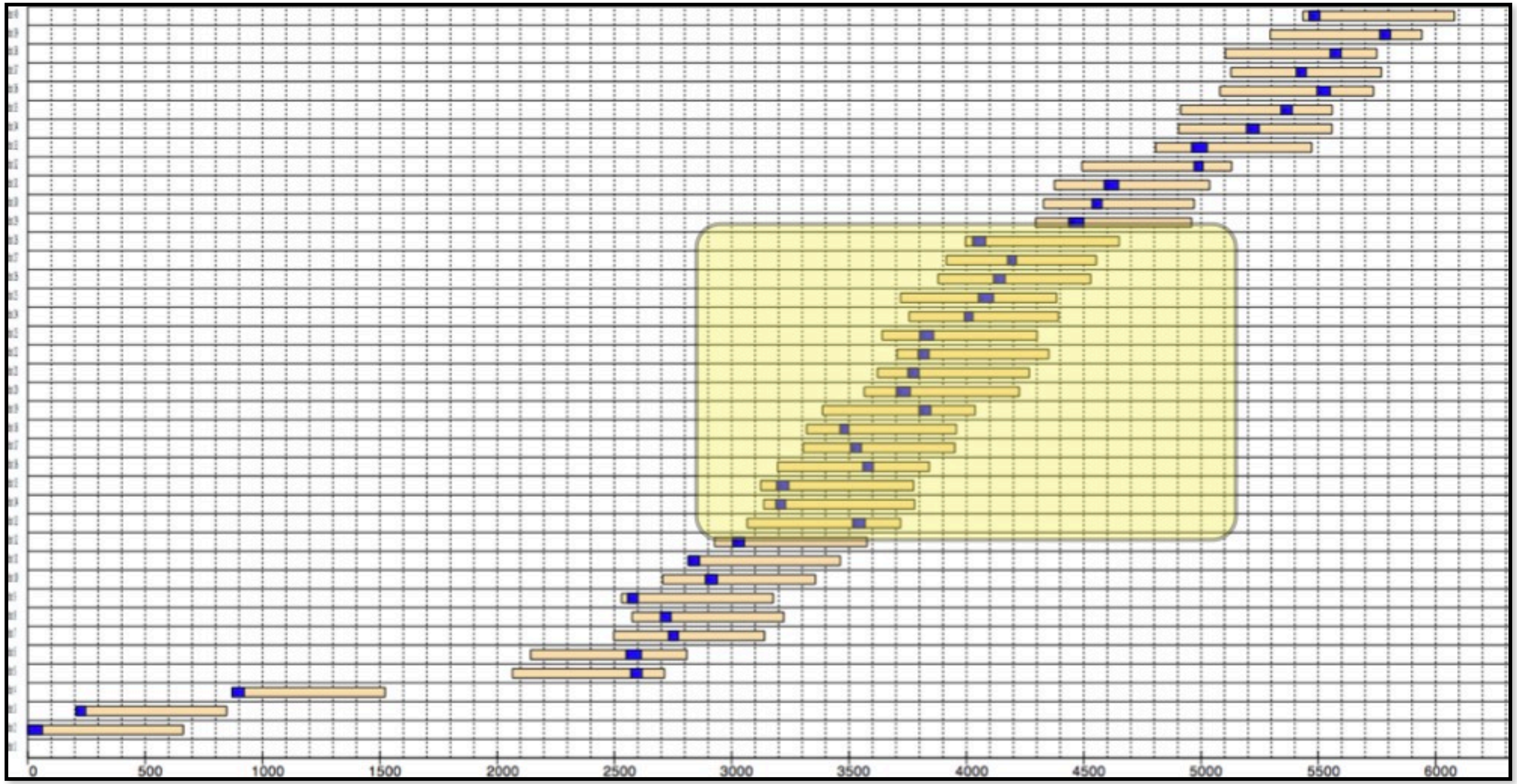
```
Scheduler sched(horizon);
Activity act[a in Activities](sched,service[a]);
UnaryResource vehicle(sched,transitionTimes);

minimize
   vehicle.transitionTimes
subject to {
   forall(a in Activities) {
      act[a].start >= ws[a];
      act[a].start <= we[a];
      act[a] requires vehicle;
   }
}
```

