J Sched
DOI 10.1007/s10951-016-0469-x

A Step Counting Hill Climbing Algorithm applied to University Examination Timetabling

Yuri Bykov[1] · **Sanja Petrovic**[1]

Abstract This paper presents a new single-parameter local search heuristic named step counting hill climbing algorithm (SCHC). It is a very simple method in which the current cost serves as an acceptance bound for a number of consecutive steps. This is the only parameter in the method that should be set up by the user. Furthermore, the counting of steps can be organised in different ways; therefore, the proposed method can generate a large number of variants and also extensions. In this paper, we investigate the behaviour of the three basic variants of SCHC on the university exam timetabling problem. Our experiments demonstrate that the proposed method shares the main properties with the late acceptance hill climbing method, namely its convergence time is proportional to the value of its parameter and a non-linear rescaling of a problem does not affect its search performance. However, our new method has two additional advantages: a more flexible acceptance condition and better overall performance. In this study, we compare the new method with late acceptance hill climbing, simulated annealing and great deluge algorithm. The SCHC has shown the strongest performance on the most of our benchmark problems used.

Keywords Optimisation · Metaheuristics · Simulated annealing · Late acceptance hill climbing · Step counting hill climbing · Exam timetabling

✉ Sanja Petrovic
sanja.petrovic@nottingham.ac.uk

Yuri Bykov
mail@yuribykov.com

[1] Nottingham University Business School, Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK

1 Introduction

A single-parameter local search metaheuristic called late acceptance hill climbing algorithm (LAHC) was proposed by Burke and Bykov (2008). The main idea of LAHC is to compare in each iteration a candidate solution with the solution that has been chosen to be the current one several iterations before and to accept the candidate if it is better. The number of the backward iterations is the only LAHC parameter referred to as "history length". An extensive study of LAHC was carried in (Burke and Bykov 2012) where the salient properties of the method have been discussed. First, its total search/convergence time was proportional to the history length, which was essential for its practical use. Also, it was found that despite apparent similarities with other local search metaheuristics such as simulated annealing (SA) and great deluge algorithm (GDA), LAHC had the underlying distinction, namely it did not require a guiding mechanism like, for example, cooling schedule in SA. This provided the method with effectiveness and reliability. It was demonstrated that LAHC was able to work well in situations where the other two heuristics failed to produce good results.

Although LAHC is a relatively new algorithm, its unique characteristics attracted a particular attention of the research community. A number of authors have published their own studies on LAHC applied to different problems, such as lock scheduling (Verstichel and Vanden Berghe 2009), liner shipping fleet repositioning (Tierney 2013), balancing two-sided assembly lines (Yuan et al. 2015), travelling purchaser problem (Goerler et al. 2013), etc. In addition, several modifications of this method were proposed, such as late acceptance randomised descent algorithm (Abuhamdah 2010) and multi-objective late acceptance algorithm (Vancroonenburg and Wauters 2013). A reheating mechanism was embedded in

Published online: 02 April 2016

LAHC by Swan et al. (2013). Also, LAHC was hybridised with other techniques (Abdullah and Alzaqebah 2014) and was successfully employed in the recently developed hyper-heuristic approach, see (Özcan et al. 2009) and (Jackson et al. 2013).

Besides its presence in the scientific literature, LAHC appeared to be beneficial in the scientific competitions and real-world applications. This method won the first place prize in the International Optimisation Competition (http://www.solveitsoftware.com/competition) in December 2011. Furthermore, this method was employed in entry algorithms by two research groups (J17 and S5) in ROADEF/EURO Challenge 2012 (http://challenge.roadef.org/2012/en/). They won fourth and eighth places respectfully in their categories. LAHC won the first place prize in VeRoLog Solver Challenge 2014 in June 2014 (http://verolog.deis.unibo.it/news-events/general-news/verolog-solver-challenge-2014-final-results). Also, LAHC is currently employed in at least two real-world software systems: Rasta Converter project hosted by GitHub Inc. (US) (https://github.com/ilmenit/RastaConverter) and OptaPlanner, an open source project by Red Hat (http://www.optaplanner.org).

Motivated by the success of LAHC, the idea of single-parameter and cooling schedule free local search methodology is developed further (Bykov and Petrovic 2013). We propose a new metaheuristic, which keeps all the good characteristics of LAHC but it is even simpler, more powerful and offers some additional advantages. The method is named step counting hill climbing algorithm (SCHC). In this study, we present a comprehensive investigation into the properties of this algorithm. The evaluation of SCHC is carried out on the university exam timetabling problem. In the choice of the problem domain, we follow many previous authors who consider the exam timetabling to be a good benchmark in their experiments. Over the years almost all metaheuristic methods were applied to the examination timetabling, such as SA (Thompson and Dowsland 1996), tabu search (Di Gaspero and Schaerf 2001), GDA (Burke et al. 2004), evolutionary methods (Erben 2001), multi-criteria methods (Petrovic and Bykov 2003), fuzzy methods (Petrovic et al. 2005) and grid computing (Gogos et al. 2010). In addition, the exam timetabling was widely used in the evaluation of different hybrid methods (Merlot et al. 2003; Abdullah and Alzaqebah 2014), as well as hyper-heuristics (Burke et al. 2007). The initial study of the late acceptance hill climbing algorithm was also done on the exam timetabling problems (Burke and Bykov 2008). More information about the exam timetabling studies can be found in a number of survey papers, including Carter et al. (1996), Schaerf (1999), Burke and Petrovic (2002) and Qu et al. (2009).

The exam timetabling is usually defined as a minimization problem. Hence, to make the description of the algorithm consistent with our experiments in the rest of our paper we assume a lower value of the cost function the better quality of a result

The paper is organised as follows: the description of SCHC is given in the next section. In Sect. 3, we describe our experimental environment including the benchmark datasets and experimental software. The investigation into the properties of our method is presented in Sect. 4, while Sect. 5 is devoted to the performance of SCHC and its comparison with other techniques. Finally, Sect. 6 presents the comparison of the SCHC results with the published ones, followed by some conclusions and discussion about the future work.

2 Step counting hill climbing algorithm

2.1 Description of the basic SCHC heuristic

The idea of the SCHC is to embed a counting mechanism into the hill climbing (HC) algorithm in order to deliver a new quality to the method. Similar to HC, our heuristic operates with a control parameter, which we refer to as a "cost bound" (B_c). The cost bound denotes the best non-acceptable value of the candidate cost function, i.e. at each iteration the algorithm accepts any candidate solution with the cost lower (better) than B_c and rejects the ones whose cost is higher (worse) than B_c. The acceptance of the candidates with cost equal to B_c depends on particular situation (see below). From this point of view, in the greedy HC the cost bound is equal to the cost of the best found solution, which can be changed at any iteration. In contrast, the main idea of the SCHC is *to keep the given cost bound not just for one, but for a number of consecutive iterations*.

As other local search heuristics, SCHC starts from a random initial solution and the value of B_c is equal to the initial cost function. Then the algorithm starts counting the consecutive steps (n_c) and when their number exceeds a given counter limit (L_c) the cost bound is updated, i.e. we make it equal to the new current cost while the counter n_c is reset to 0. After further L_c steps B_c is updated and n_c is reset again and this is repeated until a stopping condition is met. Throughout the search, the algorithm accepts only candidates with the cost less than the current cost bound, which means that with each update the value of the cost bound becomes lower and lower; this guarantees the search progresses towards the best achievable solution.

