Late Acceptance Hill Climbing for The Liner Shipping Fleet Repositioning Problem

Kevin Tierney*

Abstract
Late Acceptance Hill Climbing (LAHC) has been shown to be an effective local search method for several types of optimization problems, such as on certain types of scheduling problems as well as the traveling salesman problem. We apply LAHC to a central problem in the liner shipping industry, the Liner Shipping Fleet Repositioning Problem (LSFRP). The LSFRP involves the movement of vessels between routes in a liner shipping network subject to complex costs and timing restrictions. We show that despite LAHC's promising performance on other problems, it is unable to achieve the performance of simulated annealing on the LSFRP.

Keywords: liner shipping, fleet repositioning, maritime transportation, late acceptance hill climbing

1 Introduction

Late Acceptance Hill Climbing (LAHC) has been the subject of a number of recent papers in which it has shown strong performance versus other heuristic approaches [3, 10, 1, 5, 2]. LAHC is characterized by its simplicity, in that it involves a standard hill climbing algorithm with a modified solution acceptance criterion. Instead of only accepting solutions that have an objective value greater than the current solution, LAHC also accepts solutions with an objective value greater than the objective value of a solution in a particular previous iteration.

Given LAHC's good performance on both scheduling and routing problems, it seems a natural choice for use in solving the Liner Shipping Fleet Repositioning Problem (LSFRP) [8, 9, 7]. The LSFRP consists of moving vessels between cyclical routes called *services* in a liner shipping network. During repositioning, vessels may undertake a number of activities to help lower costs or even earn profits for a liner shipper. Thus, the LSFRP involves routing vessels from their initial service to a shared goal service while preventing multiple vessels from performing the same activity. In addition, a number of activities in the LSFRP, such as sailing or visiting certain ports, can vary in length and must therefore be scheduled. These activities pose an additional difficulty in that the costs vary with the activity duration. Finally, vessels may carry cargo to earn revenue for the liner shipper while repositioning. This means that in addition to a routing and scheduling problem, there is a multi-commodity flow as well. Finding high-quality solutions to the LSFRP is important, since even small percentage improvements to the objective function of the problem can mean savings hundreds of thousands of dollars.

In this extended abstract, we compare an LAHC approach for the LSFRP to the current state-of-the-art methods, a MIP model solved with CPLEX 12.4 from [9] and [7], as well as a simulated annealing (SA) approach from [7]. We show that despite LAHC's favorable performance against SA on other problems, we are unable to achieve performance gains over SA on the LSFRP on either the confidential or public datasets from [7].

2 LSFRP

We briefly describe the LSFRP, and refer readers to any one of [8, 9, 7] for a more detailed description of the problem.

Liner shipping networks consist of a set of cyclical routes, called services, that visit ports on a regular, usually weekly, schedule. Liner shipping networks are designed to serve customer's cargo demands, but over time the economy changes and liner shippers must adjust their networks in order to stay competitive. Liner shippers add, remove and modify existing services in their network in order to make changes to the network. Whenever a new service is created, or an existing service is expanded, vessels must be *repositioned* from their current service to the service being added or expanded. Vessel repositioning is expensive due to the cost of fuel (in the region of hundreds of thousands of dollars) and the revenue lost due to cargo flow disruptions. Given that liner shippers around the world reposition hundreds of vessels per year, optimizing vessel movements can significantly reduce the economic and environmental burdens of containerized shipping, and allow shippers to better utilize repositioning vessels to transport cargo.

The aim of the LSFRP is to maximize the profit earned when repositioning a number of vessels from their initial services to a service being added or expanded, called the goal service.

Liner shipping services are composed of multiple *slots*, each of which represents a cycle that is assigned to a particular vessel. Each slot is composed of a number of *visitations*, which can be thought of as port calls, i.e. a specific time when a vessel is scheduled to arrive at a port. A vessel that is assigned to a particular slot

*Kevin Tierney is with the IT University of Copenhagen, Copenhagen S, Denmark and the Institute for Information Systems, University of Hamburg, Hamburg, Germany (email: `kevt@itu.dk`) sequentially sails to each visitation in the slot.

Vessel sailing speeds can be adjusted throughout repositioning to balance cost savings with punctuality. The bunker fuel consumption of vessels increases cubically with the speed of the vessel. *Slow steaming*, in which vessels sail near or at their minimum speed, therefore, allows vessels to sail more cheaply between two ports than at higher speeds, albeit with a longer duration (see, e.g., [4]). We linearize the bunker consumption of each repositioning vessel in order to more easily model the LSFRP.

