Contents lists available at ScienceDirect

Computers & Operations Research

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

Multi-neighborhood local search optimization for machine reassignment problem

Zhuo Wang, Zhipeng Lü, Tao Ye[*]

School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

ARTICLE INFO

Available online 31 October 2015

Keywords:
Machine reassignment
Google ROADEF/EURO Challenge
Local search
Neighborhood structure
Partition search

ABSTRACT

As the topic of the Google ROADEF/EURO Challenge 2012, machine reassignment problem (denoted as MRP) is an important optimization problem in load balance of cloud computing. Given a set of machines and a set of processes running on machines, the MRP aims at finding a best process-machine reassignment to improve the usage of machines while satisfying various hard constraints. In this paper, we present a metaheuristic algorithm based on multi-neighborhood local search (denoted as MNLS) for solving the MRP. Our MNLS algorithm consists of three primary and one auxiliary neighborhood structures, an efficient neighborhood partition search mechanism with respect to the three primary neighborhoods and a dynamic perturbation operator. Computational results tested on 30 benchmark instances of the ROADEF/EURO Challenge 2012 and comparisons with the results in the challenge and the literature demonstrate the efficacy of the proposed MNLS algorithm in terms of both effectiveness and efficiency. Furthermore, several key components of our MNLS algorithm are analyzed to gain an insight into it.

1. Introduction

Cloud computing [1] has emerged as a new technology to support massive computing and storage task. A number of cloud computing services are currently being offered, such as Google Docs and Amazon Elastic Computing Cloud. A cloud computing platform is maintained by a data center which consists of a large pool of computing resources and storage devices shared by end users. As the services are expanding quickly, the issue of resource consumption of data centers is becoming increasingly important. Therefore, allocating computing resources and user requests in a cost-efficient way to improve the usage of servers plays a key role in cloud computing [2].

Aiming at finding an optimization solution to improve the usage of a set of machines, a challenging problem denoted by machine reassignment problem (MRP) has been proposed as the subject of the ROADEF/EURO Challenge 2012 by Google [3]. The MRP consists of reassigning processes among servers to improve the usage of machine resources while satisfying various constraints. The constraints which cannot be violated include capacity constraints, conflict constraints, spread constraints, dependency constraints and transient usage constraints. The objective function modeling the efficiency of the reassignment incorporates load cost, balance cost and three kinds of migration costs of processes.

For the MRP, several state-of-the-art algorithms have been proposed in the literature. Mehta et al. [4] introduced a constraint programming (CP) formulation of the problem, and proposed a CP-based large neighborhood search to address it. Malitsky et al. [5] further studied the impact of the parameters used in the proposed CP-based large neighborhood search. Gavranović et al. [6] proposed a variable neighborhood search which consists of four kinds of neighborhood structures. Brandt et al. [7] proposed a large neighborhood search algorithm that uses a constraint programming to find improving solutions. Masson et al. [8] put forward a multi-start iterated local search which explores two kinds of neighborhoods. Portal [9] proposed a randomized local search based on simulated annealing (SA) [10] for solving the MRP. Jaśkowski et al. [11] proposed a hybrid metaheuristic approach which consists of a fast greedy hill climber and a large neighborhood search. Portal et al. [12] put forward a simulated annealing algorithm using two neighborhoods to address the MRP. Lopes et al. [13] introduced a linear integer programming (IP) formulation for the MRP, and proposed an iterated local search algorithm for solving this problem.

The MRP belongs to the family of the assignment problem [14–16]. In particular, there is a similar theoretical assignment problem of MRP in the literature which is called generalized assignment problem (GAP) [17,18]. The GAP deals with a minimum cost

[*] Corresponding author.
E-mail addresses: 575880296@qq.com (Z. Wang), zhipeng.lv@hust.edu.cn (Z. Lü), yeetao@gmail.com (T. Ye).

http://dx.doi.org/10.1016/j.cor.2015.10.015
0305-0548/© 2015 Elsevier Ltd. All rights reserved.

assignment of a number of jobs to a set of agents such that every job is assigned to exactly one agent and the resource constraint for each agent is satisfied. It has many practical applications, such as vehicle routing [19–21] and facility location [22–24], and has been extensively studied in the literature. The approaches for GAP can be divided into two main categories: metaheuristic algorithms and exact algorithms.

The metaheuristic algorithms for solving the GAP include: a variable depth search algorithm by Amini and Racer [25]; a tabu search (TS) [26] and simulated annealing by Osman [27]; a genetic algorithm [28] by Chu and Beasley [29]; variable depth search algorithms by Yagiura et al. [30,31]; a guided genetic algorithm by Lau and Tsang [32]; an adaptive search heuristic based on greedy randomized adaptive search procedure (GRASP) [33] and Max-Min ant system (MMAS) [34] by Lourenço and Serra [35]; an ejection chain approach by Yagiura et al. [36]; a path relinking approach with ejection chains by Yagiura et al. [37]; differential evolution algorithms by Tasgetiren et al. [38]; a bees algorithm with an ejection chain neighborhood by Özbakir et al. [39] and so on.

