Mark Demore, 2d Lt

CSCE686 – Dr. Lamont

Homework 6

**A)**

**1.11:**

Input: A node subset X ⊂ V , a fleet K of vehicles and a capacity C ≥ 0.

Output: Routes {Ai ⊆ Vi : i ∈ K} for each vehicle with each route of capacity at most C.

Objective: maximize | ∪i∈K Ai ∩ X|, the number of nodes covered

Greedy Algorithm for CVRP:

input : A fleet K of vehicles, a subset X ⊂ V and a capacity C

output: A set H of routes with capacity at most C, one route for each vehicle

X0 ← X

for i ∈ K do

Ai = O(X0 ∩ Vi , C, i)

X0 ← X0\Ai

end

return H = {Ai : i ∈ K}

Some additional constraints for the CVRP problem would be: to minimize the distance each vehicle travels, each route needs to be completed in a given amount of time, and vehicles are picking up new deliveries along the route.

**1.17:**

The objective function for minimizing total distance for all vehicles in the VRP is difficult, because when a customer is moved between vehicles, both routes need recalculated entirely. Which, in itself, is difficult, as it is unlikely the customer will be at the beginning or end of the old or new route.

**B)**

**i)** Let G = (V, A) be a graph where V = {1, …, n} is a set of vertices representing cities with the depot located at vertex 1, and A is the set of arcs. With every arc (i, j) i =/=j is associated a non-negative cost matrix C = (cij). Constraints: (i) each city in V\{1} is visited exactly once by exactly one vehicle; (ii) all vehicle routes start and end at the depot. Objective: minimize cost. In simple terms, there are a set of locations that need to be visited by a vehicle with a fixed cost for travel between locations, all vehicles start and end their route at the same location, the goal is to minimize cost while visiting each location. The complexity of the PD is NPC.

**ii) *algorithm domain requirements speciﬁcation form:***

*• name: A\* (Di, Do)*

*• domains: Di is set-of-candidates,*

*Do are sets of solutions (solution space of subsets)*

*• operations:*

*I(x); x in Di; x is a possible candidate from input set*

*O(x,z); x in Di, z in Do; z is an satisfying solution*

***algorithm domain design speciﬁcation form:***

*• name: A\* (Di, Do)*

*• domains: Di is set-of-candidates, Do are the sets of solutions, Dp is set of partial solutions (one vehicle’s route)*

*• imports: ADT set, list, queue, real/integer/character*

*• operations:*1

*I(x); x in Di*

*O(x,z); x in Di, z in Do;*

*“condition on z being a satisfying solution”*

*I’(x,y); x in Di, y in Dp; condition on y being a partial solution in Dp*

*Dp is the “open” list; Dc is the ”closed” list*

**–** *deﬁne state*

**– *next-state-generator***

*i)* ***selection*** *of a partial solution y in Dp based upon its superiority and put in Dc and delete from Dp*

*“based upon heuristic cost function”*

*ii)* ***Generation*** *of* all *next states xj of y*

**– *feasibility*** *(xj , y) − > boolean [if true union (xj , y) and put result* *in* *Dp]*

**– *solution*** *(y) − > boolean; z = y; delay termination and ﬁnd all*

*“optimal” solutions (if satisfying accept first solution)*

**–** *objective solution(Dp) − > “ordered set over Dp”*

**– *heuristics***  *come from problem domain insight:*

***-- Attempt use PD next state generator to reduce set-of candidates* *ASAP***

***-- Attempt to generate a combination once and only once in combinatorial problem domain***

***-- Attempt to generate early pruning condition simple solution check***

***algorithm domain intermediate speciﬁcation form: (iterative)***

*• Heuristics: distance to next point, distance back to depot*

*• Data structures: input – graph: set of nodes (locations), set of edge weight (cost between each location), output – list of sets (route for each vehicle)*

***algorithm domain function speciﬁcation form: (iterative)***

*•*  Function A\*(initial, Expand, Goal, Cost, Heuristic)

q <- New-Priority-Queue()

Insert(initial, q, Heuristic(initial))

**while** q is not empty

**do** current <- Extract-Min(q)

**if** Goal(current) then **return** solution

**for** each next in Expand(current)

**do** Insert(next, q, Cost(next) + Heuristic(next))

return failure

**iii)**

**iv)**

**v)**

**References:**

[1] Pisinger, D., & Røpke, S. (2007). A general heuristic for vehicle routing problems. Computers & Operations Research, 34(8), 2403-2435. <https://doi.org/doi:10.1016/j.cor.2005.09.012>

[2] Astar\_example.doc

[3] <http://www.optimization-online.org/DB_FILE/2018/06/6645.pdf>

[4] <https://en.wikipedia.org/wiki/Vehicle_routing_problem>

[5] Laporte, Gilbert. *The Vehicle Routing Problem: An overview of exact and approximate algorithms.* <https://d1wqtxts1xzle7.cloudfront.net/53945946/The_Vehicle_Routing_Problem_An_overview.pdf?1500738179=&response-content-disposition=inline%3B+filename%3DThe_Vehicle_Routing_Problem_An_overview.pdf&Expires=1591466025&Signature=ODZoPD1jXypmNQy~8bRYSMrYP4dYzUXyDcq~IFpSgj2BWyyntLeXr5ovUVed8dd~SlSUUauqV8CXDmbak-7V9T8kCPBSj6u-Nvf1F6dUh2TN54uSYNFvmLzs3eags13vDVps5QaJAgMozyN04q5EoJKPKHzr46Lxk7~2F7QINyK5Ta9HH5i3YSois5NfvOaa9bnjS~0UmVWgcTeL9taq2Ojc0d6hQAmU9fUdU0Ove~lEcZik0vqu3z1SyJodQOoTAuXNQxnrWmoIQtvVt-~3JIs0iGPE6Dq4dl6X0SLhnGe12mXzp9CGw5kjYHCYK2psLa5BOs6lVh4YDp6PSlcYGQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA>

[6] Lecture4\_gs\_bfs\_20.doc