差分进化算法介绍

Differential Evolution algorithm

卢光富

1.1引言

差分进化算法（DE）是一种新兴的进化计算技术，由Storn 和 Price 为解决切比雪夫多项式问题而提出来的。

DE算法保留了基于种群的全局搜索策略，采用实数编码，基于差分的简单变异操作以及一对一的竞争生存策略，降低了遗传操作的复杂性。DE算法具有较强的全局收敛能力和鲁棒的性能，适合求解一些复杂的优化问题。

目前DE算法已经在许多领域得到了应用，譬如调度，化工生产，电力系统，运筹学，现代农业与食品安全，环境保护等等。

1.2标准差分进化算法

(1)初始化

DE算法中的每个个体均对应问题的一个解 $x_i = [x_{i,1}, x_{i,2}, ...x_{i,d}]$

x_i 表示种群中的第 i 个个体。

DE算法初始化时，算法随机产生N个个体构成初始种群，如下所示：

$$x_{i,k}(0) = l_k + rand() \bullet (u_k - l_k),\ k = 1,2,...,d, \quad i = 1,2,...,N$$

1.2标准差分进化算法

(2)变异操作

通过变异操作，将初始化产生的个体 $x_i(g)$ 生成一个目标个体 $v_i(g)$，常见的变异操作机制如下：

$$DE/rand/1: v_i(g) = x_{r1}(g) + F \bullet [x_{r2}(g) - x_{r3}(g)]$$

$$DE/best/1: v_i(g) = x_{best}(g) + F \bullet [x_{r1}(g) - x_{r2}(g)]$$

$$DE/rand/2: v_i(g) = x_{r1}(g) + F \bullet [x_{r2}(g) - x_{r3}(g)] + F \bullet [x_{r4}(g) - x_{r5}(g)]$$

$$DE/best/2: v_i(g) = x_{best}(g) + F \bullet [x_{r1}(g) - x_{r2}(g)] + F \bullet [x_{r3}(g) - x_{r4}(g)]$$

$$DE/target-to-best/1: v_i(g) = x_i(g) + F \bullet [x_{best}(g) - x_i(g)] + F \bullet [x_{r1}(g) - x_{r2}(g)]$$

1.2标准差分进化算法

(3)交叉操作

标准DE算法的交叉操作在变异操作以后进行，通过将初始化个体 x_i 的部分变量用目标个体 v_i 的对应变量替换。从而生成测试个体 u_i 。

$$u_i(j) = \begin{cases} v_i(j), & if \quad r_j \leqslant CR \quad or \quad j = n_j, \\ x_i(j), & otherwise, \end{cases} \qquad j = 1, 2, \ldots n$$

1.2标准差分进化算法

(4)选择操作

标准DE算法采用贪婪选择方式，对于每个初始化个体 x_i 和测试个体 u_i ，选择好的个体进入下一代，即

$$X_i = \begin{cases} U_i, & if \quad f(U_i) \leqslant f(X_i) \\ X_i, & otherwise \end{cases}$$

