European Journal of Operational Research 258 (2017) 70–78

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

The late acceptance Hill-Climbing heuristic

Edmund K. Burke, Yuri Bykov*

School of Electronic Engineering and Computer Science, Queen Mary University of London, Mile End Road, London, E1 4FZ, UK

ARTICLE INFO

Article history:
Received 16 September 2014
Accepted 8 July 2016
Available online 15 July 2016

Keywords:
Combinatorial optimization
Metaheuristics
Late acceptance hill climbing
Timetabling
Travelling salesman problem

ABSTRACT

This paper introduces a new and very simple search methodology called Late Acceptance Hill-Climbing (LAHC). It is a local search algorithm, which accepts non-improving moves when a candidate cost function is better than it was a number of iterations before. This number appears as a single algorithmic input parameter which determines the total processing time of the search procedure. The properties of this method are experimentally studied in this paper with a range of Travelling Salesman and Exam Timetabling benchmark problems. Also, we present a fair comparison of LAHC with well-known search techniques, which employ a cooling schedule: Simulated Annealing (SA), Threshold Accepting (TA) and the Great Deluge Algorithm (GDA). In addition, we discuss the success of the method in winning an international competition to automatically solve the Magic Square problem. Our experiments have shown that the LAHC approach is simple, easy to implement and yet is an effective search procedure. For most of the studied instances (especially for the large sized ones), its average performance is better than competitor methods. Moreover, the LAHC approach has an additional advantage (in contrast to the above cooling schedule based methods) in its scale independence. We present an example where the rescaling of a cost function in the Travelling Salesman Problem dramatically deteriorates the performance of three cooling schedule based techniques, but has absolutely no influence upon the performance of LAHC.

1. Introduction

A family of algorithms, often labeled under the term *local search*, represents a wide range of techniques, across a broad spectrum of problems. Generally, these algorithms have the following advantages: they are relatively simple in implementation, moderate in CPU time requirement and quite effective for large-scale problems. A typical local search procedure starts from a random initial solution, which is iteratively improved by accepting or rejecting candidate solutions. The simplest local search algorithm is the greedy Hill-Climbing (HC), which was one of the earliest search techniques (Appleby, Blake, & Newman, 1960). HC accepts only candidates with the same or better cost than the current one. This method usually produces relatively low quality results in a quick CPU time. The better results (in a longer CPU time) can be achieved using alternative techniques, which allow the limited acceptance of worsening moves. The common property of these methods is that their acceptance condition is regulated by a *control parameter* (such as *temperature*, *threshold* or *water level*), which is varied in the course of the search. The shape of this variation (schedule) has a significant impact on the overall performance of these techniques and this has been extensively investigated over the years.

One of the most well studied local search techniques is Simulated Annealing (SA) proposed by Kirkpatrick, Gelatt, and Vecchi (1983). It is a stochastic algorithm, which accepts worse candidates with probability $P=exp[(C-C^*)/T]$ (here and everywhere below we assume a minimization problem). In this expression, C and C^* are respectively the cost functions of a current and a candidate solution and T is the control parameter called "temperature" (its variation is called the *cooling schedule*). Several authors have proposed initial temperatures so that a certain percentage of worsening moves are accepted at the beginning. Different sources suggest different values for this percentage. Examples include between 40 percent and 90 percent (Johnson, Aragon, McGeoch, & Schevon, 1989), 75 percent (Thomson & Dowsland, 1996) and 95 percent (Cohn & Fielding, 1999). The final value of the temperature should be close to zero. One of the most popular cooling schedules (called "geometric cooling") is represented by the following expression: $T_i = T_{i-1}{}^*\beta$, i.e., the temperature at the ith iteration is equal to the previous temperature T_{i-1} multiplied by a user-defined *cooling factor* β ($0<\beta<1$). However, some authors have suggested the use of alternatives, such as the "quadratic cooling schedule" (Anderson, Vidal, & Iverson, 1993) or even increases in the temperature, such as adaptive cooling (Thompson & Dowsland, 1996) or reheating (Osman, 1993).

* Corresponding author.
E-mail addresses: e.burke@qmul.ac.uk (E.K. Burke), mail@yuribykov.com (Y. Bykov).

http://dx.doi.org/10.1016/j.ejor.2016.07.012
0377-2217/

In addition to Simulated Annealing, a number of other similar search techniques have been proposed. One that is particularly close to SA is the Threshold Accepting (TA) method, which is also known as "Deterministic Simulated Annealing" (Dueck & Scheurer, 1990). Here, the candidate solution is accepted if C^*- $C \leq T$ where T is a control parameter (called the "threshold"), which again is varied based on its schedule. Another deterministic variant (proposed by Dueck 1993) is the Great Deluge Algorithm (GDA). In contrast to TA, its control parameter B (called the "level") serves as an upper bound of the candidate cost function. Thus, the algorithm accepts worse candidates with the cost equal or less than the current value of the level, i.e., when $C^* \leq B$. In classical GDA, it was recommended that the initial level be equal to the initial cost function and that it should be lowered linearly during the search. However, other propositions have also appeared in the literature, such as: initialization with a higher level (Burke & Newall, 2003), non-linear level lowering (Obit, Landa-Silva, Ouelhadj, & Sevaux, 2009), reheating-like mechanisms (McMullan 2007) or modified acceptance condition (Burke & Bykov, 2016). There are several other methods based on this pattern such as "Old Bachelor Acceptance" (Hu, Kahng, & Tsao, 1995) or "Weight Annealing" (Nino & Schneider, 2005).

