**Gensbo说明文档**

produced by cmso4re group

2019.4

**目录**

[1 摘要 1](#_Toc10551216)

[2 简介 2](#_Toc10551217)

[3 优化问题 3](#_Toc10551218)

[3.1 基于仿真器的优化 3](#_Toc10551224)

[3.2 基于函数的优化 3](#_Toc10551225)

[4 单目标优化算法 5](#_Toc10551226)

[4.1 标准粒子群优化算法（PSO） 5](#_Toc10551229)

[4.1.1 基本公式 5](#_Toc10551235)

[4.1.2 算法流程 6](#_Toc10551239)

[4.1.3 程序参数设置 6](#_Toc10551240)

[4.2 改进小波简化粒子群优化算法（ISPSOWM） 7](#_Toc10551241)

[4.2.1 算法原理 7](#_Toc10551242)

[4.2.2 算法组成 7](#_Toc10551243)

[4.2.3 算法流程 9](#_Toc10551244)

[1.1.1 程序参数设置 9](#_Toc10551245)

[5 多目标优化算法 12](#_Toc10551246)

[5.1 全面学习小波粒子群优化算法（CLPSOWM） 12](#_Toc10551249)

[6 约束处理方法 13](#_Toc10551250)

[6.1 普通罚函数方法（动态剩余罚函数法） 13](#_Toc10551252)

[6.2 Oracle罚函数法 14](#_Toc10551253)

[7 变量处理方法 17](#_Toc10551254)

[7.1 连续变量 17](#_Toc10551256)

[7.2 离散变量 17](#_Toc10551257)

[7.3 连续离散变量 17](#_Toc10551262)

[7.4 非连续离散变量 17](#_Toc10551263)

[7.5 二元变量 17](#_Toc10551264)

[8 程序 19](#_Toc10551265)

# 摘要

GenSBO（General Simulation Based Optimizer）是一个面向仿真的优化器。它使用智能优化算法，通过调用仿真器获取当前方案的评价值，以此帮助算法获得可靠的优化方案结果。

# 简介

# 优化问题

## 基于仿真器的优化

通过仿真器获得当前方案的评价值。

## 基于函数的优化

通过（目标）函数获得当前方案（解）的评价值（函数值）。

寻优模型形式遵循CEC标准[1]，统一函数形式如下：

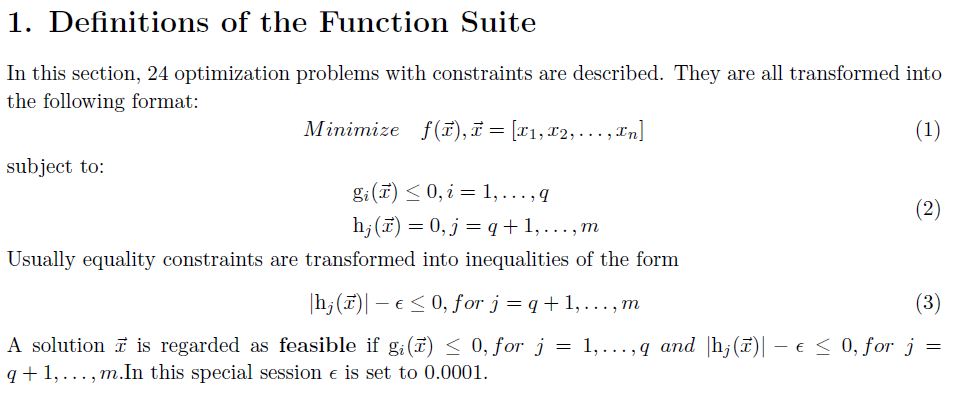
( 1 )

( 2 )

对等式约束存在允许误差：

( 3 )

