Comparison of Algorithms solving School and Course Time Tabling Problems using the Erlangen Advanced Time Tabling System (EATTS)

Peter Wilke · Helmut Killer

Abstract Seven algorithms are applied to two common time tabling problems: school and course time tabling. The algorithms were implemented using the EATTS (Erlangen Advanced Time Tabling System) which allows to compare and evaluate the algorithms regarding their performance and ability to solve the problems.

Keywords Great Deluge · Harmony Search · Simulated Annealing · Genetic Algorithms · Tabu Search · Immune System · Real World Problems · EATTS · Erlangen Advanced Time Tabling System

1 Introduction

Recently we had to deal with two real world problems, namely school and course time tabling, and jumped on this opportunity to investigate regarding their ability and performance to solve the given problems. All algorithms have been implemented using the current version of EATTS (Erlangen Advanced Time Tabling System [Wil10]), to allow comparison and evaluation.

2 The Problems

2.1 School 2009 Time Tabling

The data for our School 2009 time tabling problem represents an existing school with students from year 1 to 10. In this scenario the classes and their subjects are given, while class rooms and time slots have to be assigned to the events. Teachers can be assigned fixed to class/subject pairs, but don't have to. Here it is sufficient that one student represents the entire class, details given in table 1.

Peter Wilke
Universitaet Erlangen-Nuernberg, Department Informatik, Martensstrasse 3, 91058 Erlangen, Germany
Ph.: +49 (9131) 85-27998
E-mail: Peter.Wilke@Informatik.Uni-Erlangen.DE Helmut Killer
E-mail: Helmut.Killer@gmx.DE

Events:	178
Resources of type TimeSlot:	81
Resources of type Class:	14
Resources of type Student:	14
Resources of type Teacher:	28
Resources of type Room:	37
Resources of type Subject:	78
Resources of type Asset:	0
Resources of type LessonProperties:	178
Resources of type Building:	1
Resources:	431
Resources to be assigned:	118
Number of possible solutions:	$> 10^{439}$

Table 1 The main characteristics of the School 2009 example

2.2 MuT 2009 courses at a university

Our university organises a girl-and-technology (in german: „Maedchen und Technik“, abbreviated MuT) 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 2.

Events:	229
Resources of type TimeSlot:	57
Resources of type Subject:	52
Resources of type Girl:	170
Resources:	279
Resources to be assigned:	170
Number of possible solutions:	$> 10^{656}$

Table 2 The main characteristics of the MuT 2009 example

3 Algorithms

The following algorithms were implemented:

- Genetic Algorithms
- Immune System
- Harmony Search
- Tabu Search
- Simulated Annealing
- Great Deluge
- Walk Down Jump Up

Table 3 shows the used algorithms and their main characteristics, indicated by a mark in the corresponding row. The characteristics are:

population In each iteration one or more solution candidates are produced and substitute older solutions.

trajectory In each iteration only one solution candidate exists.

history The algorithms depends (at least partial) on its history of computational steps.

limit At each step of the computation the limit, which may vary during the computation, determines if the newly generated solution is accepted as new current solution candidate.

round based In each round, i.e the iteration step, one or more solutions are generated, but only one solution is selected for the next iteration step.

references For detailed information about the algorithms please consider the recommended bibliographic references for reading.

	population	trajectory	history	limit	round based	references
Genetic Algorithms	x					[Gol89, GD91, Whi89, Sys91, BT95a, BT95b]
Immune System	x				x	[MKM06]
Harmony Search	x					[ABKG08, Gee09]
Tabu Search		x	x		x	[GS01, KH05]
Simulated Annealing		x		x	x	[Hel04, vL87, FSAPMV08]
Great Deluge		x		x	x	[BBNP04]
Walk Down Jump Up		x			x	[Kil09, WK10]

Table 3 Used Algorithms and their characteristic properties

All algorithms were implemented using the EATTS Erlangen Advanced Time Tabling System framework and all runs were performed using computers as specified in table 4.

The following results are achived with sequential versions for Simulated annealing, Great Deluge, Walk Up Jump Down, while parallel versions were used for Genetic Algorithm, Harmony Search, Immune System and Tabu Search. All experiments were run on the same computer so results are comparable.

Two different setups for the experience where used. One setup was designed to achieve the fastest reduction of costs. The other setup was used to see the long term improved behaviour of the algorithms, e.g. does the algorithms profit form excessive computation. Both setups starts with randomly generated initial solutions.

