An Application of Late Acceptance Hill-Cimbing to the Traveling Purchaser Problem

Article *in* Lecture Notes in Computer Science · September 2013

DOI: 10.1007/978-3-642-41019-2_13

3 authors:

Andreas Goerler
University of Hamburg
2 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Frederik Schulte
University of Hamburg
13 PUBLICATIONS **24** CITATIONS
SEE PROFILE

Stefan Voss
University of Hamburg
434 PUBLICATIONS **5,096** CITATIONS
SEE PROFILE

An Application of Late Acceptance Hill-Climbing to the Traveling Purchaser Problem

Andreas Goerler, Frederik Schulte and Stefan Voß

Institute of Information Systems (IWI), University of Hamburg, Von-Melle-Park 5, 20146 Hamburg, Germany
{a.goerler@gmx.de,frederik.schulte@uni-hamburg.de, stefan.voss@uni-hamburg.de}

Abstract. Late Acceptance Hill Climbing (LAHC) is a recent metaheuristic in the realm of local search based procedures. The basic idea is to delay the comparison between neighborhood solutions and to compare new candidate solutions to a solution having been current several steps ago. The LAHC was first presented at the PATAT 2008 conference and successfully tested for exam timetabling, the traveling salesman problem (TSP) and the magic square problem and the results seemed extraordinary. The purpose of this paper is to analyze the behavior of the method and to provide some extended understanding about its success and limitations. To do so, we investigate the method for a generalized version of the TSP, the traveling purchaser problem.

Keywords: Metaheuristic, Late Acceptance Hill-Climbing, Traveling Purchaser Problem.

1 Introduction

The Traveling Purchaser Problem (TPP) is a well-known generalization of the Traveling Salesman Problem (TSP) [13, 15, 17, 21] with wide applicability [10, 18] and it occurs in many real-world applications related to routing, warehousing and scheduling. Starting from home a purchaser can travel to a set of markets providing different products at different prices. The aim is to fulfill a shopping list of products while minimizing the sum of traveling and purchasing costs. We apply a recent metaheuristic to the TPP, namely the Late-Acceptance Hill-Climbing (LAHC) heuristic. LAHC may be regarded as a simplification or modification of simulated annealing or threshold accepting. It was originally introduced by Burke and Bykov [5] and it won the 1st prize in the International Optimization Competition. LAHC is a one-point iterative search procedure with the general idea of delaying the comparison between neighborhood solutions. The intention behind this late acceptance strategy is to avoid the problem of getting stuck in a local optimum that many greedy search procedures have. The method was

successfully tested for some problems and the purpose of this paper is to examine if an application of the LAHC to the TPP is also successfully possible.

The investigation of the properties of late acceptance strategies (LAS) is still in an early stage. Consequently, only few contributions in literature deal with LAS for heuristics and metaheuristics. Abuhamdah [1] proposed LAHC and a randomized version for solving course timetabling problems. In that case the randomized version performed better than the originally presented LAHC of [5]. Özcan et al. [14] presented a set of hyper-heuristics utilising different heuristic selection methods combined with LAS for examination timetabling. Verstichel and Berghe [20] applied LAS to a lock scheduling problem. Their experiments showed that the application of the late acceptance criterion within their local search heuristic led to an improvement of the solution quality in every instance.

The remainder of this paper is organized as follows. In Section 2 we describe the TPP in more detail. In Section 3 we explain LAHC and show how to possibly apply it while using different neighborhoods. Finally, examples of computational experiments are reported.

2 The Traveling Purchaser Problem

Consider a set $V = \{1, 2, \ldots, m\}$ of m markets, a set $K = \{1, 2, \ldots, n\}$ of n products and a domicile (or home-market) $s \in V$. Let c_{ij} (with $i, j \in V$) denote the cost of travel from market i to market j. In the symmetric TPP it is assumed that $c_{ij} = c_{ji}\ \forall\, i, j$. Every product k ($k \in K$) is available in a subset of markets $V_k \subseteq V_s$ (with $V_s := V - \{s\}$) and if a product k is available at market i, p_{ik} presents the cost of k at i. Otherwise, p_{ik} is set to a prohibitively large number M. It is implicitly assumed that if a product k is available at market i, its available quantity q_{ki} is sufficient to satisfy the demand d_k [15]. This model formulation represents a symmetric uncapacitated version of the TPP. The overall idea of the TPP is to generate a tour through a subset of the m markets starting and ending at s, such that all n products are purchased while minimizing the sum of travel and purchasing costs.

