Gliwice, 27-01-2022 r.

Selected optimization methods

Documentation of the final project

Title: **Tabu search**

Author (Authors): Jakub Maćkowiak, Kacper Niemczynowski, Sandra Świeczak

Field of studies: Informatics (sem. V)

Theoretical description:

Tabu search is a metaheuristic search method using local search optimization methods. Local (neighbourhood) searches take a potential solution to a problem and check its immediate neighbours to find better potential solution. But local search methods tend to get stuck in regions where many solutions are equally fit.

Tabu search uses a local search procedure to iteratively move from one potential solution x to an improved solution x' in the neighbourhood of x, until some stopping criterion has been satisfied (generally, an attempt limit or a score threshold). Local search procedures often become stuck in poor-scoring areas or areas where scores are very similar. In order to avoid these pitfalls and explore regions of the search space that would be left unexplored by other local search procedures, tabu search carefully explores the neighbourhood of each solution as the search progresses. The solutions admitted to the new neighbourhood, $N^*(x)$, are determined by memory structures. Using these memory structures, the search progresses by iteratively moving from the current solution x to an improved solution x' in $N^*(x)$.

The implementation of tabu search stores the visited solutions or provided sets of rules. If a potential solution has been previously visited within a short-term period or if it has broken a rule, it is marked as "**tabu**" so that the algorithm does not inspect that possibility again.

Tabu restrictions can be not always inviolable. When a move (candidate solution) from a tabu list is better than any visited so far, its tabu classification may be ignored. A condition that allows such a move is called an **aspiration criterion**.

Tabu Search is a method in which successive solutions belonging to the neighbourhood of the current solution are browsed. So, it is possible that certain areas of the solution space of X never get checked. A **diversification strategy** is a procedure that allows to view different areas of solution space. If certain conditions are met, for example by a pair of iterations no better solution has been found, or it is not possible to find a solution outside the tabu list, diversification allows to generate a new starting point. Information (such as a tabu list) gathered from previous searches is kept in memory.

Exemplary calculations:

Let us consider the problem of finding an optimal permutation of 7 numbers. The order in which these numbers are arranged determine the overall sum assigned to each permutation. The problem is to find the order which maximizes the sum.

The permutation can be changed by swapping to numbers. Each swap is changing the objective function value with the value shown in the second column of top candidates' table. In the example the neighbourhood is assumed as list of permutations which have only two different numbers' position and the length of tabu list is assumed as 3.

Let us take a permutation **2573461** as a starting current solution. The value of this permutation is **10**.

Tabu list

	2	3	4	5	6	7
1						
2						
3						
4						
5						
6						

Top candidates

5,4	6	best
7,4	4	
3,6	2	
2,3	0	
4,1	-1	

Current solution

2	***5***	***7***	***3***	***4***	***6***	***1***

In the first iteration the best candidate solution is to swap **5** and **4**, because the value increases to **16**. Swap is added to the tabu list.

Tabu list

	2	3	4	5	6	7
1						
2						
3						
4	3					
5						
6						

Top candidates

3,1	2	best
2,3	1	
3,6	-1	
7,1	-2	
6,1	-4	

Current solution

2	***4***	***7***	***3***	***5***	***6***	***1***

In the second iteration the best current solution is to swap **3** and **1.** The value of the objective function increases to **18** and swap is added to the tabu list.

Tabu list

	2	3	4	5	6	7
1		3				
2						
3						
4		2				
5						
6						

Top candidates

1,3	-2
2,4	-4
7,6	-6
4,5	-7
5,3	-9

T best

best

Current solution

2	***4***	***7***	***1***	***5***	***6***	***3***

In the third iteration the best candidate solution is to swap **3** and **1** again, but this pair of numbers is on the tabu list. The next top candidate solution is to swap **2** and **4.** The value of the objective function decreases to **14** and swap is added to the tabu list.

