




Travelling Salesman Problem

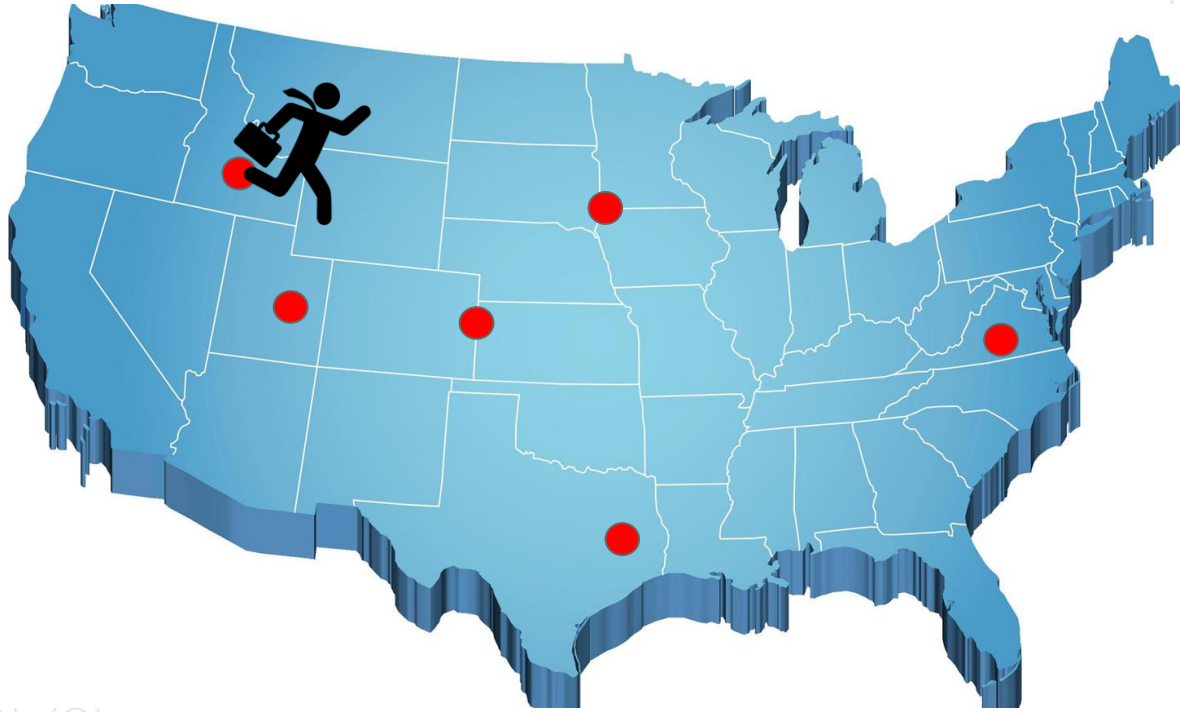
Group 26: Chang Kon Han, John Law, Wesley Yep



Outline

- ◎ How can the A* algorithm be used to solve TSP?
 - ◎ How can the parallelising be done with ParaTask?
 - ◎ What tweaks can be done to improve performance and scalability?
 - ◎ How does the parallel implementation compare to the sequential version?
- 

Travelling Salesman Problem



Greedy Approach



Optimal Solution



(6) Travel salesman problem TSP

NP-hard

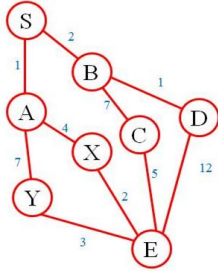
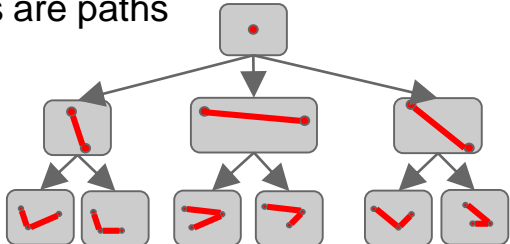

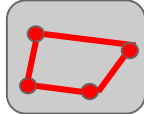
Travel salesman problem (TSP) is another ~~NP-complete~~ problem.

- ▶ MST can't solve TSP as the cycle has not to be completed (TSP is a cycle)
- ▶ Salesman needs to go home (what's the cost going home?)
- ▶ We can use MST to approximate the solution
- ▶ As solving TSP by exhaustive approach is not a best choice

A* Algorithm

- ◎ Tree Search Algorithm
- ◎ Heuristic Search
- ◎ Best First Search Algorithm
- ◎ $f(n) = g(n) + h(n)$

A* for TSP

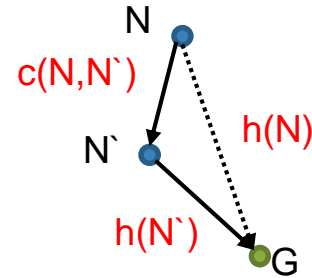
A* for Path Finding	A* for TSP
<p data-bbox="214 285 484 317">States are cities</p> 	<p data-bbox="977 285 1248 317">States are paths</p> 
<p data-bbox="214 601 629 634">Goal state is a single city</p> 	<p data-bbox="977 601 1528 634">Goal state is a Hamiltonian Cycle</p> 
<p data-bbox="214 814 813 896">Neighbours are all the cities that are connected to a city</p>	<p data-bbox="977 814 1692 940">Neighbours are all graphs that can be formed by the addition of one edge from an existing state</p>