Exact algorithms for GAP are mainly based on branch and bound framework, e.g., a branch and bound algorithm by solving a series of binary knapsack problems to determine the bounds by Ross and Soland [40], a new algorithm employing both column generation and branch and bound by Savelsbergh [41], a breadth-first branch and bound algorithm by Haddadi and Ouzia [42]. Woodcock and Wilson [43] proposed a hybrid algorithm combining both branch and bound and tabu search for solving the GAP.

In this paper, we propose a multi-neighborhood local search based metaheuristic algorithm, denoted as MNLS, for solving the machine reassignment problem. Our algorithm integrates several different neighborhood structures, two of which are usually employed in the literature for solving the GAP and the other two are tailored for the MRP. In addition, a neighborhood partition search mechanism and a dynamic perturbation operator are introduced to enhance the search efficiency of the MNLS.

The remaining part of the paper is organized as follows. Section 2 presents the problem description and a mathematical formulation of the MRP. In Section 3, we give the main framework and specific components of our MNLS algorithm in details. In Section 4, several key components of our algorithm are investigated. In Section 5, we present computational results of our MNLS algorithm and extensive comparisons with the results in both the challenge and the literature. The paper is concluded in Section 6.

2. Machine reassignment problem

2.1. Problem description

The machine reassignment problem considered in this paper consists of reassigning processes to machines in accordance with a given set of constraints. A machine has several kinds of resources such as CPU and RAM, and runs processes which consume these resources. Initially, each process is allocated to a machine. In order to improve the usage of machines, processes can be moved from one machine to another. Two types of constraints are defined: those which must be strictly satisfied under any circumstances (hard constraints) and those which are not necessarily satisfied but whose violations should be desirably minimized (soft constraints). Feasible moves are limited by hard constraints and have costs composed of violations of soft constraints. A reassignment that satisfies all hard constraints is called a feasible solution. The objective of this problem is to find a feasible solution which minimizes the total weighted violations of soft constraints.

Let $M=\{m_1, m_2, \dots, m_k\}$ be the set of machines, and $P=\{p_1, p_2, \dots, p_w\}$ the set of processes. A solution of this problem can be represented by a vector Map of length w, where $Map(p)$ corresponds to the machine assigned to process p. Additionally, let Map_0 be the given initial solution, so $Map_0(p)$ is the original machine assigned to process p correspondingly.

A number of constant and variable symbols are presented in Table 1. Note that *bool* is the truth indicator function which takes value of 1 if the given proposition is true and 0 otherwise. With these notations, we can describe the problem in a formal way.

The five hard constrains are:

- H_1. Capacity constraints. The usage of a machine for every resource cannot exceed its corresponding capacity.

  $$\forall m \in M, r \in R, U(m,r) \le C(m,r)$$

- H_2. Conflict constraints. Processes of the same service must run on distinct machines.

  $$\forall s \in S, (p_i, p_j) \in s^2, p_i \ne p_j \Rightarrow Map(p_i) \ne Map(p_j)$$

- H_3. Spread constraints. The number of locations where processes of the same service run cannot be less than a given minimum limit.

  $$\forall s \in S, \sum_{l \in L} \min(1, |\{p \in s | Map(p) \in l\}|) \ge S\_\min(s)$$

- H_4. Dependency constraints. If a service s^a depends on a service s^b, i.e., $(s^a, s^b) \in SD$, then each process of s^a should run in the neighborhood of a s^b process.

  $$\forall (s^a, s^b) \in SD, \forall p^a \in s^a, \exists p^b \in s^b \Rightarrow NE(Map(p^a)) = NE(Map(p^b))$$

- H_5. Transient usage constraints. When a process is moved from one machine to another, some resources are consumed twice. For example, disk space is occupied on both machines when a process is removed from one machine to another. Let $TR \subseteq R$ be the subset of resources which need transient usage, i.e., the processes using resources in TR require capacity on both the original machine and the current machine. The transient usage constraints are:

  $$\forall m \in M, \mathrm{tr} \in TR, U(m,\mathrm{tr}) + TU(m,\mathrm{tr}) \le C(m,\mathrm{tr})$$

  The five soft constraints are:

- S_1. Load cost. $SC(m,r)$ is the safety capacity of a resource $r \in R$ on a machine $m \in M$. The load cost is defined per resource and corresponds to the used capacity above the safety capacity:

  $$f_1(r) = \sum_{m \in M} \max(0, U(m,r) - SC(m,r))$$

- S_2. Balance cost. As having available CPU resource without having available RAM resource is useless for future assignments, one objective of this problem is to balance available resources. The idea is to achieve a given target on the available ratio of two different resources [3]. $B \subseteq R^2 \times \mathrm{N}$ is the set of triples. For a given triple $b = (r_1^b, r_2^b, target^b) \in B$, the balance cost is:

  $$f_2(b) = \sum_{m \in M} \max(0, target^b \cdot A(m, r_1^b) - A(m, r_2^b))$$

- S_3. Process move cost. Some processes are painful to move. To model this soft constraint a process move cost is defined: