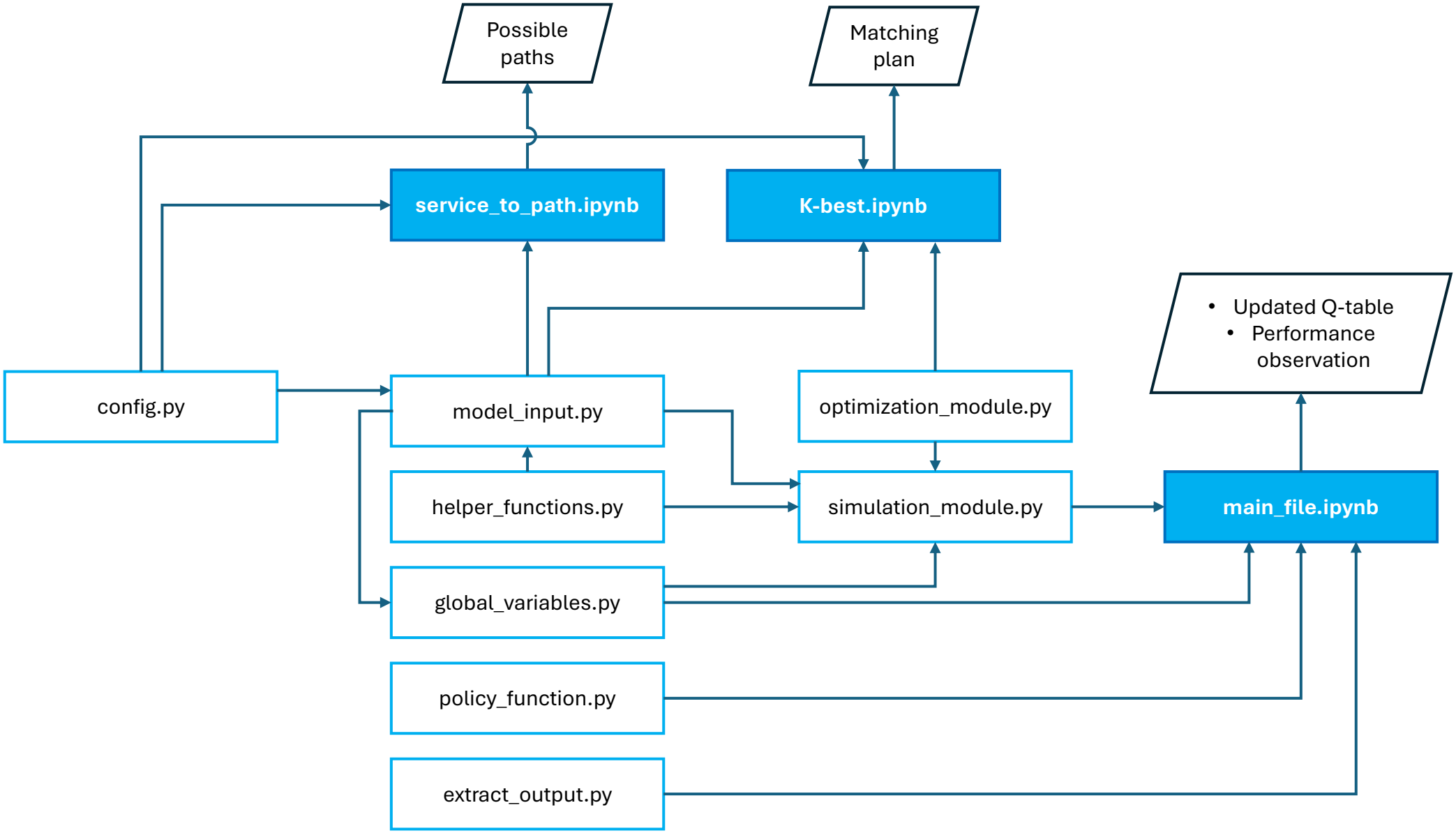
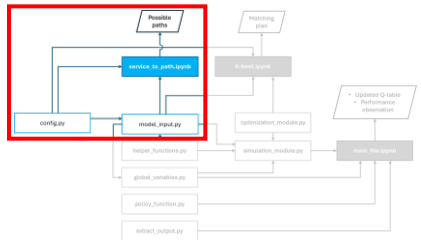
Flow Diagram

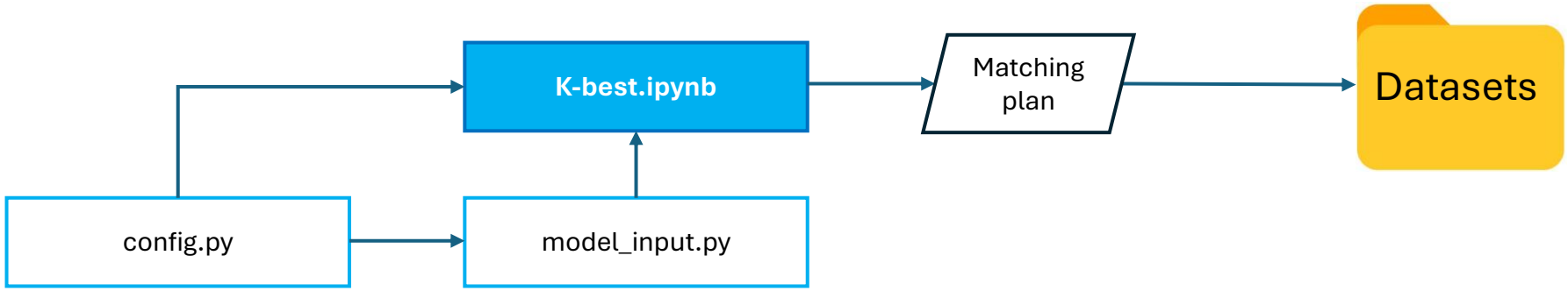
Learning Assisted Hybrid Simulation-Optimization Model

