Diversified Late Acceptance Search

Majid Namazi [1,2], Conrad Sanderson [2,3], M.A. Hakim Newton [1], M.M.A. Polash [1], Abdul Sattar [1,2]

[1] Griffith University, Australia
[2] Data61, CSIRO, Australia
[3] University of Queensland, Australia

Abstract. 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 worsening moves, by keeping a circular array of fitness values of previously visited solutions and comparing the fitness values of candidate solutions against the least recent element in the array. While the straightforward strategy followed by LAHC has proven effective, there are nevertheless situations where LAHC can unfortunately behave in a similar manner to HC, even when using a large fitness array. For example, when the same fitness value is stored many times in the array, particularly when a new local optimum is found. To address this shortcoming, we propose to improve both the diversity of the accepted solutions and the diversity of values in the array through new acceptance and replacement strategies. The proposed Diversified Late Acceptance Search approach is shown to outperform the current state-of-the-art LAHC method on benchmark sets of Travelling Salesman Problem and Quadratic Assignment Problem instances.

Keywords: Local Search, Late Acceptance, Diversification.

1 Introduction

Local search algorithms are typically efficient and scalable approaches to solve large instances of real world optimisation problems [12]. Such algorithms use the following overall approach: starting from an *initial solution*, iteratively move from one solution to another, with the aim to eventually arrive at a good solution. The initial solution is often generated randomly or by using a specialised method. Then, in each iteration, a *candidate solution* is obtained by modifying the *current solution* using a *perturbation method*. If the candidate solution in a given iteration satisfies a given *acceptance criterion*, it is used as the starting point for the next iteration. Otherwise, the current solution in the given iteration becomes the starting point for the next iteration. The traditional Hill Climbing (HC) approach is a local search method that strictly uses a greedy strategy as its acceptance criterion [3]. HC accepts the candidate solution only if its fitness value is better (smaller in minimisation problems and larger in maximisation problems) than that of the current solution. This greedy strategy typically leads the search to quickly being trapped in a local optimum.

An important challenge in designing a local search algorithm is to find a good balance between interleaving *diversification* and *intensification* phases during search [12]. *Diversification* means exploring the solution space as widely as possible, with the intent of ideally finding a globally optimum solution. In contrast, *intensification* means

improving the current solution in order to converge to the best local solution as quickly as possible. The perturbation method as well as the acceptance criterion need to take this balancing issue into account. As HC does not explore solutions that are worse than the current solution in each iteration, HC uses a very high level of intensification at the cost of very low level of diversification. Overall, the HC algorithm converges quickly to a local optimum, but the quality of its solutions is often not high [6,7]. Diversification strategies are hence necessary to provide better solutions.

There are well-studied acceptance criteria that, with the aim to avoid or escape local optima, also accept worsening moves, rather than simply accepting only better candidate solutions. Simulated Annealing (SA) [13] uses a stochastic acceptance criterion, where worsening moves are accepted with a probability that depends on the difference in the fitness values of the current solution and the candidate solution, with the probability exponentially diminishing over time. Threshold Acceptance (TA) [10] is a deterministic acceptance criterion, which accepts worsening moves if the difference in the fitness values of the current and the candidate solution is below a given threshold. The Great Deluge Algorithm (GDA) [9,15,16] accepts worsening moves if the fitness value of the candidate solution is below a given level. Each of the above acceptance criteria has a parameter whose *initial value* and a *variation schedule* must be defined beforehand. Unfortunately, obtaining a suitable initial value and variation schedule is difficult to achieve, and is often problem domain dependent and/or problem instance dependent [5,7,15]. This can make practical use of SA, TA and GDA quite finicky.

In contrast to the above approaches, Late Acceptance Hill Climbing (LAHC) search [6,7] is a relatively straightforward technique which deterministically accepts worsening moves and has no complicated parameters. An array with a predefined length stores the fitness values of previously visited solutions. Fitness values of candidate solutions are compared against the least recent element in the array. Since the fitness values from previous iterations can be worse than that of the current solution, a candidate solution that is worse than the current solution can be accepted. As the search progresses, the array is deterministically updated with fitness values of new solutions. The use of the fitness array thus brings about search diversity. The larger the length of the array, the better the diversity level. Overall, LAHC exhibits better diversification in terms of the explored solutions and provides solutions which typically have higher quality than HC [6,7]. Moreover, LAHC has been successful in several optimisation competitions [2,18], and has been used in real world applications [17].

Despite the promising aspects of LAHC, in this work we observe that there are situations where LAHC can unfortunately behave in a similar manner to HC, even when using a large fitness array. For example, when the same fitness value is stored many times in the array, particularly when a new local optimum is found. In this case, the fitness values in the array are iteratively replaced with the new local optimum fitness value, thereby reducing diversity.