Phase-out & Phase-in The repositioning period for each vessel starts at a specific time when the vessel may cease normal operations, that is, it may stop sailing to scheduled visitations and go somewhere else. Each vessel is assigned a different time when it may begin its repositioning, or *phase-out* time. After this time, the vessel may undertake a number of different activities to reach its goal service at low cost. In order to complete the repositioning, each vessel must *phase in* to a slot on the goal service before a time set by the repositioning coordinator. After this time, normal operations on the goal service are set to begin, and all scheduled visitations on the service are to be undertaken. In other words, the repositioning of each vessel and optimization of its activities takes place in the period between two fixed times, the vessel's earliest phase-out time and the latest phase-in time of all vessels. When a port is visited that is not in the initial or goal service, or is visited out of order, it is called an *inducement*. If a port on the initial or goal service is left off of the repositioning vessel's schedule, it is called an *omission*.

Cargo and Equipment Revenue is earned through delivering *cargo* and *equipment* (typically empty containers). We use a detailed view of cargo flows. Cargo is represented as a set of port to port demands with a cargo type, a latest delivery time, an amount of TEU[1] available, and a revenue per TEU delivered. We subtract the cost of loading and unloading each TEU from the revenue to determine the profit per TEU of a particular cargo demand. In contrast to cargo, which can be seen as a multi-commodity flow where each demand is a commodity with a start and end port, equipment can be sent from any port where it is in surplus to any port where it is in demand. We consider both *dry* and *reefer* (refrigerated) cargo, and vessels are assigned a maximum number of reefer containers and a maximum number of all types of containers.

Some ports have equipment, but are not on any service visited by repositioning vessels. These ports are called *flexible* ports, and are associated with flexible visitations. The repositioning coordinator may choose the time a vessel arrives at such visitations, if at all. All other visitations are called *inflexible*, because the time a vessel arrives is fixed.

[1]TEU stands for *twenty-foot equivalent unit* and represents a single twenty-foot container.

Sail-on-service (SOS) Opportunities While repositioning, vessels may use certain services to cheaply sail between two parts of the network. These are called *SOS opportunities*. There are two vessels involved in SOS opportunities, referred to as the *repositioning vessel*, which is the vessel under the control of a repositioning coordinator, and the *on-service vessel*, which is the vessel assigned to a slot on the service being offered as an SOS opportunity. Repositioning vessels can use SOS opportunities by replacing the on-service vessel and sailing in its place for a portion of the service. SOS opportunities save significant amounts of money on bunker fuel, since one vessel is sailing where there would have otherwise been two. Using an SOS can even earn money from the *time-charter bonus*, which is money earned by the liner shipper if the on-service vessel is leased. Utilizing an SOS is subject to a number of constraints, which are described in [9].

Asia-CA3 Case Study Figure 1 shows a subset of a real repositioning scenario in which a vessel must be repositioned from its initial service (the phase-out service), the Chennai-Express, to the goal service (the phase-in service), the Intra-WCSA. The Asia-CA3 service is offered as a SOS opportunity to the vessel repositioning from Chennai Express to Intra-WCSA. One possible repositioning could involve a vessel leaving the Chennai Express at TPP, and sailing to HKG where it can pick up the Asia-CA3, thereby replacing the on-service vessel. The repositioning vessel would then sail along the Asia-CA3 until it gets to BLB, where it can join the Intra-WCSA. Note that no vessel sails on the backhaul of the Asia-CA3, and this is allowed because very little cargo travels on the Asia-CA3 towards Asia.

Modeling the LSFRP We construct a graph containing a number of special structures that take into LSFRP constraints, such as sail-on-service opportunities, the phase-out, and the phase-in. Each node in the graph represents a port visiting a particular port at a particular time. The path of a vessel through the graph defines a repositioning plan for the vessel. The paths of the vessels are required to be node-disjoint, in order to prevent two vessels from being in the same place at the same time or undertaking the same activity. We refer readers to the appendix of [7] for a formal construction of the graph. Using this graph, [9] and [7] present mixed-integer programming models for the LSFRP, which focus on the flows of the vessels through the graph, as well as the flow of cargo along the vessel flows.

