Travelling Salesman Problem

for

Evolutionary Computation

Prepared by William Reid (a1215621), Alec Bellati (a1608934),
Samantha Peachey (a1192722) & Matthew Hart (a1193380)

School of Computer Science,
The University of Adelaide

August 31, 2014

1 Introduction

After much discussion amongst the project team, three algorithms were formulated. But it was evident that some of them relied on some type of percentage values. Given we had little experience in what these percentages would be or could mean, we decided on running the operators and mutators multiple times on a simple data set to ascertain how they performed. The steps taken to acquire this data are as follows:

1. Each run was performed 10 times, with a population of 50 and for 10,000 generations.
2. For each run, the mutators/operator would begin at 0% (chance of being selected). If the mutators/operator was not selected, no operation would occur for that cycle.
3. This chance of being selected would increase by 5% after set of 10 runs
4. Once the probability of being selected had reached 100%, the tests began on the next operator/mutator.
5. Each mutators/operator was tested
6. Steps 1-5 were repeated for each selection method.

This provided us with 8 graphs, with the most significant Edge Recombination shown below. The remaining graphs have been ommited from this report but can be obtained from the team if required.

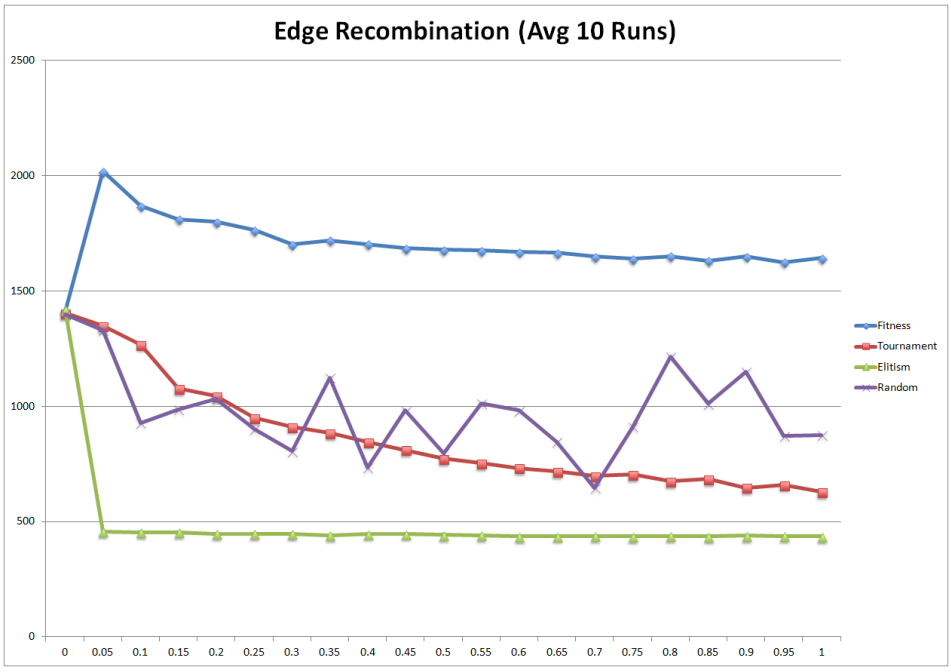

Figure 1: Edge Recombination solution accuracy at percentage levels

2 Algorithms

The first two algorithms are similar to those found in the recommended text for this course (GA and GP). The team made some minor modifications to better reflect the results from the TSP.

The three algorithms are:

1. Genetic Algorithm (GA)
2. Genetic Programming (GP)
3. Statistical Advantage

*NOTE: For testing, each algorithm was run at least 10 times for each population/generation pair. The log files associated with these testing runs can be found in the main project files, split into appropriate files and folders. Due to the format of the log files, please ensure you fully expand "column A" when looking at the data to capture all related information.

2.1 A Modification to GA

The Genetic Algorithm (GA) has the potential to perform mutation and cross-over operations sequentially, based on specific probabilities. The steps below explain the process for our use of the GA.

For each generation:

1. Select the best of the current population, duplicate it and set it aside.
2. Choose two individuals at random.
3. While the current population size is less than the maximum population size, do the following
4. Perform a cross-over, based on probability Pc (creates two new children).
5. Perform a mutation (on the same set), based on probability Pm.
6. Add the children back into the population and continue from step 3.
7. If the current population size is equal to the maximum population size, perform a selection method, chosen based on probability Ps.
8. Start the next generation at step 1 until maximum number of generations reached.

