Evolutionary Computer Practical Assignment Part 2

Alice Forehand

03/02/2013

Algorithms and Operators

Herein I compare the performance of a number of metaheuristic algorithms on the graph-bipartitioning problem. In addition to running Multi-Start Local Search (MLS), Iterated Local Search with perturbation step sizes 0.01, 0.03, and 0.05 (ILS-p01, -p03, and –p05, respectively) and Genetic Local Search with population sizes 25 and 50 (GLS-pop25 and –pop50, respectively), I implemented an Adaptive Pursuit algorithm for Iterated Local Search (AdaP). I included five perturbation sizes for the AdaP algorithm to choose from: (0.01 0.02 0.03 0.04 0.05). I suspected that if the ideal perturbation size was among those used in the standard ILS algorithms, AdaP would outperform the inferior ones and underperform the superior one. On the other hand, if the ideal step size was closer to one of the intermediate values I included for AdaP, then perhaps AdaP would outperform all the ILS algorithms.

Performance Measures

I represented the solutions as a pair of equal-sized, mutually exclusive sets of vertices. This representation made performing operations such as crossover very simple because, rather than keeping track of bit values and positions, I could call simple set functions. To evaluate the performance of the algorithms, I used a fitness measure equal to the number of cut edges in the graph—that is, the number of edges connecting vertices in opposite partitions. I measured the runtime in terms of fitness evaluations. For MLS, this value represents the sum of the fitness evaluations in all the local searches in a run. For ILS and GLS, it represents the number of fitness evaluations before reaching the best solution in a run. I chose to measure runtime in somewhat different ways for these algorithms because in ILS and GLS, I wanted to capture the time of convergence, of which there is none in MLS. Not including the evaluations that go into finding suboptimal solutions at the end of a run deflates the runtime values for ILS and GLS somewhat, but I feel that this measure is more accurate in that it records when the final solution was actually found. Furthermore, I observed that the clock time required to run all these algorithms was proportional to this measure (which I only measured with my eyes and did not keep formal track of).

Due to computational restrictions, I could not produce 1000 optima per run in a reasonable amount of time (I extrapolated the total time required when my first run of MLS had not finished after 4 hours), and instead settled for 100 optima per run. This affects the overall performance of the algorithms, and I am unlikely to have found a global optimum. However the relative performance of each algorithm should be unaffected. I completed 50 runs of MLS, and 30 runs of each of the other algorithms.

Results and Discussion

G-500 data:

Data set	Fitness (cut edges): min (mean SD)		Runtime (fitness evaluations): (mean SD)
MLS	85	(92.0 3.0463092)	(4.3122115E8 1.1850476E7)
ILS-p01	74	(83.4 5.251032)	(4774262.5 1266885.5)
ILS-p03	72	(80.7 4.352394)	(8976563.0 3079884.3)
ILS-p05	76	(83.3 4.2043624)	(1.1955783E7 4799016.0)
GLS-pop25	83	(89.933334 4.829999)	(4.352725E7 1089525.9)
GLS-pop50	82	(94.066666 7.352701)	(5.4813372E7 1402345.0)
AdaP	73	(81.36667 4.5128226)	(7729386.0 2096354.8)

For the G-500 graph, there was a fair amount of spread in fitness, but there is a huge spread in runtime, with three orders of magnitude represented among the means. There are two clusters in terms of fitness, with ILS and AdaP algorithms in the better, and the GLS and MLS in the worse. Interestingly, the amount of runtime required by the ILS algorithms (and their standard deviations) scaled with the size of the mutation step, which may be due to the larger partial solutions available from one iteration to the next with a smaller mutation size. Unsurprisingly, population size affected the runtime of GLS. A larger population size also affected overall performance, with a larger population performing worse overall and having a much higher variance. When considering how disruptive the crossover operator is to the structure of the solutions, preserving as little as 75% in early iterations, the greater diversity offered by a larger population is detrimental, rather than beneficial. Because of this, GLS outperforms MLS only in runtime (although it does so to a great degree). In future attempts at the graph-bipartitioning problem, I would seek a less disruptive crossover operator that preserved much larger partial solutions by employing a linkage tree or some other dependency tracking structure.

