文章编号：2096-1618(2024)01-0119-12

改进正余弦优化算法及在公交排班模型中应用

黄 艳， 吴泽忠
(成都信息工程大学应用数学学院，四川 成都 610225)

摘要：针对正余弦算法在搜索过程中存在收敛精度低、易陷入局部最优等缺点，提出一种基于正态变异自适应的改进正余弦算法(MSCA)。首先，将变异操作引入 SCA 算法，进行种群初始化；其次，引入惯性权重来修正位置更新方程，保留更多的信息和产生有前途的候选解；提出一种平衡勘探开发的非线性转换参数递减策略，在搜索过程中跳出局部最优解状态；最后，改进基于正态变异算子的位置更新策略，增加局部搜索空间。再选取 26 个国际标准测试函数对改进的算法进行测试，结果表明，MSCA 算法在收敛精度、收敛速度和收敛稳定性上，更优于其他改进算法。除此之外，还将改进后的 MSCA 应用在城市公交排班中，通过实验仿真得到改进的 MSCA 的最优目标函数值优于原有算法 SCA，以及公交排班符合客流的实际分布。

关 键 词：正余弦算法；正态变异因子；大规模优化问题；公交排班
中图分类号：TP301.6 **文献标志码**：A
doi：10.16836/j.cnki.jcuit.2024.01.018

0 引言

智能公交系统是网络交通系统(ITS)中研究的一个主要方向。它的任务是定位和跟踪、附加导航、命令分发、动态信息发布、搜索旅游路线等。它的建立将最大程度提高车和路资源的利用率，提高公交服务质量，从而创造巨大的社会经济效益，因此智能公交系统技术的研究具有深远的意义。

启发式算法是一种模拟自然界中群居动物间相互合作的工作方式而设计的优化算法。典型的启发式算法包括粒子群算法(PSO)[1]，鲸鱼（WOA)算法[2]，蝴蝶（BOA）算法[3]，飞蛾扑火（MFO）算法[4]，大红斑蝶（MBO）算法[5]，蚯蚓（EOA）算法[6]，大象放牧(EHO)算法[7]等。启发式算法广泛应用于生物信号处理和过程控制等实际问题中。

本文主要研究澳大利亚学者 Seyedali Mirjalili 在 2016 年所提出的正余弦(SCA)算法[8]，该算法模拟正余弦函数行为而设计。目前 SCA 广泛应用于结构设计、目标控制、多功能设计等实际问题。通过对 SCA 算法的实验测试，发现 SCA 在搜索最优解的过程中存在停滞、早熟收敛、开发不足的问题。此外，在利用 SCA 解决多模态问题时，容易出现局部最优的问题。因此，避免早熟收敛到局部最优和提高精度已经成为 SCA 的两个重要研究方向。对此，研究人员开发了许多 SCA 变体来改进算法。

在之前的研究中，徐松金等[9]提出一种改进的正余弦算法(ISCA)，引入惯性权重提高算法收敛精度和加快收敛速度，并采用反向学习策略进行种群初始化，提高了初始种群的多样性和样本质量。ISCA 在 8 个高维度测试函数上测试，该算法在 7 个测试函数中收敛到理论最优值 0，说明它在处理高维函数时有较好的寻优能力。张校非等[10]提出一种改进的正余弦算法(ZSCA)引入自适应变异算子提高种群多样性，以及加权惯性权重平衡算法的局部和全局搜索能力将 r_1 由线性递减函数变为指数递减函数，加强迭代后期局部搜索能力。ZSCA 在 10 个经典的单峰、多峰函数上测试，该算法在 3 个测试函数中收敛到理论最优值 0，其余 7 个测试函数均优于 SCA 算法，说明有较好的改进效果。

在处理实际问题中越来越多地遇到高维度优化问题，目前 SCA 及其改进的算法还未被广泛应用到高维度的优化问题中，针对 SCA 的基本缺点以及为更好解决高维度的优化问题，本文提出一种改进的正余弦算法(MSCA)。在该算法中，利用基于遗传算法的变异策略来更新种群的初始位置，提高种群多样性和种群质量。加入惯性权重修正位置更新方程，可以保留更多的信息并产生有前途的候选解。此外提出了一种平衡勘探开发的非线性转换参数递减策略和基于正态变异的位置更新策略。所提出的 MSCA 算法扩广到求解从 30 维到 5000 维的高维全局优化问题，群体规模为 30，迭代次数为 1000。实验结果表明，该算法不仅收敛速度快，而且具有较高的精度和稳定性。

收稿日期：2021-09-29
基金项目：国家自然科学基金资助项目(71962030)

1 公交排班系统分析

1.1 城市公交调度时间规划分析

运营车辆排班智能设计是公交车辆智能排班设计[21]需要解决的典型问题之一,是城市日常交通管理、车辆管理以及公交车和司机运营的重要依据。合理的调度可以提高公共交通的效率,降低运营成本,减少乘客的等待时间,提高服务质量。

1.2 公交线路设计模型

1.2.1 车辆行驶模型

保持车速恒定,车辆行驶距离:

$$s=vt \tag{1}$$

式中,s 为行驶路程,v 为行驶速度,t 为行驶时间。

假设模型中第一辆车的位置为零点,d 为两辆相邻车辆之间的距离,且车辆总数为 i,则第 j 辆车的位移为

$$l_n=vt_j+(i-j)\cdot d \tag{2}$$

1.2.2 乘客上下车模型

假设乘客到达率不变,乘客的出行时间不变,车辆在车站的停车时间为

$$\text{Stop\_time}=\text{get\_time}\cdot\text{Num\_fare} \tag{3}$$

式中,Num_fare 为到达车站点的乘客数量,get_time 为每位乘客上车的耗费时间。则到达站点时刻乘客的数量为:

$$\text{Num\_fare}=\text{Num}_t\text{\_fare}\cdot t_{\text{w\_bus}} \tag{4}$$

式中,Num_t_fare 为单位时间内聚集等待的乘客数量,$t_{\text{w\_bus}}$ 为该站点一辆公交车辆的候车时间。结合式(3)、(4):

$$\text{Stop\_time}=\text{Num}_t\text{\_fare}\cdot t_{\text{w\_bus}}\cdot\text{get\_time} \tag{5}$$

车从起始位置到达第一个站,完成将所有乘客运载上车时所需要耗费的总时间为

$$T_0=t+\text{Stop\_time} \tag{6}$$

经过了 n'个公交站时车辆所耗费的总的时间为

$$T_w=n'\times(t+\text{Stop\_time})=n'\times t+n'\times\text{Num\_fare}\times t_{\text{w\_Bus}}\times\text{get\_time} \tag{7}$$

