Parallel Computing (CS F422)
II Semester 2019-2020
Assignment 2
Plots and data analysis (P1)

The 2 tree search heuristics were run on different inputs and trends were observed,

Since the path chosen depends on the spawning of threads, which is random and may lead to varying time for the same input and thread. To obtain a general trend, multiple observations are taken for each input.

- **Input 1:**

2 6 13 5

7 3 1 12

11 9 10 8

14 15 0 4

Table 1 shows the execution times for the given input in seconds. The entries for 1 and 2 threads for puzzle_a are missing as the execution did not complete within 180 seconds, and time was assumed to be close to infinite in relative terms.

Number of threads	**1**	**2**	**4**	**8**	**16**	**32**	**64**
puzzle_a (single work queue)	-	-	39.21 100.58 46.90 46.13	29.91 34.53 34.57 6.75	14.19 13.05 15.11 4.99	8.79 8.22 0.57 0.64	24.55 7.47 2.08 12.85
puzzle_b (multiple work queues)	15.33 3.93 3.26 46.17	0.74 9.12 23.93 0.91	2.01 9.82 40.24 0.51	13.51 0.74 3.96 2.66	3.07 7.08 2.24 2.35	0.39 2.02 2.02 0.77	7.17 0.36 0.50 4.34

Table 1 (Execution time readings for input 1)

Figure 1 shows a plot for average execution time (of the given 4 readings) vs logarithm (base 2) of number of threads corresponding to puzzle_a and puzzle_b.

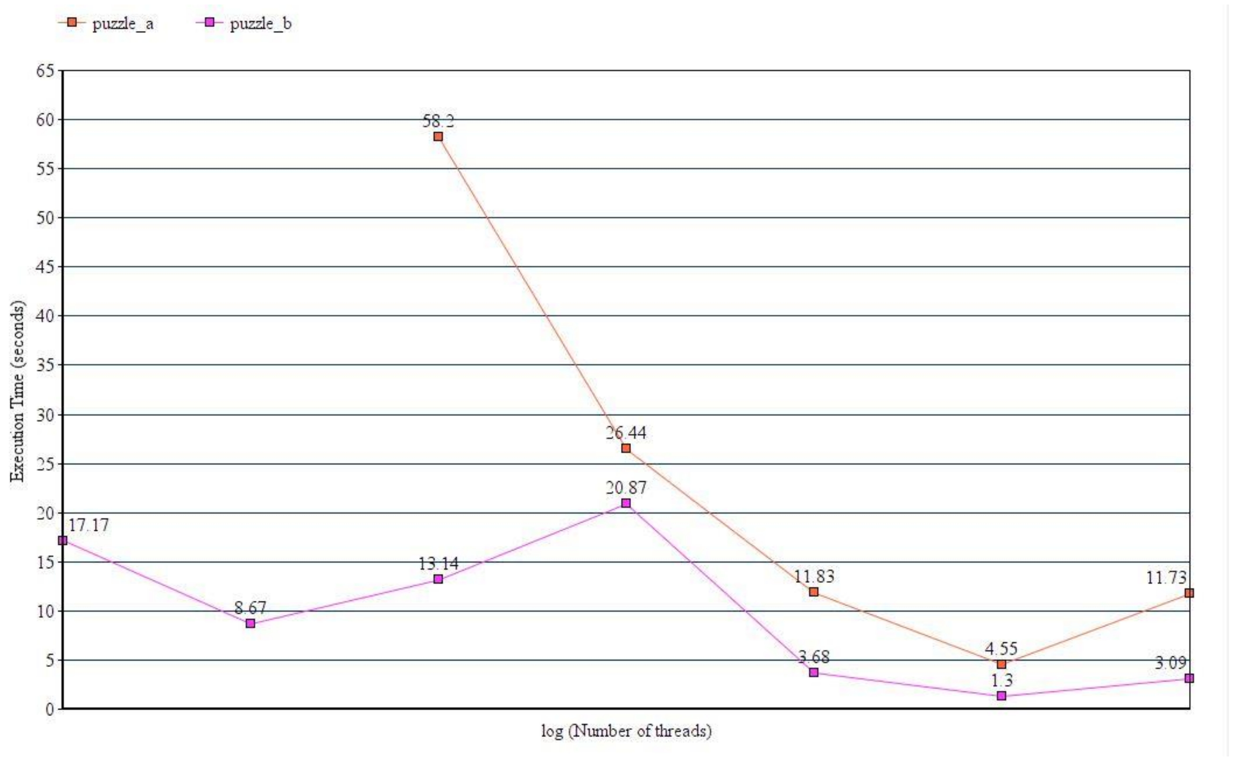

Figure 1 (Plot for execution time vs log of number of threads for input 1)

The theoretical speedup is calculated by using the equation,

$$speedup = \frac{T_{seq}}{T_{parallel}} = \frac{T_p}{T_1}$$

Table 2 consists of the speedup calculated using the execution time readings stated in Table 1.

Number of threads	**1**	**2**	**4**	**8**	**16**	**32**	**64**
puzzle_a (single work queue)	-	-	Close to infinite	Close to infinite	Close to infinite	Close to infinite	Close to infinite
puzzle_b (multiple work queues)	1.00	1.98	1.30	0.82	4.66	13.20	5.55

Table 2 (Calculated theoretical speedup, corresponding to input 1)

Figure 2 shows the plot of theoretical speedup vs logarithm (base 2) of number of threads.

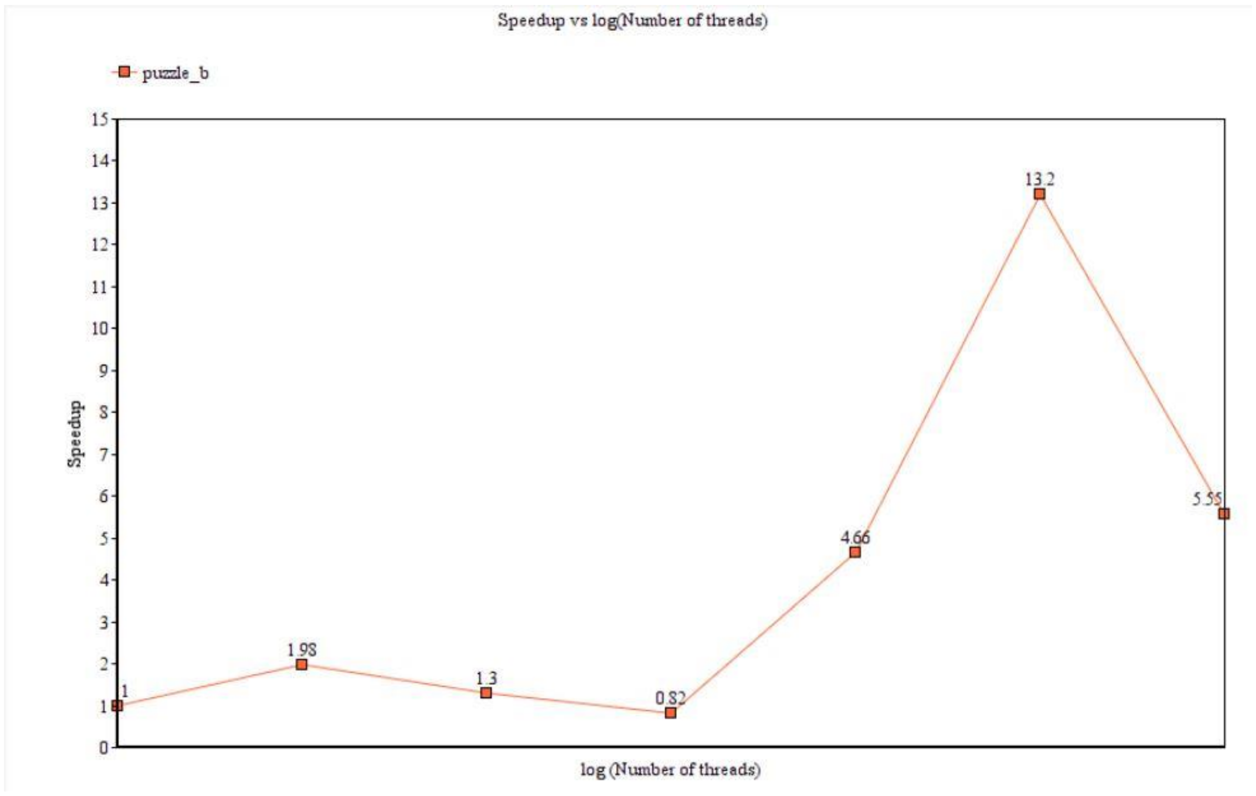

Figure 2 (Plot of speedup vs logarithm of number of threads corresponding to Input 1)

