Available online at www.sciencedirect.com
SCIENCE DIRECT
Discrete Optimization 3 (2006) 63–77
DISCRETE OPTIMIZATION
www.elsevier.com/locate/disopt

Iterative patching and the asymmetric traveling salesman problem

Marcel Turkensteen[a,*], Diptesh Ghosh[c], Boris Goldengorin[b], Gerard Sierksma[b]

[a] *Faculty of Economics, University of Groningen, P.O. Box 800, 9700 AV Groningen, The Netherlands*
[b] *Faculty of Economics, University of Groningen, The Netherlands*
[c] *P&QM Area, Indian Institute of Management, Ahmedabad, India*

Received 31 August 2004; received in revised form 26 April 2005; accepted 17 October 2005
Available online 24 January 2006

Abstract

Although Branch-and-Bound (BnB) methods are among the most widely used techniques for solving hard problems, it is still a challenge to make these methods smarter. In this paper, we investigate *iterative patching*, a technique in which a fixed patching procedure is applied at each node of the BnB search tree for the Asymmetric Traveling Salesman Problem. Computational experiments show that iterative patching results in general in search trees that are smaller than the classical BnB trees, and that solution times are lower for usual random and sparse instances. Furthermore, it turns out that, on average, iterative patching with the Contract-or-Patch procedure of Glover, Gutin, Yeo and Zverovich (2001) and the Karp–Steele procedure are the fastest, and that 'iterative' Modified Karp–Steele patching generates the smallest search trees.

Keywords: Asymmetric traveling salesman problem; Branch-and-Bound; Upper bounding; Patching

1. Introduction

The Asymmetric Traveling Salesman Problem (ATSP) is usually solved exactly by means of Branch-and-Bound (BnB) algorithms and Branch-and-Cut (BnC) algorithms; see [8]. In BnB type algorithms, an Assignment Problem (AP) is solved at every node of this tree, and the value of the optimal AP solution serves as a lower bound of the ATSP solution. A part of the search tree can be discarded when its lower bound exceeds an upper bound. This upper bound is usually the value of a shortest complete tour found so far. A class of heuristics applied to construct such a tour is *patching*. The question is: at which nodes of the search tree should such a tour be constructed? Patching at a node may reduce the search tree and the solution time, but if the reduction is too small, the overall solution time is increased due to the time invested in patching.

In the literature, the most effective BnB methods do not patch at each node; see for example, [13,3]. These methods use a best first search strategy, i.e., the subproblem with the smallest lower bound is solved first. According to these studies, patching at every node is too time-consuming.

In this paper, we consider a BnB algorithm that applies depth first search, which means that the most recently generated subproblem is solved first. This strategy requires algorithms to use much less computer memory than do

* Corresponding author.
E-mail address: m.turkensteen@rug.nl (M. Turkensteen).

1572-5286/$ - see front matter
doi:10.1016/j.disopt.2005.10.005

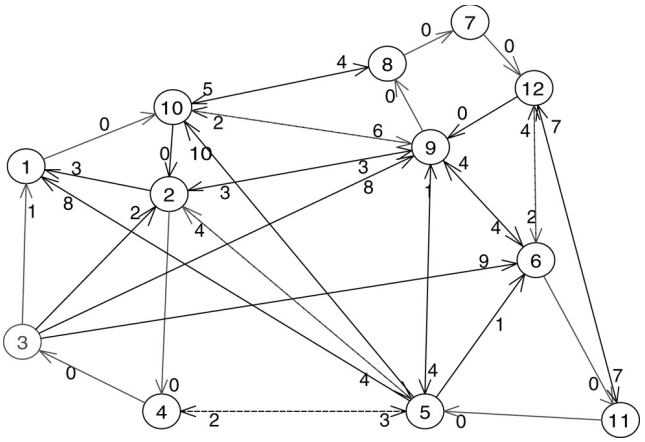

Fig. 1. ATSP instance.

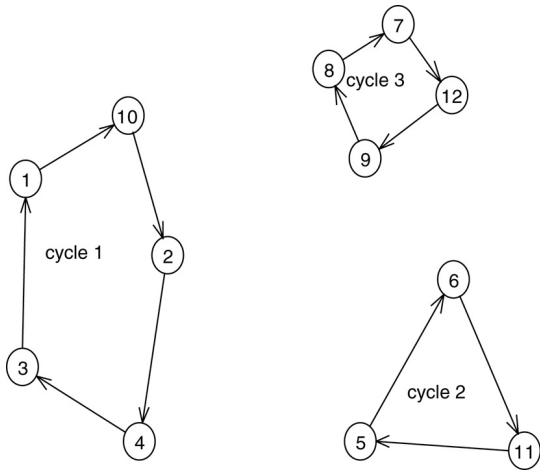

Fig. 2. Minimum cycle cover.

best first strategies. Hence, it is useful for solving large problems. We apply *iterative patching*, in which a fixed patching procedure is applied at every node of the BnB depth first search tree. Four iterative patching procedures are considered in our computational experiments. These procedures are described in [5].

Given a set of locations and the distance between any pair of locations, the ATSP is the problem of finding a shortest Hamiltonian tour; i.e., a shortest round trip visiting each location exactly once. Fig. 1 is an example of an underlying graph that defines an instance of an ATSP. The nodes of the graph represent locations, and the arcs the connections between the locations. A number next to an arrowhead denotes the cost of traveling along that arc.

General instances of the ATSP are often solved to optimality by means of enumeration algorithms, in which a fraction of all feasible solutions are checked. BnB methods explore the solution space by using a search tree. We discuss BnB algorithms that solve an Assignment Problem (AP) at each node of the corresponding search tree. After solving the AP a minimum cycle cover F is obtained, say, consisting of k cycles ($k \geq 1$). In the example of Fig. 2, three cycles are generated. If $k > 1$, the subcycles in F can be *patched* into a complete tour. BnB algorithms use the value of a patching solution as an upper bound by which nodes of the search tree are fathomed.

A *patching operation* is the simultaneous deletion of two arcs from a cycle cover and the insertion of two other arcs, such that the number of cycles is reduced by one. In our example, two patching operations are needed for the generation of a complete tour (see Fig. 3), namely first arcs (2,4), (5,6) are deleted and (2,6) and (5,4) are inserted, and then we delete (12,9) and (2,6) and insert (2,9) and (12,6). The resulting tour is generally feasible but not optimal.