To address the above shortcoming, we propose a new search approach termed Diversified Late Acceptance Search (DLAS). The approach uses: **(i)** a new acceptance strategy which increases diversity of the accepted solutions, and **(ii)** a new replacement strategy to improve the diversity of the values in the fitness array by taking worsening, improving, and sideways movement scenarios into account. These strategies improve

the overall diversity of the search. The proposed DLAS approach is shown to outperform the current state-of-the-art LAHC method on benchmark sets of Travelling Salesman Problem (TSP) instances and Quadratic Assignment Problem (QAP) instances.

We continue the paper as follows. Section 2 overviews the LAHC algorithm, followed by discussing its problems in Section 3. Section 4 presents the proposed DLAS algorithm. Section 5 provides comparative evaluations. The main findings are summarised in Section 6.

2 Late Acceptance Hill Climbing

Local search algorithms start from an *initial solution* S_0. The current solution S_k in each iteration k is then modified by a given perturbation method M to generate a new candidate solution $S'_k = M(S_k)$. Next, using a given *acceptance criterion* $\mathcal{A}$, the candidate solution S'_k is either accepted or rejected, meaning either $S_{k+1} = S'_k$ if $\mathcal{A}(k) = \mathsf{true}$, or $S_{k+1} = S_k$ if $\mathcal{A}(k) = \mathsf{false}$. Assume F_k and F'_k denote the fitness values of solutions S_k and S'_k, respectively. For convenience, we assume *minimisation problems*, where one solution is *better* than the other if fitness value of the former is less than that of the latter. In HC, $\mathcal{A}(k) = \mathsf{true}$ iff $F'_k \leq F_k$, and so $F_k \geq F_{k+1}$ for all $k \geq 0$. This means that HC accepts only non-worsening moves, ie., sideways moves or improving moves.

The most recent version of LAHC [7], shown in Fig. 1, accepts candidate solution S'_k if its fitness value F'_k is better than or equal to the fitness value F_k of the current solution S_k, as in HC. Also, for a given *history length* L, candidate solution S'_k is accepted if its fitness value F'_k is better than the fitness value F_{k-L} of the then current solution S_{k-L} at iteration $k - L \geq 0$. In other words, $\mathcal{A}(k) = F'_k \leq F_k$ or $F'_k < F_{k-L}$ for $k \geq L$. Since F_{k-L} is usually (not always as in HC) greater than F_k, the candidate solution S'_k can be accepted at iteration $k \geq L$, even if $F'_k > F_k$. LAHC thus accepts worsening moves like TA and GDA and thereby aims to avoid or escape from local minima.

```
1  proc LAHC
2    Initialise curr solution S, compute F
3    Specify length L for fitness array Φ
4    forall k ∈ [0, L), Φ[k] ← F
5    l ← 0, S* ← S, F* ← F        // best S*
6    while termination-criteria   // iter k
7      S' ← M(S), compute F' // perturb
8      if F' ≤ F or F' < Φ[l]     // accept
9        S ← S', F ← F'
10     if F < F*     // new best
11       S* ← S, F* ← F
12     if F < Φ[l]     // replace in Φ
13       Φ[l] ← F
14     l ← l + 1 mod L     // index of Φ
15   return S*, F*
```

Fig. 1. Late Acceptance Hill Climbing (LAHC) algorithm, adapted from [7].

To achieve memory efficiency, LAHC is typically implemented using a circular *fitness array* Φ of size L to store fitness values of previous L solutions. Initially, all the the values in Φ are set by F_0. Then, in each iteration k, candidate solution S'_k is accepted if $F'_k \leq F_k$ or $F'_k < \Phi[k \bmod L]$. After the acceptance or rejection in each iteration k, ie., $F_{k+1} = F'_k$ or $F_{k+1} = F_k$, the value in $\Phi[k \bmod L]$ is set to F_{k+1} just whenever F_{k+1} is better than $\Phi[k \bmod L]$. Overall, LAHC exhibits better diversification level with a larger L [4,7], as this allows comparison with further earlier solutions which are most likely further worse as well.

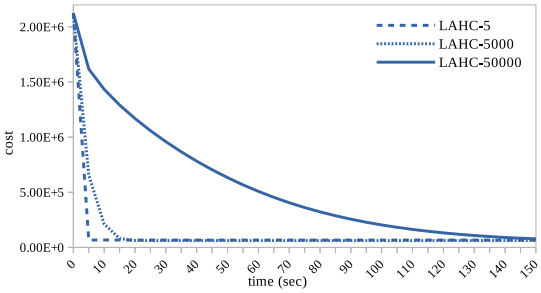

Fig. 2. Search progress for the first 150 seconds while solving the benchmark U1817 TSP instance via LAHC with $L \in \{5, 5000, 50000\}$. Further progress is shown in Fig. 3 below. To aid clarity, results for DLAS are not shown as they effectively cover LAHC with L=5 at the given scale.