3.1 Simulated Annealing

Basically Simulated Annealing maps the cooling down process of matter to optimization problems. The standard version uses an acceptance probability function which is fixed during cool down, in our implementation we allow modifications of this function over time, i.e. the probability of making the transition from the current state s to a candidate

	QuadCore	DuoCore
CPU:	Intel Core4Quad 2.8 GHz	Intel Core2Duo 3,0 GHz
RAM:	8 GB	4 GB
OS:	Debian Linux Kernel 2.6 x86-64	Suse Linux Kernel 2.6 i686
JVM:	Java 6 32 Bit Server JVM	Java 6 32 Bit Server JVM

Table 4 Specs of the computers used

new state s' is specified by the acceptance probability function P(e,e',T,t), that depends on the energies e = E(s) and e' = E(s') of the two states, on a global time-varying parameter T called the temperature, and the time t representing the time spend on the cooling process so far. In our simulations we used this parameter to adapt the temperature decrease on the maximum available time to cool down, i.e. the temperature always reaches it's minimum when the computation deadline is reached.

The performance of Simulated Annealing can be characterized as a slow starter but winner, see fig. 9. It's clearly visible that Simulated Annealing reduces the costs slower than all other algorithms except Great Deluge, which is designed to perform in an linear manner, see section 3.6.

Fig. 1 shows the details of the descent of the costs. The dots show the current solutions under review and the line indicates the current best solution. The gap between the dots and the line is the middle range of a distribution of solutions. The ranges is determined by an acceptance criteria (upper limit) and the current best solution (lower limit).

Remark: To make results more easily to interpret the x-axis (time) in the performance plots has been scaled to extend phases of fast decline and to condense phases of stagnation.

Experiments with parallel versions of the Simulated Annealing algorithms haven't shown sufficient results and will be subject to a closer investigation.

The starting temperature is chosen randomly as part of the random initial solution. Cooling rate was initialy 0.99 and was adapted linearly to reach 0.0 at the end of the given computation time.

3.2 Tabu Search

Tabu Search algorithm is trajectory based but can effectively be computed using multi threading because for the calculation of a new solution at first a set of neighbours is calculated and evaluated and this can be done in parallel. The best speed up is achieved when the number of threads matches the number of available cores respectively CPUs. On a Core2Quad CPU 19.4 iterations/second are computed when only one core is used, the multi-threading version achieves 45.3 iterations/second yielding a speed up factor of 2.33.

Fig. 2 shows the details when Tabu Search solves the MuT 2009 example. In contrast to the nearly evenly solution candidate distribution of Simulated Annealing here the solution candidates are concentrated near the currently best solution.

The size of the tabu list was 40, 200 neighbours were generated in each step and the champion was the initial solution of the next iteration.