3 LAHC

The LAHC algorithm was first introduced in [3] and was further investigated in [10, 1, 5, 2]. LAHC consists of a standard hill climbing algorithm with one key difference: the acceptance criterion compares candidate solution to a solution ℓ iterations ago, in addition to the solution from the previous iteration. Thus, LAHC is allowed

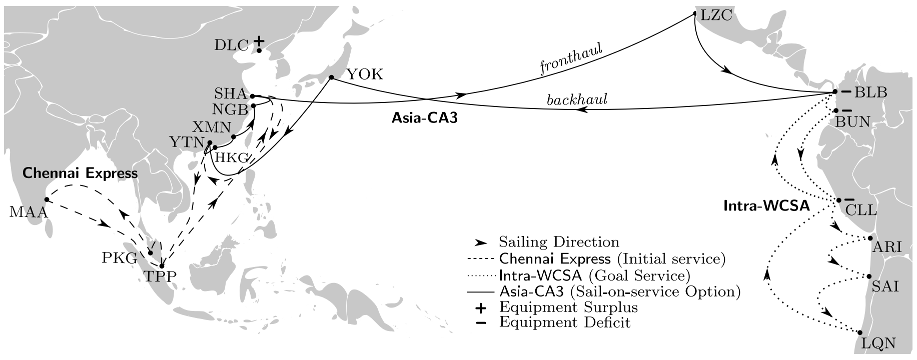

Figure 1: A subset of a case study of the LSFRP, from [8].

to accept degenerating solutions as long as their objective is less than the objective evaluation ℓ iterations ago, helping to prevent LAHC from becoming trapped in a local optimum as in the case of a standard hill climbing algorithm.

Algorithm 1 shows a generic LAHC procedure. The LAHC function accepts three parameters: the optimization problem to solve, p, the objective evaluation (i.e. the fitness function), f, and the number of solutions to store, ℓ. The LAHC is initialized in lines 2 – 7. First, an initial solution is constructed. Then, an array for storing the last ℓ solutions is initialized to store ℓ copies of the initial solution. The following procedure is then performed (lines 9 – 16) until some set of convergence criteria is satisfied. The index of the solution ℓ iterations ago is stored in v, and a candidate solution, s', is proposed by the SELECTNEIGHBOR(s) function. The candidate solution is accepted if the objective of s' is less than the objective of the solution ℓ iterations ago, or the objective is less than the immediately previous solution. The incumbent s^* is then updated to store the best solution seen so far. Finally, the solution is stored in the solution list and the iteration counter is incremented.

Algorithm 1 The LAHC algorithm, based on the pseudo code in [2].

```
1: function LAHC($p$, $f$, $\ell$)
2:     $iter \leftarrow 0$
3:     $s^* \leftarrow$ CREATESOLUTION($p$)
4:     $s \leftarrow s^*$
5:     $sols \leftarrow$ ARRAY($\ell$)
6:     for $k \in \{0, \dots, \ell - 1\}$ do
7:         $sols[k] \leftarrow f(s^*)$
8:     repeat
9:         $v \leftarrow iter \bmod \ell$
10:        $s' \leftarrow$ SELECTNEIGHBOR(S)
11:        if $f(s') < f(sols[v])$ or $f(s') < f(s)$ then
12:            $s \leftarrow s'$
13:        if $f(s') < f(s^*)$ then
14:            $s^* \leftarrow s'$
15:        $sols[v] \leftarrow s$
16:        $iter \leftarrow iter + 1$
17:    until Convergence criteria met
18:    return $s^*$
```

Initial Solutions Three initial solution heuristics were introduced in [7]: direct route, greedy, and shortest path. The direct route heuristic solves a linear assignment problem to match each vessel with a particular final phase-in visitation, ignoring cargo and equipment. The greedy and shortest path heuristics take into account cargo and equipment, but in a non-exact fashion in which the profit from cargo and equipment is added to certain nodes to offset sailing costs. The greedy heuristic generates paths by always selecting outgoing visitation with maximal profit along a vessel path, whereas the shortest path heuristic takes into account the cost of a path so far. All of the heuristics generate solutions that are feasible in terms of node disjointness, but may be temporally infeasible due to flexible nodes.

Neighborhoods There are four neighborhoods in [7] for modifying a solution: add, remove, random path, and demand destination completion. The add neighborhood adds a random visitation to the path of a vessel, and the drop neighborhood removes one. The random path neighborhood drops all visitations after a randomly chosen visitation of a random vessel and performs a random walk to the graph sink. Finally, the demand destination completion neighborhood chooses a random visitation on a random vessel's path that is the origin of some demand that is not being delivered and attempts to create a path to one of the demand's destinations such that the demand can be delivered. Note that all random decisions are performed uniformly at random.