Heuristic

Admissible

- ⊙ Never overestimates cost to the goal

Consistent

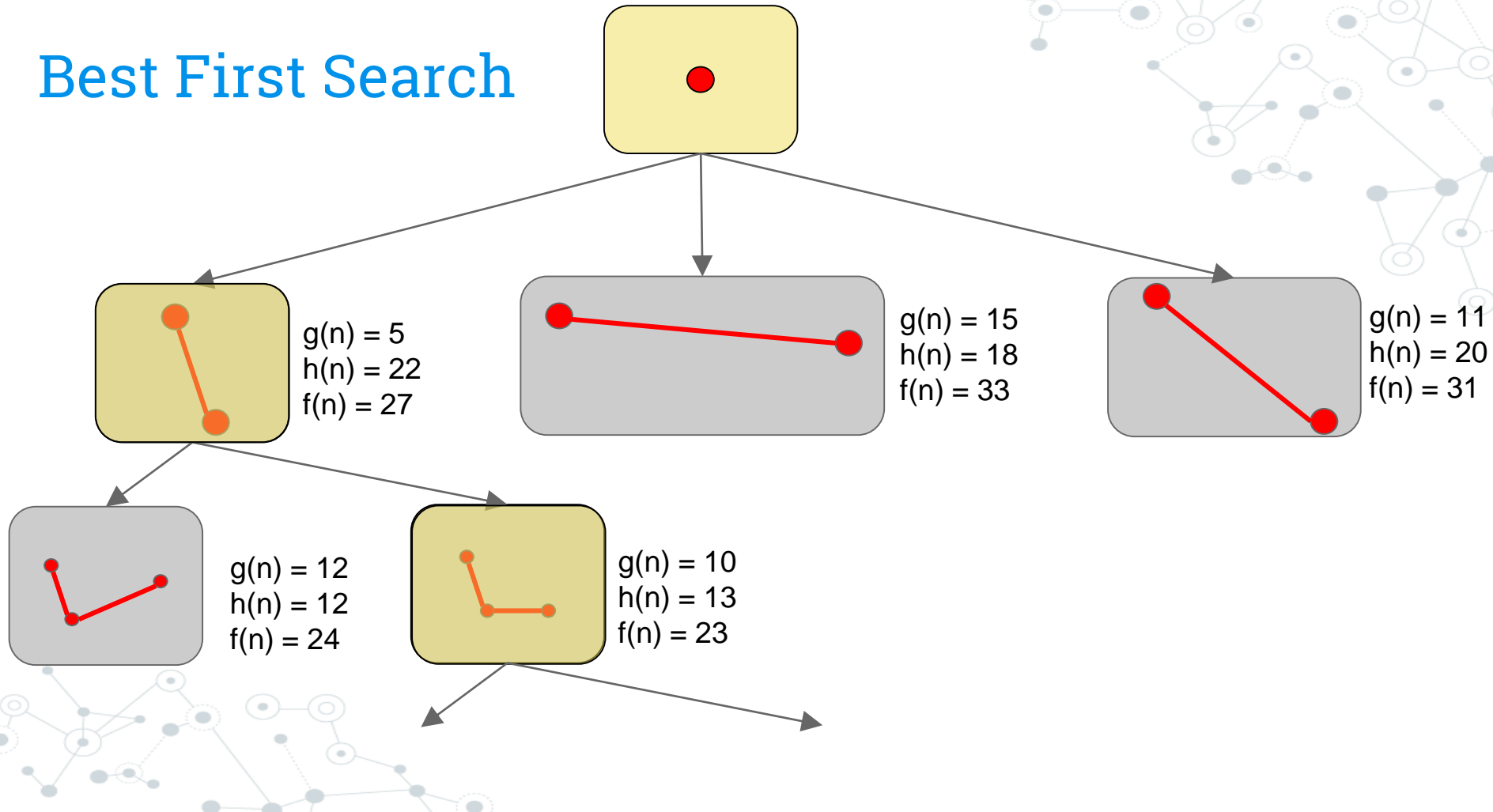


- ⊙ $h(N) \leq c(N, N') + h(N')$

Minimum Spanning Tree (heuristic)



Best First Search



A* for TSP: Pseudocode

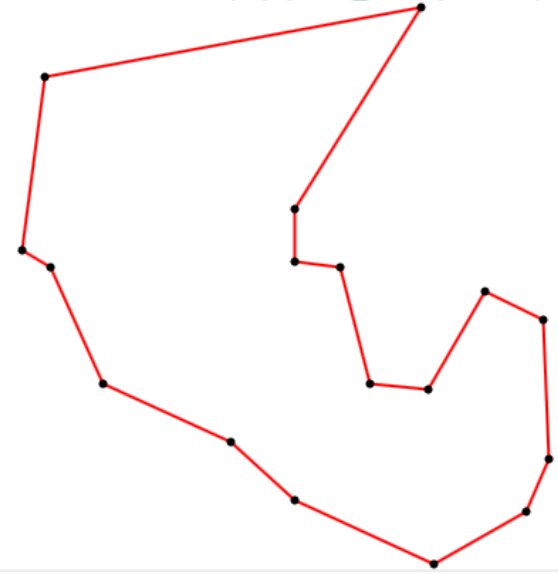
Algorithm 1: A* for TSP

```
1  $s_0 \leftarrow state(initialCity);$ 
2  $frontier \leftarrow priorityqueue();$ 
3  $frontier.add(s_0);$ 
4  $s \leftarrow s_0;$ 
5 while  $s$  not hamiltonian cycle do
6    $children \leftarrow s.expand();$ 
7   foreach  $child$  in  $children$  do
8      $frontier.add(child, childValue);$ 
9   end
10   $s \leftarrow frontier.poll();$ 
11 end
12  $s \leftarrow s + initialCity;$ 
13 return  $s;$ 
```

```
NAME : Test17
COMMENT : 17 cities
TYPE : TSP
DIMENSION : 17
EDGE_WEIGHT_TYPE : EUC_2D
NODE_COORD_SECTION
1 21 1
2 54 87
3 14 45
4 34 78
5 12 5
6 50 12
7 34 32
8 54 37
9 64 45
10 55 45
11 3 21
12 45 2
13 57 92
14 24 56
15 87 88
16 33 22
17 99 23
EOF
```

Libraries

TSPLIB
TSPLIB4J



<http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/>
<https://github.com/dhadka/TSPLIB4J>

