10th International Conference of the Practice and Theory of Automated Timetabling
PATAT 2014, 26-29 August 2014, York, United Kingdom

Improvement by Combination

How to increase the Performance of Optimisation Algorithms by combining them

Johannes Ostler · Peter Wilke

Abstract Working with different optimisation algorithms leads to the observation that different types of solutions are generated, disclosing their different nature, their pros and cons. We investigated the question whether or not the combination of optimisation algorithms yield even better results than each of its constituents or will their drawbacks make things even worse.

Keywords Optimisation Algorithms · Combination of Algorithms · Performance Improvement

1 Introduction

Two runs of random based optimisation algorithms lead to different solutions, especially if different optimisation algorithms are used [Wilke and Ostler(2008)].There are different approaches to escape local minima like slow cooling in Simulated Annealing or jump to a worse solution like in Walk Dow Jump Up.We want to combine the good performance of neighbourhood based algorithms and the advantage of genetic operators.

Johannes Ostler
E-mail: Johannes.Ostler@informatik.uni-erlangen.de

Peter Wilke
University of Erlangen-Nuernberg
Computer Science Department
Multi Criteria Optimisation Group
Martensstrasse 3, 91058 Erlangen, Germany
E-mail: Peter.Wilke@fau.de

2 Basic Algorithms

2.1 Algorithms

The basic algorithms are plain vanilla implementations of Genetic Algorithm[Goldberg(2013)], Simulated Annealing[Kirkpatrick et al(1983)Kirkpatrick, Gelatt, and Vecchi, Preiss(1998)], Late Acceptance [Burke and Bykov(2008)] and Walk Down Jump Up [Wilke and Killer(2010)]. The used plain vanilla genetic algorithm implementation is corresponding to [Ho Pham L. Huy Anh and Nam(2011)].

All the basic algorithms are implemented for solving different timetabling problems defined in the REC model[Ostler and Wilke(2010)] of the EATTS [Gröbner et al(2003)Gröbner, Wilke, and Büttcher].

So the algorithms in the stand alone version are used for solving the problems. Several runs of the algorithms on the same problem data base provide the reference values for the combined algorithms.

For the reference runs we used a population with 10 or 20 individuals. The mutation rate was between 20 and 40 percent and between 0.5 and 5 percent of the not fixed resource lists of the mutated individuals were changed. The champion (i.e. the best individual) was excluded from mutation. The next generation consisted of the best and 5 or 10 other individuals. 40 % good individuals, which means cost better than average costs, and 60% random based individuals. To fill the population two point cross over was used. So 4 or rather 9 individuals were be created in every iteration

2.2 Data Base

2.2.1 University Exercise Group Planning

The Exercise Group Planning Problem assigns students to a list of exercise groups. The constraints are the limitation of the group size, the first and second choice selections made by the students and their wish to share the group with the best friends.

2.2.2 GAT 2011 courses at a university

Our university organises a girl-and-technology week each year to attract more female students to technical subjects. In this scenario the tutors and time slots for the events are fixed while students (not classes) have to be assigned to the project of their choice, details given in table 1. Each student has a list of 4 preferred courses and can declare friends with whom she wants to share the same courses.

Table 1 Statistics of example problems

University Exercise Group Planning

Events:	25
Courses	25
Students	798
possible solutions	$\binom{798}{25}$

GAT 2011 example

Events:	229
TimeSlots:	57
Subjects:	52
Girls:	170
possible solutions	10^{1313}

3 Combining Algorithms

Preceding research [Wilke and Ostler(2008)] shows, that different random based runs mostly lead to different results, especially if different algorithms were used. Combining these different solutions by a Crossover operator could be a useful way for escaping local minima.

The essential idea is to combine the run of the Genetic Algorithm with other optimisation algorithms. For example Simulated Annealing can be used in the optimisation step or Walk Down Jump Up to generate decent individuals for the start population. The algorithms used in the optimisation step will optimise the solution by using random based decisions. And the Crossover operation over the different solutions will provide new solutions that are not in the neighbourhood of a local minimum.

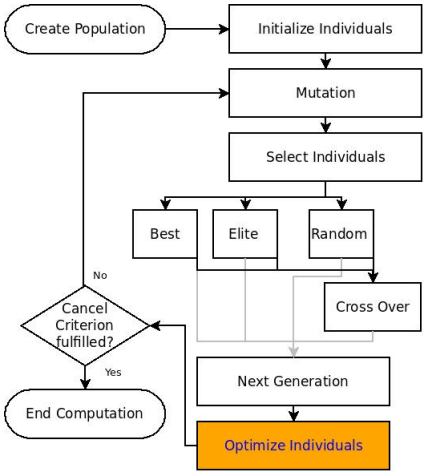


Fig. 1 Genetic Algorithm - modified implementation

For the test runs we used the same parameters as for the plain vanilla version. The interesting modification was an additional modification step. We optimised the new generation by short runs of different optimisation algorithms. We used Late Acceptance, Walk Down Jump Up, Simulated Annealing and Hill Climbing. In the optimisation step for every individual will be

selected one of these algorithms by random. The runs ends after 2 seconds or 100000 iterations. The computation of a modified Genetic Algorithm runs 200 iterations.

