**Algorithm Expansion Document**

Before adding algorithms, you need to know the simulation process in the WorkflowSim system developed by the university of southern California. By summarizing, its simulation process can be described as follows:

①WorkflowParse.java parses the XML file to get the taskList;

②ClusteringEngine.java does clustering to get jobList;

③WorkflowEngine.java submits tasks to FogBroker.java according to the structure of the workflow(only submit tasks that have no parent tasks or that whose all parent tasks have already been finished);

④FogBroker.java invokes the scheduling algorithm to assign VMS to the tasks received;

⑤FogDevice.java processes the tasks that have been assigned to VM and returns every task’s processing result;

⑥FogBroker.java returns the processing results to WorkflowEngine.java, and then WorkflowEngine.java chooses to re-process the task or end the simulation, or continue to submit the task according to the processing results and the number of tasks have been processed. If WorkflowEngine.java needs to continue to submit the task, then return to Step ③.

Here is how to add algorithms to the system:

**(1)add non-intelligent algorithms**

In the WorkflowSim system developed by the university of southern California, all the workflow scheduling algorithms used are non-intelligent algorithms. Thus, if you want to add a non-intelligent algorithm to the system, all you need to do is to [imitate](D:/Program%20Files%20(x86)/Youdao/Dict/8.5.0.0/resultui/html/index.html#/javascript:;) the MINMIN algorithm that comes with the system and do the following two operations.

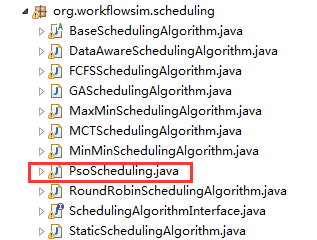
①Make your algorithm inherits BaseSchedulingAlgorithm.java, and then put your algorithm class under org. workflowsim. scheduling;

②Add your algorithm’s name in org.workflowsim.utils.Parameters.SchedulingAlgorithm.

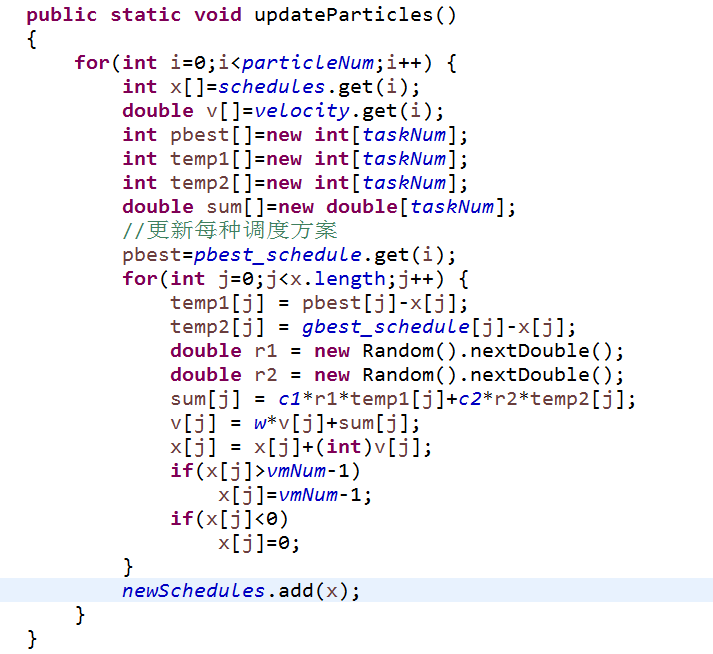
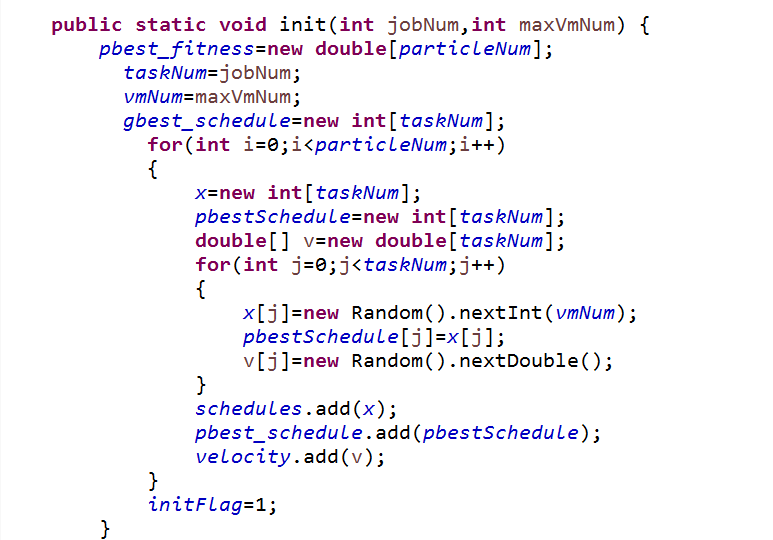
**(2)add intelligent algorithms**

