|  |  |  |
| --- | --- | --- |
| 南农大  软件开发方法论  接口编程实验报告  南农校徽 | | |
|  | 题 目: | 基于web service的PSO与DE算法的设计与实现 |
|  | 姓 名: | Jerry |
|  | 学 院: | 人工智能学院 |
|  | 专 业: | 计算机科学与技术 |
|  | 学 号: | 2022 |
|  | 指导教师: | 姜海燕 职称: 教授 |
| 2023年 4月 25 日 | | |
| 南京农业大学研究生院制 | | |

基于面向对象、基于构件和面向web服务的DE算法和PSO算法的设计与实现

摘要：在软件开发领域，面临着不断增长的复杂度、开发周期延长、成本增加以及维护难度加大等问题。而在这种背景下，代码复用被认为是解决软件危机的一种重要手段。本文探讨了面向对象、基于构件以及面向web服务三种软件开发方法在封装标准差分进化算法[1]和标准PSO[2]算法方面的应用，并实现了一个可视化界面程序，通过标准测试函数调用已封装的标准差分进化算法和标准PSO算法，展示了代码级复用和组件级复用的实际效果。本文介绍了算法的具体实现方法，详细描述了测试实验的过程和结果和项目中使用的开发框架、工具和部署方法，以及服务器端的分布式部署方式。实验结果表明，面向对象、基于构件以及面向web服务三种软件开发方法均可有效地实现算法的封装，提高代码复用性，并且在可视化界面程序中使用时，具有良好的性能和稳定性。

关键词：面向对象；组件级复用；web服务；分布式部署

Design and Implementation of DE and PSO Algorithms based on Object-oriented, Component-based, and Web Service-oriented Software Development Methods

Abstract: In the field of software development, there are increasing problems of complexity, longer development cycles, higher costs, and greater difficulty in maintenance. In this context, code reuse is considered an important means to solve the software crisis. This paper explores the application of three software development methods: object-oriented, component-based, and web service-oriented, in encapsulating standard differential evolution (DE) and particle swarm optimization (PSO) algorithms. A visual interface program is implemented to demonstrate the practical effects of code-level reuse and component-level reuse through calling the encapsulated standard DE and PSO algorithms with standard test functions. This paper describes the specific implementation methods of the algorithms, the detailed process and results of the testing experiments, the development frameworks, tools, and deployment methods used in the project, as well as the distributed deployment of the server-side. The experimental results show that the three software development methods of object-oriented, component-based, and web service-oriented can effectively encapsulate algorithms, improve code reuse, and have good performance and stability when used in visual interface programs.

Keywords: Object-oriented, Component-level reuse, Web services, Distributed deployment

一、实验目的

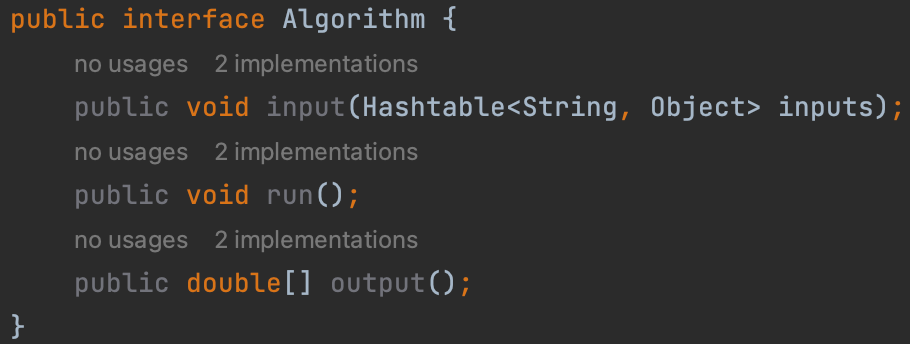
本文的实验目的在于探索如何使用面向对象、基于构件以及面向web服务共3种软件开发方法封装标准差分进化算法和标准PSO算法，并且实现通过标准测试函数调用已封装的标准差分进化算法和标准PSO算法，并给出测试结果。封装也是面向对象编程中的一项非常重要的技术, 它隐藏了与用户无关的内部技术的复杂性。这样, 无论所使用的对象如何复杂, 从使用者的角度看来, 都是一个易操作的“黑盒子”。该“黑盒子”对外提供一个公共接口, 通过该接口即可轻松实现组件化的代码复用[3]。通过本文的实验，我们希望能够掌握如何使用Java语言进行软件开发，并且深入理解面向对象、基于构件以及面向web服务的开发方法的优点与适用场景。

二、面向对象的设计与实现

本小节，我们采用了面向对象的设计思想，将标准差分进化算法和标准PSO算法进行封装，设计了一个接口类Algorithm、一个父类FAlgorithm和实现类PSO、DE，具体的设计和实现方法如下：

2.1接口设计

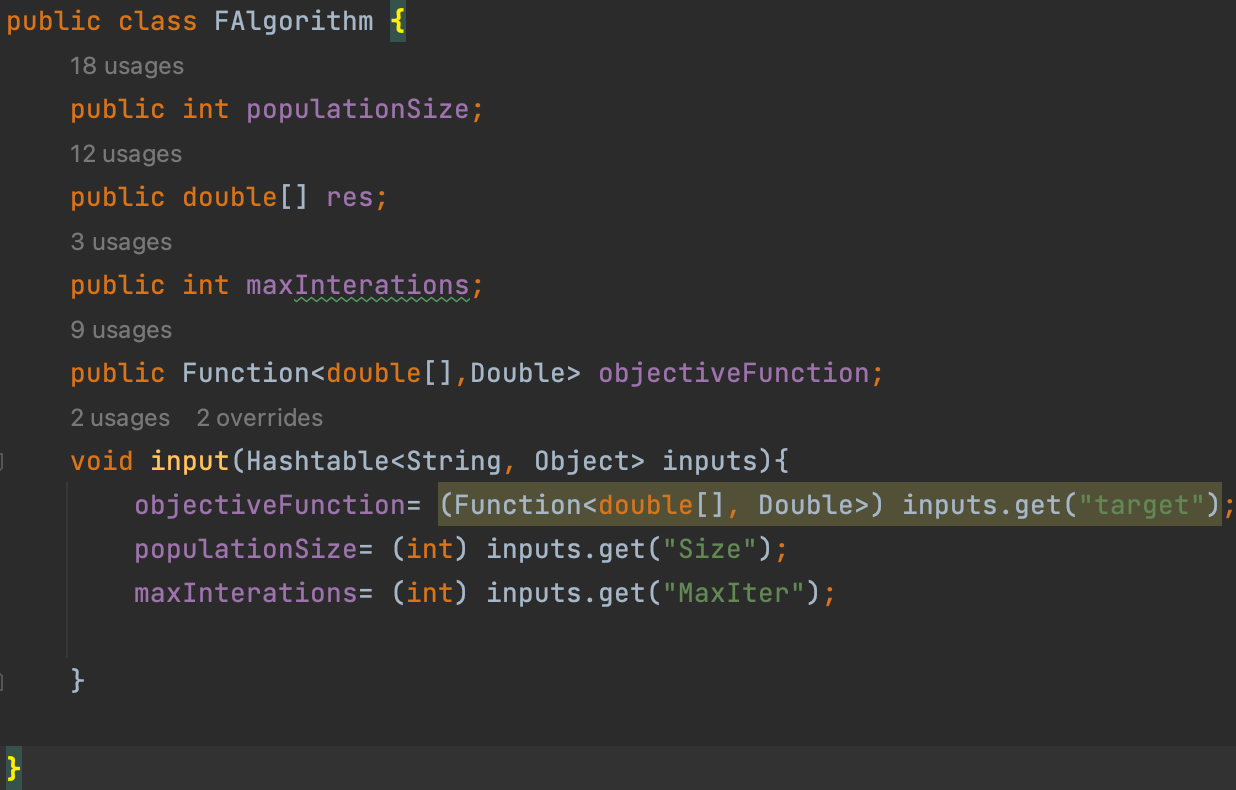
为了实现算法的封装和复用，我们首先设计了一个Algorithm接口，该接口定义了算法的基本操作，包括算法初始化、获取结果等。Algorithm接口的定义如下：