2.2 A Modification to GP

Unlike the GA, the Genetic Programming (GP) approach has the potential to perform mutation OR cross-over methods seqentially, based on specific probabilities (but not both). The steps below explain the process for our use of the GP.

For each generation:

1. Select the best of the current population, duplicate it and set it aside.
2. Choose two individuals at random.
3. While the current population size is less than the maximum population size, do the following
4. Given a probability *Pc* perform either a cross-over OR *(1-Pc)* - the probability that a mutation would be chosen - for this entire generation. Either:
    - Perform a cross-over on the two individuals (creates two new children).
    - Duplicate a single individual from the two randomly chosen and perform a mutation (creates a single new child).
5. Add the children back into the population and continue from step 3.
6. If the current population size is equal to the maximum population size, perform a selection method, based on probability *Ps*.
7. Start the next generation at step 1 until maximum number of generations reached.

Both these algorithms obtain good approximations, with the GA traditionally finding the solution faster. However since the GP has the possibility to retain a wide range solutions, it generally has marginally better results.

2.3 Statistical Advantage

Using the statistics mentioned in the introduction, the team took the best aspects of algorithms 1 & 2 and forumalted a new approach using very specific statistics.

For each generation:

1. Select the best of the current population, duplicate it and set it aside.
2. Clone the current population to form a "mutants" population
3. Clone the current population to form a "cross-over" population
4. For every individual in the "mutants" population:
    - Perform inversion 50% of the time
    - Perform insert or swap 45% of the time
    - Perform scramble the remaining 5%
    - Add the mutated children back into the population
5. Until the cross-over population is empty, choose two individuals at random and remove them from the cross-over population. Then do:
    - Edge Recombination 40% of the time
    - Order Crossover 30% of the time
    - PMX Crossover 20% of the time
    - Cycle Crossover the remaining 10%
    - Add the children back into the population
6. Perform a selection method:
    - Elitism and Tournment 95% of the time
    - Fitness-Proportional the remaining 5% of the time
7. Start the next generation at step 1 until maximum number of generations reached.

While this method is moderately slower then algorithm 1 and 2, with smaller population sizes (20 or less) this algorithm can generate solutions that are 15% more accurate. With population sizes more than 20, solution accuracy ranges from 1 - 0.1%, but what we observed was that this algorithm took considerably less generations to find an accurate solution.

3 Issues

3.1 Infinite Looping on Small Populations

With smaller populations there is the potential for an infinite loop in each of the algorithms. It is unknown where the issue occurs as it is very intermittent but the most likely cause is that the tours very quickly become equal. This is not through solution duplication but simply each of them individually finding the same solution through our TSP methods.

This has since been rectified but not extensively tested and thus deserves mentioning in this report. If duplicates are detected they are mutated to form new solutions. However the affect on run-time has also not been extensively studied.

3.2 Fitness-Proportional

During our test runs it was discovered that the Fitness-Proportional (FP) selection method was performing a maximisation rather than a minimisation. This greatly affected our results and the choice of its use in our algorithms. This issue has since been rectified but there was not sufficient time to re-run the tests.

3.3 Inver-Over Algorithm

The Inver-Over operator discussed in the supplied paper produced some of the more accurate solutions. Our implementation gives good results but indications show results could be improved. Further analysis of its implementation would be required to rectify any issues found, but at this late stage its results are accurate enough for us to publish in this report.

As a side note, Inver-Over also appears to be the slowest algorithm from the three implemented and this factor should be considered when looking at the accuracy of solutions.

3.4 Pr2392

This data set is the largest of those supplied and took 7 hours to run for a population size 50 at 10,000 generations. While we produced some data, only small portions are available and were only run once for each population/generation pair. More time spent in code review may produce faster methods and algorithms and would also allow us to perform additional tests.

4 Results

A large portion of data has been left off this report for simplicity and succinctness. The log files containing this data can be found in the relevant folders of this project. The following tables reference population sizes of 50 and 100. Tables using population sizes of 10 and 20 can be found in the Appendix.