1.3标准DE算法流程及其特点

(1)算法流程

BEGIN

步骤1：随机生成N个解作为初始种群。

步骤2：对于每个初始化个体 x_i

步骤2.1：选择变异操作产生临时解 v_i

步骤2.2：将临时解的个体与初始化的个体交叉产生新的测试个体 u_i 。

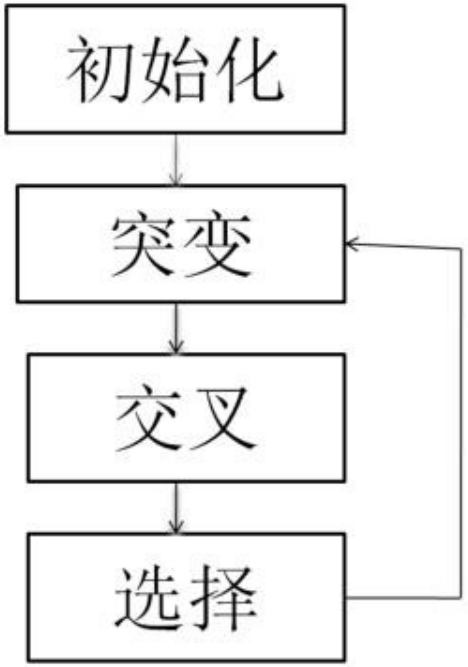
步骤2.3：将测试个体与初始化的个体进行比较，选择较好的个体。

步骤3：如果终止条件满足，输出最优解：否则，返回步骤2。

END

1.3标准DE算法流程及其特点

按照以上DE算法流程，我们可以得到：



1.3标准DE算法流程及其特点

(2)DE具有如下优点：

1.算法通用性较强，不依赖于问题信息。

2.算法原理简单易于实现。

3.易于与其它算法结合构造成有效的混合算法。

(3)DE具有如下缺点：

1.算法局部搜索能力较弱。

2.算法搜索性能对参数有一定的依赖性。

2.1 DE算法在调度中的应用

DE算法及其变种解过很多调度领域的问题，部分问题如下：

Tasgetiren M F, Sevkli M, Liang Y C, et al. A particle swarm optimization and differential evolution algorithms for **Job Shop Scheduling** Problem[J]. International Journal of Operations Research, 2006, 3(2): 120-135.

Fatih Tasgetiren M, Liang Y C, Sevkli M, et al. Particle swarm optimization and differential evolution for the **single machine** total weighted tardiness problem[J]. International Journal of Production Research, 2006, 44(22): 4737-4754.

Fatih Tasgetiren M, Pan Q K, Suganthan P N, et al. A discrete differential evolution algorithm for **the no-wait flowshop scheduling problem** with total flowtime criterion[C]//Computational Intelligence in Scheduling, 2007. SCIS'07. IEEE Symposium on. IEEE, 2007: 251-258.

2.1 DE算法在调度中的应用

Pan Q K, Tasgetiren M F, Liang Y C. A discrete differential evolution algorithm for the **permutation flowshop** scheduling problem[J]. Computers & Industrial Engineering, 2008, 55(4): 795-816.

Pan Q K, Wang L, Gao L, et al. An effective hybrid discrete differential evolution algorithm for the **flow shop scheduling with intermediate buffers**[J]. Information Sciences, 2011, 181(3): 668-685.

Tasgetiren M F, Pan Q K, Suganthan P N, et al. A differential evolution algorithm for the **no-idle flowshop** scheduling problem with total tardiness criterion[J]. International Journal of Production Research, 2011, 49(16): 5033-5050.

2.1 DE算法在调度中的应用

DE应用到调度领域，需要解决的是由连续转变为间断，下面我们以流水车间调度问题为例来进行讲解。

题目介绍

n 个工件组成集合$J = \{1, 2, \ldots, n\}$，m个机器组成集合$M = \{1, 2, \ldots, m\}$。每个工件，被安排在机器1,2,...,m上加工。在任何时间，每个机器最多只能加工一个工件，每个工件最多只能在一个机器上加工。每两个连续的机器K和$K+1$之间，存在一个容量为Bk的缓冲区。每个工件必须通过缓冲区 Bk，在由K到$K+1$机器时。如果 Bk 已满，工件必须滞留在当前的机器 K上，直到缓冲区有空位。鉴于所有作业的释放时间是零，所有机器的设置时间都包括在加工时间中，目标是找到一个加工工件的顺序，从而使最大完工时间最小。

2.1 DE算法在调度中的应用

在有缓冲区的流水车间中，在两个连续的机器之间有特定容量的缓冲区。在完成一个机器上的加工之后，工件或者直接在下一个机器上加工或者不得不暂存在缓冲区中。如果缓冲区已满并且下一台机器正在加工，这个工件必须在当前的机器上等待，从而堵塞该机器无法加工其它的工件，直到一个缓冲单元或者下一台机器可用。

