European Journal of Operational Research 209 (2011) 228–240

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

A primogenitary linked quad tree approach for solution storage and retrieval in heuristic binary optimization

Minghe Sun *

Department of Management Science and Statistics, College of Business, The University of Texas at San Antonio, San Antonio, TX 78249, United States

ARTICLE INFO

Article history:
Received 22 December 2009
Accepted 22 September 2010
Available online 29 September 2010

Keywords:
Primogenitary linked quad tree
Data structure
Heuristic procedures
Binary optimization
Combinatorial optimization

ABSTRACT

A data structure, called the primogenitary linked quad tree (PLQT), is used to store and retrieve solutions in heuristic solution procedures for binary optimization problems. Two ways are proposed to use integer vectors to represent solutions represented by binary vectors. One way is to encode binary vectors into integer vectors in a much lower dimension and the other is to use the sorted indices of binary variables with values equal to 0 or equal to 1. The integer vectors are used as composite keys to store and retrieve solutions in the PLQT. An algorithm processing trial solutions for insertion into or retrieval from the PLQT is developed. Examples are provided to demonstrate the way the algorithm works. Another algorithm traversing the PLQT is also developed. Computational results show that the PLQT approach takes only a very tiny portion of the CPU time taken by a linear list approach for the same purpose for any reasonable application. The CPU time taken by the PLQT managing trial solutions is negligible as compared to that taken by a heuristic procedure for any reasonably hard to solve binary optimization problem, as shown in a tabu search heuristic procedure for the capacitated facility location problem. Compared to the hashing approach, the PLQT approach takes the same or less amount of CPU time but much less memory space while completely eliminating collision.

Many combinatorial optimization problems are NP hard and, therefore, are very difficult to solve. Exact algorithms can solve small problem instances and heuristic procedures are usually employed for large ones. Researchers have developed many metaheuristic methods, such as simulated annealing (Kirkpatrick et al., 1983), tabu search (Glover, 1989, 1990a,b; Glover and Laguna, 1997), scatter search (Glover et al., 2000), genetic algorithms (Holland, 1992), and ant colony optimization (Deneubourg et al., 1983; Deneubourg and Goss, 1989). These metaheuristic methods provide frameworks or guidelines in forming a strategy for solving hard combinatorial optimization problems. To solve a specific type of problems, a metaheuristic method has to be tailored to form a specific heuristic procedure to take advantage of the problem structure. Many heuristic procedures using these metaheuristic methods have been developed for many hard combinatorial optimization problems. For problems with realistic sizes, heuristic procedures usually take a very long computation time to find good, but not necessarily optimal, solutions. In the solution process, many trial solutions are evaluated.

The purpose of this study is to develop a data structure approach to store and retrieve trial solutions in heuristic procedures for binary optimization problems, a specific but the most common type of combinatorial optimization problems. The data structure is called the primogenitary linked quad tree (PLQT), a quad tree with special structures. This approach is so efficient that the computation time it takes is unnoticeable as compared to that a heuristic procedure takes. Therefore, it is a handy tool for researchers to use in developing their heuristic procedures for binary optimization problems. The major contribution of this study is the application of the PLQT to the storage and retrieval of trial solutions in heuristic procedures for binary optimization problems. Another contribution is the development of different ways to represent binary solutions with integer vectors.

In Section 1, the binary optimization problem and the necessity for trial solution storage and retrieval are discussed. In Section 2, alternative ways of representing trial solutions are discussed and two ways of using integer vectors are proposed. The PLQT and algorithms managing it are described in Section 3. Examples demonstrating the insertion of integer vectors into and the retrieval of integer vectors from a PLQT are given in Section 4. Computational results are presented in Section 5. Concluding remarks and summaries are given in Section 6.

1. Introduction

A binary optimization problem involves a set of objects and a subset of them meeting certain restrictions needs to be selected

* Tel.: +1 (210) 458 5777; fax: +1 (210) 458 6350.
E-mail address: minghe.sun@utsa.edu
URL: http://faculty.business.utsa.edu/msun

0377-2217/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2010.09.034

based on one or more criteria or objectives. In a project or investment portfolio selection problem (Stummer and Sun, 2005), for example, the objects are the projects or investment instruments. A portfolio is a selected subset. The major objective is the maximization of the total expected return on investment although there are other objectives in a multiple criteria problem (Stummer and Sun, 2005). The restrictions include the limited budgets, and the diversification and balancing requirements among others. In a facility location problem (Delmaire et al., 1999; Ducati et al., 2004; Sun, 2006a, 2009a), as another example, the objects are the potential sites to locate facilities. A subset of these sites is selected to have the facilities actually established. Each site has a fixed cost to establish and to operate the facility and has fixed costs or variable costs to serve the clients, such as transporting the products to the customers. The major objective is to minimize the total costs. The restrictions are to meet the client demands possibly within the capacities of the selected facilities. In a variable or feature selection problem (Sun, 2009b; Sun and Xiong, 2003) in regression or classification analysis, the objects are the independent variables and their functions, such as monomials. A subset of these variables is selected to fit a regression function or a set of discriminant functions. The criteria may be the minimization of the regression or classification errors while avoiding overfitting by using a minimum number of independent variables.