4.1 Algorithm 3 - Statistical Advantage

Graph	Generations	Pop. (50)	Std. Dev.	Pop. (100)	Std. Dev	Optimal
eil51	5000	441.78	7.73	444.69	5.44	
eil51	10000	441.56	4.29	438.69	2.83	426
eil51	20000	439.99	5.07	438.42	4.44	
eil76	5000	573.48	6.86	572.25	8.14	
eil76	10000	569.29	8.42	568.77	7.24	538
eil76	20000	565.62	5.12	564.98	10.99	
eil101	5000	678.51	8.02	674.68	7.37	
eil101	10000	674.14	9.97	667.60	8.02	629
eil101	20000	671.94	9.49	665.15	5.32	
st70	5000	697.96	15.55	696.62	11.08	
st70	10000	709.53	12.97	699.76	10.56	675
st70	20000	708.59	18.43	694.49	13.24	
lin105	5000	15 427.99	459.76	15 104.01	398.54	
lin105	10000	15 241.1	296.81	15 168.46	304.11	14379
lin105	20000	15 291.38	222.79	15 244.23	300.12	
kroA100	5000	22 518.89	500.06	22 152.25	201.86	
kroA100	10000	22 676.03	545.22	22 017.07	375.71	21282
kroA100	20000	22 003.82	443.16	22 634.07	655.46	
kroC100	5000	22 710.04	776.10	22 072.13	698.07	
kroC100	10000	21 903.62	649.14	22 204.83	404.13	20749
kroC100	20000	21 731.14	352.56	21 994.39	496.11	
kroD100	5000	22 644.56	557.92	22 327.91	483.54	
kroD100	10000	22 473.66	477.63	22 445.25	324.32	21294
kroD100	20000	22 721.88	570.16	22 347.49	545.79	
pcb442	5000	96 239.32	3 048.54	77 051.78	1 536.84	
pcb442	10000	72 984.8	1 590.48	60 730.04	1 093.8	50788
pcb442	20000	61606.66	645.55	56 607.36	562.46	
pr2392	10000	3 056 757.6	N/A	2 494 563.1	N/A	378032
pr2392	20000	2 005 398.4	N/A	1 523 738.9	N/A	

Figure 2: Algorithm 3 average tour length for each population/generation pair, including standard deviation

4.2 Inver-Over

Graph	Generations	Pop. (50)	Std. Dev.	Pop. (100)	Std. Dev	Optimal
eil51	5000	466.33	9.38	461.85	11.42	
eil51	10000	458.41	10.36	463.92	15.12	426
eil51	20000	466.30	8.25	456.76	8.17	
eil76	5000	647.77	28.82	609.49	11.67	
eil76	10000	613.83	14.53	599.29	13.11	538
eil76	20000	603.20	18.17	589.13	N/A	
eil101	5000	873.54	17.57	760.95	22.29	
eil101	10000	755.03	11.91	714.21	21.75	629
eil101	20000	717.43	14.20	N/A	N/A	
st70	5000	796.79	18.76	770.93	30.10	
st70	10000	753.34	21.51	735.24	32.17	675
st70	20000	732.38	23.41	N/A	N/A	
lin105	5000	23 755.57	1 012.01	18 265.84	626.77	
lin105	10000	18 602.81	904.40	16 560.07	545.29	14379
lin105	20000	16 651.62	731.76	N/A	N/A	
kroA100	5000	31 888.93	1 500.20	26 880.80	744.78	
kroA100	10000	26 358.19	796.65	24 129.03	664.03	21282
kroA100	20000	24 274.44	858.13	23 440.57	602.10	
kroC100	5000	31 886.57	1 324.18	26 454.78	1 217.01	
kroC100	10000	25 852.20	476.38	23 176.62	594.62	20749
kroC100	20000	23 248.06	715.89	N/A	N/A	
kroD100	5000	31 393.99	580.22	25 857.97	558.29	
kroD100	10000	26 403.75	943.46	22 909.55	347.01	21294
kroD100	20000	23 975.81	506.93	N/A	N/A	
pcb442	5000	270 908.68	6 124.81	202 287.48	3 531.54	
pcb442	10000	199 139.10	5 524.34	145 609.79	3 218.59	50788
pcb442	20000	144 937.47	4 608.89	108 035.10	2 966.01	

Figure 3: Inver-Over algorithm average tour length for each population/generation pair, including standard deviation

5 Conclusion

Given these results we found that while our algorithm takes slightly longer to complete than others, it does produce more accurate results and generally obtains them at a lower generation. Improvements to the code may render faster run times and more accurate results. Many of our methods changed after we ran the statistical analysis of the mutators and operators and this would change how our algorithms perform.

The Inver-Over algorithm produces its best results with larger populations and even larger generations. Issues in our implementation may have lead to the notion that our algorithms performed marginally better than the results produced with Inver-Over.

In addition, the algorithms above have the potential to produce tours that are similar or even the same. Further research must be done in order to combat this problem in a suitable run-time. Lastly, investigations into starting with potential (rather than random) solutions may also increase solution accuracy with less generations. Our results from running the pr2392 data set shows that an increase in generations would yield even better results.