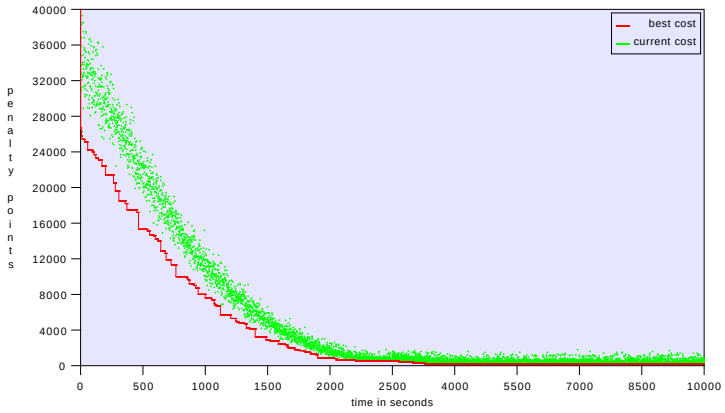

Fig. 1 Simulated Annealing solving the School 2009 problem

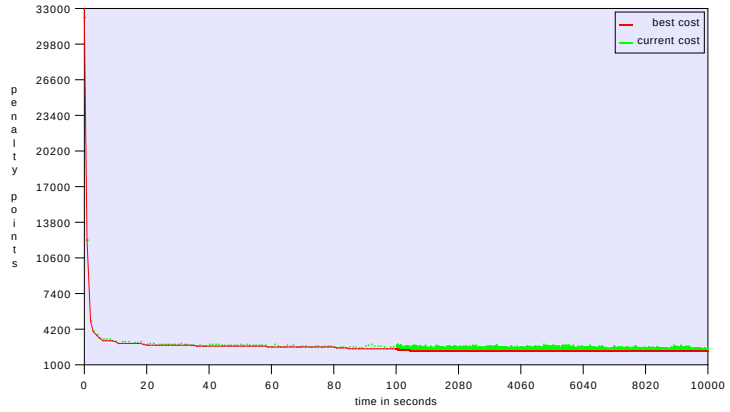

Fig. 2 Tabu Search solving the MuT 2009 problem

3.3 Genetic Algorithm

In our implementation all genetic operators have a complement operator, which reverses the effect of the operator. This speeds up the computation because it enables multiple operator applications. A crossover requires significant more computing time than a mutation, therefore reversal of a mutation leading to a lethal chromosome after a successful crossover avoids the waste of the time spend for the crossover. So our implementation reverses all genetic operations when they don't lead to a better solution.

Genetic Algorithms are population based and therefore a good candidate for parallel computation. We implemented an Island Ferry concept [CP95] which we optimized for use on multi-core CPUs. For each CPU core a genetic algorithm instance is started and

all instances exchange their best solutions periodically over time, so all populations reach the current global lowest cost value.

Fig. 3 shows the usual descent of a Genetic Algorithm with its typical plateaus.

Experiments were run with population sizes of 10 or 100, approx. half of the population is replaced in each generation. A two-point cross-over and a mutation rate of 0.005 was used.

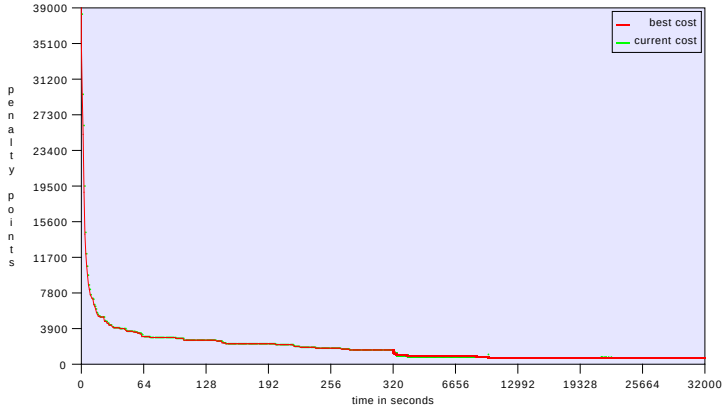

Fig. 3 Genetic Algorithm solving the School 2009 problem

3.4 Harmony Search

We first modified the original Harmony Search algorithm to allow variants with a wider bandwidth as suggested by [MFD07]. But the results were disappointing. Better results were achieved when stagnation was dissolved using randomly generated parameter values or chosen from an interval (shake).

In each iteration Harmony Search generates a new solution which can be post-processed by the same hybrid operators as used in Genetic Algorithms.

Harmony Search can be computed in threads, but the threads depend on the same Harmony Memory. The number of threads should match the number of cores/CPUs, otherwise threads sharing a core/CPU will fall behind the other threads and produce solution candidates which are based on outdated versions of the Harmony Memory. This means that precious computing time is dissipated.

Fig. 4 shows a rapid decline in the first seconds followed by a much smoother and slower descent, a quite different behaviour compared to the other algorithms. This nearly 90 degree turn is an extreme example but in all our Harmony Search experiments similar turns were observed. The parameters used were Harmony Memory Consideration Rate 0.99, Pitch Adjusting Rate PAR 0.01, Harmony Memory Size HMS 10 or 100, bandwidth = 0.1.

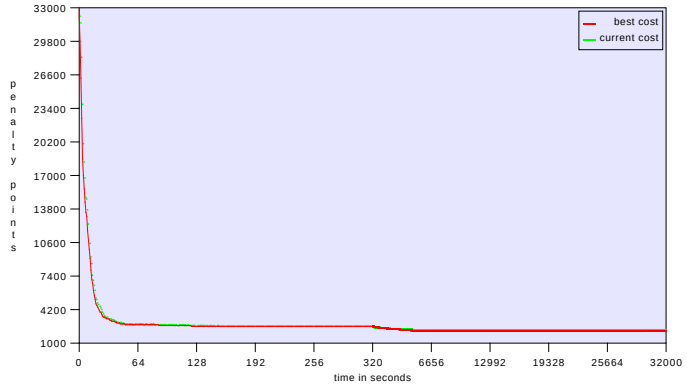

Fig. 4 Harmony Search solving the MuT 2009 problem

3.5 Immune System

The original Immune System suggests three variants: clonal selection, immune network and negative selection. Best results on our problems were achieved using the negative selection variant which basically eliminates candidates (so called detectors) with fitness under the mean fitness value of the population.

