A Late Acceptance Strategy in Hill-Climbing for Exam Timetabling Problems

Edmund K. Burke, Yuri Bykov

Automated Scheduling, Optimisation and Planning Group,
School of Computer Science & IT,
The University of Nottingham, UK
Phone: +44 (0)115 846 7663
Email: {ekb,yxb}@cs.nott.ac.uk

Over the years, many variants, extensions and adaptations of local search techniques have appeared in the literature. Some of them have become extremely famous, such as Simulated Annealing. Other ones have almost been forgotten, for example, "Record-to-Record Travelling" (Dueck 1993) or the "Old Bachelor Acceptance Algorithm" (Hu et al. 1995). In this abstract, we are proposing another new local search strategy. It is simple and, in many ways, rather obvious. We have made an exhaustive search through the optimization literature, but have not found the same idea in a published paper. We have termed this method the "*Late Acceptance Strategy*". However, if it eventually emerges that the same approach has already been published then this paper could be considered to be a reminder about a "forgotten" way of improving a local search procedure.

As the basis of our proposed method, we take a simple Hill-Climbing algorithm. Although its performance is known to be relatively worse than that of more sophisticated metaheuristics, it is still very popular thanks to its simplicity. It is also widely used in different hybridizations, such as guided, multi-start or variable neighborhood search methods (see Glover and Kochenberger 2003; Burke and Kendall 2005). Hill Climbing is an iterative process. At each iteration, a current solution is used to determine the acceptance of a new candidate. In other words, a candidate solution is compared with a current one and accepted when its cost function is not worse. Our idea is to delay this comparison, namely: *to compare the candidate solution with a solution, which was "current" several steps before*. Here, each current solution still takes on the role of an acceptance benchmark, but it will be used at *later* steps.

In a similar way to Tabu Search, this algorithm maintains a list of a given length L. However, the list contains no information about executed moves, but it does contain previous values of the cost function $\hat{C}_k$, where $k \in \{1...L\}$. At each iteration, a candidate cost is compared with the last element of the list $\hat{C}_L$. The comparison method is the same as in Hill-Climbing, i.e. the candidate solution is accepted when its cost is equal to or better than $\hat{C}_L$. After the comparison and the acceptance procedure, the new current cost (it can be equal either to the previous cost or to the cost of the accepted candidate) is inserted into the beginning of the list (for later comparison), and correspondingly, the last element is removed from the list. It could be also suggested that, at the beginning of the search, all elements of the list are the same and are equal to the initial cost function.

Although, in Tabu Search, the increase of the length of the tabu list always causes higher computational expense, the Late Acceptance strategy is free from this drawback. First, we do not have to enumerate the complete list at each step: just shift forward its elements. Second, we can eliminate the shifting by simple recalculation. Here the "physical" list remains static, but its "virtual" beginning v is calculated dynamically as a reminder of the division of the current number of iteration I to the length of the list L ($v = I \bmod L$). Thus, the search can maintain a list of any length without extra expense. The complete search procedure is summarized in Algorithm 1.

Algorithm 1 General purpose Late Acceptance Hill-Climbing methodology (LAHC)

```
Produce an initial solution $s$
Calculate initial cost function $C(s)$
for all $k \in \{0...L-1\}$ do $\hat{C}_k \leftarrow C(s)$
Assign the initial number of iteration $I \leftarrow 0$;
do until a chosen stopping condition:
  Construct a candidate solution $s^*$
  Calculate its cost function $C(s^*)$
  $v \leftarrow I \bmod L$
  if $C(s^*) \leq \hat{C}_v$
  then accept candidate ($s \leftarrow s^*$)
  Insert cost value into the list $\hat{C}_v \leftarrow C(s)$
  Increment the number of iteration $I \leftarrow I+1$
end do
```

We can indicate several motivations behind the Late Acceptance strategy. Firstly, this method allows some worsening moves, which (as might be expected) should prolong the search time and simultaneously help to avoid local minima. Secondly, it depends on a single genuine parameter, i.e. the length of the list *L* (rather than a function, like in Simulated Annealing or the Great Deluge Algorithm). Therefore, it could be seen to be less vulnerable to inadequate parameterization. Thirdly, it draws upon an idea of Laguna and Glover (1996) of the "intelligent" use of information, collected during the search (but in a different way than that used in Tabu Search). Here, the list of previous cost functions determines a pattern for further decisions, and this pattern reflects specific properties of a current problem's neighborhood. In this context, it can be viewed as a yet unstudied variant of Adaptive Memory Programming (see Taillard et al. 2001).

We investigated the performance of the proposed method by applying it to Exam Timetabling problems using a model from our previous studies (see Burke et al. 2004; Burke and Bykov 2008). It starts from the Saturation Degree initialization procedure. Afterwards an iterative search is used, which performs the following moves: we move a random exam into a random timeslot (using Kempe chains in the case of infeasibility) and we swap two randomly chosen timeslots. The search is run for as long as it is able to improve a current solution, i.e. we stop it when no further improvement is possible. In the present study, convergence is considered to have taken place after 50000 idle moves. However, optimizing this parameter is the subject of further investigation (see the discussion below). 13 benchmark problems were taken from the University of Toronto collection. The notation from Qu et al. (2009) is used. Statistics were collected during three experiments, while running each experiment 20 times over each problem.