本文取。



# 单目标优化算法

## 标准粒子群优化算法（PSO）

粒子群优化算法（particle swarm optimization, PSO）最早由Kennedy和Eberhart于1995年系统提出[2]，此后经过二十多年的实践与发展，在各类科学与工程领域取得了较大成功。PSO模拟鸟群的觅食过程进行全局寻优，是一种基于群体智能的元启发式算法。标准粒子群算法（也称经典粒子群算法）是经过初步完善的广受认可的PSO基本形式，也是之后众多改进算法的基石。

### 基本公式

标准粒子群算法的位置和速度更新公式为：

( 4 )

式中，

——当前寻优代数；

——粒子编号；

——粒子变量维度；

——惯性权重（inertia weight），文献推荐取值0.8~1.2，也可采用动态惯性权重以取得更好的寻优效果；

——加速度因子（acceleration coefficients）（分别为认知因子和社会因子），用于平衡算法个体认知部分与社会认知部分的权重，文献推荐取值为2，也可采用动态取值方法。

——随机系数，一般采用U(0,1)；

——编号为的粒子当前得到的最优适应值对应的位置值，称为个体历史最优（personal best）；

——编号为的粒子当前得到的全局最优适应值对应的位置值，称为全局最优（global best）。

### 算法流程

1. 设置算法参数；
2. 初始化粒子位置与速度；
3. 评价粒子的适应值；
4. 更新粒子个体历史最优和全局最优信息；
5. 更新粒子的位置和速度；
6. 重复（3）、（4）、（5），直到满足收敛条件。

### 程序参数设置

首先创建优化器实例，然后通过命令optimizer.set\_options设置参数。

代码示例：

# 创建优化器实例  
        optimizer = PSO(problem)  
  
        # 设置参数  
        optimizer.set\_options('para\_name', value)

标准粒子群算法执行参数变量名及取值示于下表。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 参数 | 变量名 | 取值类型 | 默认值 | 取值范围 | 含义 |
| 算法选择 | pso\_mode | 字符串 | ispsowm | standard\_pso, ispsowm | standard\_pso-标准粒子群算法；  ispsowm-改进小波简化粒子群算法 |
| 粒子群规模 | swarm\_size | 整型 | 30 | >0 | 寻优粒子数量 |
| 最大寻优代数 | step\_max | 整型 | 500 | >0 | 算法寻优代数设定上限 |
| 惯性权重取值方式 | w\_type | 字符串 | linear | linear, fixed | linear-惯性权重值随寻优代数的增大而线性减小；  fixed-取常数 |
| 惯性权重 | w | 浮点数 | 0.8 | >0 | w\_type取fixed时，惯性权重的值 |
| 惯性权重取值区间 | w\_range\_so | 浮点数 | [0.4,1.2] | >0 | [wini,wend]:  wini –惯性权重初值；  wend –惯性权重末值 |
| 认知因子 | c1 | 浮点数 | 2.0 | >0 | 加速因子中的认知因子 |
| 社会因子 | c2 | 浮点数 | 2.0 | >0 | 加速因子中的社会因子 |
| 收敛精度 | precision | 浮点数 | 0.001 | >0 | 寻优程序设定当种群粒子个体历史最优（集合）的标准差小于收敛精度时收敛。 |
| 粒子群邻域结构 | neighborhood | 字符串 | star | star, ring | star-星型（全互联型）；  ring-环形 |
| 是否开启并行计算 | if\_mp | 布尔值 | False | False, True | 是否针对评价函数的计算开启cpu并行计算（多线程），建议评价函数计算耗时较大时开启。 |
| 并行计算cpu核心数 | mp\_core\_num | 整型 | 2 | >1 | 并行计算采用的cpu核心数 |
| 是否初始化内点（可行解） | if\_ini\_cons | 布尔值 | False | False, True | 是否要求初始化的粒子必须含有可行解（不违反约束）；  通过创建一个sopso优化器，以约束函数为目标函数求解可行解。 |
| 内点个数 | ini\_step | 整型 | 1 | >0 | 当初始化内点过程中得到的内点数量低于该值时，将重复创建sopso优化器（更新运行参数）求解新的可行解，直至满足数量要求或者达到设定的重复求解次数 |
| 初始化内点的最大重复求解次数 | ini\_feasible\_x\_num | 整型 | 30 | >0 | 重复创建sopso优化器（更新运行参数）求解新的可行解的次数上限 |
| 初始化内点中sopso的运行参数设置 | 'ini\_swarm\_size':50,  'ini\_step\_max':1000,  'ini\_precision':1e-3 | 对应:  'swarm\_size'  'step\_max'  'precision' | | 更新策略：  ini\_swarm\_size += 10  ini\_step\_max \*= 1.1  ini\_precision /= 5 | |
| 是否收集寻优过程产生的可行解及其适应度函数值 | if\_get\_feasible\_x | 布尔值 | False | False, True | 若该值为True，则程序会在寻优过程中收集获得的可行解并输出 |
| 可行解个数上限 | feasible\_x\_num | 整型 | 100 | >0 | 当获得的可行解个数超过该值时，将根据目标函数值的分布密度优先删除密度高区域的可行解，直至可行解个数等于该上限值 |
| 当可行解个数满足要求时是否停止寻优 | if\_get\_feasible\_x\_only | 布尔值 | False | False, True | 若该值为True，则当可行解个数达到设定的上限时将停止寻优，并返回当前寻优结果与可行解集合 |