当第 n 辆和第 $n-1$ 辆车相遇时,位移相同。即

$$l_n=l_{n-1} \tag{8}$$

$$l_n=v\cdot t_n+(m-n)d=v\times t_{n-1}+(m-n+1)d \tag{9}$$

$$v\times t_n=v\times t_{n-1}+d \tag{10}$$

则可得到:

$$\Delta t=\frac{d}{v} \tag{11}$$

可以推广到,第 n 部车和第 b 部车,相遇的表达式为

$$\Delta t_{n-b}=(n-b)\frac{d}{v} \tag{12}$$

根据 T_w 可得:

$$\Delta t'_{n-b}=\text{Num\_fare}\times\text{get\_time}\times(t_{wn}-t_{wb}) \tag{13}$$

又 $\Delta t'_{n-b}=\Delta t_{n-b}$,则:

$$(n-b)\frac{d}{v}=\text{Num}_t\text{\_fare}\times\text{get\_time}\times(t_{wn}-t_{wb}) \tag{14}$$

1.3 公交排班问题模型设计

1.3.1 模型假设

(i)各公交车的车型相同;

(ii)公交车按调度时间启动,速度稳定,无交通堵塞和事故发生;

(iii)各乘客在各时段内到站时间服从均匀分布。

1.3.2 定义变量

使用的变量如表 1 所示。

表 1 线路参数设置

参数	取值	参数	取值
时间 t	L/v	相邻车辆间发车间隔的最大值($T_{\min}$)	10
车速 v	20	相邻车辆间发车间隔的最小值($T_{\min}$)	1
早班发车时刻	6:00	相邻车辆发车间隔之差的限制值 ε	4
末班发车时刻	21:00	各车的平均期望满载率 $\theta(0\leqslant\theta\leqslant1)$	0.8
统一票价 P/(元/人)	1	乘客等车时间成本权系数 α	200
整个线路的站台数 n	24	公司的运营收益权系数 β	1
总的发车次数/m	100	乘务员人数 f	2
调研线路总的长度 L/km	26.5	乘务员工资 p	20
车辆运营的单位损耗成本 C/(元/(车·公里))	3.5	满载时车的容量 Q/(人/车)	100

1.3.3 建立目标函数

(i) 约束条件

(1)对平均满载率进行约束:

$$\frac{\sum_{i=1}^{m}\sum_{j=1}^{n}r_j tP}{m\times Q}>\theta \tag{15}$$

式中,i 表示第 i 次车,$i=1,2,\cdots,m$; j 表示第 j 个车站,$j=1,2,\cdots,n$;车容量 Q 表示满载时车辆的容量;θ 表示各车的平均期望满载率($0\leqslant\theta\leqslant1$)。$r_j$ 表示在调

度周期内第 j 站停留时间。

(2) 对相邻两列车的最长与最短发车间隔 $x_i=\text{time}_i-\text{time}_{i-1}$ 的约束:

$$T_{\min}<x_i<T_{\max},i=2,3,\cdots,m \tag{16}$$

式中,$T_{\max}$ 表示相邻车辆之间的最大距离;$T_{\min}$ 表示相邻车辆之间的最小距离。

(3)对相邻列车之间的时差进行限制,以确保发车时间的连续性,公式如下:

$$|x_{i+1}-x_i|<\varepsilon,i=2,3,\cdots,m \tag{17}$$

式中,ε 表示相邻车辆之间的间隔时间的限制值。

(4)每辆车的运营时间不大于规定的运营时间,即

$$t_{sj}\geqslant t_{tj} \tag{18}$$

(ii)计算公交公司的运营时间成本

设某一城市某时期单位时间的候车换算费用为 λ,则一天内的等车总时间成本为

$$c'=\lambda\times\sum_{i=1}^{m}\sum_{j=1}^{n}r_j\times\frac{(t_i-t_{i-1})^2}{2} \tag{19}$$

(iii)计算公交公司的运营收益

公交公司的运营收益为整体收益减去运营成本。而运营成本分为固定成本(建设、维护成本等)和可变成本。可变成本包括司机和乘客的工资、机动车辆的燃油消耗、车辆折旧和其他费用。从而整体收益 R 为

$$R=\sum_{t=t_{i-1}}^{t}\sum_{i=1}^{m}\sum_{j=1}^{n}r_jtP \tag{20}$$

运营成本为

$$C\times L\times m+t\times f\times p \tag{21}$$

公交公司的运营收益 R' 为

$$R'=\sum_{t=t_{i-1}}^{t}\sum_{i=1}^{m}\sum_{j=1}^{n}r_jtP-(C\times L\times m+t\times f\times p) \tag{22}$$

公司经营收益权系数为 β;乘客等待时间的成本权重系数 α。根据二次乘法函数,确定公交规划优化模型的目标函数:

$$\min z=\alpha\times\left(\lambda\times\sum_{i=2}^{m}\sum_{j=1}^{n}r_j\times\frac{(t_i-t_{i-1})^2}{2}\right)-$$
$$\beta\times\left(\sum_{t=t_{i-1}}^{t}\sum_{i=1}^{m}\sum_{j=1}^{n}r_jtP-C\times L\times m+t\times f\times p\right) \tag{23}$$

2　标准的正余弦算法(SCA)

标准正余弦算法(SCA)是澳大利亚学 Seyedali Mirjalili 在 2016 年提出的优化算法。该算法先选取一组随机解,然后模拟正弦波和余弦波的数学函数理论经过不断的发展和探索,逐步接近整体理想解。解的位置公式为

$$X_i^{(t+1)}=\begin{cases}X_i^{(t)}+r_1\cdot\sin(r_2)\cdot|r_3P_i^{(t)}-X_i^{(t)}|,r_4<0.5\\X_i^{(t)}+r_1\cdot\cos(r_2)\cdot|r_3P_i^{(t)}-X_i^{(t)}|,r_4\geqslant0.5\end{cases} \tag{24}$$

式中:$X_i^{(t)}$ 为第 i 个粒子在第 t 次迭代时的位置向量;t 为当前迭代数;$P_i^{(t)}$ 为第 t 次迭代的第 i 个粒子最优解。r_1 为线性递减的转换参数,其表达式为

$$r_1=a-a\cdot\frac{t}{T_{\max}} \tag{25}$$

其中,t 表示当前迭代次数,$T_{\max}$ 表示迭代的总次数,$a>0$ 是常量;这里 r_2 是一个在[0,2π]的随机变量,用于获取下一个解决方案的运动方向。r_3 是一个在[-2,2]的随机变量,提供对于 $P_i^{(t)}$ 的随机权重,去强调($r_3>1$)或者淡化($r_3<1$) 第 i 个粒子的目标解 $P_i^{(t)}$ 对于当前解的位置向量的影响。r_4 在式(24)中的正弦($r_4<0.5$)和余弦函数($r_4\geqslant0.5$)之间切换。