Observations:

- It is impossible to solve the 15 puzzle solver using a single thread even with the best first search heuristic.
- The speedup tends to be infinite in case of a single working queue, as it is difficult to solve this instance only with a single thread, this input situation lead us to visualize the essence of multithreading.
- For multiple working queue, we are able to solve the puzzle even with a single thread. This shows that the multiple working queues help us analyze multiple paths simultaneously and hence reduces the run-time.
- The speedup increases till 32 threads and drops down at 64 threads due to the overheads due to locks, cancel, joining and false cache misses while accessing heap and hash table.

➢ **Input 2:**

1 2 4 8

5 0 6 3

9 10 7 12

13 14 11 15

Table 3 shows the execution times for the given input in seconds.

Number of threads	**1**	**2**	**4**	**8**	**16**	**32**	**64**
puzzle_a (single work queue)	0.0013 0.0014 0.0019 0.0013	0.0024 0.0021 0.0031 0.0023	0.0045 0.0029 0.0074 0.0028	0.0036 0.0033 0.0041 0.0045	0.0043 0.0050 0.0052 0.0048	0.0068 0.0105 0.0084 0.0099	0.0142 0.0150 0.0144 0.0133
puzzle_b (multiple work queues)	0.0017 0.0031 0.0037 0.0018	0.0026 0.0040 0.0027 0.0029	0.0028 0.0030 0.0029 0.0030	0.0034 0.0038 0.0038 0.0037	0.0056 0.0054 0.0050 0.0050	0.0088 0.0079 0.0078 0.0091	0.0138 0.0132 0.0136 0.0141

Table 3 (Execution time readings for input 2)

Figure 3 shows a plot for average execution time (of the given 4 readings) vs logarithm (base 2) of number of threads corresponding to puzzle_a and puzzle_b.