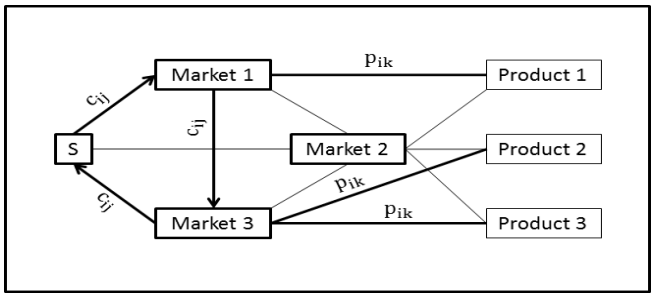

Fig. 1. Example of the TPP (cf. [21])

Figure 1 visualizes the TPP. In this example we have a cycle through a subset of the three markets, starting at the domicile and visiting markets 1 and 3. Product 1 is purchased at market 1 and products 2 and 3 at market 3, shown in boldface.

We assume that each product is available in at least one market and that the purchaser may pass through a market any number of times without purchasing. Or, he may purchase as many products as there are available at each market. A capacitated version of the TPP assumes that the available quantities q_{ki} might be insufficient to satisfy the demand, i.e. it is assumed for each $k \in K$ that q_{ki} and d_k satisfy $0 < q_{ki} \leq d_k$ $(\forall i \in V_k)$ and $\sum_{j \in V_i} q_{kj} \geq d_k$ [16]. Other modifications of the TPP with additional capacity constraints, like a limit on the maximum number of markets to be visited and a limit on the number of items bought per market are also important [9].

The TPP is known to be NP-hard, since it reduces to the NP-hard TSP if each product is only available at one market and each market carries only one product. This complexity indicates that only problems of moderate size can be solved by an exact method, a suggestion that is strengthened by the scarce number of literature dealing with exact methods for the TPP. Exact algorithms proposed in the literature include a branch-and-bound algorithm calculating a lower bound by solving a relaxation similar to a simple plant location problem [18]. Branch-and-cut procedures can be found in [12, 17] being able to solve instances up to 200 markets and 200 products. An approach based on constraint programming and Lagrangean relaxation is presented in [6] improving some of the earlier approaches. Despite these contributions, the literature on the TPP is mostly directed towards the development of heuristics and metaheuristics; see, e.g., [8, 13, 19, 2]. This includes extensions of methods known from the TSP like the famous nearest neighbor method (NN) [3] as well as classical add and drop procedures [21]. First metaheuristics for the TPP were proposed in [21] showing that simulated annealing was outperformed by different dynamic tabu search strategies. Other approaches also include tabu search [4], a local search algorithm with two neighborhoods [16] and a transgenetic algorithm inspired by horizontal gene transfer and endosymbiosis [7].

3 Late Acceptance Strategy

LAHC is based on a simple hill-climbing algorithm. The procedure starts from a single initial solution and iteratively modifies it to produce candidate solutions. The simple hill-climbing algorithm is a greedy procedure that is regarded to be very fast but with poor results as it tends to get stuck in a local optimum very quickly. To overcome this problem, the basic idea of LAHC is to delay the comparison of solutions by "memorizing" previous current solutions in a list of a particular length, called fitness array F_a of length L_{fa} ($F_a = \{f_0, f_1, \ldots, f_{L_{fa-1}}\}$ with f_a denoting the cost function value of solution a. Contrary to pure hill-climbing (note that LAHC with $L_{fa} = 1$ degenerates into pure hill-climbing) LAHC compares a candidate not to its direct previous current solution but to

the last element of the fitness array. Therefore, the LAHC accepts a decline of the objective function value by a candidate in comparison to the direct previous solution if the candidate has the same or a better objective function value than the solution at the end of the list. If accepted, the candidate solution is put on the list and the last element is removed from the list. To eliminate the shifting of the whole list the authors propose the use of a "virtual" shifting of the fitness array. Therefore, they compute the virtual beginning v at the I^{th} iteration by $v = I \bmod L_{fa}$, i.e., v is equal to the division of the current number of iterations I to the length of the list L_{fa}. Now the candidate solution is compared with the cost function f_v and if accepted the value of the candidate solution is assigned to f_v.

