European Journal of Operational Research 263 (2017) 72–81

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Exploiting variable associations to configure efficient local search algorithms in large-scale binary integer programs☆

Shunji Umetani

Graduate School of Information Science and Technology, Osaka University, 1–5 Yamadaoka, Suita, Osaka 565-0871, Japan

ARTICLE INFO

Article history:
Received 28 April 2016
Accepted 11 May 2017
Available online 31 May 2017

Keywords:
Combinatorial optimization
Heuristics
Set covering problem
Set partitioning problem
Local search

ABSTRACT

We present a data mining approach for reducing the search space of local search algorithms in a class of binary integer programs including the set covering and partitioning problems. The quality of locally optimal solutions typically improves if a larger neighborhood is used, while the computation time of searching the neighborhood increases exponentially. To overcome this, we extract variable associations from the instance to be solved in order to identify promising pairs of flipping variables in the neighborhood search. Based on this, we develop a 4-flip neighborhood local search algorithm that incorporates an efficient incremental evaluation of solutions and an adaptive control of penalty weights. Computational results show that the proposed method improves the performance of the local search algorithm for large-scale set covering and partitioning problems.

1. Introduction

The Set Covering Problem (SCP) and Set Partitioning Problem (SPP) are representative combinatorial optimization problems that have many real-world applications, such as crew scheduling (Barnhart, Johnson, Nemhauser, Savelsbergh, & Vance, 1998; Hoffman & Padberg, 1993; Mingozzi, Boschetti, Ricciardelli, & Bianco, 1999), vehicle routing (Agarwal, Mathur, & Salkin, 1989; Baldacci, Christofides, & Mingozzi, 2008; Bramel & Simchi-Levi, 1997; Hashimoto et al., 2009), facility location (Boros et al., 2005; Farahani, Asgari, Heidari, Hosseininia, & Goh, 2012) and logical analysis of data (Boros et al., 2000; Hammer & Bonates, 2006). Real-world applications of SCP and SPP are comprehensively reviewed in Ceria, Nobili, and Sassano (1997) and Balas and Padberg (1976), respectively.

Given a ground set of m elements $i \in M = \{1, \ldots, m\}$, n subsets $S_j \subseteq M$ ($|S_j| \geq 1$), and their costs $c_j \in \mathbb{R}$ ($\mathbb{R}$ is the set of real values) for $j \in N = \{1, \ldots, n\}$, we say that $X \subseteq N$ is a cover of M if $\bigcup_{j \in X} S_j = M$ holds. We also say that $X \subseteq N$ is a partition of M if $\bigcup_{j \in X} S_j = M$ and $S_{j_1} \cap S_{j_2} = \emptyset$ for all $j_1, j_2 \in X$ hold. The goals of SCP and SPP are to find a minimum cost cover and partition X of M, respectively. In this paper, we consider the following class of Binary Integer Programs (BIPs) including SCP and SPP:

$$\begin{aligned} \text{minimize} \quad & \sum_{j \in N} c_j x_j \\ \text{subject to} \quad & \sum_{j \in N} a_{ij} x_j \leq b_i, \quad i \in M_L, \\ & \sum_{j \in N} a_{ij} x_j \geq b_i, \quad i \in M_G, \\ & \sum_{j \in N} a_{ij} x_j = b_i, \quad i \in M_E, \\ & x_j \in \{0, 1\}, \quad j \in N, \end{aligned} \tag{1}$$