Parallel A*

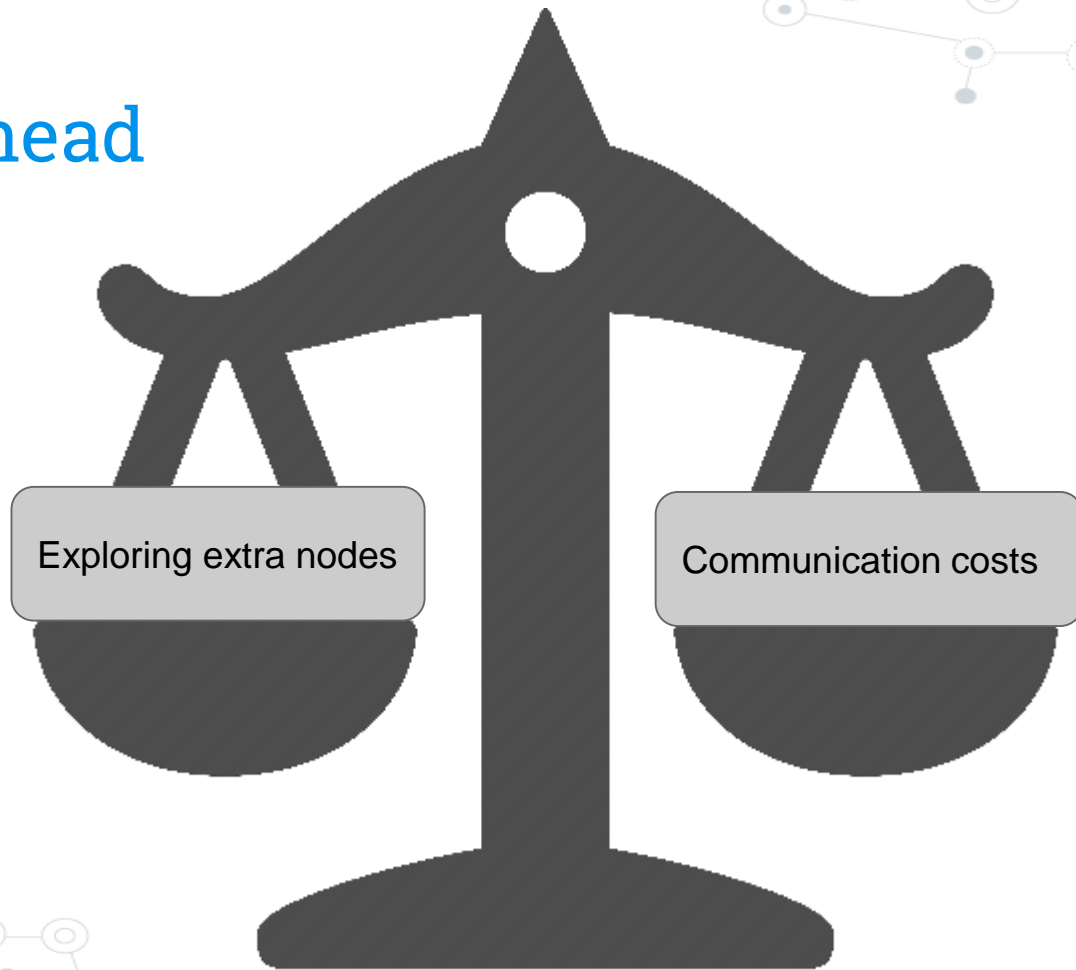
Task Parallelism

©Difficult due to the sequential nature of expanding node

Data Parallelism

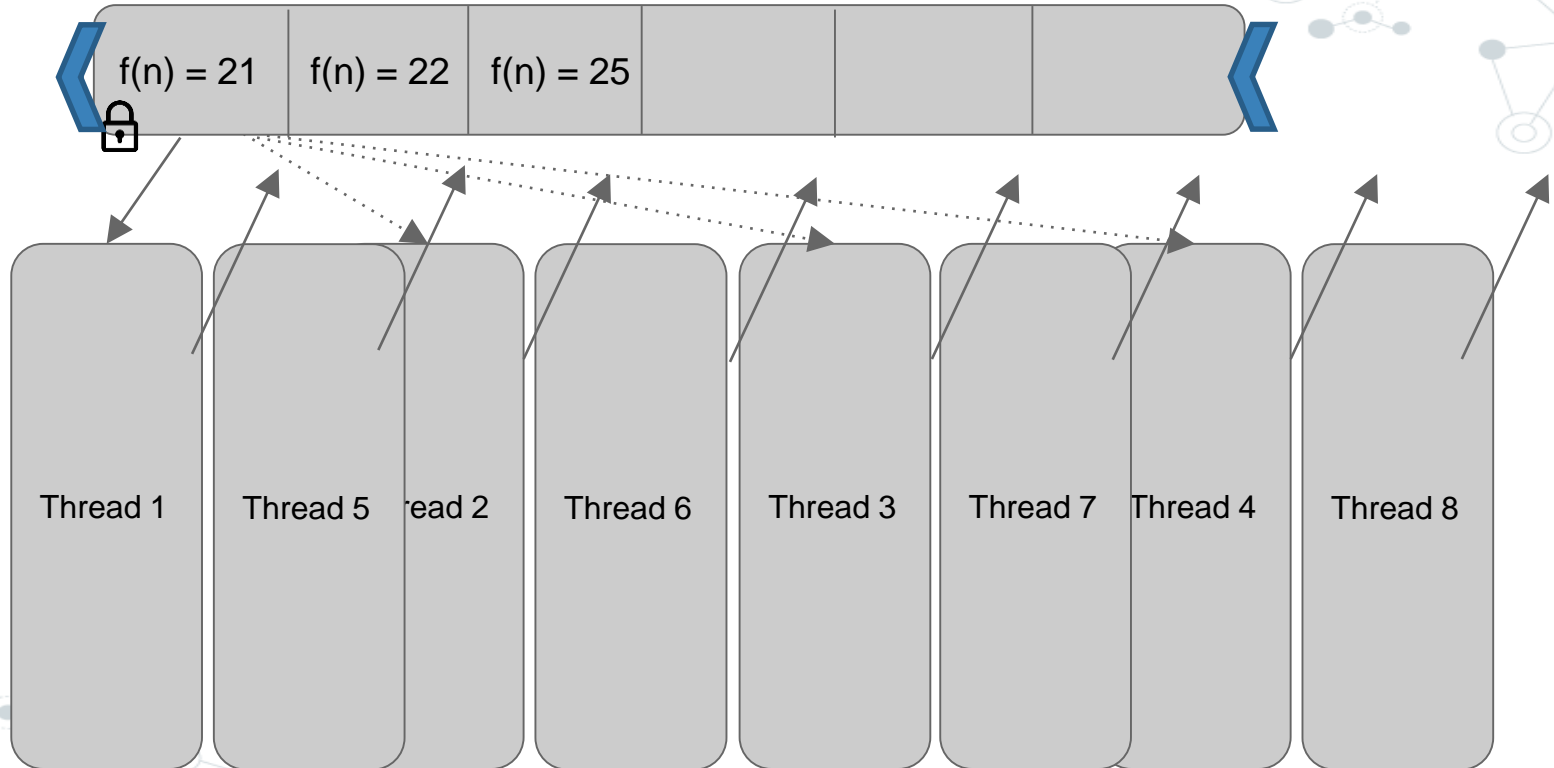
- ©Centralised, using shared frontier (Centralised List A*)
- ©Localised, using shared memory (Blackboard)
- ©Localised, using message passing (HDA*)