Because Immune System is population based it is a good candidate for multi threaded computation. The detectors can be generated and evaluated in parallel because they don't depend on each other. The main algorithm wait until all detectors are generated and evaluated, therefore it is recommended for efficiency reasons to generate generators in numbers which are multiples of the number of available cores/CPUs.

Fig. 5 shows a remarkable pattern in the distribution of solution candidates. Three clearly distinct bands are visible: the first close above the champion, the second a short gap above the first and a very thin third quite far away from the other two. The first two bands represent the candidates below/above the mean fitness.

Fig 6 shows even more bands. The gap between the bands corresponds to the given penalty cost. The line indicating the drops down immediately after start and is stable for the rest of the run.

For Immune System a population size of 10 or 100 was used and approx. half of the population was replaced in each iteration. Parameters were maxMultiMoves 10.0, Multimove tweak, allEqualFactor 1.1, all equal tweak.

3.6 Great Deluge

The original Great Deluge algorithm has a linear descending limit. Generated solutions with cost above this limit are rejected and below are accepted. The best solution is always saved. A small modification is the introduction of a best solution backup. When current solution cost can't get below the limit for a certain amount of time, the saved best solution can be restored if it has lower cost, so the algorithm is revived.

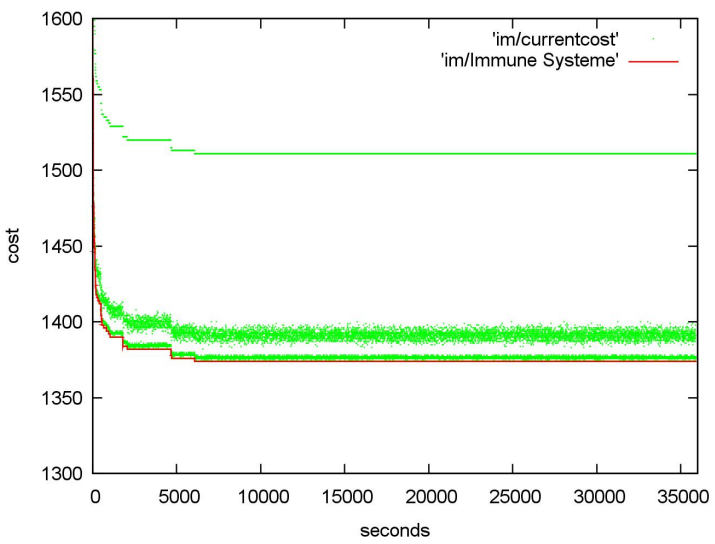

Fig. 5 Immune Systeme solving the MuT 2009 problem

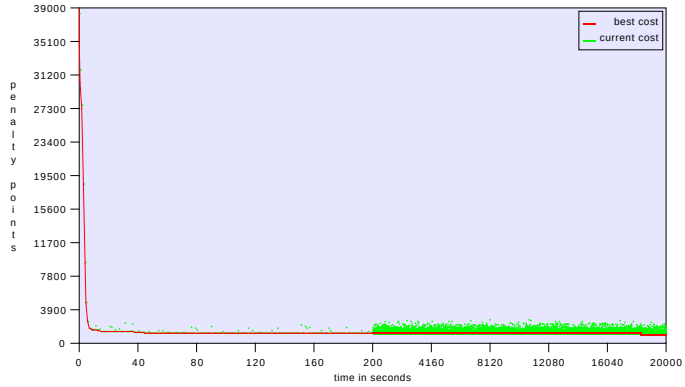

Fig. 6 Immune System solving the School 2009 problem

Fig. 7 shows the behaviour of Great Deluge in detail. The line indicates the best solution while the dots represent solution candidates. The sharp edge on top of the solution cloud is due to the limit function.

The linear decreasing rate was adopted to the expected runtime, i.e. initial cost divided by runtime in seconds giving the value to be subtracted in each step.

