

# 复杂优化问题的高效求解软件操作 说明

---

目录

- 一、前言 .....3
  - 1.1 文档说明.....3
  - 1.2 背景 .....3
  - 1.3 目标 .....3
- 二、概要设计.....4
  - 2.1 运行环境.....4
  - 2.2 软件框架及主要功能.....4
- 三、详细设计.....6
  - 3.1 优化问题自定义模块设计.....6
  - 3.2 算法求解模块.....7
  - 3.3 指标计算模块和运行结果展示模块.....11
- 四、软件使用说明（操作步骤） .....13

# 一、前言

## 1.1 文档说明

本文档旨在介绍复杂优化问题的高效求解软件 V1.0 的功能需求和约束等内容，以便用户和开发人员对本软件的运行环境、功能和性能需求等的初始规定达成共识。同时，本文档还对软件系统的操作界面和过程进行描述，为系统的详细设计和开发提供依据。

## 1.2 背景

目标优化问题是科研和实际工程中应用非常广泛的一类问题，诸如地图的最优路线选择、能源系统中的多目标权衡优化、机器学习中的下游任务优化器等等，都直接或间接地可以被建模为目标优化问题。目标优化问题根据目标数的不同，可以分为单目标优化问题（目标数为 1）、多目标优化问题（目标数为 2 或 3）以及超多目标优化问题（目标数为 4 及以上）。对于决策向量为实数的情形，可以称该目标优化问题为连续型目标优化问题。通常，由于问题的困难性和复杂性，往往无法使用传统的数学方法进行求解。相反，进化算法可以很好的解决这类问题。目前已经有很多进化算法被提出，用于解决各类目标优化问题，进化算法设计的好坏根本地影响了解决问题的效率和质量。设计一个通用、方便的目标优化问题求解器，提供强大的求解能力，可以让科研人员和工程师快速地求解他们需要优化的问题。此外，提供图形化界面和运行过程结果的展示，可以方便科研人员或工程师查看求解效果。

## 1.3 目标

本软件是一个相对通用的目标优化求解器，可以方便地添加多种目标优化问题，软件将自动调用相应的进化算法进行求解，并提供图形化界面和运行过程与结果的展示，方便科研人员或工程师查看求解的效果。本软件基于 windows 的 MATLAB 平台进行开发，设计了一个相对通用的目标优化求解器，可以方便地添加多种目标优化问题，软件将自动调用相应的算法进行求解，并提供图形化界面和运行过程与结果的展示，方便科研人员或工程师查看求解的效果。此外，本软件应该无需额外安装其他库，仅需 MATLAB 自带的数学库、UI 库等即可，方便用户一键使用。

## 二、概要设计

### 2.1 运行环境

#### 2.1.1 硬件设备

CPU：3.0GHz 及以上

RAM：4GB 及以上

硬盘：10GB 及以上

#### 2.1.2 软件环境需求

操作系统：Win10 及以上

运行环境：MATLAB R2020b 及以上

### 2.2 软件框架及主要功能

复杂优化问题的高效求解软件主要包含优化问题自定义模块、算法求解模块、指标计算模块、运行结果展示模块等功能系统主要功能模块及其关系如图 2-1 所示。整个程序通过模块化设计以降低整个程序的耦合性，使得拓展性与稳健性得到保障。

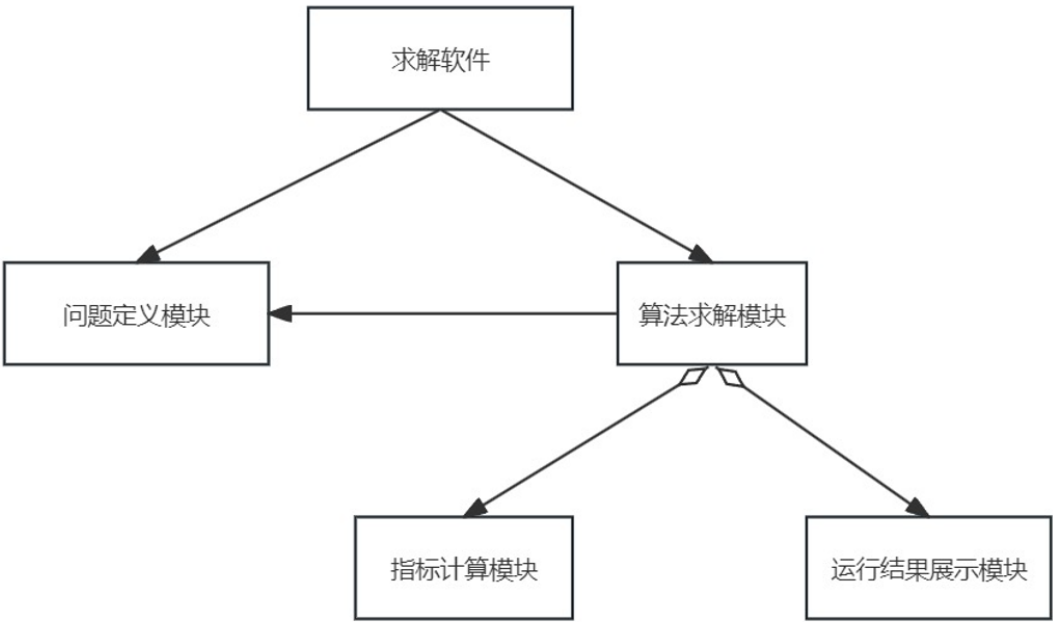


图 2-1 复杂优化问题的高效求解软件功能模块

问题定义模块允许用户以 MATLAB 文件（后缀名为.m）的形式，按照规范定义待求解的目标优化问题。这个规范可以使得系统自动读取用户自定义的问题，分析问题属性。在 UI 界面上，用户可以看到先前编写的待求解问题。当用户选择待求解问题后，系统自动识别问题的属性，并调用算法求解模块。算法求解模块调用相应的进化算法，并根据选择的问题进行求解。本系统中，进化算法包含三类：无约束单目标进化算法、多约束单目标进化算法、多目标进化算法。这三类算法的具体实现分别采用了作者提出的三个基于增益策略的进化算法。算法求解过程中，会调用运行结果展示模块，展示算法运行过程中某一时刻的运行快照。算法求解结束后，指标计算模块会计算必要的指标，这一般在多目标优化问题上会调用得更多。之后，调用运行结果展示模块，展示最终的运行情况，包括最终种群、最优解（集）、指标值等等。

程序的总体流程图如图 2-2 所示：

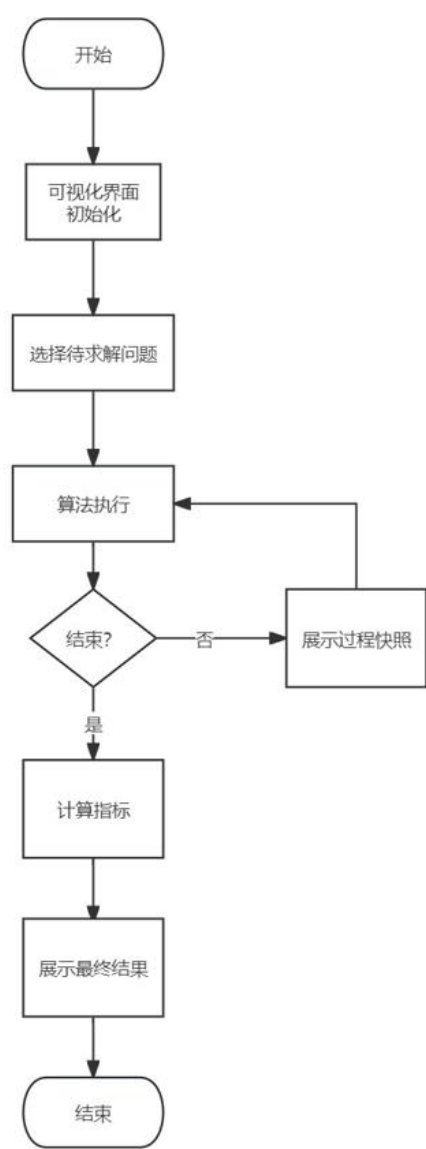


图 2-2 复杂优化问题的高效求解软件流程图

### 三、详细设计

#### 3.1 优化问题自定义模块设计

本软件允许用户按照规定的格式，自定义待优化问题集。设计的思路是定义了一个 **PROBLEM** 超类。该父类定义了一个优化问题需要的基本属性和方法。表 3-1 和表 3-2 分别展示了部分属性和方法：

属性	解释
N	种群大小
M	目标数
D	决策向量维度
FE	已消耗评估次数
maxFE	最大评估次数
Lower	变量下界
Upper	变量上界
Optimum	问题的最优值或最优解集

表 3-1 PROBLEM 类部分属性

方法	解释
Setting	设置问题默认属性
Initial	初始化种群
Evaluation	评估个体（或种群）
Repair	修复无效解
CalObj	计算目标值
CalCon	计算约束违反值
CalMetric	计算指标值
DisplayDec	显示种群的决策向量
DisplayObj	显示种群的目标向量

表 3-2 PROBLEM 类部分方法

其中，每个具体的 **PROBLEM** 子类都必须重写 **Setting()** 方法和 **CalObj()** 方法，每个具体的问题都有独特的属性和计算目标值的方式。

通过继承 **PROBLEM** 父类，用户只要重写了 **Setting** 方法和 **CalObj** 方法，并可选择重写 **CalCon** 等方法，就可以很方便地定义一个新的问题。在设计上，属于解耦的一种方式，问题自定义模块只要定义必要的和问题有关的属性和方法，而不需要管如何被算法调用和执行。

以下展示 CEC2010 F1 的问题定义文件 CEC2010\_F1.m

```
classdef CEC2010_F1 < PROBLEM
% <single> <real> <constrained>
    methods
```

```

function Setting(obj)
    obj.M = 1;
    if isempty(obj.D); obj.D = 10; end
    obj.lower = zeros(1,obj.D);
    obj.upper = zeros(1,obj.D) + 10;
    obj.encoding = ones(1,obj.D);
end
function PopObj = CalObj(obj,PopDec)
    Z = PopDec - repmat(obj.O(1:size(PopDec,2)),size(PopDec,1),1);
    PopObj =
    -abs(sum((cos(Z).^4),2)-2*prod((cos(Z).^2),2))./sqrt(sum(repmat(1:size(Z,2),size(Z,1),1)
    ).*(Z.^2),2)));
end
function PopCon = CalCon(obj,PopDec)
    Z = PopDec - repmat(obj.O(1:size(PopDec,2)),size(PopDec,1),1);
    PopCon(:,1) = 0.75 - prod(Z,2);
    PopCon(:,2) = sum(Z,2) - 7.5*size(Z,2);
end
end
end

```

第二行的注释中，有三个标签，分别为<single> <real> <constrained>。这分别表示该问题集是单目标优化问题、连续型和多约束，总的来说就是多约束连续单目标优化问题。该标签会被系统读取，然后调用相应合适的算法进行求解。

## 3.2 算法求解模块

### a) 算法执行模块流程（与 PROBLEM 的交互设计）

前面已说明了 PROBLEM 类的设计和组成。在算法函数中，只需将 PROBLEM 类对象作为函数参数，即可访问具体问题的设置、属性和方法。假设问题对象名为 p，那么可以调用 p.Initial() 初始化种群；可以根据 p.FE 与 p.maxFE 的比较判断算法是否应该停止；可以使用 p.calObj()、p.Evaluation() 对个体或种群进行计算；可以使用 p.calCon() 计算个体约束违反值；可以根据 p.calMetric() 计算指标，然后使用 p.DisplayDec() 和 p.DisplayObj() 显示当前种群的决策向量和目标向量。

### b) 无约束单目标进化算法

本软件采用的无约束单目标进化算法是 DEANC/GAIN 算法。这是作者设计的增益策略，结合李学强[1]等设计的 DEA/NC 算法，得到的基于增益辅助的 DEANC/GAIN 算法。种群中每个个体都要计算增益，增益定义为  $|x_{t+1} - x_t| - |x_t - x_{t-1}|$ 。当增益值大于 0 时，认为该个体有较好的发展潜力，因此分配更多机会搜索，生成优质个体。当增益值小于等于 0 时，认为该个体的发展潜力一般或不足，因此将搜索机会分配给增益值大于 0 的个体进行搜索。基于这个

策略的改进，DEANC/GAIN 算法比原始的 DEA/NC 算法将得到更出色的优化结果。

图 4-1 展示了该算法的流程图。

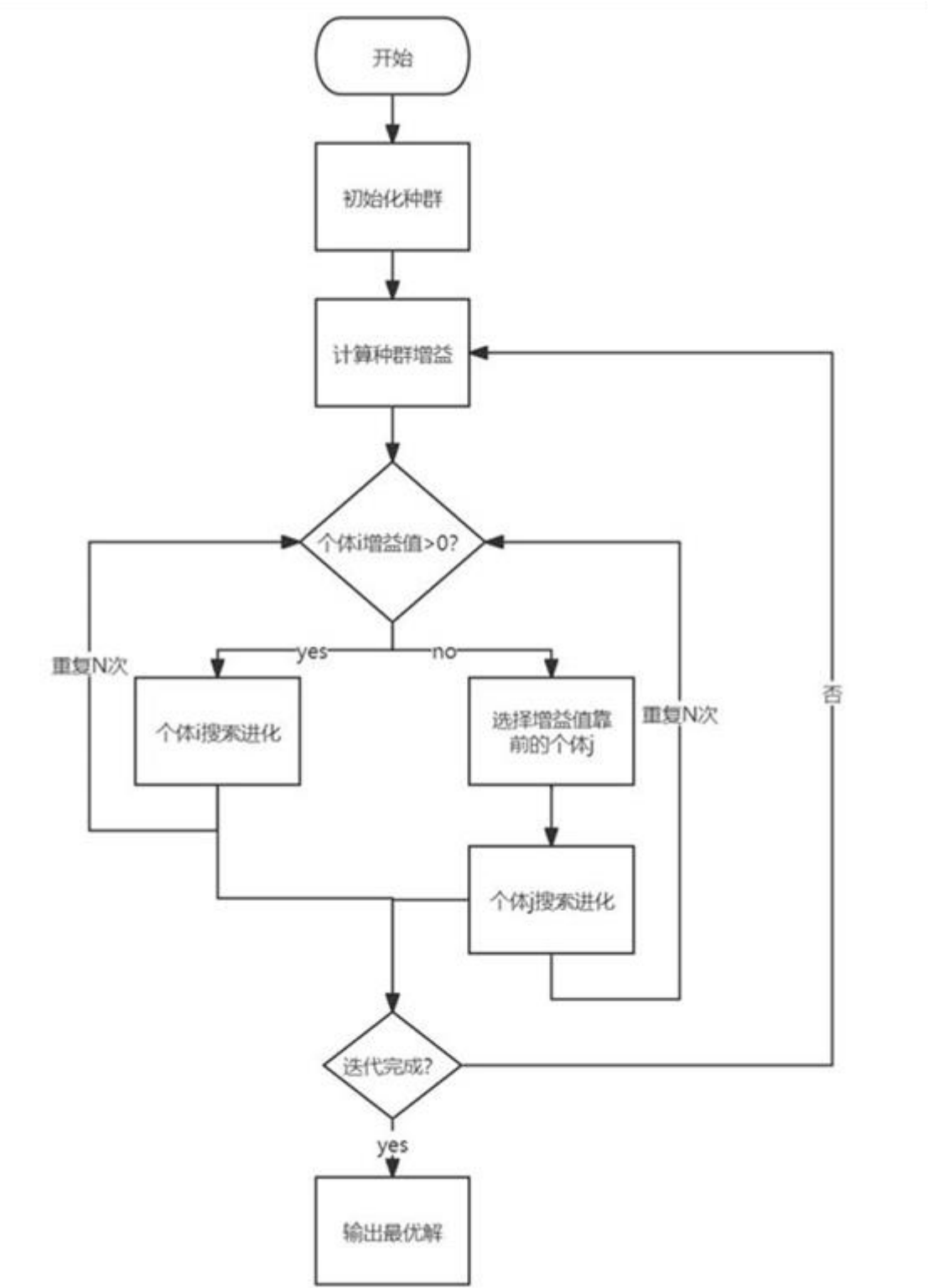


图 4-1 DEANC/GAIN 算法流程

c) 多约束单目标进化算法

本软件采用的多约束单目标进化算法是 DASDE/GAIN 算法。这是作者设计的增益策略，结合 Xu[2]等设计的 DEA/DE 算法，得到的基于增益辅助的 DASDE/GAIN 算法。种群中每个个体都要



计算增益，目标增益定义为  $(x_t - x_{t+1}) - (x_{t-1} - x_t)$ ，约束增益定义为  $(Con_t - Con_{t+1}) - (Con_{t-1} - Con_t)$ 。对于目标增益和约束增益综合看待，当增益值大于 0 时，认为该个体有较好的发展潜力，因此分配更多机会搜索，生成优质个体。当增益值小于等于 0 时，认为该个体的发展潜力一般或不足，因此将搜索机会分配给增益值大于 0 的个体进行搜索。基于这个策略的改进，DASDE/GAIN 算法比原始的 DEA/DE 算法将得到更出色的优化结果。

对于当前种群  $p$ ，分为可行解  $p1$  和不可行解  $p2$  两个子种群

1.对于可行解  $p1$ ，因为约束违反值都为 0，所以通过目标增益指导搜索。具体来说，如果当前个体的目标增益  $\geq 0$ ，则什么都不需要做，按照原算法进行即可；否则，选目标增益排名靠前（优先目标增益排序，然后再是适应值排序）的个体代替当前个体进行搜索进化。

2.对于不可行解  $p2$ ，通过约束增益和目标增益共同指导搜索。具体来说，如果当前个体的约束增益  $\geq 0$ ，则什么都不需要做，按照原算法进行即可；否则，按照约束增益排序，即约束增益大于 0 的排在最前面；然后按照目标增益和适应值综合排序。选该排名较前的个体代替当前个体进行搜索进化。

两个种群互不干扰，即  $p1$  的个体不会用于  $p2$ ，反之亦然。

上述两点可以总结成以下伪代码。

```
sorted_feasible 表示对  $p1$  排序后的下标
counter_feasible = 0
sorted_infeasible 表示对  $p2$  排序后的下标
counter_infeasible = 0
for i = 1 : popsize
    selected_index = i; // 当前选择去进化搜索的个体，首先默认为 i
    if  $X_i$  是可行解
        if gain_obj( $X_i$ ) < 0 // 目标增益为负数
            counter_feasible++;
            selected_index = sorted_feasible(counter_feasible)
        end
    end
    elseif  $X_i$  是不可行解
        if gain_con( $X_i$ ) < 0 // 约束增益为负数
            counter_infeasible++;
            selected_index = sorted_infeasible(counter_infeasible)
        end
    end
end
```

```
Xi = evolution(selected_index); // 使用当代选择的个体进行搜索进化，然后更新种群
end
```

流程图如 4-2 所示

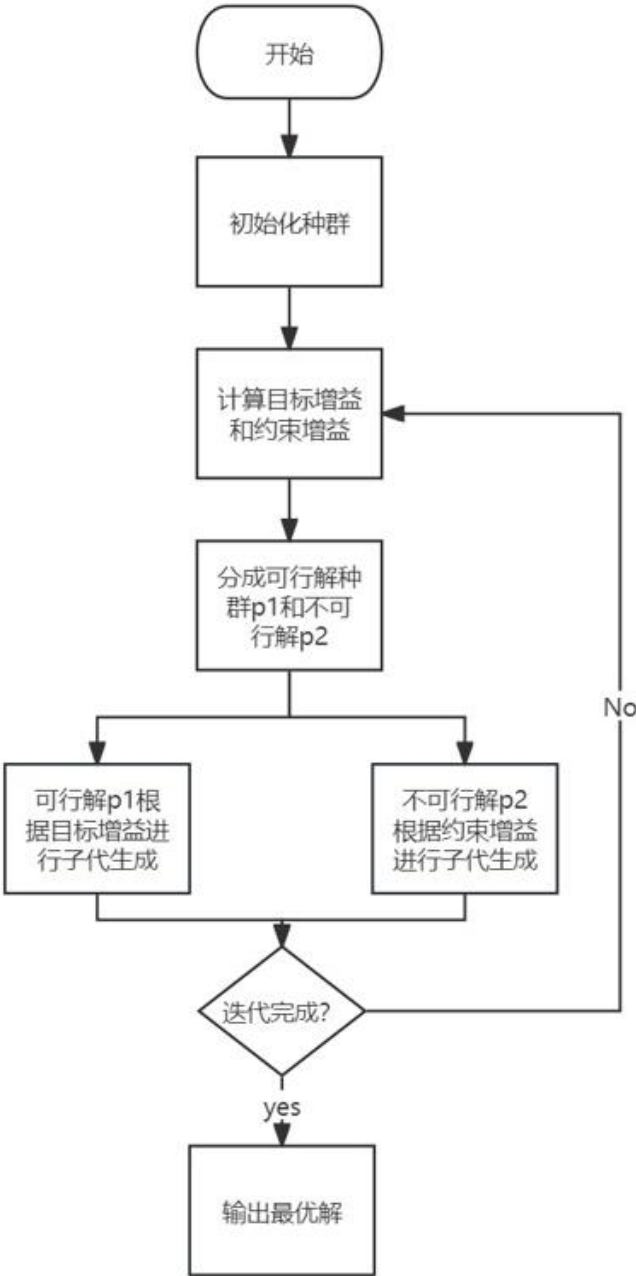


图 4-2 DASDE/GAIN 算法流程

d) 多目标进化算法

本软件采用的多目标进化算法是 PaRPEA/GAIN 算法。这是作者设计的增益策略，结合 Xiang[3] 等设计的 PaRPEA 算法，得到的基于增益辅助的 PaRPEA/GAIN 算法。种群中每个个体都要计算增益，目标增益定义分为两种：① $gain = \text{SUM}((f(x_t) - f(x_{t+1})) - (f(x_{t-1}) - f(x_t)))$  ② $gain = (d(x_t) - d(x_{t+1})) - (d(x_{t-1}) - d(x_t))$ ， $d(x)$ 表示个体  $x$  在目标空间距离当前参考点的距离。当增益值大于 0 时，认为该个体有较好的发展潜力，因此分配更多机会搜索，生成优质个体。当增益值小于等于 0 时，认为该个体的发展潜力一般或不足，因此将搜索机会分配给增益值大于 0 的个体进行搜

索。基于这个策略的改进，PaRPEA/GAIN 算法比原始的 PaRPEA 算法将得到更出色的优化结果。

本小节的流程图和图 4-1 类似，这里不再赘述，只是增益的计算方式不同。

### 3.3 指标计算模块和运行结果展示模块

图 2-1 功能模块关系图中可以看到，指标计算模块和运行结果展示模块被设计为与算法模块是聚合的关系，意味着这两个模块在算法执行时被调用。

指标计算模块一般用于多目标优化问题中。每一个指标都是一个单独的 MATLAB 文件，接口如下：

```
function score = Indicator(Population, optimum)
```

其中，Indicator 可以是任何具体的性能指标名。Population 是一个种群，一般用于获取最终种群获得的最优值（或最优解集）；optimum 是真实的最优值。例如，IGD 的计算如下所示：

```
PopObj = Population.best.objs;
function score = IGD(Population,optimum)
if size(PopObj,2) ~= size(optimum,2)
score = nan;
else
score = mean(min(pdist2(optimum,PopObj),[],2));
end
end
```

在算法执行后，调用某一个指标函数，就可以获得对应的指标值。

在算法每一代执行完毕后，都会调用运行结果展示模块。具体的操作是会调用一个函数句柄 outputFcn。该句柄将运行的快照显示在 UI 上。

```
function DefaultOutput(Algorithm,Problem)
clc; fprintf('%s on %d-objective %d-variable %s (%6.2f%%), %.2fs\n',class(Algorithm),Problem.M,Problem.D,class(Problem),Problem.FE/Problem.maxFE*100,Algorithm.metric.runtime);
if Problem.FE >= Problem.maxFE
if Algorithm.save < 0
if Problem.M > 1
Population = Algorithm.result{end};
if length(Population) >= size(Problem.optimum,1); name = 'HV'; else; name = 'IGD';
end
value = Algorithm.CalMetric(name);
figure('NumberTitle','off','Name',sprintf('%s : %.4e\nRuntime : %.2fs',name,value(end),Algorithm.CalMetric('runtime')));
title(sprintf('%s on %s',class(Algorithm),class(Problem)), 'Interpreter','none');
top = uimenu(gcf,'Label','Data source');
g = uimenu(top,'Label','Population (obj.)','CallBack',{@(h,~,Pro,P)eval('Draw(gca);Pro.DrawObj(P);cb_menu(h);'),Problem,Population});
uimenu(top,'Label','Population (dec.)','CallBack',{@(h,~,Pro,P)eval('Draw(gca);Pro.DrawDec(P);cb_menu(h);'),Problem,Population});
uimenu(top,'Label','True Pareto front','CallBack',{@(h,~,P)eval('Draw(gca);Draw(P,{'\it f\rm_1','\it f\rm_2','\it f\rm_3'});cb_menu(h);'),Problem.optimum});
cellfun(@(s)uimenu(top,'Label',s,'CallBack',{@(h,~,A)eval('Draw(gca);Draw(A.CalMetric(h.Label),'-k','LineWidth',1.5,'MarkerSize',10,{'Number of function evaluations',strrep(h.Label,'_',''),''),[]});cb_menu(h);'),Algorithm),{'IGD','HV','GD','Feasible_rate'});
set(top.Children(4),'Separator','on');
```

```

        g.Callback{1}(g,[],Problem,Population);
    else
        best = Algorithm.CalMetric('Min_value');
        if isempty(best); best = nan; end
        figure('NumberTitle','off','Name',sprintf('Min value : %.4e
Runtime : %.2fs',best(end),Algorithm.CalMetric('runtime')));
        title(sprintf('%s
on %s',class(Algorithm),class(Problem)), 'Interpreter', 'none');
        top = uimenu(gcf, 'Label', 'Data source');
        uimenu(top, 'Label', 'Population
(dec.)', 'Callback', {@(h,~,Pro,P)eval('Draw(gca);Pro.DrawDec(P);cb_menu(h);')}, Problem, Algorit
hm.result{end}));

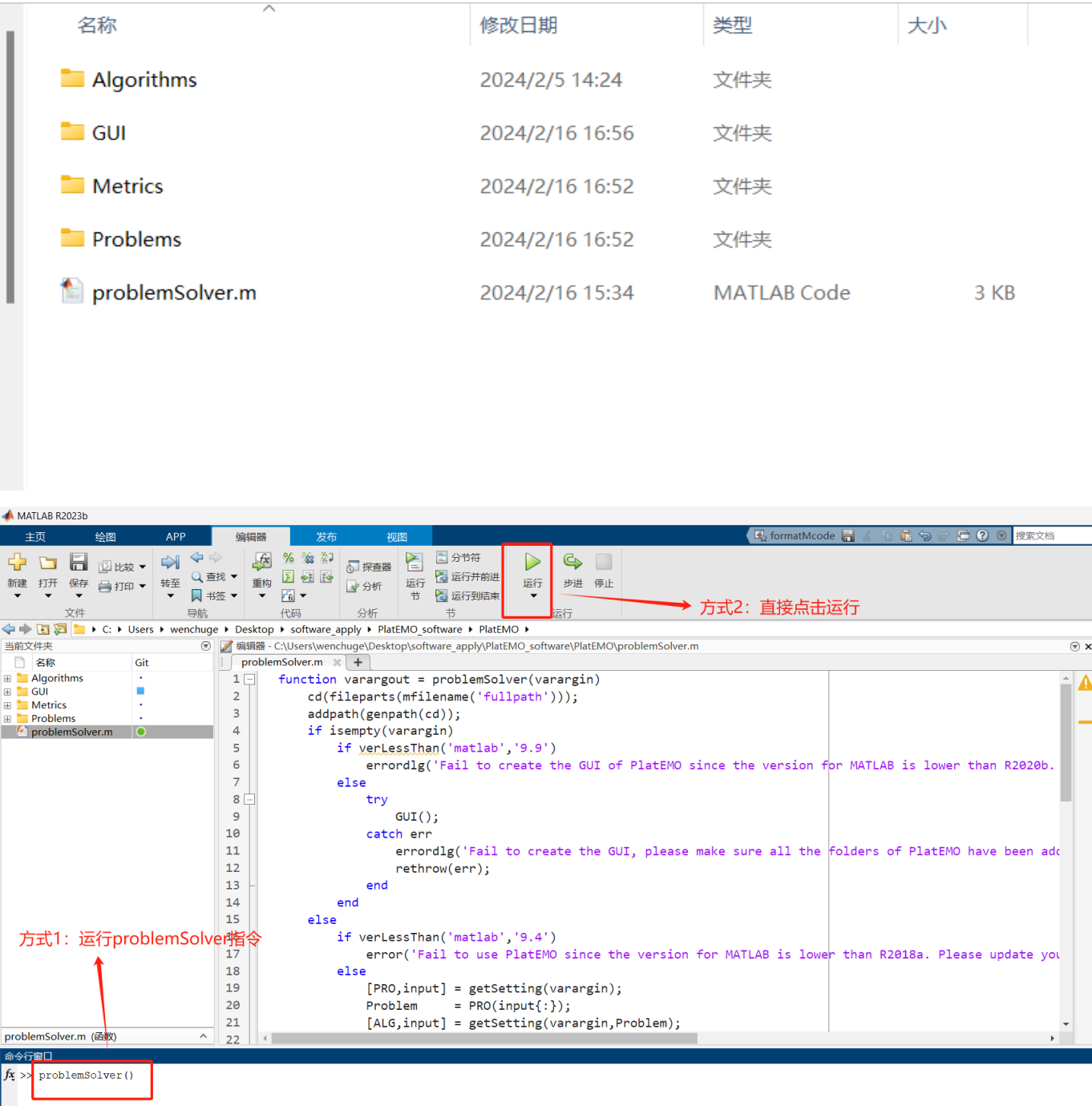
cellfun(@(s)uimenu(top, 'Label', s, 'Callback', {@(h,~,A)eval('Draw(gca);Draw(A.CalMetric(h.Labe
l),'-k.'', 'LineWidth', 1.5, 'MarkerSize', 10, {'Number of function
evaluations', strrep(h.Label, '_ ', '
'),[]});cb_menu(h);')}, Algorithm)), {'Min_value', 'Feasible_rate'});
        set(top.Children(2), 'Separator', 'on');
        top.Children(2).Callback{1}(top.Children(2), [], Algorithm);
    end
elseif Algorithm.save > 0
    folder = fullfile('Data', class(Algorithm));
    [~,~] = mkdir(folder);
    file =
fullfile(folder, sprintf('%s_%s_M%d_D%d_', class(Algorithm), class(Problem), Problem.M, Problem.D
));
    runNo = 1;
    while exist([file, num2str(runNo), '.mat'], 'file') == 2
        runNo = runNo + 1;
    end
    result = Algorithm.result;
    metric = Algorithm.metric;
    save([file, num2str(runNo), '.mat'], 'result', 'metric');
end
end
end

```

四、 软件使用说明（操作步骤）

步骤 1：登录、打开界面

在 matlabR2020b 及以上版本中打开项目文件夹（含有 problemSolver.m 的目录下），在命令行窗口运行指令：problemSolver( )，或直接打开 problemSolver.m 文件，点击 matlab 的运行按钮，即可打开软件初始界面。如下图所示



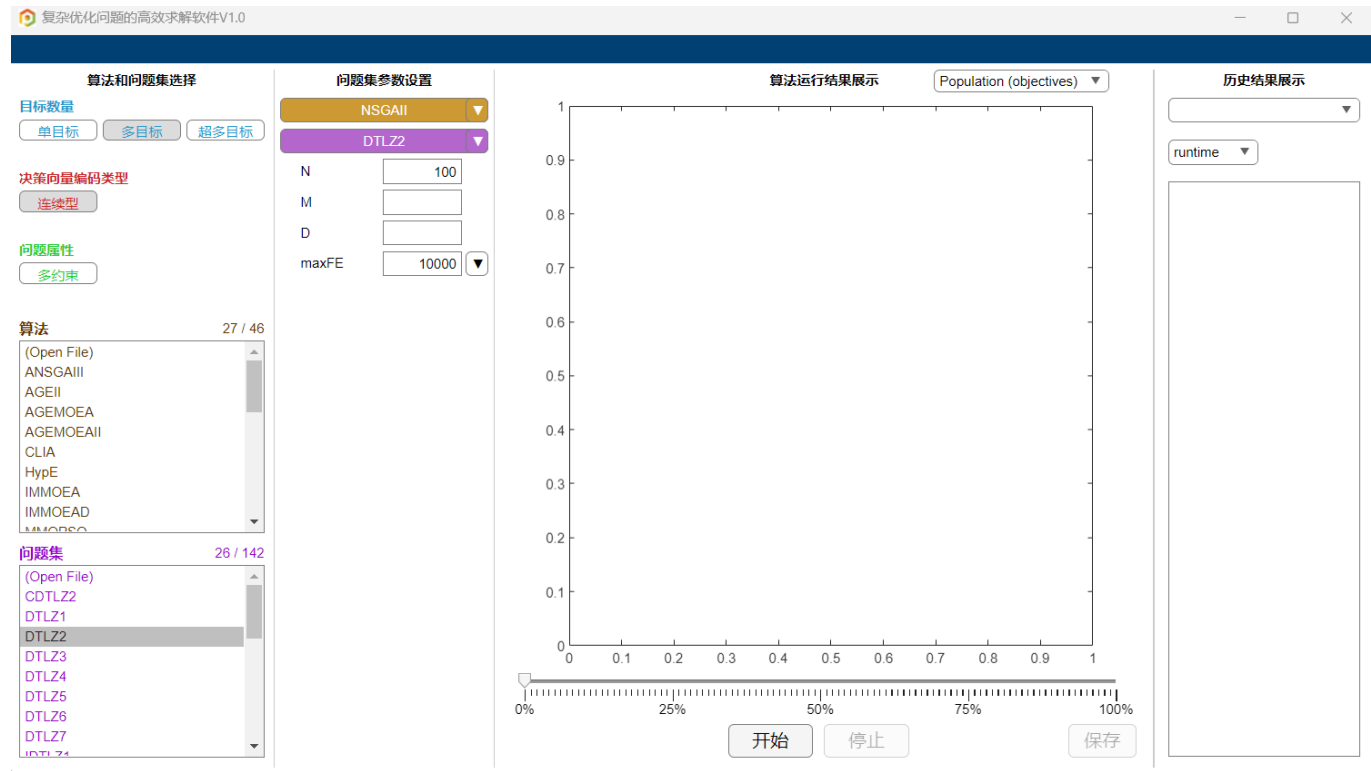


图 4-1 软件初始界面

步骤 2：选择问题集

首先选择目标数量（单目标、多目标、超多目标），选择是否为多约束，然后在问题集列表（界面第一栏左下方）中选择用户自定义的问题集，此时界面第二栏就会变成被选择的问题集，如下图所示。接下来进入步骤 3。

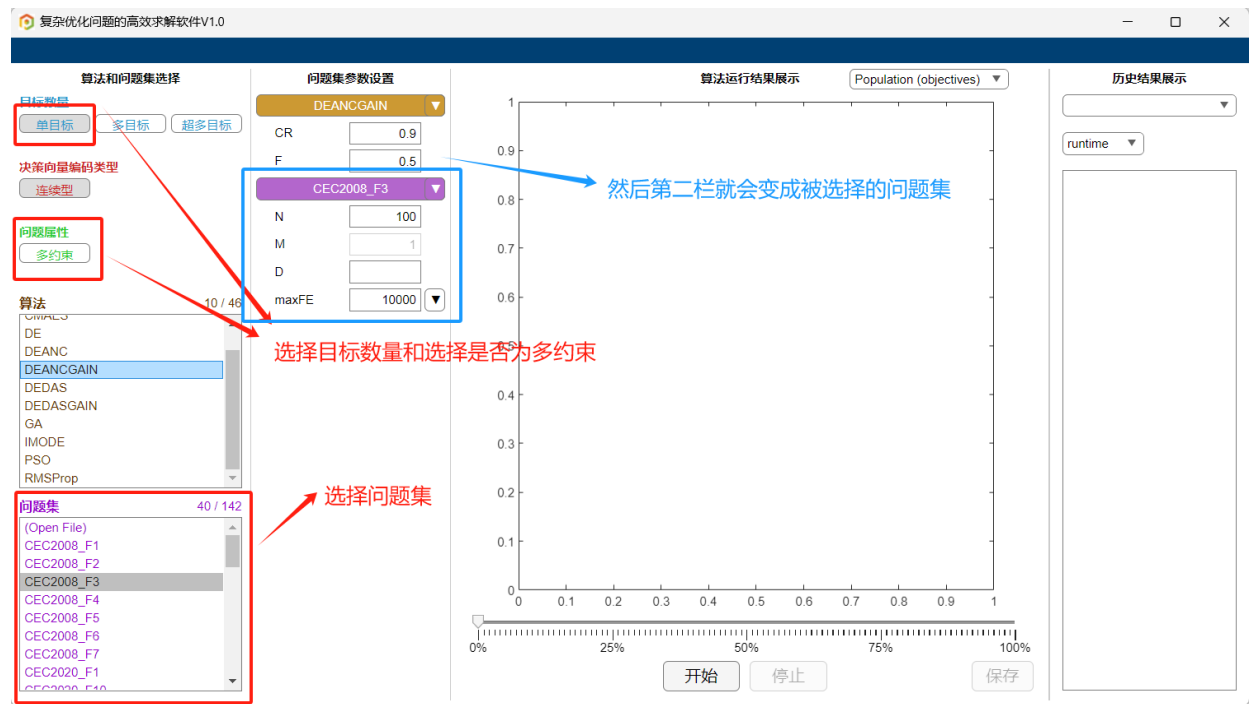


图 4-2 选择问题集

步骤 3：设置问题集参数

输入问题集的设置参数，如种群大小  $N$ 、问题维度  $D$  和最大评估次数  $\text{maxFE}$ 。也可以选择不填，那么将会使用软件默认的设置参数。如下图所示

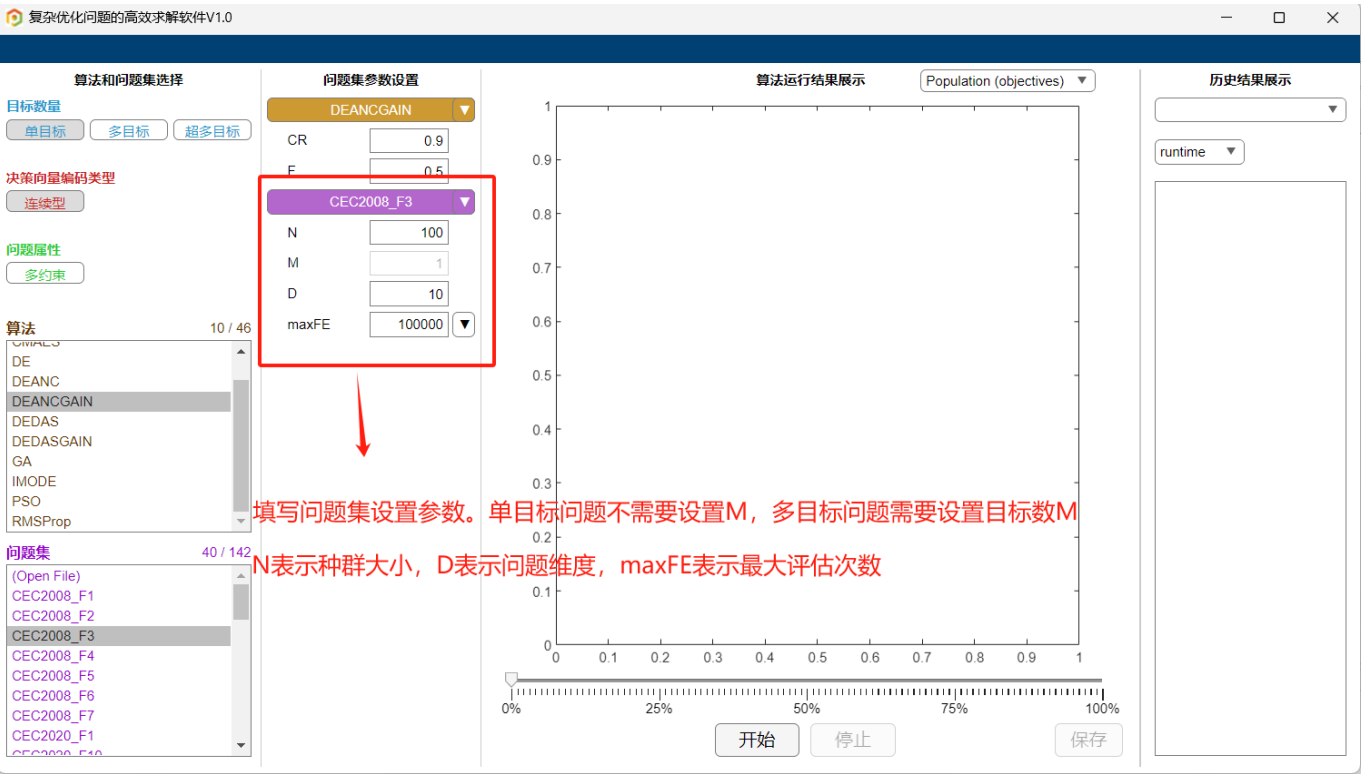
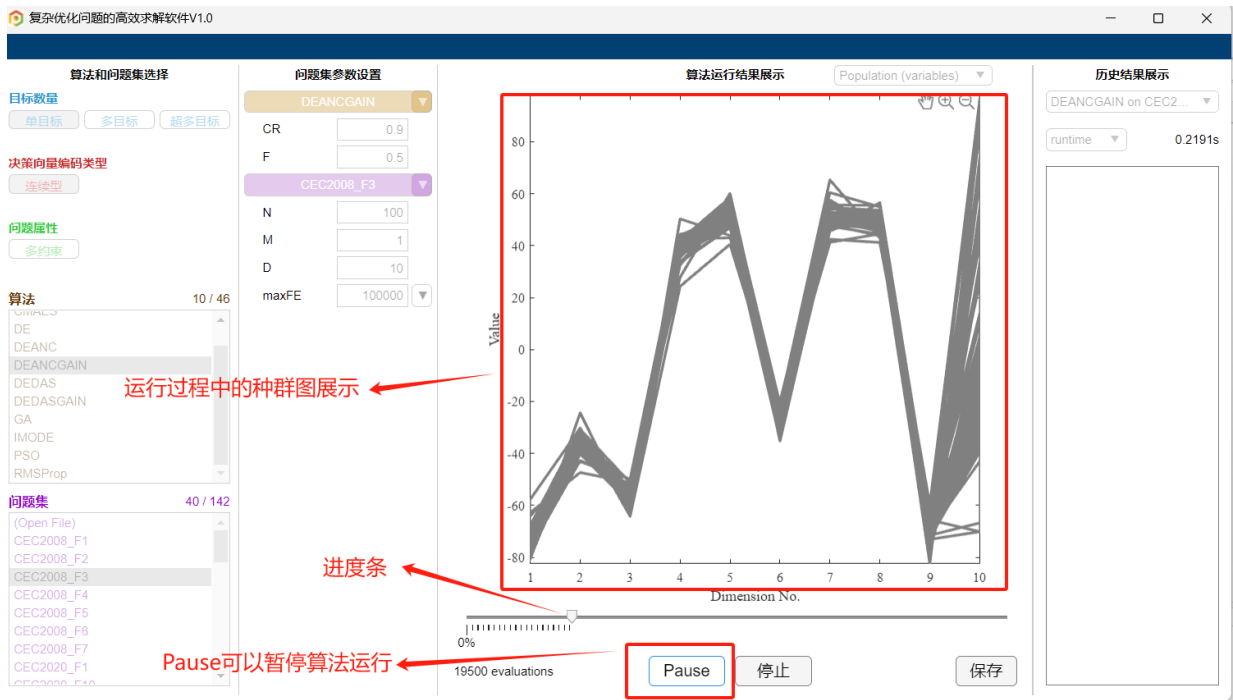
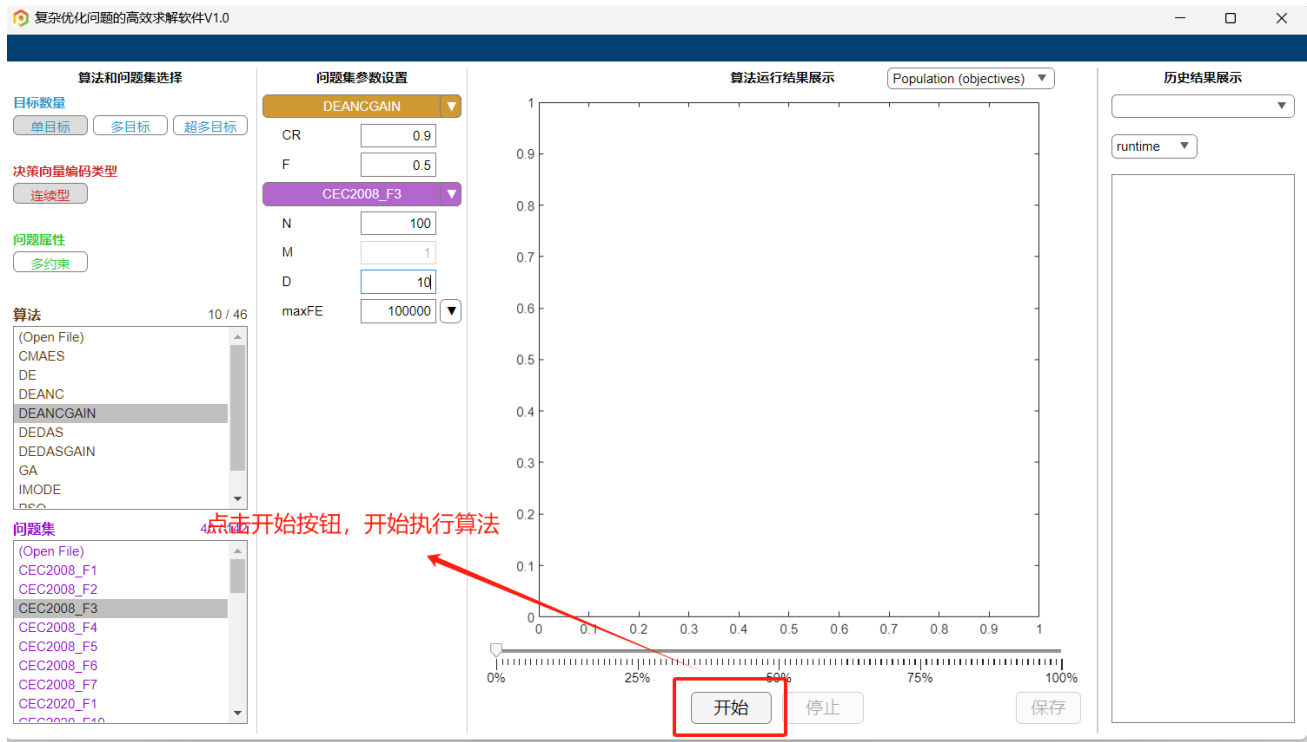


图 4-3 问题集参数设置

步骤 4：算法开始、暂停、停止、运行过程结果展示

点击“开始”按钮，算法开始运行。期间，可以看到算法运行过程中种群的变化过程。可以通过“Pause”按钮暂停算法运行。可以点击“Continue”按钮继续算法运行。可以通过“停止”按钮，停止算法的运行。如下图组所示





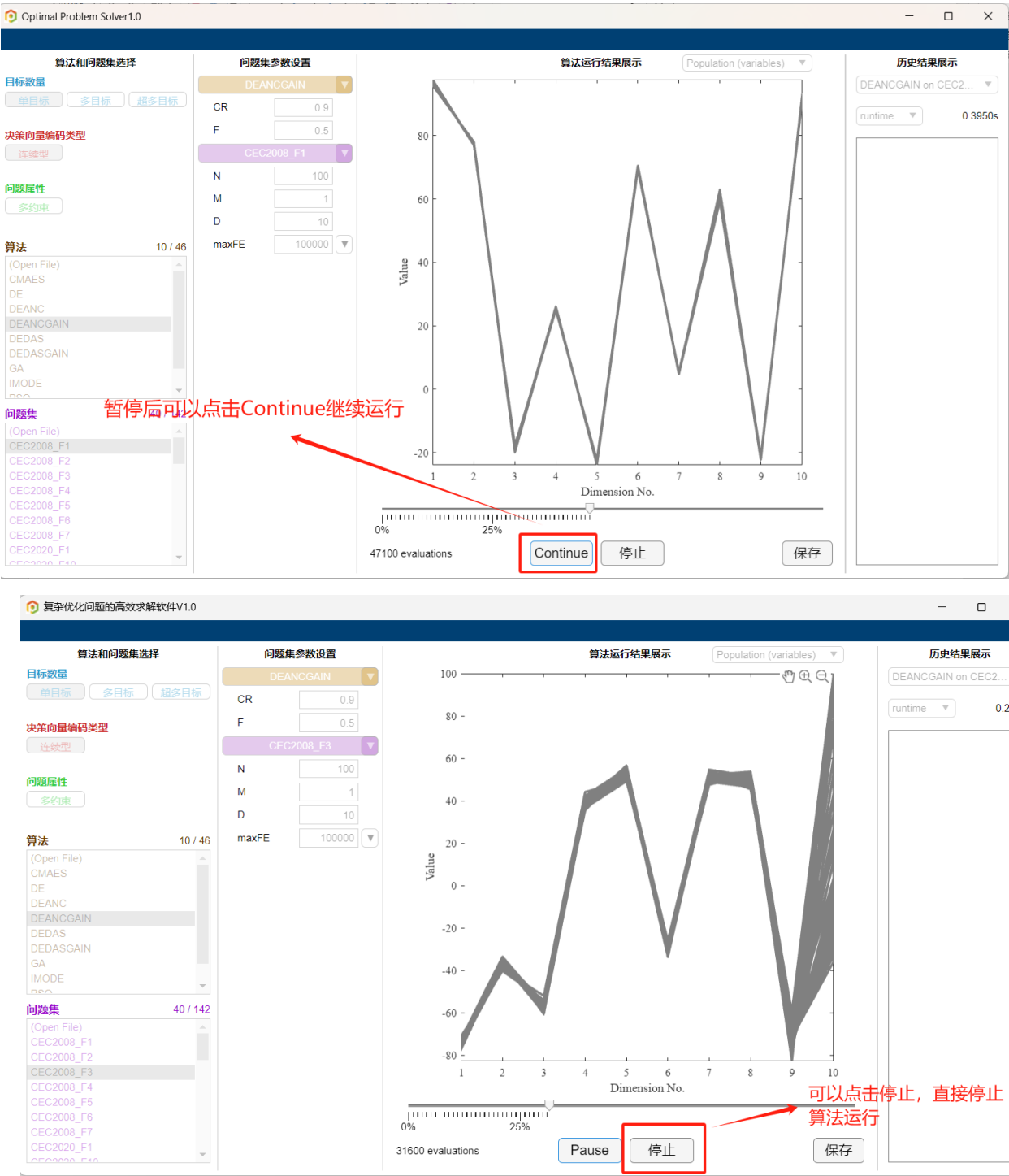
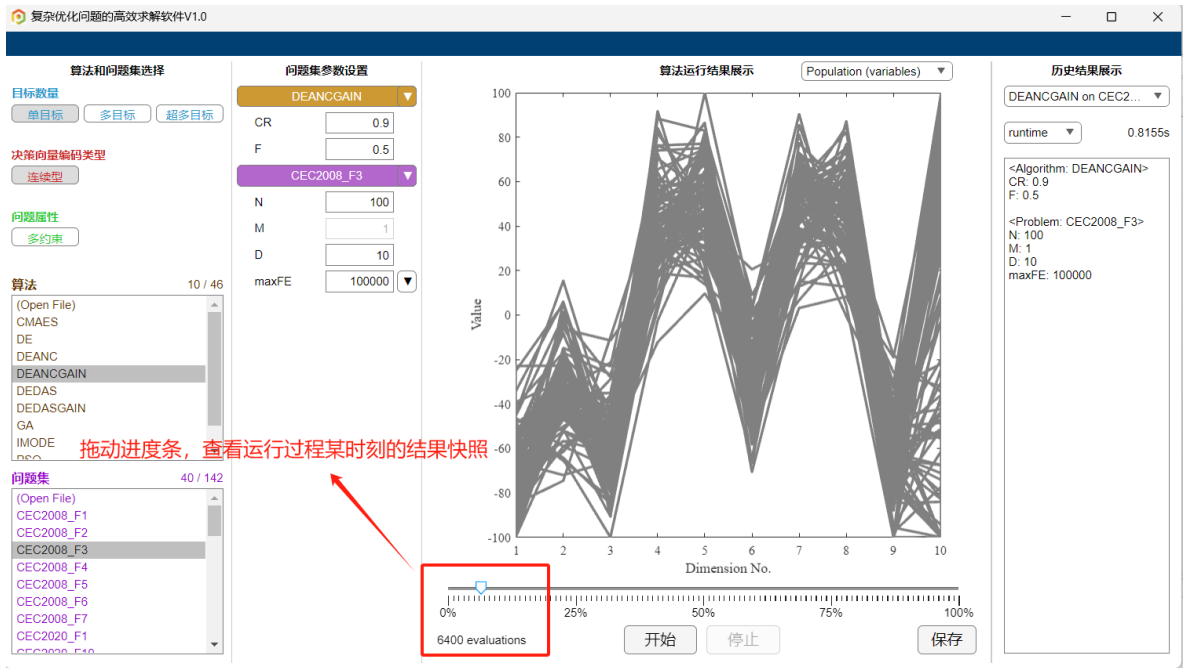
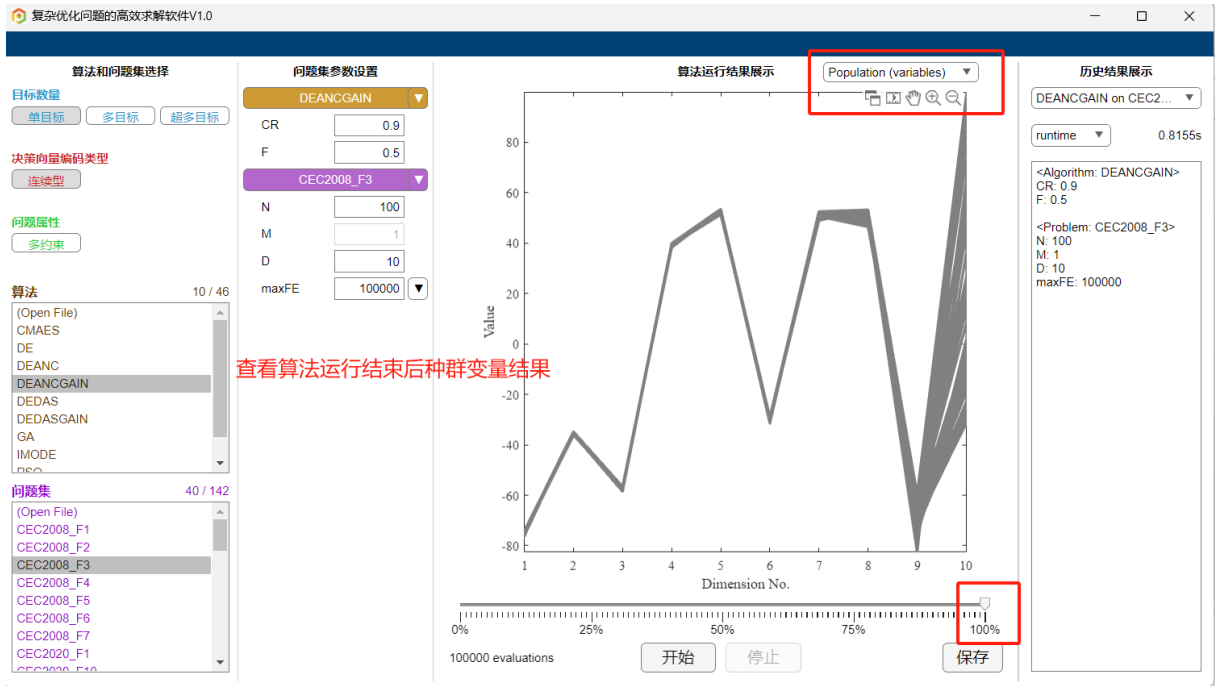


图 4-4 算法开始、暂停、停止、运行过程结果展示

步骤 5：运行结束结果展示、切换结果、拖动进度条查看过程快照

算法运行结束后，可以在第三栏右上角点击查看不同结果，如种群变量、最小值等等。可以拖动进度条，查看进化过程中某一时刻的结果快照。



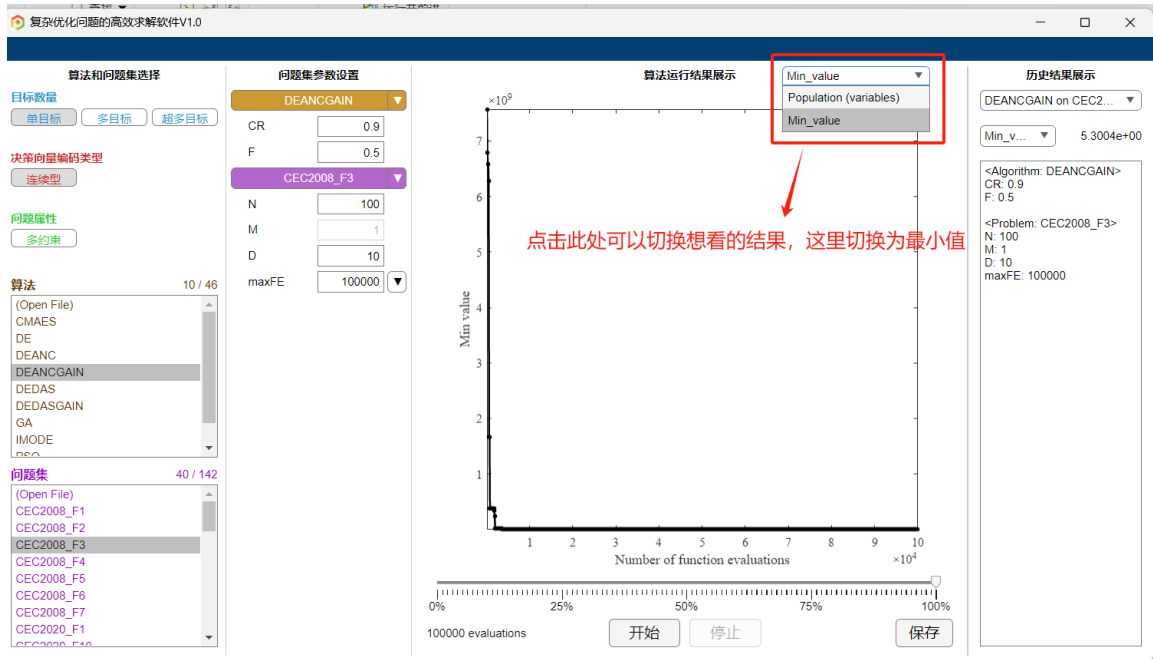
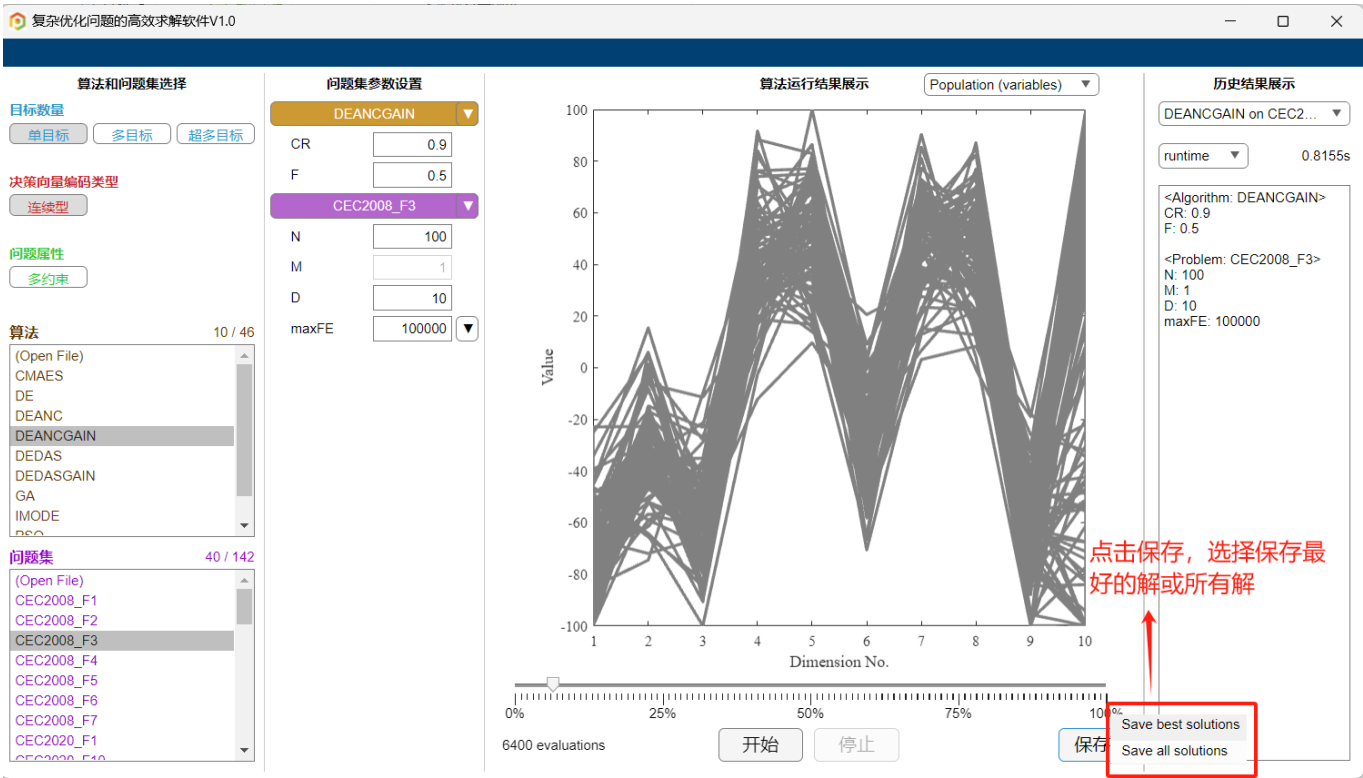