## 改进小波简化粒子群优化算法（ISPSOWM）

### 算法原理

结合简化粒子群算法（SPSO）和小波粒子群算法（PSOWM）的优势，改善种群搜索的多样性，提高算法全局寻优能力。

### 算法组成

1. 简化粒子群算法核心：

SPSO将粒子的速度信息整合进位置信息中，因此算法只保留粒子的位置更新公式。同时引入反向搜索机制，增强算法的全局寻优能力。

粒子更新公式如下[3]：

( 5 )

式中，

( 6 )

其中，

——随机系数，一般采用U(0,1)；

——反向搜索阈值，一般取较小值，否则会对算法的收敛性产生影响。

1. 小波粒子群算法核心：

PSOWM按一定概率对粒子位置进行小波变异，然后再进行位置的更新。

运用小波理论进行变异操作可提高算法的稳定性（小波母函数满足可容许条件）和收敛能力。

小波变异操作如下[4]：

( 7 )

式中，

——变异后的；

——小波函数值，此处选Morlet小波函数，其计算公式如下：

( 8 )

由于Morlet小波母函数99%的能量都包含在[-2.5,2.5]区间中，所以取值范围是区间[-2.5a, 2.5a]中的伪随机数。尺度参数a的计算公式如下：

( 9 )

式中，

——单调递增函数的形状参数；

g——a的上限值；

k——当前的迭代次数；

num——最大迭代次数。

随a的增大而减小，a随代数k的增大而增大，所以随代数k的增大而减小。当最大值等于1时，，算法的搜索空间很大；而当越接近1时，参数a的值很大，以致的最大值很小，算法的搜索空间比较小。即算法在早期有较好的全局搜索能力，在晚期有较好的局部搜索能力，算法的收敛性好。

IPSOWM重要参数如下表所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 粒子群数目 | 变异概率Pm | 形状参数 | g |
| 增加粒子群数目可提高搜索空间的多样性，但会消耗计算资源与时间。 | 决定进行变异操作的粒子个数，太大容易破坏已有信息。  粒子数越大，Pm值取越小。 | 越大，早期a的变化速率越大，相应的下降速率也越大。  （5-0.1，1~线性变化） | g越大，a的最大值越大，即min(| |)越小，搜索空间越小。  （1000或10000） |

### 算法流程

1. 设置算法参数；
2. 初始化粒子位置；
3. 评价粒子的适应值；
4. 更新粒子个体历史最优和全局最优信息；
5. 执行小波变异操作；
6. 按SPSO公式更新粒子的位置；
7. 重复（3）、（4）、（5）、（6），直到满足收敛条件。