Based on the above description of the simulation process, you may have noticed that in the WorkflowSim system, a complete scheduling algorithm needs to be called once before each batch of tasks are processed to assign VM to these tasks. But when you add an intelligent algorithm to the system, such an approach will not work. Because the intelligent algorithm needs iteration and evaluation function, but we can’t evaluate the current scheduling scheme until a complete workflow have been executed. Therefore, in order to add an intelligent algorithm to the system, we need to decompose it and then integrate it into the system. Here is an example of the steps to add the PSO algorithm to the system.

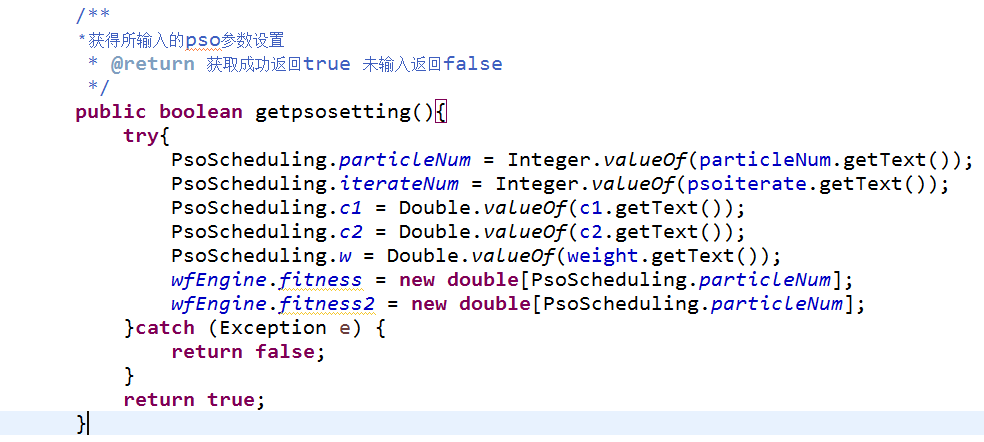
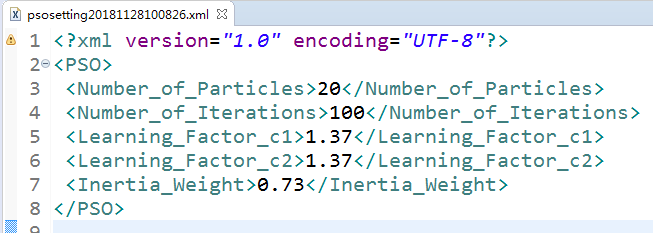
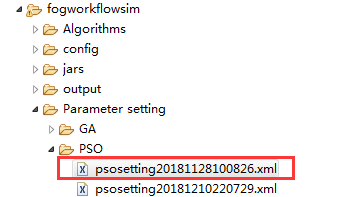
①create a java class named PsoScheduling in org.workflowsim.scheduling;



②Write PsoScheduling methods;