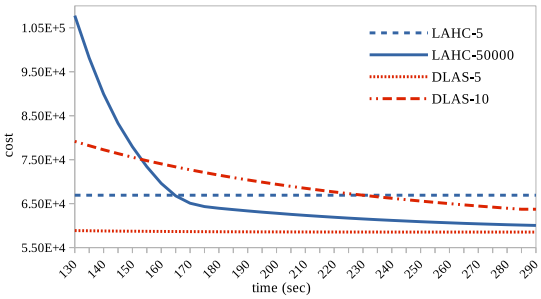

Fig. 3. Search progress of LAHC and DLAS with various L values in later iterations of solving the U1817 instance. Note the vertical axis scale change compared to Fig. 2. LAHC with L=5 converges quicker than LAHC with L=50000, but obtains a worse solution. DLAS with L=5 obtains a better solution than LAHC. Furthermore, DLAS with L=5 converges much quicker than LAHC with L=50000.

3 Problems with LAHC

We have empirically observed that for some problems LAHC unfortunately behaves in a similar manner to HC and does not accept worsening moves. Figs. 2 and 3 show typical search progress trend while solving the benchmark U1817 TSP instance (see Sec 5.2 for details on TSP instances). A similar pattern is seen in other benchmark instances. For a small value of L, LAHC is quickly trapped in a local optimum, leading to a poor quality solution. In that case, even using restart techniques does not help obtain a high quality solution [4,7]. While using larger values of L leads to the search being less prone to trapping, it comes at the cost of slow convergence speed. As such, the solution quality in this case can be very poor if not enough time is allotted. This characteristic of LAHC makes it less useful for applications in time-constrained systems where a high-quality solution must be found quickly.

The poor performance of LAHC is due to the following reasoning. Consider the LAHC algorithm in Fig. 1. Assume that in a given iteration, all the values in the fitness array Φ are equal to the fitness value F_* of a newly found best solution S_*, where S_* is a hard-to-improve or a local optimum solution. This happens when a new overall best solution S_* with fitness value F_* is found and F remains to be equal to F_* for at least L consecutive iterations. In this case, no worsening moves with larger fitness values than F_* will be accepted anymore, and if S_* is a local optimum then the search is trapped in that solution. Clearly, this is the situation HC reaches when it is trapped in a local optimum. We show in the experiments section that even when using a large value for L, LAHC behaves like HC in solving many problems in a large proportion of the iterations.

4 Proposed Diversified Late Acceptance Search

We propose a new search approach that aims to obtain high diversity level and high convergence speed, all while not suffering from the abovementioned drawbacks of LAHC. We overview the approach as follows. We aim to keep or obtain larger fitness values in the fitness array when the search encounters non-improving moves (diversification). Furthermore, we cautiously replace large fitness values with small values when the search accepts improving moves (intensification). Lastly, our acceptance criterion is more relaxed than LAHC (diversification).

4.1 Acceptance Strategy

Comparing the fitness values of the candidate solutions with a larger value than $\Phi[k \bmod L]$, if there is any in the fitness array, arguably increases diversity of accepted solutions. So, our acceptance strategy is to compare the fitness value F'_k of the candidate solution S'_k with the maximum fitness value in the fitness array Φ, instead of comparing it just with $\Phi[k \bmod L]$. The new candidate solution S'_k would be accepted if $F'_k = F_k$ or $F'_k < \Phi_{\max}$, ie., the maximum value in the fitness array Φ. The first condition makes possible accepting new candidate solutions with fitness values equal to $\Phi_{\max}$ when all the values in Φ are the same, especially in the initial and final iterations of the search. Accepting candidate solutions with smaller fitness values than $\Phi_{\max}$ in other iterations increases the level of acceptable worsening moves and thereby increases the diversity level of the search. We show how to efficiently find and maintain the maximum value in the fitness array Φ in Section 4.3.

```
1  proc DLAS
2    Initialise curr solution S, compute F
3    Specify length L for fitness array Φ
4    forall k ∈ [0, L), Φ[k] ← F
5    Φ_max ← F, N ← L
6    l ← 0, S_* ← S, F_* ← F        // best S_*
7    while termination-criteria     // iter k
8      F⁻ ← F                       // previous
9      S' ← M(S), compute F'        // perturb
10     if F' ≤ F or F' < Φ_max      // accept
11       S ← S', F ← F'
12     if F < F_*                   // new best
13       S_* ← S, F_* ← F
14     if F > Φ[l]                  // replace in Φ
15       Φ[l] ← F
16     else if F < Φ[l] and F < F⁻
17       if Φ[l] = Φ_max            // decrement
18         N ← N − 1
19       Φ[l] ← F                   // replace
20       if N = 0                   // recompute
21         compute Φ_max,N
22     l ← l + 1 mod L              // index of Φ
23   return S_*, F_*
```

Fig. 4. Proposed Diversified Late Acceptance Search (DLAS).