where $a_{ij} \in \{0, 1\}$ and $b_i \in \mathbb{Z}_+$ ($\mathbb{Z}_+$ is the set of nonnegative integer values). We note that $a_{ij} = 1$ if $i \in S_j$ holds and $a_{ij} = 0$ otherwise, and $x_j = 1$ if $j \in X$ holds and $x_j = 0$ otherwise. That is, a column vector $\boldsymbol{a}_j = (a_{1j}, \ldots, a_{mj})^\top$ of the matrix (a_{ij}) represents the corresponding subset $S_j = \{i \in M \mid a_{ij} = 1\}$, and the vector $\boldsymbol{x}$ also represents the corresponding cover (or partition) $X = \{j \in N \mid x_j = 1\}$. For notational convenience, we denote $M = M_L \cup M_G \cup M_E$. For each $i \in M$, let $N_i = \{j \in N \mid a_{ij} = 1\}$ be the index set of subsets S_j that contain the elements i, and let $s_i(\boldsymbol{x}) = \sum_{j \in N} a_{ij} x_j$ be the left-hand side of the ith constraint.

Continuous development of mathematical programming has much improved the performance of exact and heuristic algorithms and this has been accompanied by advances in computing machinery. Many efficient exact and heuristic algorithms for large-scale SCP and SPP instances have been developed

☆ A preliminary version of this paper was presented in Umetani (2015).

E-mail address: umetani@ist.osaka-u.ac.jp

http://dx.doi.org/10.1016/j.ejor.2017.05.025

(Atamtürk, Nemhauser, & Savelsbergh, 1995; Barahona & Anbil, 2000; Bastert, Hummel, & de Vries, 2010; Borndörfer, 1998; Boschetti, Mingozzi, & Ricciardelli, 2008; Caprara, Fischetti, & Toth, 1999; Caprara, Toth, & Fischetti, 2000; Caserta, 2007; Ceria, Nobili, & Sassano, 1998; Linderoth, Lee, & Savelbergh, 2001; Umetani & Yagiura, 2007; Wedelin, 1995; Yagiura, Kishida, & Ibaraki, 2006), many of which are based on a variant of the *column generation method* called the *pricing method* that reduces the number of variables to be considered in the search by using Linear Programming (LP) and/or Lagrangian relaxation. However, many large-scale SCP and SPP instances still remain unsolved because there is little hope of closing the large gap between the lower and upper bounds of the optimal values. In particular, the equality constraints of SPP often make the pricing method less effective because they often prevent solutions from containing highly evaluated variables together. In this paper, we consider an alternative approach for extracting useful features from the instance to be solved with the aim of reducing the search space of local search algorithms for large-scale SCP and SPP instances.

In the design of local search algorithms for combinatorial optimization problems, the quality of locally optimal solutions typically improves if a larger neighborhood is used. However, the computation time of searching the neighborhood also increases exponentially. To overcome this, extensive research has investigated ways to efficiently implement neighborhood search, which can be broadly classified into three types: (i) reducing the number of candidates in the neighborhood (Pesant & Gendreau, 1999; Shaw, Backer, & Furnon, 2002; Yagiura & Ibaraki, 1999; Yagiura, Kishida, & Ibaraki, 2006), (ii) evaluating solutions by incremental computation (Michel & Van Hentenryck, 2000; Van Hentenryck & Michel, 2005; Voudouris, Dorne, Lesaint, & Liret, 2001; Yagiura & Ibaraki, 1999), and (iii) reducing the number of variables to be considered by using LP and/or Lagrangian relaxation (Caprara, Fischetti, & Toth, 1999; Ceria, Nobili, & Sassano, 1998; Umetani, Arakawa, & Yagiura, 2013; Yagiura, Kishida, & Ibaraki, 2006).

To suggest an alternative, we develop a data mining approach for reducing the search space of local search algorithms. That is, we construct a k-nearest neighbor graph by extracting variable associations from the instance to be solved in order to identify promising pairs of flipping variables in the neighborhood search. We also develop a 4-flip neighborhood local search algorithm that flips four variables alternately along 4-paths or 4-cycles in the k-nearest neighbor graph. We incorporate an efficient incremental evaluation of solutions and an adaptive control of penalty weights into the 4-flip neighborhood local search algorithm.