3.7 Walk Down Jump Up Algorithm

Walk Down Jump Up [WK10, Kil09] combines hill climbing, jump operator and Great Deluge. It begins with an initial random solution and starts it's descent until a local

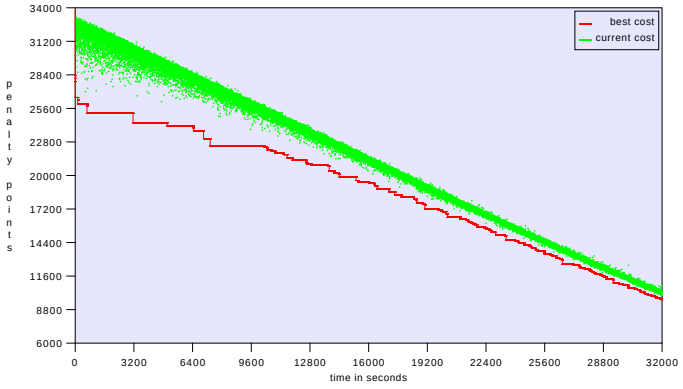

Fig. 7 Great Deluge solving the MuT 2009 problem

minimum is reached. Then the jump operator is used to set an higher acceptance limit, i.e. all newly generated solutions with costs below this limit are accepted and each following iteration will decrease this limit until a new local minimum is reached. If the new local minimum is not better than the old one the height of the jump is increased, otherwise the search continues from the newly found local minimum and jump height is reset. A small modification is best solution backup, when current solution reached a local minimum which is above best solution, then best solution is restored before next jump.

Walk Down Jump Up is trajectory based and again therefore multi threading is not really straight forward. Because the Jump Up operator can jump pretty far even if the current solution is quite good it males sense to start the Walk Down Jump Up algorithm on all available cores/CPUs and use the Island Ferry mechanism to exchange the best solutions.

Fig. 8 shows details of the performance. Walk Down Jump Up has a very steep descent phase of approximately 200 seconds at the very beginning followed by a long stagnation phase for the rest of the 20.000 seconds run.

4 Summary

The performance of all 7 algorithms applied to the 2 real world problems is shown in fig. 9. The diverse nature of the problems has different impacts on the algorithms. While the Immune System algorithms is leading head on head with Simulated Annealing and Walk Down Jump Up when solving the MuT 2009 example, is comes in second to last when solving the School 2009 example. Great Deluge shows excellent performance on the School 2009 example, but is extremely bad on the MuT 2009 example.

Table 5 resp. 6 show the results achieved in 10 hours.

While all algorithms except the Great Deluge algorithm show nearly equal performance, the situation for the School 2009 example is quite heterogeneous. Here the field is lead by Walk Up Jump Down, Great Deluge and Simulated Annealing, a mid field

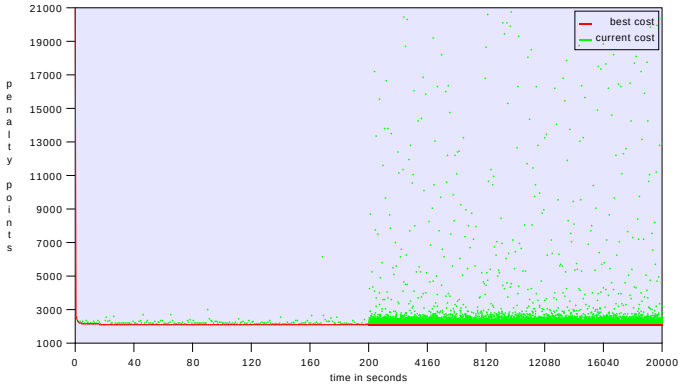

Fig. 8 Walk Down Jump Up solving the MuT 2009 problem

consisting of Genetic Algorithm and Harmony Search, and trailed by Immune System and Tabu Search.

MuT 2009 for 10h	solved	final cost	stagnation after
Genetic algorithm	yes	2216	6h:6m
Great Deluge	no	7300	no
Harmony Search	yes	2196	9h:42m
Immune System	yes	2142	1h:48m
Simulated Annealing	yes	2130	2h:56m
Tabu Search	yes	2273	2h:5m
WalkDownJumpUp	yes	2138	4h:6m

Table 5 Comparison of all 10h runs for the MuT 2009 problem

School 2009 for 10h	solved	final cost	stagnation after
Genetic algorithm	no	628	9h:56m
Great Deluge	yes	256	4h:57m
Harmony Search	yes	319	9h:18m
Immune System	no	1055	5h:5m
Simulated Annealing	yes	206	1h:10m
Tabu Search	no	1182	0h:0m:5s
WalkDownJumpUp	yes	278	0h:12m

Table 6 Comparison of all 10h runs for the School 2009 problem

A closer look on the time available for the computation shows that two algorithms, namely Genetic Algorithm and Harmony Search, can profit from additional computing time, while the others don't improve their costs significantly relative to the solutions

found after only 10 seconds, but the hard constraints become fulfilled, so the solutions become "more and more" valid. This observation is true for both examples.