3.1 Initial Solutions

As described above the LAHC starts from a single initial solution. In case of the TPP this solution is represented by the cost function value of an initial tour through the considered graph. An initial tour for the TPP is determined by applying the NN to all markets without considering any purchases. The NN method is described as follows: Starting at the depot s the purchaser goes to the market with the lowest travel costs from the depot. From there he visits the nearest market that was not visited before. The procedure is repeated until there are no unvisited markets left and the purchaser returns to the depot. The obtained TSP tour can be transformed into a TPP tour by buying each product at the market where it is available at the lowest possible price. An alternative approach is described by Burke and Bykov [5]. They build an initial feasible path for a TSP instance and randomly exchange markets to ensure different initial solutions. Starting with such an inital tour we apply a simple drop procedure [21]. In each step it drops a market giving the largest reduction to the objective function value. If no more reduction is possible or infeasibility occurs, the procedure is terminated and the obtained solution serves as the initial solution ($f(\sigma)$) for the LAHC algorithm.

3.2 Neighborhood Definitions

Several neighborhoods may be applied. The first approach is the IMP1-procedure from [21]. Starting with a feasible solution in each iteration exactly one market is excluded from the tour (called drop step) whose removal gives the best improvement or least deterioration of the cost function value f (note that strategic oscillation into infeasibility is permitted). Afterwards a number of consecutive insertions (add steps) are performed as long as an improvement in the cost function value is achieved. IMP1 is terminated after a complete iteration, i.e., when a market that has been previously removed was added again. This set of moves defines a neighborhood of candidate solutions.

The l-ConsecutiveExchange procedure from [17] represents a generalization of IMP1. Given a feasible solution σ, a starting value l is chosen ([17] obtained best results for $2 \leq l \leq 25$). Afterwards the procedure has two steps,

namely l-ConsecutiveDrop and a step for restoring feasibility if needed. For l-ConsecutiveDrop l markets are selected according to an estimation of the trade-off between the travel cost reduction occuring by dropping the l markets and the increase in purchasing costs due to the fact that the products that were purchased in the dropped l markets have to be bought elsewhere. These trade-off evaluations are performed for each possible path consisting of $l+1$ consecutive edges belonging to the initial tour. The different paths are ranked according to the travel cost reduction minus the purchase costs increase and the path with the highest rank is dropped. After dropping the l markets NN is applied to reduce the travel costs of the new (possibly infeasible) tour σ.

If feasibility has to be restored, new markets are added. To guarantee the generation of a diverse neighborhood only markets are permitted to be added that have not been in the initial feasible tour. The restore feasibility step proceeds with computing subsets of permitted markets ($V_p \subseteq V$). For the capacitated TPP the non-satisfied amount of each product k in the infeasible tour has to be calculated and only subsets of markets are permitted that are selling the required quantity of product k needed to restore feasibility. For each market $i \in V_p$ the travel cost increase ($\rho(i, \sigma)$), describing the increase in travel costs if market i is added to σ, and the decrease in purchasing costs ($\mu(i, \sigma)$) for adding i is computed as product k may be available at a cheaper price at market i. Markets in the selected subset V_p are added one after another according to their trade-off between travel cost increase and purchase cost decrease. Finally, we apply LAHC to possibly re-optimize the obtained tour.

