European Journal of Operational Research 232 (2014) 221–233

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

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

Innovative Applications of O.R.

Simulated annealing and tabu search approaches for the Corridor Allocation Problem ☆

H. Ahonen, A.G. de Alvarenga, A.R.S. Amaral *

Departamento de Informática, Universidade Federal do Espírito Santo, 29060-900 Vitória, ES, Brazil

ARTICLE INFO

Article history:
Received 11 July 2012
Accepted 9 July 2013
Available online 17 July 2013

Keywords:
Facilities planning and design
Tabu search
Simulated annealing
Combinatorial optimization

ABSTRACT

In the Corridor Allocation Problem, we are given n facilities to be arranged along a corridor. The arrangements on either side of the corridor should start from a common point on the left end of the corridor. In addition, no space is allowed between two adjacent facilities. The problem is motivated by applications such as the arrangement of rooms in office buildings, hospitals, shopping centers or schools. Tabu search and simulated annealing algorithms are presented to minimize the sum of weighted distances between every pair of facilities. The algorithms are evaluated on several instances of different sizes either randomly generated or available in the literature. Both algorithms reached the optimal (when available) or best-known solutions of the instances with $n \leqslant 30$. For larger instances with size $42 \leqslant n \leqslant 70$, the simulated annealing implementation obtained smaller objective values, while requiring a smaller number of function evaluations.

1. Introduction

The Corridor Allocation Problem (Amaral, 2012), hereafter abbreviated as CAP, seeks to arrange n facilities without overlap along a corridor subject to two constraints: the arrangement on both sides of the corridor should start from a common point on the left end of the corridor; and no space is allowed between two adjacent facilities. We are given the length l_i of each facility i and the nonnegative amount of flow c_{ij} between facility i and j. The width of the corridor is considered to be negligible, thus the distance between two facilities i and j with respect to a certain CAP layout is the x-distance between their centers. The *total flow cost* is a weighted sum of the distances between each pair of facilities using the parameters c_{ij} as weights. The objective of the CAP is to find a layout that minimizes the total flow cost over all possible layouts.

The CAP has close relationships with other problems in the literature. For example, in the *single row facility layout problem* (SRFLP) we wish to find an arrangement of the facilities of known lengths, all placed at the same side of the corridor, so as to minimize the total flow cost. Surveys (Anjos and Liers, 2012; Kothari and Ghosh, 2012) and several methods are presented in the SRFLP literature: *exact methods* (e.g. Amaral, 2006, 2008, 2009b, 2009a; Anjos and Vannelli, 2008; Simmons, 1969), *heuristic methods* (e.g. Datta et al., 2011; de Alvarenga et al., 2000; Kothari and Ghosh, 2013), and *lower bounding methods* (e.g. Amaral and Letchford, 2012; Anjos et al., 2005; Anjos and Yen, 2009; Hungerländer and Rendl, 2012).

The CAP is also close to the *double row layout problem* (DRLP) (e.g. Amaral, 2013; Chung and Tanchoco, 2010; Heragu and Kusiak, 1988; Zhang and Murray, 2012), which is originally motivated by the arrangement of machines in a manufacturing system. In the DRLP, the facilities are placed on both sides of the corridor but the upper and lower arrangements do not have to start from a common point; and some space may be allowed between adjacent facilities.

Amaral (2012) presented a mixed integer programming (MIP) formulation of the CAP. However, the CAP is NP-Hard, which limits the efficiency of an exact approach. For example, a CAP instance of size of $n = 15$ could not be solved to optimality by cplex after 8.6 hours of execution time. Therefore, there is a quest for heuristic approaches that can efficiently deal with large instances of the problem. In this regard, Amaral (2012) proposed a metaheuristic algorithm for the CAP and tested instances of size $n = 30$ facilities.

A recent working paper by Ghosh and Kothari (2012) develops a hybrid genetic algorithm and a scatter search algorithm with path-relinking for the CAP. Another recent working paper by Hungerländer and Anjos (2012) studies the CAP and presents lower bounds using semidefinite optimization and a heuristic to obtain feasible layouts from the solutions of the semidefinite programming (SDP) relaxations.

☆ This work was carried out while the last author was with CEG/IST – Instituto Superior Técnico, Technical University of Lisbon, Portugal.

* Corresponding author. Tel.: +55 27 4009 2679.
E-mail addresses: hannu@inf.ufes.br (H. Ahonen), agomes@inf.ufes.br (A.G. de Alvarenga), amaral@inf.ufes.br (A.R.S. Amaral).

0377-2217/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2013.07.010

The CAP is important from a practical point of view. It has applications such as the arrangement of rooms in office buildings, hospitals, shopping centers or schools (Amaral, 2012). Other applications include the layout design problem where the facility is divided into bays, which are arranged along a central spine. This spine configuration is a common design for the layout in semiconductor wafer fabrication facilities (Yang and Peters, 1997).