In the first experiment, we estimated a lower bound of the performance of our method, i.e. when L=1 (which corresponds to pure Hill-Climbing). The second experiment was carried out with L=500 for all problems. In the third experiment, we have empirically customized the length of the list for each benchmark problem in order to run the search for approximately 10 minutes. All results (best and average costs, average run times and *L* in the third experiment) are presented in Table 1.

Table 1 Performance of HC and LAHC for benchmark problems

Problem	Pure HC (L=1)		LAHC with L=500		LAHC with custom list length		
	Result (best/av)	Av. time (sec)	Result (best/av)	Av. time (sec)	L	Result (best/av)	Av. time (sec)
Car92	4.33/4.52	34	3.93/4.08	184	2000	3.81/3.92	604
Car91	5.24/5.46	55	4.77/4.89	329	1300	4.58/4.68	665
Ear83 I	36.62/37.94	3	33.22/34.13	18	14000	32.65/32.91	450
Hec92I	10.94/11.60	1	10.32/10.70	2	120000	10.06/10.22	590
Kfu93	13.99/14.72	6	13.02/13.40	113	5000	12.81/13.02	882
Lse91	11.11/12.02	5	10.08/10.53	150	4000	9.86/10.14	641
Pur93	4.88/5.08	584	4.32/4.39	3332	100	4.53/4.71	747
Rye92	8.79/9.15	9	8.17/8.36	68	5000	7.93/8.06	901
Sta83I	157.17/157.51	1	157.03/157.13	2	60000	157.03/157.05	587
Tre92	8.74/8.99	7	8.09/8.25	38	9000	7.72/7.89	608
Uta92I	3.62/3.72	43	3.29/3.37	219	1500	3.16/3.26	805
Ute92	25.57/26.45	1	24.87/25.03	8	30000	24.79/24.82	528
Yor83I	38.07/39.27	3	36.34/37.17	15	19000	34.78/35.16	502

As we expected, the increase of L increases the computational cost and simultaneously helps to achieve much better solutions. In Table 2 we give a brief comparison of our results with best previously published ones using the survey of Qu et al. (2009).

Table 2 Comparison of LAHC with best published results

Problem	Carter et al. 1996	Casey and Thompson 2003	Yang and Petrovic 2005	Burke et al. 2009	Caramia et al. 2008	LAHC
Car92	6.2	4.4	3.93	4.0	6.0	**3.81**
Car91	7.1	5.4	**4.5**	4.6	6.6	4.58
Ear83I	36.4	34.8	33.7	32.8	**29.3**	32.65
Hec92I	10.8	10.8	10.83	10.0	**9.2**	10.06
Kfu93	14.0	14.1	13.82	13.0	13.8	**12.81**
Lse91	10.5	14.7	10.35	10.0	**9.6**	9.86
Pur93	**3.9**	-	-	-	-	4.32
Rye92	7.3	-	8.53	-	**6.8**	7.93
Sta83I	161.5	**134.9**	158.35	159.9	158.2	157.03
Tre92	9.6	8.7	7.92	7.9	9.4	**7.72**
Uta92I	3.5	-	**3.14**	3.2	3.5	3.16
Ute92	25.8	25.4	25.39	24.8	**24.4**	24.79
Yor83I	41.7	37.5	36.35	37.28	36.2	**34.78**

It can be seen that this simple and straightforward approach produces strong results.

Finally we should point out some general issues.

- We have presented here a very early study of a new proposed technique. Obviously, its properties require further investigation. For example, when the list is quite long the presented algorithm tends to make an extremely slow improvement in the final phase of the search. We have observed that sometimes 50000 idle moves were not sufficient for the recognition of "true" convergence. This number should probably be calculated as a percentage of the total number of moves, but this issue needs to be more deeply studied.
- In addition to the proposed algorithm, the basic idea of the Late Acceptance approach could be implemented in different ways. For example, we can imagine an alternative variant where the benchmark cost is not taken from the end of the list, but chosen randomly over all its elements.

- We have applied here the proposed technique to Exam Timetabling problems. However, we think that the Late Acceptance technique could be effective as a general purpose strategy, which can be applied to any problem where Hill Climbing is applicable. We have already tested it on Grid Scheduling problems and it showed a high level of performance.
- We presented here the Late Acceptance strategy applied within Hill Climbing. However it can be embedded into any search method, where a candidate cost is compared with a current one. For example, we could propose the use of the Late Acceptance strategy with Simulated Annealing, Functional Annealing or Threshold Acceptance methods.

References

Burke, E. K., Bykov, Y., Newall, J., & Petrovic, S. (2004). A time-predefined local search approach to exam timetabling problems. *IIE Transactions*, *36*(6), 509-528.

Burke, E. K., & Kendall, G. e. (2005). *Search methodologies: introductory tutorials in optimization and decision support techniques*, Springer: New York.