3.3 LAHC Algorithms

The general approach of the LAHC algorithms starts with the calculation of an initial solution. Afterwards the length of the fitness array L_{fa} and an initial l are specified. An inner loop performs an iterative procedure to remove l-consecutive markets from the initial tour σ. If the removal of the l markets leads to an infeasible tour the following add procedure grants that the feasibility is restored. This procedure randomly adds markets from a list of earlier dropped markets at random position within the tour of the candidate solution. To discourage staying at local optima the procedure does not allow the add of markets dropped in the current iteration. Markets are added until the solution of the candidate tour is feasible and its penalty can no longer be reduced. Finally, the method LAHCTSP tries to improve the order of markets in the candidate tour. After building the neighborhoods LAHC is called. If the cost function value of the candidate $f(\sigma')$ is better or equal to the value of the solution f_v with v being the solution at the beginning of the virtual list the candidate is accepted and replaces f_v; otherwise it is rejected. DeltaPenalty follows the idea of an incremental evaluation to reduce CPU time, where only changes in the cost function values are calculated and compared to changes in cost function value of neighborhood solutions rather than calculating the complete objective function value in every iteration. After accepting or rejecting a candidate solution the iteration number I is set to $I + 1$.

```
Input: A TPP instance
σ := InitialTSP (V)
σ := InitialDrop (σ)
Calculate initial cost function f(σ)
Specify L_fa
For all k ∈ {0...L_fa−1} f_k := f(σ)
Specify l
First iteration I = 0
σ' := σ
repeat
   while l ≥ 1 do
      σ'' := σ'
      σ' := l - ConsecutiveDrop (σ, l)
      if f(σ'') ≥ f(σ') or σ'' not feasible then
         l := l − 1
      else
         σ' := σ''
      end if
   end while
   repeat
      σ'' := σ'
      σ'' : AddRandomMarket (σ'')
   until σ'' ≥ σ' AND σ'' is feasible
   σ' := LAHCTSP (σ'')
   f(σ') = f(σ) + DeltaPenalty (σ)
   v := I mod L_fa
      if f(σ') ≤ f(σ) or f(σ') ≤ f_v then
         σ := σ'
         f_v := f(σ)
         I := I + 1
until stopping condition
return σ
```

Fig. 2. Pseudocode of the LAHC Algorithm

We developed two different types of LAHC algorithms to solve the TPP. The first algorithm (LAHC, see Figure 2) strictly applies the list approach of LAHC for the evaluation of candidate solution and the LAHCTSP method. The candidate solution is created by a sequence of a drop step, an add step and a TSP heuristic and accepted if the LAHC constraints are fulfilled. The second algorithm (sLAHC) simplifies this idea and accepts changes after an evaluation in the drop step, add step and the TSP heuristic.

4 Computational Results

The algorithms described in Section 3.3 are implemented in Java and run on a Intel Core 2.50GHz with 8 GB RAM. We tested the performance of the two algorithms on the instances of Class 1 (http://webpages.ull.es/users/jriera/TPP.htm) defined by Singh and van Oudheusden [18]. This class contains 33-market symmetric instances and the product prices are generated according to a uniform distribution in an interval of [1,500]. The routing costs are taken of the 33-vertex TSP described in Karg and Thompson [11] and the first vertex corresponds to the depot. All markets sell all products and the quantity of products varies between 50 and 500. The algorithms runs were aborted after a number of 15000 iterations for every instance.

25 instances with 50 to 250 products have been solved to optimality (note that the optimal values for these instances are also provided at the webpage mentioned above). Table 1 compares the results of the sLAHC and the LAHC algorithm to these optimal values. Optimal values for the instances with 300 to 500 products have not yet been presented. In Table 2 we provide the objective function values for 25 instances out of this range with a number of products from 300 up to 500. The results show, that both algorithms achieve optimal results or small optimality gaps for almost all tested instances and outperform known results from literature (see Table 3). Especially for small instances the sLAHC algorithm seems to perform better than the LAHC algorithm. The LAHC tends to get stuck in a local optimum quickly for those instances. The sLAHC, in contrast, has a lower risk to stay early at a local optimum but seems to be weaker for larger instances. For small instances the LAHC fitness array seems to guide the LAHC algorithm to local optima while the sLAHC can come closer to the global optimum without the guidance of a LAHC fitness array. For large instances this disadvantage seems to turn into an advantage and the LAHC seems benefit from the LAHC mechanism. The sLAHC achieves optimal solutions for all five instances with 50 products and performs more homogenously over all instances without any spikes. Note that the results in Table 1 were created with a single batch run. For small instances the LAHC results highly depend on the initial feasible solution, which were generated randomly. Thus exceptionally large optimality gaps might occur. The average gaps are usually much lower like, e.g., for instance 3 another run led to a 0% instead of a 14% optimality gap.