In this paper, tabu search and simulated annealing algorithms are implemented for the CAP. An adequate neighborhood structure and a local search procedure, which are exploited by the two algorithms, are explained in the next section. Then, Section 3 presents the elements of the tabu search implementation such as the tabu status mechanism, aspiration criterion and diversification. Section 4 describes the simulated annealing implementation and its constituents: the classical simulated annealing procedure and the reversed simulated annealing procedure, which starts with a low temperature and increases it. In Section 5, the tabu search and simulated annealing algorithms are evaluated on some instances with $n \leqslant 15$ taken from Simmons (1969), Amaral (2006) and Amaral (2012); and on some others having $n = 30$ facilities from Anjos and Vannelli (2008). In addition, the algorithms are evaluated on even larger instances: thirty-six randomly generated instances having $n = 60$ and a selected set of "sko" instances with sizes 42, 49, 56 and 64 of Anjos and Yen (2009); and two AKV instances of sizes 60 and 70 of Anjos et al. (2005).

2. The neighborhood structure and local search procedure

The tabu search and simulated annealing implementations both exploit the same neighborhood structure described in Section 2.1. Similarly, both implementations contain calls to the local search procedure explained in Section 2.2.

2.1. The neighborhood structure

The neighborhood structure used in this work is defined by two sets of swap moves: set C of column swaps and set E of row element swaps. The swaps are applied to a $2 \times n$ matrix representation of an n-facility arrangement, in which a non-zero value of a row element indicates the position of a facility on the corresponding side of the corridor. A zero value means that the facility is not present at that side of the corridor. That is, an arrangement is a matrix $A = (a_{ij})$ with elements

$$a_{ij} = \begin{cases} k, & \text{if facility } j \text{ is at position } k \text{ on side } i, \\ 0, & \text{if facility } j \text{ is not on side } i, \end{cases} \quad (1)$$

where $i = 1,2$; $j = 1,2,\ldots,n$; and $k \in \{1,2,\ldots,n\}$.

For example, if $n = 5$, the matrix

$$A = \begin{pmatrix} 0 & 2 & 0 & 1 & 0 \\ 3 & 0 & 1 & 0 & 2 \end{pmatrix} \quad (2)$$

represents an arrangement, in which facilities 2 and 4 (cf. the elements of the first row at columns 2 and 4) are on one side of the corridor in the order 4 and 2, while the order of facilities on the other side of the corridor is 3, 5 and 1.

A *column swap* is defined as an exchange of two columns in the matrix. This kind of swap may result in changing facility order only on one side of the corridor or in moving a facility from one side of the corridor to the other. For example, the swap of columns 2 and 4 in (2) puts facility 2 as the first and facility 4 as the second facility on the upper side of the corridor, and the swap of columns 1 and 2 modifies the arrangements on both sides of the corridor placing facility 1 as the second facility on the upper side of the corridor and facility 2 as the third facility on the lower side of the corridor.

A *row element swap* exchanges elements of a given column and associates a given position to the new non-zero element. For example in (2), a swap of the elements in the third column with a given new position, say 3, for the non-zero element changes facility 3 to be the third facility on the upper side of the corridor. If the new given position of the non-zero element happens to be before other positions on the new side, an update of the succeeding positions may be needed. For example, a swap of the elements in the third column with the given new position for the non-zero element being 1 means that facility 3 should become the first facility on the upper side of the corridor; then, the positions of facilities 4 and 2 need to be updated to be 2 and 3, respectively. Similarly the positions on the side from which a facility was removed must be adjusted. In our example this means assigning facility 5 to position 1 and facility 1 to position 2.

The number of column swaps is equal to $n(n-1)/2$. The number of row element swaps will depend on the current matrix, if all possible destination positions are allowed to the swap element. In our implementation only the last position is allowed. The number of row element swaps is, in this case, equal to n.

2.2. Local search

Another common element to the tabu search and simulated annealing implementations consists of calls to a local search procedure called Procedure 1.

By which the current solution S with cost $c(S)$ is iteratively improved by application of both column and row element swaps as long as this is possible.

Procedure 1. localSearch

```
    input: Solution S, array Swaps
    output: Best solution SBestNow
1   STest ← S; SNow ← S
2   SBestNow ← S; minVal ← c(S);
3   repeat
4     improved ← FALSE;
5     for i = 0 to Swaps.length do
6       STest← applySwap(Swaps[i], SNow);
7       if c(STest) < minVal then
8         SNow ← STest;
9         SBestNow ← SNow;
10        minVal ← c(SNow);
11        improved ← TRUE;
12      end
13    end
14  until improved = FALSE;
```

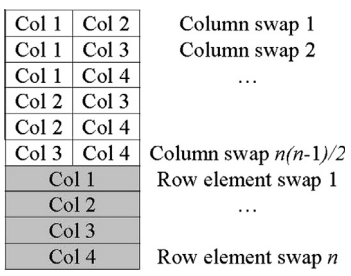

Fig. 1. The array *Swaps* for $n = 4$. The array *Swaps* consists of the two types of swaps: column swaps and row element swaps.