4.2 Replacement Strategy

Our replacement strategy has two parts. In the first one, if the fitness value F_{k+1} of the new current solution S_{k+1} is larger than $\Phi[k \bmod L]$, the value in $\Phi[k \bmod L]$ is always replaced by F_{k+1}. Such a replacement is avoided in the latest version of the LAHC algorithm to increase intensification of the search. However, this replacement increases the probability of accepting more worsening moves in the future iterations and thereby results in much better solutions in our method. The second part of our replacement strategy is that if F_{k+1} is smaller than $\Phi[k \bmod L]$, the replacement must be done just when F_{k+1} is smaller than the fitness value F_k as well. Such a replacement strategy

avoids replacing other large values in the fitness array in a series of consecutive steps if our method falls in a plateau or in a local optimum.

We note that the combination of these two replacement methods is new and is different from replacing just in acceptance or just in improving moves. An illustration of the proposed method, especially the replacement strategy, is given in Section 4.4.

4.3 Diversified Late Acceptance Search

Fig. 4 shows the pseudo code for the proposed method using the above acceptance and replacement strategies. We have termed the proposed method as Diversified Late Acceptance Search (DLAS).

Variables $\Phi_{\max}$ and N in Fig. 4 are respectively always equal to the maximum value in the fitness array and the number of occurrences of that value in the array. In Line 5, $\Phi_{\max}$ and N are initialised by F and L. In every iteration in Line 8, F^- holds the previous value of F. In Line 10, new candidate solution S' is accepted if $F' = F$ or $F' < \Phi_{\max}$. In Line 14, if $F > \Phi[l]$, replacement is made. Otherwise, in Line 16, if $F < \Phi[l]$ and $F < F^-$, replacement is made. However, before making the replacement this time, if $\Phi[l]$ is equal to $\Phi_{\max}$, N is decremented by one. In Line 20, if N is equal to zero, the values of $\Phi_{\max}$ and N are recomputed by checking all the values in the fitness array. Overall, Lines 10 & 14–20 show the most important parts of the proposed DLAS.

4.4 DLAS Replacement Scenarios

Fig. 5 shows eight possible combinations of values of F, F^- and $\Phi[l]$ compared to each other and corresponding replacement rules.

Worsening Moves. In cases (1)–(3) in Fig. 5, worsening moves take place. In case (1), the fitness value of the new current solution F is still smaller than $\Phi[l]$. In this case,

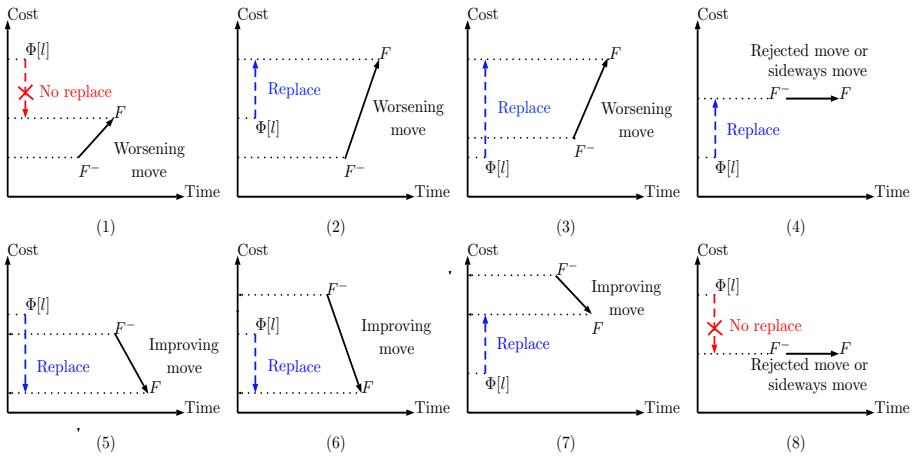

Fig. 5. All possible combinations of values of F, F^- and $\Phi[l]$ compared to each other and corresponding replacement rules in the proposed DLAS approach. See the text for detailed explanation.

contrary to LAHC, replacement is not allowed in the proposed DLAS method. This avoidance of replacement actually preserves the large values in the fitness array Φ when DLAS does not improve the current solutions in some consecutive iterations, and at the same time the fitness values of the new worse solutions are not larger than the corresponding values in the fitness array Φ. In cases (2) and (3), the fitness value of the new current solution F is greater than $\Phi[l]$. In both these cases, contrary to LAHC, replacement is allowed in DLAS to increase diversity of values in the fitness array Φ.