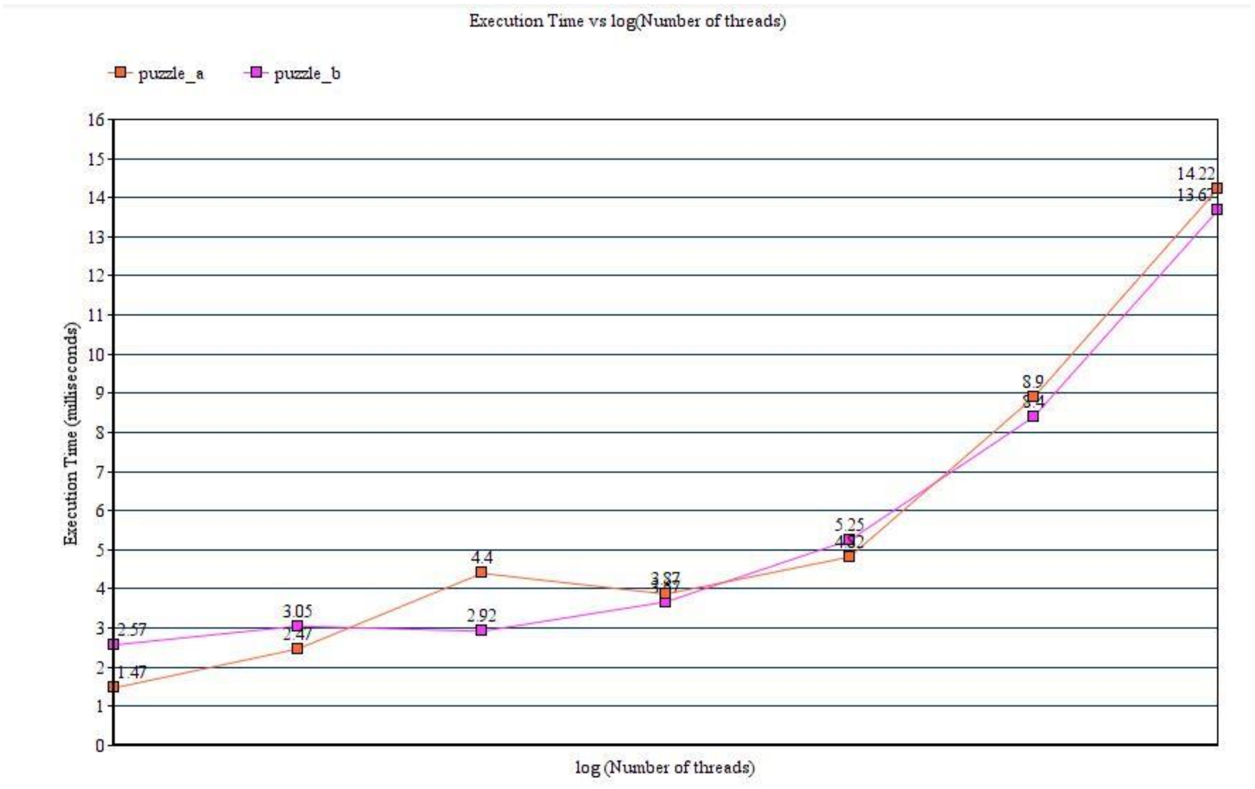

Figure 3 (Plot for execution time vs log of number of threads for input 2)

Table 4 consists of the speedup calculated using the execution time readings stated in Table 3.

Number of threads	1	2	4	8	16	32	64
puzzle_a (single work queue)	1.00	0.59	0.34	0.38	0.30	0.16	0.10
puzzle_b (multiple work queues)	1.00	0.84	0.88	0.70	0.49	0.30	0.19

Table 4 (Calculated theoretical speedup, corresponding to input 2)

Figure 4 shows the plot of theoretical speedup vs logarithm (base 2) of number of threads.

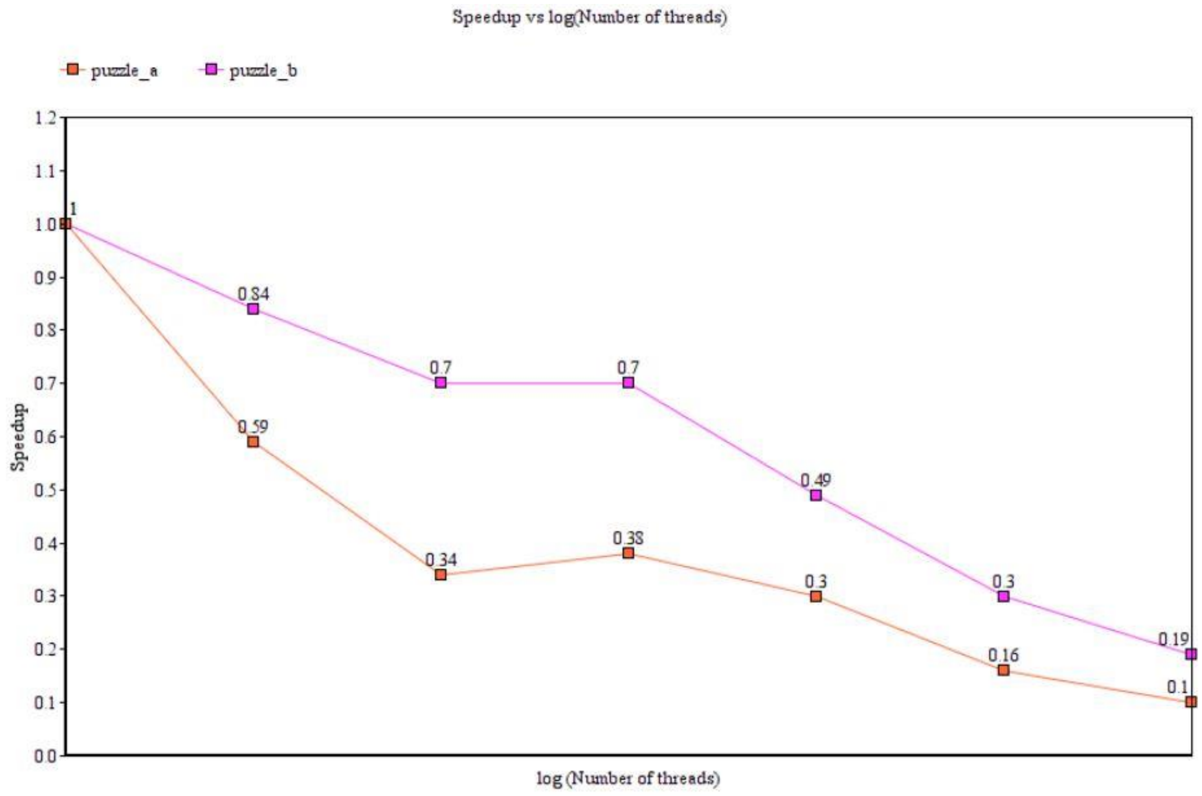

Figure 4 (Plot of speedup vs logarithm of number of threads corresponding to Input 2)

Observations:

- For an easier puzzle, which needs less number of moves, even though the program runs in order of milliseconds, the use of more number of threads may increase the execution time due to the overheads of threading specified above.
- For an easier puzzle, use of single work queue may also prove to be better than multiple work queues.