The proposed algorithm has only one input parameter and that is the value of the counter limit L_c, which should be specified by the user. Apart from that no additional initialisation is required in the method. Our preliminary tests have shown that in order to guarantee the progress of the search, SCHC has to accept candidate solutions with the cost only better (not better or equal) than B_c. However, there is one exception to this rule; it is worthy to accept the candidates

with the cost equal to B_c when the current cost is also equal to B_c, which happens after the update of the cost bound. This is provided by a combination of the SCHC acceptance rule with the greedy rule, i.e. a candidate is also accepted if it is better or equal to the current cost. This enhancement of the acceptance rule provides an extra effectiveness of the search, especially at its final stage where it helps to avoid the effect of "premature convergence". Thus, at each iteration the SCHC acceptance rule can be expressed by formula (1).

$$C\left(s^*\right) < B_c \quad \text{or} \quad C\left(s^*\right) \leq C\left(s\right) \tag{1}$$

In this formula, $C(s)$ represents the current cost and $C(s^*)$ the candidate cost, where s and s^* are the current and the candidate solutions, respectively. The complete pseudocode of the initial variant of the SCHC algorithm, which will be called in the rest of this study as "SCHC-all" heuristic is presented in Fig. 1.

2.2 Further variants of SCHC

Apart from the basic counting mechanism introduced in the previous section there are many other ways to count the steps during the search. This is a major source of flexibility in the method. For example, we can make it dependent on the quality of candidate solutions. Among possible implementations of this idea, in this study we focus our investigations on three variants of SCHC, which:

1. Counts all moves (SCHC-all).
2. Counts only accepted moves (SCHC-acp).
3. Counts only improving moves (SCHC-imp).

As two additional SCHC heuristics, SCHC-acp and SCHC-imp, are different from SCHC-all by their counting mechanisms only, we present the pseudocodes of just these mechanisms. They are depicted in Figs. 2 and 3, which contain only the parts different from Fig. 1, while the main search

```
Produce an initial solution s
Calculate an initial cost function C(s)
Initial cost bound Bc:=C(s)
Initial counter nc:=0
Specify Lc
Do until a chosen stopping condition
   Construct a candidate solution s*
   Calculate the candidate cost function C(s*)
   If C(s*)<Bc or C(s*)≤C(s)
      Then accept the candidate s:=s*;
      Else reject the candidate s:=s
   Increment the counter nc:=nc+1
   If nc ≥ Lc
      Then update the bound Bc:=C(s);
           reset the counter nc:=0
```

Fig. 1 The pseudocode of SCHC-all heuristic

```
   If C(s*)<Bc or C(s*)≤C(s)
      Then accept the candidate s:=s*;
           increment the counter nc:=nc+1
      Else reject the candidate s:=s
   If nc ≥ Lc
      Then update the bound Bc:=C(s);
           reset the counter nc:=0
```

Fig. 2 The counting/acceptance mechanism of the SCHC-acp heuristic

```
   If C(s*)<C(s)
      Then increment the counter nc:=nc+1
   If C(s*)<Bc or C(s*)≤C(s)
      Then accept the candidate s:=s*;
      Else reject the candidate s:=s
   If nc ≥ Lc
      Then update the bound Bc:=C(s);
           reset the counter nc:=0
```

Fig. 3 The counting/acceptance mechanism of SCHC-imp heuristic

loop written in the first eight lines of Fig. 1 is the same for all variants.

Of course, the variety of potential counting mechanisms is not limited to the ones discussed above. For example, it is possible to develop an intermediate heuristic between SCHC-all and SCHC-acp. This can be done when incrementing the counter n_c by 1 at all moves and by 2 at accepted moves. Also, we can count unaccepted moves or accepted worsening ones either alone or in any combination. Furthermore, if overlooking the single-parameter conception, we can propose SCHC allowing L_c to vary. For example, we can select two different values of the counter limit and interchange them during the search in order to adapt it to some particular search conditions, or we can increase L_c as the run progresses, for example, each time the search goes idle. Along with that, any alternative rule for the variation of L_c could be defined. Interestingly, the SCHC-acp with fixed L_c can be regarded as SCHC-all with variable L_c and vice versa. Finally, the whole counting mechanism can be completely transformed at any point of the search. Thus, the simplicity of the dynamic adjustment of the search process suggests that SCHC can serve as a good platform for experiments with different search patterns, for investigating the search behaviour and for developing self-adaptive heuristics.

3 The experimental environment

3.1 Exam timetabling problem

The university exam timetabling problem, on which we evaluate the proposed SCHC algorithm, is a difficult NP-hard problem. It represents mathematically a real-world task of assigning university exams to timeslots and usually rooms.

Generally, these problems contain a high variety of hard and soft constraints. The hard constraints should not be violated in a feasible solution, while the violations of the soft constraints should be minimised. The remaining number of the violations of soft constraints in a solution denotes its quality, measured by a cost function. The most common hard constraint is that no students should sit two exams in the same time. Other hard constraints reflect the limitations on room capacities, the utilisation of specific equipment, pre-defined sequences of exams, etc. The soft constraints represent students, examiners and administration preferences, such as time intervals between student's exams, additional time for marking large exams, etc.

Although the exam timetabling requirements are different in different institutions, in this study we use the specification given in the examination track of the Second International Timetabling Competition ITC2007 (http://www.cs.qub.ac.uk/itc2007/). This site contains a collection of 12 real-world exam timetabling instances, which we use as benchmarks in our experiments with SCHC. Also, this site provides an online validator of the results and a complete description of the hard and soft constraints, which is also published in (McCollum et al. 2010). The characteristics of the instances are presented in Table 2 in Sect. 4.3, in order to facilitate a study into relation between the problem characteristics and the results of our experiments.

3.2 Application details

In this study we adopted software, which was used in experiments with LAHC and described in (Burke and Bykov 2012). It is developed in Delphi 2007 and run on PC Intel Core i7-3820 3.6 GHz, 32 GB RAM under OS Windows 7 64 bit.

The search algorithm starts from the generation of a feasible initial solution. The exams are assigned to timeslots using the Saturation Degree graph colouring heuristic. At the same time the exams are randomly assigned to rooms. If the solution is not feasible, then some exams are rescheduled and the initialization procedure starts again. After the generation of the initial solution the heuristic search is run where at each iteration a candidate solution is produced using four types of moves:

- *Room move* a random exam is moved into a different, randomly chosen room within the same timeslot.
- *Shift move* a random exam is moved into different, randomly chosen timeslot and room. If this move generates an infeasible solution, the algorithm tries to restore the feasibility using the Kempe Chain procedure, which was studied for Graph Colouring Problem by Johnson et al. (1991).
- *Swap move* the algorithm selects two random exams and swaps their timeslots. The rooms again are chosen randomly, while the Kempe Chains are used in case of infeasibility.
- *Slot move* two randomly chosen timeslots are interchanged including all their exams and rooms.