Improving Moves. In cases (5)–(7), improving moves take place. In cases (5) and (6), the fitness value of the new current solution F is smaller than $\Phi[l]$. In both these cases, as in LAHC, replacement is allowed to optimistically increase the intensification of the search. In case (7), the fitness value of the new current solution F is still greater than $\Phi[l]$. Contrary to LAHC, replacement is allowed in DLAS to increase diversity of values in the fitness array.

Sideways Moves or Rejected Moves. In cases (4) and (8), there are two possible outcomes: a candidate solution is not accepted, or a sideways move occurs. In case (4), the fitness values of the previous and the new current solutions, ie., F^- and F, are greater than $\Phi[l]$. In this case, contrary to LAHC, replacement is allowed in DLAS to increase diversity of the accepted solutions in future iterations. In case (8), the fitness values of the previous and the new current solutions are smaller than $\Phi[l]$. In this case, contrary to LAHC, replacement is not allowed in DLAS. This avoidance of replacement actually avoids replacing all the values in the fitness array Φ in consecutive iterations when DLAS falls in a plateau or local optimum.

5 Comparative Evaluation

In this section the performance of proposed DLAS method is compared against the most recent version of LAHC (as described in Sec. 2) as well as the recently proposed Step Counting Hill Climbing (SCHC) [8] approach. Briefly, in SCHC a fitness bound and a counter limit are used instead of a fitness array. The fitness bound is initialised by the fitness of the initial solution and the counter limit is similar to the length of the fitness array. In each iteration, a candidate solution is accepted if its fitness is equal to or better than that of the current solution or better than the fitness bound. Whenever the number of iterations becomes a factor of the counter limit, the fitness bound is made equal to the fitness of the current solution.

The proposed DLAS algorithm, as well as LAHC and SCHC, are general purpose local search algorithms for solving any optimisation problem. Hence, we use sets of Travelling Salesman Problems (TSPs) and Quadratic Assignment Problems (QAPs) just to compare the relative performance of the three algorithms, and not to improve the best known solutions for the individual problems. All experiments were ran on the same computing cluster with a 500 Mb memory limit. Each node of the cluster is equipped with Intel Xeon CPU E5-2670 processors running at 2.6 GHz.

5.1 Time Cutoff and Fitness Array Length

To provide a fair comparison, we use time cutoff as the stopping condition. However, as each instance has its own size and complexity level, we decided to solve all of them

first with LAHC using a reasonably large fitness array size L. We initially performed 50 runs of the LAHC algorithm (with L=50000) on each instance, with the stopping condition as getting trapped in a local optimum for at least 10% of the total running time. Then we took the longest running time across the 50 runs as the cutoff time for each instance. We then ran all three algorithms with just the cutoff time as the stopping condition 50 times for each unique value for L.

The reported results in the following subsections are the averages of 50 runs on each instance using the best performing value for L. For example, Fig. 3 compares LAHC and DLAS algorithms in the later steps of solving U1817 TSP instance using various values for L. The figure shows that given 290 seconds as the cutoff time for this instance, L=50000 and L=5 are the best values for LAHC and DLAS algorithms, respectively.

5.2 Experiments on TSP instances

Every TSP instance includes a set of cities or points on a map. The cities are all connected with each other by symmetric roads of given distances or lengths. The goal of solving such a TSP instance is to find the shortest closed tour that includes all the cities such that every city is visited exactly once. We took all the symmetric Euclidean distance TSP instances with 1,000 to 10,000 cities from the well-known TSPLIB benchmark dataset at `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`. We used the same source code and the same perturbation heuristic provided by the authors of [7] for solving the TSP instances. The perturbation heuristic randomly divides a given tour into two parts and then reverses one part [14].

Table 1 shows the results on TSP instances using LAHC and SCHC with L=50000 and DLAS with L=5. The size of each instance is taken as the number in the name of the instance, which indicates the number of cities. In 20 out of 23 instances, the proposed DLAS method with L=5 has found better solutions than both LAHC and SCHC with L=50000. In 17 of those instances the differences are statistically significant based on t-test with the confidence level of 0.95. The results also show that in small instances (with small number of cities), DLAS finds better solutions in less time, while in large instances it does not get trapped in a local optimum quickly and continues to search for a better solution. For example, for the largest instance in the last line of the table with the time cutoff of 2545 seconds, LAHC and SCHC are quickly trapped in a local optimum and cannot improve their last found solutions. In contrast, the proposed DLAS method continues to improve its solutions until almost the end of the cutoff time.

