Expert Systems With Applications 64 (2016) 412–422

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Reinforcement learning based local search for grouping problems: A case study on graph coloring

Yangming Zhou[a], Jin-Kao Hao[a,b,*], Béatrice Duval[a]

[a] LERIA, Université d'Angers, 2 Bd Lavoisier, 49045 Angers, France
[b] Institut Universitaire de France, Paris, France

ARTICLE INFO

Article history:
Received 6 February 2016
Revised 25 June 2016
Accepted 10 July 2016
Available online 4 August 2016

Keywords:
Grouping problems
Reinforcement learning
Heuristics
Learning-based optimization

ABSTRACT

Grouping problems aim to partition a set of items into multiple mutually disjoint subsets according to some specific criterion and constraints. Grouping problems cover a large class of computational problems including clustering and classification that frequently appear in expert and intelligent systems as well as many real applications. This paper focuses on developing a general-purpose solution approach for grouping problems, i.e., reinforcement learning based local search (RLS), which combines reinforcement learning techniques with local search. This paper makes the following contributions: we show that (1) reinforcement learning can help obtain useful information from discovered local optimum solutions; (2) the learned information can be advantageously used to guide the search algorithm towards promising regions. To the best of our knowledge, this is the first attempt to propose a formal model that combines reinforcement learning and local search for solving grouping problems. The proposed approach is verified on a well-known representative grouping problem (graph coloring). The generality of the approach makes it applicable to other grouping problems.

1. Introduction

Grouping problems aim to partition a set of items into a collection of mutually disjoint subsets according to some specific criterion and constraints. Grouping problems naturally arise in numerous domains. Well-known grouping problems include, for instance, the graph coloring problem (GCP) (Elhag & Özcan, 2015; Galinier & Hao, 1999; Garey & Johnson, 1979; Lewis, 2009) and its variants like selective graph coloring (Demange, Monnot, Pop, & Reis, 2012; 2014), partition graph coloring (Fidanova & Pop, 2016; Pop, Hu, & Raidl, 2013), sum coloring (Jin, Hamiez, & Hao, 2016) and bandwidth coloring (Lai, Hao, Lü, & Glover, 2016), timetabling (Elhag & Özcan, 2015; Lewis & Paechter, 2007), bin packing (Falkenauer, 1998; Quiroz-Castellanos et al., 2015), scheduling (Kashan, Kashan, & Karimiyan, 2013) and clustering (Agustn-Blas et al., 2012). Formally, given a set V of n distinct items, the task of a grouping problem is to partition the items of set V into k different groups g_i $(i = 1, \ldots, k)$ (k can be fixed or variable), such that $\cup_{i=1}^{k} g_i = V$ and $g_i \cap g_j = \emptyset, i \neq j$ while taking into account some specific constraints and optimization objective. For instance, the graph coloring problem is to partition the vertices of a given graph into a minimum number of k color classes such that adjacent vertices must be put into different color classes.

According to whether the number of groups k is fixed in advance, grouping problems can be divided into constant grouping problems or variable grouping problems (Kashan et al., 2013). In some contexts, the number of groups k is a fixed value of the problem, such as identical or non-identical parallel-machines scheduling problem, while in other settings, k is variable and the goal is to find a feasible grouping with a minimum number of groups, such as bin packing and graph coloring. Grouping problems can also be classified according to the types of the groups. A grouping problem with identical groups means that all groups have similar characteristics, thus naming of the groups is irrelevant. Aforementioned examples such as identical parallel-machines scheduling, bin-packing and graph coloring belong to this category. Another category of grouping problems have non-identical groups where the groups are of different characteristics. Hence, swapping items between two groups will result in a new grouping, such as the non-identical parallel-machines scheduling problem.

Many grouping problems, including the examples mentioned above are NP-hard, thus computationally challenging. Consequently, exponential times are expected for any algorithm to solve such a problem exactly. On the other hand, heuristic and meta-heuristic methods are often employed to find satisfactory sub-optimal solutions in acceptable computing time, but without

* Corresponding author. Fax: +33241735073.
E-mail addresses: zhou.yangming@yahoo.com (Y. Zhou), hao@info.univ-angers.fr (J.-K. Hao), bd@info.univ-angers.fr (B. Duval).

http://dx.doi.org/10.1016/j.eswa.2016.07.047
0957-4174/

ensuring the optimality of the attained solutions. A number of heuristic approaches for grouping problems, in particular based on genetic algorithms, have been proposed in the literature with varying degrees of success (Falkenauer, 1998; Galinier & Hao, 1999; Quiroz-Castellanos et al., 2015). These approaches are rather complex since they are population-based and often hybridized with other search methods like local optimization.

In this work, we are interested in investigating a general-purpose local search methodology for grouping problems which employs machine learning techniques to process information collected from the search process with the purpose of improving the performance of heuristic algorithms. Indeed, previous work has demonstrated that machine learning can contribute to improve optimization methods (Baluja et al., 2000; Battiti & Brunato, 2014; Hafiz & Abdennour, 2016). The studies in these areas have pursued different objectives, illustrated as follows.