2. 2-flip neighborhood local search

Local Search (LS) starts from an initial solution $\boldsymbol{x}$ and then iteratively replaces $\boldsymbol{x}$ with a better solution $\boldsymbol{x}'$ in the neighborhood $\mathrm{NB}(\boldsymbol{x})$ until no better solution is found in $\mathrm{NB}(\boldsymbol{x})$. For some positive integer r, let the r-flip neighborhood $\mathrm{NB}_r(\boldsymbol{x})$ be the set of solutions obtainable by flipping at most r variables in $\boldsymbol{x}$. We first develop a 2-Flip Neighborhood Local Search (2-FNLS) algorithm as a basic component of our algorithm. In order to improve efficiency, the 2-FNLS first searches $\mathrm{NB}_1(\boldsymbol{x})$, and then searches $\mathrm{NB}_2(\boldsymbol{x}) \setminus \mathrm{NB}_1(\boldsymbol{x})$ only if $\boldsymbol{x}$ is locally optimal with respect to $\mathrm{NB}_1(\boldsymbol{x})$.

The BIP is NP-hard, and the (supposedly) simpler problem of judging the existence of a feasible solution is NP-complete. We accordingly consider the following formulation of the BIP that allows violations of the constraints and introduce the following penalized objective function with penalty weights $w_i^+ \in \mathbb{R}_+$ ($\mathbb{R}_+$ is the set of nonnegative real values) for $i \in M_L \cup M_E$ and $w_i^- \in \mathbb{R}_+$ for $i \in M_G \cup M_E$.

$$
\begin{aligned}
\text{minimize} \quad & z(\boldsymbol{x}) = \sum_{j\in N} c_j x_j + \sum_{i\in M_L\cup M_E} w_i^+ y_i^+ + \sum_{i\in M_G\cup M_E} w_i^- y_i^- \\
\text{subject to} \quad & \sum_{j\in N} a_{ij}x_j - y_i^+ \le b_i, && i \in M_L, \\
& \sum_{j\in N} a_{ij}x_j + y_i^- \ge b_i, && i \in M_G, \\
& \sum_{j\in N} a_{ij}x_j - y_i^+ + y_i^- = b_i, && i \in M_E, \\
& x_j \in \{0,1\}, && j \in N, \\
& y_i^+ \ge 0, && i \in M_L \cup M_E, \\
& y_i^- \ge 0, && i \in M_G \cup M_E.
\end{aligned} \tag{2}
$$

For a given $\boldsymbol{x} \in \{0,1\}^n$, we can easily compute optimal $y_i^+ = |s_i(\boldsymbol{x}) - b_i|_+$ and $y_i^- = |b_i - s_i(\boldsymbol{x})|_+$, where we denote $|x|_+ = \max\{x, 0\}$.

Because the region searched by a single application of LS is limited, LS is usually applied many times. When a locally optimal solution is found, a standard strategy is to update the penalty weights and to resume LS from the obtained locally optimal solution. We accordingly evaluate solutions by using an alternative function $\tilde{z}(\boldsymbol{x})$, where the original penalty weight vectors $\boldsymbol{w}^+$ and $\boldsymbol{w}^-$ are replaced by $\tilde{\boldsymbol{w}}^+$ and $\tilde{\boldsymbol{w}}^-$, respectively, and these are adaptively controlled during the search (see Section 6 for details).

We first describe how 2-FNLS is used to search $\mathrm{NB}_1(\boldsymbol{x})$, which is called the 1-flip neighborhood. Let

$$
\Delta\tilde{z}_j(\boldsymbol{x}) = \begin{cases} \Delta\tilde{z}_j^{\uparrow}(\boldsymbol{x}) & j \in N \setminus X \\ \Delta\tilde{z}_j^{\downarrow}(\boldsymbol{x}) & j \in X, \end{cases} \tag{3}
$$

be the increase in $\tilde{z}(\boldsymbol{x})$ due to flipping $x_j \leftarrow 1 - x_j$, where