These four types of moves are selected randomly in equal proportions. If the move produces an infeasible candidate, it is just rejected and a new iteration is started. The iteration loop is terminated when no further improvement is possible, i.e. at the convergence state. This state is detected when the number of non-improving (idle) moves since the last improvement reaches at least 1 % of the total number of moves. Also the number of idle moves should be greater than 100 in order to prevent the termination at the beginning. The reason of the use of this stopping condition is discussed in the next section.

4 The investigation into the properties of SCHC

4.1 Cost drop diagrams with different L_c

The study of the properties of SCHC starts from the analysis of the algorithmic response to the variation of its single parameter. Hence, in our first experiment, we investigate the algorithm's cost drop diagrams. To produce these diagrams we run each variant of SCHC three times with different values of $L_c = 2000$, 10000 and 20000. Each second the current cost was depicted as a point on a plot where the horizontal axis represents the current time and the vertical axis represents the current cost. An example of such a diagram for *Exam_1* problem produced by SCHC-all heuristic is given in Fig. 4. The diagrams produced for other instances by all three studied SCHC heuristics are similar to the presented one. The difference between these diagrams is just in their time and cost scales and that will be discussed in detail in the next section.

The diagram in this figure demonstrates two major properties of SCHC:

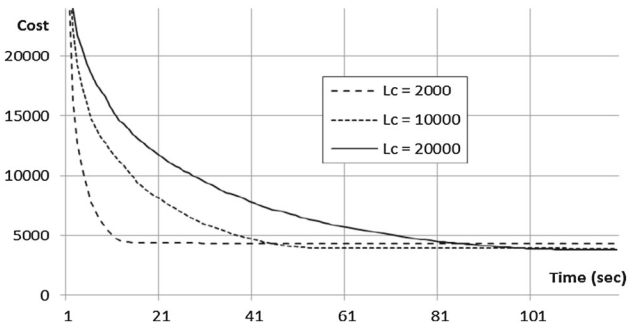

Fig. 4 The cost drop diagram of SCHC-all heuristic applied to *Exam_1* problem with different values of L_c

Table 1 The average results of SCHC with $L_c = 0$ and $L_c = 200$

Instance	$L_c = 0$ (HC)		$L_c = 200$					
			SCHC-all		SCHC-acp		SCHC-imp	
	Cost	Time (s)	Cost	Time (s)	Cost	Time (s)	Cost	Time (s)
Exam_1	6664	0.655	5961	1.27	4866	9.2	4777	11.1
Exam_2	972	0.218	794	0.53	700	1.2	684	1.33
Exam_3	12967	1.65	11082	3.18	10006	7.78	9867	8.34
Exam_4	18183	0.494	16585	0.97	14673	14.9	14276	20.4
Exam_5	4007	0.392	3444	0.99	3321	1.54	3220	2.0
Exam_6	27456	0.116	26814	0.242	26457	0.75	26434	0.84
Exam_7	6501	0.775	5811	1.69	5057	6.04	5012	6.6
Exam_8	10115	0.414	9267	0.826	8086	7.08	7999	7.52
Exam_9	1392	0.054	1284	0.097	1179	0.24	1176	0.26
Exam_10	14070	0.1	13590	0.185	13415	0.344	13384	0.377
Exam_11	37253	2.44	32305	5.06	29431	13.1	28956	14.0
Exam_12	5707	0.024	5559	0.038	5436	0.074	5429	0.08

1. This algorithm converges, i.e. the heuristic search procedure lowers the cost until a certain value, after which it does not provide any further improvement. This property is the same as for other local search methods including HC, SA, LAHC, etc. The presence of this property suggests the use of a common termination condition for such a technique; the search should be stopped exactly at the moment of convergence (the earlier or later termination reduces the effectiveness of the method (see Burke and Bykov (2012)). The identification of the convergence state can be done by a well-established procedure available in the literature described in the previous section, i.e. when the number of idle moves exceeds a given limit.
2. The variation of L_c affects the convergence time. The larger the counter limit the slower the current cost drops and it takes longer time to reach the convergence state. Having a search that is automatically terminated at the point of convergence, a user can regulate the total search time by varying L_c. This property has also similarities with other methods. For example, in LAHC the history length also affects the convergence time while in SA the search time can be regulated by the user-defined cooling schedule.

4.2 The comparison of SCHC with hill climbing

An analysis of the second property of SCHC suggests that its search time can be prolonged to any extent with the increase of the counter limit, i.e. the value of L_c has no theoretical upper bound. However, it has the lower bound, which is 0. In this case, the counter is updated at each iteration and all three proposed variants of the method degenerate into greedy Hill Climbing. The same effect will be achieved when assigning any negative value to L_c. This is the fastest variant of SCHC, so the increase of the counter limit does make sense only if this allows to achieve better final results than HC.

To demonstrate that, in the next series of experiments we have applied the discussed above termination rule and run SCHC with $L_c = 0$, which is the equivalent to HC, and all three variants with $L_c = 200$. Each variant was run on each benchmark instance 50 times. The average results and run times are presented in Table 1.

In this table all results produced by either variant of SCHC with $L_c = 200$ are better than that of HC, although to a different extent. In SCHC-all the search time is approximately twice longer than in HC, which causes a modest improvement of the results. However, in SCHC-acp and SCHC-imp the increase of the search time is relatively higher and, correspondingly, the improvement of the results is even more distinct.

4.3 The investigation into L_c-diagrams

In our next series of experiments, the time-related properties of SCHC are investigated more deeply by analysing L_c-time and L_c-cost diagrams. To construct these diagrams, we run the three studied variants of SCHC on each benchmark instance large number of times (over 1500) while randomly varying L_c. At each run, the specified counter limit and the resulting run time and cost were recorded. After completing the calculations, the experimental data were aggregated in a form of diagram. Figure 5 demonstrates the dependence of the run time on L_c for *Exam_1* dataset solved by SCHC-all heuristic. Here each point represents the result of a single run, and its position corresponds to the specified L_c (in the

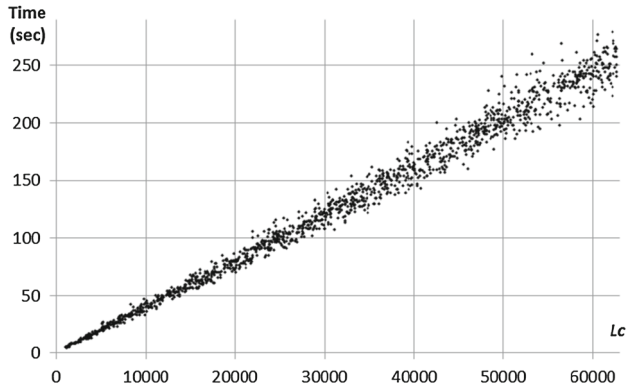

Fig. 5 The dependence of the run time on L_c for *Exam_1* dataset with SCHC-all heuristic

horizontal axis) and the resulting run time T (in the vertical axis).

Although the points on this diagram are relatively scattered, which is typical to any stochastic method, their general distribution forms more or less a straight line. This suggests that the convergence time is approximately proportional to the value of the algorithmic parameter L_c. This property of SCHC is similar to the LAHC one studied by Burke and Bykov (2012), where the authors mentioned its high practical importance, i.e. when the angle coefficient (ratio T/L_c) of the distribution is known, the complete search procedure can be fitted into a given available time. The importance of the pre-definition of the search time was underlined in (Burke et al. 2004) especially for long-time searches, which are practiced in pursuit for a higher quality of results.