其中，input方法用于算法参数的初始化，inputs为算法需要用到的参数，由于算法的参数格式并不统一，并且函数也要作为参数进行传输，考虑到数据格式的复杂性在这里使用Hashtable类型数据对算法的输入参数进行存储；run方法用于运行算法；output方法用于获取当前的最优解，由于最优解包含当前的结果和对应的坐标因此采用double数组进行存储，第一个位置存放结果信息，其余位置存放坐标信息。在本课程实验中，PSO类和DE类分别实现了Algorithm接口。

2.2父类设计

为了避免重复编写代码，我们设计了一个父类FAlgorithm，该类实现了算法的公共部分，以便PSO和DE算法对共有部分进行复用。FAlgorithm的定义如下：



在FAlgorithm类中，我们定义了DE算法和PSO算法共有的参数，如种群尺寸（populationSize）、迭代次数（MaxInterations）、目标函数（objectiveFunction）等。针对算法初始化方法，也对上述共有参数进行了初始化。

2.3算法实现

在PSO类和DE类中，我们分别实现了Algorithm接口，并继承了FAlgorithm类，实现了标准PSO算法和标准差分进化算法的具体逻辑。

2.3.1 PSO算法简介

PSO算法是一种模拟鸟群寻食行为的全局优化算法。在PSO算法中，每个粒子都有一个位置和速度，通过不断调整位置和速度来搜索最优解。每个粒子的速度由当前速度、个体最优解和全局最优解三个部分组成，通过这三个部分的加权和来更新速度。位置则根据更新后的速度进行调整。具体来说，PSO算法可以描述为以下步骤：

PSO算法伪码如下：

Inputs:

swarm\_size - 粒子群大小

max\_iter - 最大迭代次数

c1, c2 - 学习因子

w - 惯性因子

search\_space - 搜索空间

objective\_func - 目标函数

Outputs:

gbest - 全局最优解

gbest\_cost - 全局最优解的目标函数值

Initialization:

Initialize positions, velocities, and pbest for each particle--初始化每个粒子的位置、速度和个体最优解

Evaluate objective function for each particle and set pbest as current position--将每个粒子的目标函数值设为当前位置，并将其作为个体最优解

Iteration:

for i=1 to max\_iter do--for循环进行迭代，共迭代max\_iter次

for j=1 to swarm\_size do--每次迭代中，对于每个粒子，执行以下操作

Update velocity of particle--更新粒子的速度

Update position of particle--更新粒子的位置

Evaluate objective function for particle--计算粒子的目标函数值

Update pbest if necessary--如果该粒子的目标函数值比其个体最优解的目标函数值更优，则将其当前位置作为个体最优解

Update gbest if necessary--如果该粒子的目标函数值比全局最优解的目标函数值更优，则将其当前位置作为全局最优解

end for

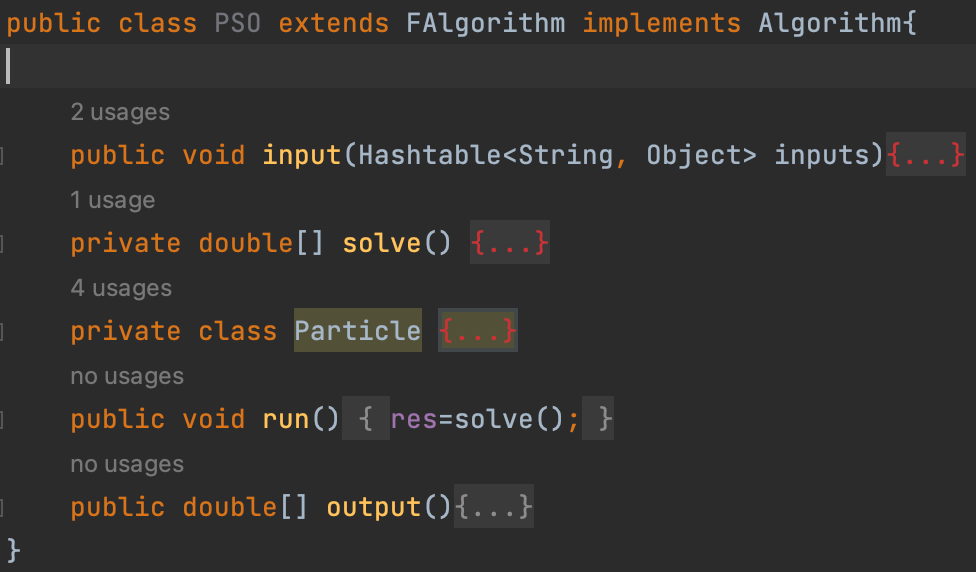
end for

Return:

gbest, gbest\_cost

2.3.2 PSO算法实现

在PSO类中，继承了FAlgorithm类，并且实现了Algorithm接口。PSO算法的输入和输出也是实现了Algorithm接口的input()和output()方法，其中input()方法会将输入的参数存储在FAlgorithm类的成员变量中，并在run()方法中调用私有方法solve()求解，最后通过output()方法将结果返回。



首先，PSO类需要实现Algorithm接口中定义的算法输入、输出、运行等方法以适应系统需求。

其次，PSO类继承了FAlgorithm父类，该父类实现了算法中公有的方法和属性，例如种群大小、目标函数、初始化方法等，PSO类在input方法里直接使用了这些方法和属性。

最后，在PSO类中，solve()方法和Particle类是私有的，主要用于具体实现粒子群优化算法。

2.3.3 DE算法简介

标准差分进化算法 (Standard Differential Evolution Algorithm, 简称DE算法) 是一种基于群体智能的优化算法。DE算法的基本思想是通过对当前种群中的个体进行差分变异和交叉操作，生成一些新的解，并将其与原有的解进行比较和选择，从而逐步逼近全局最优解。

具体地，DE算法将一个种群看做一个解空间，其中每个个体代表一个解。在DE算法中，首先从种群中选择三个个体，然后对其进行差分变异操作，生成一个新的个体。接着将新的个体与当前个体进行交叉操作，得到一个试验解。如果试验解的适应度值优于当前解，则用试验解替换当前解。

标准差分进化算法中的参数包括种群大小、最大迭代次数、缩放因子 F 和交叉概率 CR 等。其中，缩放因子 F 控制变异操作的幅度，交叉概率 CR 控制交叉操作的概率。

DE算法伪码如下：

Inputs:

pop\_size - 种群大小

max\_iter - 最大迭代次数

F - 缩放因子

CR - 交叉概率

search\_space - 搜索空间

objective\_func - 目标函数

Outputs:

best\_solution - 最优解

best\_fitness - 最优解的目标函数值

Initialization:

Initialize population randomly within search space--随机初始化种群中每个个体的位置

Iteration:

for i=1 to max\_iter do

for j=1 to pop\_size do

Choose three random individuals from population--从种群中随机选择三个个体，作为差分进化算法的输入

Perform mutation operation using F--利用差分进化算法的公式，生成一个试验向量

Perform crossover operation using CR--利用交叉概率CR，将个体和试验向量进行交叉，得到一个新的个体

Evaluate objective function for trial vector--对生成的试验向量求解目标函数

Update population if necessary--如果试验向量的适应度优于原始个体，则更新个体

end for

end for

Return:

best\_solution, best\_fitness

2.3.4 DE算法实现

在DE类中，继承了FAlgorithm类，并且实现了Algorithm接口。DE算法的输入和输出也是实现了Algorithm接口的input()和output()方法，其中input()方法会将输入的参数存储在FAlgorithm类的成员变量中，并在run()方法中调用私有方法optimize()求解，最后通过output()方法将结果返回。



DE类中也定义了一些私有方法，用于具体实现DE算法。DE算法的主要求解过程在optimize()方法中实现，其中包含了DE算法的核心逻辑。DE算法中需要生成一定数量的个体并不断更新个体的参数，更新方式包括变异和交叉等操作，直到达到停止条件为止。在solve()方法中。

1. 基于构件的设计与实现

基于构件的设计可以提高代码的复用性，降低代码的耦合度，使得代码更加可维护和可扩展。在本项目中，基于IDEA，编程语言采用Java，使用Maven管理工具将算法实现部分打包成了独立的构件，可以方便地在其他项目中引用。同时，也将接口类Algorithm单独打包成一个jar包，这是因为接口类只包含算法接口的声明，而不涉及具体的算法实现。将其单独打包成jar包可以方便其他项目中引用算法接口而无需引用整个算法实现部分，提高了代码的可重用性和灵活性。因此需要生成三个jar包，分别对应Algorithm接口类、PSO实现类和DE实现类。

3.1 Maven简介

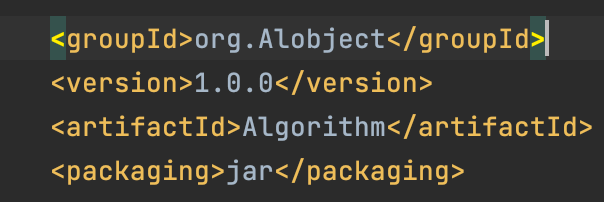
Maven是一个基于Java平台的项目管理工具，它可以帮助我们完成项目构建、依赖管理、自动化测试、打包发布等工作。它通过一个项目对象模型（POM）文件来描述项目，通过插件来完成各种任务。Maven采用的是约定大于配置的思想，即通过一系列的默认约定来减少配置工作，从而提高项目开发的效率。

3.2 Maven打包配置

在Maven中，可以通过在pom.xml文件中配置<packaging>标签指定打包类型，在本次项目中，我们使用Maven来进行打包。在Maven中，一个项目被描述为一个POM文件，它包含了项目的各种信息，如项目名称、版本、依赖、插件等。我们可以通过在POM文件中定义相关配置，来实现项目的打包。

在本项目中，具体实现如下：

1. 以生成Algorithm接口构件为例，在Algorithm、PSO、DE的pom.xml文件中进行如下配置



groupId是指定项目所属的组，通常是公司或组织名。artifactId是指定项目的唯一标识符，通常是项目的名称。version是指定项目的版本号。packaging是指定项目打包的类型，例如jar、war、ear等。在这个例子中，项目的类型是jar，表示该项目将被打包为一个JAR文件。

1. 由于PSO和DE需要实现接口中定义的方法和继承父类方法因此，在PSO和DE的pom.xml文件中还需要引入Algorithm和FAlgorithm依赖。



其中包含了3个依赖：

Algorithm依赖，指定了组ID为"org.Alobject"，artifact ID为"Algorithm"，版本号为"1.0.0"，作用域为"compile"，表示需要在编译时使用Algorithm的jar包。

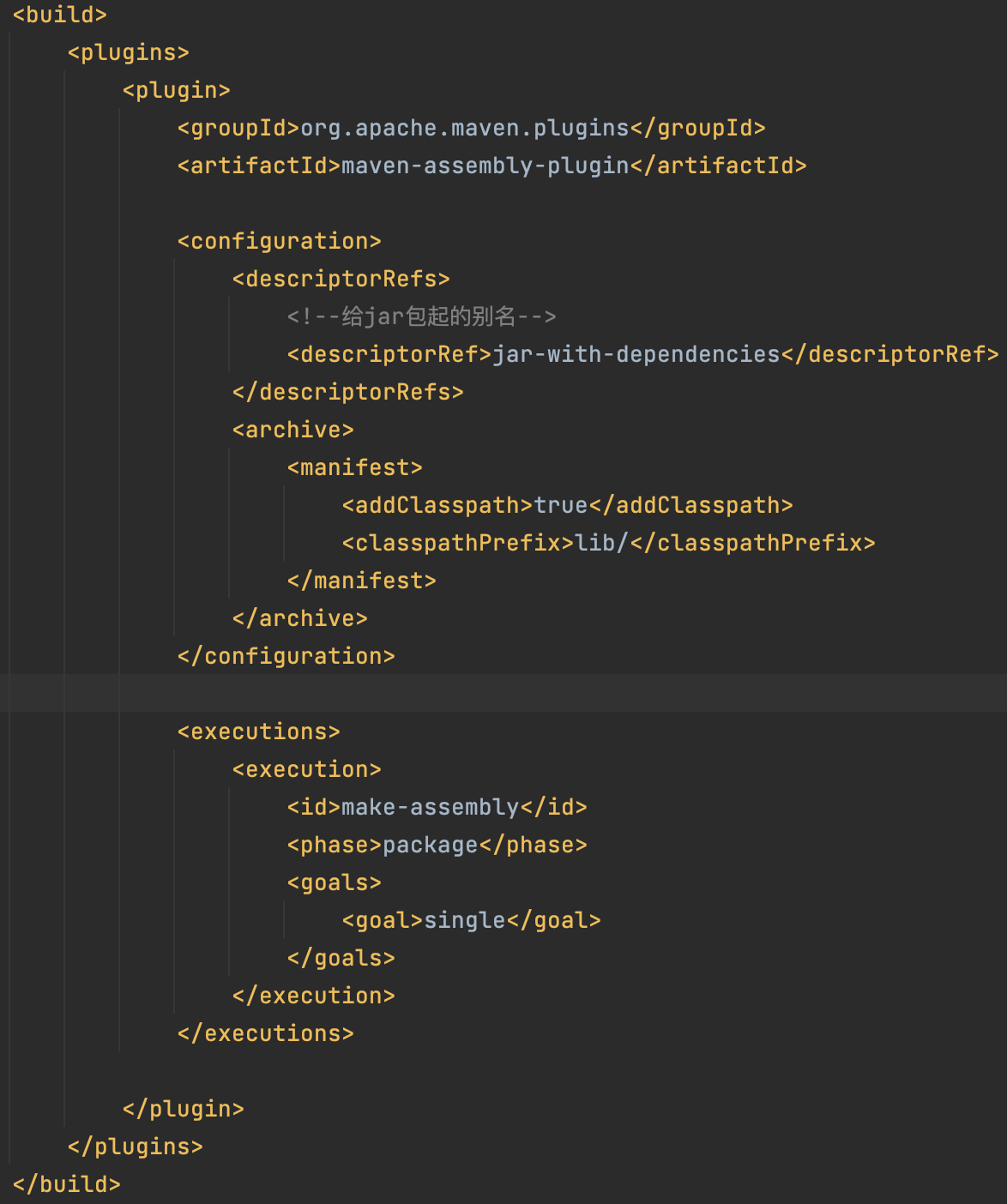
FAlgorithm依赖，指定了组ID为"org.Alobject"，artifact ID为"FAlgorithm"，版本号为"1.0.0"，作用域为"compile"，表示需要在编译时使用FAlgorithm的jar包。