Overhead

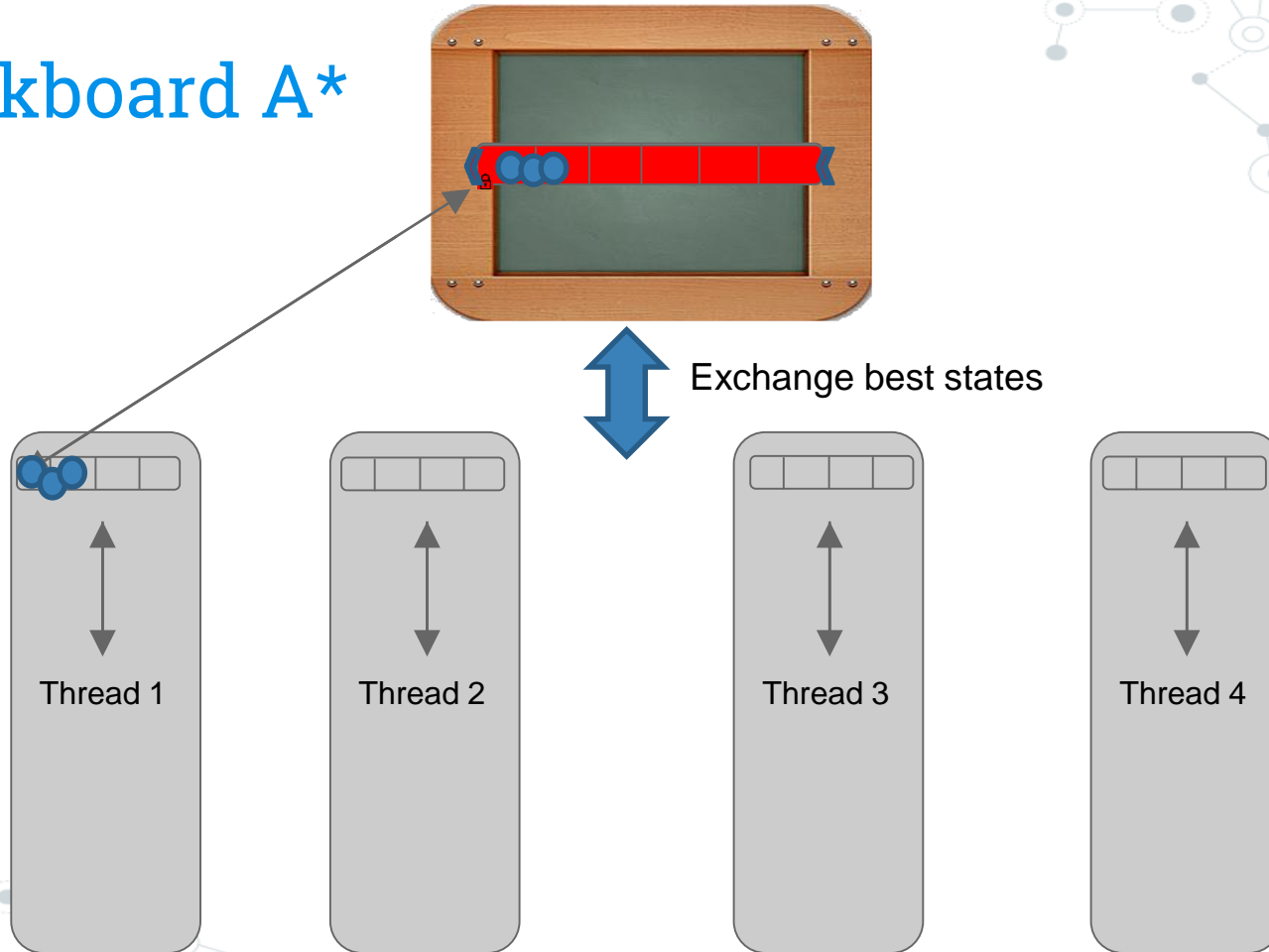


Centralised Shared Frontier A*

```
TASK(*) public void solveTSP() { ...
```



Blackboard A*

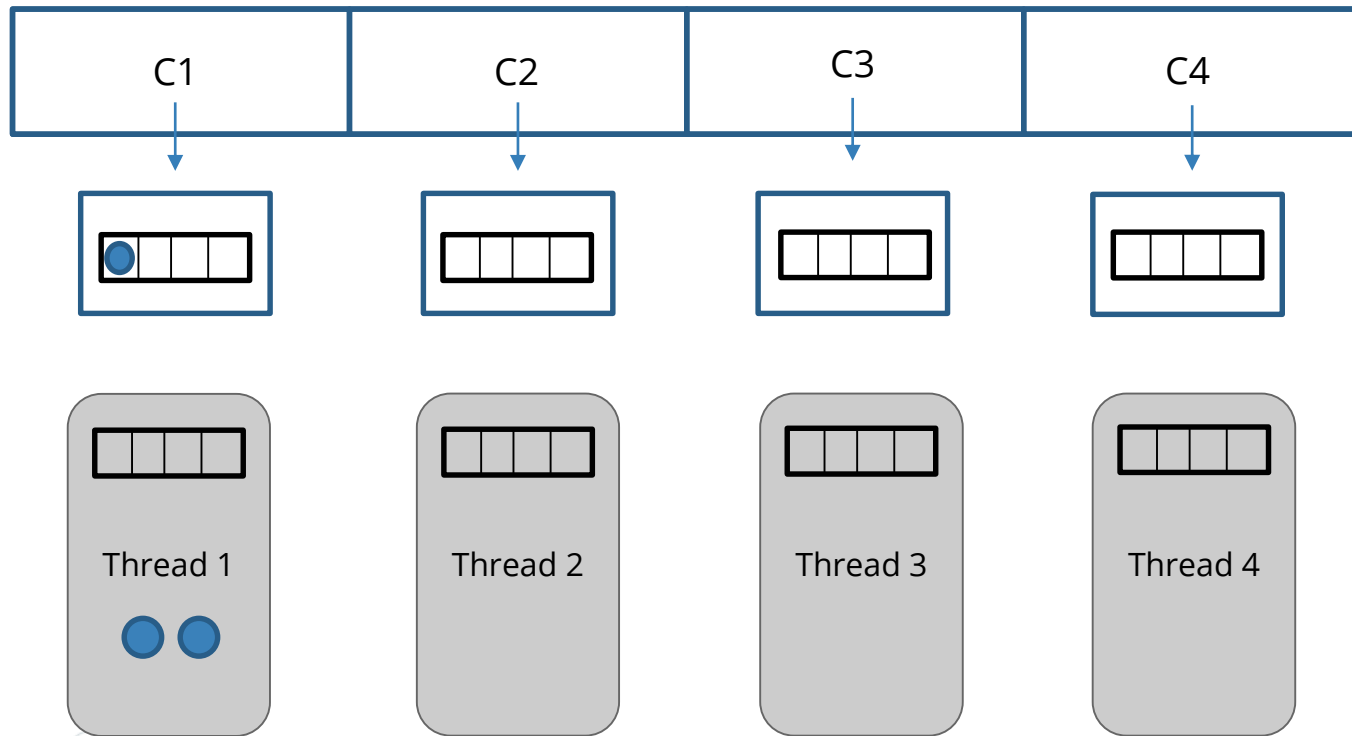


Blackboard Distributed A*

- ✓ Minimum access to shared data structure
- ✓ Reduced synchronisation cost
- ☹ Performance depends on parameters
(e.g. Threshold and number of states copied)

Hash Distributed A*

Shared Channel



Scalable, Parallel Best-First Search for Optimal Sequential Planning

Akihiro Kishimoto, Alex Fukunaga, Adi Botea

Hash Distributed A*

- ✓ Reduced synchronization overhead
- ✓ Simple hash based work distribution

Scalable, Parallel Best-First Search for Optimal Sequential Planning

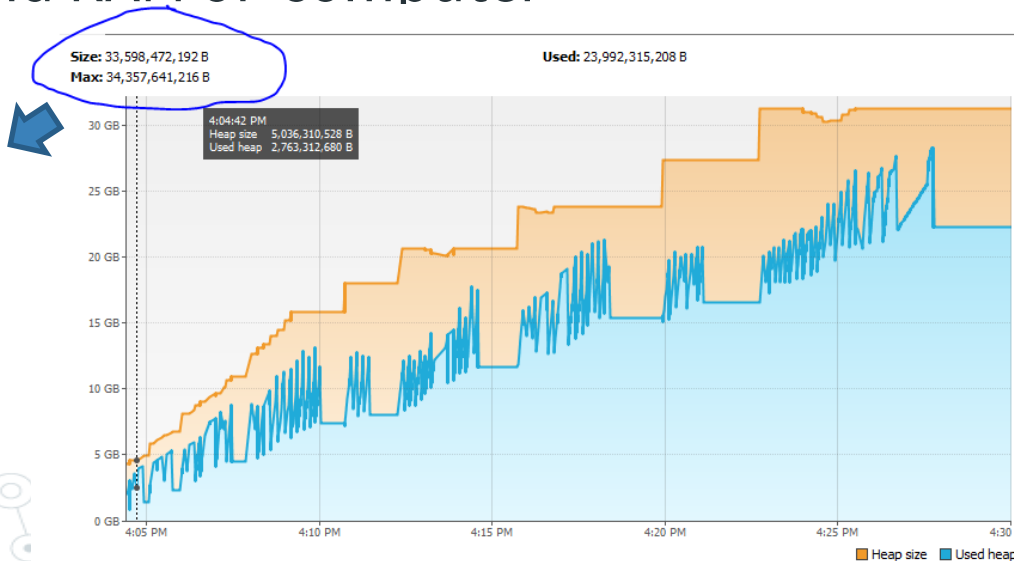
Akihiro Kishimoto, Alex Fukunaga, Adi Botea

TSP A* Demos

Scalability

- A* for TSP uses exponential memory in the worst case
- Problems with the Java Garbage Collector, JVM heap size, and RAM of computer

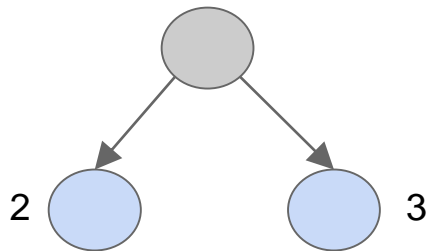
Heap
Size: 33,598,472,192 B
Max: 34,357,641,216 B
(Over 32 GB!)