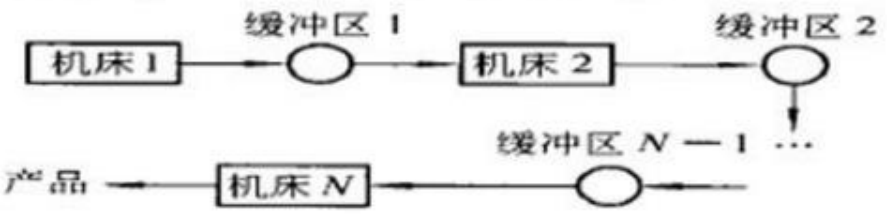


图 1　有 N 道工序的生产任务

2.1 DE算法在调度中的应用

$\pi=\{\pi(1),\pi(2),...,\pi(n)\}$ 代表工件要处理的顺序，$d_{\pi(i),k}$ 为离开机器K 的时间。

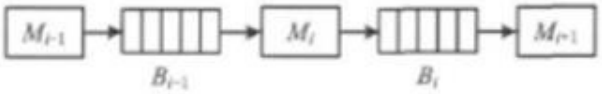


图1 缓冲区布局图

$$
\begin{aligned}
&d_{\pi(1),1}=p_{\pi(1),1}\\
&d_{\pi(1),k}=d_{\pi(1),k-1}+p_{\pi(1),k},\quad k=2,3,\ldots,m\\
&d_{\pi(i),1}=d_{\pi(i-1),1}+p_{\pi(i),1},\quad i=2,3,\ldots B_1+1\\
&d_{\pi(i),k}=\max(d_{\pi(i-1),k},d_{\pi(i),k-1},)+p_{\pi(i),k},\quad i=2,3,\ldots B_k+1,\ k=2,3,\ldots,m-1\\
&d_{\pi(i),1}=\max(d_{\pi(i-1),1}+p_{\pi(i),1},d_{\pi(i-B_1-1),2}),\quad i>B_1+1\\
&d_{\pi(i),k}=\max(\max(d_{\pi(i-1),k},d_{\pi(i),k-1},)+p_{\pi(i),k},d_{\pi(i-B_k-1),k+1}),\quad i>B_k+1,\ k=2,3,\ldots,m-1\\
&d_{\pi(i),m}=\max(d_{\pi(i-1),m},d_{\pi(i),m-1},)+p_{\pi(i),m},\quad i=2,3,\ldots n
\end{aligned}
$$

2.1 DE算法在调度中的应用

设定有四个工件和三台机器。
缓冲区B_1=1,B_2=1,加工时间给定如下：

$$[p_{j,k}]_{4\times3}=\begin{pmatrix}1&3&1\\1&2&2\\1&1&2\\1&1&1\end{pmatrix}$$

$$d_{\pi(1),1}=p_{\pi(1),1}=1$$
$$d_{\pi(1),2}=d_{\pi(1),1}+p_{\pi(1),2}=4$$
$$d_{\pi(1),3}=d_{\pi(1),2}+p_{\pi(1),3}=5$$
$$d_{\pi(2),1}=d_{\pi(1),1}+p_{\pi(2),1}=2$$
$$d_{\pi(2),2}=\max(d_{\pi(1),2},d_{\pi(2),1})+p_{\pi(2),2}=6$$
$$d_{\pi(2),3}=\max(d_{\pi(1),3},d_{\pi(2),2})+p_{\pi(2),3}=8$$
$$d_{\pi(3),1}=\max(d_{\pi(2),1}+p_{\pi(3),1},d_{\pi(1),2})=4$$
$$d_{\pi(3),2}=\max(\max(d_{\pi(2),2},d_{\pi(3),1})+p_{\pi(3),2},d_{\pi(1),3})$$
$$d_{\pi(3),3}=\max(d_{\pi(2),3},d_{\pi(3),2})+p_{\pi(3),3}=10$$
$$d_{\pi(4),1}=\max(d_{\pi(3),1}+p_{\pi(4),1},d_{\pi(2),2})=6$$
$$d_{\pi(4),2}=\max(\max(d_{\pi(3),2},d_{\pi(4),1})+p_{\pi(4),2},d_{\pi(2),3})$$
$$d_{\pi(4),3}=\max(d_{\pi(3),3},d_{\pi(4),2})+p_{\pi(4),3}=11$$

