Sistemas Inteligentes
Grado en Ingeniería Informática
elisa.guerrero@uca.es
Departamento Ingeniería Informática

Lecture 2. Metaheuristics: single solution approaches

1. TABOO SEARCH

Read de following text extracted from the paper of M. Laguna, where an implementation of tabu search applied to the n-queens problema is proposed.
Answer the red questions in a new document.

A Guide to Implementing Taboo Search

Manuel Laguna

Introduction

The term *taboo search* (TS) is now familiar to many researchers and practitioners in the field of artificial intelligence. This paper is written for those who are interested in taboo search and would like to implement it effectively.

In order to discuss TS implementation issues, a combinatorial problem with a simple and regular structure is desirable as an example. The *n-queens* problem provides an excellent context for our discussion. The n-queens is considered a classical combinatorial problem in the artificial intelligence (AI) literature and has been extensively used as a benchmark for AI search. In fact, this problem has even found some practical significance in the area of VLSI testing and traffic control (Sosic and Gu 1989).

The N-Queens Problem

In simple terms, the n-queens problem consists of placing *n* queens on a *n* x *n* chessboard in such a way that no two queens capture each other. Thus, no two queens should be placed on the same row, the same column, or the same diagonal. If two queens are placed such that they are able to capture each other, it is said that a "collision" has occurred.

Therefore, from an optimization point of view the problem is to find a configuration that minimizes the total number of collisions.
The optimal solution to this problem is obviously zero, since it is known that there exists at least, one configuration for which no collisions occur.
Let the queens be indexed by the letter *i* (for $i = 1, \ldots, n$), and let queen *i* be placed always on the *ith* column. If we let C(i) be the index of the row where queen i is placed, then a configuration is given by the following permutation:

$$C = \{C(1), C(2), \ldots, C(n)\}$$

which fully specifies the exact position of the *n* queens on the chessboard.
This representation guarantees that no two queens will attack each other on the

same row or the same column. The problem is then to minimize the total number of collisions on the diagonals.

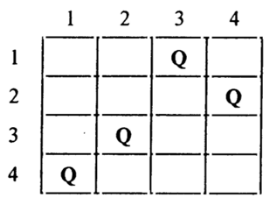

Figure 1 shows a 4 x 4 chessboard that corresponds to the permutation C= {4,3,1,2}. Note that this configuration has a total of 2 collisions (i.e., queens 1 and 2 as well as queens 3 and 4 are attacking each other).

Basic Taboo Search

The philosophy of taboo search is to derive and exploit a collection of principles of intelligent problem solving. This algorithm starts with a single solution, and searches for better solutions, applying actions and moving between neighbour states

A fundamental element underlying taboo search is the use of flexible memory. From the standpoint of taboo search, flexible memory embodies the dual processes of creating and exploiting structures to take advantage of history (hence combining the activities of acquiring and profiting from information).

Before presenting implementation details, we focus on the 7-queens problem to introduce and illustrate the basic components of taboo search. First we assume that an initial solution for this problem can be constructed, e.g., randomly.

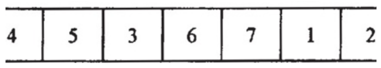

Figure 1. Initial State

1. Represent in a chess board this initial state
2. How many collisions are?
3. Design an evaluation function for this problem according to this representation (Matlab)

Pairwise exchanges (or swaps) are frequently used to define neighborhoods in permutation problems, identifying moves that lead from one solution to the next. In our problem, a swap exchanges the position of two queens as illustrated in Figure 2. Therefore, the complete neighborhood of a given current solution consists of the 21 adjacent solutions that can be obtained by such swaps.

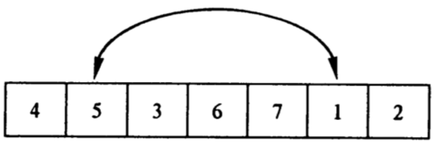

Figure 2. Swaping contents of Queen 2 and 6

4. Design a function to return all the successors states from current, ordered by their evaluation function (Matlab)

As a basis for preventing the search from repeating swap combinations tried in the recent past, potentially reversing the effects of previous moves by interchanges that might return to previous positions, we will classify as taboo all swaps composed of any of the most recent pairs of such queens; in this case, for illustrative purposes, the three most recent pairs. This means that a queen pair will be kept taboo for a duration (tenure) of 3 iterations.

