UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Engineering, Built Environment and IT
Department of Computer Science

COS 314

Assignment 1 - Search

Due: 06 April 2023

Question (30 Marks)

1. **Instructions**

    1. A zipped folder containing the data for this assignment is attached.
    2. Only Java or C++ may be used to complete this assignment.
    3. The use of libriaries is strictly prohibited
    4. The code and jar files and report are to be uploaded in the form of a zipped folder. Additionally, read-me instructions are to be included.
    5. **NB: The report will not be marked without submission of the code.**
    6. Submission is through ClickUP

2. The one-dimensional bin packing problem is a classic optimization problem in which a set of items of different sizes must be packed into a minimum number of bins, each with a fixed capacity. The goal is to minimize the number of bins used while ensuring that no bin exceeds its capacity. In mathematical terms, the problem can be formulated as follows: Given a set of items with sizes a_1, a_2, ..., a_n, where each item has a positive size, and unlimited number of bins with a fixed capacity C, the goal is to find the minimum number of bins required to pack all the items such that the total size of the items in each bin does not exceed C. The problem is known to be NP-hard, which means that finding an optimal solution for large instances is difficult and time-consuming. Therefore, various approximation algorithms and heuristics have been developed to solve the problem efficiently in practice. In this assignment we evaluate the effectiveness of single point searches namely, Iterated Local Search and Tabu Search to solve the problem.
You are given 5 datasets each containing a number of problem instances obtained from the 1-dimensional bin packing problem domain. Perform the following tasks.

    1. Develop a program that uses the Iterated Local Search to solve all the instances of the 1-dimensional bin problem contained in each dataset. (9 marks)
    2. Develop a program that uses the Tabu Search Algorithm to solve all the instances of the 1-dimensional bin problem contained in each dataset. (9 marks)
    3. Submit a documented report (PDF) that contains the following:
        - A technical specification of the programs in 1 and 2 in the form of an experimental set-up including a detailed description of the algorithm configuration.(3 marks)
        - Presentation of results as specified in Tables 1 and 2 (4 marks)
        - Discussion and conclusion (critical comparative analysis of the results i.e of the two algorithms on the bin packing problem). (4 marks)

Table 1: Sample Table showing the number of problem instances, for each of the data set categories, which were solved to optimality or near-optimality (one bin from the optimum), for ILS and Tabu Search. The best performing algorithm should be shown in bold.

Problem Set	ILS			Tabu		
	Opt.	Opt.-1	Sum	Opt.	Opt.-1	Sum
Falkenauer_T (80)	40	40	80	4	38	80
Falkenauer_U (80)	0	80	80	18	25	43
Scholl_1 (720)	0	0	0	0	0	0
Scholl_2 (480)	0	0	0	0	0	0
Scholl_3 (10)	0	0	0	0	0	0
Schwerin_1 (100)	0	0	0	0	0	0
Schwerin_2 (100)	0	0	0	0	0	0
Hard28 (28)	0	0	0	0	0	0
Wäscher (17)	0	0	0	0	0	0
Total (1615)	0	0	0	0	0	0

Table 2: Sample Table showing the average runtimes (in seconds) averaged across each dataset as well as all problem instances for ILS and Tabu Search).

Problem Set	ILS	Tabu
Falkenauer_T	0.00	0.00
Falkenauer_U	0.00	0.00
Scholl_1	0.00	0.00
Scholl_2	0.00	0.00
Scholl_3	0.00	0.00
Schwerin_1	0.00	0.00
Schwerin_2	0.00	0.0
Hard28	0.00	0.0
Wäscher	0.00	0.0
Total	0.00	0.00

Hints:

To apply **Tabu search** to 1-dimensional bin packing, you need to define a set of candidate solutions that represent different ways of packing the items into the bins. One possible approach is to use a greedy algorithm to pack the items into the bins, and then use Tabu search to refine the solution by exploring the neighborhood of the initial solution. The neighborhood of a solution can be defined as the set of all possible solutions that can be obtained by making a single modification to the initial solution, such as swapping two items between two bins or moving an item from one bin to another. The Tabu list can be used to keep track of the recent moves and prevent the algorithm from making the same move twice.

In the context of 1-dimensional bin packing, you can use **Iterated Local Search** by first generating an initial solution, for example, using a simple heuristic such as the first-fit or best-fit algorithm. You can then apply a perturbation operator to the solution, for example, by randomly removing items from the bins and then reinserting them in a different order. After the perturbation, you can use local search to optimize the solution by making small local moves, such as swapping two items between bins or shifting an item within a bin. The process of perturbation and local search is repeated for a fixed number of iterations or until a stopping criterion is met.