Test data for maze of size 7x7 with generation threshold = 10000
(column represents genome length, row represents number of threads)

	20	40	80	100	120	140	160	180	200
4	0.118s	0.137s	0.154s	0.232s	0.439s	0.491s	0.376s	0.628s	0.780s
5	0.144s	0.148s	0.198s	0.214s	0.396s	0.380s	0.914s	1.201s	0.848s
6	0.109s	0.169s	0.235s	0.708s	0.518s	1.148s	0.832s	0.672s	0.671s
7	0.131s	0.172s	0.235s	0.369s	0.540s	1.026s	1.306s	0.931s	1.080s
8	0.150s	0.187s	0.537s	0.922	0.695s	0.698s	0.915s	0.677s	0.875s

Test data for maze of size 21x21 with generation threshold = 10000
(column represents genome length, row represents number of threads)

	20	40	80	100	120	140	160	180	200
4	0.111s	0.137s	0.153s	0.171s	0.732s	0.956s	0.534s	0.512s	0.750s
5	0.139s	0.173s	0.297s	0.234s	0.268s	1.225s	0.953s	1.229s	1.216s
6	0.126s	0.145s	0.484s	0.816s	0.812s	0.545s	0.676s	0.710s	0.623s
7	0.130s	0.176s	0.257s	0.414s	1.078s	1.102s	0.828s	1.081s	1.040s
8	0.140s	0.549s	0.663s	0.824s	0.737s	1.099s	1.086s	1.176s	0.842s

Some Bugs Found during Testing:

When testing what could happen when I adjust the ratio of number of mixer and mutator threads, sometimes “deadlock” happens in some subset of worker threads and the program never stops as the main thread waits for joining all worker threads. But this happens most frequently when the number of mixer threads is 1. I realized that sometimes, the mutator threads will not be woken up after it calls “listen” on the offspring queue because it is empty and no further offspring genome is pushed in by mixer. This is because when some mutator threads figure that we need to stop, it sets the terminating flag to be true and the mixer terminates before another mutator thread starts to listen on the offspring queue. This causes the offspring queue unable to wake up the blocking mutator threads that should’ve been woken up and aborted because the job is finished.

My solution is to use another flag in the queue data structure to indicate that the queue is finished. Another method called “finalize” is introduced so that when one thread sets the

terminating flag to true, it also calls to method to wake up all threads currently still waiting on the queue. And when these sleeping threads are woken up, they would just return without retrieving data from the queue.

Conclusion and Observation:

From the tests above, unsurprisingly, when the genome length gets longer, the time it takes to finish is getting longer. But what was out of expectation is that when the number of threads goes from 4 which is the number of cores my laptop got to two times the number, the time it takes to finish is increasing. My guess is that maybe more than 4 threads is redundant and the overhead of spawning and waiting for all to join and the overhead of context-switching is pretty big compared to the amount of work each thread is undertaking. I chose ¼ of the threads to be mixers threads as they do less work and ¾ to be mutators. Maybe this configuration is not optimal for large cases but it works the best for 4 to 8 threads with genome length between 20 to 200. Of course, when the dimension of the maze gets bigger, the runtime becomes longer.