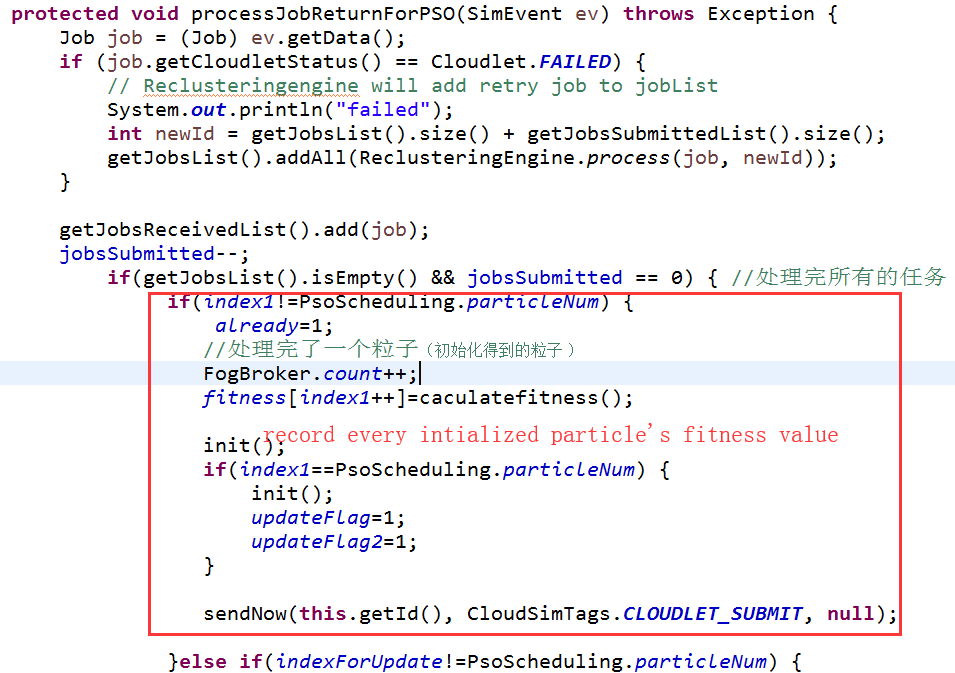
③Separate the parameters(such as Iteration number, learning factor, inertia weight, population size) from PsoScheduling.java and put them into an XML file,and then write a method to get parameter settings;