4 Computational Results

We replace the SA component of the algorithm in [7] with an LAHC. Thus, both approaches use the same neighborhoods, initial solution, and even the same restarting procedures. To further ensure that a fair comparison is achieved, we perform the same parameter tuning on LAHC as on SA. Table 1 gives an overview of the parameters considered for LAHC, along with the best parameters found during hand tuning. We use the greedy initial starting solution proposed in [7], as little difference was observed when running the SA on the different starting solutions. The parameters γ, ℓ^{Cargo}, and ℓ^{Eqp} control the cost of flexible arcs, the factor of profit earned from cargo, and the factor of profit earned from equipment, respectively, in the greedy ini-

Category	Parameter	Discretized domain values.	LAHC
GH	γ	0.0, 0.1, 0.25, 0.5, 0.75	0.25
	ℓ^{Cargo}	0.0, 0.1, 0.25, 0.5, 0.75, 0.95	0.1
	ℓ^{Eqp}	0, 10, 50, 100, 250, 500, 1000	100
LAHC	ℓ	10, 100, 1000, 10000, 100000	10000
	r^{Itrs}	1000, 10000, 100000	10000
	$r^{Restart}$	10, 100, 1000	10
Penalty	p	$\{1,2,5,7\}\times10^{\{5,6\}}$, 1×10^7	2×10^6
	p^{PI}	$\{1,2,5,7\}\times10^{\{5,6\}}$, 1×10^7	2×10^6
	p^T	$\{1,2,5,7\}\times10^{\{5,6\}}$, 1×10^7	5×10^6

Table 1: The discretized parameter domains used in hand tuning are given with parameters classified into several categories. The best parameter for LAHC is shown on the right.

tial heuristic. The LAHC parameters ℓ, r^{Itrs}, and $r^{Restart}$ determine the list length, the number of failed iterations before restarting, and the maximum number of restarts, respectively. Finally, the parameters p, p^{PI}, and p^T describe the amount of penalization in the objective for non-node disjoint paths, multiple vessels utilizing the same phase-in slot, and temporal infeasibility within a vessel path, respectively.

We allow both the LAHC and SA to run for a maximum of 10 minutes of CPU time, or until convergence. The SA was allowed 20 reheats, in which the current SA temperature is raised to a constant percentage of its initial value (see, e.g., [6]). For each instance in our dataset, we ran both LAHC and SA 25 times, each time with a different random seed.

We use the same *score* function as in [7] in order to measure the improvement of solutions found over the optimal solutions (when the optimal is known). We use this score function instead of the optimality gap because the optimal objective is sometimes positive, but the heuristics only find negative objectives, leading to meaningless optimality gap calculations. The score for a particular instance is given by

$$score(inst) = \frac{opt(inst) - heuristic(inst)}{opt(inst) - base(inst)},$$

where $heuristic(inst)$ is the objective found by a heuristic algorithm (either SA or LAHC), $base(inst)$ is the objective of the worst starting solution (from the three heuristics used in [7]) and $opt(inst)$ is the optimal objective for the instance, if one is known. The closer the score is to 0, the better the SA performance. Note that in cases where the baseline is the same as the optimal solution (i.e. the starting heuristic found the optimal solution), we set the score to 0 for clarity.

Table 2 shows our results[2] on the confidential dataset from [7]. The table provides the baseline solution provided by the initial solution heuristics, the optimal solution found by CPLEX (when available), and then the objective and score for the best solution found by LAHC (SA), as well as the average objective and score over 25 runs on each instance. The bottom two rows of the table provide the average and standard deviation of the columns, with the exception of the LAHC and SA averages, where the average (standard deviation) over all instances and runs is provided, rather than give an average of averages. Although LAHC has relatively equal performance with the SA on smaller instances (repo1c – repo31c), on larger instances the SA is able to find significantly better solutions, for both the best solution found and the average across the 25 runs on each instance. However, of important note, LAHC is able to find a solution to instance repo42c, a reference instance from our industrial collaborator, which outperforms the one found by SA with the greedy initial starting heuristic. However, SA also finds this solution when using the direct route initial solution heuristic in [7]. Averaging across all of the best objectives found, SA is able to find more than \$240,000 in additional profit over LAHC, and \$200,000 more on average. To gauge the statistical significance of our results, we use a t-test with the null hypothesis that SA does not significantly outperform LAHC. Comparing the best objectives found, we can reject the hypothesis with $p = 0.0065$, and across all executions of SA and LAHC, we reject the hypothesis with a $p < 1 \times 10^4$. Thus, we can conclude that SA outperforms LAHC on the confidential dataset.