Our further experiments reveal that the above property is common for SCHC. Although Fig. 5 presents the linear behaviour of SCHC-all variant applied to *Exam_1* instance, in all other diagrams for three variants of SCHC and 12 benchmark instances the points are also distributed linearly. The diagrams differentiate only by the angles of the produced distributions, i.e. the angle coefficients T/L_c are highly different for different instances and variants of SCHC. However, our preliminary observations have revealed certain tendencies in the values of these coefficients. In particular, the highest values are typical for SCHC-imp, slightly lower for SCHC-acp and much lower for SCHC-all heuristics. Considering a single variant of SCHC, a certain dependency of the coefficients on the size of a dataset can be also noticed. The general tendency is the following: the larger the instance, the larger the angle coefficient. The coefficients together with the main characteristic of our benchmark instances sorted by the number of exams are shown in Table 2.

In this table, the proposed tendency of the dependence of T/L_c on the instance size can be observed for the majority of problems. However, there are a number of exceptions to this rule. For example, the coefficients for *Exam_2* instance are much smaller than for the other problems of the same or even lower size. Another anomaly is in *Exam_4* problem. This dataset has only one room, so the "room moves" described in Sect. 3.2 are not applicable here. De facto, we are dealing here with a different problem formulation. The oddity of this problem is also seen in its coefficients. The coefficient for SCHC-all fits into the above tendency, but coefficients for both SCHC-acp and SCHC-imp are much higher than can be expected for a problem of this size. In addition, *Exam_3* and *Exam_11* instances represent the same dataset where *Exam_11* is just a more constrained variant, i.e. it should be scheduled into a less number of timeslots and rooms. Correspondingly, we see the different values of T/Lc for these problems in Table 2.

This tendency was also observed in LAHC by (Burke and Bykov 2012), who proposed that some other factors, together with the size of problems, could also affect the values of the angle coefficients, such as constraints, conflict density, etc. If assuming that the amount of these factors somehow defines the *hardness* of a problem, then the angle coefficient might reflect, to some extent, this "bulk" hardness. The investigation into the hardness of different problems represents an important area of combinatorial optimisation studies. There are a number of theoretical publications, where the authors investigate the hardness based on the analysis of problem characteristics, for example (Smith-Miles and Lopes 2012). However, the idea proposed here suggests an alternative way, i.e. to use a *heuristic measure* of the hardness of different problems. This means that a heuristic algorithm can serve not just for solving a problem, but also as a tool for measuring its hardness.

To advocate the importance of the investigations into the hardness of optimisation problems, we analyse L_c-cost diagrams of the datasets. These diagrams are plotted using the previous experimental data, which were already used in the diagram in Fig. 5, but now the diagrams show the dependence of the final cost on the specified L_c. The example of such a diagram for SCHC-all heuristic applied to *Exam_1* problem is shown in Fig. 6. Here, once again, the result of each run is depicted as a point, whose horizontal coordinate represents L_c and the vertical coordinate represents the final cost.

This diagram demonstrates a clear dependence of the final cost on L_c and correspondingly on the total run time as the time is linearly dependent on L_c, i.e. the larger the counter limit (the longer the search)—the better the result. For example, despite the scatter, any, even the worst one, result with $L_c = 50000$ is *guaranteed better* than any of the results with $L_c = 5000$. Obviously, this diagram confirms the opinion of Johnson et al. (1989) for SA that "up to a certain point, it seems to be better to perform one long run than to take the best of a time-equivalent collection of shorter runs".

However, an opinion exactly opposite to the Johnson's ones is present in the literature. Many authors consider to

Table 2 The characteristics of the ITC2007 instances and their T/L_c coefficients

Instance	Number of exams	Number of timeslots	Number of rooms	Density	Coefficient T/L_c $(\times 10^{-3})$		
					SCHC-all	SCHC-acp	SCHC-imp
Exam_12	78	12	50	0.18	0.105	0.23	0.28
Exam_9	169	25	3	0.078	0.26	0.99	1.12
Exam_10	214	32	48	0.05	0.47	1.17	1.35
Exam_6	242	16	8	0.062	0.64	3.4	3.8
Exam_4	273	21	1	0.15	1.72	56	73
Exam_8	598	80	8	0.046	3.1	36	42
Exam_1	607	54	7	0.5	4.2	59	65
Exam_2	870	40	49	0.012	1.94	4.8	5.5
Exam_3	934	36	48	0.026	8.7	29	31
Exam_11	934	26	40	0.026	12.8	47	53
Exam_5	1018	42	3	0.0087	3.5	7.1	7.6
Exam_7	1096	80	15	0.019	6.3	28	33

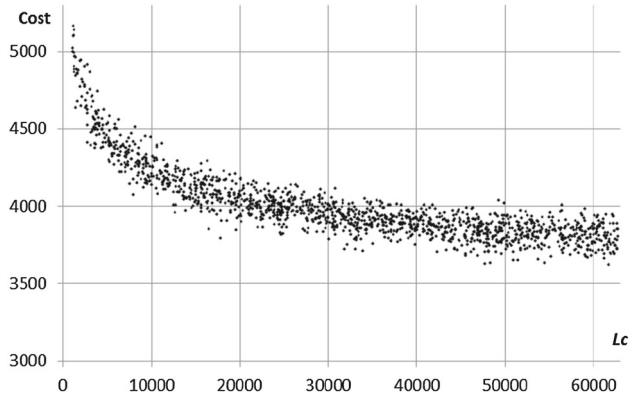

Fig. 6 The dependence of the final cost on L_c for *Exam_1* problem with SCHC-all heuristic

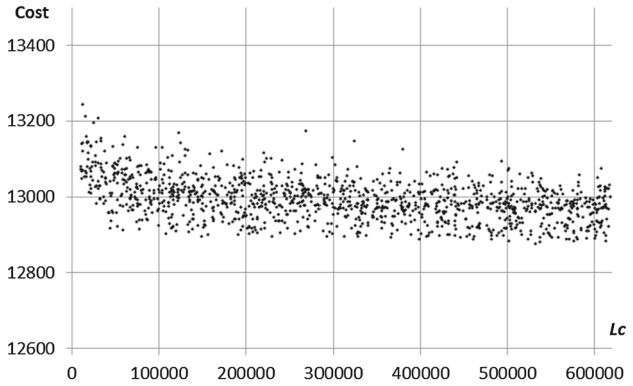

Fig. 7 The dependence of the final cost on L_c for *Exam_10* problem with SCHC-all heuristic

be more effective to produce a number of short runs while employing various "multi-start" or "reheating" strategies and then to pick up the best result. For example, Boese et al. (1994) indicated that "several studies have shown greedy multi-start superior to simulated annealing in terms of both solution quality and run time".

Having two so contrasting opinions from the trusted sources we can assume that the origin of such a dilemma is in the diversity of the studied problems. For example, in our exam timetabling collection the shape of L_c-cost diagrams is not the same for all benchmark datasets in contrast to the L_c-time diagrams, which are quite similar. As an illustration, Fig. 7 depicts the L_c-cost diagram for *Exam_10* problem, whose shape is quite different from Fig. 6.