The results also show that even when using a very large value for L in LAHC and SCHC, in about half of the iterations (especially for large instances), LAHC and SCHC undesirably behave like HC. This includes iterations in which the maximum value in the fitness array in LAHC and the fitness bound in SCHC are equal to the last found best solution. In contrast, the percentage of iterations in which DLAS behaves like HC is zero. In other words, even when using very small fitness arrays, there is always room for worsening moves to be accepted by DLAS. This indicates that the combination of the new acceptance and replacement strategies in DLAS is more effective in increasing the diversity level of the search than just increasing the length of the fitness array.

Figs. 6 and 7 show that DLAS with L=5 has a high convergence speed (due to the small fitness array size) and converges almost as fast as HC. It also shows that DLAS

Table 1. Results on TSP instances for LAHC and SCHC with L=50000, and DLAS with L=5. In the first column, the size of each instance is taken as the number in the name of the instance, which indicates the number of cities. The 2nd column is the best known solution cost reported in the literature for each instance. The 3rd column is the time cutoff value used by all methods for each instance. The 4th column shows the deviations of the best solution cost from the best known solution cost for each instance. The 5th column shows the time spent by each algorithm to find the best solution. The 6th column shows percentage of iterations in which each algorithm undesirably behaves like HC. Shading denotes winning numbers where the differences are statistically significant.

Instance name	Best known sol. cost	Time cutoff	Dev. from the best known solution			Time to find the last best sol.			% of iterations behaving like HC		
			LAHC	SCHC	DLAS	LAHC	SCHC	DLAS	LAHC	SCHC	DLAS
Dsj1000	18659688	100	924536	705626	**339555**	80	66	52	21	36	**0**
Pr1002	259045	120	6265	6552	**4795**	78	63	51	37	47	**0**
U1060	224094	150	4560	5647	**4193**	84	68	55	45	54	**0**
Vm1084	239297	155	**5884**	6593	5927	79	65	51	51	60	**0**
Pcb1173	56892	160	1910	2118	**1306**	81	77	49	52	52	**0**
D1291	50801	165	2612	1856	**1404**	111	88	93	35	49	**0**
Nrw1379	56638	177	2024	2159	**1180**	117	93	90	37	51	**0**
Fl1400	20127	180	**290**	324	901	116	92	33	43	57	**0**
U1432	152970	200	3513	4139	**2022**	125	114	176	45	55	**0**
Fl1577	22249	250	**466**	524	634	153	139	108	50	57	**0**
D1655	62128	270	2424	2464	**1550**	153	120	160	43	59	**0**
Vm1748	336556	280	10328	11009	**8967**	163	125	173	45	59	**0**
U1817	57201	290	2320	2461	**1450**	189	146	244	41	59	**0**
D2103	80450	309	5846	6137	**2660**	194	161	279	39	47	**0**
U2152	64253	320	2598	2956	**1350**	211	198	292	46	51	**0**
U2319	234256	350	3625	3837	**2557**	258	228	347	45	56	**0**
Pr2392	378032	370	19557	16025	**9003**	238	167	274	40	58	**0**
Pcb3038	137694	521	6530	7118	**3116**	324	267	384	42	51	**0**
Fl3795	28772	1110	1542	1547	**1202**	802	769	666	65	72	**0**
Fnl4461	182566	1150	9607	10558	**3978**	454	419	940	62	69	**0**
Rl5915	565530	1200	36974	39929	**19232**	718	613	1198	48	59	**0**
Rl5934	556045	1320	35718	38535	**34863**	812	664	814	46	60	**0**
Pla7397	23260728	2545	962561	990251	**916947**	1926	1818	2542	59	70	**0**

with L=5 ends up with a better solution than LAHC and SCHC with L=50000, and HC for the Fnl4461 instance.

5.3 Experiments on QAP instances

Every QAP instance includes two same-size sets of locations and facilities. The locations are all connected with each other by symmetric links of given distances or lengths. There is a flow between every pair of facilities with a given weight. The goal of solving such a QAP instance is assigning each facility to a location such that the sum of weights of flows between every two facilities multiplied by the distances between their assigned locations is minimised.

