Operations Research Letters 44 (2016) 366–369

Contents lists available at ScienceDirect

Operations Research Letters

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

What is the best greedy-like heuristic for the weighted set covering problem?

Francis J. Vasko [a,*], Yun Lu [a], Kenneth Zyma [b]

[a] Department of Mathematics, Kutztown University, Kutztown, PA 19530, USA
[b] Computer Science Department, Kutztown University, Kutztown, PA 19530, USA

ARTICLE INFO

Article history:
Received 9 September 2015
Received in revised form
13 March 2016
Accepted 13 March 2016
Available online 22 March 2016

Keywords:
Heuristics
Weighted set covering problem
Greedy algorithms
Column knowledge functions
Row knowledge functions

ABSTRACT

The greedy heuristic for the weighted set covering problem is a "column knowledge" construction heuristic where cost and row coverage information are used to insert columns into the solution. In this paper, we analyze the performance of construction heuristics that expand on the column knowledge functions described by Vasko and Wilson (1984) and row knowledge functions described by Ablanedo-Rosas and Rego (2010). If redundant columns are removed from solutions, then the basic greedy heuristic gives essentially the best results.

1. Introduction

The set covering problem (SCP) is a classical combinatorial optimization problem. Its goal is to find a subset of columns that cover all the rows of a zero–one matrix at minimal cost. Many real life problems can be modeled with the SCP, such as strategic planning [12], production planning [13], facility location problems [9], area selection in conservation biology [6], as well as a variety of scheduling and routing problems.

Karp [4] showed that the SCP is NP-complete, which means that though there exist algorithms to solve the SCP precisely, which are based on branch-and-bound and branch-and-cut techniques [2], these algorithms can only solve instances of the SCP of a limited size and are time consuming. Therefore, considerable effort has been focused on discovering heuristics and metaheuristics that can find optimal or near optimal solutions to large-scale SCP problems in a reasonable amount of time. The latest works on metaheuristic approaches for the SCP include Genetic Algorithms, Ant Colony Optimization, Simulated Annealing, and Tabu Search, just to name a few. For a discussion of recent heuristic and metaheuristic approaches for solving the SCP, we suggest you consult Ren, Feng, Ke and Zhang [7].

Greedy heuristics construct a feasible solution to an SCP by adding columns to the solution until all the rows of the matrix are covered. These heuristics are fast and simple to implement, but are usually used only as starting points in more complex metaheuristics. For example, they may be used to generate an initial population for population-based metaheuristics [5,8] or used to generate the columns that comprise a concentrated problem in a heuristic concentration approach to the SCP [11].

In this paper, we simply wish to determine how well greedy, single-pass, construction heuristics perform (*on their own*) based on combining both row and column knowledge functions motivated by the work of Vasko and Wilson [10] and Ablanedo-Rosas and Rego [1].

In Section 2, we will formally define the weighted set covering problem. In Section 3, we will first define a family of greedy, single-pass, column knowledge construction heuristics based on the work of Vasko and Wilson [10]. Next, we will define a family of greedy, single-pass, row knowledge construction heuristics based on the work of Ablanedo-Rosas and Rego [1]. Finally, we will combine the work of Vasko and Wilson [10] and Ablanedo-Rosas and Rego [1] to define a family of single-pass construction heuristics that can use both column knowledge as well as row knowledge information in constructing a solution to the weighted set covering problem. In Section 4, we will empirically test the performance of 50 of these heuristics on 65 weighted set covering problems from Beasley's OR test library [3]. Finally, we will summarize our results in Section 4.

* Corresponding author. Tel.: +1 610 683 4417; fax: +1 610 683 4765.
E-mail address: vasko@kutztown.edu (F.J. Vasko).

http://dx.doi.org/10.1016/j.orl.2016.03.007
0167-6377/

2. Problem statement and background

2.1. Problem statement

The set covering problem (SCP) is the problem of covering the rows of an m-row, n-column, zero–one matrix (a_{ij}) by a subset of the columns at minimum cost. Specifically, let $A = (a_{ij})$ be an $m \times n$ zero–one matrix with $M = \{1, 2, \ldots, m\}$ and $N = \{1, 2, \ldots, n\}$ sets of rows and columns respectively. Column j is said to cover a row $i \in M$ if $a_{ij} = 1$. Each column $j \in N$ has associated with it a positive real cost c_j. The goal is to find a subset $S \subseteq N$, where each row $i \in M$ is covered by at least one column $j \in S$, with a minimum cost.

A mathematical formulation for the SCP is:

$$\text{Minimize} \quad \sum_{j=1}^{n} c_j x_j \qquad (0)$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \geq 1, \quad i = 1, \ldots, m, \qquad (1)$$

$$x_j \in \{0, 1\}, \quad j = 1, \ldots, n. \qquad (2)$$

where x_j is one if column j is in the solution and zero otherwise. Constraint set (1) ensures that each row is covered by at least one column and constraint set (2) ensures that the x_j's take on only the values zero or one. In this paper, we will focus on the weighted set covering problem (WSCP). That is, when the c_js are not all equal to one. When all the c_js are equal to one, the problem is referred to as the minimum cardinality or unit-cost set covering problem.

2.2. Background

Ablanedo-Rosas and Rego [1] introduced a number of normalization rules, i.e., row knowledge functions, used to generate surrogate constraints for the SCP and showed that these rules perform better than the classical greedy rule (when no redundant columns are removed). Previously, Vasko and Wilson [10] designed a heuristic for solving large set covering problems efficiently using a procedure that incorporates a set of column knowledge functions. In the next section, we will define a family of single-pass construction heuristics based on both the work of Vasko and Wilson [10] and the work of Ablanedo-Rosas and Rego [1] that use both row and column knowledge functions.