Except for the very small values of L_c, this diagram does not expose a sensible dependence of the final cost on the value of the counter limit, i.e. the average and the best costs are almost the same either with smaller L_c or with larger L_c, although the latter causes a longer search time. In this situation, a more effective strategy is to produce multiple short runs and pick up the best result among them.

However, the idea of the selection of the best search strategy by plotting the L_c-cost diagrams has no practical value. First, this diagram can be obtained only after running the algorithm a large number of times, which incurs huge computational expenses (in our experiments it took around 4 days of continuous runs for each diagram). Second, after all these runs the problem is already solved and the best search strategy is identified just post-factum.

In this study, we propose an idea of how to overcome this handicap and to bring the above reasoning close to the practice. Our hypothesis is that the research into the problem hardness could help us to make the choice of a suitable optimisation method with far less efforts. To support this hypothesis, we have analysed the L_c-cost diagrams for all our instances. They are not presented in this paper in order to avoid a cumbrousness but all of them are available in the journal's online supplement. The generated diagrams can be classified by their shapes into two

groups. Diagrams, which follow the pattern shown in Fig. 6, are characteristic to 6 out of 12 datasets: *Exam_1*, *Exam_3*, *Exam_5*, *Exam_7*, *Exam_8* and *Exam_11*. For the remaining 6 datasets: *Exam_2*, *Exam_4*, *Exam_6*, *Exam_9*, *Exam_10* and *Exam_12* the shapes are close to the one given in Fig. 7. When comparing these two groups with Table 2, we can observe quite a strong correlation between the shape of the diagram and the value of T/L_c. The problems with the larger values of T/L_c (harder ones) have more distinct dependence of cost on L_c, i.e. the shapes are close to Fig. 6, while the diagrams with the shape shown in Fig. 7 are more typical to the problems with the smaller values of T/L_c, which are presumably less hard ones. It seems that the revealed connection between the shape of the L_c-cost diagram and the T/L_c coefficient is stronger than the connection between the shape of the diagram and the size of a problem.

Although this study presents experiments with 12 instances and three SCHC heuristics only, the results have demonstrated that for at least these datasets we can already skip the awkward diagram building stage and identify the best search strategy based just on the value of T/L_c, which is obtainable by a single short-time run, i.e. if this value is relatively low, then a multi-start approach is preferred. Otherwise, if this value is relatively high, a single long run will be more effective. Of course, the justification of this hypothesis and its practical implementation require much more extensive study on a larger number of benchmark problems and with other SCHC heuristics. However, our preliminary tests with the travelling salesman and grid scheduling problems have revealed the same algorithmic behaviour. Therefore, we believe that this represents a very promising direction of a future research.

4.4 Cost drop diagrams of different variants of SCHC

The question about the choice of the best search strategy is not limited to the above example. The investigations in this field are especially relevant to SCHC, which supposes a large number of variants and extensions where it is necessary to make a choice between them. In this situation, it is important to estimate the difference in the search behaviour between different heuristics. Hence, the following series of experiments are designed to analyse search strategies employed by our three variants of SCHC. Similar to the first experiment we do that by plotting their cost drop diagrams. Moreover, taking into consideration the T/L_c coefficients we can now tune our SCHC heuristics in order to provide the same convergence time for each of the variants. The value of L_c can be calculated by dividing the required time by the T/L_c coefficient from Table 2. In our test, the three heuristics were run with *Exam_1* dataset for 100 s, hence the calculated values of L_c were: SCHC-all: 23800, SCHC-acp: 1695 and SCHC-imp: 1538. The resulting cost drop diagrams are presented in Fig. 8.

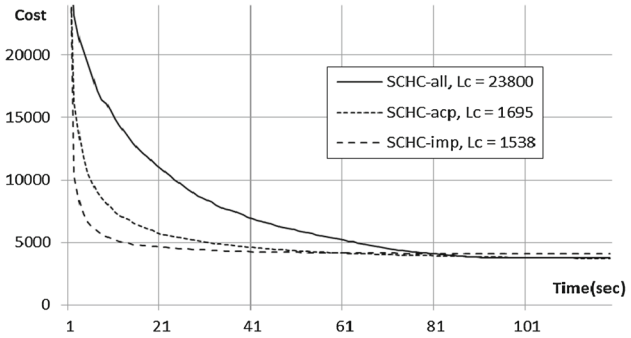

Fig. 8 The cost drop diagrams of different SCHC heuristics with *Exam_1* instance

The analysis of these diagrams gives an idea about the difference in search strategies between these heuristics. SCHC-imp jumps quickly into the region of low-cost solutions and then spends the most of the search time in this region while slowly improving the quality of result. The SCHC-acp heuristic does the same, but slightly smoother; it goes into the region of low-cost solutions more slowly and spends less time staying there. In contrast, the SCHC-all heuristic pays much more attention to exploring the high-cost solutions. It spends in that region about a half of the search time while the time spent for the final improvement is much shorter.

The convergence time in these diagrams is the same for each heuristic so the quality of a final result might be dependent on how the particular search strategy fits into the problem's landscape. Our experiments presented in the next section show that there is no general preference to any of the heuristics. Their performance is highly problem-dependent and different problems acquire different best performed strategies. However, together with the shapes of the diagrams some other internal properties of these variants could affect their performance on different instances. This issue warrants a further investigation.

5 A comparison of SCHC with other methods

The developed variants of SCHC were compared with SA, GD and LAHC algorithms. Firstly, we test the performance of these methods on original ITC2007 problems. Secondly, we test the reliability of these methods using an artificially created non-linear optimisation problem.

5.1 A performance test

In Sect. 4.3 we have demonstrated that the performance of SCHC can be represented in the form of time-cost diagrams depicted in Figs. 6 and 7. Such diagrams show that for some

problems there could be an evident dependence of the quality of results on the total search time. Moreover, the final results produced by SCHC even in the same CPU time are scattered within certain cost interval. Obviously a good comparison method should take into consideration such a behaviour of SCHC as well as the behaviour of the other methods.

To investigate that, we repeated the experiment described in Sect. 4.3 with the methods selected for the comparison, i.e. we run them many times (around 500) while randomly varying their time-related parameters: cooling factor in SA, decay rate in GDA and history length in LAHC. The random variation of the parameters was organised in such a way, that we got uniform time-cost diagrams similar to the ones presented in Figs. 6 and 7. An interested reader can found these diagrams for all benchmark datasets produced by SA together with the ones for SCHC in the journal online supplement. A visual comparison of these diagrams indicates that for each dataset the diagram shapes of SA and SCHC are quite similar, which implies the similar behaviour of these methods. The same is also relevant to GDA and LAHC. As an illustration, the time-cost diagrams produced by SA for *Exam_1* and *Exam_10* problems are presented in Figs. 9 and 10. When comparing them with Figs. 6 and 7, respectively, we can see that for the same instances the shapes of the diagrams are very similar even being produced by different methods.