Iterative Deepening A*

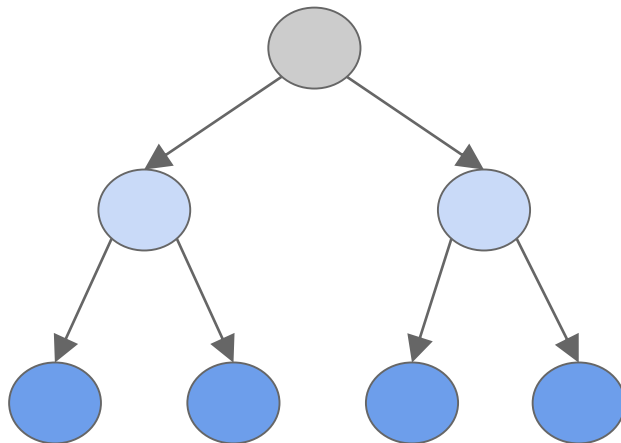
Iteration 1

Bound = 1



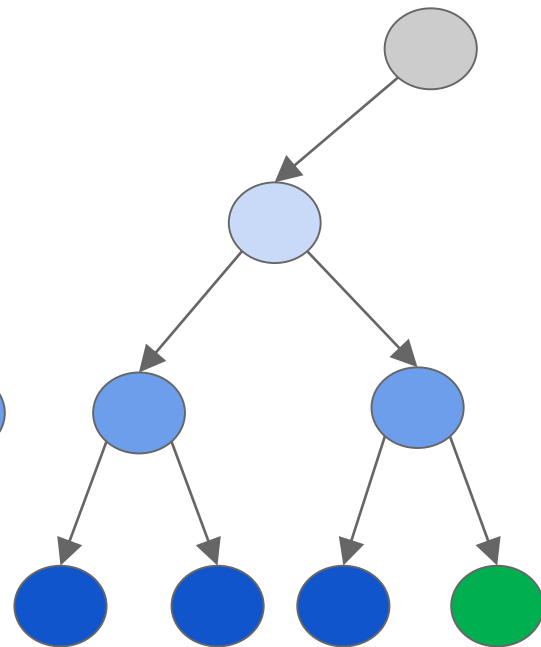
Iteration 2

Bound = 2



Iteration 3

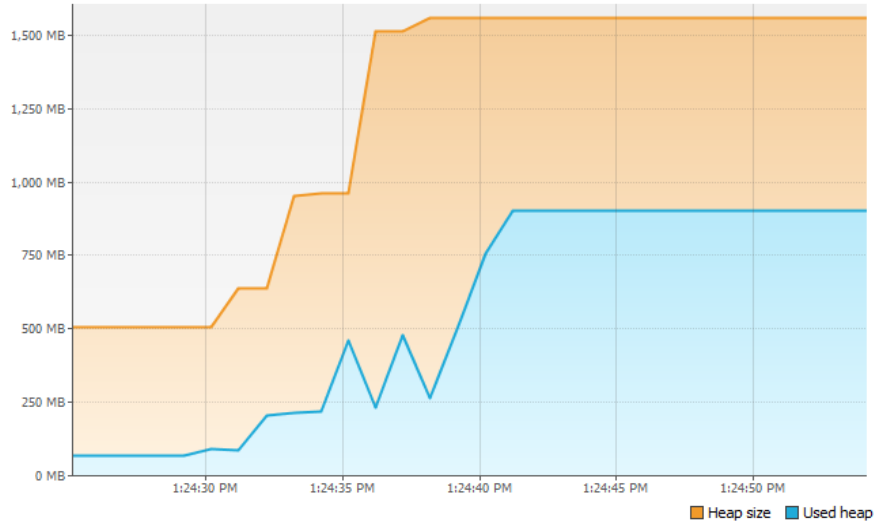
Bound = 6



Difference in Memory Usage

Size: 1,639,448,576 B
Max: 8,575,254,528 B

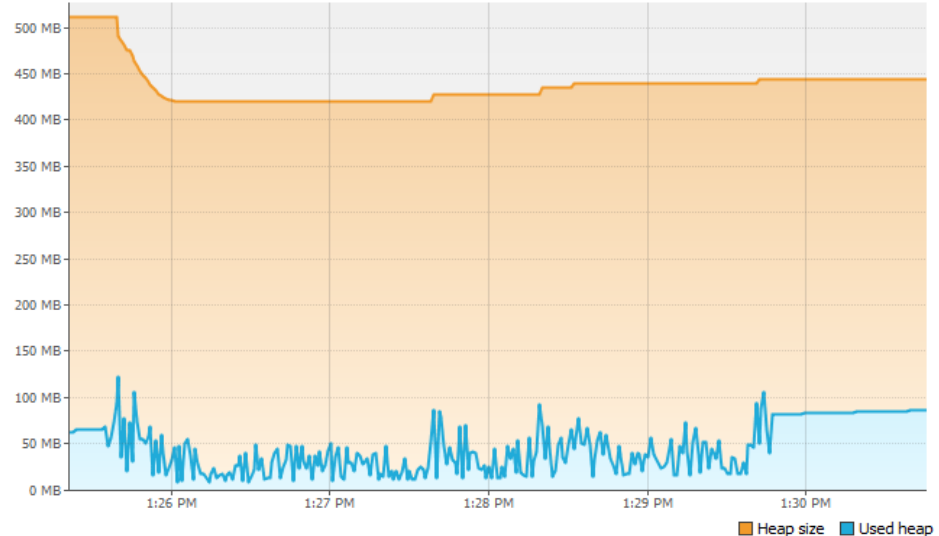
Used: 951,469,088 B



Sequential A* algorithm

Size: 467,664,896 B
Max: 8,575,254,528 B

Used: 91,680,248 B

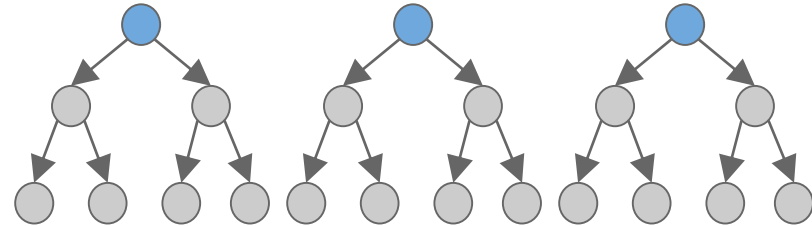
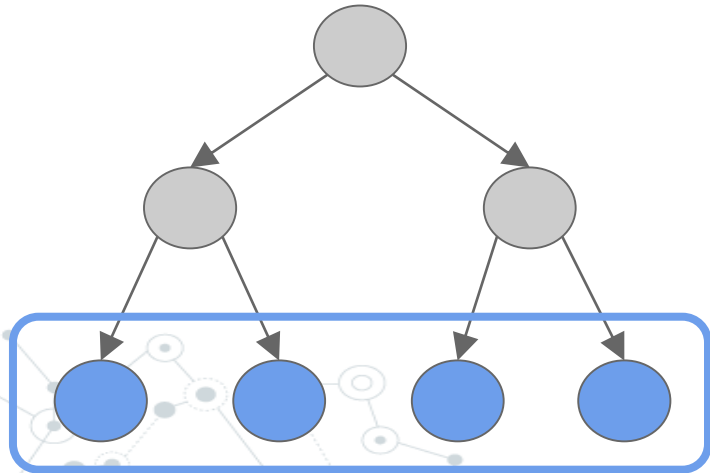


IDA* algorithm



Parallel Iterative Deepening A*

2 phase process

- Perform a depth limited breadth first search to generate a queue of states
- Perform IDA* on each state in the queue



Parallel multithreaded IDA* heuristic search
Mahafzah, Basel A.