<scope>compile</scope> 表示该依赖项是在编译时和运行时都需要的。

maven-assembly-plugin依赖，指定了组ID为"org.apache.maven.plugins"，artifact ID为"maven-assembly-plugin"，版本号为"2.2-beta-5"，表示使用maven-assembly-plugin插件来打包项目。

其中，maven-assembly-plugin是Maven中用来打包项目的插件，可以将多个jar包打包成一个可执行的jar包。

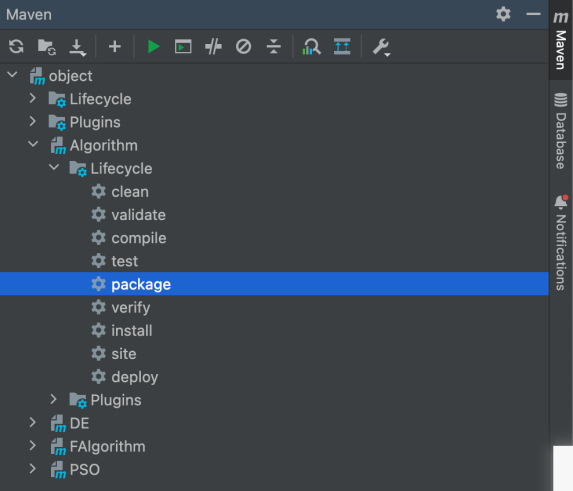
1. 由于PSO和DE需要将依赖一起打包，因此还需要在pom.xml中进行相应的配置。



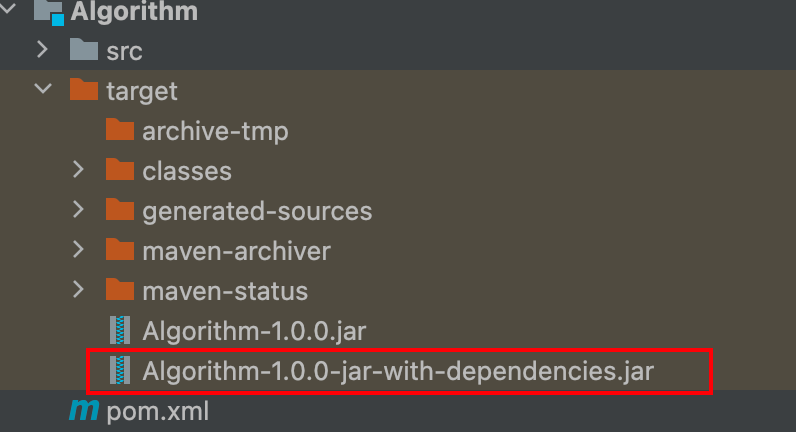
通过这段配置代码，Maven会在打包阶段生成带依赖的可执行Jar包，同时在MANIFEST.MF中加入Class-Path属性，以便在运行时引入依赖。这样，我们就可以将这个Jar包拷贝到其他机器上进行调用，而无需担心缺少依赖的问题。

3.3使用Maven打包

在IDEA右侧点击Maven图标，选择对应的需要打包的文件，双击该文件目录下的package按钮，即可生成jar包。



生成的jar包会在target目录下



3.4 PSO和DE构件测试

本小节针对已经生成的PSO构件和DE构件进行测试。

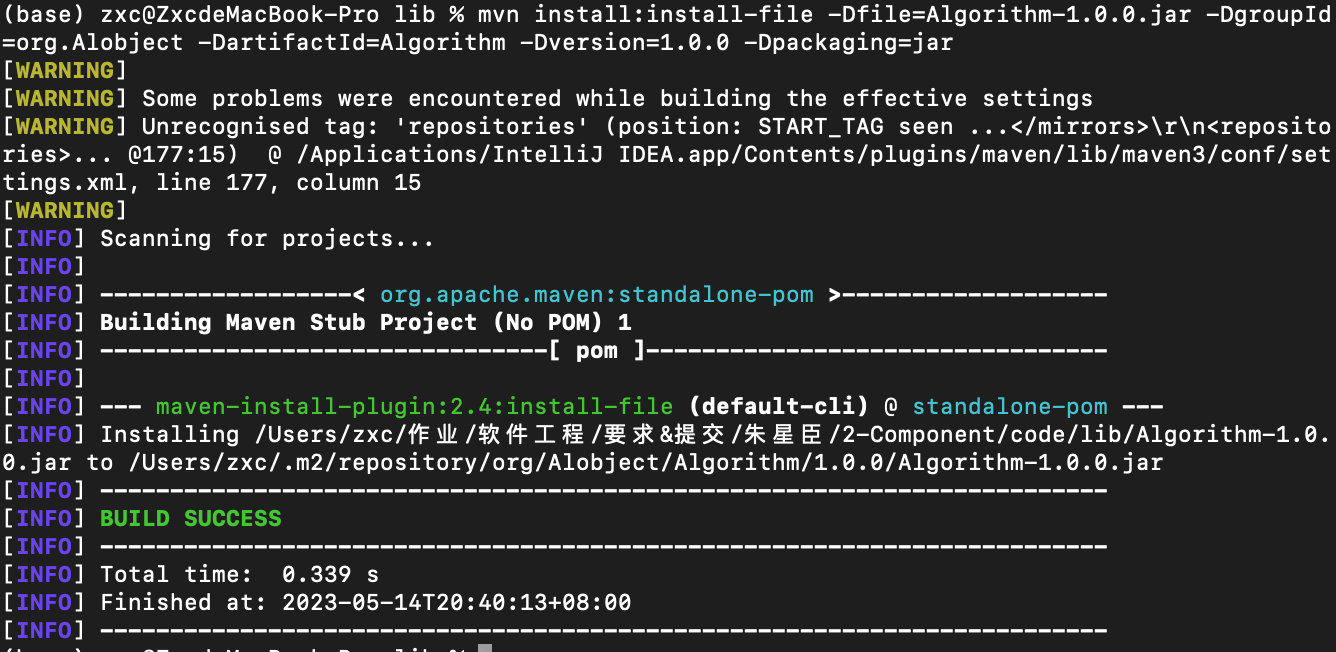
1. 将Algorithm、PSO和DE包加入到maven本地仓库

打开命令行窗口，进入生成好的jar包所在的目录。运行以下命令将jar包安装到本地仓库中：

mvn install:install-file -Dfile=<path-to-jar> -DgroupId=<group-id> -DartifactId=<artifact-id> -Dversion=<version> -Dpackaging=<packaging>

其中，<path-to-jar>是生成好的jar包的路径，<group-id>是项目的组ID，<artifact-id>是项目的Artifact ID，<version>是项目的版本号，<packaging>是项目的打包类型

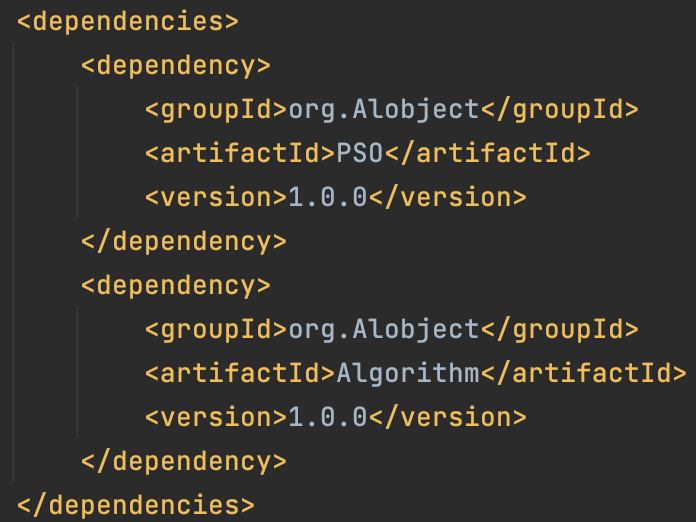
下图是将Algorithm加入到maven本地仓库，PSO和DE也是类似。



运行完命令后，Maven会将jar包安装到本地仓库中。可以在本地仓库的目录（通常位于用户目录下的.m2目录）中查看安装的jar包。在项目中可以直接引用该jar包。