- **Input 3:**

9 1 5 8

0 12 13 10

6 4 14 2

3 15 7 11

Table 5 shows the execution times for the given input in seconds.

Number of threads	**1**	**2**	**4**	**8**	**16**	**32**	**64**
puzzle_a (single work queue)	2.22 2.14 2.14 2.15	0.21 0.75 0.22 4.39	1.02 0.01 0.09 0.05	0.05 0.03 0.13 0.43	0.21 0.02 0.24 11.52	0.05 0.02 0.50 0.23	5.43 2.78 1.81 0.03
puzzle_b (multiple work queues)	1.94 1.68 0.56 2.58	0.50 7.19 0.03 1.10	0.33 1.13 0.08 0.10	0.10 0.61 0.48 0.14	0.02 0.03 0.47 0.22	0.21 0.04 1.57 0.04	5.15 0.25 10.94 0.17

Table 5 (Execution time readings for input 3)

Figure 5 shows a plot for average execution time (of the given 4 readings) vs logarithm (base 2) of number of threads corresponding to puzzle_a and puzzle_b.

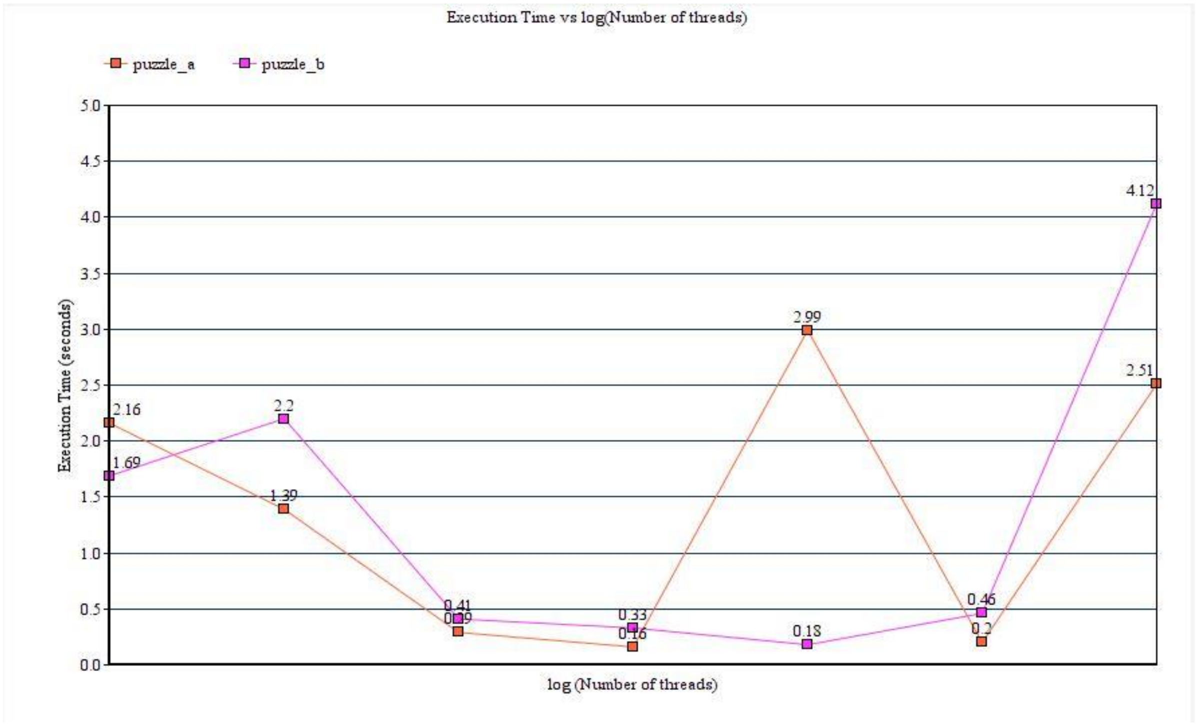

Figure 5 (Plot for execution time vs log of number of threads for input 3)

Table 6 consists of the speedup calculated using the execution time readings stated in Table 5.