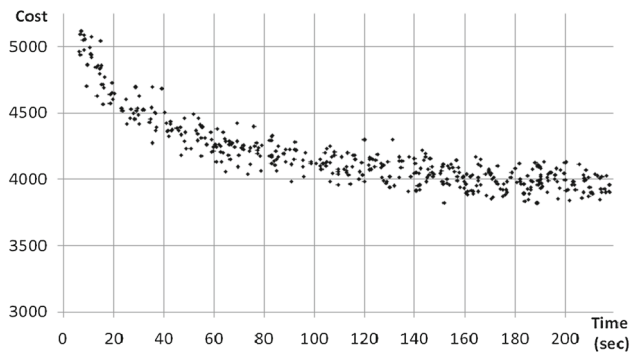

Fig. 9 The time-cost diagram produced by SA for *Exam_1* problem

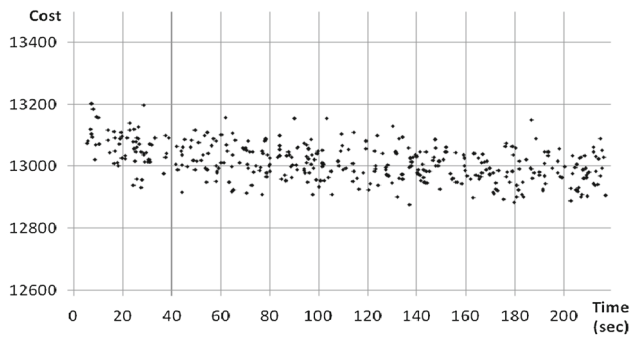

Fig. 10 The time-cost diagram produced by SA for *Exam_10* problem

When different algorithms show similar behaviour in respect of computing time, the differences between them should be evaluated to make conclusions about their performance. To rely only on the visual comparison of noisy curves is not satisfactory, so in order to get a more detailed information we apply a "cut-off" approach proposed in (Burke and Bykov 2012). To explain this method, we divided the diagrams in Figs. 9 and 10 by vertical gridlines into equal segments of 20 s length, which gives in total 10 segments. When observing these segments separately, the distribution of points within each of them gives an idea about the performance of the method being run within the corresponding time boundaries. For example, the points in segment (160,180) in Fig. 9 show that SA being run on *Exam_1* problem for 160–180 s is able to achieve final cost values approximately between 3850 and 4150, which is on average 4000. The cut-off approach employs a usual evaluation of average costs but takes into account the computing time. In this method, the average performance of an algorithm is represented by a set of average values calculated for all segments. This enables the sets produced by different methods to be compared in a table.

In our experiments, we have produced the cut-off sets for SA, GDA, LAHC and the three studied variants of SCHC for all our benchmark instances. To ensure the adequate performance of SA, we use general suggestions from the literature for its parameterization. We employ a geometric cooling schedule while the initial temperature is set up in such a way so that in the initial phase of the search the algorithm accepts 85 % of non-improving moves. In contrast, GDA, LAHC and SCHC do not require any special initialization procedure. Table 3 presents the resulting cut-offs for *Exam_1* dataset, where the best results over six heuristics are highlighted in bold.

This table demonstrates the clear superiority of two variants of SCHC (acp and imp) over the other methods on runs longer than 20 s. On shorter runs LAHC performs better, but both LAHC and SCHC-all slightly underperform on the longer runs. Nevertheless, SA performs much inferior to both LAHC and all variants of SCHC on the runs of any length. Finally, GDA has the worst performance than any other method.

We produced the same tables for all benchmark instances, which show quite problem-dependent performance of different methods. They are not included in the paper, but are available in the online supplement. To give an idea about this performance, in Table 4, a compilation of the collection of cut-offs for all datasets for the middle interval of 100–120 s is presented.

In the given running time, the three variants of SCHC have produced in total 8 over 12 best results, LAHC has got just 2 overall best results, but for 7 problems it performs better than at least one variant of SCHC. The general performance of SA

Table 3 The cut-offs for SA, GDA, LAHC and SCHC (all, acp, imp) for *Exam_1* dataset

CPU time (s)	SA	GDA	LAHC	SCHC-all	SCHC-acp	SCHC-imp
0–20	4879	5224	**4640**	4646	4656	4711
20–40	4513	4804	4305	4330	**4276**	4286
40–60	4342	4593	4190	4150	4147	**4129**
60–80	4228	4496	4084	4061	4066	**4053**
80–100	4175	4439	4029	4029	4007	**3985**
100–120	4109	4358	3986	3978	**3927**	3952
120–140	4087	4335	3939	3924	3906	**3900**
140–160	4041	4300	3912	3890	3872	**3863**
160–180	3982	4251	3899	3869	**3851**	3859
180–200	3996	4213	3864	3852	**3809**	3819

Table 4 The cut-offs for SA, LAHC and SCHC (all, acp, imp) for other ITC2007 datasets

Instance	SA	GDA	LAHC	SCHC-all	SCHC-acp	SCHC-imp
Exam_2	**392**	436	405	404	401	405
Exam_3	8177	8750	7967	7945	**7916**	7987
Exam_4	12982	13496	**12705**	12746	13297	13291
Exam_5	2598	2903	2591	2581	**2569**	2575
Exam_6	25445	25566	**25388**	25455	25447	25465
Exam_7	4015	4334	3877	3905	3859	**3855**
Exam_8	7119	7688	6901	**6899**	6951	6916
Exam_9	958	999	958	951	953	**944**
Exam_10	13008	13081	12996	12992	**12985**	12995
Exam_11	25525	26926	24782	**24535**	24825	24791
Exam_12	**5156**	5216	5189	5179	5190	5195

is considerably weaker, namely on 8 problems its results are worse than results of either LAHC or SCHC and only on two instances SA performs the best. Once again, GDA has the worst performance over six methods.

To further study the general tendencies in the performance of the compared methods on different datasets, we present another compilation of the cut-off results for all instances. For each instance we count the number of segments over the whole time interval where each method performs the best. Table 5 presents such numbers for all 12 datasets, where the largest numbers of the "winning" segments are highlighted by bold.

The analysis of this table confirms a distinctive good performance of SA on *Exam_2* and *Exam_12* problems. In both cases it wins in 9 over 10 time segments. For other 10 problems, different variants of SCHC perform the best across segments. For example, SCHC-all has a distinctive performance on *Exam_6*, *Exam_8* and *Exam_11* problems, SCHC-imp has a distinctive performance on *Exam_9*, while SCHC-acp and SCHC-imp both have a good performance on *Exam_1*, *Exam_7* and *Exam_10* problems. Finally, on *Exam_3* all three variants of SCHC have approximately the same performance.

In our analysis, of a particular interest is the comparison of the results given in Tables 4 and 5 with the values of T/L_c in Table 2, which seems to reflect the problem hardness. First, SCHC wins on all six relatively harder problems. Second, SA shows a good performance just on datasets that are to some extent uncommon. For example, *Exam_12* is exceptionally small problem; *Exam_2* problem has disproportionally low values of T/L_c (see Sect. 4.3). The oddity of *Exam_4* problem was also discussed previously. It could happen that the performance of SCHC with the single-room problem is more dependent on the shape of cost drop diagram (see Fig. 8) than with other ones.