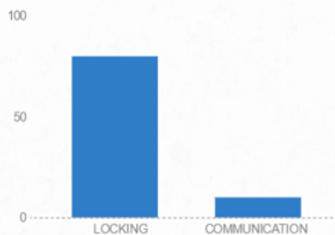


IDA* and Parallel IDA*

Demo

TIME SPENT

Centralised



MEMORY

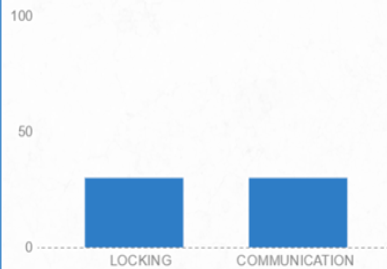


SPEEDUP



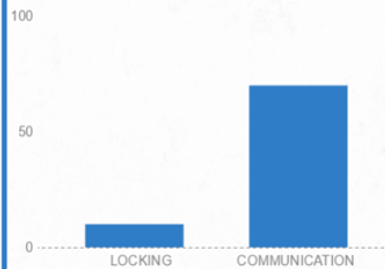
4.2

Blackboard



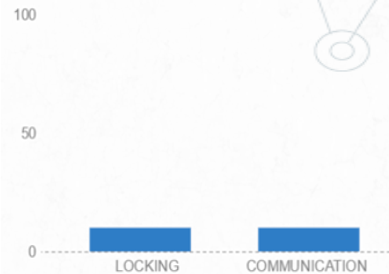
4.7

Hash Distributed



4.0

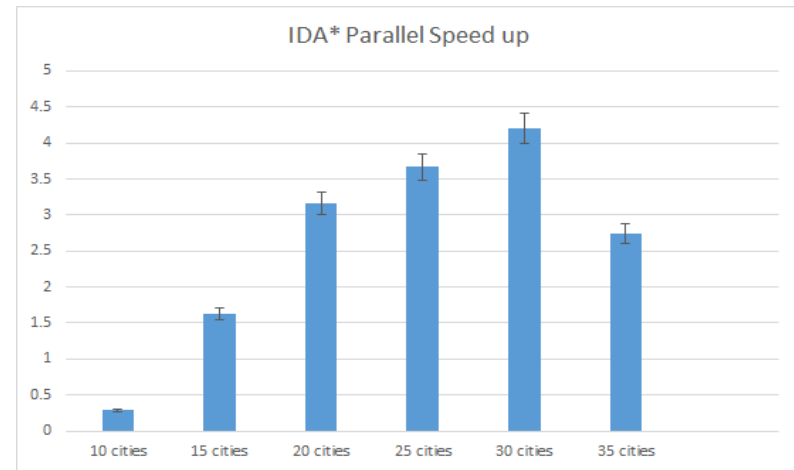
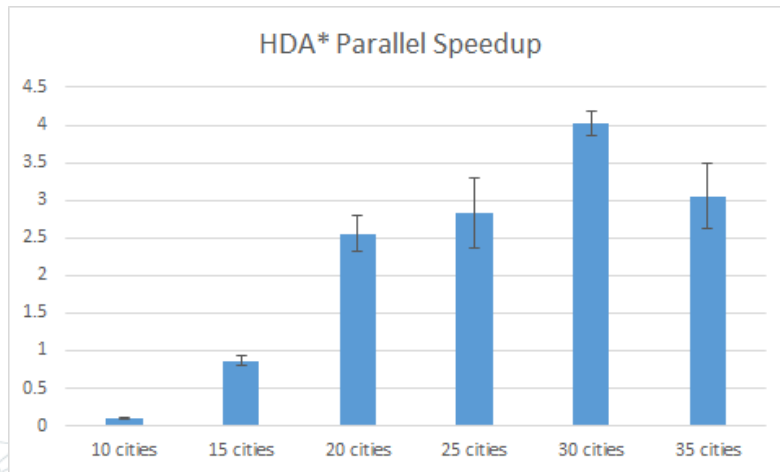
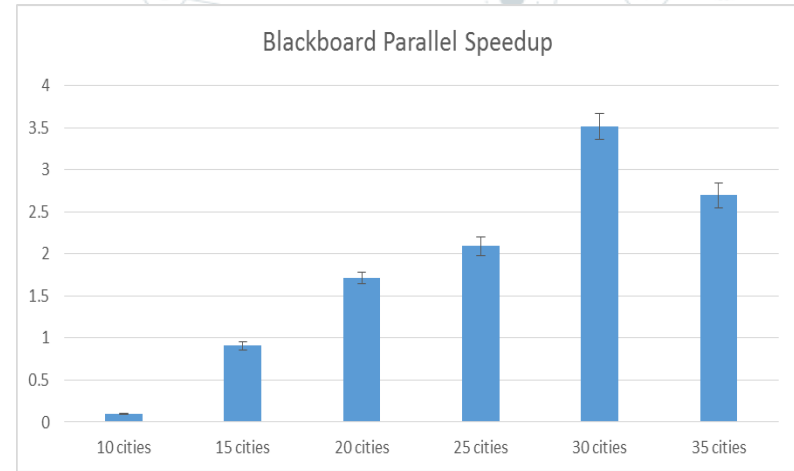
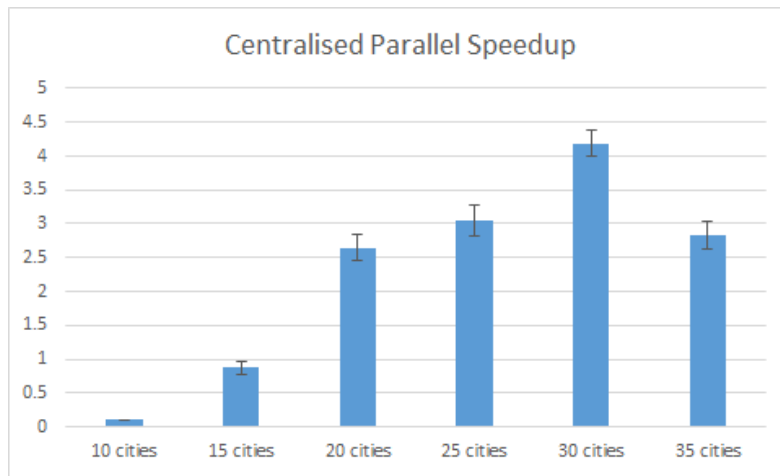
Parallel IDA*



4.2

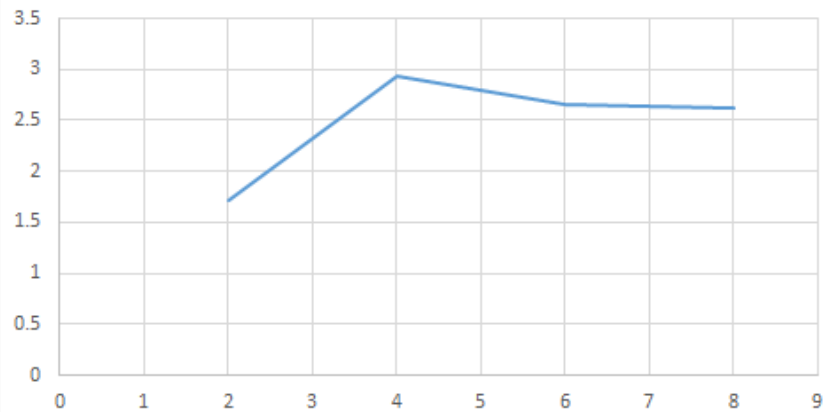
(Compared to sequential IDA*)

4 cores, 8 logical processors, hyperthreading, 27 city problem

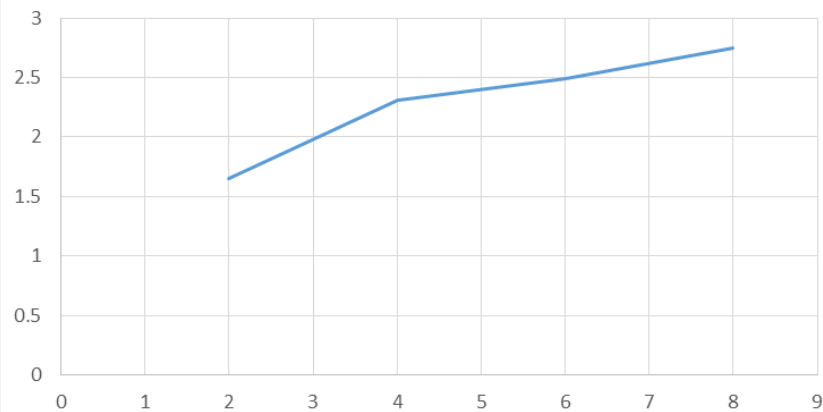


4 cores, 8 logical processors, hyperthreading

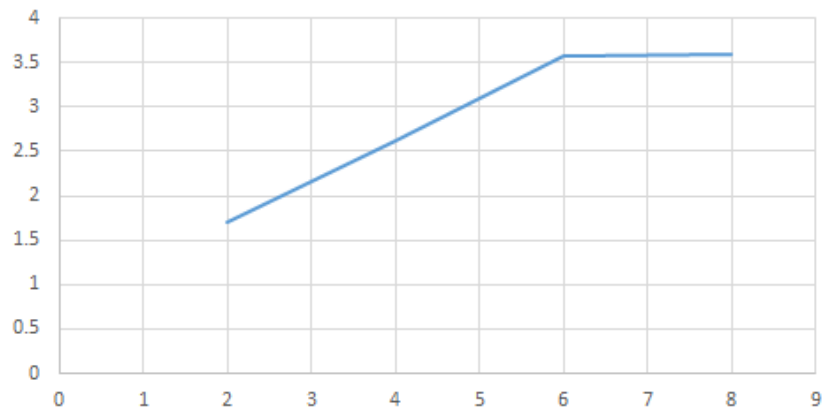
Speedup vs Processors - Centralised



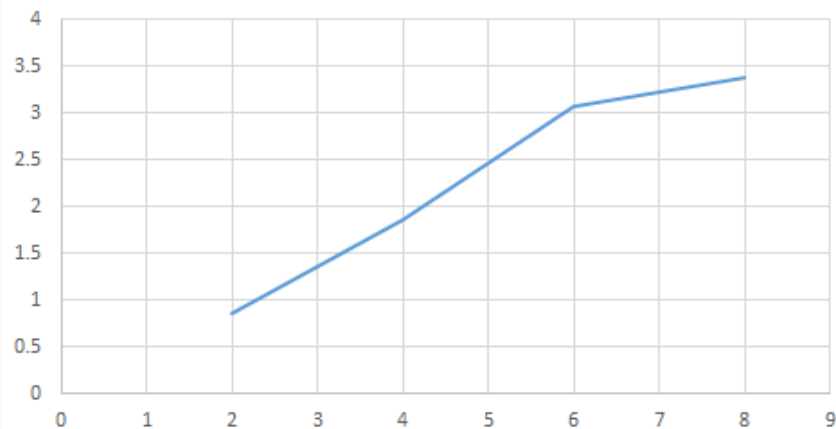
Speedup vs Processors - Blackboard



Speedup vs Processors - HDA*



Speedup vs Processor - Parallel IDA*



Future work

- ◎ Try out larger problem (50+ cities)
- ◎ Implement distributed termination algorithm for HDA* and Blackboard
- ◎ Write our report