Significance Testing

Two-tailed t-test comparing fitness on G-500 graph with α=0.05 (non-significant differences highlighted):

Data set	(t-stat DoF)	ILS-p01	ILS-p03	ILS-p05	GLS-pop25	GLS-pop50
MLS		(8.182273 40.907486)	(12.501315 46.191498)	(9.883667 47.365494)	(2.1057427 43.0439)	(-1.4658774 35.06107)
ILS-p01			(2.168307 56.070095)	(0.08142256 55.352245)	(-5.015648 57.599518)	(-6.4662066 52.47513)
ILS-p03				(-2.353292 57.930695)	(-7.778427 57.38239)	(-8.5685215 47.10074)
ILS-p05					(-5.6737604 56.918552)	(-6.9624877 46.132576)
GLS-pop25						(-2.5734475 50.099255)
AdaP		(-1.6085148 56.717834)	(0.58240783 57.924168)	(-1.716859 57.711662)		

In the above chart, and those below, positive values indicate that the algorithm at the top outperformed the algorithm to the left. Significant differences are apparent among nearly all algorithms in terms of fitness. As predicted, the performance of AdaP was comparable to that of ILS, and although the differences were not significant, it slightly outperformed the inferior p01 and p05 algorithms, and slightly underperformed the superior p03. ILS-p03 was significantly better than all other algorithms and MLS and GLS-pop50 were significantly worse than all except each other. ILS-p01 and –p05 showed nearly identical performance. GLS-pop25 significantly outperformed –pop50, although it underperformed all other algorithms except MLS.

Despite a perturbation size of 0.3 displaying the best performance by a significant margin, the Adaptive Pursuit algorithm settled on a value of 0.1 for all runs. However, my algorithm only returned the parameter that was preferred at the end of the run, so it is likely that the algorithm favored larger perturbation steps toward the beginning of the runs, and preferred smaller ones toward the end. My reasoning for saying this is that the Adaptive Pursuit algorithm outperformed both ILS-0.1 and ILS-0.5, although not significantly.

Two-tailed t-test comparing runtimes on G-500 graph with α=0.05 (non-significant differences highlighted):

Data set	(t-stat DoF)	ILS-p01	ILS-p03	ILS-p05	GLS-pop25	GLS-pop50
MLS		(252.06747 50.853317)	(238.86276 59.381863)	(221.70134 70.542145)	(229.72081 50.373463)	(222.023 51.26676)
ILS-p01			(-6.91144 38.54058)	(-7.9249377 33.022488)	(-127.02903 56.72894)	(-145.02423 57.411606)
ILS-p03				(-2.861627 49.42397)	(-57.92673 36.146374)	(-74.18728 40.52902)
ILS-p05					(-35.139015 31.981579)	(-60.680866 59.922977)
GLS-pop25						(-46.950836 33.916748)
AdaP		(6.608021 47.689507)	(-1.8335294 51.122757)	(-4.4203386 39.678757)		

There were huge differences among the runtimes of each algorithm, often by orders of magnitude. The only insignificant difference, perhaps unsurprisingly, was between the Adaptive Pursuit algorithm and ILS-03, the median performer among the ILS algorithms in terms of runtime. In runtime as well as fitness, AdaP underperformed the best ILS algorithm and outperformed the others. Accounting for both fitness and runtime, adaptive pursuit appears to be a highly effective strategy. As with fitness, but on a much, much worse sale, MLS underperformed all other algorithms, even GLS-pop50, which came in second-to-last. Every algorithm outperformed those that were more disruptive, with ILS-p01 standing out as the most efficient.