Tabu list

	2	3	4	5	6	7
1		2				
2			3			
3						
4				1		
5						
6						

Top candidates

4,5	6	**T** best
5,3	2	
7,1	0	
1,3	-3	**T**
2,6	-6	

Current solution

4	***2***	***7***	***1***	***5***	***6***	***3***

In the fourth iteration the best candidate solution is to swap **4** and **5** again. The value of the objective function will increase to **20** which is the best value in all four iterations. The aspiration criterion allows such a move, so the swap is again added to the beginning of the tabu list (the oldest one is removed, because the list length is equal to 3). The value of the objective function increase to **20**.

The iteration will continue searching as long as specified stopping condition (for example time or step limit) is not satisfied.

Supporting computer program:

Input:

- *f* - given function
- *x1, x2, y1, y2* - restrictions of the given function
- *rad* - '' radius'' of searched area per point
- *sizePath* – number of steps for the algorithm to be taken
- *sizeAround* - number of randomly chosen points in the searched area

Output:

- *ultimateMinimum* (the lowest value found for all runs)
- *ListLinePlot*

Algorithm:

For k ≤ ***sizePath***:

random starting point with respect to the restrictions

For i ≤ ***sizeAround***:

If a point is not in the tabu list:

If point takes the lower value than the previous one:

point=***ActiveMinimum***

append point to the tabu list

If the lower value cannot be find in the given ***sizeAround*** area:

ActiveMinimum is the new ***UltimateMinimum***

choose a new random starting point

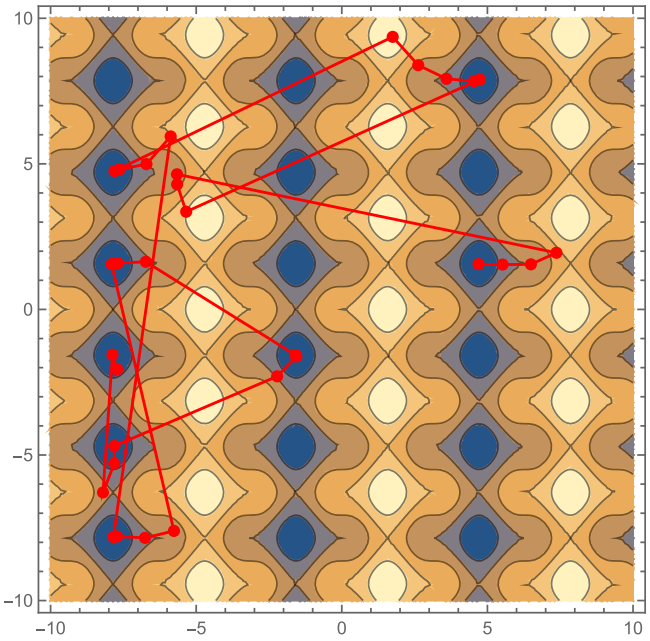
return *UltimateMinimum*

Result interpretation:

The result is an approximate minimum point and a graph of the distance travelled while selecting the next points (all the ActiveMinimum values taken during the computation).

Example of using the program:

Let us take a function $f = \sin^3 x + \cos^2 y, y \in [-10,10], x \in [-10,10]$. Radius of search area for any point must be less than 1. The algorithm will take 40 steps, in each it will randomly search for 500 points. The plot of the function and the path is shown in the Figure 1.



1: Plot of the given function with a path of reached points.

The clusters of points are the results of searching for better solutions than the previous ones in a specific area, which were taken into the tabu list. All the long lines between the points are the cases when the diversification was used, when the algorithm was no longer able to find a better point in the searched area.

The approximate minimum point of the given function is a point $\{-1.58501, -1.57002\}$ and equals -0.999696. Of course, there exist more of local minimum of this specific function which are the same as the mentioned one, but the tabu search algorithm was searching another better solution after finding the previous one. The best point is only the best heuristic approximation of the solution.

Enclosures:

Maćkowiak_Niemczynowski_Świeczak.nb