Number of threads	1	2	4	8	16	32	64
puzzle_a (single work queue)	1.00	1.55	7.44	13.50	0.72	10.80	0.86
puzzle_b (multiple work queues)	1.00	0.76	4.12	5.12	9.38	3.67	0.41

Table 6 (Calculated theoretical speedup, corresponding to input 3)

Figure 6 shows the plot of theoretical speedup vs logarithm (base 2) of number of threads.

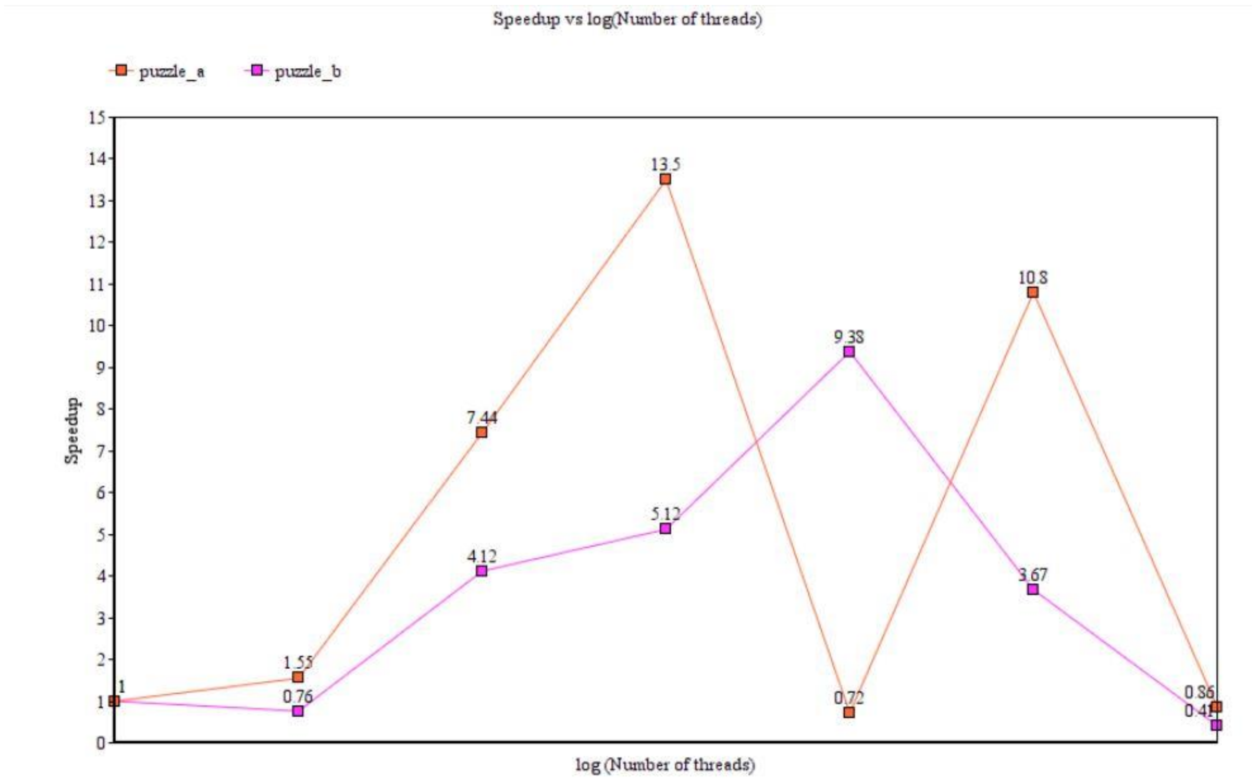

Figure 6 (Plot of speedup vs logarithm of number of threads corresponding to Input 3)

Observations:

- For multiple working queues, the speedup first increases with the number of threads, then decreases due to the threading overheads and false sharing of heap and hash table.
- On an average, the execution time for multiple working queues is less than the execution time for a single working queue. Hence, the efficiency of the algorithm is more if we use multiple working queues.
- The sudden drop in execution time for 16 threads in a puzzle_a model may be by chance as no theoretic reasons support this drop and rise behavior.