We took all QAP instances with at least 80 locations and facilities from the well-known QAPLIB benchmark dataset at `http://anjos.mgi.polymtl.ca/qaplib/`. We used the same source code and the same perturbation heuristic provided in

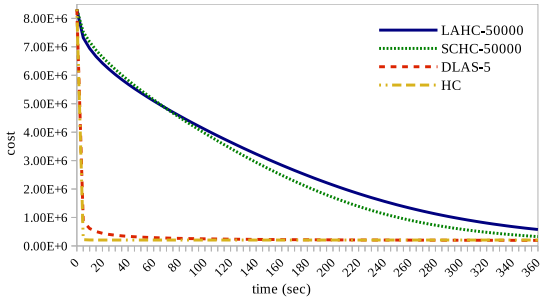

Fig. 6. Search progress for the first 360 seconds while solving the benchmark Fnl4461 TSP instance via HC, LAHC and SCHC with L=50000, and DLAS with L=5.

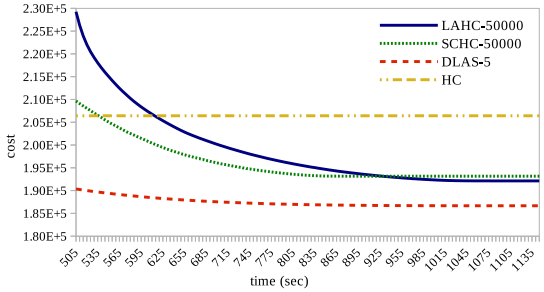

Fig. 7. As per Fig. 6, but in later iterations. Note the scale change of the vertical axis. The proposed DLAS approach obtains a better solution than HC, LAHC and SCHC. Furthermore, DLAS converges quicker than LAHC.

`http://mistic.heig-vd.ch/taillard/` for solving the QAP instances. The perturbation heuristic randomly selects two locations and swaps their assigned facilities.

Table 2 shows the results on QAP instances using LAHC and SCHC with L=50000 and DLAS with L=10, respectively. In 15 out of 24 instances, the proposed DLAS method with L=10 found better solutions than both LAHC and SCHC with L=50000. In 10 of those instances the differences are statistically significant based on t-test with the confidence level of 0.95. Notably, the results also show that in most of the instances, especially small ones, DLAS finds better solutions in considerably less time. The last column shows that even using a very large value for L, in about 10% of the iterations LAHC behaves like HC. For SCHC, the figure is 20%. In contrast, the percentage of iterations in which DLAS behaves like HC is zero.

6 Conclusions

The well-known Late Acceptance Hill Climbing (LAHC) search algorithm strives to escape or avoid local optima by deterministically accepting worsening moves. LAHC stores fitness values of a predefined number of previous solutions in a fitness array and compares fitness values of candidate solutions against the least recent element in the array, rather than simply against the fitness value of the current solution. The fitness values stored in the array are deterministically replaced as the search progresses. Unfortunately, the behaviour of LAHC can become similar to that of traditional Hill Climbing search (ie., getting trapped in a local minimum) when the same fitness value is stored many times in the fitness array, particularly when a new local optimum is found.

Table 2. Results on QAP instances for LAHC and SCHC with L=50000, and DLAS with L=10. The size of each instance is taken as the number in the name of the instance, which indicates the number of locations or facilities. Explanations for the other columns are as per Table 1.

Instance name	Best known sol. cost	Time cutoff	Dev. from the best known solution			Time to find the last best sol.			% of iterations behaving like HC		
			LAHC	SCHC	DLAS	LAHC	SCHC	DLAS	LAHC	SCHC	DLAS
Lipa80a	253195	20	1607	1564	**1411**	14	11	8	1.3	0.3	**0.0**
Tai80a	13499184	21	330957	354263	**264177**	15	12	15	0.5	0.0	**0.0**
Lipa80b	7763962	26	39769	190699	**0**	22	17	8	8.0	28.5	**0.0**
Tai80b	818415043	27	4227835	**3574665**	979737	20	17	6	8.1	16.8	**0.0**
Sko81	90998	24	222	178	**113**	19	16	5	4.7	14.8	**0.0**
Lipa90a	360630	23	2045	2024	**1893**	19	15	13	0.0	1.0	**0.0**
Lipa90b	12490441	36	51015	20709	**0**	29	22	11	15.0	33.2	**0.0**
Dre90	1838	35	1575	1615	**1450**	16	12	8	0.0	6.3	**0.0**
Sko90	115534	28	321	310	**219**	26	21	8	1.2	10.0	**0.0**
Sko100a	152002	40	**190**	239	218	32	25	11	4.6	16.8	**0.0**
Tai100a	21052466	35	460894	486157	**378092**	23	18	29	0.0	0.9	**0.0**
Sko100b	153890	52	175	173	**160**	30	24	10	9.3	16.0	**0.0**
Tai100b	1185996137	55	**2711882**	2823207	5124004	34	29	13	12.6	38.3	**0.0**
Sko100c	147862	42	147	132	**121**	32	26	11	6.6	15.6	**0.0**
Sko100d	149576	42	**241**	246	245	30	24	10	10.7	23.8	**0.0**
Sko100e	149150	42	**150**	165	156	31	25	10	5.8	19.7	**0.0**
Sko100f	149036	42	237	232	**204**	33	26	11	7.7	16.9	**0.0**
Wil100	273038	35	**149**	171	241	32	26	10	2.5	12.8	**0.0**
Dre110	2264	37	2031	2057	**1782**	25	19	18	1.7	4.9	**0.0**
Esc128	64	21	0	0	0	6	5	0.3	70.0	77.0	**0.0**
Dre132	2744	65	2522	2543	**2140**	39	30	39	4.7	10.8	**0.0**
Tai150b	498896643	105	**1511339**	1669639	2641722	73	61	56	9.2	22.8	**0.0**
Tho150	8133398	130	9615	9282	**6894**	80	65	79	14.1	23.8	**0.0**
Tai256c	44759294	60	**128527**	132333	134885	35	27	54	16.9	30.9	**0.0**

