TAWANDA-CODER

Github

BACKGROUND

In Assignment 1 we were give 5 datasets each containing several problem instances obtained from the 1-dimensional bin packing problem domain. We were tasked with creating two programs that use Iterative Local Search and Tabu Search Algorithm respectively to solve all the instances from the 1-dimensional bin problem contained in each dataset.

SYSTEM ENVIROMENT:

The performance of a search algorithm such as tabu search can depend on a variety of factors, including the hardware and software environment of the computer on which it is executed.

OS Name *Microsoft Windows 10 Home Single Language*
System Model *HP Pavilion Aero Laptop 13-be0xxx*
System Type *x64-based PC*
Processor *AMD Ryzen 7 5800U with Radeon Graphics, 1901 Mhz, 8 Core(s), 16 Logical Processor(s)*
Installed Physical Memory (RAM) *8,00 GB*
Physical Disk Type *Solid State Drive 512.00 GB*

PROGRAM 1: ILS

My first Step was to import the optimum values and datasets into my program. I used the *java.io.File* and *java.util.Scanner* libraries to handle the text files and deal with the various directories the different problem instances are located. *ReadOptimumValues.java* and *ReadDataSetValues.java* deal with the importation of data into the program. I created the *OptimumValueClass* and the *items* classed represented in *OptimumValueClass.java* and *items.java* respectively to better represent and deal with data.

After importing all the data into my program, I started with the ILS function located in *ProblemInstance.java*.I followed the following steps:

1. I randomly Initialized the solution/local optimum using *Next Fit Algorithm*. Next Fit algorithm checks to see if the current item fits in the current bin , if so, then place it there, otherwise start a new bin.I chose the Next Fit Algorithm because of its simplicity.It does not require complex data structures making it good for systems with limited computational resources.
2. I iterate for 2 500 times with the stopping condition that the local optimum is equal to the given know optimum. In each iteration I perturb to get a new solution. I perturb the solution by swapping two random items. I then use the next fit algorithm to count the number of bins. If the new solution is better than the current solution , I make the new solution , my current solution.
3. I then return the best solution found during the iterations as my local optimum.

Problem Set	ILS (**2 500 iterations**)					
	Opt.	Opt. - 1	Opt. - 2	Opt. -3	Opt. – n > 3	Total
Falkenauer_T (80)	0	0	1	19	60	80

Falkenauer_U (80)	0	0	0	0	80	80
Scholl_1 (720)	0	4	29	68	619	720
Scholl_2 (480)	119	108	53	44	156	480
Scholl_3 (10)	0	0	0	0	10	10
Schwerin_1 (100)	0	99	1	0	0	100
Schwerin_2 (100)	0	93	7	0	0	100
Hard28 (28)	0	0	0	0	28	28
Wäscher (17)	0	12	1	2	2	17
Total	119	316	92	133	955	1615

PROGRAM 2: TABU

My first Step was to import the optimum values and datasets into my program. I used the *java.io.File* and *java.util.Scanner* libraries to handle the text files and deal with the various directories the different problem instances are located. *ReadOptimumValues.java* and *ReadDataSetValues.java* deal with the importation of data into the program. I created the *OptimumValueClass* and the *items* classed represented in *OptimumValueClass.java* and *items.java* respectively to better represent and deal with data.

After importing all the data into my program, I started with the ILS function located in *ProblemInstance.java*.I followed the following steps:

1. I randomly Initialized the solution/local optimum using *Next Fit Algorithm*. Next Fit algorithm checks to see if the current item fits in the current bin , if so, then place it there, otherwise start a new bin.I chose the Next Fit Algorithm because of its simplicity. It does not require complex data structures making it good for systems with limited computational resources.
2. I iterate for 2 500 times with the stopping condition that the local optimum is equal to the given know optimum. In each iteration I perturb to get a new solution. I perturb the items list by swapping two random items.
3. I created a tabu list using a linked list data structure. I the tabu list uses the principle of FIFO (First In First Out).I gave the tabu list a max size of 10.If the size becomes larger than I remove the first element in the tabu list as its been there the longest. If we meet a new unique solution which currently doesn't exists, we add it to the back of the linked list. When we get a new solution, we check whether the solution is already on the tabu list. If the solution exists on the tabu list I deem is "tabu"/forbidden, and the algorithm, skips it and moves to the next solution.
4. If the solution is not tabu, I then use the next fit algorithm to count the number of bins. If the new solution is better than the current solution , I make the new solution , my current solution.
5. I then return the best solution found during the iterations as my local optimum.

Problem Set	TABU (**2 500 iterations**)					
	Opt.	Opt. - 1	Opt. - 2	Opt. -3	Opt. – n > 3	Total
Falkenauer_T (80)	0	0		0	80	80
Falkenauer_U (80)	0	0	0	0	80	80

Scholl_1 (720)	0	0	0	2	718	720
Scholl_2 (480)	61	103	54	43	219	480
Scholl_3 (10)	0	0	0	0	10	10
Schwerin_1 (100)	0	98	2	0	0	100
Schwerin_2 (100)	0	91	9	0	0	100
Hard28 (28)	0	0	0	0	28	28
Wäscher (17)	0	9	3	1	4	17
Total	61	301	68	46	1139	1615

TIMES PROGRAM 1: ILS

Problem Set	Time (nano seconds)
Falkenauer_T (80)	1532821
Falkenauer_U (80)	1532146
Scholl_1 (720)	1094963
Scholl_2 (480)	1093142
Scholl_3 (10)	1088860
Schwerin_1 (100)	878099
Schwerin_2 (100)	830749
Hard28 (28)	1174439
Wäscher (17)	1077782
Average Times	1144778

TIMES PROGRAM 2: TABU

Problem Set	Time (nano seconds)
Falkenauer_T (80)	1139567
Falkenauer_U (80)	913121
Scholl_1 (720)	868471
Scholl_2 (480)	710035
Scholl_3 (10)	779650
Schwerin_1 (100)	836076
Schwerin_2 (100)	839056
Hard28 (28)	784392
Wäscher (17)	743629
Average Times	845999

COMPARATIVE ANALYSIS

When comparing the results we can see that Tabu search is faster that Iterative Local Search , in most cases in the context of this assignment *(average times ILS : 1144778 nanoseconds vs TABU : 845999 nanoseconds).*The reason why Tabu might be faster than ILS in most cases is because it utilizes a tabu list to keep track of previously visited solutions in my case the last 10 and this restricts the search to unexplored areas in the solution space. By doing this we avoid getting stuck in the local optima and explore a broader range of solutions.

In contrast, Iterative Local Search explores the solution space by iteratively improving the current solution. It does not use a tabu list, so it may revisit previously explored solutions, including suboptimal ones. As a result, the algorithm may get stuck in local optima and require more iterations to find a globally optimal solution.