The algorithm offers a good speed up when running the optimisation steps parallel. Because there are only synchronisation points needed after running the optimisation steps.

4 Results

4.1 Test Environment

The tests were running for every configuration 5 times. The best and worst case were dropped and the average value of the remaining results was computed. The combined algorithm was running in sequential mode. The speed up of running the optimisation steps parallel would distort the results.

4.2 Chart Types

4.2.1 Population Comparison Chart - PCC

The Population Comparison Chart PCC shows the quality change of some individuals and the best solution during the optimisation process. The vertical axis shows the costs, the horizontal axis the iteration. The costs are computed before the cancel criterion were checked meaning after the cross over or the optimisation step. For better scaling the first 10 iterations are dropped. If an individual dies it will be replaced by a child born in the cross over process. This chart provides a good look on changes of the different individuals of a genetic algorithm.

4.2.2 Genetic Step Chart - GSC

The Genetic Step Chart plots the population average costs after one specific step of the genetic algorithm in relation to the iteration. The vertical axis shows the costs, the horizontal axis the iteration. The specific steps are shown in table 2.

Table 2 Genetic Step Chart - GSC

mutate	the average costs after mutation
select	the average costs after selecting individuals (reduced population size)
cross over:	the average costs after cross over individuals (recovered population size)
optimise	the average costs after optimisation step
best	the costs of the best individual at the end of the iteration

The GSC shows the costs changes of the population after the single steps of the iteration. Some steps increase the costs like mutation, other will bring them down like the optimisation or selection.

4.2.3 Algorithm Comparison Chart - ACC

The Algorithm Comparison Chart ACC shows the comparison between different optimisation algorithms. For comparison issues a standardised base is need. The iteration counter is not suitable, because the computation time is quite different between one iteration of different algorithms. The only useful base for comparison seems to be execution time. But execution time depends also on external influences. So the tests must run several times and the average must be computed while dropping best and worst solution.

The ACC shows the best cost value on the vertical axis corresponding to the time value on the horizontal axis. For better scaling the costs axis the first seconds are dropped. The ACC contains one line per executed algorithm. With ACC different runs of the same algorithm with different parameter setting can also be compared.

4.3 Test and Results

4.3.1 Results EGP Problem

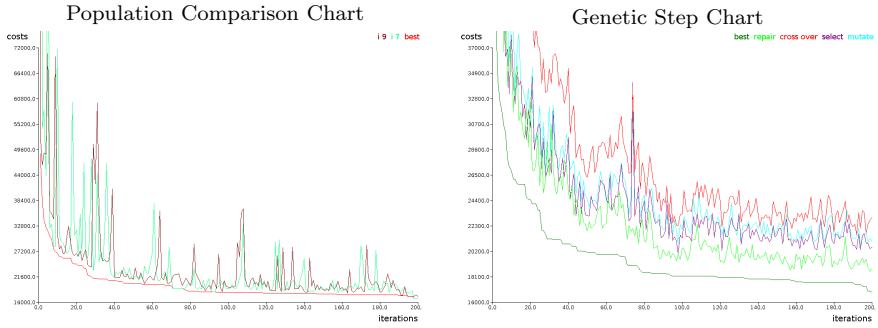
First we run the combined algorithm with a high mutation rate of 5 or 3 percent and the best individual was not mutated. The effect was that the best individual was improved by the different runs of optimisation algorithms, but there was no "genetic effect", which means that the other individuals which were mutated after every step have no chance to get best individual. So a sequential run of different optimisation algorithms on the best individual would have the same effects.

After setting the mutation rate to a half percent the results were different. The Population Comparison Chart 2 shows that after some iterations the best individual is reset by another individual. For better clearness we select two individuals and the best cost line. If the best cost line is shown in red colour a not selected individual is the best.

The Genetic Step Chart shows that the average cost increases after mutation step between 10 and 15 percent. Then the selection of some individual decrease the average costs a little bit. But the crossover operator increases the costs between 20 and 50 percent. The optimisation step decreases the costs so that the costs after optimisation are lower than after the mutation step. The most important insight by looking on the GSC is that the average costs decreases.

The Algorithm Comparing Chart shows the comparison of Walk Down Jump Up, Simulated Annealing and Late Acceptance and the combined genetic algorithm in comparison. After 320 seconds Simulated Annealing offers

Fig. 2 EGP Problem



the best solution, but after 1500 seconds the best individual of the combined algorithm is better. Simulated Annealing stagnates at this time, the combined is still improving. Late Acceptance offers a good solution after 2240 seconds but is also stagnating after this.

So after 3200 seconds the combined algorithm leads to the best solution with 16639 penalty points the results are following (table 3):

4.3.2 GAT Problem

The Algorithm Comparison Chart for the Girls and Techniques Problem shows the problem of neighbourhood bases algorithms: stagnation in local minima. This time ranking of the reference algorithm is inverted. Walk Down Jump Up leads to the best result. But this solution has with 7237 over the double costs of the combined algorithm, which solution has 3519 penalty points.