➢ Input 4:

9 5 6 14

7 2 13 12

4 11 1 8

0 15 10 3

Table 7 shows the execution times for the given input in seconds. The entries for 1 thread for puzzle_a is missing as the execution did not complete within 180 seconds, and time was assumed to be close to infinite in relative terms.

Number of threads	**1**	**2**	**4**	**8**	**16**	**32**	**64**
puzzle_a (single work queue)	-	6.49 0.51 36.86 136.44	0.42 0.57 56.38 41.12	3.07 0.99 0.22 32.14	0.10 0.61 15.36 13.09	5.10 1.46 0.10 11.24	0.89 9.59 0.78 0.05
puzzle_b (multiple work queues)	1.60 9.55 12.97 18.57	4.65 7.83 14.56 1.71	4.27 1.25 0.35 3.54	3.21 2.05 10.49 1.54	1.76 3.74 0.19 0.70	0.22 0.34 1.65 0.38	5.73 2.98 0.73 2.59

Table 7 (Execution time readings for input 4)

Figure 7 shows a plot for average execution time (of the given 4 readings) vs logarithm (base 2) of number of threads corresponding to puzzle_a and puzzle_b.

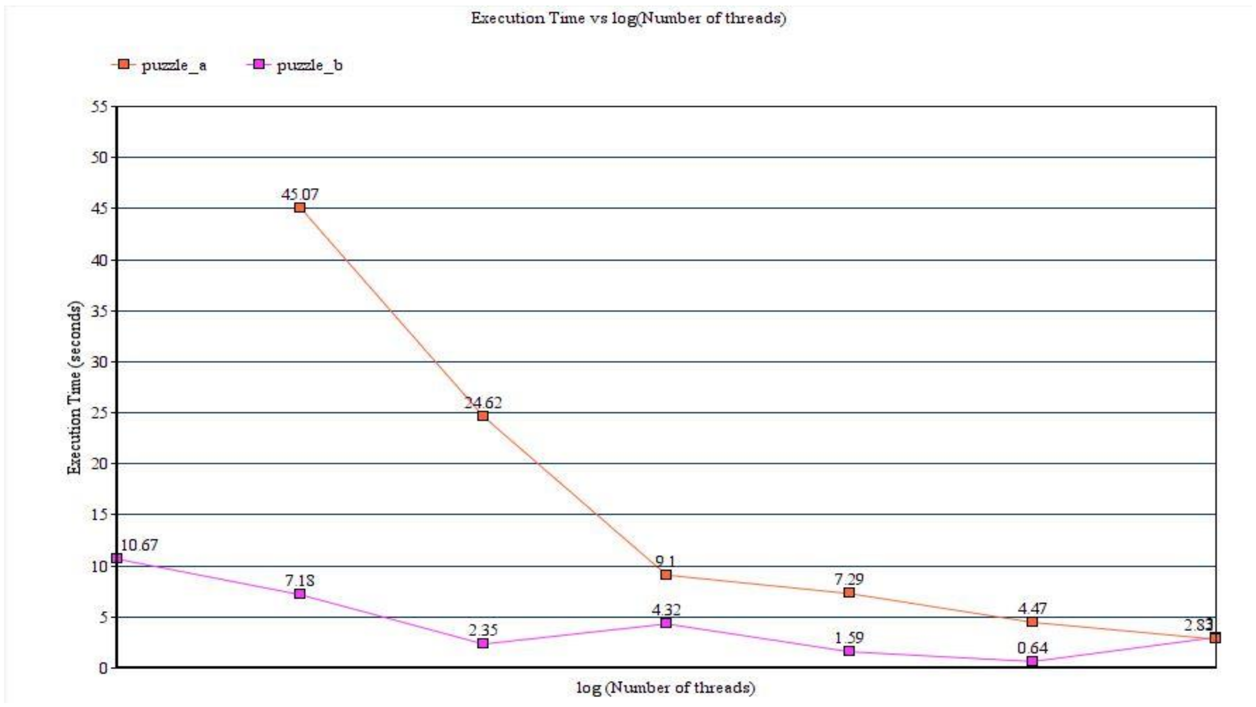

Figure 7 (Plot for execution time vs log of number of threads for input 4)

Table 8 consists of the speedup calculated using the execution time readings stated in Table 7.