Overall winner is Simulated Annealing, best results and least expensive runtime costs make this algorithm the best choice for the given problems.

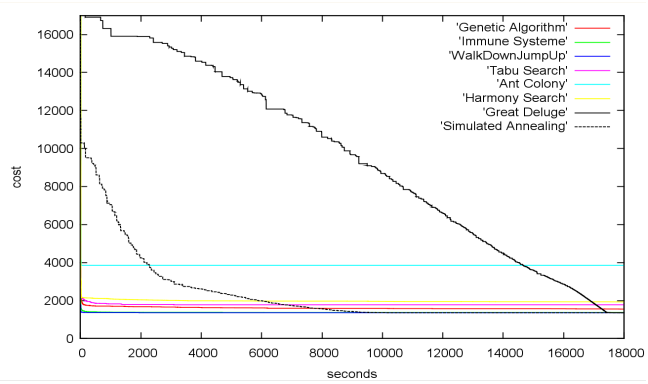

Fig. 9 Performance of all algorithms solving the MuT-Problem over 5 hours

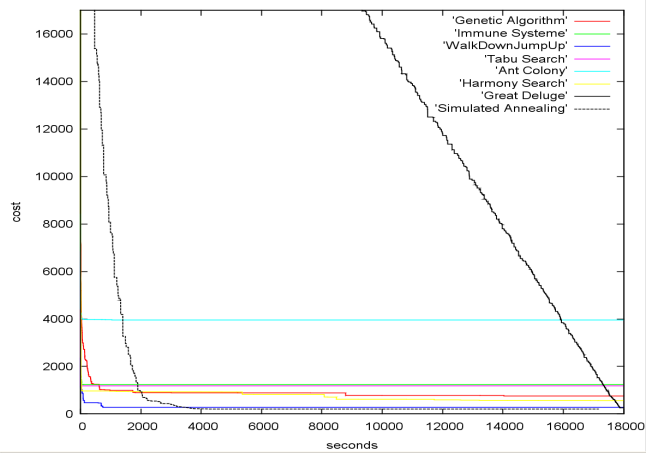

Fig. 10 Performance of all algorithms solving the School-Problem over 5 hours

5 Conclusion

The investigations have shown that it is a good idea to test several algorithms on their ability to solve a given problem, even if the problems look quite similar.

Future work will be the extension of our database of problem descriptions and implementation of additional algorithms.