In all of these algorithms, the variation of the control parameter is regulated by arbitrarily defined schedules: in SA it is called the "cooling schedule". In our study, we also generalize this term for TA and GDA because their schedules are undetermined.

The cooling schedule has its strong and weak points. A strong point is that it enables an explicit way of processing time management. This feature is important in many situations where increasing the search time can lead to better final results. However, in order to effectively use the long searches in practice, their processing time should be pre-defined (see Burke, Bykov, Newall, & Petrovic, 2004). A weak point of the cooling schedule is that its optimal form is problem/instance-dependent and generally indefinite. That is the reason for the existence of a number of empirical recommendations regarding cooling schedules, which are more or less effective for a range of studied problems. However, there is no guarantee that such a proposition will work for a new problem. In this paper, we present an example, where just a rescaling of the cost function dramatically deteriorates the performance of evaluated cooling-schedule based methods.

In (Burke & Bykov, 2008), we proposed the initial idea of a non-arbitrary control parameter, the value of which is obtained from the previous history of the search. We have called this method the Late Acceptance strategy. This also builds upon work presented at the CEC'09 conference (Ozcan, Birben, Bykov, & Burke, 2009). By drawing on our initial presentation, a number of researchers from different institutions have taken up the idea of late acceptance and started separate studies on this method applied to different problems such as: lock scheduling (Verstichel & Vanden Berghe, 2009), liner shipping fleet repositioning (Tierney, 2013) and balancing two-sided assembly lines (Yuan, Zhang, & Shao, 2015). Some of the researchers went beyond the case studies and have proposed their own modifications of our approach, such as: Late Acceptance Randomized Descent algorithm (Abuhamdah 2010) or Multiobjective Late Acceptance algorithm (Vancroonenburg & Wauters, 2013). In addition, this method was hybridized with other techniques (Alzaqebah & Abdullah, 2014) and investigated in respect of the recently developed hyper-heuristic approach (Jackson, Özcan, & Drake, 2013).

Furthermore, in December 2011, a late acceptance based algorithm won the 1st place prize in the International Optimisation Competition. In July 2012, two research teams ranked 4th and 7th places in the ROADEF/EURO Challenge 2012 whilst employing the idea of late acceptance in their entry algorithms. In June 2014 a research group (called CODeS) from KU Leuven, Belgium won the 1st place in the VeRoLog 2014 International Competition (http://verolog.deis.unibo.it) using the Late Acceptance method. In addition, our method has been embedded into at least two real-world software systems: the 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).

In this paper, we adapt our algorithm, investigate its properties and compare its performance with related methods (SA, TA and GDA). In order to provide prompt comprehensive information about LAHC to other researchers, we made an earlier version of this paper available as an institutional technical report (Burke & Bykov, 2012). Subsequent communication with a range of readers (especially with those who decided to implement our technique) revealed improvements that were required in that report. Therefore, this paper represents a significantly altered version of that institutional report: the methodological description is rewritten, unnecessary reasoning is removed, the experimental layout is changed and all experiments are completely re-calculated.

The description of our technique is presented in the next section. In Section 3, we present an experimental study of the properties of the proposed method. Section 4 contains a comparison of the new algorithm with existing techniques. In Section 5 we discuss the practical effectiveness of the proposed heuristic as evidenced by its success in the International Optimisation Competition. A summary, conclusions and further perspectives are presented in Section 6.

2. Late Acceptance Hill Climbing

The initial idea of the *late acceptance* heuristic is rather simple: the control parameter in the acceptance condition is taken from the history of the search. This heuristic could be viewed as an extension of HC with just one difference: in greedy Hill Climbing a candidate solution is compared with the immediate current one, but in the Late Acceptance Hill Climbing (LAHC) a *candidate is compared with that solution, which was the current several iterations before*. Except for the new acceptance condition, the other details of LAHC are the same as in other local search methods (such as HC, SA, TA or GDA), i.e., the algorithm is started from a random initial solution and iteratively accepts or rejects candidates until a stopping condition occurs.

Basically, LAHC can employ its acceptance rule while maintaining a list of a fixed length L_h (history length) of previous values of the current cost function. The candidate cost is compared with the last element of the list and if better, then the candidate is accepted. After the acceptance procedure, the list is updated i.e., the new current cost is added to the beginning of the list and the last element is removed from the end of the list. Note that the added current cost is equal to the candidate cost in the case of accepting only, but in the case of rejecting it is equal to the previous value.

The length L_h appears as a single algorithmic (and user-specified) parameter for this technique. The actual initialization of the list can be done automatically, even there is no previous history at the first iteration. We see here two possible variants: we can either produce L_h always accepted moves and record the values of objective function or just assign all elements of the list to be equal to the initial cost. Preliminary tests did not show any visual difference in the performance between these variants, but we consider the second variant to be preferable as it saves CPU time. It should be noted that when the initial list contains all values which are much higher than the initial cost, the second variant just turns into the first one. However, we do not recommend the initialization of the list by values which are much less than the initial cost, as in that case LAHC turns into HC at the beginning.