U-500 data:

Data set	Fitness : min (mean SD)		Runtime (mean SD)
MLS	88	(119.08 8.835979)	(6.220393E8 8.6830726E8)
ILS-p01	71	(94.066666 11.732954)	(2986548.5 513652.22)
ILS-p03	90	(106.166664 8.201964)	(4949728.5 1695863.5)
ILS-p05	89	(105.63333 9.727909)	(6034818.0 2858247.8)
GLS-pop25	96	(117.96667 17.090899)	(3.190184E7 1335509.1)
GLS-pop50	99	(119.566666 15.98475)	(4.0614176E7 889905.0)
AdaP	81	(98.1 8.564851)	(3908429.8 954690.0)

Here we see the same kind of clustering as on the other graph, although this time, ILS-p01 outperforms the others, not –p03. Again, AdaP falls squarely between the first and second place ILS algorithms in both runtime and fitness. And again, MLS has an atrociously slow runtime, but it competes somewhat better with the GLS algorithms than on the G-500 graph.

Significant Testing

Two-tailed t-test comparing fitness on U-500 graph with α=0.05 (non-significant differences highlighted):

Data set	(t-stat DoF)	ILS-p01	ILS-p03	ILS-p05	GLS-pop25	GLS-pop50
MLS		(10.086162 48.75356)	(6.6210074 64.841774)	(6.1920233 56.607773)	(0.33122468 38.462032)	(-0.1532956 39.81621)
ILS-p01			(-4.6295466 51.87949)	(-4.156707 56.075848)	(-6.314588 51.36672)	(-7.0438185 53.218628)
ILS-p03				(0.22957765 56.389664)	(-3.409347 41.684944)	(-4.0851607 43.28056)
ILS-p05					(-3.4350784 46.00557)	(-4.078421 47.88994)
GLS-pop25						(-0.37449336 57.74225)
AdaP		(1.5207717 53.071487)	(-3.7257817 57.89164)	(-3.1835196 57.084377)		

Here we see significance and insignificance in similar places, however this time AdaP performs relatively better, only insignificantly underperforming ILS-p01 and significantly outperforming the others, which were insignificantly different from each other. GLS-pop25 performed relatively worse on this graph, falling within range of both MLS and GLS-p50.

Two-tailed t-test comparing runtimes on U-500 graph with α=0.05 (non-significant differences highlighted):

Data set	(t-stat DoF)	ILS-p01	ILS-p03	ILS-p05	GLS-pop25	GLS-pop50
MLS		(5.0412602 49.00006)	(5.025258 49.00062)	(5.016392 49.001778)	(4.8057804 49.000385)	(4.7348366 49.000175)
ILS-p01			(-6.0683455 34.276485)	(-5.749263 30.871178)	(-110.68388 37.39599)	(-200.57782 46.392696)
ILS-p03				(-1.7882689 47.16654)	(-68.38833 54.97835)	(-101.99722 43.845398)
ILS-p05					(-44.908283 41.086483)	(-63.268414 34.56998)
GLS-pop25						(-29.734673 50.511692)
AdaP		(4.657646 44.49151)	(-2.9306648 45.70347)	(-3.8648803 35.39118)		

As with the G-500 graph, here again we see large significant differences among runtimes, with only one insignificant difference, this time between ILS-03 and ILS-05, rather than AdaP. AdaP has more pronounced differences in runtime performance from the other ILS algorithms than with the other graph, similar to its change in fitness performance. Here we also see a decrease in difference between MLS and the other algorithms, but a closer look reveals that this is due to MLS's gigantic standard deviation caused by a few of its runs taking an order of magnitude longer to complete than the rest. Otherwise, all the algorithms once again outperform those that are more disruptive.

Conclusion

AdaP is great, and for this type of problem, ILS is a much better performer in terms of both fitness and runtime. If GLS is going to be any good, it needs to have mechanisms in place that preserve larger solutions than the current standard uniform crossover operator. MLS is terrible and no one should use it ever.