A Algorithm 3 - Population sizes 10 and 20

Graph	Generations	Pop. (10)	Std. Dev.	Pop. (20)	Std. Dev.	Optimal
eil51	5 000	446.83	6.33	441.08	6.41	
eil51	10 000	442.27	5.59	444.11	5.36	426
eil51	20 000	442.48	5.86	438.24	6.24	
eil76	5 000	585.21	9.11	575.47	9.54	
eil76	10 000	577.53	8.42	573.55	11.20	538
eil76	20 000	573.49	6.75	570.25	7.55	
eil101	5 000	716.95	15.39	690.46	10.53	
eil101	10 000	688.55	11.48	675.91	9.67	629
eil101	20 000	678.76	5.12	673.64	8.96	
st70	5 000	728.84	26.89	722.19	12.05	
st70	10 000	713.03	11.63	706.77	20.97	675
st70	20 000	713.48	11.61	705.96	15.48	
lin105	5 000	17 156.13	645.81	15 739.99	535.03	
lin105	10 000	16 081.00	472.31	15 626.97	237.32	14379
lin105	20 000	15 302.95	329.66	15 334.64	328.24	
kroA100	5 000	25 253.41	954.48	22 878.31	616.48	
kroA100	10 000	23 070.92	347.02	22 464.41	637.83	21282
kroA100	20 000	22 529.74	581.37	22 352.99	547.85	
kroC100	5 000	24 820.32	1 170.59	22 834.04	709.91	
kroC100	10 000	22 493.53	485.53	22 078.61	682.23	20749
kroC100	20 000	22 138.73	409.17	22 124.05	1 009.10	
kroD100	5 000	24 647.57	749.81	23 017.77	406.74	
kroD100	10 000	23 226.92	503.06	22 780.31	605.72	21294
kroD100	20 000	22 770.76	348.92	22 781.16	506.69	
pcb442	5 000	178 888.2	3 478.37	132 498.2	3 303.17	
pcb442	10 000	131 461.3	2 976.89	98 771.97	2 330.01	50788
pcb442	20 000	99 203.22	3 220.94	75 883.49	1 166.67	

Figure 4: Algorithm 3 average tour length for each population/generation pair, including standard deviation

B Inver-Over Algorithm - Population sizes 10 and 20

Graph	Generations	Pop. (10)	Std. Dev.	Pop. (20)	Std. Dev.	Optimal
eil51	5000	593.01	22.53	509.45	9.03	
eil51	10000	514.41	18.72	479.48	11.03	426
eil51	20000	471.08	12.78	456.91	9.77	
eil76	5000	966.32	54.42	799.32	47.24	
eil76	10000	774.71	47.53	677.16	22.09	538
eil76	20000	676.79	25.19	614.64	14.09	
eil101	5000	1401.26	54.48	1088.67	31.09	
eil101	10000	1111.20	35.19	913.08	40.50	629
eil101	20000	903.32	37.34	783.14	23.10	
st70	5000	1 267.89	82.03	1 038.04	39.43	
st70	10000	1 009.88	52.17	852.45	37.02	675
st70	20000	861.78	48.17	746.72	25.49	
lin105	5000	44 044.38	2 401.31	32 750.89	1 465.65	
lin105	10000	32 736.23	1 758.02	24 631.69	952.46	14379
lin105	20000	24 829.67	1 292.07	19 645.18	714.72	
kroA100	5000	59 015.86	3 394.76	45 688.96	2 880.30	
kroA100	10000	43 493.75	1 630.63	33 880.82	1 319.19	21282
kroA100	20000	35 225.44	990.67	27 837.08	944.19	
kroC100	5000	58 834.73	4 133.61	43 923.46	3 875.26	
kroC100	10000	44 677.12	2 154.33	33 366.69	2 187.98	20749
kroC100	20000	34 123.54	2 020.24	27 877.48	12 17.99	
kroD100	5000	57 416.18	1 468.99	44 534.29	2 197.09	
kroD100	10000	44 755.56	2 381.86	34 598.08	1 285.96	21294
kroD100	20000	33 474.88	1 032.04	27 396.77	634.89	
pcb442	5000	494 553.09	6 369.51	390 051.50	7 892.51	
pcb442	10000	387 135.73	12 228.42	292 111.80	12 606.27	50788
pcb442	20000	293 912.99	9 048.97	221 711.49	5 067.23	

Figure 5: Inver-Over algorithm average tour length for each population/generation pair, including standard deviation