Walk Down Jump Up - a new Hybrid Algorithm for Time Tabling Problems

Peter Wilke · Helmut Killer

Abstract A new trajectory time tabling algorithm is introduced. Because of it's nature the algorithm is called Walk Down Jump Up. The algorithm is described in a basic version and an advanced version. The performance of the algorithm when solving two real world problems is discussed.

Keywords timetabling · hybrid algorithm · time tabling algorithm · walk-down-jump-up

1 Introduction

Working with school or course time tabling problems leads to the usual "suspects" like Simulated Annealing, Tabu Search or Genetic Algorithms. If the achieved results are not acceptable then variants of these algorithms come into play trying to tailor the algorithm to fit the problem even better.

Here we would like to introduce a hybrid algorithm which is inspired by hill climbing, jumps and Simulated Annealing variants like Great Deluge [BBNP04].

Our algorithm is divided in two phases: first costs are reduced with a fast descending acceptance rate, second on stagnation the acceptance rate is increased at one go by a certain amount and phase one starts again and so on. That's why it is called Walk Down Jump Up.

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

2 Walk Down Jump Up Algorithm

Walk Down Jump Up is a trajectory based local search algorithm. Overall idea of the algorithm is based on changes of the acceptance rate. The idea for the algorithm comes on one hand from observing hillclimbing algorithm where only new solutions with better or equal costs than previous solution are accepted. Hillclimbing can reduce costs fast but then gets stuck in stagnation. On the other hand the idea is inspired by Great Deluge algorithm where new solutions are accepted when costs are below a falling cost limit. When the cost limit falls too fast then there will be stagnation, if it falls too slow no acceptable solution will be found in the estimated time. So our conclusion was that the acceptance rate should fall fast and on stagnation where all neighbour solutions to current solution have higher costs, it is necessary to set the acceptance limit to far above this surrounding cost values, so that it can jump up to a more far away solution and walk down to a new stagnation cost value that is hopefully lower than the stagnation cost before. The image in mind is that of the moon surface where many sinkholes exist and the task is to find one of the lowest. The astronaut walks down into the crater until lowest position is reached, then he jumps up above the border and tries to walk down into another crater which is hopefully deeper and so on.

2.1 Algorithm workflow

At the beginning an initial solution is generated randomly. At this stage the current best solution is equal to the initial solution and the jump distance is set to zero. Now a loop begins, if the solution isn't good enough. A new solution is generated by modifying the current solution using certain random move operators. If new solution has better or equal costs it is accepted and solutions with costs worse than the current solution are rejected. This leads typically to a local minimum and a stagnation phase. To get out of this trap the *jump operator* is enabled. When it is applied the threshold of accepted solution is increased from the costs for the current solution to a temporary limit. This limit is calculated by multiplying the current costs with an integer jump factor. After the Jump Operator is enabled the limit is decreased with every new iteration step by 1. If stagnation occurs again the current costs are compared with the costs of the last stagnation phase. If they are better, the solution is saved as new best solution, if they are worse but costs are in the same ball park as before, the search is continued according to the cost distance up to 10 times longer to finally yet get a better cost value. If the costs are still worse the jump factor is increased and the jump operator is applied again. Fig. 1 shows the control flow of the algorithm.

2.2 Random move operators

In each step the Walk Down Jump Up algorithm modifes current solution randomly to get a new solution with hopefully better costs. The initial solution is a random assignment of resources like students, teachers, rooms and timeslots to events. Modification of such solution means ressources are randomly assigned to or removed from some events, or directly exchanged. This leads to conflicts between the resources and

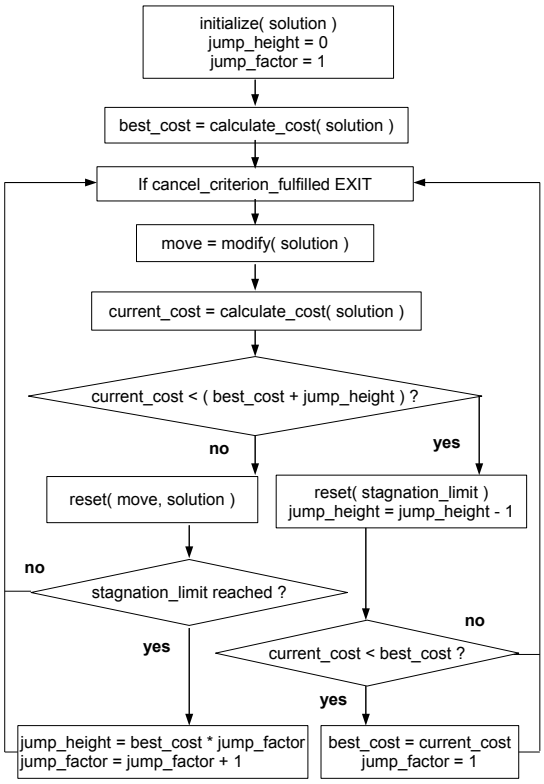


Fig. 1 Flow chart showing Walk Down Jump Up algorithm

violates constraints which increase solution costs or solves conflicts and satisfies constraints like "compact time table" which leads to lower costs. In one algorithm step only very few events are changed by a move operator and after cost evaluation the move is accepted or discarded.