The column headings in the tables are described as follows:

sLAHC	:	simplified Late Acceptance Hill Climbing;
$LAHC_{5000}$	:	Late Acceptance Hill Climbing with $L_{fa} = 5000$;
$LAHC_{10000}$	:	Late Acceptance Hill Climbing with $L_{fa} = 10000$;
V	:	number of markets;
K	:	number of products;
V^*	:	number of markets involved in the best solution;
V^*_{LA}	:	number of markets involved the best solution of the LAHC;
$\%gap$	:	quality of the LAHC over an optimal solution;

Table 1. Numerical results Class 1 (50 - 250 products)

			sLAHC		$LAHC_{5000}$		$LAHC_{10000}$	
V	K	V^*	$\%gap$	V^*_{LA}	$\%gap$	V^*_{LA}	$\%gap$	V^*_{LA}
33	50	9	0	9	1,021	9	1,986	8
33	50	8	0	8	1,019	8	3,226	9
33	50	8	0	8	14,108	8	14,174	8
33	50	8	0	8	12,726	7	1,593	8
33	50	9	0	9	8,660	8	9,695	9
33	100	11	0,237	11	1,039	10	0,289	9
33	100	10	0,703	10	4,104	11	8,940	10
33	100	11	0,971	10	3,434	10	3,800	10
33	100	11	0,94	11	5,886	10	2,736	10
33	100	9	1,232	11	1,613	10	3,429	10
33	150	12	0,95	11	3,316	12	2,506	12
33	150	14	1,928	12	2,079	13	3,000	12
33	150	13	0,771	11	2,785	12	0,268	12
33	150	13	1,659	12	4,632	13	2,575	12
33	150	14	1,864	13	3,845	12	3,706	12
33	200	12	2,795	13	3,783	13	1,863	13
33	200	14	2,227	12	4,410	14	4,093	12
33	200	14	0,778	15	3,689	14	4,943	14
33	200	15	2,792	11	4,352	14	3,603	14
33	200	14	1,177	12	7,648	13	3,199	13
33	250	15	1,915	14	3,238	15	1,954	15
33	250	15	0,958	16	2,300	15	3,874	16
33	250	15	4,846	15	5,872	16	4,669	16
33	250	16	2,963	15	3,171	15	2,818	17
33	250	15	2,647	15	0,641	15	2,462	16

Furthermore, we compare different list lengths of the LAHC fitness array. No variation of the list length is presented for the sLAHC algorithm as it could be seen throughout the experiments that a variation of the list length has no significant impact due to the fact that the late acceptance strategy is only used for calculating the TSP in the sLAHC as described in Section 3.3. It can be seen that the calculation with a list length $L_{fa} = 10000$ compares favorably to the case with a list length of $L_{fa} = 5000$. The LAHC achieves better objective function values for most instances with a longer list length.

In Table 3 we compare our results to results from literature, i.e., with the (truncated) branch-and-cut approach of Laporte et al. [12]. As they built an average over five random instances of Class 1 we also calculated the average results for our algorithms to allow a comparison. Table 3 shows that the sLAHC algorithm performs significantly better on average than the average results of

[12]. The LAHC implementation performs also better than the approach of [12] for the instances with more than 50 products.

Table 2. Numerical results Class 1 for unknown instances (300-500 products)

	sLAHC		$LAHC_{5000}$		$LAHC_{10000}$	
V K	$f(\sigma)$	V^*_{LA}	$f(\sigma)$	V^*_{LA}	$f(\sigma)$	V^*_{LA}
33 300	14576	17	14637	15	14346	16
33 300	14712	16	14733	14	14710	14
33 300	14463	16	14695	16	14354	15
33 300	14347	16	14189	19	14410	15
33 300	14530	15	14471	17	14329	16
33 350	15901	17	15990	19	16006	18
33 350	15047	13	14665	17	15035	16
33 350	16383	15	16464	16	16108	17
33 350	16252	17	16400	16	16023	17
33 350	16118	16	16398	17	16249	18
33 400	17790	18	17805	17	17905	19
33 400	16906	18	16836	17	16832	16
33 400	17335	18	17547	19	17480	18
33 400	17313	19	17824	18	17703	19
33 400	17825	17	18029	17	17832	19
33 450	18140	18	18360	18	18309	19
33 450	18164	18	18312	20	18097	18
33 450	18555	17	18677	19	18737	18
33 450	17819	17	17952	20	17875	18
33 450	19021	18	19081	18	18795	18
33 500	19897	19	19821	21	19827	19
33 500	19888	18	19604	19	20028	19
33 500	19774	20	19967	19	19885	19
33 500	19863	18	19925	19	19883	19
33 500	19497	20	19818	19	19643	20