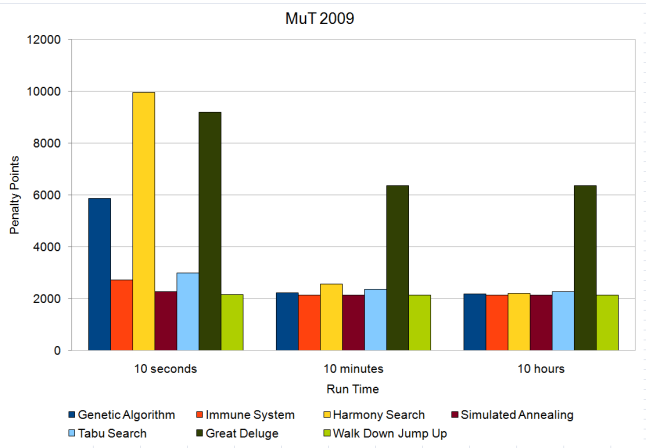

Fig. 11 All algorithms and their performance on the MuT 2009 example

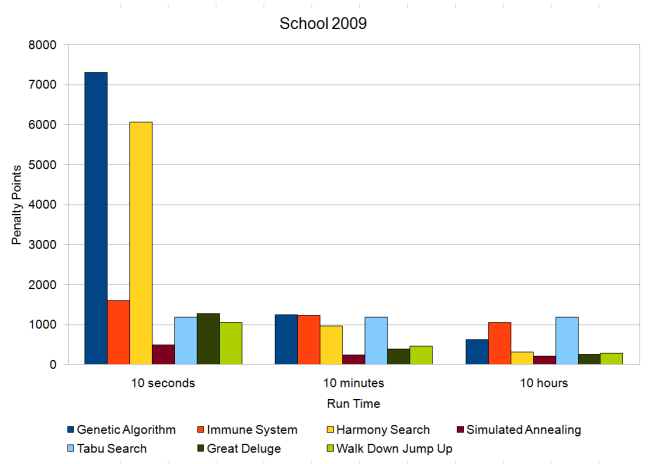

Fig. 12 All algorithms and their performance on the School 2009 example

Acknowledgements We would like to thank all our colleagues at EATTS for their support and special thanks to Dimo Korsch, Sabine Helwig, Johannes Ostler for their contributions to the implementation and test of the EATTS algorithms toolbox.

References

[ABKG08] Mohammed Azmi Al-Betar, Ahamad Tajudin Khader, and Taufiq Abdul Gani. A harmony search algorithm for university course timetabling. 2008.

[BBNP04] Edmund Burke, Yuri Bykov, James Newall, and Sanja Petrovic. A time-predefined local search approach to exam timetabling problems. 2004.

[BM10] E. Burke and Barry McCollum, editors. *Springer Lecture Notes in Computer Science*. Springer-Verlag, 2010.

[BT95a] Tobias Blickle and Lothar Thiele. A comparison of selection schemes used in genetic algorithms. *TIK-Report*, 11, 1995.

[BT95b] Tobias Blickle and Lothar Thiele. A mathematical analysis of tournament selection. *Genetic Algorithms: Proceedings of the 6th International Conference*, 1995.

[CP95] E. Cantu-Paz. A summary of research on parallel genetic algorithms. Technical Report 95007, IlliGAL Report, 1995.

[FSAPMV08] Juan Frausto-Solis, Federico Alonso-Pecina, and Jaime Mora-Vargas. An efficient simulated annealing algorithm for feasible solutions of course timetabling. In *MICAI 2008: Advances in Artificial Intelligence*, volume 5317 of *Lecture Notes in Computer Science*, pages 675–685. Springer Berlin / Heidelberg, 2008. ISBN 978-3-540-88635-8.

[GD91] David E. Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, pages 69–93, 1991.

[Gee09] Zong Woo Geem, editor. *Music-Inspired Harmony Search Algorithm: Theory and Applications*. 2009.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[GS01] Luca Di Gaspero and Andrea Schaerf. Tabu search techniques for examination timetabling. In *Practice and Theory of Automated Timetabling III*, volume 2079 of *Lecture Notes in Computer Science*, pages 104–117. Springer Berlin / Heidelberg, 2001. ISBN 978-3-540-42421-5.

[Hel04] Sabine Helwig. Erweiterung eines frameworks für zeitplanungsprobleme. Master's thesis, Universität Erlangen–Nürnberg, 2004.

[KH05] Graham Kendall and Naimah Mohd Hussin. An investigation of a tabu-search-based hyper-heuristic for examination timetabling. In *Multidisciplinary Scheduling: Theory and Applications*, pages 309–328. Springer US, 2005. ISBN 978-0-387-25266-7 (Print) 978-0-387-27744-8 (Online).

[Kil09] Helmut Killer. Entwurf und implementierung von algorithmen für zeitplanungsprobleme. Master's thesis, Universität Erlangen-Nürnberg, Germany, 2009.

[MFD07] M. Mahdavi, M. Fesanghary, and E. Damangir. An improved harmony search algorithm for solving optimization problems. 2007.

[MKM06] Muhammad Rozi Malim, Ahamad Tajudin Khader, and Adli Mustafa. Artificial immune algorithms for university timetabling. 2006.

[Sys91] G. Syswerda. A study of reproduction in generational and steady-state genetic algorithms. *Foundations of Genetic Algorithms*, pages 94–101, 1991.

[vL87] Peter J.M. van Laarhoven. *Simulated annealing*. D. Reidel Publishing Company, 1987.

[Whi89] Darrell Whitley. The genitor algorithm and selection pressure. *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116–121, 1989.

[Wil10] Peter Wilke. The Erlangen Advanced Time Tabling System (EATTS) Version 5. In Burke and McCollum [BM10], page submitted.

[WK10] Peter Wilke and Helmut Killer. Walk Up Jump Down - a new Hybrid Algorithm for Time Tabling Problems. In Burke and McCollum [BM10], page submitted.