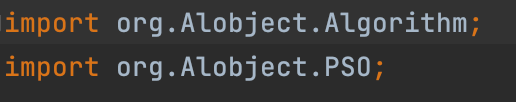
1. 添加相关依赖

在测试项目的pom.xml文件中添加相应的依赖，以测试PSO算法为例：



1. 导入引用类

在引用PSO和DE构件的Java类中，使用import语句导入相应的类，例如：

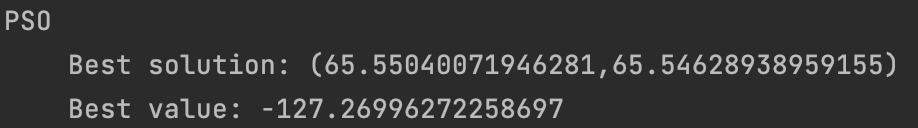


1. 实例化类并应用

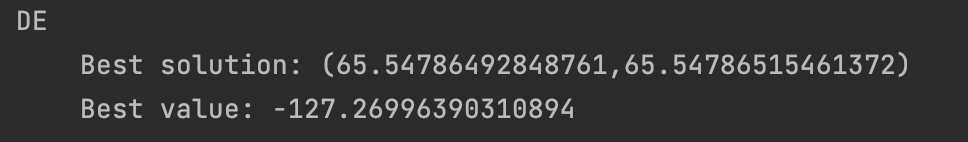
通过接口对PSO和DE类进行实例化，然后就可以在项目中引用PSO或DE构件，并使用其中提供的算法实现相应的功能了。



下图是利用测试schwefel函数对PSO算法形成的构件的测试结果



下图是利用测试schwefel函数对DE算法形成的构件的测试结果



1. 面向Web service 的设计与实现

本部分介绍了如何使用CS模式进行面向Web Service的设计与实现。具体地，服务器端采用了Spring MVC框架进行开发，使用了PSO和DE构件，而客户端与服务器端共处一个局域网内，使用Tomcat8进行了分布式部署。

4.1Spring MVC框架简介

Spring MVC是一种基于Java的MVC框架，用于Web应用程序的开发。它使用了Spring框架的核心功能，并添加了Web相关功能，提供了一种优雅的方法来构建Web应用程序。

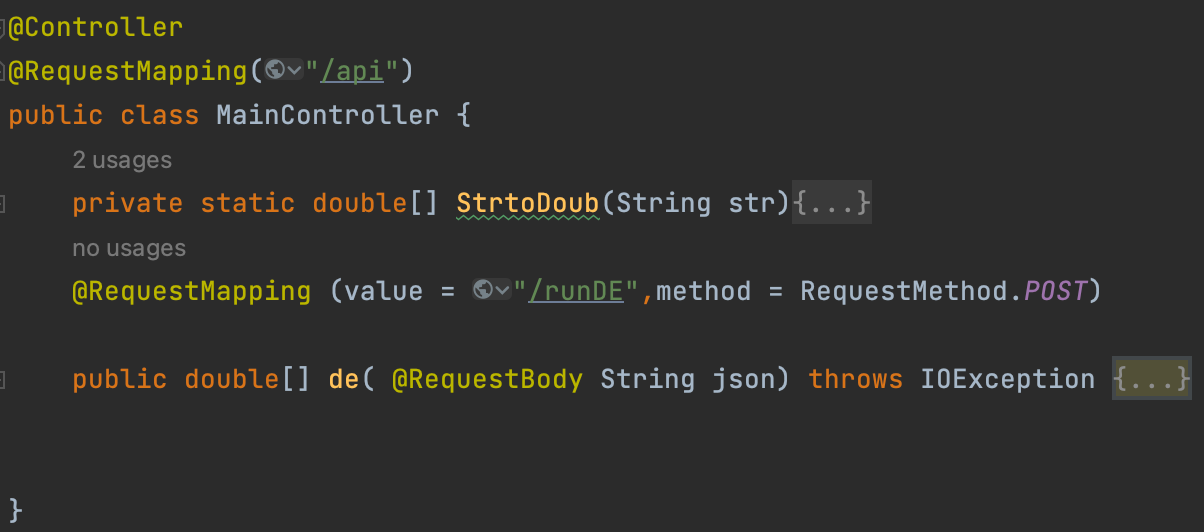
Spring MVC框架的核心部分是控制器（Controller）层，它处理来自客户端的请求并生成相应的响应。在Spring MVC框架中，控制器通过注解或XML配置来实现。

4.2服务器开发

在本设计中，服务器端采用了Spring MVC框架进行开发。具体地，使用了Maven进行项目管理，以DE为例配置文件如下：

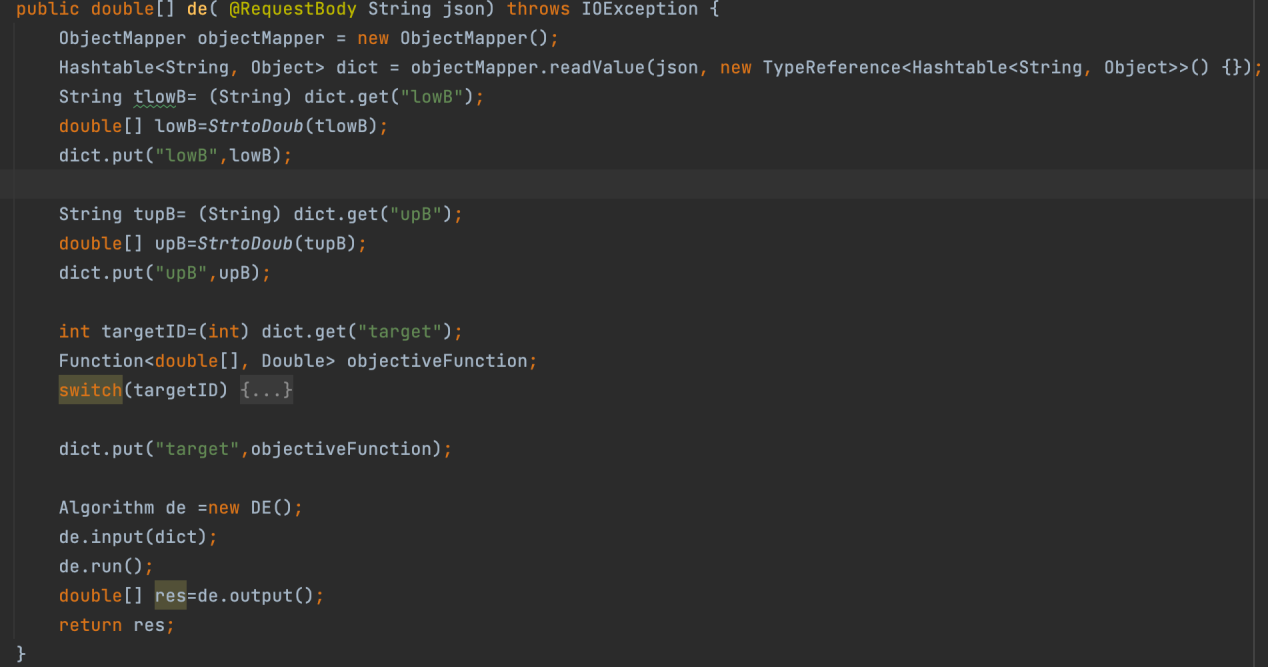


在Spring MVC框架中，控制器通过注解或XML配置来实现。以下是控制器类的代码示例：



在以上代码中，@Controller注解表示该类是一个Spring MVC控制器。@RequestMapping注解用于映射请求路径，请求方法为POST。

通过@RequestMapping注解来定义请求的映射和请求类型。在该代码中，定义了一个MainController类，其中包含了一个静态方法StrtoDoub用于将字符串转换为双精度浮点数数组，并定义了一个de方法，该方法利用@ResponseBody注解来将返回值转换为json格式的字符串返回给客户端，并解析出其中的参数，并调用目标函数进行计算，最后返回计算结果。