Conclusion

- ◎ A* for TSP is slightly different than traditional A* for Path Finding
- ◎ Parallelisation can be either centralised or distributed
- ◎ The parallelisation overhead is either communication cost, synchronisation cost, or exploring extra nodes
- ◎ Parallel IDA* can be used to reduce memory usage
- ◎ The different parallel implementations scale differently over the number of cities and processors



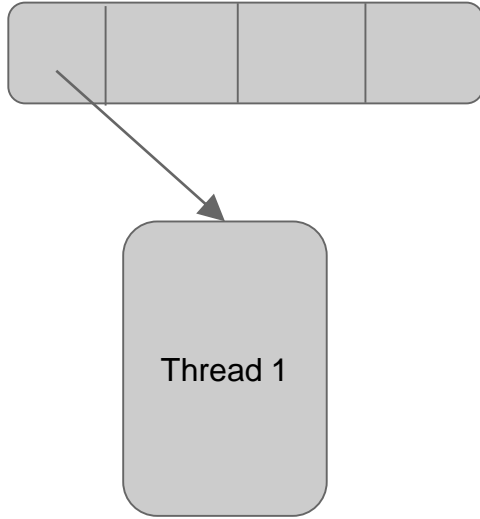
Thanks!

Any questions?

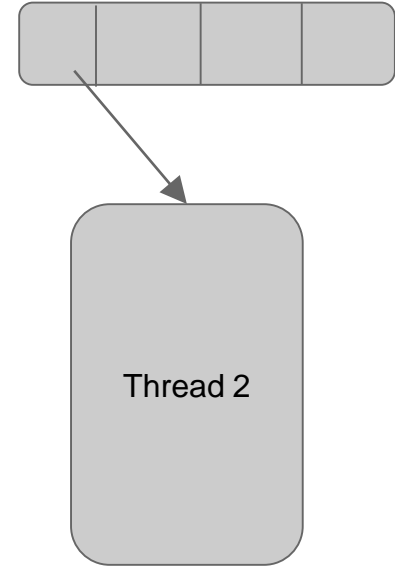
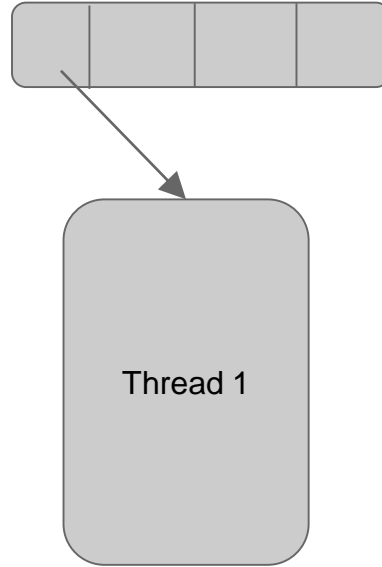


Why does parallel explore more nodes?

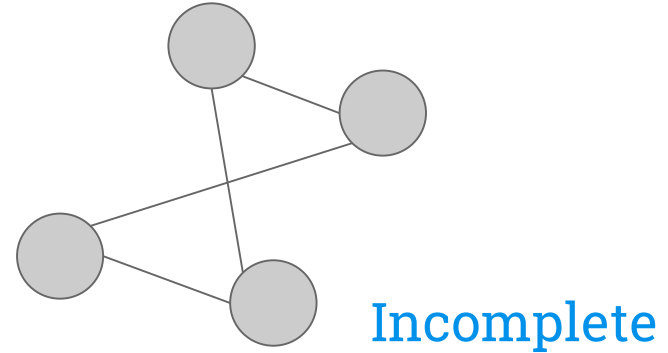
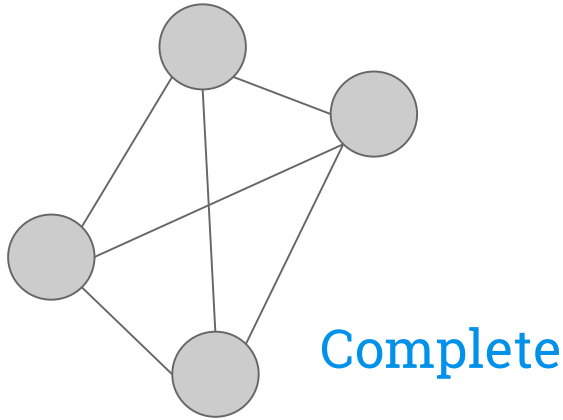
Sequential