3. A family of single-pass construction heuristics

As previously defined, let M be the set of all rows, M_j the set of rows covered by column j; N the set of all columns, N_i the set of columns covering row i. Let R be the set of uncovered rows and S the set of columns included in a solution (i.e. $S = \{j \varepsilon N \mid x_j = 1\}$). The pseudo code for the family of column knowledge heuristics based on the work of Vasko and Wilson [10] is described below.

3.1. A family of greedy column knowledge single-pass construction heuristics

Step 0: Set $R = M$, $S = \varnothing$, $t = 1$, where $M = \{1, 2, \ldots, m\}$. Go to Step 1.

Step 1: If $R = \varnothing$, go to Step 2.
Otherwise, let $K_j = |M_j \cap R|$ for all $j \in N$, where $N = \{1, 2, 3, \ldots, n\}$, and choose $j^* = \arg\min\{f(c_j, K_j) \mid K_j > 0\}$ where $f(c_j, K_j)$ is defined below.
Set $R = R \setminus M_{j^*}$, $S = S \cup \{j^*\}$, $t = t + 1$, $M_j = \{i \in M \mid a_{ij} = 1\}$, and K_j denotes the number of positive coefficients of x_j in those rows of the current constraint set not yet covered. Go to Step 1.

Step 2: Sort the elements of S based on decreasing c_j values. Consider the elements $i \in S$ in order, and if $S \setminus \{i\}$ is a feasible cover, set $S = S \setminus \{i\}$.
When all $i \in S$ have been considered, S defines a prime cover (a cover with no redundant columns).

Vasko and Wilson [10] made use of the following seven column knowledge functions $f(c_j, K_j)$: (1) c_j, (2) c_j/K_j, (3) $c_j/(\log_2 K_j)$, (4) $c_j/(K_j \log_2 K_j)$, (5) $c_j/(K_j \ln K_j)$, (6) c_j/K_j^2, (7) $(c_j)^{1/2}/K_j^2$.

Using each of the seven column knowledge functions, seven solutions to an SCP can be constructed that are not necessarily unique. Step 2 is used to very efficiently remove any unneeded (redundant) columns that may have entered the solution early, but are no longer needed in the final solution. We will now define pseudo code for a family of row knowledge heuristics based on the work of Ablanedo-Rosas and Rego [1].

3.2. A family of greedy row knowledge single-pass construction heuristics

Step 0: Set $R = M$, $S = \varnothing$, $t = 1$, where $M = \{1, 2, \ldots, m\}$. Go to Step 1.

Step 1: If $R = \varnothing$ go to Step 2.
Otherwise, let $L_i = \sum_{j \in N_i} a_{ij}$, $\forall i \in R$, where L_i is the sum of all ones in row i and $g(L_i)$ is a row knowledge function defined below.
Choose $j^* = \arg\min\{c_j/[\sum_{i \in R} a_{ij} g(L_i)] \mid j \varepsilon N \setminus S\}$.
Set $R = R \setminus M_{j^*}$, $S = S \cup \{j^*\}$, $t = t + 1$, where $N = \{1, 2, 3, \ldots, n\}$ and $M_j = \{i \in M \mid a_{ij} = 1\}$. Go to Step 1.

Step 2: Sort the elements of S based on decreasing c_j values. Consider the elements $i \in S$ in order, and if $S \setminus \{i\}$ is a feasible cover, set $S = S \setminus \{i\}$. When all $i \in S$ have been considered, S defines a prime cover (a cover with no redundant columns).

Ablanedo-Rosas and Rego [1] made use of the following ten row knowledge functions $g(L_i)$:

1. $g(L_i) = 1$: All 1's (Chvatal's Rule)
2. $g(L_i) = \frac{1}{L_i}$: Absolute Coefficient Sum (ACS)
3. $g(L_i) = \left(\frac{1}{L_i}\right)^2$: Second Power ACS
4. $g(L_i) = \left\lfloor \left(\frac{1}{L_i}\right)^{1/2} \right\rfloor$: Floor of Half Power ACS
5. $g(L_i) = \left(\frac{1}{L_i}\right)^{1/2}$: Half Power ACS
6. $g(L_i) = \left(\frac{1}{L_i}\right)^{3/2}$: 1.5 Power ACS
7. $g(L_i) = \frac{1}{L_i - 1}$: Adjusted ACS
8. $g(L_i) = \left(\frac{1}{L_i - 1}\right)^2$: Second Power Adjusted ACS
9. $g(L_i) = \left(\frac{1}{L_i - 1}\right)^{1/2}$: Half Power Adjusted ACS
10. $g(L_i) = \left(\frac{1}{L_i - 1}\right)^{3/2}$: 1.5 Power Adjusted ACS.

Functions 1 through 10 were studied by Ablanedo-Rosas and Rego [1], but as indicated in the pseudo code above, they were **only** combined with one column knowledge function—the basic greedy. We note that their fourth row knowledge function, because it is taking the floor, will be 1 for $L_i = 1$ and 0 for $L_i > 1$. To avoid zero division in the algorithm, we set this function equal to 1 in all cases. We could just compare the basic greedy heuristic (column knowledge function #2) to the other six column knowledge functions and to the ten row knowledge functions, but we were interested in defining a family of greedy, single-pass construction heuristics which used both column and row knowledge in selecting the next column to enter the solution.