Flow Overview



Path Generation

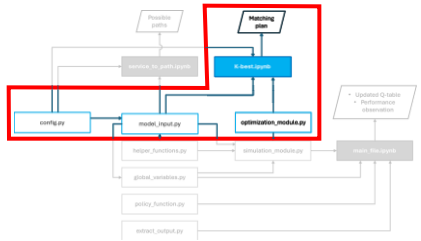


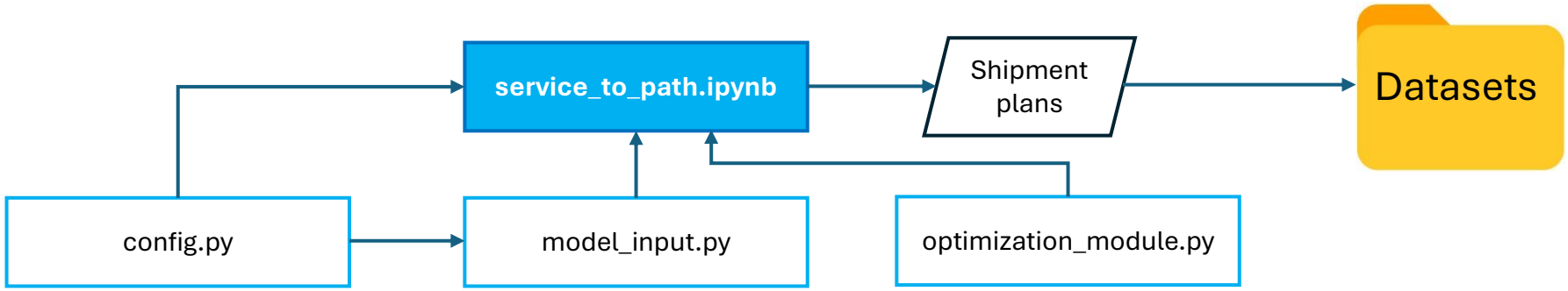


The ***service_to_path*** file is a path generation algorithm to construct possible paths within a multimodal network. The output will be used for the **optimization module**. Several key connections with other files are as follows:

- ***Config.py*** provides the file names and paths to call the service data
- ***Model_input.py*** provides the cost parameters to calculate the associated cost with each path
- The csv output consists of possible paths is stored according to the path set in the ***config.py***