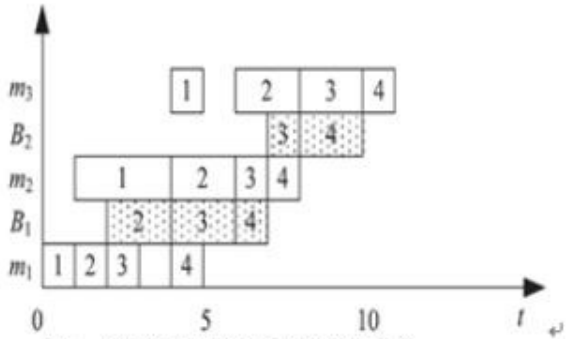


图 1. 计算最大完工时间的例子

2.1 DE算法在调度中的应用

(1)初始化

针对流水车间，DE算法初始化的目标是产生NP个工件排序。

```
void Heuristic::initial()//种群初始化
{
    int i,j,g;
    for(i=0;i<NP;i++)
    {
        for(j=0;j<job_number;j++)
        {
            g=rand();
            kin[j]=g;
        }
        lpt();//求解出对应的工件序列
    }

}
```

```
16667 23007 5703 1382
25013 19374 23877 5487
21458 6901 22996 20174
20433 3815 9587 22487
Press any key to continue
```

(2)变异操作

(1 0 2 3)
(0 2 1 3)
(1 0 3 1)
(3 0 2 1)

2.1 DE算法在调度中的应用

g=1,r=2

```
for(i=0;i<NP;i++)
{   o=rand()%100*0.01;
    g=rand()%job_number;
    r=rand()%job_number;//随机产生两个体
    for(j=0;j<job_number;j++)
    {
        if(o<F)//突变规模因子
            X[i][j]=m[g][j]-m[r][j];//进行突变操作中的差分操作
        else
            X[i][j]=0;
    }
}
for(i=0;i<NP;i++)
{
    makespan[i]=objective_function(m[i]);
}//计算初始群体每个个体的完工期
r=min(makespan,NP);//找到最小的完工期
for(i=0;i<job_number;i++)
{   best[i]=m[r][i];//选择最好的个体
}
for(i=0;i<NP;i++)
{
    for(j=0;j<job_number;j++)
    {
        V[i][j]=(best[j]+X[i][j]+job_number)%job_number;//在最好个体的基础上加上差分量并且取余
    }
    //cout<<endl;
}
```

(-1 2 -2 2)

假定第三个个体是最好的（3 0 2 1）

（2 2 0 3）

（2 0 3）

2.1 DE算法在调度中的应用

(3)交叉操作

```
for(i=0;i<NP;i++)
    for(j=0;j<job_number;j
    {
        o=rand()%100*0.01;
        if(o<CR)//设定交叉
        {
            for(k=j;k<job_
                V2[i][k]=V
            V2[i][k-1]=-10
        }
    }
```

假设（0　3　1）是最好的排序

下面继续插入操作

(2　0　3　1)

(0　2　3　1)

(0　3　2　1)

(0　3　1　2)

0　3）

（1　0　3）

（0　1　3）

（0　3　1）

(0　3　1　2)

后半部分是进行插入操作，最后一般还会加上一些局部搜索来加速收敛的过程。常见的操作有交换，插入等等。

2.1 DE算法在调度中的应用

将交叉后（0　3　1　2）与初始化（1　0　2　3）对比，选择较好的个体给下一代。

(4)选择操作

```
void    Heuristic::selection()//选择
{
    int i,j,k;
    double l,t,r,Z,B,C,M;
    k=0;
    for(j=0;j<machine;j++)
          k=job[j]+k;
    Z=10*job_number*machine;
    l= k/Z;
    t=l*h;
    for(i=0;i<NP;i++)
    {
        r=objective_function(m[i])-objective_function(Ur2[i]);
        B=r/t;
        C=exp(B);
        M=rand()%100*0.01;//交叉完的个体和初始个体比较
        if(M<min1(1,C))
            for(j=0;j<job_number;j++)
                m[i][j]=Ur2[i][j];
    }
}
```

2.1 DE算法在调度中的应用

除了以上大体步骤之外，有时候为了便于搜到较好的解，可以外挂上一些额外的启发式，如NEH,DC等，但是这样做无疑会增加程序运行的时间，所以需要做一些权衡。

谢谢！