Table 3 Comparing Results EGP and GAT Problem

	EGP problem		GAT Problem	
Algorithms	Costs	Diff to Combined	Costs	Diff to Combined
Late Acceptance	17334	+4.2 %	7237	+106 %
Simulated Annealing	18808	+13.0 %	7507	+113 %
Walk Down Jump Up	20676	+24.2 %	10795	+207 %

5 Summary

So over all the combined algorithm could be a good alternative to neighbourhood search based algorithms. Like other genetic algorithms the usage of main storage for n individuals is n times higher than the usage of neighbourhood search based algorithms.

But if enough time and storage could be spend the combined algorithm should lead to good solutions. If a multi core or distributed hardware can be used there is good speed up while running the optimisation step parallel.

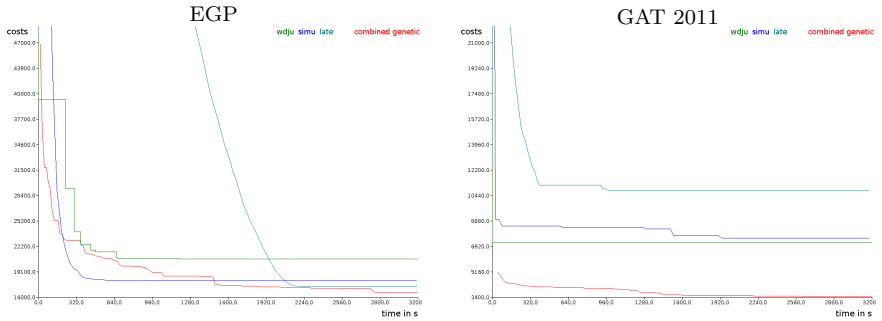


Fig. 3 Algorithm Comparison Chart of the Problems

6 Outlook

The setting of the parameters, especially of the mutation rate has big influence to the performance of the algorithm. So an automated parameter setting could be helpful. Especially the adaptation of run time and selection probability of the neighbourhood based algorithms corresponding to their average cost improvement could have positive effects to runtime and solution quality.

References

[Burke and Bykov(2008)] Burke EK, Bykov Y (2008) A late acceptance strategy in hill-climbing for exam timetabling problems. In: Proceeding of the 7th international conference on the Practice and Theory of Automated Timetabling, Patat2008, URL http://w1.cirrelt.ca/∼ patat2008/PATAT_7_PROCEEDINGS/Papers/Post-WD2a.pdf

[Goldberg(2013)] Goldberg DE (2013) Genetic Algorithms. Pearson Education, URL http://books.google.de/books?id=6gzS07Sv9hoC

[Gröbner et al(2003)Gröbner, Wilke, and Büttcher] Gröbner M, Wilke P, Büttcher S (2003) A standard framework for timetabling problems. In: Practice and Theory of AutomatedTimetabling IV, Springer Berlin / Heidelberg, Lecture Notes in Computer Science, vol 2740, pp 24–38, DOI 10.1007/b11828, URL http://www.springerlink.com/content/2mj0rwlpbp5uvh1v/

[Ho Pham L. Huy Anh and Nam(2011)] Ho Pham L Huy Anh KKA, Nam NT (2011) Modeling identification of the nonlinear robot arm system using miso narx fuzzy model and genetic algorithm. In: Robot Arms, InTech, pp 1–10, URL http://http://openaccessbooks.org/book/robot-arms

[Kirkpatrick et al(1983)Kirkpatrick, Gelatt, and Vecchi] Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671 – 680

[Ostler and Wilke(2010)] Ostler J, Wilke P (2010) The erlangen advanced timetabling system (eatts) unified xml file format for the specification of timetabling systems. In: Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling, Patat2010 - Queens University Belfast, pp 447–464, URL http://www.cs.qub.ac.uk/∼ b.mccollum/patat10/Proceedings_patat10.pdf

[Preiss(1998)] Preiss BR (1998) Data Structures and Algorithms with Object-Oriented Design Patterns in Java, http://www.brpreiss.com/books/opus5/html/page474.html, chap Simulated Annealing, p 474

[Wilke and Killer(2010)] Wilke P, Killer H (2010) Walk down jump up - a new hybrid algorithm for time tabling problems. In: Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling, Patat2010 - Queens University Belfast, pp 440–446, URL http://www.cs.qub.ac.uk/∼ b.mccollum/patat10/Proceedings_patat10.pdf

[Wilke and Ostler(2008)] Wilke P, Ostler J (2008) Solving the School Time Tabling Problem Using Tabu Search, Simulated Annealing, Genetic and Branch & Bound Algorithms. In: Burke E, Gendreau M (eds) PATAT '08 Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, URL http://www5.informatik.uni-erlangen.de/Forschung/Publikationen/2008/Wilke08-STS.pdf