

Step 1: initialization:

**Python code** calls Initial\_Sub-problems\_VRP.exe, which reads input\_node.csv, input\_link.csv and input\_agent.csv, to generate the initial feasible solution by sequential vehicle loading. The solution includes output\_agent.csv, GAMS\_input\_set.txt, GAMS\_input\_parameter.txt.

* Output\_agent.csv has the vehicle routing and served passenger’s information.
* GAMS\_input\_set.txt and GAMS\_input\_parameter.txt are used for the restricted master problem solved in GAMS.
* Internal\_GAMS\_input\_incid\_veh\_path\_arc.csv, Internal\_GAMS\_input\_arc\_cap.csv, Internal\_GAMS\_input\_incid\_veh\_path\_pax.csv, Internal\_GAMS\_input\_set.csv, Internal\_GAMS\_input\_veh\_path\_cost.csv, Internal\_GAMS\_input\_veh\_path\_selection.csv, are used to record the initial solution and then will be used to update the new network.

Step 2: Solve restricted master problem in GAMS

**Python code** calls Vehicle assignment\_master.gms, which reads GAMS\_input\_set.txt and GAMS\_input\_parameter.txt , and outputs marginal files, GAMS\_arc\_cap\_marg.csv, GAMS\_vehicle\_weight\_marg.csv, GAMS\_passenger\_pickup\_marg.csv for the marginal cost/dual price calculation.

Step 3: solve the sub-problems

**Python code** calls Sub-problems\_VRP.exe, which reads input\_node.csv, input\_link.csv and input\_agent.csv, GAMS\_arc\_cap\_marg.csv, GAMS\_vehicle\_weight\_marg.csv, and GAMS\_passenger\_pickup\_marg.csv to generate new extreme points/paths for each vehicle.

The outputs include new path related information, including Sub\_GAMS\_input\_arc\_cap.csv, Sub\_GAMS\_input\_incid\_veh\_path\_arc.csv, Sub\_GAMS\_input\_incid\_veh\_path\_pax.csv, Sub\_GAMS\_input\_set.csv, Sub\_GAMS\_input\_veh\_path\_cost.csv, Sub\_GAMS\_input\_veh\_path\_selection.csv. In addition, the output\_agent\_internal.csv is used to record the output agent solution of each iteration.

**Python code** calls g\_UpdateNetworkData() to updates the new network based on the previous extreme points/paths and the current extreme points/paths.

Then it calls g\_output\_GAMS\_inputs() to regenerate the GAMS\_input\_set.txt and GAMS\_input\_parameter.txt as the new inputs for the restricted master problem in GAMS.

**Python code** calls g\_output\_agent\_VRP\_solution() to have output\_agent.csv to record the agent solution of each iteration

Then go back to step 2 to solve the restricted master problem and iteration by iteration.

When the iteration is the last iteration, **Python code** calls g\_output\_GAMS\_solution() to have

* GAMS\_vehicle\_path\_solution.csv(final solution from GAMS:lamda(v,k));
* output\_GAMS\_solution\_agent.csv(final solution with path details: the path node/time/state sequence of path k of vehicle v);

|  |
| --- |
| if \_\_name\_\_=='\_\_main\_\_':  iteration\_number=10  current\_number = 2  # obtain the initial feasble solution  # generate the initial solution of sub-problems  //output: output\_agent.csv as the initial solution/extreme point.  os.system(r'"C:\Users\jliu215\Desktop\DW-Interaction\2\_Small\_base\_test\_DW\_Interaction\Initial\_Sub-problems\_VRP.exe"')  g\_output\_initial\_agent\_VRP\_solution()  # generate the marginal values  //output: GAMS\_arc\_cap\_marg.csv; GAMS\_vehicle\_weight\_marg.csv; GAMS\_passenger\_pickup\_marg.csv;  ws\_INI = GamsWorkspace()  master\_problem\_INI = ws\_INI.add\_job\_from\_file("C:\\Users\\jliu215\\Desktop\\DW-Interaction\\2\_Small\_base\_test\_DW\_Interaction\\Vehicle assignment\_master.gms")  master\_problem\_INI.run()    # iteration by itetation to imporve the solution  for i in range(2,iteration\_number+1):  // Py\_output\_num\_VRP.csv:output the iteration number to VRP.exe to record the kth paths/iterations  g\_output\_ite\_num\_VRP(i,iteration\_number)  print("iteration\_number:%d" % i)  # generate the new solution of sub-problems  // output\_agent\_internal.csv:to record the output agent solution each iteration. Sub\_GAMS\_input\_arc\_cap.csv, Sub\_GAMS\_input\_incid\_veh\_path\_arc.csv, Sub\_GAMS\_input\_incid\_veh\_path\_pax.csv, Sub\_GAMS\_input\_set.csv, Sub\_GAMS\_input\_veh\_path\_cost.csv, Sub\_GAMS\_input\_veh\_path\_selection.csv, those files are from VRP.exe based on the path solution of each iteration.  os.system(r'"C:\Users\jliu215\Desktop\DW-Interaction\2\_Small\_base\_test\_DW\_Interaction\Sub-problems\_VRP.exe"')  // the current new network is updated based on the previous path solution and current path solution, as the input of GAMS.  g\_UpdateNetworkData()  //based on the latest network, the required txt formats for GAMS is regenerated.  g\_output\_GAMS\_inputs()  // output the agent path solution based on the current result and previous results.  g\_output\_agent\_VRP\_solution()  # generate the marginal values of master problem  //output: GAMS\_arc\_cap\_marg.csv; GAMS\_vehicle\_weight\_marg.csv; GAMS\_passenger\_pickup\_marg.csv;  ws = GamsWorkspace()  master\_problem = ws.add\_job\_from\_file("C:\\Users\\jliu215\\Desktop\\DW-Interaction\\2\_Small\_base\_test\_DW\_Interaction\\Vehicle assignment\_master.gms")  master\_problem.run()  if(i==iteration\_number):  // at the last iteration, we output the finally identifiable solution.(1)output: GAMS\_vehicle\_path\_solution.csv(final solution from GAMS:lamda(v,k)); (2)output\_GAMS\_solution\_agent.csv(final solution with path details: the path node/time/state sequence of path k of vehicle v);  g\_output\_GAMS\_solution()  print("well done!")  // output\_agent\_path\_arc\_incidence.csv is useless, and was used to output the path arc incidence for GAMS previously. |