A Tabu Search Algorithm for the Quadratic Assignment Problem

ALFONSAS MISEVICIUS  
misevi@soften.ktu.lt  
Department of Practical Informatics, Kaunas University of Technology, Studentų St. 50-416a,  
LT-3031 Kaunas, Lithuania  

Received December 13, 2002; Revised November 6, 2003; Accepted February 11, 2004

Abstract. Tabu search approach based algorithms are among the widest applied to various combinatorial optimization problems. In this paper, we propose a new version of the tabu search algorithm for the well-known problem, the quadratic assignment problem (QAP). One of the most important features of our tabu search implementation is an efficient use of mutations applied to the best solutions found so far. We tested this approach on a number of instances from the library of the QAP instances—QAPLIB. The results obtained from the experiments show that the proposed algorithm belongs to the most efficient heuristics for the QAP. The high efficiency of this algorithm is also demonstrated by the fact that the new best known solutions were found for several QAP instances.

Keywords: heuristics, meta-heuristics, tabu search, quadratic assignment problem

1. Introduction

The quadratic assignment problem (QAP) is formulated as follows. Let two matrices $A = (a_{ij})_{n \times n}$ and $B = (b_{kl})_{n \times n}$ and the set $\Pi$ of permutations of the integers from $1$ to $n$ be given. Find a permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n)) \in \Pi$ that minimizes

$$z(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi(i)\pi(j)}.$$

The context in which Koopmans and Beckmann [20] first formulated this problem was the facility location problem. In this problem, one is concerned with locating $n$ facilities (units) on $n$ locations (sites) with some physical products flowing between the facilities, and with distances between the locations. The element $a_{ij}$ is the "flow" from the facility $i$ to facility $j$, and the element $b_{kl}$ represents the distance between the locations $k$ and $l$. The permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ can be interpreted as an assignment of facilities to locations (so, the permutation element $\pi(i)$ denotes the location what facility $i$ is assigned to). Solving the QAP means searching for an assignment $\pi$ that minimizes the "transportation cost" between facilities—$z$.

An important area of the application of the QAP is computer-aided design (CAD), more precisely, the placement of electronic components into positions on a board (chip) [16]. Other applications of the QAP are: campus planning [8], image processing (design of grey patterns) [35], ranking of archaeological data [21], testing of electronic devices [9], typewriter keyboard design [5], etc. (see, for example, [3]).
It has been proved that the QAP is NP-hard [27]. Problems of size, say \( n > 30 \), are not, to this date, practically solvable in terms of obtaining exact solutions. Therefore, heuristic approaches have to be used for solving medium- and large-scale QAPs, among them: ant algorithms [11], constructive algorithms [12], genetic algorithms [10, 23, 37], greedy randomized search (GRASP) [22], iterative improvement [26, 29], simulated annealing [2, 7], and others (see, for example, [3]). Starting from 1990, several authors applied tabu search to the QAP, first of all, Battiti and Tecchiolli [1], Skorin-Kapov [30], Taillard [34].

This paper is organized as follows. In Section 2, a tabu search approach is outlined. Section 3 describes the details of the proposed tabu search algorithm for the quadratic assignment problem. The results of the computational experiments on the various QAP instances are presented in Section 4. Section 5 completes the paper with concluding remarks.

2. Tabu search for combinatorial optimization problems

The very basic definitions related to the optimization problems and tabu search are as follows. Let \( S \) be a set of solutions \((s_1, s_2, \ldots)\) of a combinatorial optimization problem with objective (fitness) function \( f: S \to \mathbb{R}^1 \). Furthermore, let \( N: S \to 2^S \) be a neighbourhood function which defines for each \( s \in S \) a set \( N(s) \subseteq S \) — a set of neighbouring solutions of \( s \). Each solution \( s' \in N(s) \) can be reached from \( s \) by an operation called a move. Generally, the move follows the objective function evaluation, which is called a trial.

Tabu search technique was developed independently by Glover [13, 14], and Hansen and Jaumard [17]. This method has become very popular and is widely used for a variety of problems [15]. Tabu search, like simulated annealing, is based on the neighbourhood search with local-optima avoidance but in a rather deterministic way. The key idea of tabu search is allowing climbing moves when no improving neighbouring solution exists. However, some moves are to be forbidden at a present search iteration in order to avoid cycling.

So, tabu search starts from an initial solution \( s \), maybe, randomly generated in \( S \) and moves repeatedly from a solution to a neighbouring one. At each step of the procedure, a set (subset) \( N(s) \) of the neighbouring solutions of the current solution \( s \) is considered and the move that improves most the objective function value \( f \) is chosen. If there are no improving moves, tabu search chooses one that least degrades the objective function. Roughly speaking, a move is performed to the best neighbour \( s' \) in \( N(s) \) (even if \( f(s') > f(s) \)).

In order to avoid the returning to the local optimal solution just visited, the reverse move must be forbidden (prohibited). This is done by storing this move (or an attribute of the move) in a memory (or more precisely short-time-memory) managed like a circular list \( T \) and called a tabu list. In that way, the tabu list keeps information on the last \( h \) (\( h = |T| \)) moves which have been done during the search process. The parameter \( h \) is called a tabu list size (length); the size of tabu list is crucial: if it is too small, cycling will occur, whereas if it is too large, it will restrict the search in “promising regions”. Thus, a move from \( s \) to \( s' \) is considered as tabu if it (or its attribute) is contained in the list \( T \). This way of proceeding hinders the algorithm from returning to a solution reached in the last \( |T| \) iterations. However, it might be worth returning after a while to a solution visited previously to search in another direction. Consequently, an aspiration criterion is introduced to permit the tabu status to be dropped under certain favourable circumstances (for example, a tabu move from \( s \) to \( s' \) is