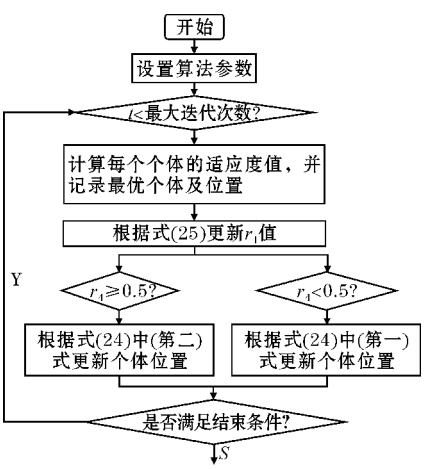

图 1　SCA 算法流程图

3　改进的正余弦优化算法

传统的正余弦算法存在收敛精度低、易局部最优、收敛速度慢等缺点。将通过 4 个修改来改进 SCA。以下将详细介绍改进后的正余弦算法(MSCA)。

3.1　自适应变异

SCA 有结构简单、较强的通用性等优点,但同时存在收敛精度低、易陷入局部最优等缺点。MSCA 算法利用了遗传算法的群体变异思想,对 SCA 算法进行变异操作。在原始群体中,随机选择部分个体,以一定的概率改变个体值。变异操作扩大了在迭代过程中减小的搜索空间,使个体跳出局部最优状态,保持了物种的多样性。

3.2 修正位置方程

在传统的 SCA 中,在前期其他搜索代理都被全局代理所吸引,它们可能在没有完全搜索空间时就提前收敛。这意味着 SCA 可以很容易地进入成熟状态。根据式(24)得知 SCA 的位置更新方式是 $X_i^{(t)}$ 向全局最优解靠近从而产生新的备选解。然而,这种方式较容易陷入本地优化,从而降低 SCA 的性能。

为提高 SCA 的性能(不同于以往的工作),引入惯性权重系数 $w(t)$,并对式(24)所描述的位置进行修正,修正方程如下:

$$X_i^{(t+1)}=\begin{cases} w(t)\cdot X_i^{(t)}+r_1\cdot\sin(r_2)\cdot|r_3p_i^{(t)}-X_i^{(t)}| \\ w(t)\cdot X_i^{(t)}+r_1\cdot\cos(r_2)\cdot|r_3p_i^{(t)}-X_i^{(t)}| \end{cases} \tag{26}$$

式中,$w(t)$为惯性权重系数,t 表示当前迭代,$T_{\max}$表示迭代的总次数。由式(26)可知,w 是十分重要的参数,在前期 w 值较大此时进行“全局搜索”,后期 w 值较小此时有利于“局部搜索”。因此,w 从初始值 w_1 到结束值 w_2 是线性递减的,惯性权重系数方程如下:

$$w(t)=w_1-(w_1-w_2)\cdot\left(\frac{t}{T_{\max}}\right)^{\frac{1}{t}}+w_2 \tag{27}$$

与原始算法 SCA 中位置更新式(24)相比,式(26)能够为位置的更新提供更多的信息,生成更加优化的候选解,以此有助于提高 SCA 算法的搜索能力。

3.3 改进的转换参数策略

根据原始正余弦(SCA)算法[8],SCA 中主要存在 4 个转换参数:r_1、r_2、r_3 和 r_4,其中 r_1 是勘探转换为开采的关键参数。r_1 为从全局勘探到局部开发提供一个切换。r_1通常是从常数 a 线性递减到零。在搜寻最优解的初始阶段,线性递减策略的转换参数 r_1 具备较好的搜索能力但收敛性差; 搜索后期,收敛精度高,但易陷入局部理想解。此外,由于 SCA 的搜索过程是复杂且非线性的,r_1 的线性递减策略并不能真实有效的反映搜寻过程。因此,对 r_1 进行了修正,采取了一种非线性递减策略,提出一种新的转换参数方程:

$$r_1=(a_1-a_2)\cdot\exp\left(-\frac{t^2}{(\lambda\cdot T_{\max})^2}\right) \tag{28}$$

式中,t 表示当前迭代次数,$T_{\max}$表示迭代的总次数。λ 为非线性指数,a_1 和 a_2 分别为常数 a 的初值和终值。

3.4 正态变异算子最优位置更新

原算法容易陷入理想局部状态。本文提出了正态变异对种群个体进行变异操作,正态分布具有较好的搜寻和干扰能力,正态变异算子将会适当地干扰寻优个体,使其重新游向新的位置,并能得到最好的全体解决计划,避免算法陷入局部最优解。对第 i 个个体在第 j 维的正态变异位置更新公式为

$$\bar{X}_{ij}^{(t)}=X_{ij}^{(t)}+N(0,1)\cdot X_{ij}^{(t)} \tag{29}$$

式中,$N(0,1)$为标准正态分布,其方差为 1,均值为 0; $\bar{X}_{ij}^{(t)}$ 为第 i 个个体当前搜索空间第 j 维的最优位置。

综上所得,将基于遗传算法中变异策略的种群初始化以及对 r_1 的非线性递减策略、$w(t)$的非惯性权重位置变换改进、最后进行正态变异算子更新位置这 4 种改进方法结合起来,开发了 MSCA 算法。

表 2 MSCA 的算法伪代码

算法 2 MSCA 算法

```
Begin
  设置算法参数,并引入正态变异算子对种群进行初始化;
while(t<T_max) do
  for i=1 to N do
    根据式(27)、(28)分别得到 w(t),r1 的值;
    if (r4<0.5) do
      根据式(26)中式(a)更新个体位置;
    else if (r4≥0.5) do
      根据式(26)中式(b)更新个体位置;
    end if
  end for
  对当前种群个体根据式(29)执行正态变异操作;
  更新种群个体的最佳位置与适应度值;
  t=t+1;
end while
end
```

4 数值实验和结果分析

4.1 测试标准函数及测试函数设置

从文献[2]中选取 19 个国际标准测试函数,包括 12 个单峰函数和 7 个多峰函数。从文献[11]中选取 6 个高维(高达 5000)的测试函数。然后,将改进后的 MSCA 算法与初始算法 SCA[8]、WOA[2]、ALO[12]、GWO[13]及改进算法 ZSCA[10]、ISCA[9]在多种维度下进行比较,依此判断正余弦算法是否优于其他启发式算法。具体测试函数表达式及搜索区间等见表 3。

4.2 测试函数实验结果与分析

4.2.1 对低维测试结果

在实验之前,先对算法进行参数设置,其中 SCA、WOA、ALO、GWO 的参数分别来自文献[8]、[2]、[12]、[13]。经过一系列模拟实验,取定改进算法 MSCA 中 $w_1=0.08$; $w_2=0.01$; $a_1=0.1$; $a_2=0$; $\lambda=15$; a 的参数来自文献[2]。参数设置见表 4 所示。