2.3 Advanced Version - intelligent jump height

In the basic version of Walk Down Jump Up algorithm the jump height factor is increased by one after every walk down whose final cost is worse than before. This leads to more higher jumps after every algorithm walk down until new best cost is reached and jump height is reset to one. In the advanced version of the algorithm the jump height values which were successful are saved in a history list. When a new jump height has to be determined, the jump height factor is chosen randomly from the history list or a random value between 1 and the specified maximum jump height factor that is e.g. 10 times current cost value. The experiments show that the advanced version is able to reach better final costs than basic version.

3 Experiments and Results

Walk Down Jump Up was developed when we worked on two real world problems, namely school timetabling and univerity course time tabling. Walk Down Jump Up has been implemented using the current version of EATTS (Erlangen Advanced Time Tabling System [Wil10]). Because also many other algorithms like Simulated Annealing, Genetic Algorithm and Tabu Search are available in EATTS the algorithm performance can easily be compared.

3.1 The Problems

3.1.1 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. Beside calculating a feasible solution the main constraint is to also generate a timetable as compact as possible.

Events:	178
Resources of type TimeSlot:	81
Resources of type Class:	14
Resources of type Teacher:	28
Resources of type Room:	37
Resources of type Subject:	78
Resources of type LessonProperties:	178

Table 1 The main characteristics of the School 2009 example

3.1.2 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. Each student has a list of 4 preferred courses and can declare 4 friends with whom she wants to share the same courses.

Events:	229
Resources of type TimeSlot:	57
Resources of type Subject:	52
Resources of type Girl:	170

Table 2 The main characteristics of the MuT 2009 example

3.2 Results

Figure fig. 2 shows the performance when solving the MuT 2009 problem. The red line indicates the current best solution, while the green dots show current solution, which can be worse then the best solution because of the jump operator and/or the random changes made to find new solutions. When the runtime is limited to 1 second Walk Down Jump Up shows a fast descent of costs followed by a long phase of slow descent. When applied to the MuT 2009 problem Walk Down Jump Up performed equally good to Simulated Annealing [vL87, FSAPMV08] and slightly better than Tabu Search [GS01, KH05] and Immune Systems [MKM06].

On the School 2009 problem Walk Down Jump Up achieved the best results together with Harmony Search [ABKG08, Gee09], Simulated Annealing, Great Deluge [BBNP04] when long runs, i.e. several hours, are evaluated. On short runs, i.e. several seconds, Walk Down Jump Up was head on with Tabu Search and Great Deluge, but all three were outperformed by Simulated Annealing. For a more detailed comparison of Walk Down Jump Up with six other algorithms see [WK10]

4 Conclusion

Walk Down Jump Up has proven that it is able to solve real world problems efficient and effective. But the results have also shown that there is still room for improvements.

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.

[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.

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

[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.

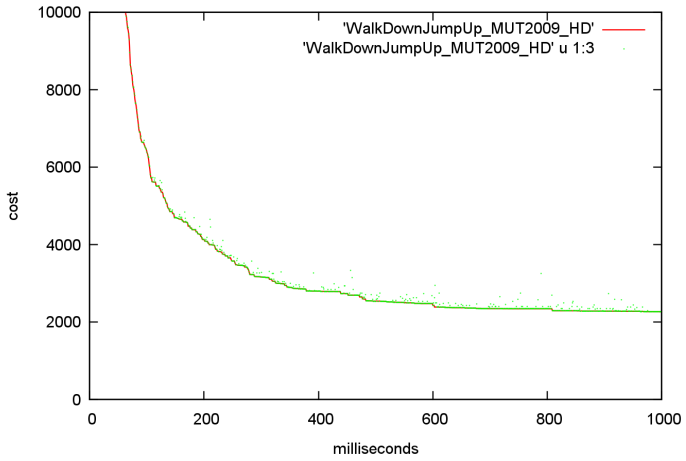
[KH05] Graham Kendall and Naimah Mohd Hussin. An investigation of a tabu-search-based hyper-heuristic for examination timetabling. In *Multidisciplinary Scheduling: Theory and Applications*, pages 309–328. Springer US, 2005. ISBN 978-0-387-25266-7 (Print) 978-0-387-27744-8 (Online).