图 4-5 运行结束结果展示、切换结果、拖动进度条查看过程快照

## 步骤 6: 结果保存

可以点击“保存”按钮，保存结果。如下图所示



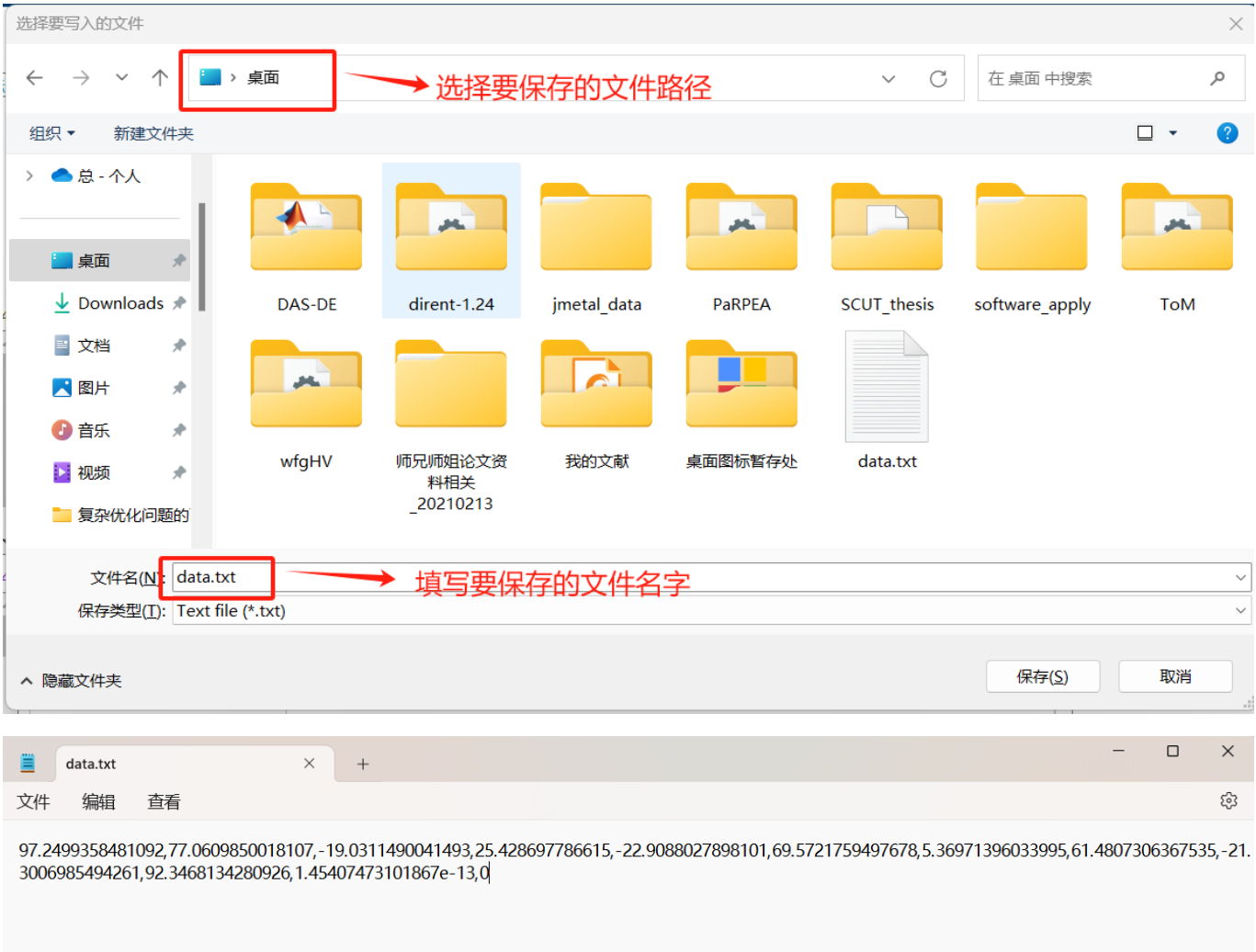


图 4-6 保存结果展示

步骤 7：查询历史运行结果

点击界面最右侧的历史结果展示，可以查询历史运行结果。如下图所示

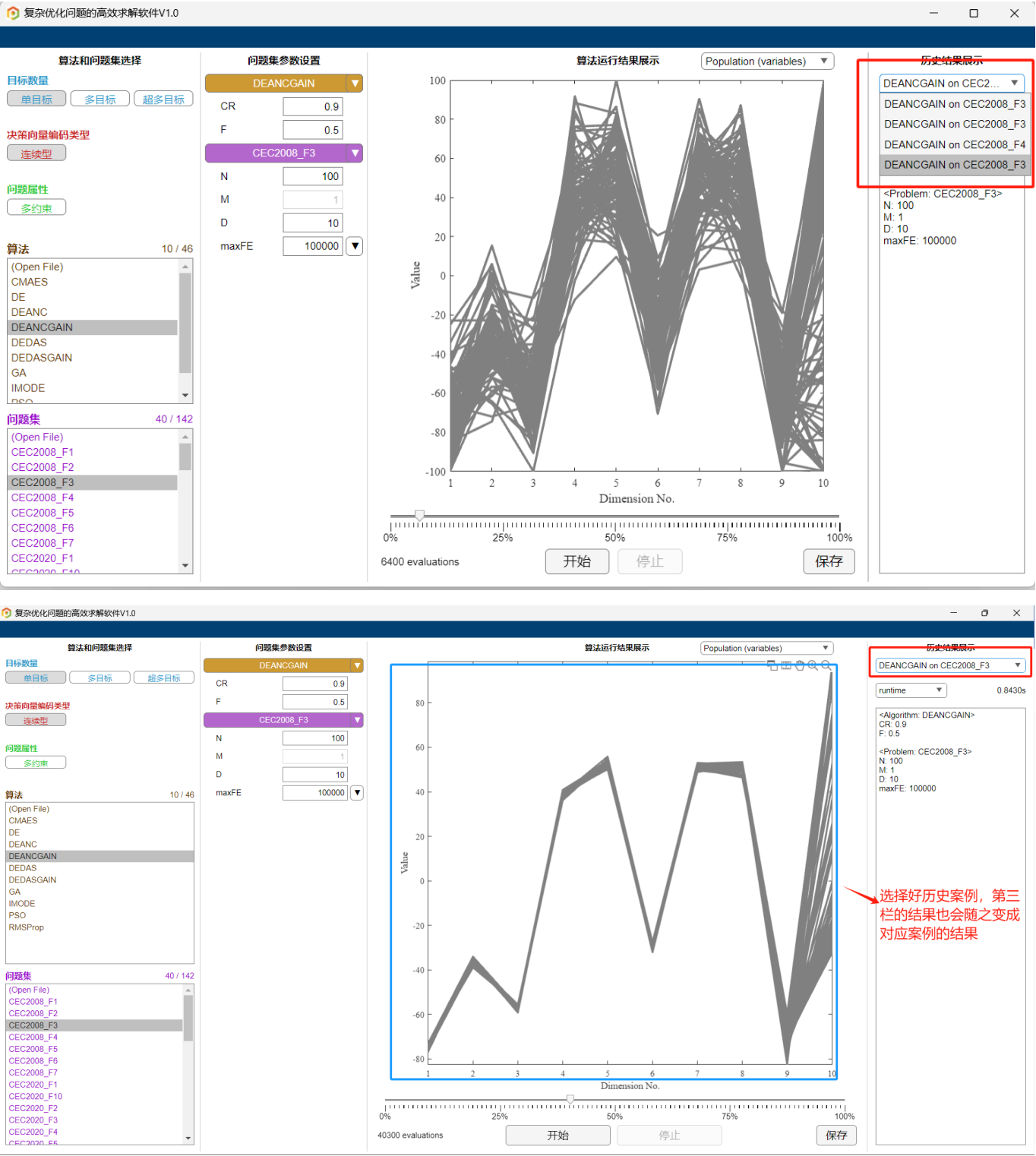


图 4-7 历史结果展示