COS 314 Assignment 1
Keelan Matthews – u21549967

Technical Specification

Objective
The aim of this assignment is to implement the Iterated Local Search and Tabu Search algorithms and evaluate their performance on the given datasets. The objective is to optimize these algorithms to obtain as many optimal or near-optimal solutions as possible, while minimizing the computation time.

Hardware Used
The assignment was completed on an Asus Vivobook Pro 17 laptop with the following specifications:

- CPU: Intel Core i7-8550U, with a clock speed of up to 4.0 GHz
- RAM: 16GB
- Operating System: Windows 11
- Development Environment: Visual Studio Code

Experimental Design
At the start, the initial solution was computed using the first-fit heuristic, and a perturbation approach was employed by removing a value from a random bin and trying to place it in another random bin. However, this method did not produce desirable results, as all solutions were at least 2 units away from the known optimum, and in most cases, the difference was even greater. The first-fit and random bin placement approach were utilized for the entire program. Later, it was replaced by the best-fit heuristic, which resulted in the final algorithms analyzed below.

Iterated Local Search Algorithm
The Iterated Local Search Algorithm adopts the best-fit heuristic throughout its lifespan. The algorithm starts with the generation of the initial solution, which involves sorting the values in non-increasing order and packing them into the bins via the best-fit heuristic. Then, for a specified number of iterations, a new solution is created by perturbing the previous solution. Half of the values from a bin are removed and placed in new bins, one by one, using the best-fit heuristic.

Following this, a local search is executed on the new solution by removing items from bins and placing them into a new bin if it fits. Finally, the bin count of the new solution is compared to the bin count of the best solution, and it is updated if the bin count of the new solution is smaller than that of the best solution.

Tabu Search Algorithm

The Tabu Search Algorithm uses the same perturbation and initial solution algorithms as the Iterated Local Search Algorithm to ensure that the search space is consistent and comparable between the two algorithms. Unlike the Iterated Local Search Algorithm, the Tabu Search Algorithm does not use a local search algorithm after perturbation. Instead, it stores the perturbed solution in a table and only considers solutions that are not already in the table, as the ones already stored are considered taboo since they have already been visited.

In the Tabu Search Algorithm, the fitness value of the bin count is compared between the new solution and the best solution, but only if the new solution is not already in the table. If the new solution is smaller than the best solution, it is updated. However, if the new solution is in the table, the algorithm skips it and continues for a specified number of iterations.

Parameters

Both algorithms use the parameters MAX_CAPACITY and MAX_ITERATIONS. MAX_CAPACITY is obtained from the second line of each text file and represents the maximum capacity of the bins. MAX_ITERATIONS is obtained from the first line of each text file and determines the number of perturbation and comparison processes executed. The specific values of these parameters varied from 10 to 1000 depending on the dataset being used.

Additionally, the MAX_TABU_SIZE parameter is used in the Tabu Search algorithm to determine the size of the Tabu List. This parameter is set equal to the size of the dataset.

Evaluation Metrics

Table 1 Optimal values computed

Problem Set	ILS			Tabu		
	Opt.	**Opt.-1**	**Sum**	Opt.	Opt.-1	Sum
Falkenauer_T (80)	**0**	**20**	**80**	0	20	80
Falkenauer_U (80)	**6**	**26**	**80**	6	26	80
Scholl_1 (720)	**549**	**112**	**720**	550	111	720
Scholl_2 (480)	**316**	**119**	**480**	313	122	480
Scholl_3 (10)	**0**	**8**	**10**	0	5	10
Schwerin_1 (100)	**15**	**85**	**100**	12	88	100
Schwerin_2 (100)	**41**	**59**	**100**	42	58	100
Hard28 (28)	**5**	**23**	**28**	5	23	28
Waescher (17)	**2**	**15**	**17**	2	15	17
Total (1615)	**934**	**467**	**1615**	930	468	1615

Table 2 Average times per dataset

Problem Set	Average times (s)	
	ILS	**Tabu**
Falkenauer_T	0.064	**0.058**
Falkenauer_U	0.043	**0.041**
Scholl_1	0.013	**0.011**
Scholl_2	0.032	**0.027**
Scholl_3	2.906	**2.535**
Schwerin_1	0.007	**0.005**
Schwerin_2	0.009	**0.006**
Hard28	0.043	**0.039**
Waescher	0.106	**0.062**
Total (s)	3.223	**2.784**

Results

After analyzing the results obtained from both the Iterated Local Search and Tabu Search algorithms, it's evident that each algorithm has its own unique strengths and weaknesses.

In Table 1, we observe that Iterated Local Search generated more optimal solutions than Tabu Search across all datasets. Although both algorithms produced a similar number of near-optimal solutions, Iterated Local Search computed four more optimal solutions than Tabu Search. However, it's worth noting that the difference between the two algorithms is relatively small, and therefore, the choice of the preferred algorithm should also consider other factors such as computational resources and time.

In Table 2, we can see that Tabu Search performed much more efficiently than Iterated Local Search, with a shorter total execution time. This is a critical factor to consider in applications where time is important. Moreover, while Tabu Search generated fewer optimal solutions compared to Iterated Local Search, the difference is still relatively small, and therefore, the Tabu Search algorithm remains a viable option in situations where efficiency and speed are critical factors.

After evaluating both the best-fit and first-fit heuristics, it was found that the former was more efficient. This was attributed to its ability to move each value with more thoughtfulness and strategy, rather than random movement as in the case of first-fit. As a result, best-fit resulted in better placement overall, leading to more effective sorting of each bin when compared to the first-fit heuristic.

Conclusion

In conclusion, the choice of the algorithm to use should depend on the specific requirements of the problem being solved. If generating optimal solutions is the top priority, then Iterated Local Search is the preferred algorithm. However, if computational efficiency and time are critical factors, then Tabu Search is the better option.

Charted Data