5.2 A reliability test

In their study of the LAHC algorithm, Burke and Bykov (2012) concluded that the absence of a cooling schedule made LAHC more reliable than the cooling schedule-based local search methods. This was demonstrated by evaluating the performance of different algorithms on a specially designed artificial problem whose cost function is non-linearly rescaled. This approach is adopted for the evaluation of the SCHC algorithm. In this series of experiments, *Exam_1*

Table 5 The numbers of winning segments of SA, LAHC and SCHC (all, acp, imp) for all datasets

Instance	SA	GDA	LAHC	SCHC-all	SCHC-acp	SCHC-imp
Exam_1	–	–	1	–	4	**5**
Exam_2	**9**	–	–	–	1	–
Exam_3	–	–	–	**4**	3	3
Exam_4	1	–	3	**6**	–	–
Exam_5	–	–	–	1	**7**	3
Exam_6	–	–	2	**7**	2	–
Exam_7	–	–	1	–	**6**	4
Exam_8	–	–	4	**7**	–	–
Exam_9	–	–	–	1	1	**8**
Exam_10	–	–	–	1	**5**	4
Exam_11	–	–	1	**9**	–	–
Exam_12	**9**	–	1	–	–	–

dataset was used with transformation (2) applied to its cost function. Thus, in the new problem, the cost function C^{res} is represented as a cubical polynomial of the original cost C.

$$C^{\text{res}} = C^3 - 48000 \times C^2 + 770 \times 10^6 \times C \tag{2}$$

Expression (2) represents a monotonically increasing function because its first derivative is a quadratic polynomial with a positive first coefficient and a negative discriminant, and therefore, this derivative is always positive. Hence, all original local and global optima are preserved in the new problem, i.e. when solution A has a higher cost than solution B in the original problem, it holds true in the new problem also. The new and the original problems are the same from the point of view of dominance relations between solutions, while the rescaling differentiates only the distances between solutions (usually called as "delta costs"). Consequently, rescaling expressed by (2) affects the performance of algorithms which evaluate delta costs, such algorithms are SA or GDA. However, it has no effect on algorithms which employ the ranking of solutions such as HC or LAHC. With the same initial randomization, the search paths of these algorithms is the same for the original and the rescaled problems and they will achieve the same final results. The proposed SCHC is also based on the solution ranking and does not evaluate delta costs (see pseudocodes in Figs. 1, 2, 3), therefore a monotonic rescaling of the cost function should not affect the performance of the algorithm.

To verify empirically this proposition, we run the same experiment as in Sect. 5.1 on the rescaled problem. Apart from the new problem formulation, all other experimental conditions remain the same as explained. Only the initial temperature of SA was tuned again in order to comply with the literature suggestion that there is 85 % of non-improving moves at the beginning. The results of this test are shown in Table 6. To simplify an assessment of these results, we present in this table their non-rescaled values.

The results confirm that the rescaling given by (2) considerably deteriorates the performance of SA and to the less extent GDA but does not affect any studied variant of SCHC in the same way as LAHC. This example supports a contention that SCHC is more reliable than SA. In this experiment, we resort to a highly non-linear artificial problem specially designed for the purpose of enhancing the effect

Table 6 The cut-offs for SA, GDA, LAHC and SCHC (all, acp, imp) for *Exam_1* dataset with the rescaled cost function

CPU time (s)	SA	GDA	LAHC	SCHC-all	SCHC-acp	SCHC-imp
0–20	5487	5526	**4636**	4671	4657	4663
20–40	5227	5089	4307	4322	**4277**	4285
40–60	5091	4893	4152	4138	4116	**4096**
60–80	5013	4792	4073	4071	**4029**	4033
80–100	4977	4773	4003	4027	3958	**3943**
100–120	4929	4710	3953	3973	**3922**	3934
120–140	4907	4604	3921	3910	**3885**	3892
140–160	4892	4574	3891	3875	3862	**3858**
160–180	4852	4495	3854	3864	**3828**	3839
180–200	4845	4445	3828	3827	**3794**	3808

Table 7 The comparison of our best results with ITC2007 and post-competition ones

Instance	ITC2007 web best	Muller 2008	McCollum 2009	Gogos et al. 2010	SCHC-acp
Exam_1	4370	4356	4633	4128	**3647**
Exam_2	400	390	405	**380**	385
Exam_3	10049	9568	9064	7769	**7487**
Exam_4	18141	16591	15663	13103	**11779**
Exam_5	2988	2941	3402	2513	**2447**
Exam_6	26585	25775	25880	25330	**25210**
Exam_7	4213	4088	4037	**3537**	3563
Exam_8	7742	7565	7461	7087	**6614**
Exam_9	1030	–	1071	**913**	924
Exam_10	14778	–	14374	13053	**12931**
Exam_11	34129	–	29180	24369	**23784**
Exam_12	5264	–	5693	**5095**	5097

of the non-linearity on the search process. However, when the non-linearity of a problem is not so distinct, the deterioration of the SA performance might be less clear, although it is still present. If we assume that the presence of a high number of hard constraints, which exclude infeasible solutions from the search space, somehow provides a non-linear effect, then this could explain the underperformance of SA also on the original problems.

6 Comparison with published results

To complete the evaluation of SCHC performance, we compare its results with the actual best ITC2007 results and with the results presented in pre-competition (Muller 2008) and post-competition publications: McCollum et al. (2009) and Gogos et al. (2010). In this series of experiments, we respect the ITC2007 restrictions on the maximum run time and the number of independent runs. The maximum run time was calculated using the ITC2007 benchmarking application; for our experimental PC it was 204 s. Also, the best result was selected over 10 independent runs on each benchmark problem. Our best results together with the published ones are presented in Table 7 where the best results are highlighted in bold. All our best results are verified using the online validator provided by ITC2007 organisers.

In this comparison, we use SCHC-acp variant because in our previous experiments it has shown the strongest performance (see Table 4). To provide the convergence of the algorithm exactly in the given time, the value of L_c was calculated for each instance based on the values of T/L_c from Table 2. In such a way we employ some beforehand collected information about the algorithmic behaviour on benchmark instances and therefore we do not claim that this series of experiments mimics our participation in the competition. The most of the post-competition studies do not pursue that goal either which is in line with the discussion by McCollum et al. (2009) that adherence to the competition rules in any post-competition publication is rather artificial because many additional factors should be taken into account. The results of Gogos et al. (2010) can be considered as the best up to date; however, the authors indicated that they were produced "under no hardware or time limit", i.e. without following ITC2007 rules at all. Hence, in this comparison, we have an advantageous position over the actual ITC2007 results, the same position with Muller (2008) and McCollum et al. (2009) but the position of Gogos et al. (2010) is more advantageous than ours.

This table once again demonstrates the strong performance of our proposed method. For eight benchmark problems SCHC has achieved results better than the best previously published ones. Although for four remaining problems our results are inferior to Gogos et al. (2010), they are still very competitive. It is interesting to observe that *Exam_2* and *Exam_12* are among the instances on which the method by Gogos et al. (2010) performed better, which complies with the discussions provided in the previous sections.

7 Conclusions

In this study, we proposed a new local search algorithm: SCHC and investigated its behaviour. The exam timetabling problem was chosen as benchmark. Our experiments have revealed that SCHC shares a number of properties with LAHC:

- SCHC has a strong performance on the benchmark problems.
- SCHC operates with a single input parameter, counter limit L_c, which affects the search/convergence time.

- The search time is approximately proportional to L_c and the coefficient of proportionality is usually larger for the problems, which seemed to be harder.
- SCHC does not employ any type of cooling schedule.
- SCHC is more reliable than cooling schedule based methods (e.g. SA), i.e. it works well on specific problems where SA fails to produce good results.