The results on the public dataset from [7] are shown in Table 3. On this dataset LAHC is able to slightly outperform SA in terms of score both in the case of the best solution found and the average solution value. However, in terms of the average best solution value as well as the average overall solution value found, SA outperforms LAHC, in the former case by 5.5% and in the latter case by nearly 20%. These results further confirm that LAHC performs well on the small instances of the LSFRP, as good or better than SA, however on larger instances SA is able to outperform. We tested the same null hypothesis on the public dataset as on the confidential dataset, and are unable to reject the hypothesis that SA outperforms LAHC when comparing the best objectives found ($p = 0.7$), however when looking at all instance seed pairs, we can reject the hypothesis with $p < 1 \times 10^4$.

Figure 2 shows the performance of LAHC versus SA for all instance seed pairings on the confidential dataset. Points above the line $y = x$ indicate better performance for SA on an instance seed pair, while points below the line mean that LAHC had a better objective value. While LAHC does outperform on several instances, overall it is unable to match SA on the majority of the runs. Figure 3 shows the same data as Figure 2 for the public dataset. On this dataset, SA and LAHC have similar performance on many instances, although SA is the better choice on several instance clusters.

Although the exact reasons LAHC does not perform as well as the SA on the LSFRP must still be investigated, one possible reason is that the LSFRP contains a multi-commodity flow. To the best of our knowledge, LAHC has not yet been tried on a problem with such a component. When a multi-commodity flow is embedded into a vehicle flow, as in the LSFRP, small changes to the paths of the vehicles can cause large changes in the objective. It is possible that LAHC is unable to respond as effectively to such changes as SA, which takes

[2]All experiments were performed on AMD Opteron 2425 HE processors.

