

powing 参考资料

概要: 幂尺度变换适应度计算

描述:

该函数对目标函数值 **ObjV** 进行幂尺度变换, 使其变成由 k 影响的幂尺度尺度。计算公式: $F' = (-ObjV)^k + 1$:
遵循“最小适应度为 0”的约定(特殊情况除外)。

语法:

```
FitnV = powing(ObjV)
FitnV = powing(ObjV, LegV)
FitnV = powing(ObjV, LegV, k)
FitnV = powing(ObjV, LegV, k, SUBPOP)
```

详细说明:

ObjV 为一个保存着个体对应的目标函数值的列向量。
LegV 是一个可选参数, 保存着个体对应的可行性的列向量, 0 表示该个体是非可行解, 1 表示是可行解。
 k (可选参数) 为 $(0, +\infty]$ 上的正实数, 若缺省或为 **None**, 则默认值为 1。
 $k < 1$ 时, 适应度大的个体经过幂指数变换后适应度相差较小, 此时为“保持种群多样性策略”;
 $k > 1$ 时, 适应度大的个体经过幂指数变换后适应度相差会较大, 此时为“精英策略”。
SUBPOP (可选参数) 表示子种群的数量, 要求能够被种群个体数整除。缺省时默认为 1。
FitnV 是记录着种群个体适应度值的列向量。
该函数遵循”目标函数值越大适应度越小”的约定。

特别注意:

本函数是根据传入参数 **ObjV** 来计算适应度的, 且遵循“种群目标函数值越大, 适应度越小”的原则, 因此在调用本函数前, 需要对传入的 **ObjV** 乘上'maxormin'(最大最小化标记)。但是, 由于返回的是 **FitnV**, 它与 **ObjV** 在含义上无关了, 因此不需要对其乘上'maxormin' 进行还原。

应用实例:

```
考虑有 10 个个体的种群, 其当前目标值 ObjV 如下情况。

ObjV = np.array([[ 1],[ 2],[ 3],[ 4],[ 5],[10],[ 9],[ 8],[ 7],[ 6]])
LegV = np.array([[ 1],[ 1],[ 1],[ 1],[ 1],[ 1],[ 1],[ 1],[ 1],[ 1]])
FitnV = powing(ObjV, LegV, 2, 2)  #幂尺度变换适应度计算
```

得到 FitnV:

FitnV =
$$\begin{pmatrix} 2 \\ 1.5625 \\ 1.5625 \\ 1.0625 \\ 1 \\ 1 \\ 1.0625 \\ 1.25 \\ 1.5625 \\ 2 \end{pmatrix}$$