- Algorithm selection and analysis. For instance, Hutter et al. used machine learning techniques such as random forests and approximate Gaussian process to model algorithm's runtime as a function of problem-specific instance features. This model can predict algorithm runtime for the propositional satisfiability problem, traveling salesperson problem and mixed integer programming problem (Hutter, Xu, Hoos, & Leyton-Brown, 2014).
- Learning generative models of solutions. For example, Ceberio, Mendiburu, and Lozano (2013) introduced the Plackett–Luce probability model to the framework of estimation of distribution algorithms and applied it to solve the linear order problem and the flow-shop scheduling problem.
- Learning evaluation functions. For instance, Boyan and Moore (2001) proposed the STAGE algorithm to learn an evaluation function which predicts the outcome of a local search algorithm as a function of state features along its search trajectories. The learned evaluation function is used to bias future search trajectories towards better solutions.
- Understanding the search space. For example, Porumbel, Hao, and Kuntz (2010a) used multidimensional scaling techniques to explore the spatial distribution of the local optimal solutions visited by tabu search, thus improving local search algorithms for the graph coloring problem. For the same problem, Hamiez and Hao (1993) used the results of an analysis of legal k-colorings to help finding solutions with fewer colors.

In this paper, we present the reinforcement learning based local search (RLS) approach for grouping problems, which combines reinforcement learning techniques with a descent-based local search procedure (Section 3). Our proposed RLS approach belongs to the above-mentioned category of learning generative models of solutions. For a grouping problem with its k groups, we associate to an item a probability vector with respect to each possible group and determine the group of the item according to the probability vector (Sections 3.1 and 3.2). Once all items are assigned to their groups, a grouping solution is generated. Then, the descent-based local search procedure is invoked to improve this solution until a local optimum is attained (Section 3.3). At this point, the probability vector of each item is updated by comparing the item's groups in the starting solution and in the attained local optimum solution (Section 3.4). If an item does not change its group, then we reward the selected group of the item, otherwise we penalize the original group and compensate the new group (i.e., expected group). There are two key issues that need to be considered, i.e., how do we select a suitable group for each item according to the probability vector, and how do we smooth the probabilities to avoid potential search traps. To handle these issues, we design two strategies: a hybrid group selection strategy that uses a noise probability to switch between random selection and greedy selection (Section 3.2); and a probability smoothing mechanism to forget old decisions (Section 3.5).

To evaluate the viability of the proposed RLS method, we use the well-known graph coloring problem as a case study (Section 4). GCP is one representative grouping problem which has been object of intensive studies in the past decades (Section 4.1). GCP and its variants like bandwidth coloring have numerous applications including those arising in expert and intelligent decision systems including school timetabling (Ahmed, Özcan, & Kheiri, 2013), frequency assignment in mobile networks (Lai & Hao, 2015) and structural analysis of complex networks (Xin, Xie, & Yang, 2013). Popular problems like Sudoku and geographical maps of countries are two other application examples of GCP. For our experimental assessment of the proposed method, we test our approach on the popular DIMACS and COLOR02 benchmarks which cover both randomly generated graphs and graphs from real applications (Section 4.2). The experimental results demonstrate that the proposed approach, despite its simplicity, achieves competitive performances on most tested instances compared to many existing algorithms (Section 4.4). With an analysis of three important issues of RLS (Section 4.3), we show the effectiveness of combining reinforcement learning and descent-based local search. We also assess the contribution of the probability smoothing technique to the performance of RLS. We draw useful conclusions based on the computational outcomes (Section 5).

2. Reinforcement learning and heuristic search

In this section, we briefly introduce the principles of reinforcement learning (RL) and provide a review of some representative examples of using reinforcement learning to solve combinatorial optimization problems.

2.1. Reinforcement learning

Reinforcement learning is a learning pattern, which aims to learn optimal actions from a finite set of available actions through continuously interacting with an unknown environment. In contrast to supervised learning techniques, reinforcement learning does not need an experienced agent to show the correct way, but adjusts its future actions based on the obtained feedback signal from the environment (Gosavi, 2009).

There are three key elements in a RL agent, i.e., states, actions and rewards. At each instant a RL agent observes the current state, and takes an action from the set of its available actions for the current state. Once an action is performed, the RL agent changes to a new state, based on transition probabilities. Correspondingly, a feedback signal is returned to the RL agent to inform it about the quality of its performed action.

2.2. Reinforcement learning for heuristic search

There are a number of studies in the literature where reinforcement learning techniques are put at the service of heuristic algorithms for solving combinatorial problems. Reinforcement learning techniques in these studies have been explored at three different levels.

Heuristic level where RL is directly used as a heuristic to solve optimization problems. In this case, RL techniques are used to learn and directly assign values to the variables. For example, Miagkikh and Punch (1999) proposed to solve combinatorial optimization problems based on a population of RL agents. Pairs of variable and value are considered as the RL states, and the branching strategies as the actions. Each RL agent is assigned a specific area of the search space where it has to learn and find good local solutions. Another example is presented in (Torkestani & Mey-