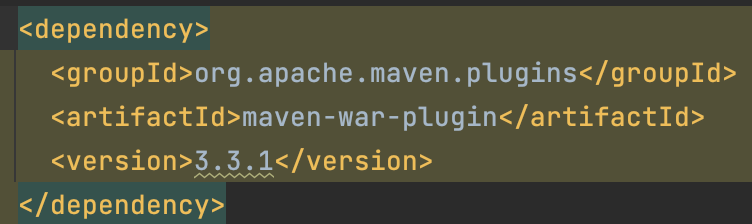
在de方法中，首先使用ObjectMapper（ObjectMapper 是一个 JSON 序列化/反序列化[4]库）将json字符串反序列化成 Hashtable<String, Object> 类型的对象，然后通过Hashtable对象获取各个参数的值，并调用StrtoDoub方法将字符串转换为双精度浮点数数组。接着根据目标函数的类型选择对应的函数进行计算，并将目标函数作为参数传递给DE算法进行计算。最后，将计算结果返回给客户端。

4.3 Tomcat简介

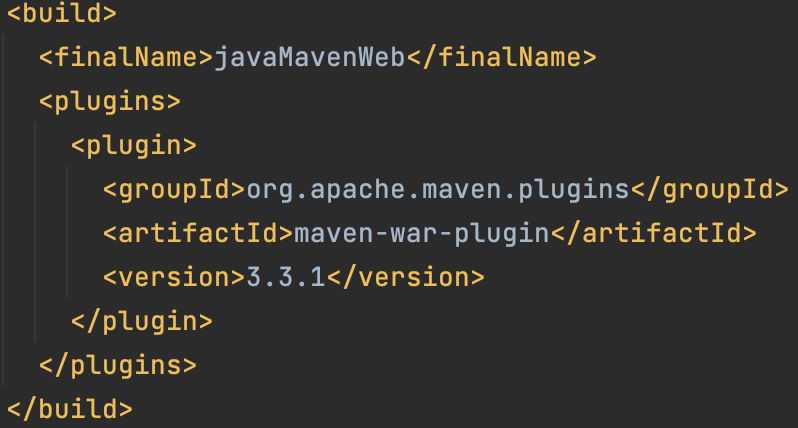
Tomcat是一个开源的Web服务器，也是一个Servlet容器，它能够运行Java Servlet和JavaServer Pages（JSP）等Web应用程序。Tomcat由Apache软件基金会开发和维护，其代码实现了Java Servlet和JavaServer Pages等相关技术的规范。Tomcat的主要特点是开源、轻量级、灵活性高、易于配置和扩展性强。

4.4 Tomcat部署

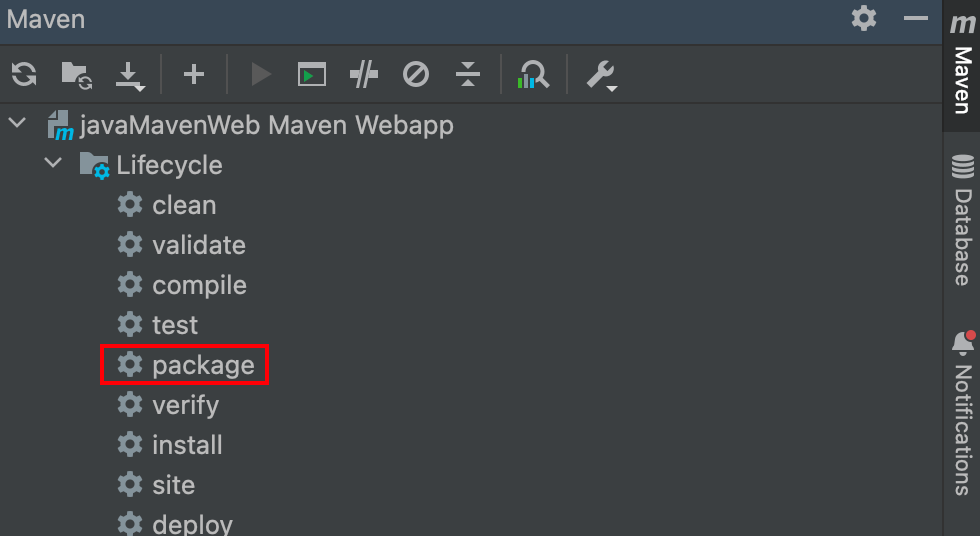
1. 生成war包  
    首先在pom.xml文件中添加生成war包的依赖



然后对build项进行配置



然后双击Maven工具中的package，即可将项目打包成war



1. 部署war包

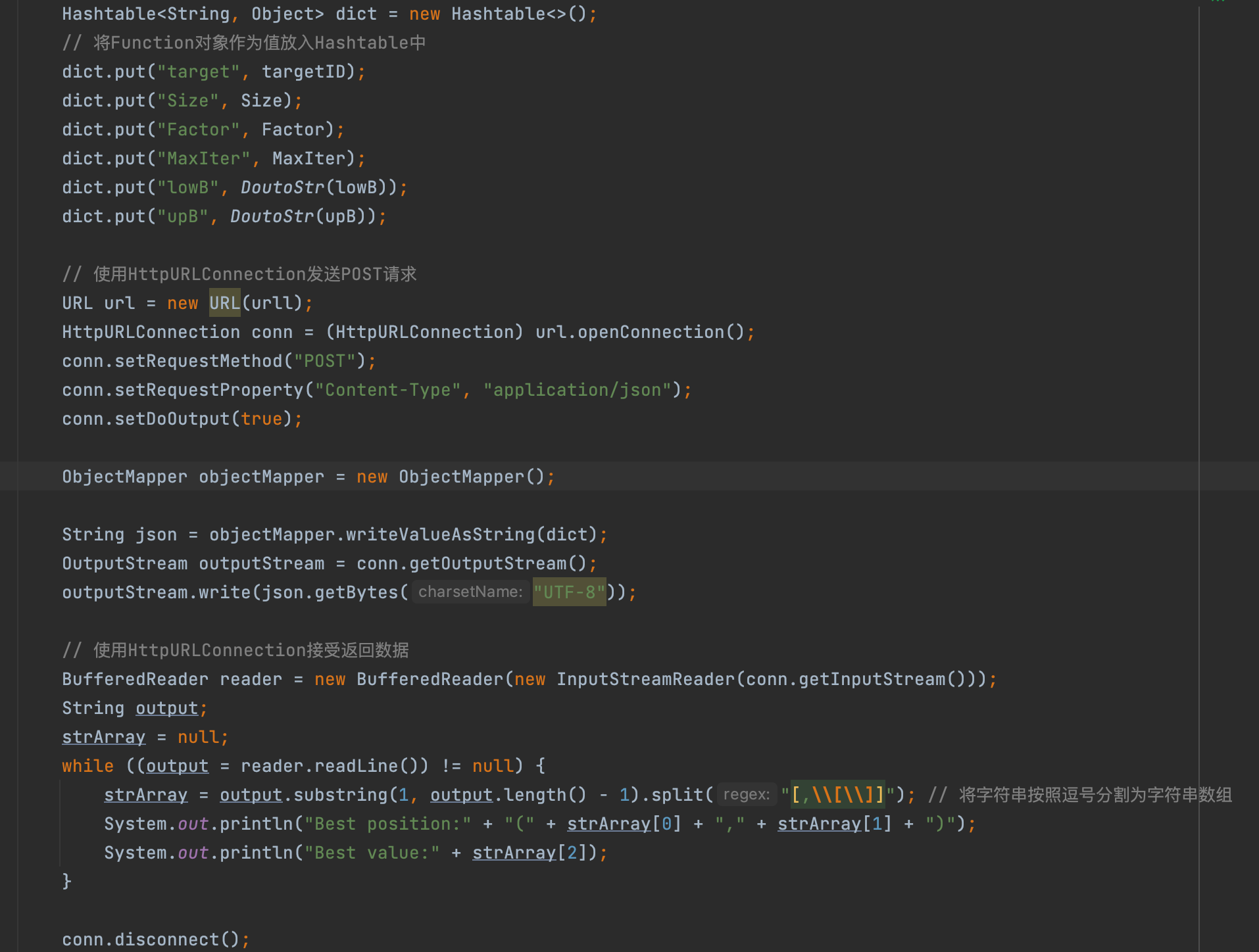
首先将我们编写的服务器端代码打包成一个war包，将该war包放入Tomcat的webapps目录下。然后进入conf文件夹，找到server.xml配置文件，将