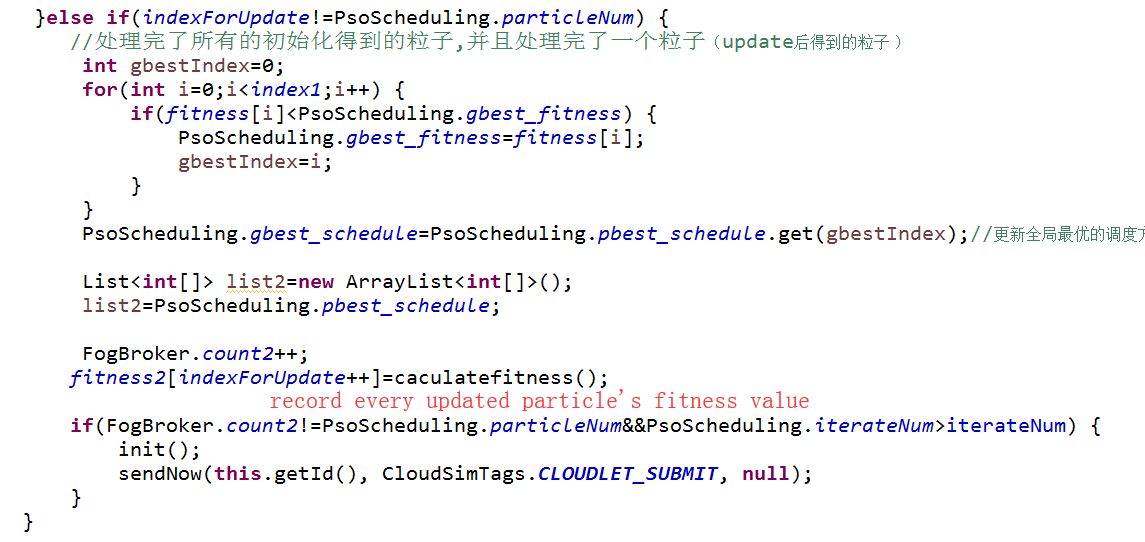


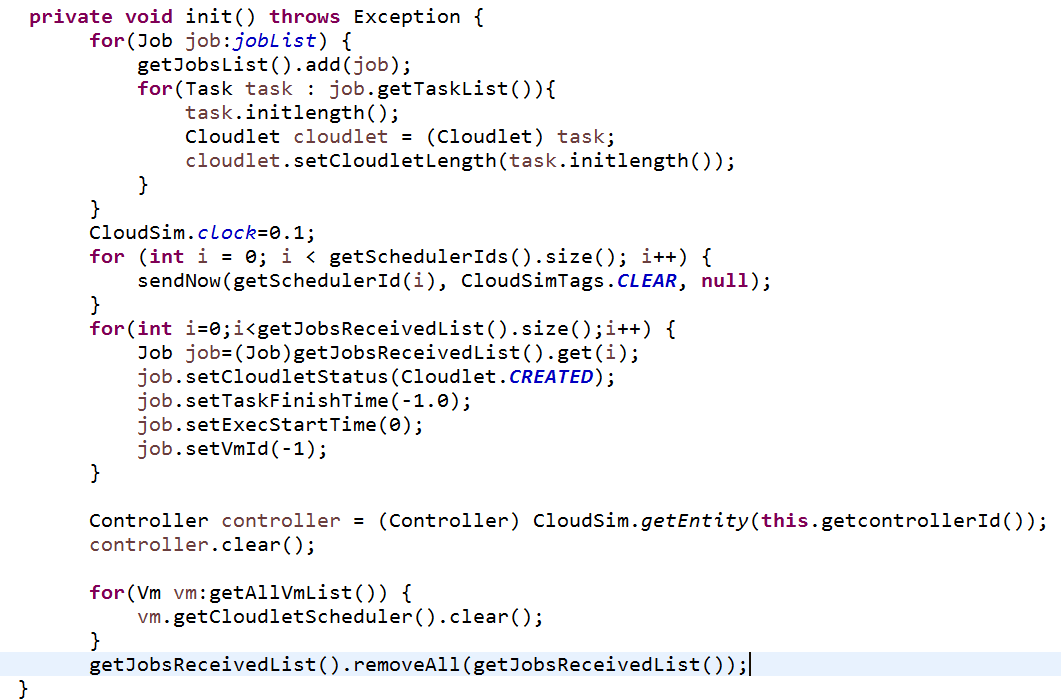
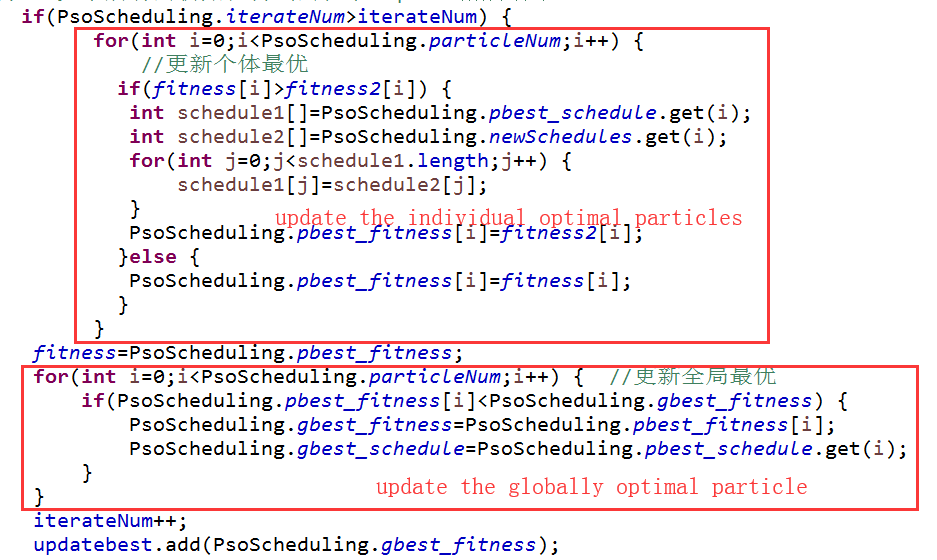
④Modify the code snippet labeled CloudSimTags.CLOUDLET\_RETURN in org.workflowsim.WorkflowEngine. processEvent(SimEvent). This method is to make the method named “processJobReturnForPSO” be executed when the algorithm used is PSO;