Number of threads	**1**	**2**	**4**	**8**	**16**	**32**	**64**
puzzle_a (single work queue)	-	Close to infinite	Close to infinite	Close to infinite	Close to infinite	Close to infinite	Close to infinite
puzzle_b (multiple work queues)	1.00	1.48	4.54	2.47	6.71	16.67	3.77

Table 8 (Calculated theoretical speedup, corresponding to input 4)

Figure 8 shows the plot of theoretical speedup vs logarithm (base 2) of number of threads.

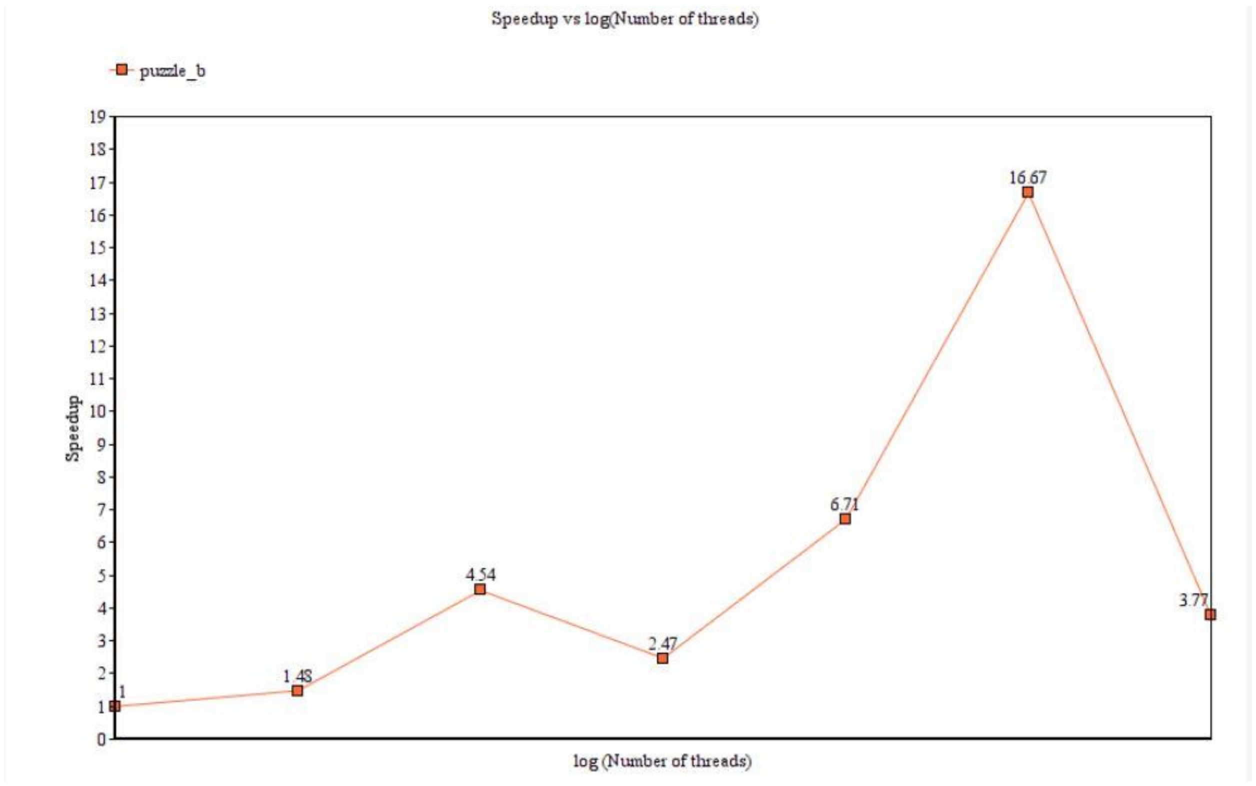

Figure 8 (Plot of speedup vs logarithm of number of threads corresponding to Input 3)

Observations:

- It is impossible to solve the 15 puzzle solver using a single thread even with the best first search heuristic.
- The speedup tends to be infinite in case of a single working queue, as it is difficult to solve this instance only with a single thread, this input situation lead us to visualize the essence of multithreading.
- For multiple working queue, we are able to solve the puzzle even with a single thread. This shows that the multiple working queues help us analyze multiple paths simultaneously and hence reduces the run-time.
- The speedup increases till 32 threads and drops down at 64 threads due to the overheads due to locks, cancel, joining and false cache misses while accessing heap and hash table.