Table 3. Comparison of results

	sLAHC		$LAHC_{5000}$		$LAHC_{10000}$		Laporte et al. [12]
V K	$\%gap$	V^*_{LA}	$\%gap$	V^*_{LA}	$\%gap$	V^*_{LA}	$\%gap$
33 50	0,000	8,4	7,507	8	6,135	8,4	6,312
33 100	0,816	10,4	3,215	10,2	3,839	9,8	4,150
33 150	1,434	11,8	3,331	12,4	2,411	12	4,760
33 200	1,954	12,6	4,776	13,6	3,540	13,2	8,567
33 250	2,666	15	3,044	15,2	3,155	16	5,478

Figure 3 shows an iterations-to-target chart for the LAHC and the sLAHC. It can be seen that the LAHC has a higher average starting value than the sLAHC. For the sLAHC a rapid drop in the average objective function value can be observed in the first hundred iterations and afterwards the sLAHC curve is flatter than the LAHC. For both algorithms it is apparent, that the most improvement in the average objective function value occurs in the first thousand iterations. After 10000 iterations almost no more significant improvement takes place. The figure also illustrates that the sLAHC has a lower average objective function value than the LAHC.

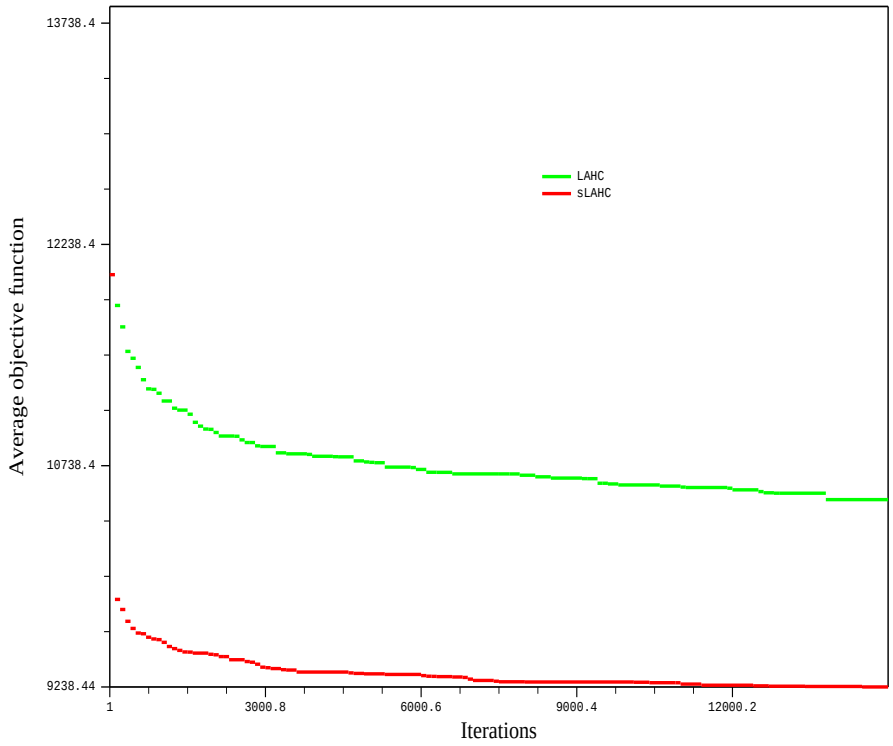

Fig. 3. Iterations-to-Target evaluation (LAHC/sLAHC)

5 Conclusion

We have applied Late Acceptance Hill Climbing to the Traveling Purchaser Problem. We combined several ideas from literature for building initial solutions and

for defining neighborhoods. We tested our approach on academic benchmarks and the results indicate that the LAHC procedure is suitable for solving the TPP. Furthermore, we also presented a simplified application of LAHC which led to promising results, especially for instances with a smaller number of products. The relatively weak performance of LAHC for small instances seems to indicate a small drawback of LAHC in comparison to other metaheuristics. While LAHC requires less effort for parameter setting, it does not allow a fine tuning of acceptance probabilities like, e.g., simulated annealing allows. On the other hand, LAHC might work well with an auto-adaptive parameter tuning that adjusts the neighborhood construction methods and the LAHC fitness array length. This could be a promising future research approach.