<Engine name="Catalina" defaultHost="localhost">中的localhost改为服务器的ip地址、

将<Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">中的localhost改为当前服务器的ip地址即可。最后找到bin目录下的startup.bat文件，运行该文件即可启动Tomcat。

4.5客户端开发

客户端开发是Web服务的一个重要组成部分，其目的是向服务器发起请求，获取返回的数据并进行处理。在本项目中，我们采用了Java编程语言进行客户端开发，使用了HttpURLConnection类与服务器进行通信。核心部分如下：

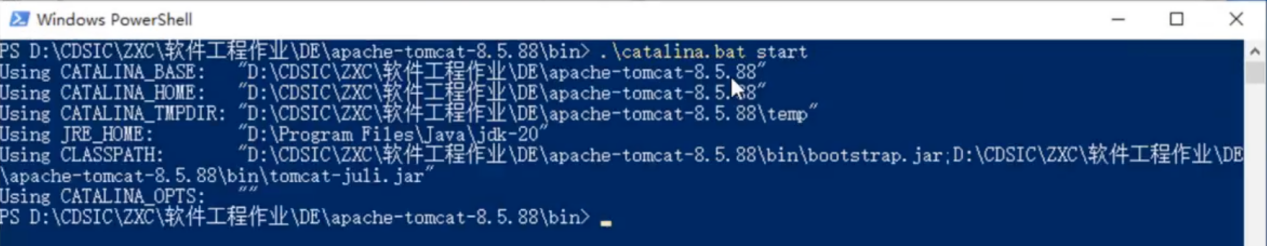


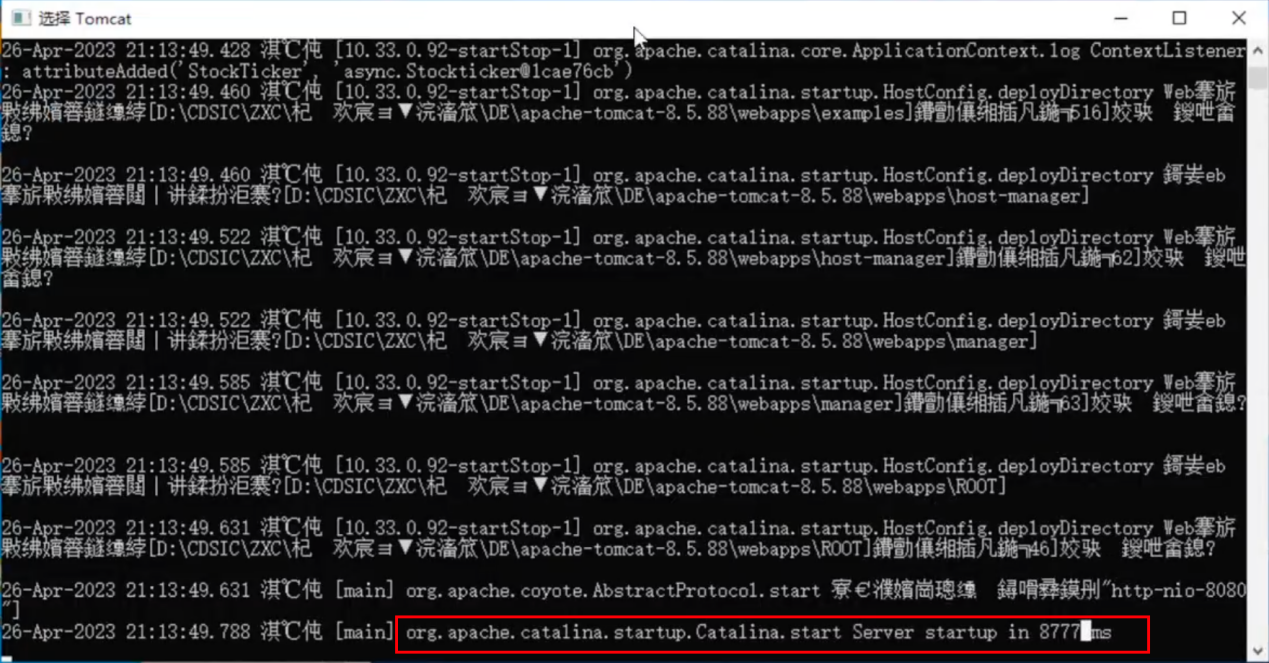
首先，我们需要构造一个URL对象，指定服务器的地址和端口号，使用该URL对象获取HttpURLConnection对象，并设置请求方法为POST，即向服务器提交数据。在此基础上，我们设置请求头的Content-Type为application/json，以便服务器能够正确解析我们提交的数据。接着，我们将待提交的数据Hashtable序列化为String类型数据，并将其写入HttpURLConnection对象的输出流中。

然后，我们可以通过调用HttpURLConnection对象的getInputStream()方法来获取服务器返回的数据。为了方便处理，我们使用BufferedReader类读取服务器返回的数据，并将其转换为字符串。在本项目中，我们将返回的数据解析为一个包含最优位置和最优值的字符串数组。最后，我们关闭连接并断开与服务器的通信。