表3　26 个多维标准测试函数

函数	搜索区间	$f_{\min}$
Sphere: $F_1(x)=\sum_{i=1}^{n}x_i^2$	$[-100,100]$	0
Schewefel 2.22: $F_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	$[-10,10]$	0
Schewefel 1.2: $F_3(x)=\sum_{i=1}^{n}(\sum_{j-1}^{i}x_j)^2$	$[-100,100]$	0
Schewefel 2.21: $F_4(x)=\max_i\{\lvert x_i\rvert,1\leqslant i\leqslant n\}$	$[-100,100]$	0
Rosenbrock: $F_5(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	$[-30,30]$	0
Quartic: $F_6(x)=\sum_{i=1}^{n}ix^4$	$[-1.28,1.28]$	0
Noise: $F_7(x)=\sum_{i=1}^{n}ix_i^4+\text{random}[0,1)$	$[-1.28,1.28]$	0
Cigar: $F_8(x)=x_1^2+10^6\sum_{i=2}^{n}x_i^6$	$[-100,100]$	0
Tablet: $F_9(x)=10^6\cdot x_1^2+\sum_{i=2}^{n}x_i^6$	$[-1,1]$	0
Pixon & Price: $F_{10}(x)=(x_1-1)^2+\sum_{i=2}^{n}i(2x_i^2-x_{i-1})^2$	$[-10,10]$	0
Elliptic: $F_{11}(x)=\sum_{i=2}^{n}(10^6)^{(i-1)/(n-1)}\cdot x_i^2$	$[-100,100]$	0
Sum squares: $F_{12}(x)=\sum_{i=2}^{n}ix_i^2$	$[-10,10]$	0
Zakharov: $F_{13}(x)=\sum_{i=1}^{n}x_i^2+(\sum_{i=1}^{n}0.5ix_i)^2+(\sum_{i=1}^{n}0.5ix_i)^4$	$[-5,10]$	0
Sum-power: $F_{14}(x)=\sum_{i=1}^{n}\lvert x_i\rvert^{(i+1)}$	$[-1,1]$	0
Rastrigin: $F_{15}(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	$[-5.12,5.12]$	0
Inverted Cosine Mixture: $F_{16}(x)=0.1n-(0.1\sum_{i=1}^{n}\cos(5\pi x_i)-\sum_{i=1}^{n}x_i^2)$	$[-1,1]$	0
Griewank: $F_{17}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	$[-600,600]$	0
Pathological: $F_{18}(x)=\sum_{i=2}^{n}0.5+\frac{\sin^2(\sqrt{100x_{i-1}^2+x_i^2})-0.5}{1+0.001(x_{i-1}^2-2x_{i-1}x_i+x_i^2)^2}$	$[-100,100]$	0
Ackley: $F_{19}(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+20+e$	$[-32,32]$	0
Weierstrass's: $F_{20}(x)=\sum_{i=1}^{n}[\sum_{k=0}^{k_{\max}}a^k\cos(2\pi b^k\times(x_i+0.5))]-n\sum_{k=0}^{k_{\max}}a^k\cos(\pi b^k)$ $k_{\max}=10,a=0.5,b=3$	$[-50,50]$	0
Alpine: $F_{21}(x)=\sum_{i=1}^{n}\lvert x_i\sin(x_i)+0.1x_i\rvert$	$[-10,10]$	0
Generalized schaffer: $F_{22}(x)=0.5+((\sin(\sum_{i=1}^{n}x_i^2))2-0.5)\times(1+0.001(\sum_{i=1}^{n}x_i^2))^{-2}$	$[-100,100]$	0
Schaffer: $F_{23}(x)=0.5+\frac{\sin^2(\sqrt{\sum_{i=1}^{n}x_i^2})-0.5}{(1+0.001(\sqrt{\sum_{i=1}^{n}x_i^2}))^2}$	$[-100,100]$	0
Bohachevsky: $F_{24}(x)=\sum_{i=1}^{n-1}[x_i^2+2x_{i+1}^2-0.3\cos(3\pi x_i)-0.4\cos(4\pi x_{i+1})+0.7]$	$[-15,15]$	0
Salomon: $F_{25}(x)=1-\cos(2\pi\sqrt{\sum_{i=1}^{n}x_i^2})+0.1\sqrt{\sum_{i=1}^{n}x_i^2}$	$[-100,100]$	0
Stretched V-sine: $F_{26}(x)=\sum_{i=1}^{n-1}(x^2{}_i+2x_{i+1}^2)^{0.25}\cdot((\sin 50(x_i^2+x_{i+1}^2)^{0.1})^2+1)$	$[-10,10]$	0

表4 参数设置表

算法	主要参数
MSCA	$a=[2,0]$; $w_1=0.08$; $w_2=0.01$; $a_1=$; $a_2=0$; $\lambda=15$
SCA	$a=[2,0]$
WOA	$a_1=[2,0]$; $a_2=[-2,-1]$; $b=1$
ALO	$c=1.496$
GWO	$a=[2,0]$

从表 5 可以得出,MSCA 是 5 种算法中最优的。在测试的 26 个函数中,MSCA 有 25 个函数的测试值是比较算法中最好的。F1、F2、F3、F4、F6、F8、F9、F11、F12、F13、F14、F15、F16、F17、F18、F20、F21 、F22、F23、F24、F25、F26 这 22 个函数已经收敛到理论最优值 0,算法 MSCA 在函数 F5 的收敛精度还没有 ALO 和 SCA 的收敛精度高,但是 F7、F10、F19 对比其他 4 种算法的效果是最好的。

表5 5 种算法的寻优结果比较($D=30$)