			LAHC				SA			
			Best		Avg.		Best		Avg.	
ID	Baseline	Optimal	Obj.	Score	Obj.	Score	Obj.	Score	Obj.	Score
repo1c	-55.88	-33.91	-33.91	0.000	-33.91	0.000	-33.91	0.000	-33.91	0.000
repo2c	-55.88	-33.91	-33.91	0.000	-33.91	0.000	-33.91	0.000	-33.91	0.000
repo3c	-71.13	-55.58	-62.54	0.448	-62.54	0.448	-55.58	0.000	-55.58	0.000
repo4c	-47.70	-6.30	-8.11	0.044	-8.11	0.044	-8.11	0.044	-8.11	0.044
repo5c	-7.68	0.44	0.44	0.000	0.44	0.000	0.44	0.000	0.44	0.000
repo6c	-7.68	0.44	0.44	0.000	0.44	0.000	0.44	0.000	0.44	0.000
repo7c	-68.95	83.20	83.20	0.000	83.20	0.000	83.20	0.000	83.20	0.000
repo8c	-7.68	0.44	0.44	0.000	0.44	0.000	0.44	0.000	-1.22	0.204
repo9c	-7.68	0.44	0.44	0.000	0.44	0.000	0.44	0.000	-1.17	0.198
repo10c	-13.43	205.76	193.06	0.058	193.06	0.058	193.06	0.058	193.06	0.058
repo11c	39.73	205.76	193.06	0.076	193.06	0.076	193.06	0.076	193.06	0.076
repo12c	-13.43	210.34	205.11	0.023	205.11	0.023	205.11	0.023	205.11	0.023
repo13c	-13.43	210.56	205.34	0.023	205.34	0.023	205.34	0.023	205.34	0.023
repo14c	-13.43	210.56	205.34	0.023	205.34	0.023	205.34	0.023	205.34	0.023
repo15c	-52.80	4.91	0.50	0.076	0.50	0.076	0.50	0.076	0.50	0.076
repo16c	-183.10	4.91	0.50	0.023	0.50	0.023	0.50	0.023	0.50	0.023
repo17c	-52.46	-16.64	-30.39	0.384	-30.40	0.384	-30.39	0.384	-30.40	0.384
repo18c	-52.46	-13.38	-27.13	0.352	-27.13	0.352	-27.13	0.352	-27.13	0.352
repo19c	-52.46	-13.38	-30.39	0.435	-30.40	0.436	-30.39	0.435	-30.40	0.436
repo20c	-44.07	-20.15	-29.90	0.408	-29.91	0.408	-29.90	0.408	-29.90	0.408
repo21c	-44.07	-20.15	-29.90	0.408	-29.91	0.408	-29.90	0.408	-29.90	0.408
repo22c	-48.29	-20.15	-31.72	0.411	-31.74	0.412	-31.72	0.411	-31.73	0.411
repo23c	-32.56	14.07	-15.76	0.640	-16.38	0.653	-15.03	0.624	-16.44	0.654
repo24c	-87.45	-46.30	-52.86	0.160	-52.86	0.160	-52.86	0.160	-52.86	0.160
repo25c	-87.75	-41.07	-50.17	0.195	-50.17	0.195	-50.17	0.195	-50.17	0.195
repo26c	-95.49	-41.07	-57.91	0.309	-57.91	0.309	-57.91	0.309	-57.91	0.309
repo27c	-35.42	2.89	-2.26	0.135	-3.10	0.156	-2.26	0.135	-2.26	0.135
repo28c	-37.32	2.67	-2.48	0.129	-3.28	0.149	-2.48	0.129	-2.48	0.129
repo29c	-38.43	2.67	-2.48	0.125	-3.23	0.144	-2.48	0.125	-2.48	0.125
repo30c	-53.41	0.62	-4.10	0.087	-4.25	0.090	-4.10	0.087	-4.10	0.087
repo31c	-45.80	3.55	2.55	0.020	2.45	0.022	2.55	0.020	2.55	0.020
repo32c	-56.54	1.57	-2.88	0.077	-3.05	0.080	-2.88	0.077	-2.88	0.077
repo33c	-51.95	2.38	-2.06	0.082	-2.61	0.092	-2.06	0.082	-2.54	0.091
repo34c	-286.30	-	-1.29	-	-8.64	-	5.08	-	-0.39	-
repo35c	-1812.71	-	45.31	-	30.31	-	63.43	-	38.04	-
repo36c	-666.46	-	31.73	-	16.63	-	49.76	-	35.47	-
repo37c	-586.78	-	37.45	-	6.19	-	40.97	-	27.48	-
repo38c	-684.21	-	47.72	-	29.15	-	65.69	-	37.50	-
repo39c	-432.91	-	39.21	-	18.60	-	49.71	-	35.96	-
repo40c	-452.54	-	29.32	-	5.98	-	47.94	-	24.83	-
repo41c	-274.47	-	-13.24	-	-16.51	-	-14.52	-	-18.44	-
repo42c	-124.61	-	155.02	-	114.04	-	152.76	-	93.09	-
repo43c	-223.81	-	69.65	-	54.62	-	71.36	-	56.13	-
repo44c	-274.65	-	86.14	-	57.79	-	93.05	-	62.10	-
∅	-166.26	24.43	25.15	0.156	20.08	0.159	27.56	0.142	22.13	0.155
σ	306.33	81.26	74.81	0.176	72.95	0.176	75.20	0.168	72.39	0.168

Table 2: Best objective of LAHC and SA over 25 runs on the confidential dataset, in thousands.

