2.3.1 Technical specification

To compare the performance of Local Iterated Search and Tabu Search algorithms on the one bin packing problem, we used the same single problem set, i.e Schwerin_1. We executed each algorithm for a set amount of trials, comparing the optimal values of each text file contained within the problem set. We used the same configuration for both algorithms initially:

- An initial solution was generated using the first fit heuristic, placing the item in the first available bin without exceeding the given capacity, otherwise a new bin would've been created.
- Neighborhood: We used the Swap neighborhood, where we swapped one item from one bin to another in each iteration.
- Stopping Criteria: We set a maximum number of iterations to 1000.

PROGRAM 1 → LOCAL ITERATED SEARCH

Algorithm configuration:

Iterated Local Search:

1. Generate and initial solution using constructive heuristic or random method.

   Most likely will use first fit in order go generate our initial solution
2. Apply local search algorithm to improve the solution
3. Obtain local optimum
4. Perturb the local optimum to obtain a new solution

   Methods like shaking, randomization and mutation.
5. Apply the local search algorithm to the new solution
6. If new solution better than current, update current.

For the initial solution the first fit heuristic was used.
The first fit heuristic was used instead of the best fit heuristic as the asymptotic complexity is smaller in comparison. O(nlogn) instead of a O(n^2) since we don't need to compare all the items in the bin and move items in the bins. This leads to faster convergence as we iterate faster.

In order to find the local optima items were moved between bins until we cannot move any more items, or make any other possible better moves, i.e. we get stuck, although as solution might still exist on an alternative path.

Perturbing the solution allowed diversity by removing items from bins and reintroducing them in a different order. We did this a random number of time to ensure that we can escape from the local optima and that we get a unique solution.

Similar to searching for the local optima. Now we are just using the perturbed solution. We continue until no further improvements can be found or max number of iterations.

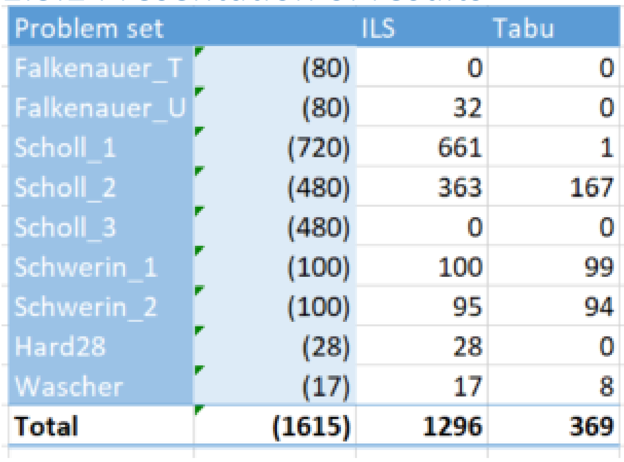
PROGRAM 2 → TABU SEARCH

Algorithm configuration:

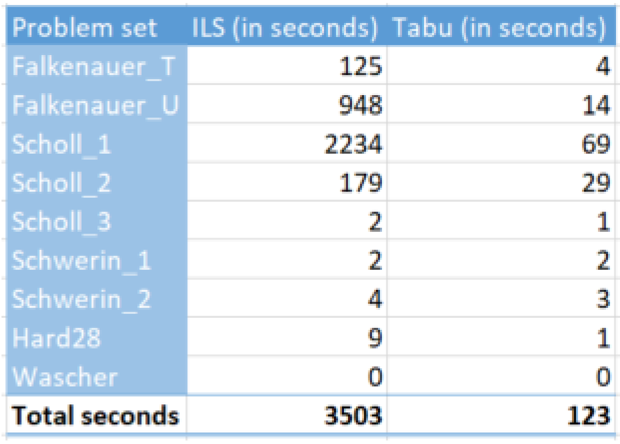
```
Set x=x0 (initial candidate solution)
    Set length(L) = z (maximum tabu list length)
    Set L={} (initialise the tabu list)
   repeat
    generate a random neighbor x'
    if x' not contained within L
       if length(L)>z then
          remove oldest solution from L
          set x' as an element of L
       end;
       end;
       if x' < x then
          x = x'
          end;
          until stopping criteria is met
          return x;
```

First fit heuristic was used to in order to obtain an initial candidate solution.
We then iterate the max number of times, in this case 1000, and obtain a neighbouring solution by swapping elements that have been randomly selected from 2 randomly selected bins.
We then check in our list whether the current solution has been visited before, if not, we add it to the list and determine whether we have a better solution.

2.3.2 Presentation of results



Problem set		ILS	Tabu
Falkenauer_T	(80)	0	0
Falkenauer_U	(80)	32	0
Scholl_1	(720)	661	1
Scholl_2	(480)	363	167
Scholl_3	(480)	0	0
Schwerin_1	(100)	100	99
Schwerin_2	(100)	95	94
Hard28	(28)	28	0
Wascher	(17)	17	8
Total	**(1615)**	**1296**	**369**



Problem set	ILS (in seconds)	Tabu (in seconds)
Falkenauer_T	125	4
Falkenauer_U	948	14
Scholl_1	2234	69
Scholl_2	179	29
Scholl_3	2	1
Schwerin_1	2	2
Schwerin_2	4	3
Hard28	9	1
Wascher	0	0
Total seconds	**3503**	**123**

2.3.3 Discussion and Conclusion

As evident from the tables above, Local Iterated Search has a calculated success optimization rate of 80% contrasting to Tabu Search which only had 23% success in optimizing the given data set.

In regards to the speed at which the algorithm optimizes the data set for the 1 bin packaging problem, we can clearly see that Tabu executes much faster in comparison to the Local Iterated Search. As Tabu search uses memory in order to avoid cycles or revisiting previous solutions, and not reaching or get stuck in a local optima.
Due to the above mentioned attribute of the Tabu Search Algorithm, the algorithm tends to explore our state space, accepting possible poorer solutions, worsening our bin optimization solutions.

However, Iterated Local Search, i.e. ILS, uses a better or equal approach when accepting possible solutions. This increases the time of calculation in order to try and obtain an optimal value as the function iterates through the local optima finding function, perturbing function and a local search function for our max number of iterations. The stopping criteria for this program was set as either improvement of the solution or a timeout in terms of the max number of iterations.

In conclusion, as the goal of the 1 bin packaging problem was to optimize the number of items in the least amount of bins without exceeding the capacity, ILS is the better algorithm, as it has a better optimization success rate.