函数	WOA Mean±St. dev	ALO Mean±St. dev	GWO Mean±St. dev	SCA Mean±St. dev	MSCA Mean±St. dev
F1	1.41E-30±4.91E-30	2.59E-10 ±1.65E-10	6.59E-28±6.34E-05	0.03797±0.11306	**0±0**
F2	1.06E- 21±2.39E-21	1.842E-06 ±6.58E-07	7.18E-17±0.02901	1.71E-5±2.29E-5	**0±0**
F3	5.39E-07±2.93E- 06	6.068E-10±6.34E-10	3.29E-06±79.14958	31.38346±29.36588	**0±0**
F4	0.07258±0.39747	1.361E-08±1.81E-09	5.61E-07±1.31509	22.74132±10.42242	**0±0**
F5	27.86558±0.76363	**0.34677±0.10958**	26.81258±69.90499	6.09354±4.10395	**21.5413±0.30615**
F6	2.37E-233±0	3.045E-16±3.61E-16	1.32E-101±6.81E-101	3.93E-4±0.00164	**0±0**
F7	0.00143±0.00115	0.00429±0.00509	0.00221±0.10029	0.03927±0.02989	**3.72E-5±4.52E-5**
F8	6.88E-225±0	738E-12±102.6E-12	3.64E-114±1.98E-113	1.41E-12±6.63E-12	**0±0**
F9	2.756E-223±0	0.93868±0.82482	1.22E-128±5.41E-128	1.11E-7±4.01E-7	**0±0**
F10	0.66674±1.12E-04	2.48632±3.00718	3.00718±**2.53E-06**	1.31291±2.42292	**0.66667±4.45E-6**
F11	9.74E-148±3.71E-147	4.53E-06±2.18E-06	7.71E-57±1.14E-56	0.44458±1.153659	**0±0**
F12	7.24E-76±2.002E-75	1.37411±1.35383	4.33E-29±4.55E-29	0.00264±0.00797	**0±0**
F13	4.49E-02±2.3 E-02	1.63E-02±15.77 E-02	7.32E-08±6.02E-08	6.51664±9.10461	**0±0**
F14	1.23E-110±3.06E-110	9.67E-07±4.95E-07	9.02E-96±2.85E-95	7.03E-8±2.51E-7	**0±0**
F15	0±0	0.27329±0.06858	0.310521±47.35612	1.73362±1.20117	**0±0**
F16	0±0	1.62045±0.38081	0±0	6.84E-6±3.09E-5	**0±0**
F17	0.00029±0.00159	0.08502±0.04005	0.00449±0.00666	0.29422±0.26342	**0±0**
F18	1.79E-07±5.86E-07	0±0	1.09E-05±2.38E-05	1.75E-4±2.17E-4	**0±0**
F19	7.4043±9.89757	0.00739±0.00709	1.06E-13±0.07784	16.35322±7.58252	**8.88E-16±0**
F20	0±0	0±0	14.52325±1.66214	4.81463±0.76677	**0±0**
F21	2.33E-81±1.27E-80	5.77897±5.52849	1.62E-04±5.11E-04	0.09711±0.26041	**0±0**
F22	0.00281±0.00149	0.12926±0.07168	0.00313±3.567E-10	0.06986±0.071598	**0±0**
F23	0.01955±0.01841	0.46511±0.03088	0.02623±0.01371	0.06888±0.05571	**0±0**
F24	0±0	11.41716±3.17911	0±0	0.00392±0.00861	**0±0**
F25	0.06991±0.04655	22.59654±19.17924	0.09987±9.59E-10	0.28696±0.46234	**0±0**
F26	1.96E-61±1.05E-60	4.12949±5.11201	3.058E-15±2.29E-15	0.23473±0.180675	**0±0**

表 6 是将 MSCA 与其他 2 种改进算法及原始算法 SCA 进行比较,2 种改进算法分别是:结合非惯性权重、参数 r_1 指数型递减、引入自适应变异因子的改进正余弦算法 ZSCA[10]。结合反向学习初始化、非线性惯性权重的改进正余弦算法 ISCA[9]。这 4 种算法,均在 $N=30$、最大迭代次数 $T_{max}=1000$,维数 $D=30$ 的设置下进行且每个函数独立运行 30 次。从表 6 可以看到 MSCA 在函数 F5 的收敛精度未有 SCA 的收敛精度高,但优于 ISCA、ZSCA,在 F5、F7、F10、F19 未能收敛到最优值 0,但函数 F7、F10、F19 对比其他 3 种算法的效果是最好的。ISCA 在 F5、F7、F10、F18、F19、F20 未能收敛到最优值 0,ZSCA 在 F2、F4、F5、F7、F10、F13、F18、F19、F20、F21、F23 均未收敛到最优值 0,说明改进后的 MSCA 是优于 ISCA 和 ZSCA 的。

表 6　MSCA 算法与其他改进算法的结果比较(D=30)

函数	SCA Mean ± St. dev	ISCA Mean ± St. dev	ZSCA Mean ± St. dev	MSCA Mean ± St. dev
F1	0.03797±0.1130563	0±0	0±0	**0±0**
F2	1.7128E-5±2.289E-5	0±0	4.435214E-293±0	**0±0**
F3	31.383462±29.36588	0±0	0±0	**0±0**
F4	22.74132±10.42242	0±0	7.111191E-270±0	**0±0**
F5	**6.093537±**4.103953	28.42989±0.401724	28.17187±**0.208449**	21.5413±0.306146
F6	3.9296E-4±0.001642	0±0	0±0	**0±0**
F7	0.03927±0.02989	7.58785E-5±6.4461E-5	5.174E-5±4.4288E-5	**3.7176**E-**5±4.522**E-**5**
F8	1.4131E-12±6.632E-12	0±0	0±0	**0±0**
F9	1.1135E-7±4.0045E-7	0±0	0±0	**0±0**
F10	1.31291±2.42292	0.666667±3.2703E-7	0.66667±2.97434E-8	**0.666669±4.4451**E-**6**
F11	0.444575±1.153659	0±0	0±0	**0±0**
F12	0.002644±0.007969	0±0	0±0	**0±0**
F13	6.516644±9.104613	0±0	2.10339E-286±0	**0±0**
F14	7.0250E-8±2.5054E-7	0±0	0±0	**0±0**
F15	1.7336223±1.2011697	0±0	0±0	**0±0**
F16	6.8380E-6±3.0951E-5	0±0	0±0	**0±0**
F17	0.294218±0.263424	0±0	0±0	**0±0**
F18	1.7481E-4±2.1694E-4	4.6503E-9±2.54706E-8	6.3802E-5±1.19334E-4	**0±0**
F19	16.353218±7.582516	8.88178E-16±0	8.88179E-16±0	**8.88178**E-**16±0**
F20	4.8146246±0.7667658	4.793315±0.374985	10.21701±8.29158	**0±0**
F21	0.0971116±0.2604008	0±0	2.773018E-292±0	**0±0**
F22	0.0698637±0.0715983	0±0	0±0	**0±0**
F23	0.0688781±0.0557055	0±0	0.00582956±0.0048412	**0±0**
F24	0.0039237±0.0086098	0±0	0±0	**0±0**
F25	0.2869611±0.4623359	0±0	0±0	**0±0**
F26	0.2347305±0.1806752	0±0	0±0	**0±0**

文献[14]给出了一种对优化算法的排序。Mean Absolute Error(MAE)平均绝对误差是一种简单有效的性能指标,其计算公式如下:

$$\mathrm{MAE}=\frac{\sum_{i=1}^{n}|x_i-m(x)|}{n} \tag{30}$$

式中:x_i 是最优结果的平均值,$m(x)$ 为相应的理论最优值,n 为基准函数个数。计算出的 MAE 如表 7 所示。

表 7　MAE 测试算法排名

算法	MAE	Rank
MSCA	1.163286	1
SCA	1.04519E+11	5
WOA	34.4430299	3
ALO	335056.2393	4
GWO	3.01345965	2

由表 7 知,MSCA 算法的 MAE 最小,排名为 1,进一步说明 MSCA 改进的有效性。

4.2.2　对大规模维数测试函数测试结果

为测试改进后的算法有解决超大规模维数的优化问题的能力,本文将算法 MSCA 与 SCA、ISCA、ZSCA 分别在 1000 维、2000 维、5000 维下进行测试得出其最优值 Best、平均值 mean、标准差 std。以下测试均在相同的实验参数下进行,N=30,$T_{\max}$ = 1000,每个测试函数独立运行 30 次。加粗的数据为较好的实验结果。