$$
\begin{aligned}
\Delta\tilde{z}_j^{\uparrow}(\boldsymbol{x}) &= c_j + \sum_{i\in S_j\cap(M_L\cup M_E)\cap\{l|s_l(\boldsymbol{x})\ge b_l\}} \tilde{w}_i^+ - \sum_{i\in S_j\cap(M_G\cup M_E)\cap\{l|s_l(\boldsymbol{x})< b_l\}} \tilde{w}_i^-, \\
\Delta\tilde{z}_j^{\downarrow}(\boldsymbol{x}) &= -c_j - \sum_{i\in S_j\cap(M_L\cup M_E)\cap\{l|s_l(\boldsymbol{x})> b_l\}} \tilde{w}_i^+ + \sum_{i\in S_j\cap(M_G\cup M_E)\cap\{l|s_l(\boldsymbol{x})\le b_l\}} \tilde{w}_i^-,
\end{aligned} \tag{4}
$$

are the increases in $\tilde{z}(\boldsymbol{x})$ due to flipping $x_j = 0 \to 1$ and $x_j = 1 \to 0$, respectively. 2-FNLS first searches for an improved solution obtainable by flipping $x_j \leftarrow 1 - x_j$ for $j \in N$. If an improved solution exists, it chooses j with the minimum value of $\Delta\tilde{z}_j(\boldsymbol{x})$ and flips $x_j \leftarrow 1 - x_j$.

We next describe how 2-FNLS is used to search $\mathrm{NB}_2(\boldsymbol{x}) \setminus \mathrm{NB}_1(\boldsymbol{x})$, which is called the 2-flip neighborhood. We derive conditions that reduce the number of candidates in $\mathrm{NB}_2(\boldsymbol{x}) \setminus \mathrm{NB}_1(\boldsymbol{x})$ without sacrificing the solution quality by expanding the results as shown in Yagiura, Kishida, and Ibaraki (2006). Let $\Delta\tilde{z}_{j_1,j_2}(\boldsymbol{x})$ be the increase in $\tilde{z}(\boldsymbol{x})$ due to simultaneously flipping the values of x_{j_1} and x_{j_2}.

Lemma 1. *If a solution $\boldsymbol{x}$ is locally optimal with respect to $\mathrm{NB}_1(\boldsymbol{x})$, then $\Delta\tilde{z}_{j_1,j_2}(\boldsymbol{x}) < 0$ holds only if $S_{j_1} \cap S_{j_2} \neq \emptyset$ and $x_{j_1} \neq x_{j_2}$.*

Proof. By the assumption of the lemma, $\Delta\tilde{z}_{j_1}(\boldsymbol{x}) \ge 0$ and $\Delta\tilde{z}_{j_2}(\boldsymbol{x}) \ge 0$ hold. It is clear from (4) that $\Delta\tilde{z}_{j_1,j_2} = \Delta\tilde{z}_{j_1}(\boldsymbol{x}) + \Delta\tilde{z}_{j_2}(\boldsymbol{x}) \ge 0$ holds if $S_{j_1} \cap S_{j_2} = \emptyset$.

We show that $\Delta\tilde{z}_{j_1,j_2}(\boldsymbol{x}) \ge 0$ holds if $S_{j_1} \cap S_{j_2} \neq \emptyset$ and $x_{j_1} = x_{j_2}$. First, we consider the case of $x_{j_1} = x_{j_2} = 1$. If $s_i(\boldsymbol{x}) = b_i + 1$ holds for $i \in S_{j_1} \cap S_{j_2} \cap (M_L \cup M_E)$, then decrease of the violation y_i^+ partly cancels by flipping $x_{j_1} = 1 \to 0$ and $x_{j_2} = 1 \to 0$ simultaneously. Similarly, if $s_i(\boldsymbol{x}) = b_i + 1$ holds for $i \in S_{j_1} \cap S_{j_2} \cap (M_G \cup M_E)$, then a new violation y_i^- occurs by flipping $x_{j_1} = 1 \to 0$ and