Since exchanging queens 2 and 6 is the same as exchanging queens 6 and 2, both may be represented by the pair (2, 6). Thus, a data structure such as the one shown in Figure 3 may be used.

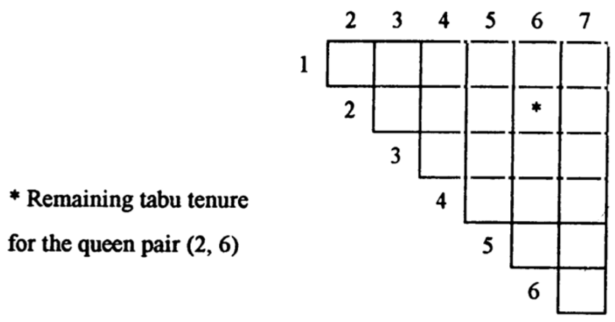

Figure 3. Data structure to represent taboo memory

Each cell of the structure in Figure 3 contains the number of iterations remaining until the corresponding queens are allowed to exchange positions again. Therefore, the cell (3, 5) has a value of zero, then queens 3 and 5 are free to exchange columns. On the other hand, cell (2, 4) has a value of 2, then queens 2 and 4 may not exchange column assignments for the next two iterations (i.e., a swap that exchanges these queens is classified taboo).

ASPIRATION CRITERION

To implement taboo restriction such as those based on queen pairs, an important exception must be taken into account. Taboo restrictions are not inviolable under all circumstances. When a taboo move would result in a solution better than any visited so far, its taboo classification may be overridden. A condition that allows such an override to occur is called an aspiration criterion.

BEST MOVE

One of the main differences between simulated annealing, and tabu search is the aggressive orientation of TS. Tabu search methods are designed to select at each step what is considered the best move available given the current search state.

The best move module is computationally more expensive than any other module in most TS procedures. In our example, finding the best swap move as defined above requires O(n^2) time. Therefore, it is extremely important that a computationally efficient procedure is used to evaluate the merit of each move:

⇒ For this reason, **a more simple approach** to generate the best swap move, is based on the **selection of a queen (variable) in each iteration, and only obtain the neighbors directly generated from this variable**. Next iteration will move to the following queen and so on.

5. Complete the following simulation

SIMULATION OF THE TABOO SEARCH: 5-Queens Problem

Iteration 0, Initial state:

5	4	3	1	2

Tabu structure

	2	3	4	5
1	0	0	0	0
2	▨	0	0	0
3	▨	▨	0	0
4	▨	▨	▨	0

Selection of a Variable (queen): 1
Candidates generated from Queen 1: all the exchanges between Queen 1 and the rest of Queens:

Selected Queen	Rest of Queens	Successors (Candidates)	Evaluation
1	2	4 5 3 1 2	4
1	3	3 4 5 1 2	4
1	**4**	**1 4 3 5 2**	**2**
1	5	2 4 3 1 5	2

Candidate selected: 1 4 3 5 2

Iteration 1
Current:

1	4	3	5	2

Exchange positions of Queens 1 and 4 (columns 5 and 1)
Tabu structure (updating)

	2	3	4	5
1			3	
2	▨			
3	▨	▨		
4	▨	▨	▨	

Candidates generated from Queen: 2

Selected Queen	Rest of Queens	Successors (Candidates)	Evaluation
2	1		
2	3		
2	4		
2	5		

Continue this simulation for at least 3 more iterations

2. SIMULATED ANNEALING

Simulate the search process using the same initial state and applying simulated annealing:

Initial State: [5   4   3   1   2]   Cost: 4

iterations	**T**	**Best until now/ Cost**	**Current /Cost**	**RANDOM Successor**	**Cost**	**Accepted?**
1	1000	[5 4 3 1 2]	4	[4 5 3 1 2] *Exchange (1,2)*	4	Yes
2	800					Yes