表 8 是 MSCA 与 SCA、ISCA、ZSCA 在 1000 维下的测试结果比较,可以看到,SCA 算法在函数 F1、F3、F5、F7、F8、F10、F11、F13、F15、F24 这些函数的测试中已经出现了维数灾难,4 种算法在 F5、F7、F10、F19 这 4 个函数上都未收敛到最优值 0,但并未出现维数灾难。而改进算法 ZSCA 除了原本在 D = 30 维时,F2、F4、F5、F7、F10、F13、F18、F19、F20、F21、F23 这些函数未收敛到最优值 0 还新增了 F3、F11、F12、F22、F25、F26 这些函数未收敛到原本的最优值 0。ISCA 除了原本在低维时 F5、F7、F10、F18、F19、F20 未能收敛到最优值 0,还新增了 F3、F4、F13、F21 这些函数未能收敛到 0。MSCA 与低维时相比较无明显变化,依旧在 F1、F2、F3、F4、F6、F8、F9、F11、F12、F13、F14、F15、F16、F17、F18、F20、F21、F22、F23、F24、F25、F26 这 22 个函数收敛到理论最优值 0。

随着维数的增大,MSCA 在 F5、F7、F10、F19 这 4 个函数的算法效果虽然比低维时有所减弱,但比其他 3 种算法更好。总的来说,MSCA 在 1000 维的大规模优化问题的寻优能力是高于其他算法的,并未出现维数灾难。

表 8 4 种算法的实验结果比较(D=1000)

函数	SCA Mean ± St. dev	ISCA Mean ± St. dev	ZSCA Mean ± St. dev	MSCA Mean ± St. dev
F1	1.61E+05±5.48E+04	0±0	0±0	**0±0**
F2	69.51990±37.1748	0±0	5.21E-287±0	**0±0**
F3	7.11E+06±1.10E+06	6.79E-254±0	1.44E-209±0	**0±0**
F4	99.09050±0.33099	8.37E-93±3.91E-92	23.13165±42.60717	**0±0**
F5	1.84E+09±4.25E+08	4.99E+02±0.05068	4.99E+02±0.26658	4.99E+02±**0.00978**
F6	0.02371±0.04818	0±0	0±0	**0±0**
F7	1.57E+04±3.98E+03	2.33E-04±3.06E-04	1.18E-04±1.19E-04	**6.73**E-**05**±**5.65**E-**05**
F8	2.26E+19±3.99E+18	0±0	0±0	**0±0**
F9	5.82912±3.63219	0±0	0±0	**0±0**
F10	2.18E+08±5.79E+07	0.68827±0.08224	**0.66667**±7.38E-07	0.99999±**2.64**E-**11**
F11	3.89E+09±2.15E+09	0±0	8.30E-264±0	**0±0**
F12	500760.25±202031.58	0±0	1.425E-262±0	**0±0**
F13	2.22E+04±4.89E+04	5.71E-130±3.12E-129	8.25E-42±4.50E-41	**0±0**
F14	3.37E-5±6.00E-5	0±0	0±0	**0±0**
F15	1.22E+03±4.99E+02	0±0	0±0	**0±0**
F16	0.00212±0.00461	0±0	0±0	**0±0**
F17	1556.72642±593.33112	0±0	0±0	**0±0**
F18	3.87E-4±4.03E-4	4.26E-11±2.33E-10	4.27E-5±1.02E-4	**0±0**
F19	19.85073±2.83046	8.88E-16±0	1.24E-15±1.08E-15	**8.88**E-**16±0**
F20	196.48231±6.72505	134.84102±2.74592	306.36228±6.71294	**0±0**
F21	162.31391±72.52722	4.89E-170±0	1.11E-134±6.05E-134	**0±0**
F22	0.49999±2.12044E-6	0±0	1.04836E-4±5.74211E-4	**0±0**
F23	0.49999±1.78E-06	0±0	0.00939±0.00177	**0±0**
F24	11925.57159±5137.39147	0±0	0±0	**0±0**
F25	12.15569±29.67712	0±0	2.654133E-250±0	**0±0**
F26	2.8757±1.65594	0±0	1.06E-71±5.26E-71	**0±0**

表 9 是 MSCA 与 SCA、ISCA、ZSCA 在 2000 维下的测试结果比较,可以看到,SCA 算法在函数 F1、F2、F3、F5、F7、F8、F10、F11、F12、F13、F15、F17、F20、F21、F24 这些函数的测试中已经出现了维数灾难,4 种改进算法都在 F5、F7、F10、F19 这 4 个函数上都未收敛到最优值 0。而改进算法 ZSCA 除了原本在 D=30 时,F2、F4、F5、F7、F10、F13、F18、F19、F20、F21、F23 这些函数未收敛到最优值 0 还新增了 F1、F2、F3、F11、F12、F22、F25、F26 这些函数未收敛到原本的最优值 0。ISCA 除了原本在低维时 F5、F7、F10、F18、F19、F20 未能收敛到最优值 0,还新增了 F2、F3、F4、F13、F21、F25 这些函数未能收敛到 0。MSCA 与低维时相比较无明显变化。随着维数的增大,MSCA 在 F5、F7、F10、F19 这 4 个函数的算法效果虽然比低维时有所减弱,但比其他 3 种算法更好。总的来说,MSCA 在 2000 维的大规模优化问题的寻优能力是高于其他算法的。

表 10 是 MSCA 与 SCA、ISCA、ZSCA 在 5000 维下的测试结果比较,可以看到,SCA 算法在函数 F1、F2、F3、F5、F7、F8、F10、F11、F12、F13、F15、F20、F21、F24 这些函数的测试中已经出现了维数灾难。而改进算法 ZSCA 除了原本在 D=30 的低维时 F2、F4、F5、F7、F10、F13、F18、F19、F20、F21、F23 这些函数未收敛到最优值 0 还新增了 F1、F3、F11、F12、F22、F25、F26 这些函数未收敛到原本的最优值 0。ISCA 除了原本在低维时 F5、F7、F10、F18、F19、F20 未能收敛到最优值 0,还新增了 F2、F3、F4、F13、F21、F25 这些函数未能收敛到 0。MSCA 与低维时相比较无明显变化。随着维数的增大,MSCA 在 F5、F7、F10、F19 这 4 个函数的算法效果虽然比低维时有所减弱,但比其他 3 种算法更好。总的来说,MSCA 在 5000 维的大规模优化问题的寻优能力是高于其他算法的,并未出现维数灾难。

表 9　4 种算法的实验结果比较(D=2000)