### 程序参数设置

首先创建优化器实例，然后通过命令optimizer.set\_options设置参数。

代码示例：

# 创建优化器实例  
        optimizer = PSO(problem)  
  
        # 设置参数  
        optimizer.set\_options('para\_name', value)

简化小波粒子群算法执行参数变量名及取值示于下表。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 参数 | 变量名 | 取值类型 | 默认值 | 取值范围 | 含义 |
| 算法选择 | pso\_mode | 字符串 | ispsowm | standard\_pso, ispsowm | standard\_pso-标准粒子群算法；  ispsowm-改进小波简化粒子群算法 |
| 粒子群规模 | swarm\_size | 整型 | 30 | >0 | 寻优粒子数量 |
| 最大寻优代数 | step\_max | 整型 | 500 | >0 | 算法寻优代数设定上限 |
| 惯性权重取值方式 | w\_type | 字符串 | linear | linear, fixed | linear-惯性权重值随寻优代数的增大而线性减小；  fixed-取常数 |
| 惯性权重 | w | 浮点数 | 0.8 | >0 | w\_type取fixed时，惯性权重的值 |
| 惯性权重取值区间 | w\_range\_so | 浮点数 | [0.4,1.2] | >0 | [wini,wend]:  wini –惯性权重初值；  wend –惯性权重末值 |
| 认知因子 | c1 | 浮点数 | 2.0 | >0 | 加速因子中的认知因子 |
| 社会因子 | c2 | 浮点数 | 2.0 | >0 | 加速因子中的社会因子 |
| 粒子群邻域结构 | neighborhood | 字符串 | star | star, ring | star-星型（全互联型）；  ring-环形 |
| 进入可行域 | if\_ini\_cons | 布尔值 | False | False, True | 是否要求初始化的粒子必须含有可行解（不违反约束） |
| 是否开启并行计算 | if\_mp | 布尔值 | False | False, True | 是否针对评价函数的计算开启cpu并行计算（多线程），建议评价函数计算耗时较大时开启。 |
| 并行计算cpu核心数 | mp\_core\_num | 整型 | 2 | >1 | 并行计算采用的cpu核心数 |
| 是否初始化内点（可行解） | if\_ini\_cons | 布尔值 | False | False, True | 是否要求初始化的粒子必须含有可行解（不违反约束）；  通过创建一个sopso优化器，以约束函数为目标函数求解可行解。 |
| 内点个数 | ini\_step | 整型 | 1 | >0 | 当初始化内点过程中得到的内点数量低于该值时，将重复创建sopso优化器（更新运行参数）求解新的可行解，直至满足数量要求或者达到设定的重复求解次数 |
| 初始化内点的最大重复求解次数 | ini\_feasible\_x\_num | 整型 | 30 | >0 | 重复创建sopso优化器（更新运行参数）求解新的可行解的次数上限 |
| 初始化内点中sopso的运行参数设置 | 'ini\_swarm\_size':50,  'ini\_step\_max':1000,  'ini\_precision':1e-3 | 对应:  'swarm\_size'  'step\_max'  'precision' | 更新策略：  ini\_swarm\_size += 10  ini\_step\_max \*= 1.1  ini\_precision /= 5 |  |  |
| 是否收集寻优过程产生的可行解及其适应度函数值 | if\_get\_feasible\_x | 布尔值 | False | False, True | 若该值为True，则程序会在寻优过程中收集获得的可行解并输出 |
| 可行解个数上限 | feasible\_x\_num | 整型 | 100 | >0 | 当获得的可行解个数超过该值时，将根据目标函数值的分布密度优先删除密度高区域的可行解，直至可行解个数等于该上限值 |
| 当可行解个数满足要求时是否停止寻优 | if\_get\_feasible\_x\_only | 布尔值 | False | False, True | 若该值为True，则当可行解个数达到设定的上限时将停止寻优，并返回当前寻优结果与可行解集合 |
| 反向搜索概率阈值 | c | 浮点数 | 0.2 | [0,1] | 粒子进行反向搜索的概率 |
| 执行小波变异的概率阈值 | pm | 浮点数 | 0.7 | [0,1] | 执行小波变异的概率 |
| 形状参数 | xi\_wm | 浮点数 | 0.5 | >0  推荐取值范围(0,5] | 单调递增函数的形状参数，取较小值时侧重局部搜索，取较大值时侧重广域搜索。 |
| 小波函数中a的上限值 | g | 浮点数 | 1000 | >0 | 小波函数中a的上限值，常取1000或10000 |