However, SCHC has several additional distinct properties:

- The counting mechanism can be implemented in a variety of ways. So, for each particular problem we can find the most suitable variant of SCHC.
- The counting mechanism in SCHC is very flexible. During the search, the value of the counter limit can be easily adjusted at any iteration to respond to the obtained results. Moreover, the whole counting mechanism can be changed throughout the search. Therefore, SCHC can serve as a good platform for developing various self-adaptive methods.
- SCHC is a very simple and transparent method, easy to understand and implement. Hence, it has a high potential in the education area. By studying SCHC as a first meta-heuristic, the students could quicker get into the ABC's of search methodologies. For this purpose, we have included SCHC into our "Multi-Heuristic Solver" software application, which is available for download from http://www.yuribykov.com/MHsolver/.

The main emphasis of this paper is on the investigation into the behaviour of the proposed algorithm. Some observed dependencies in this behaviour motivated the hypothesis that in addition to the solving optimisation problems SCHC can be also used as a tool for heuristic measuring their hardness. We have proposed that the research into the hardness of different problems might help to identify an optimal search strategy for a particular dataset. The results of our experiments provide some empirical support for our ideas. However, the justification of these hypotheses requires further and much wider investigations with different problems and variants of SCHC. Apart from that, the research into the further properties of SCHC, its behaviour with different problems and the modifications of this method (especially the self-adaptive ones) as well, as its practical and educational applications is also seen as a quite interesting subject of a future work.

Acknowledgments The work described in this paper was carried out under a Grant (EP/F033214/1) awarded by the UK Engineering and Physical Sciences Research Council (EPSRC).

References

Abdullah, S., & Alzaqebah, M. (2014). An adaptive artificial bee colony and late acceptance hill-climbing algorithm for examination timetabling. *Journal of Scheduling*, *17*, 249–262.

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

Boese, K. D., Kahng, A. B., & Muddu, S. (1994). A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, *16*, 101–113.

Burke E. K., & Bykov, Y. (2008). A late acceptance strategy in hill-climbing for exam timetabling problems: *Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT2008)*, Montreal, Canada.

Burke E. K., & Bykov, Y. (2012). The Late Acceptance Hill Climbing heuristic. Technical report CSM-192, Computing Science and Mathematics, University of Stirling, UK.

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

Burke, E. K., McColumn, B., Meisels, A., Petrovic, S., & Qu, R. (2007). A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, *176*, 177–192.

Burke, E. K., & Petrovic, S. (2002). Recent research directions in automated timetabling. *European Journal of Operational Research, 140*, 266–280.

Bykov Y., & Petrovic, S. (2013). An initial study of a novel Step Counting Hill Climbing heuristic applied to timetabling problems: *Proceedings of 6th Multidisciplinary International Scheduling Conference (MISTA 2013)*, Gent, Belgium.

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

Di Gaspero, L., & Schaerf, A. (2001). Tabu search techniques for examination timetabling. *Practice and Theory of Automated Timetabling III, Lecture Notes in Computer Science*. Berlin: Springer.

Erben, W. (2001). A grouping genetic algorithm for graph colouring and exam timetabling. *Practice and Theory of Automated Timetabling III, Lecture Notes in Computer Science*. Berlin: Springer.

Goerler, A., Schulte, F., & Voss, S. (2013). An application of late acceptance hill climbing to the traveling purchaser problem. *Computational logistics, Lecture Notes in Computer Science*. Berlin: Springer.

Gogos, C., Goulas, G., Alefragis, P., Kolonias, V., & Housos, E. (2010). Distributed Scatter Search for the examination timetabling problem. *PATAT 2010 Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling*, Belfast, UK.

Jackson W., Özcan, E., & Drake, J. H. (2013). Late acceptance-based selection hyper-heuristics for cross-domain heuristic search. *The 13th UK Workshop on Computational Intelligence*, Guildford, UK.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., & Schevon, C. (1989). Optimization by simulated annealing: an experimental evaluation. Part II, graph partitioning. *Operations Research*, *37*(3), 865–892.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., & Schevon, C. (1991). Optimization by simulated annealing: an experimental evaluation; Part II, graph coloring and number partitioning. *Operations Research*, *39*(3), 378–406.

McCollum, B., McMullan, P. J., Parkes, A. J., Burke, E. K., & Abdullah, S. (2009). An extended great deluge approach to the examina-

tion timetabling problem: *Proceedings of the 4th Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA09)*, Dublin, Ireland, 424–434.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P. J., Lewis, R., Parkes, A. J., et al. (2010). Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, *22*, 120–130.

Merlot, L., Boland, N., Hughes, B., & Stuckey, P. (2003). A hybrid algorithm for the examination timetabling problem. Practice and Theory of Automated Timetabling IV. *Lecture Notes in Computer Science*, Berlin: Springer

Muller, T. (2008). ITC2007 solver description: a hybrid approach. *Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2008)*, Montreal, Canada.

Özcan, E., Bykov, Y., Birben, M., & Burke, E. K. (2009). Examination timetabling using late acceptance hyper-heuristics: *Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC'09)*, Trondheim, Norway.

Petrovic, S., & Bykov, Y. (2003). A multiobjective optimisation technique for exam timetabling based on trajectories. *Practice and Theory of Automated Timetabling IV, Lecture Notes in Computer Science*. Berlin: Springer.

Petrovic, S., Patel, V., & Young, Y. (2005). University timetabling with fuzzy constraints. *Practice and Theory of Automated Timetabling V, Lecture Notes in Computer Science*. Berlin: Springer.

Qu, R., Burke, E. K., McCollum, B., Merlot, L. T. G., & Lee, S. Y. (2009). A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, *12*, 55–89.

Schaerf, A. (1999). A survey of automated timetabling. *Artificial Intelligence Review*, *13*, 87–127.

Smith-Miles, K., & Lopes, L. (2012). Measuring instance difficulty for combinatorial optimization problems. *Computers and Operations Research, 39*, 875–889.

Swan, J., Drake, J., Ozcan, E., Goulding, J., & Woodward, J. (2013). A comparison of acceptance criteria for the Daily Car-Pooling problem. *Computer and Information Sciences*, *III*, 477–483.

Thompson, J. M., & Dowsland, K. A. (1996). Variants of simulated annealing for the examination timetabling problem. *Annals of Operations Research*, *63*, 105–128.

Tierney, K. (2013). Late acceptance hill climbing for the liner shipping fleet repositioning. *Proceedings of the 14th EU/ME Workshop*, 21–27.

Verstichel, J., & Vanden Berghe, G. (2009). A late acceptance algorithm for the lock scheduling problem. *Logistic Management*, *2009*(5), 457–478.

Vancroonenburg, W., & Wauters, T. (2013) Extending the late acceptance metaheuristic for multi-objective optimization. *Proceedings of the 6th Multidisciplinary International Scheduling conference: Theory & Applications (MISTA2013)*. Ghent, Belgium.

Yuan, B., Zhang, C., & Shao, X. (2015). A late acceptance hill-climbing algorithm for balancing two-sided assembly lines with multiple constraints. *Journal of Intelligent Manufacturing, 26*, 159–168.