函数	SCA Mean ± St. dev	ISCA Mean ± St. dev	ZSCA Mean ± St. dev	MSCA Mean ± St. dev
F1	486038.809±184521.624	0±0	1.38E-262±0	**0±0**
F2	2756.95664±23.89563	3.05E-175±0	2.77E-137±1.09E-136	**0±0**
F3	2.67E+7±5946496.51	1.13E-230±0	1.11E-207±0	**0±0**
F4	99.56142±0.09715	9.86E-83±5.38E-82	99.6223±0.11099	**0±0**
F5	4.11E+09±9.51E+08	9.99E+02±0.05519	9.99E+02±0.27209	9.99E+02±**0.0075**
F6	0.01059±0.02277	0±0	0±0	**0±0**
F7	68533.832±15363.51778	1.60E-04±1.23E-04	1.38E-04±1.12E-04	**4.44**E-**05±4.09**E-**05**
F8	5.51E+19±9.72E+18	0±0	0±0	**0±0**
F9	18.24811±14.60892	0±0	0±0	**0±0**
F10	1.05E+09±1.98E+08	0.68661±0.0764	**0.66667±**2.85E-07	**1±0**
F11	1.39E+10±6.49E+09	0±0	1.22E-257±0	**0±0**
F12	2.34E+06±7.48E+05	0±0	1.05E-258±0	**0±0**
F13	1.51E+08±5.01E+08	1.69E-105±6.43E-105	1.59E-23±5.42E-23	**0±0**
F14	2.94E-04±0.00109	0±0	0±0	**0±0**
F15	1.99E+03±1.02E+03	0±0	0±0	**0±0**
F16	0.00412±0.00792	0±0	0±0	**0±0**
F17	2678.9745±645.2431	0±0	0±0	**0±0**
F18	3.51E-04±3.43E-04	7.67E-09±3.65E-08	3.37E-05±8.17E-05	**0±0**
F19	18.74872±3.88965	8.88E-16±0	1.01E-15±6.49E-16	**8.88**E-**16±0**
F20	4.11E+02±8.22844	2.76E+02±3.2979	6.32E+02±11.21445	**0±0**
F21	2.61E+02±1.03E+02	1.21E-164±0	1.88E-140±7.51E-140	**0±0**
F22	0.49999±5.00E-07	0±0	4.17E-04±0.00108	**0±0**
F23	0.49999±7.04E-07	0±0	0.00874455±0.00297	**0±0**
F24	3.28E+04±9.03E+03	0±0	0±0	**0±0**
F25	22.61287±48.7877	5.56E-302±0	4.48E-267±0	**0±0**
F26	2.66977±1.80511	0±0	9.54E-74±3.25E-73	**0±0**

表 10　4 种算法的实验结果比较(D=5000)

函数	SCA Mean ± St. dev	ISCA Mean ± St. dev	ZSCA Mean ± St. dev	MSCA Mean ± St. dev
F1	7.37E+05±2.35E+05	0±0	2.99E-263±0	**0±0**
F2	5272.6124±4782.64363	1.05E-172±0	2.01E-138±1.02E-137	**0±0**
F3	6.13E+07±1.05E+07	2.48E-248±0	8.31E-202±0	**0±0**
F4	99.73619±0.06249	1.01E-81±4.16E-81	99.73862±0.09331	**0±0**
F5	6.86E+09±1.11E+09	1.50E+03±0.01009	1.50E+03±0.30068	1.50E+03±**1.06**E-**02**
F6	0.00939±0.0166	0±0	0±0	**0±0**
F7	1.57E+05±3.55E+04	1.62E-04±1.26E-04	2.20E-04±2.87E-04	**6.87**E-**05±6.87**E-**05**
F8	8.69E+19±1.26E+19	0±0	0±0	**0±0**
F9	24.61622±18.49538	0±0	0±0	**0±0**
F10	2,23E+10±1.98E+09	0.69667±0.05621	**0.66657±**1.32E-7	1.24169±**0**
F11	2.47E+11±9.64E+09	0±0	2.43E-236±0	**0±0**
F12	5.62E+06±2.27E+06	0±0	1.82E-258±0	**0±0**
F13	4.63E+09±1.05E+10	2.19E-92±1.19E-91	2.61E-13±1.37E-12	**0±0**
F14	2.14E-04±6.02E-04	0±0	0±0	**0±0**
F15	2.41E+03±5.67E+03	0±0	0±0	**0±0**
F16	0.00331±0.00658	0±0	0±0	**0±0**
F17	3745.4657±764.1358	0±0	0±0	**0±0**
F18	4.31E-04±4.39E-04	1.77E-09±7.82E-09	1.11E-04±2.23E-04	**0±0±**
F19	20.74872±1.96586	8.88178E-16±0	0.87E-15±8.94E-17	**8.88178**E-**16±0**
F20	8.64E+04±10.47931	5.66E+02±3.8445	6.76E+03±14.3445	**0±0**
F21	4.69E+04±1.43E+04	5.66E-156±0	2.77E-124±8.94E-124	**0±0**
F22	0.49999±2.75E-07	0±0	6.25E-04±0.00127	**0±0**
F23	0.499999±2.68E-07	0±0	0.00809±0.00368	**0±0**
F24	5.23E+04±1.66E+04	0±0	0±0	**0±0**
F25	15.1284±39.99733	6.03E-321±0	7.95E-265±0	**0±0**
F26	3.22547±2.1031	0±0	4.78E-75±2.89E-75	**0±0**

4.2.3 实例分析及仿真

由于正余弦优化算法主要应用于无约束优化问题,而优化模型是有约束的优化问题。在优化之前,必须对模型进行一定的转换。这里引入外罚函数法将有约束的优化问题转变为无约束优化问题。

$$\min f(x)=\min z+k_1\sum_{i=2}^{m}\{\max\{0,T_{\min}-(t_i-t_{i-1})\}\}+$$
$$k_2\sum_{i=2}^{m}\{\max\{0,(t_i-t_{i-1})-T_{\max}\}\}+k_3\sum_{i=2}^{m-1}\{\max\{0,|(t_{i+1}-t_i)-$$
$$(t_i-t_{i-1})|-\varepsilon\}\}+k_4\sum_{j=1}^{n}\{\max\{0,t_{sj}-t_{tj}\}\}+$$
$$k_5\max\{0,\sum_{i=1}^{m}\sum_{j=1}^{n}r_j tP/m\times Q-\theta\} \tag{31}$$

其中, k_1、k_2、k_3、k_4 和 k_5 是影响条件的惩罚函数系数,即惩罚系数。

选取 A 市 11 路公交运营线为例,11 路共有 24 个站点。调查后,城镇单位职工平均年薪在 2020 年下半年为 9805 元/人,除去周末和公共假期后计算,半年休息时间约为 62 天,即工作 120 天,按每天平均工作时间为 8 小时计算,平均每分钟转化率为0.17元。

为更好地进行实验,本文给出参数 k_1,k_2,k_3,k_4,k_5 分别为1.4、2.2、2.1、1.8、1。利用 MATLAB 进行的仿真实验可以计算出图 2 ~4 所示的公交线路仿真,可以很好地再现公交线路上人数的变化,为公交规划系统提供了很好的参考。

