Optimization Methods and Algorithms
Report

Professors: R. Tadei, D. Manerba. Academic year: 2018/2019
Report: MZ, group 04
Students:
• Marco Florian, s247030 • Giuseppe Ognibene, s257957 • Alessandro Pagano, s260005
• Hamza Rhaouati, s216550 • Leonardo Tolomei, s222718 • Simone Valvo, s245551

1 Introduction

For the program we used 3 threads. The first one is used to count the time of execution, to read the instance specifications and to kill the other two threads. The others two calculate the best solution. One of them implements a metaheuristic based on the Tabu Search with Multi-start and the other one based on the Genetic Algorithm hybridized with Tabu Search.

2 Metaheuristic based on the Tabu Search with Multi-start

Legend
$\tilde{x}$ = current solution. $N_{\tilde{x}} = \tilde{x}$'s neighbour. $\widehat{x}$ = best solution in the neighborhood.

Representation
Vettx: it is the vector of queries with its configuration. This data structure makes impossible infeasibility caused by multiple configuration used for the same query.
gain: it is the net gain.
mem: it is the memory used.
Zi: it is the vector of active indexes.
Yc: it is the vector of active configurations.
Fc: it is the vector that shows the fixed cost added by the corresponding configuration.
actualMemc: it is the vector of memory occupations of active configurations.
eventualMemc: it is the vector that shows the memory added by the corresponding configuration.
actualGainc: it is the vector of current gains of the active configurations.
eventualGainc: it is the vector that shows the net gain, added by the corresponding configuration.

Initial solution - TSinit
We build an initial feasible solution and with a non-negative obj function, we adopt a greedy strategy trying to activate as many configurations as possible while respecting the memory constraint.

Neighborhood size - numN
We chose the value of 25 as neighborhood dimension.

Local search operator
If $\tilde{x}$ is feasible we want to increase the obj function, so we look for the 5 best and the 5 worst configurations in terms of gain, then we try to exchange all the best with all the worst ones. If $\tilde{x}$ is feasible, because it does not respect the constraint on memory, we adopt a similar procedure: we look for the 5 best and the 5 worst configurations in terms of memory, then we try to exchange all the best with all the worst ones. If $\widehat{x}$ is infeasible for 20 consecutive iterations, the configurations are randomly deactivated until it returns feasible. If we do not find new best solutions for 50 consecutive iterations, we start from another solution (TSinit). If $\widehat{x}$ is feasible for 5 iterations, we add a configuration that serves a query without configuration.

Neighborhood exploration
Our new $\tilde{x}$ will be the one with the worst-best config exchange, allowed by the tabu list, that returns a greater obj function.

Tabu List
Fifo, with tabu tenure equal to 7.
Tabu move: the couple of configurations swapped, avoiding both repetition and reversion of the move.

3 Metaheuristic based on the Genetic Algorithm hybridized with Tabu Search

Representation
Vettx: it is the vector of queries with its configuration. This data structure makes impossible infeasibility caused by multiple configuration used for the same query.

vettC: it is the vector that for every configuration assigned to a query (vettX), returns how much is compatible with the active indexes.
vettR: it is the vector that for every configuration assigned to a query (vettX), returns the ratio gain/(mem-fixed cost).

Fitness - calculateFitness
Calculated as chromosome gain - (available memory - chromosome memory).

Parents selection - selectParents
We select half the population with a probability based on fitness.

Substitution - sobstitute
Generated children are inserted in the population with a probability based on fitness.

Population replacement - sobstitute
Generated children are inserted in the population with a probability based on fitness.

Crossover with 2 crossing point - crossover
To decide the two crossing points we start from a random position inside the two vectors, we widen step by step towards the two ends, checking at every step if for one of the two chromosomes the summation of the vettR for that portion exceeds 2 times that of the other. If the value r of the right gene (considering the current window) in the first chromosome is greater than that of the gene to the right of the second chromosome, we widen to the left; otherwise we widen to the left. The crossover is applied to each pair of consecutive parents, among those chosen by the parent selection.

Mutation - mutation
We consider all the queries in ascending order of compatibility (vettC), first those already served by a configuration and then those without. For each query we look for the configuration with positive gain and with the maximum compatibility (vettC), if this compatibility exceeds a threshold we assign this new configuration to the query (updating the vettX accordingly). At the end we scroll all the queries to assign to each one the active configuration that has a major or positive (in the case of those not served by any configuration) obj function. Mutation is applied to the entire population each time. It is performed once immediately after initialization and, thereafter, whenever no crossover occurs between the selected parents, to differentiate the population.

Tabu Search - TS
It is a translation of the TS from a stand-alone algorithm to function. It is executed on every child generated by the crossover, to improve it before the population replacement.

4 Conclusions

In general we obtained an average difference of 4.6% from given optimal solution. The solutions that most deviate from the optimal value are probably due to a poor exploration of the solution space, we tried to work around with the Tabu Search, but probably it would be necessary to improve the function that initializes the solution. Even the hybrid, based on the Genetic Algorithm, which should move more randomly within the solution space, does not succeed. Analyzing the execution of the single metaheuristics, we have noticed that they behave differently according to the requests. In some cases, the Tabu Search was the first to converge to better solutions, while the hybrid started to produce improved results only later. In other cases the opposite happened. Therefore, we decided to combine both metaheuristics in a single program.