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

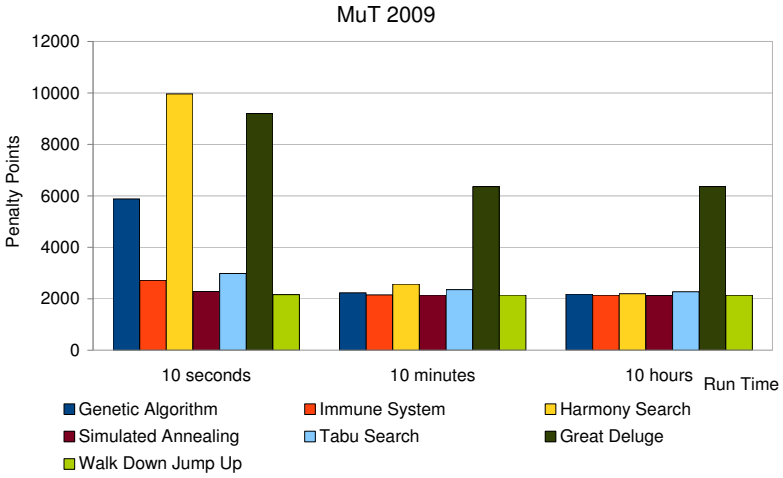
[vL87] Peter J.M. van Laarhoven. *Simulated annealing*. D. Reidel Publishing Company, 1987.

[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. Comparison of Algorithms solving School and Course Time Tabling Problems using the Erlangen Advanced Time Tabling System (EATTS). In Burke and McCollum [BM10], page submitted.

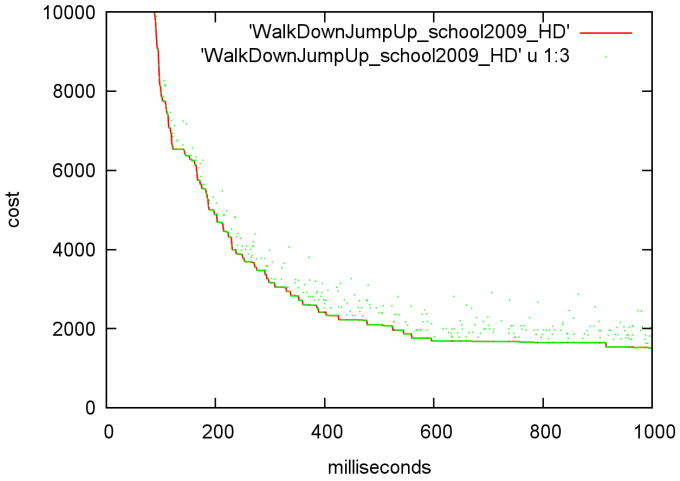


(a) Walk Down Jump Up on MuT 2009 with runtime 1 second

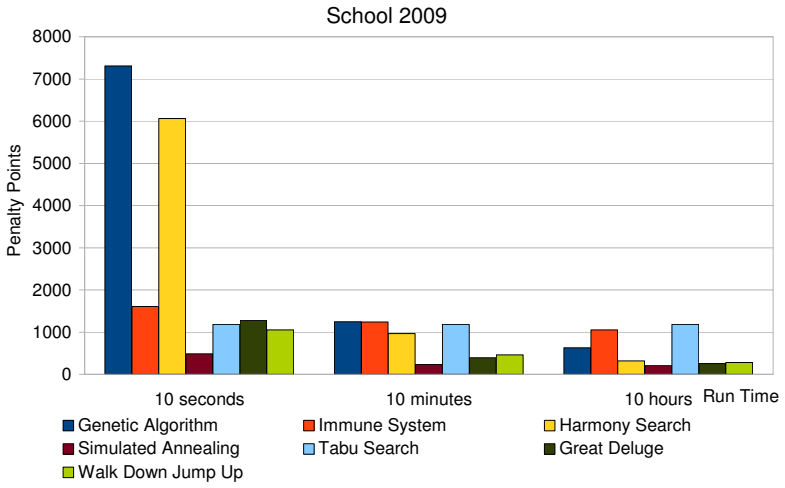


(b) Walk Down Jump Up on MuT 2009 with runtimes 10 second, minutes, hours

Fig. 2 Walk Down Jump Up solving the MuT 2009 example



(a) Walk Down Jump Up on School 2009 with runtime 1 second



(b) Walk Down Jump Up on School 2009 with runtimes 10 second, minutes, hours

Fig. 3 Walk Down Jump Up solving the School 2009 example