Common to all binary optimization problems is that the status of an object i can be represented by a binary variable y_i, *i.e.*, $y_i = 0$ if object i is not selected and $y_i = 1$ otherwise. A solution with a selected subset for a problem with n objects can be represented by a vector of n binary variables $\mathbf{y} = (y_0, y_1, \ldots, y_i, \ldots, y_{n-2}, y_{n-1})$. Each combination of the n binary variables is a possible solution. Hence, a problem with n binary variables has 2^n possible solutions. However, usually a small portion of the 2^n possible solutions is feasible, *i.e.*, satisfying all restrictions. In a heuristic procedure, usually a very small portion of the 2^n possible solutions is evaluated. In addition to binary variables, most problems also involve real, *i.e.*, continuous, variables. The number of continuous variables is denoted by n'.

Usually a binary optimization problem can be formulated as a binary mathematical programming model (Nemhauser and Wolsey, 1988; Wolsey, 1998) where a restriction is represented by a constraint and a criterion is represented by an objective function. A binary minimization problem with one objective function may be written as

$$\min \quad f(\mathbf{x}, \mathbf{y}) \tag{1}$$

$$\text{s.t.} \quad g_i(\mathbf{x}, \mathbf{y}) \geqslant 0 \quad \text{for } i = 1, \ldots, \eta, \tag{2}$$

$$\mathbf{x} \in \Re^{n'}, \quad \mathbf{y} \in B^n. \tag{3}$$

In the model, $\mathbf{x}$ represents the vector of real variables in the n'-dimensional Euclidean space $\Re^{n'}$ and B^n is the collection of all ordered n-tuples of 0s and 1s. The model is a pure binary programming model if $n' = 0$ and is a mixed binary programming model otherwise. In this study, $n > 0$ is assumed.

In a trial solution, the number of binary variables y_i with $y_i = 0$ is denoted by n_0 and that with $y_i = 1$ is denoted by n_1. Each $\mathbf{y} \in B^n$ determines a trial solution. Once $\mathbf{y}$ is determined, the trial solution can be evaluated to determine $f(\mathbf{x}, \mathbf{y})$. Evaluating a trial solution is usually very time consuming in the solution process. In a portfolio selection problem, for example, a linear programming problem needs to be solved (Stummer and Sun, 2005); in a capacitated facility location problem (Ducati et al., 2004; Sun, 2009a), a transportation problem needs to be solved; and in a single source facility location problem (Delmaire et al., 1999), a generalized assignment problem, which is itself a binary optimization problem, needs to be solved, to evaluate a solution. Therefore, each solution needs to be evaluated at most once in an efficient heuristic solution procedure. After the evaluation, the solution may be saved or stored and may be retrieved later if needed. In the solution process using a heuristic procedure, many solutions are evaluated.

When such a problem is solved with a heuristic procedure, the solution process usually follows a selective trajectory in the solution space. At a visited solution on the trajectory, a neighborhood is created by evaluating or retrieving one or more trial solutions. A move is the transition from the current solution to another in the neighborhood. A trial solution may be obtained by changing the value of one binary variable, called a simple move, or by changing one binary variable from 0 to 1 and another from 1 to 0, called an exchange or swap move. For each trial solution, the saved solutions may be searched to find out if it has already been evaluated and saved. If found, the trial solution is retrieved and does not need to be evaluated again; otherwise, the trial solution is evaluated and saved. Based on some predefined rules, the procedure selects one of these evaluated trial solutions as the next solution to be visited to move to. Once a solution is visited, it should not be visited again. If a solution is visited the second time, the same trajectory may be followed again if the predefined rules do not change and, hence, repetition or cycling may occur. Therefore, measures are usually taken by heuristic procedures to prohibit the visit of solutions which have already been visited. For this purpose, the visited solutions need to be memorized in some way.

Hence, there are two purposes for storing the evaluated solutions. One purpose is to save computation time. Once a trial solution is evaluated, it does not need to be evaluated again and only needs to be retrieved. The other purpose is to prevent repetition or cycling. Once a solution is visited, it may not be selected to visit again. However, storing and retrieving trial solutions should not take much computation time and memory space for the heuristic procedure to be efficient.

The PLQT approach developed in this study can be used by any heuristic procedure for any binary optimization problem. The PLQT is a new data structure recently developed by Sun (2006b) for fast access of data, or data with composite keys, in $\Re^K$ for $K \geqslant 1$. It is an enhancement of the more traditional quad tree data structure (Finkel and Bentley, 1974; Habenicht, 1982, 1991; Sun and Steuer, 1996a,b). Quad trees as data structures were first introduced by Finkel and Bentley (1974). Among others, quad trees have been employed in discrete multiple criteria optimization, geometric information systems, image processing, and computer aided design and computer aided manufacturing. Compared to the traditional quad tree, the PLQT uses substantially less computation time and takes considerably less memory or storage space. Sun (2006b) showed through computational experiments that the PLQT is much faster than the traditional quad tree, which is in turn much faster than the linear list, in identifying, storing and retrieving nondominated solutions in discrete multiple criteria optimization. The PLQT is even much faster in storing and retrieving trial solutions in heuristic procedures than in discrete multiple criteria optimization because the PLQT does not need to be reconstructed as in the application reported by Sun (2006b).

2. Integer vector representation of binary solutions

The number of bits of memory used by a computer language to represent an integer is denoted by b. Most computer languages use 2 or 4 bytes. If 2 bytes are used, $b = 16$, or if 4 bytes are used, $b = 32$. Although both positive and negative integers represented by b bits can be used to represent solutions, only unsigned integers are used in the following discussion for easy description. Therefore, the range of integers a computer can represent is between 0 and $2^b - 1$ and the maximum number of different integers a computer can represent is 2^b.