Burke, E. K., & Bykov, Y. (2008). An adaptive flex-deluge approach to university exam timetabling. Submitted to *INFORMS Journal of Computing.*

Burke, E. K., Eckersley, A. J., McCollumn, B., Petrovic, S., & Qu, R. (2009, to appear). Hybrid variable neighborhood approaches to university exam timetabling. *European Journal of Operational Research*.

Caramia, M., Dell'Olmo, P., & Italiano, G. (2008). Novel local search based approaches to university examination timetabling. *INFORMS Journal of Computing*, *20*, 86-99.

Carter, M. W., Laporte, G., & Lee, S. (1996). Examination timetabling: algorithmic strategies and applications. *Journal of Operational Research Society*, *47*, 373-383.

Casey, S., & Thompson, J. (2003). GRASPing the examination scheduling problem. *PATAT IV, Selected Revised Papers,* Springer LNCS 2740, 232-246.

Dueck G. (1993). New optimization heuristics. The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, *104*, 86-92.

Glover, F., & Kochenberger, G.A. (2003). *Handbook of metaheuristics*. Kluwer Academic Publishers: Dordrecht.

Hu, T. C., Kahng, A. B, & Tsao, C.-W. A. (1995). Old bachelor acceptance: a new class of nonmonotone threshold accepting methods. *ORSA Journal on Computing, 7*, 417-425.

Laguna, M., & Glover, F. (1996). What is tabu search? *Colorado Business Review, 61*, 5-12.

Qu, R., Burke, E. K., McCollumn, B., Merlot, L. T. G., & Lee, S. Y. (2009 to appear). A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*.

Taillard, E. D., Gambardella, L. M., Gendreau, M., & Potvin J.-Y. (2001). Adaptive memory programming: a unified view of metaheuristics. *European Journal of Operational Research, 135*, 1-16.

Yang, Y., & Petrovic, S. (2005). A novel similarity measure for heuristic selection in examination timetabling. *PATAT V. Revised Selected Papers.* Springer LNCS 3616, 377-396.

Summary This work presents the Late Acceptance Randomized Descent Algorithm (LARD) and the Late Acceptance Strategy in Hill Climbing (LAHC) to solve university course timetabling problems. The aim of this work is to produce an effective algorithm for assigning a set of courses (events) and students to a specific number of rooms and timeslots, subject to a set of constraints. LAHC approach was originally introduced by Burke and Bykove for exam timetabling problem. LAHC can be embedded into any search method. LARD differs from the basic LAHC, as it use randomized decent method instead of using s... 2008. A late acceptance strategy in hill-climbing for exam timetabling problems. Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling PATAT 2008, Montreal, Canada, August 2008. Bykov, Y., S. Petrovic. 2013. An initial study of a novel Step Counting Hill Climbing heuristic applied to timetabling problems. Proceedings of the 6th Multidisciplinary International Scheduling Conference MISTA 2013, Gent, Belgium, August 2013. Carter, M.W., G. Laporte, S. Lee. 1996. Examination timetabling: algorithmic strategies and applications. Journal of the Operation... The well-known Late Acceptance Hill Climbing (LAHC) search aims to overcome the main downside of traditional Hill Climbing (HC) search, which is often quickly trapped in a local optimum due to strictly accepting only non-worsening moves within each iteration. In contrast, LAHC also accepts wors-ening moves, by keeping a circular array of tness values of previously visited solutions and comparing the tness values of candidate solutions against the least recent element in the array.Â 2 Late Acceptance Hill Climbing. Local search algorithms start from an initial solution S0.Â We have empirically observed that for some problems LAHC unfortunately behaves in a similar manner to HC and does not accept worsening moves. Figs. I have implemented Simulated Annealing and I am interested in comparing the results to Late Acceptance Hill Climbing. I have found some pseudo code for Late Acceptance but need a little help to write it in Java: Produce an initial solution s Calculate initial cost function C(s) Set the initial number of steps I=0 For all k in(0.. L-1) C_hat[k]=C(s) Do until a stopping condition Construct a candidate solution s* Calculate its cost function C(s*) v = I mod L If C(s*) <=. C_hat[v] Then accept s* Insert cost value into the list C_hat[v] = C(s) Increment a number of steps I=I+1 End do. I really don't get this bit: For all k in (0.. L-1) C_hat[k]=C(s). The pseudo code is from http://www.yuribykov.com/LAHC/LAHC_wonders.pdf. java algorithm hill-climbing. The late acceptance hill-climbing (LAHC) algorithm is a single-point iterative search algorithm proposed by [8] that accepts non-improving movements when the objective function value of a candidate has a better value than the one number of iterations before. Namely, while in the standard hill-climbing algorithm a candidate solution is compared to the direct predecessor, in LAHC the candidate is compared with the current solution several iterations before. In the following, some of them are briefly reviewed. The authors of [8] apply the LAHC to exam timetabling problems reporting a competitive performance. This problem is later addressed by [50] where they present a hybridization of an adaptive artificial bee colony (ABC) and LAHC. ...