			LAHC				SA			
			Best		Avg.		Best		Avg.	
ID	Baseline	Optimal	Obj.	Score	Obj.	Score	Obj.	Score	Obj.	Score
repo1p	-52.84	-39.83	-39.83	0.000	-39.83	0.000	-39.83	0.000	-39.83	0.000
repo2p	-52.84	-39.83	-39.83	0.000	-39.83	0.000	-39.83	0.000	-39.83	0.000
repo3p	-87.80	-61.77	-61.77	0.000	-61.77	0.000	-61.77	0.000	-61.77	0.000
repo4p	-55.52	-46.62	-46.62	0.000	-46.62	0.000	-46.62	0.000	-46.62	0.000
repo5p	-49.82	-8.21	-8.21	0.000	-8.21	0.000	-8.21	0.000	-8.98	0.019
repo6p	-49.82	-8.21	-8.21	0.000	-8.21	0.000	-8.21	0.000	-8.65	0.010
repo7p	-87.80	-11.49	-11.49	0.000	-11.49	0.000	-11.49	0.000	-11.49	0.000
repo8p	-49.82	-8.21	-8.21	0.000	-8.21	0.000	-11.54	0.080	-12.36	0.100
repo9p	-49.82	-8.21	-8.21	0.000	-8.21	0.000	-12.44	0.102	-12.44	0.102
repo10p	-7.78	137.61	135.12	0.017	135.12	0.017	135.12	0.017	135.12	0.017
repo11p	-7.78	137.61	135.12	0.017	135.12	0.017	135.12	0.017	135.12	0.017
repo12p	-7.78	138.55	136.07	0.017	136.07	0.017	136.07	0.017	136.07	0.017
repo13p	-7.78	138.86	136.07	0.019	136.07	0.019	136.07	0.019	136.07	0.019
repo14p	-7.78	138.86	136.07	0.019	136.07	0.019	136.07	0.019	136.07	0.019
repo15p	-119.14	-36.59	-36.59	0.000	-37.29	0.008	-36.59	0.000	-37.76	0.014
repo16p	-228.99	-36.59	-36.59	0.000	-36.59	0.000	-36.59	0.000	-38.20	0.008
repo17p	-32.64	-9.36	-21.44	0.519	-22.57	0.567	-21.36	0.516	-22.02	0.544
repo18p	-27.07	5.22	-11.34	0.513	-11.78	0.526	-11.30	0.512	-11.37	0.514
repo19p	-36.74	5.22	-22.21	0.654	-22.90	0.670	-21.44	0.635	-22.10	0.651
repo20p	-34.08	-11.85	-20.16	0.374	-20.16	0.374	-20.16	0.374	-20.16	0.374
repo21p	-34.08	-11.85	-20.16	0.374	-20.16	0.374	-20.16	0.374	-20.16	0.374
repo22p	-41.58	-11.85	-23.47	0.391	-24.07	0.411	-23.39	0.388	-23.59	0.395
repo23p	-34.80	5.22	-22.72	0.698	-23.40	0.715	-22.72	0.698	-23.32	0.713
repo24p	-62.96	-53.89	-53.89	0.000	-53.89	0.000	-53.89	0.000	-53.89	0.000
repo25p	-66.33	-53.13	-53.89	0.058	-53.89	0.058	-53.89	0.058	-53.89	0.058
repo26p	-66.33	-53.13	-53.89	0.058	-53.89	0.058	-53.89	0.058	-53.89	0.058
repo27p	-51.56	-28.20	-28.53	0.014	-28.64	0.019	-28.53	0.014	-28.54	0.014
repo28p	-50.83	-32.13	-32.14	0.001	-32.31	0.010	-32.14	0.001	-32.16	0.001
repo29p	-49.36	-32.13	-32.14	0.001	-32.31	0.010	-32.14	0.001	-32.15	0.001
repo30p	-45.86	5.72	3.78	0.038	2.22	0.068	5.06	0.013	4.19	0.030
repo31p	-48.21	-12.08	-12.08	0.000	-12.17	0.002	-12.08	0.000	-12.08	0.000
repo32p	-44.50	-10.92	-10.92	0.000	-10.93	0.000	-10.92	0.000	-10.92	0.000
repo33p	-44.34	-10.92	-10.92	0.000	-11.19	0.008	-10.92	0.000	-11.18	0.008
repo34p	-355.56	-	-24.38	-	-27.78	-	-23.83	-	-23.97	-
repo35p	-1353.80	-	-24.58	-	-27.77	-	-24.02	-	-24.74	-
repo36p	-799.76	-	-25.55	-	-28.11	-	-23.83	-	-24.13	-
repo37p	-933.23	-	-24.25	-	-27.45	-	-23.83	-	-24.02	-
repo38p	-818.55	-	-25.48	-	-27.72	-	-23.83	-	-25.20	-
repo39p	-639.55	-	-25.80	-	-27.65	-	-23.83	-	-24.08	-
repo40p	-659.15	-	-25.51	-	-27.80	-	-23.83	-	-24.09	-
repo41p	-294.08	-	-82.21	-	-90.92	-	-70.92	-	-76.60	-
repo42p	-214.73	-	98.93	-	82.49	-	88.25	-	77.65	-
repo43p	-245.58	-	31.39	-	9.61	-	38.54	-	28.31	-
repo44p	-325.07	-	48.03	-	28.26	-	44.31	-	28.50	-
$\oslash$	-189.40	2.30	-3.02	0.115	-5.11	0.120	-2.85	0.119	-4.07	0.124
σ	295.48	60.33	57.03	0.210	56.41	0.216	56.26	0.207	55.77	0.210

Table 3: Best objective of LAHC and SA over 25 runs on the public dataset, in thousands.

