TRAVELING THIEF PROBLEM

SPLITTING THE PROBLEM: TSP SUBPROBLEM

Trivial tour

- Return the trivial tour $1, 2, ..., n$ where n is the number of vertices.

Nearest neighbor

- For every city, take the nearest neighbor that has not been visited and adds it to the tour
- Since we have a complete graph, this always leads to a solution
- Usually gives good solutions for the TSP.

2-opt solution

- Works only for metric graphs (as our instances).
- Works only for metric graphs (as our instances).
- Double every edge and generate a Eulerian path.
- Whenever we visit a city that has already been visited, shortcut the path (triangular inequality).

SPLITTING THE PROBLEM: PACKING PLAN

Constructive heuristic

- Generate scores $(score, u)$ for each item based on the potential profit.
- Choose the best possible items for this metric that don't overflow the maximal weight capacity.

Random Local Search

Neighborhood

Two packing plans are neighbors if only if the packing status of only one item is different

- Start with the empty backpack.
- Invert the packing status for a random item (take a neighbor of the current solution).
- If this represents an improvement and does not exceed the weight of the knapsack, take this as the solution.
- Repeat the process until no improvement is found for N interactions.

Evolutionary algorithm

- Start with an empty backpack.
- Invert the packing status of each item with probability $1/m$ (m is the quantity of items).
- If this represents an improvement and does not exceed the weight of the knapsack, take this as the solution.
- Repeat the process until no improvement is found for N interactions

Metropolis algorithm

- Execute a random local search but may take packing plans worst than the current solution.
- Return the best solution seen during the whole execution of the algorithm.

RESULTS

Results: Uncorrelated items

Table: Objective value of different heuristics for a given set of instances with uncorrelated items. The TSP subproblem was solved using the nearest neighbor heuristic

	a280	**d1655**	**eil51**	**ul1060**
SH	-4595	-136782	-6757	-167166
RLS	-2444	-14928	-1858	-325761
EA	42965	690949	1843	300736
Metropolis	2168	-45453	1317	-317541

Results: Uncorrelated items with similar weights

Table: Objective value of different heuristics for a given set of instances with uncorrelated items but with similar weights. The TSP subproblem was solved using the nearest neighbor heuristic

	a280	**d1655**	**eil51**	**ul1060**
SH	-161393	-2763579	-39770	-1446970
RLS	598	-218440	-1354	-115155
EA	78810	580623	746	446964
Metropolis	31771	-39170	-1255	-22768

Results: Bounded strongly correlated items

Table: Objective value of different heuristics for a given set of instances with Bounded strongly correlated items. The TSP subproblem was solved using the nearest neighbor heuristic

	a280	**d1655**	**eil51**	**ul1060**
SH	-256694	-1963700	-48846	-2241064
RLS	31394	-29619	-3313	-329533
EA	73465	1212683	-1922	408846
Metropolis	22808	-24877	-8312	-137528

Analyse

- Constructive heuristic takes a lot of items increasing the time to finish the tour. This affects negatively its objective value.
- RLS heuristic gives a reasonably good solution in a short period of time.
- EA heuristic gives a good solution but takes more time to do it.
- Metropolis should be better than the others iterative heuristics since we allow it to get out of local maximum. However, the problem is how choose a good value of T.

Possible improvements

Possible improvements

- Start the randomized algorithms with a better departing point, maybe, our constructive heuristic.
- Use more powerful computers which would allow more iterations for the randomized algorithms.
- Study for an optimal "temperature" in the metropolis algorithm.
- Try to find a direct algorithm for solving both problems at once.