# 多目标优化算法

## 全面学习小波粒子群优化算法（CLPSOWM）

在全面学习粒子群多目标优化算法的基础上对位置信息增加小波变异操作（参考4.2.1）。

# 约束处理方法

## 普通罚函数方法（动态剩余罚函数法）

普通罚函数方法在评价函数（fitness function）中加入一个惩罚项，对违反约束的解进行惩罚。加入罚函数后的评价函数表达式如下：

( 10 )

本文将目标函数形式统一为寻找最小值，因此违反约束的解会附加一项正的惩罚项值，使得该解的函数值增大。

惩罚项的计算公式如下：

( 11 )

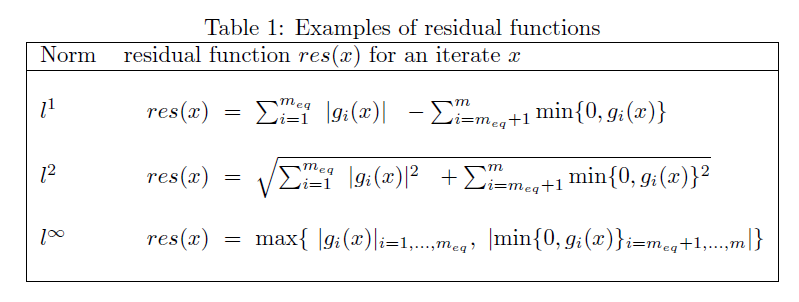
式中，

——惩罚系数，该值随寻优的进行而增大，本文取，为寻优代数，为惩罚倍数；

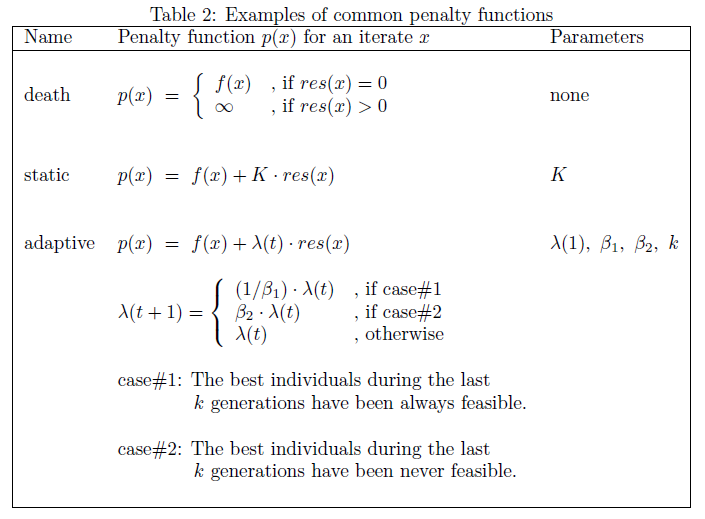
——剩余函数，该值代表解违反约束的程度，本文中该函数的计算公式为：

( 12 )

剩余函数的常见取值方式如下表所示[5]。



常见的罚函数方法如下表所示：



## Oracle罚函数法

Oracle罚函数法被认为是处理随机元启发式中的约束优化问题的一种极具潜力的新方法。它易于实现和处理，执行起来像静态惩罚函数一样健壮，并且在其他方法失败的情况下，仍然有很大的潜力找到全局最优解[5]。

Oracle惩罚方法通过引入参数将寻优问题转化成如下形式：

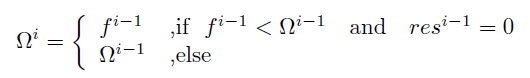
( 13 )

( 14 )

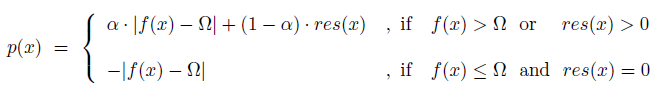
将原问题转换为求解参数。

需要注意的是，参数的初值必须大于全局最优解，否则Oracle罚函数法将失效。本文将的初值默认值设为109。

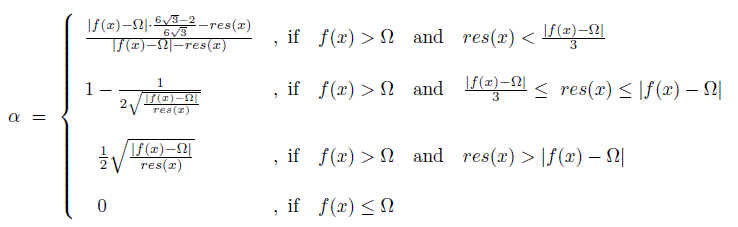
参数的更新策略如下：



使用Oracle罚函数法的评价函数形式如下：



式中，系数的计算公式如下：



## 程序参数设置

首先创建优化器实例，然后通过命令optimizer.set\_options设置参数。

代码示例：

# 创建优化器实例  
        optimizer = PSO(problem)  
  
        # 设置参数  
        optimizer.set\_options('para\_name', value)

约束处理参数变量名及取值示于下表。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 参数 | 变量名 | 取值类型 | 默认值 | 取值范围 | 含义 |
| 罚函数选择 | penalty\_type | 字符串 | common | common, oracle | common-普通罚函数方法；  oracle-Oracle罚函数法 |
| 惩罚倍数 | penalty\_times | 浮点数 | 100 | >0 | 选择“common”方法时生效，惩罚放大倍数 |
| 参数 | oracle | 浮点数 | 1e9 | >0 | 选择“oracle”方法时生效，参数的初值 |

# 变量处理方法

## 连续变量

取值范围内取值。

## 离散变量

四舍五入取整。

## 连续离散变量

取值集合中取整数值。

## 非连续离散变量

取值集合中取值。

## 二元变量

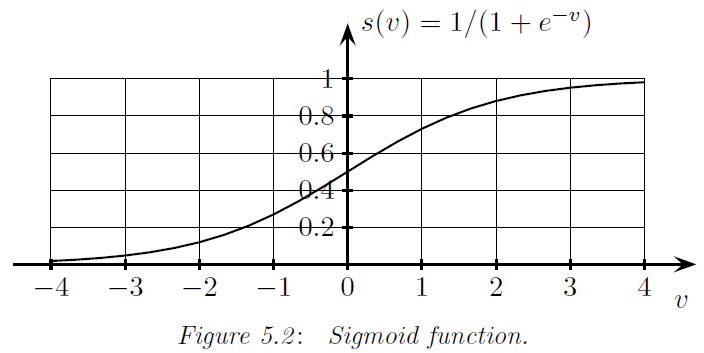
通过sigmoid函数取值0或1。

粒子位置状态更新公式为[6]：

( 15 )

( 16 )

式中，是预定阈值，常用U(0,1)，需要注意的是sig(0)=0.5。



# 程序

**参考文献**

[1] J. J. Liang T. P. R. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization[Z]. 2006.

[2] James Kennedy R. E. Particle Swarm Optimization[Z]. 1995.

[3] Clerc M. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization[C]. IEEE,1999.

[4] Tian Y G. D. L. X. Improved Particle Swarm Optimization with Wavelet-Based Mutation Operation[Z]. Berlin, Heidelberg:2012.

[5] Schlüter M., Gerdts M. The oracle penalty method[J]. Journal of Global Optimization,2010,47(2):293-325.

[6] Kennedy J., Eberhart R. C. A discrete binary version of the particle swarm algorithm[C]. IEEE,1997.