Heuristic Optimization Techniques

WS 2017 Assignment II Report

Peter Rjabcsenko 01228563 & Roland Pajuste 01428820
(Group 3)

Description of the implemented algorithms:

Ant Colony Optimization:

- initialize the spine order in ascending vertex order (i.e. 1,2,3…)
- generate an initial population of empty solutions at every time step
- for each time step, each ant generates a solution in the following way:
  it traverses the edges in a particular order and probabilistically decides to which page to assign an edge based on the inverse cost of adding said edge to a page ($\frac{1}{cost+0.1}$), (where cost is the increase of the objective function value) and pheromone value
- the pheromone matrix contains pheromone values for assigning each edge to a particular page
- the pheromone value that an ant leaves behind after completing a tour is the inverse of the cost of the generated solution ($\frac{1}{tourCost}$)
- when an ant decides on assigning an edge to a page, only the cost of adding this edge to that page is calculated and then added to the total cost, no recalculation of crossings (simple form of incremental evaluation)
- after the last time step, the solution with the best objective function value is selected from the population and returned by the algorithm

Local Search:

- one-edge-neighbourhood: 1 edge is assigned to a different page.
- random step function: pick a random edge and assign it to a random page, if this results in a lower *f(x)*, keep this solution (repeat this step # edge times).
- For the step function we use incremental evaluation in the following form: when an edge is selected for re-assigning, we calculate the number of crossings that the edge was involved in on its original page and the number of crossings it would create on its new page, we add the difference to the current *f(x)*.

Hybrid Method:

- aforementioned ACO and Local Search used in a sequential order.
- feed the best solution of the ACO population after termination to the local search

Experimental setup:

- machine used: Lenovo laptop
  processor: 2.60GHz Intel Core i5-4210M
  RAM: 16GB
- we used an upper limit of 5 minutes runtime for each algorithm

Best objective values and runtimes:

Note: mean and std. dev. is calculated over 5 runs. Empty rows mean ACO timed out before producing a complete population at time step 1

Instance	Best objective value	First pop. avg.	Last pop. avg.	Runtime (sec.)	Mean	Std. dev.
01	13	26.8	17.7	0.14	15.6	1.743
02	72	104.3	94.5	0.14	85.6	9.6
03	136	221.6	149.2	0.43	149.8	10.36
04	115	144.4	118.4	0.23	117	1.26
05	91	137.2	105.8	0.4	99.2	5.23
06	1151	1458.2	1228	2.46	1231.8	58.7
07	22573	27169.2	23229	88.03	23257.4	689.8
08	9673	10644.4	9890.4	37.03	9832.8	98.9
09	4259	4924.6	4339.7	31.1	4406.4	89.7
10	7503	9467.9	7666.8	22.6	7938.8	245.2
11	--	--	--	--	--	--
12	166202	167054	166560	300(timeout)	166485.2	153.8
13	--	--	--	--	--	--
14	--	--	--	--	--	--
15	--	--	--	--	--	--

ACO

Instance	Best objective value	Runtime (sec.) minus ACO	Mean	Std. dev.
01	13	0.0.15	15.4	1.854
02	69	0.0	78.4	7.22
03	109	0.0	133.8	13.52
04	112	0.0	114.2	2.03
05	81	0.0	89	7.56
06	1038	0.03	1095.6	55
07	19707	0.187	20673.6	954.2
08	8176	0.14	8584.2	311
09	3807	0.09	3915	100.1
10	6738	0.06	7005	238.8
11	--	--	--	--
12	148367	3.1	149867.2	1890.6
13	--	--	--	--
14	--	--	--	--
15	--	--	--	--

Hybrid:

ACO specifics:

- structure for the construction graph: we define the construction graph in terms of edges and pages. Each edge represents a vertex in the construction graph. Upon reaching a vertex an ant must decide which direction to take (which page to assign to current edge) to reach the next vertex in the graph.
- pheromone model: upon finishing a solution each ant calculates its pheromone value, that being the inverse of the solution cost, and adds this value to each edge-page pair that it selected for its solution. The pheromone matrix is a combination of all edges and pages
- concept for heuristic information: upon reaching a vertex in the graph an ant calculates the cost of taking any of the directions to the next vertex (assigning current edge to each page) and uses this local information for decision making as well
- parameter values: 15 ants, 50 time steps, 0.00001 initial pheromone value, 1 pheromone weight, 1 distance weight, 0.35 pheromone evaporation weight (65% is left after evaporation)

Further notes:

- **Parameters:** we tried a varying number of ants and timesteps and observed that a greater number of timesteps influences population convergence nicer than a greater number of ants. On the other hand drastically different pheromone and distance weights seemed to worsen the population qualities so we ended up keeping them at 1 to 1 ratio. We decided evaporation rate to be 35% by trial and error.
- **Solution representation**: we store the spine order in a list (similarly to the way it was provided in the parser framework) and the edges/pages in an adjacency matrix, where 2 vertices that form an edge correspond to their assigned page value in the matrix (0,1,2,etc.) while vertices that do not form an edge are represented by the value -1 in the matrix. In hindsight it might have been a better choice to store the edges in an adjacency list to avoid traversing all the edgeless vertex pairs of the matrix
- **Spine order:** it would have been interesting to see how the best solutions would improve in general if we started from a different spine order with every run of the ACO. Since heuristic algorithms work best if run multiple times, we definitely think this would have yielded an improvement
- **Sequential execution for Hybrid:** we chose to execute ACO and Local Search in a sequential order because it the simplest way of hybridization in this case and also allows for an easy way of customization. One can see that with this approach its simple to swap Local Edge Random search for a Local Vertex First/Best search or even for an entirely different improvement heuristic
- **ACO + Local Edge Random Search:** we decided to use local search with a random step function because compared to ACO it is fast and often provides a significant improvement of the solution for a relatively low additional time cost. Running local search on multiple top solutions and selecting the best result could also have been a reasonable course of action. We initially wanted to integrate a local search with a 1-vertex neighbourhood as well to balance out the set spine order that we use for ACO but we were not able to overcome the problem of incremental evaluation for 1-vertex neighbourhood and recalculating the whole objective function from scratch is just not feasible