⑤Add an method named “processJobReturnForPSO” in org.workflowsim.WorkflowEngine. In this method, change some parameters coupled to FogBroker according to the progress of the task processing, and record the fitness value of each particle.It should be noted that the fitness value of each particle is obtained by making the system perform a simulation. Therefore, init() method needs to be called to clean up the data from the last simulation every time before a new scheduling scheme is executed.



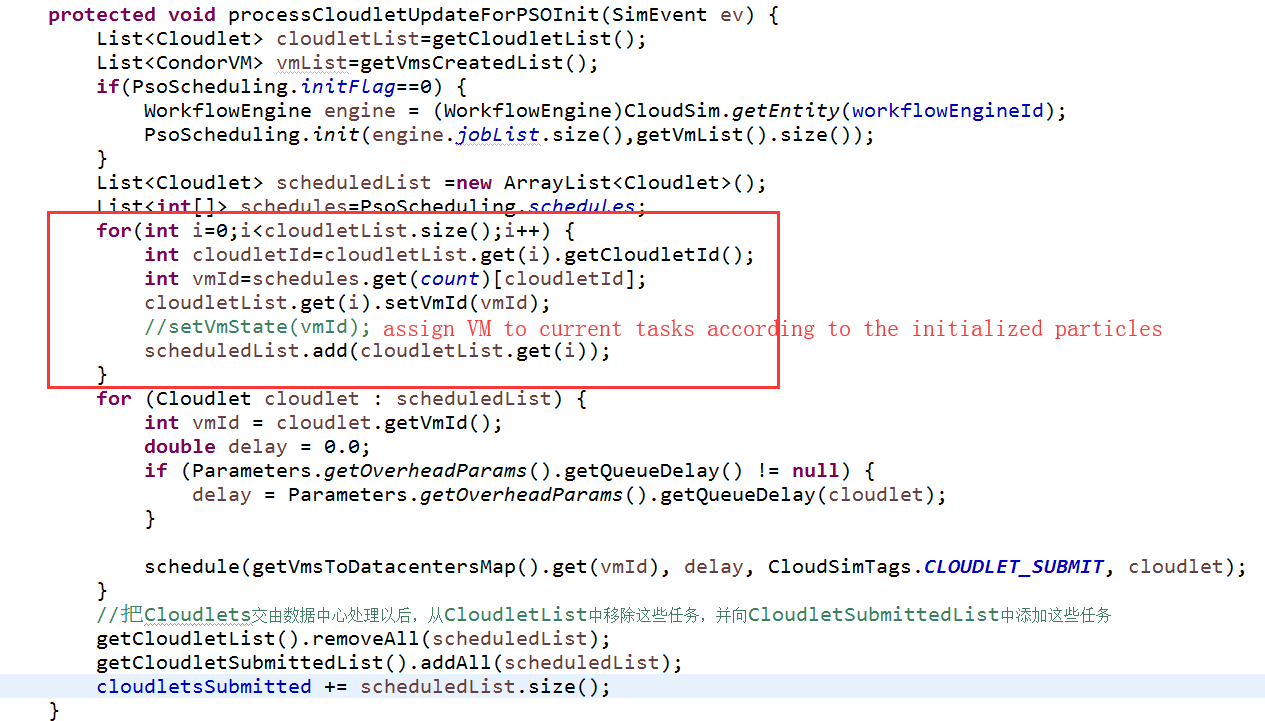




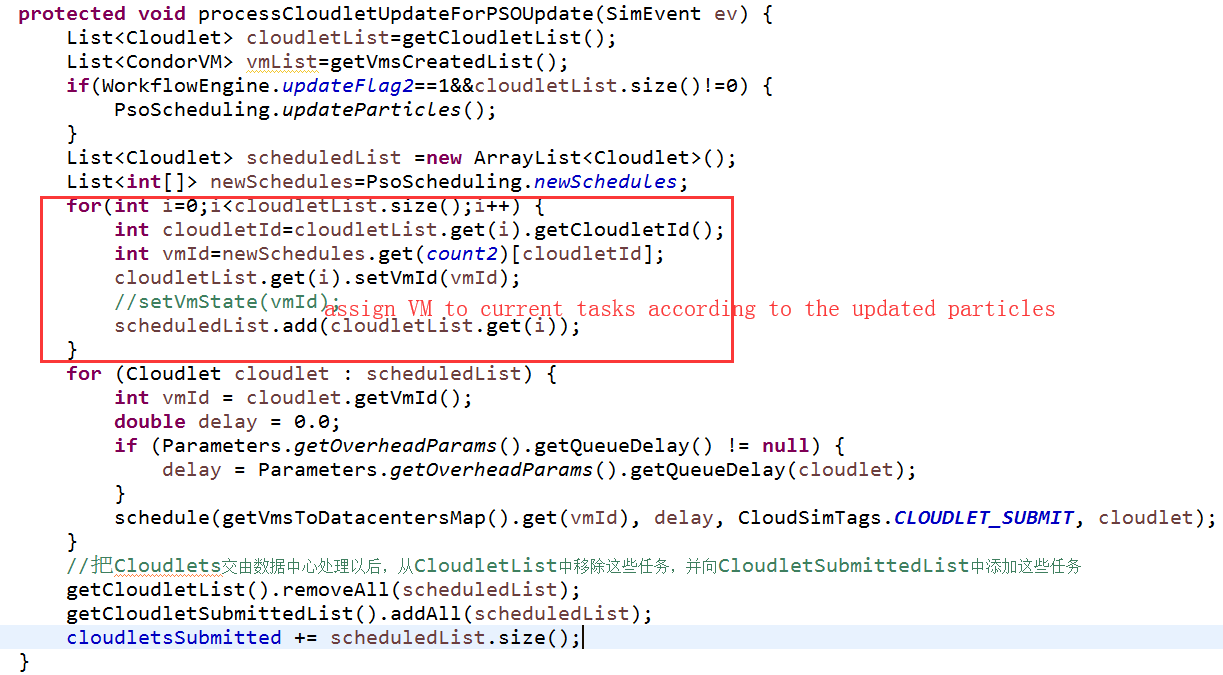
⑥Modify the code snippet labeled WorkflowSimTags.CLOUDLET\_UPDATE in org.fog.entities.FogBroker.processEvent(SimEvent). This method is to make the PSO algorithm can dynamically select the method to be executed among “processCloudletUpdateForPSOInit(SimEvent)”,”processCloudletUpdateForPSOUpdate(SimEvent)” and “processCloudletUpdateForPSOGbest(SimEvent)” according to the parameters set in step ⑤.



⑦Add a method named “processCloudletUpdateForPSOInit” in org.fog.entities.FogBroker. This method is to assign VM to current tasks according to the initialized particle swarm.



⑧Add a method named “processCloudletUpdateForPSOUpdatet” in org.fog.entities.FogBroker. This method first determines whether to call the method that updates the particle swarm in PsoScheduling.java according to the parameters in WorkflowEngine.java. And then assign VM to the current tasks according to the new particle swarm.



⑨Add a method named “processCloudletUpdateForPSOGbest” in org.fog.entities.FogBroker. This method is to assign VM to current tasks according to the globally optimal particle.