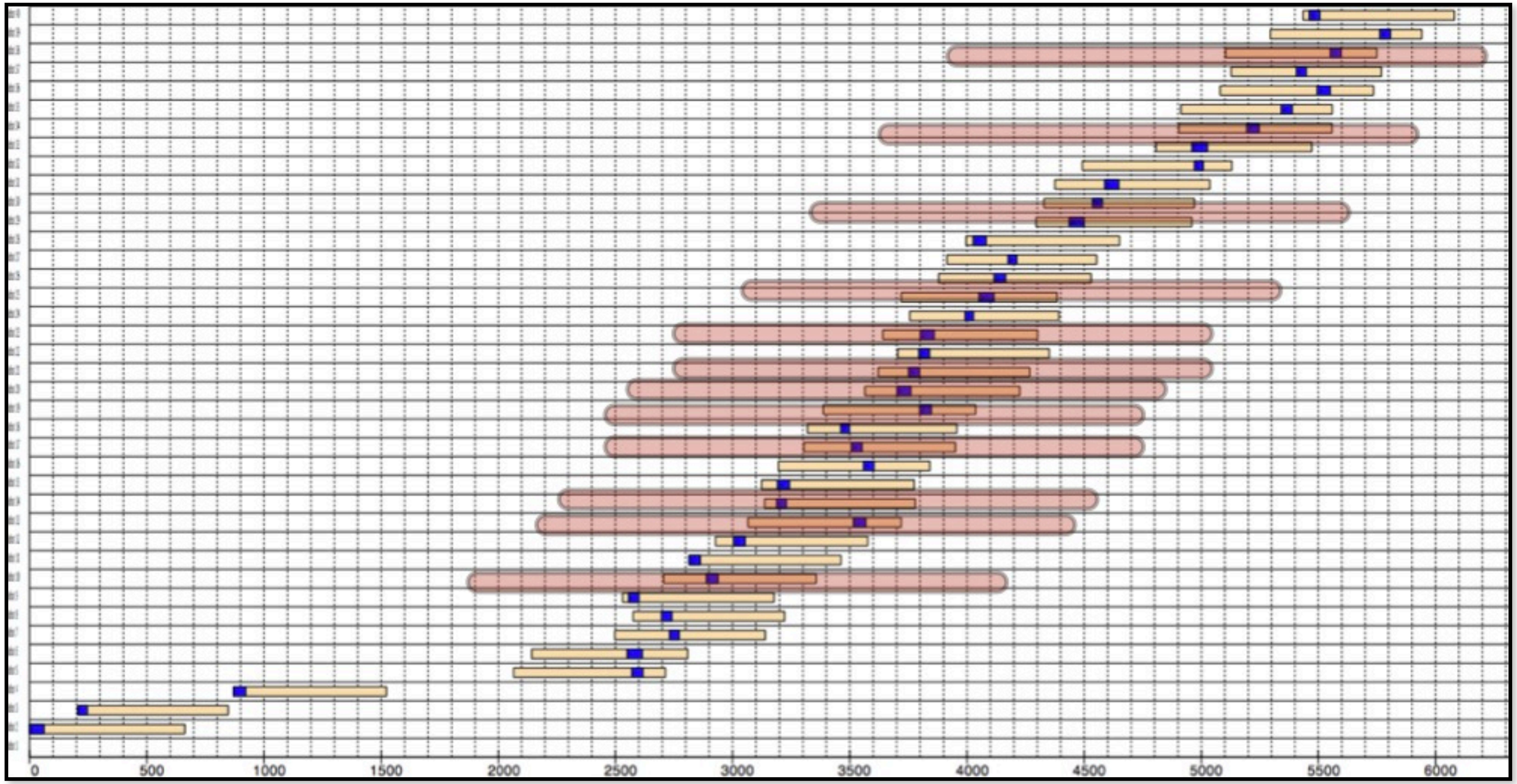
Asymmetric TSP with Time Windows



LNS for the ATSPTW



LNS for the ATSPTW



LNS for the ATSPTW

Size	BK
40	386
48	492
49	488
50	414
67	1048
86	1052
92	1111
125	1410
132	1400
152	1792
172	1897
193	2452
201	2296
233	2786

LNS for the ATSPTW

Size	BK	300s
40	386	386
48	492	487
49	488	484
50	414	414
67	1048	1048
86	1052	1051
92	1111	1093
125	1410	1409
132	1400	1382
152	1792	1783
172	1897	1870
193	2452	
201	2296	
233	2786	

LNS for the ATSPTW

Size	BK	300s	600s
40	386	386	
48	492	487	
49	488	484	
50	414	414	
67	1048	1048	
86	1052	1051	
92	1111	1093	
125	1410	1409	
132	1400	1382	
152	1792	1783	
172	1897	1870	1799
193	2452		2433
201	2296		2234
233	2786		2683

Until Next Time