Parallel

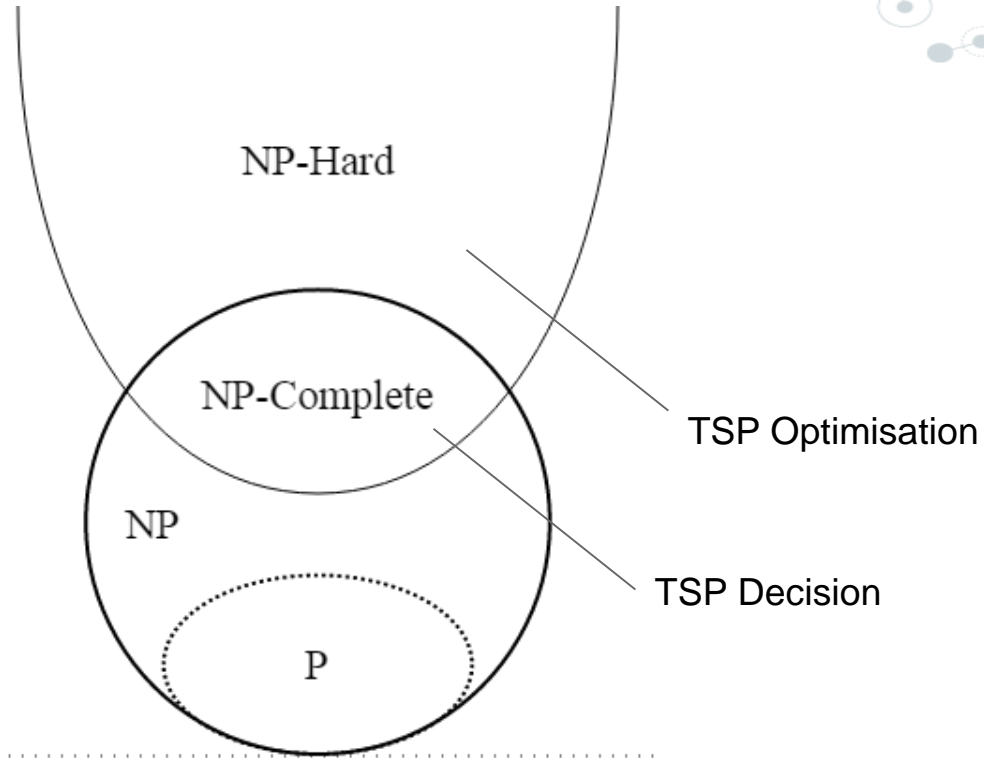


Complete Vs. Incomplete Graph



We could just add infinite costs between cities with no edges between them

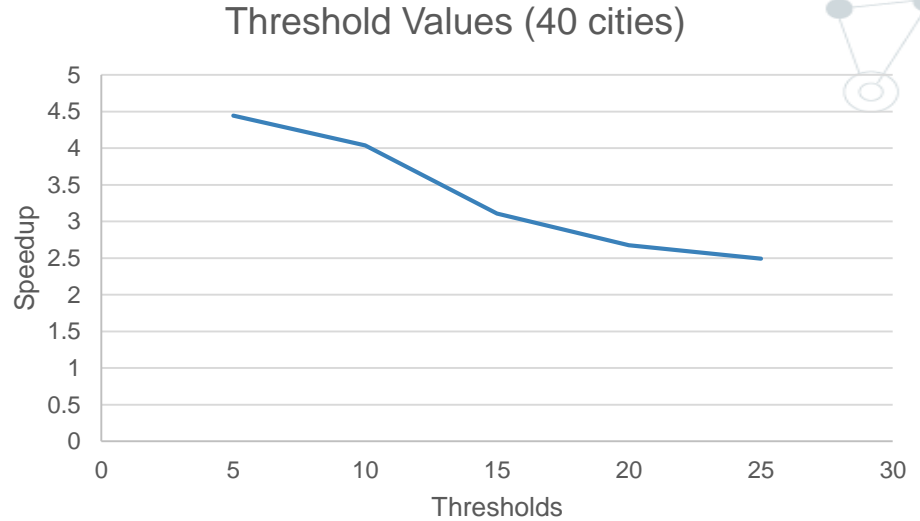
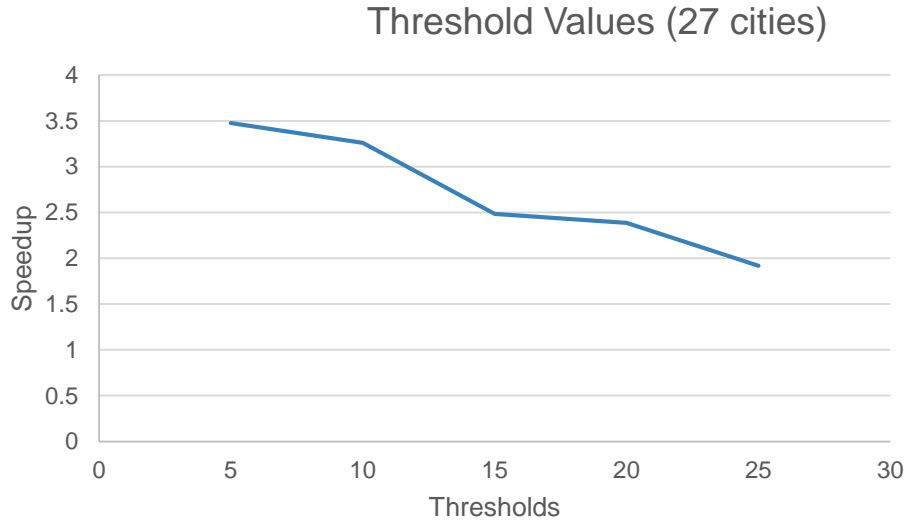
NP Complete Vs NP Hard



Other heuristic for A* algorithm

Heuristic	Description	Time Complexity
Greedy	Repeatedly pick the shortest edge	$O(n^2 \log_2(n))$
Insertion	Start with a subset of all cities, and insert the rest by some heuristic	$O(n^2)$
Christofides	Create a MST, MWM, and Euler Cycle	$O(n^3)$
MST	Create a MST	$O(n^2)$

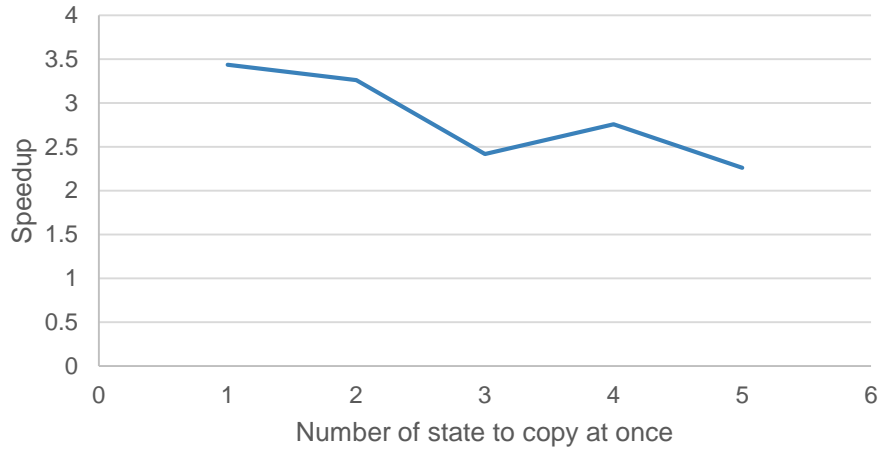
Blackboard A* Thresholds



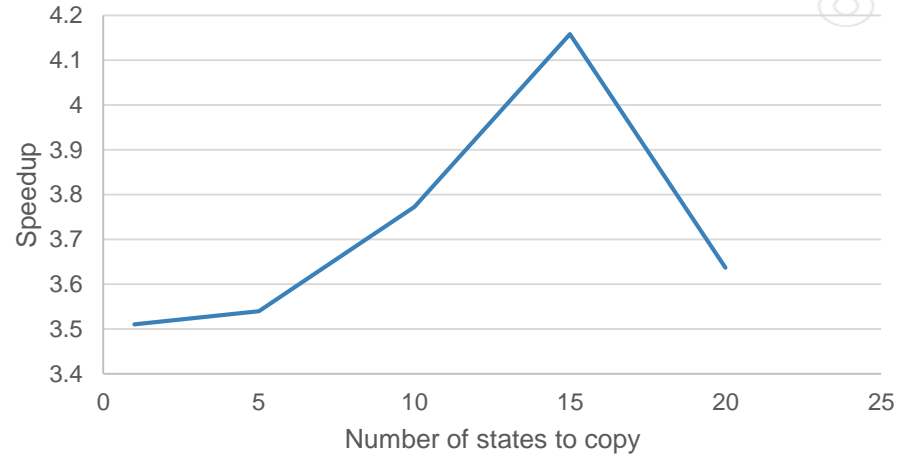
4 cores, 8 logical processors, hyperthreading

Blackboard A* Parameters

Number of states to copy (27 cities)

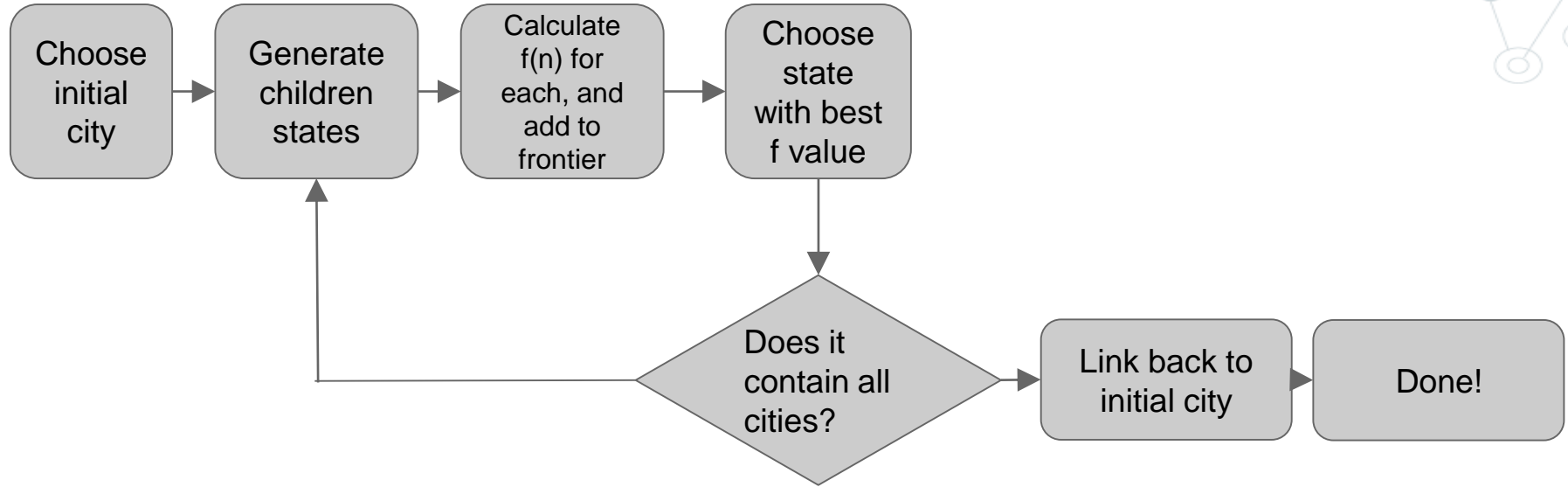


Number of states to copy (40 cities)



4 cores, 8 logical processors, hyperthreading

A* for TSP: Flowchart



Tree Search