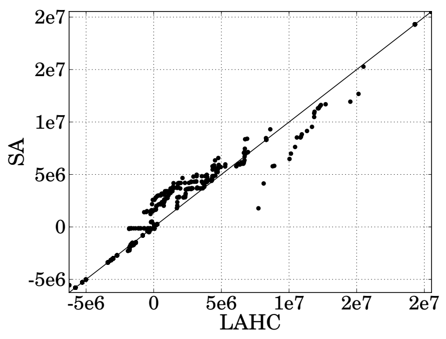

Figure 2: The objective performance of SA versus LAHC across on the confidential dataset all 25 runs of each instance.

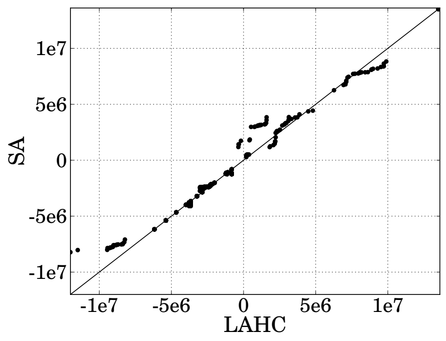

Figure 3: The objective performance of SA versus LAHC across on the public dataset all 25 runs of each instance.

into account the change in objective value in its acceptance criterion. The flexible visitations of the LSFRP also pose a challenge for LAHC, in that they too result in large objective changes based on small solution changes. Another explanation for the performance of LAHC could be that the penalized nature of the LSFRP is difficult for LAHC, as the only work on LAHC considering penalized objectives is [1].

5 Conclusion

We presented a computational comparison of LAHC with SA on the LSFRP, showing that LAHC is unable to achieve the performance of SA. Despite using the same neighborhoods as the SA, and undergoing the same tuning process, LAHC was unable to overcome local optima as effectively as SA. The LSFRP is an difficult to solve real-world problem that is important to the liner shipping industry. Thus, for future work we intend to look into hybridizations of LAHC and SA for solving the LSFRP, as well as extensions to LAHC proposed in [2].

Acknowledgments

We thank Stefan Voß for his comments and ideas regarding this work. We also thank the anonymous referee for his or her insightful comments. Kevin Tierney is sponsored by the Danish Council for Strategic Research as part of the ENERPLAN research project.

References

[1] A. Abuhamdah. Experimental result of late acceptance randomized descent algorithm for solving course timetabling problems. *IJCSNS International Journal of Computer Science and Network Security*, 10:192–200, 2010.

[2] E. K. Burke and Y. Bykov. The late acceptance hill-climbing heuristic. Technical Report CSM-192, University of Stirling, 2012.

[3] E.K. Burke and Y. Bykov. A late acceptance strategy in hill-climbing for exam timetabling problems. In *PATAT 2008 Conference, Montreal, Canada*, 2008.

[4] J. Meyer, R. Stahlbock, and S. Voß. Slow steaming in container shipping. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 1306–1314. IEEE, 2012.

[5] E. Ozcan, Y. Bykov, M. Birben, and E.K. Burke. Examination timetabling using late acceptance hyper-heuristics. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 997–1004. IEEE, 2009.

[6] J. Taheri and A.Y. Zomaya. A simulated annealing approach for mobile location management. *Computer communications*, 30(4):714–730, 2007.

[7] K. Tierney, B. Áskelsdóttir, R.M. Jensen, and D. Pisinger. Solving the liner shipping fleet repositioning problem with cargo flows. Technical Report TR-2013-165, IT University of Copenhagen, January 2013.

[8] K. Tierney, A.J. Coles, A.I. Coles, C. Kroer, A.M Britt, and R.M. Jensen. Automated planning for liner shipping fleet repositioning. In L. McCluskey, B. Williams, J.R. Silva, and B. Bonet, editors, *Proceedings of the 22nd International Conference on Automated Planning and Scheduling*, pages 279–287, 2012.

[9] K. Tierney and R. M. Jensen. The Liner Shipping Fleet Repositioning Problem with Cargo Flows. In Hao Hu, Xiaoning Shi, Robert Stahlbock, and Stefan Voß, editors, *Computational Logistics*, volume 7555 of *Lecture Notes in Computer Science 7555*, pages 1–16. Springer, 2012.

[10] J. Verstichel and G.V. Berghe. A late acceptance algorithm for the lock scheduling problem. *Logistik Management*, pages 457–478, 2009.