K-Best Solution Approach

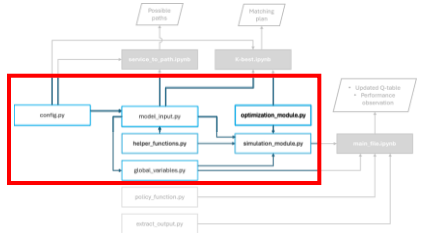


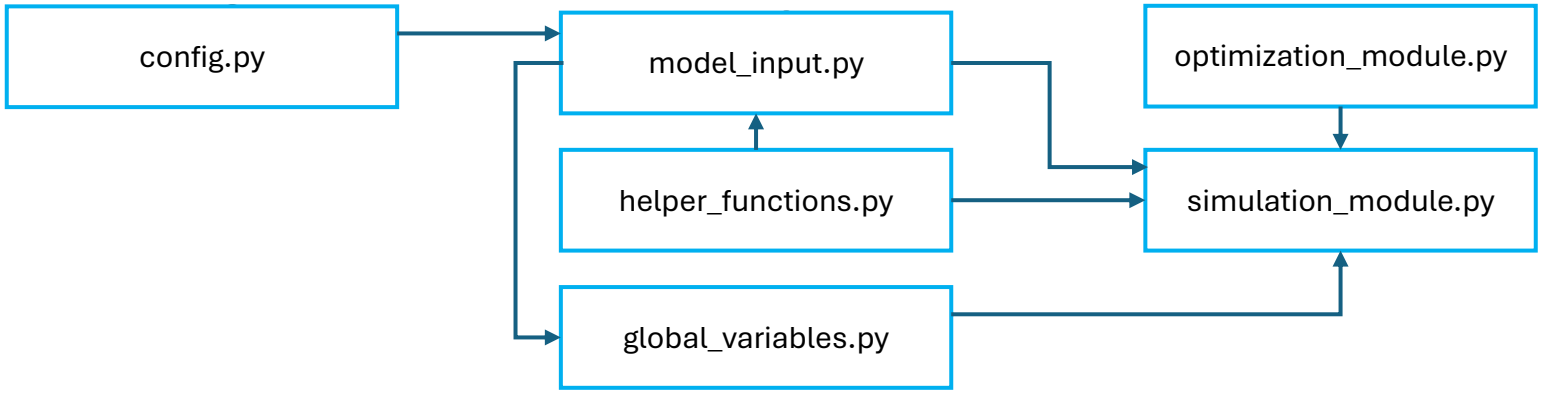


The ***K-best*** file runs the optimization module to create a solution pool (shipment plans) for each shipment. The shipment plan is itinerary consists of a service line or combination of service lines. This K-best solution approach is performed to minimize the number of optimization module triggers. It is useful to reduce the runtime during the training. Several key connections with other files are as follows:

- ***Config.py*** provides the file names and paths to call the service data
- ***Model_input.py*** provides the cost parameters to calculate the associated cost with each path
- **Optimization_module.py** provides the optimization algorithm to be used in creating the solution pool of shipment plan
- The csv output consists of the shipment request along with the solution pool is stored according to the path set in the ***config.py***

Simulation Module

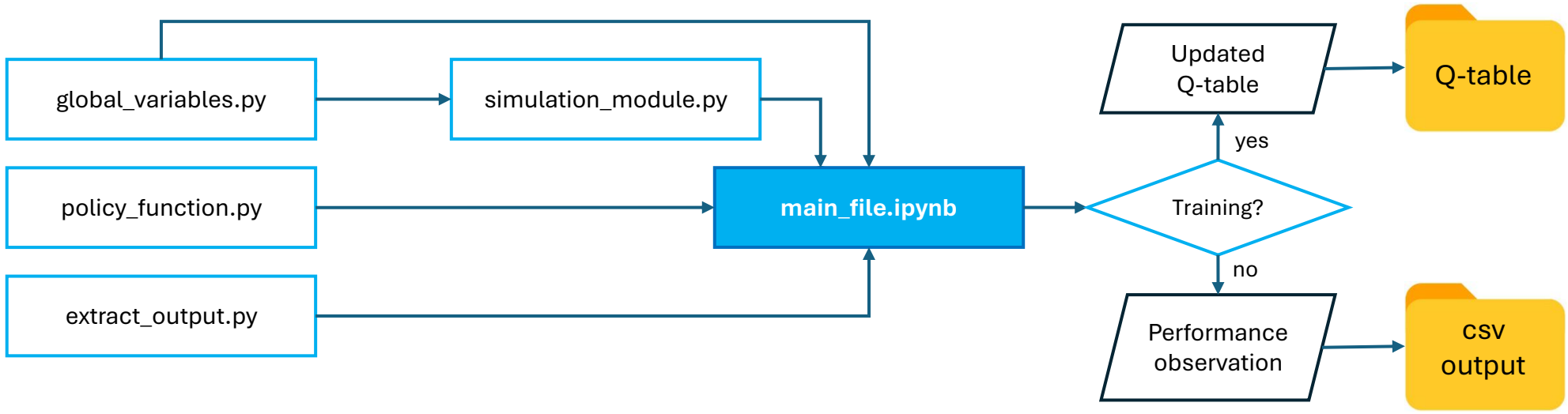


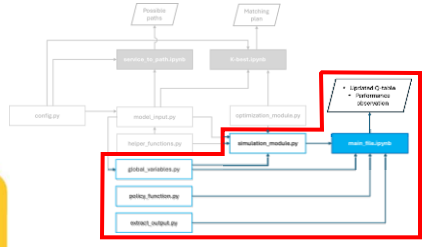


The **simulation_module.py** consists of functions to run the discrete event simulation of synchromodal transport. In general in consists of **shipment, service, and disruption classes** as the main component of the simulations. Furthermore, the **matching module and reinforcement learning** are integrated to the simulation environment for planning and reaction strategy to disruption. Several key connections with other files are as follows:

- ***Config.py*** provides the simulation settings and file paths both for input and output.
- ***Model_input.py*** provides all the input including demand/request, service network, costs, and other important parameters to the simulation.
- **Helper_functions.py** provides small functions to be used for data preprocessing, printing logs, and some processes inside the simulation
- **Global_variables.py** provides the variables to be accessed by the functions inside the simulation_module.py. Some variables are used to keep track of observation during the simulation such as cumulative costs.
- **Optimization_module.py** is called in the matching module to provide the matching plan if it is triggered

Main File





The **main_file** calls all the functions in the **simulation_module.py.** It can run the simulation through a looping process until the intended number of simulation is fulfilled according to the simulation settings in the **config.py**. Several key connections with other files are as follows:

- **Global_variables.py** provides a function to restart the global variables in the beginning of every loop.
- **Simulation_module.py** provides all the functions for the discrete event simulation to be executed.
- **Policy_function.py** provides a function to access the Q-table when the RL agent makes decision.
- The main file keep track of the observation throughout the episodic simulations and store it as the csv file in the intended path according to the **config.py**
- **Extract_output.py** provided function to track the observation for each shipment and service individually.
- The output depends whether the file runs for training or implementation.