To address the above issue, we have proposed: **(i)** a new acceptance strategy which increases diversity of the accepted solutions, and **(ii)** a new replacement strategy to improve the diversity of the values in the fitness array by taking worsening, improving, and sideways movement scenarios into account. These strategies improve the overall diversity of the search.

The proposed Diverse Late Acceptance Search (DLAS) method is shown to outperform the current state-of-the-art LAHC method on benchmark sets of Travelling Salesman Problem (TSP) and Quadratic Assignment Problem (QAP) instances. Furthermore, the combination of the new acceptance and replacement strategies in DLAS is more effective in increasing the diversity of the search than just increasing the length of the fitness array, and can lead to better quality solutions that are obtained with a lower number of search iterations (ie., less time).

Future avenues of exploration include comparative evaluation of DLAS against other LAHC variants (such as Average Late Acceptance Randomised Descent search [1]), as well as evaluation on other optimisation problems, such as high-school timetabling [5,11].

References

1. Abuhamdah, A.: 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. Afsar, H.M., Artigues, C., Bourreau, E., Kedad-Sidhoum, S.: Machine reassignment problem: the ROADEF/EURO challenge 2012. Annals of Operations Research 242(1), 1–17 (2016)
3. Appleby, J., Blake, D., Newman, E.: Techniques for producing school timetables on a computer and their application to other scheduling problems. The Computer Journal 3(4), 237–245 (1961)
4. Bazargani, M., Lobo, F.G.: Parameter-less late acceptance hill-climbing. In: Genetic and Evolutionary Computation Conference. pp. 219–226 (2017)
5. Burke, E., Bykov, Y., Newall, J., Petrovic, S.: A time-predefined local search approach to exam timetabling problems. IIE Transactions 36(6), 509–528 (2004)
6. Burke, E.K., Bykov, Y.: A late acceptance strategy in hill-climbing for examination timetabling problems. In: Conference on the Practice and Theory of Automated Timetabling (2008)
7. Burke, E.K., Bykov, Y.: The late acceptance hill-climbing heuristic. European Journal of Operational Research 258(1), 70–78 (2017)
8. Bykov, Y., Petrovic, S.: A step counting hill climbing algorithm applied to university examination timetabling. Journal of Scheduling 19(4), 479–492 (2016)
9. Dueck, G.: New optimization heuristics: The great deluge algorithm and the record-to-record travel. Journal of Computational Physics 104(1), 86–92 (1993)
10. Dueck, G., Scheuer, T.: Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. Journal of Computational Physics 90(1), 161–175 (1990)
11. Fonseca, G.H., Santos, H.G., Carrano, E.G.: Late acceptance hill-climbing for high school timetabling. Journal of Scheduling 19(4), 453–465 (2016)
12. Hoos, H.H., Stützle, T.: Stochastic local search: Foundations and applications. Elsevier (2004)
13. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983)
14. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling-salesman problem. Operations Research 21(2), 498–516 (1973)
15. McMullan, P.: An extended implementation of the great deluge algorithm for course timetabling. Computational Science – ICCS 2007. Lecture Notes in Computer Science, Vol. 4487. pp. 538–545 (2007)
16. Obit, J., Landa-Silva, D., Ouelhadj, D., Sevaux, M.: Non-linear great deluge with learning mechanism for solving the course timetabling problem. In: Metaheuristics International Conference (2009)
17. Smet, G.D., et al.: OptaPlanner User Guide. Red Hat and the community, http://www.optaplanner.org
18. Wauters, T., Toffolo, T., Christiaens, J., Van Malderen, S.: The winning approach for the Verolog Solver Challenge 2014: the swap-body vehicle routing problem. In: Belgian Conference on Operations Research (ORBEL) (2015)