References

1. A. Abuhamdah. Experimental result of late acceptance randomized descent algorithm for solving course timetabling problems. *International Journal of Computer Science and Network Security*, 10(1):192–200, 2010.
2. E. Angelelli, R. Mansini, and M. Vindigni. Look-ahead heuristics for the dynamic traveling purchaser problem. *Computers & Operations Research*, 38:1867–1876, 2011.
3. M. Bellmore and G.L. Nemhauser. The traveling salesman problem: A survey. *Operations Research*, 16:538–558, 1968.
4. F. Boctor, G. Laporte, and J. Renaud. Heuristics for the traveling purchaser problem. *Computers & Operations Research*, 30:491–504, 2003.
5. E.K. Burke and Y. Bykov. A late acceptance strategy in hill-climbing for exam timetabling problems. In *PATAT '08 Proceedings of the 7th Int. Conf. on the Practice and Theory of Automated Timetabling*, 2008.
6. H. Cambazard and B. Penz. A constraint programming approach for the traveling purchaser problem. *Lecture Notes in Computer Science*, 7514:735–749, 2012.
7. M.C. Goldbarg, L.B. Bagi, and E.F.G. Goldbarg. Transgenetic algorithm for the traveling purchaser problem. *European Journal of Operational Research*, 199:36–45, 2009.
8. B. Golden, L. Levy, and R. Dahl. Two generalizations of the traveling salesman problem. *Omega*, 9:439–441, 1981.
9. L. Gouveia, A. Paias, and S. Voß. Models for a traveling purchaser problem with additional side-constraints. *Computers & Operations Research*, 38:550–558, 2011.
10. D. Infante, G. Paletta, and F. Vocaturo. A ship-truck intermodal transportation problem. *Maritime Economics & Logistics*, 11:247–259, 2009.
11. R.L. Karg and G.L. Thompson. A heuristic approach to solving travelling salesman problems. *Management Science*, 10:225–248, 1964.
12. G. Laporte, J. Riera-Ledesma, and J. Salazar-González. A branch-and-cut algorithm for the undirected traveling purchaser problem. *Operations Research*, 51:940–951, 2003.
13. H.L. Ong. Approximate algorithms for the travelling purchaser problem. *Operations Research Letters*, 1:201–205, 1982.
14. E. Özcan, Y. Bykov, M. Birben, and E.K. Burke. Examination timetabling using late acceptance hyper-heuristics. In *IEEE Congress on Evolutionary Computation, 2009. CEC '09*, pages 997–1004, 2009.

15. T. Ramesh. Traveling purchaser problem. *Opsearch*, 18:78–91, 1981.
16. J. Riera-Ledesma and J. Salazar-González. A heuristic approach for the travelling purchaser problem. *European Journal of Operational Research*, 162:142–152, 2005.
17. J. Riera-Ledesma and J. Salazar-González. Solving the asymmetric traveling purchaser problem. *Annals of Operations Research*, 144:83–97, 2006.
18. K.N. Singh and D.L. van Oudheusden. A branch and bound algorithm for the traveling purchaser problem. *European Journal of Operational Research*, 97:571–579, 1997.
19. A. Teeninga and A. Volgenant. Improved heuristics for the traveling purchaser problem. *Computers & Operations Research*, 31:139–150, 2004.
20. J. Verstichel and G. Berghe. A late acceptance algorithm for the lock scheduling problem. In S. Voß, J. Pahl, and S. Schwarze, editors, *Logistik Management*, pages 457–478. Physica, Heidelberg, 2009.
21. S. Voß. Dynamic tabu search strategies for the traveling purchaser problem. *Annals of Operations Research*, 63:253–275, 1996.