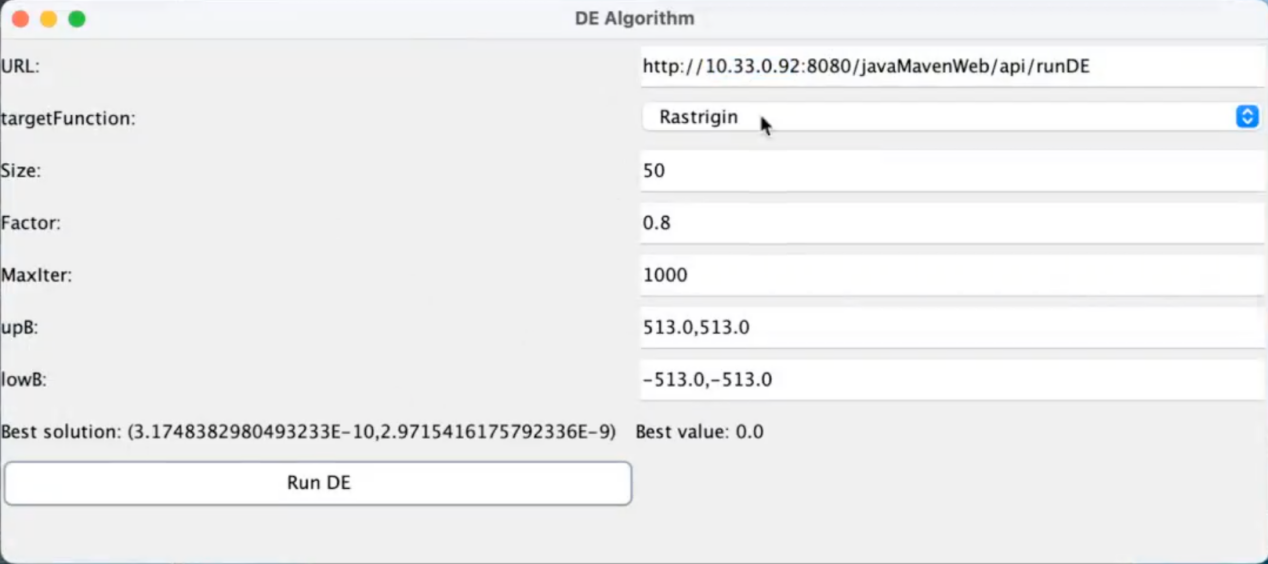
4.6 CS模式测试

首先在两台不同的电脑上对DE和PSO算法进行部署。在windows系统的电脑上进入命令行输入.\catalina.bat start即可对相应的算法进行部署，如下图

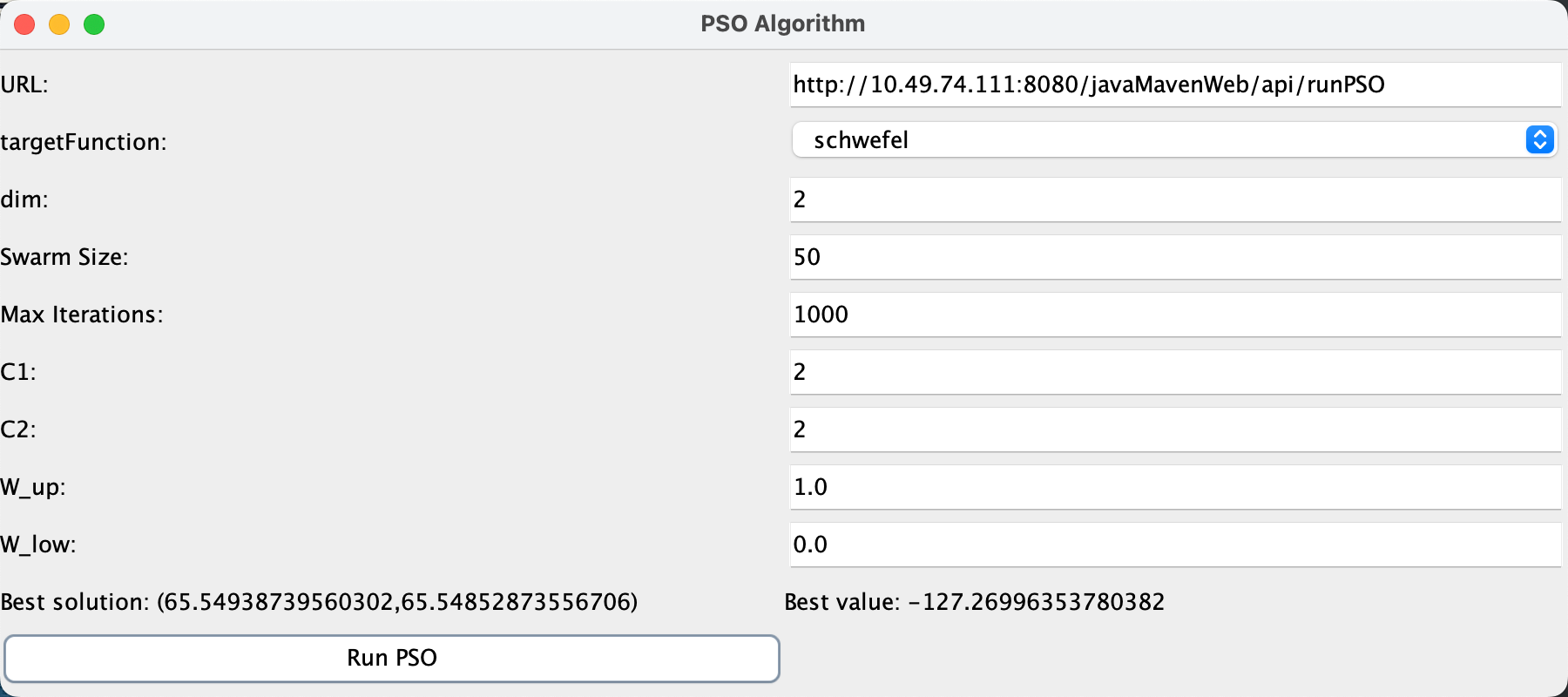




然后打开客户端，下图为客户端可视化界面，第一行输入我们需要访问的地址，然后输入相应的参数，即可对相应的方法进行调用，下图为DE算法的调用结果。



下图为PSO算法的调用结果。



1. 总结

现代软件工程技术不但具有很强的适应能力, 能够适应很多用户的个性化需求, 还具有可修改的特征, 进而快速有效地纠正其中的不足之处, 这对于降低计算机系统软件开发的风险, 提升开发水平都具有积极的意义[5]

本文中提到的面向对象、基于构件和面向 Web 服务的开发方法是现代软件开发中常用的方法，具有以下优点：

面向对象编程思想可以提高代码的可重用性和可维护性，减少代码冗余，同时也有利于程序的扩展和升级。

基于构件的开发方法可以将一个软件系统分解为多个独立的构件，每个构件都实现了特定的功能，这样可以提高系统的可重用性、可维护性和可扩展性。同时，由于构件的独立性，也可以使得不同的开发人员负责不同的构件，提高开发效率和分工协作能力。

面向 Web 服务的开发方法可以将一个软件系统分解为多个独立的服务，每个服务都提供特定的功能，这样可以使得不同的系统之间可以互相调用和交互，提高系统的可复用性和可扩展性。

因此，面向对象、基于构件和面向 Web 服务的开发方法均可以提高软件系统的可重用性、可维护性和可扩展性，同时也可以提高开发效率和分工协作能力。

参考文献

[1] Storn, R., & Price, K. (1997). Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 11(4), 341

[2]Pereira G. Particle swarm optimization[J]. INESCID Inst. Super. Techno Porto Salvo Port, 2011.

[3]杨浩,杨陟卓.基于面向对象编程的代码复用技术[J].信息技术,2017(06):52-57+61.DOI:10.13274/j.cnki.hdzj.2017.06.012.

[4] Lukasz Opyrchal, Atul Prakash. Efficient Object Serialization in Java[EB/OL].[2018-12-13]

1. 蔡贵龙.软件工程技术在计算机系统软件开发中的应用[J].电脑迷,2017(05):8.