Xuan Thao VRP Cheat Sheet

by txthao1998 via cheatography.com/147472/cs/32069/

Ortools

create dimension	**dimension_name** = 'Distance' routing.AddDimension(transit_callback_index, 0, # no slack 300_000, # vehicle maximum travel distance True, #start cumul to zero dimension_name)
activate dimension	**distance_dimension** = routing.**GetDimensionOrDie**(dimension_name)
add custom constraint to solver	routing.solver().Add(constraint_Boolean)
which vehicle visiting i	routing.**VehicleVar**(i) -1 if i not visited
value of dimension when reaching i	dimension.**CumulVar**(i)
Set value of dimension at i inside (a,b)	dimension.**CumulVar**(i).**SetRange**(a, b)
Increase speed for P&D	routing.**AddPickupAndDelivery**(pickup_i, delivery_i)
hard limited veh/cus force i dropped or served by veh1 or veh2	routing.**VehicleVar**(i).**SetValues**([-1, veh1, veh2]) when customer choose vehicle
Next visited node after i	rounting.**NextVar(i)** NextVar(i) == j is true if j is the node immediately reached from node i in the solution.
force j not after i	routing.**nextVar**(from).**removeValue**(to);
drop node, or have not been visited in search process	routing.**ActiveVar**(i) a Boolean variable that indicates if a node i is visited or not in the solution.
TW for customers	dimension.**CumulVar**(i).**SetRange**(a, b) for i not depot_idx

Ortools (cont)

TW for depot (for start point)	for veh_idx in range(num_vehicle): start_loc_index = routing.**Start**(veh_idx) node= manager.IndexToNode(start_loc_index) a = TW[node][0] b = TW[node][1] time_dim.**CumulVar**(start_loc_index).**SetRange**(a,b)
TW for endpoint	for veh_idx in range(num_vehicle): end_loc_index = routing.**End**(veh_idx) node= manager.IndexToNode(end_loc_index) a = TW[node][0] b = TW[node][1] time_dim.**CumulVar**(end_loc_index).**SetRange**(a,b)
Time visit at node i	time_var = time_dimension.**CumulVar**(index)
Return the earliest/ lastest possible time to visit node i	solution.**Min**(time_var) solution.**Max**(time_var)
Take an array of capacities (each vehicle has different capacity)	**AddDimensionWithVehicleCapacity**
Set dimension with different max val of dim and one callback	dimension_name = 'Capacity ' routing.**AddDimensionWithVehicleCapacity**(transit_callback_index, 0, # no slack [10000, 20000], # vehicle maximum travel distance True, #start cumul to zero dimension_name)

Ortools (cont)

Set dimension with different max val of dim and many callbacks	routing.**AddDimensionWithVehicleTransitAndCapacity**(**callback_func_list_by_veh**, 0, # no slack **[max_dim_vel list]**, # vehicle maximum travel time True, # start cumul to zero dimension_name)
multiple callbacks depend on vehicle.param	def callback(from_index, to_index, veh.param) callback_list = [partial(callback, veh.param) for veh in veh_list]
register multiple callback	register_callback_list= [routing.RegisterTransitCallback(f) for f in callback_list]
define cost function for many vehs with different costs callback sum_veh_k time(i,j)_k	for veh_id in range(len(veh_list)): routing.SetArcCostEvaluatorOfVehicle(callback_list[veh_id], veh_id)
set arc cost for vehicle k moving from i to j 1.	def callback_cost(fnode, tnode, veh_id) return 3 *d(i,j,veh_id) +40*t (i,j, veh_id) cost_callback_list =[partial(callback_cost, veh_id=veh_id) for veh_id in veh_list] register_cost_callback_list= [routing.RegisterTransitCallback(f) for f in cost_callback_list] for veh_id in range(len(veh_list)): routing.SetArcCostEvaluatorOfVehicle(register_cost_callback_list[veh_id], veh_id)

Ortools (cont)

create cost callback from i to j depend on vehicle.param	def cost(i, j, veh_id): return d(i,j,veh_id) + t(i,j,veh_id) cost_list = [partial(cost, veh_id=i) for i in veh_id_list] reg_cost_list = [routing.RegisterTransitCallback(f) for f in cost_list]
set different arc cost for different vehicle	routing.SetArcCostEvaluatorOfVehicle(reg_cost_list[i], i)
	staff_at_end = [] for vehicle_id in range(manager.GetNumberOfVehicles()): index = routing.End(vehicle_id) staff_at_end.append(staff_dimension.CumulVar(index))solver.Add(solver.Sum(staff_at_end) <= N)

SetFixedCostOfVehicle
SetMaximumNumberOfActiveVehicles

ortools 2

force vehicle i to visit its end location by the earliest possible time	routing.**AddVariableMinimizedByFinalizer**(time_dim.**CumulVar**(routing.**Start**(i)))
force vehicle i to leave its start location by the earliest possible time	routing.**AddVariableMinimizedByFinalizer**(time_dim.**CumulVar**(routing.**End**(i)))

CPSAT

create new Variable	x = model.NewIntVar(0, 10, 'x')
Implement b == (x >= 5)	model.Add(x >= 5).OnlyEnforceIf(b) model.Add(x < 5).OnlyEnforceIf(b.Not())