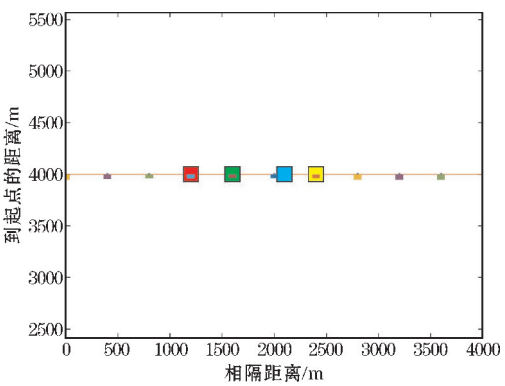

图 2 公交起步阶段

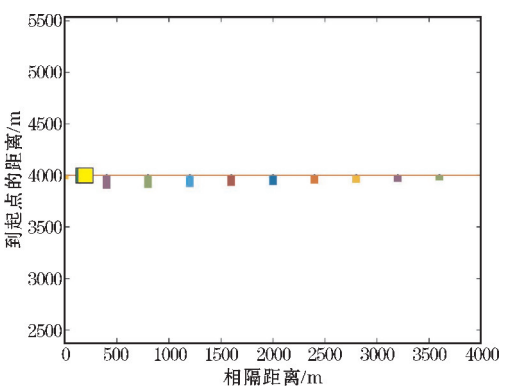

图 3 上班高峰时间阶段

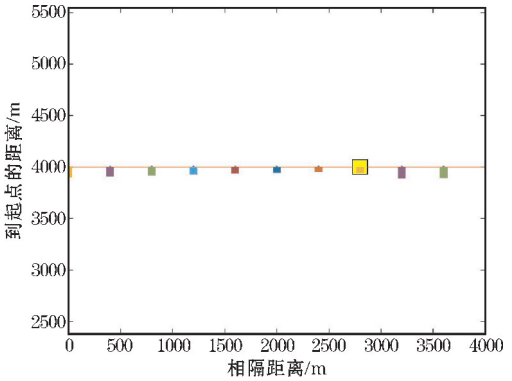

图 4 候车人数增加阶段

通过 MATLAB 计算,从实验进程可以得知,随着代数的增加,群体中最佳个体的目标函数值有明显的收敛趋势,且改进后的 MSCA 算法比原有算法 SCA 的收敛速度更快、收敛精度更高。最优个体目标值随进化代数的变化分别如图 5 所示。

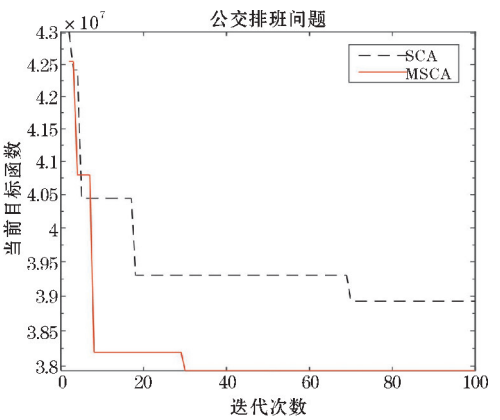

图 5 最优目标值变化

从而得到 A 市 11 路公交全天的发车时刻表如下:

6:06-6:02-6:11-6:18-6:29-6:40-6:53-7:01-7:07-7:12-7:18-7:24-7:32-7:43-7:54-8:04-8:15-8:24-8:34-8:42-8:53-9:05-9:14-9:27-9:37-9:42-9:49-9:58-10:10-10:19-10:30-10:41-10:48-10:58-11:11-11:23-11:35-11:44-11:51-12:00-12:12-11:22-12:32-12:54-13:04-13:17-13:27-13:34-13:42-13:55-14:06-14:14-14:26-14:34-14:45-14:55-15:01-15:09-15:17-15:26-15:38-15:48-15:54-16:00-16:08-16:18-16:25-16:32-16:37-16:47-16:54-17:03-17:14-17:24-17:29-17:35-17:43-17:49-17:59-18:06-18:13-18:20-18:27-18:37-18:42-18:49-18:59-19:07-19:16-19:27-19:38-19:51-20:01-20:06-20:22-20:33-20:40-20:51-21:00。

另外,SCA 算法和改进的正余弦算法 MSCA 的发车频率分布如图 6、图 7 所示。

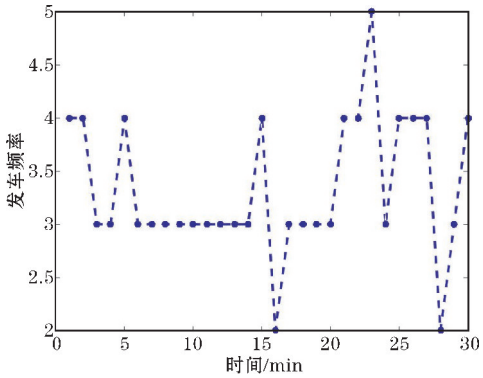

图 6　传统 SCA 全天发车频率图

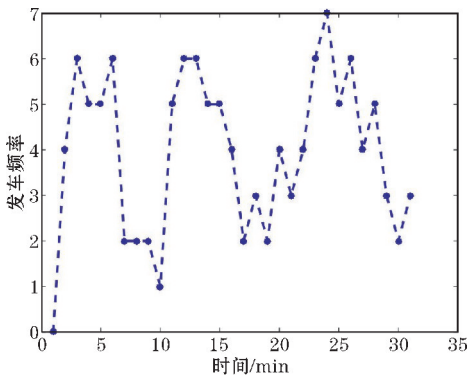

图 7　改进 MSCA 全天发车频率图

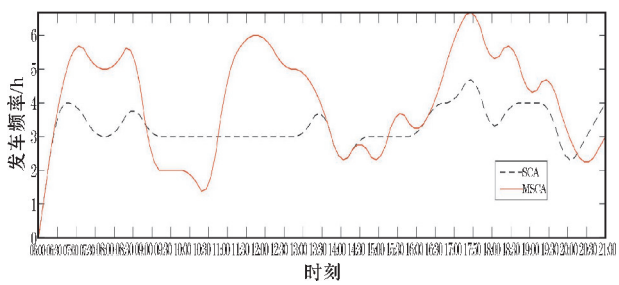

图 8　传统 SCA 和改进 MSCA 算法全天发车频率对比图

图 8 为 2 种算法发车频率对比图,可见改进的 MSCA 的最优目标函数值优于原有算法 SCA,且改进的 MSCA 算法有 3 个明显的峰值,分别对应早晚出行的 2 个高峰时间和午休的小高峰时间,更符合客流的实际分布。充分反映了公交运营商和乘客的共同利益,并进一步说明改进的 MSCA 算法是高效的,可以在巨大的规划优化搜索空间中找到更可靠的最优或近似解。

5　结束语

在原正余弦算法基础上,提出一种基于正态变异自适应正余弦算法和一种平衡勘探开发的非线性转换参数递减策;引入惯性权重来修正位置更新方程,保留更多的信息和产生有前途的候选解;改进了基于正态变异算子的位置更新策略,增加了局部搜索空间。通过 26 个国际标准测试函数进行实验,得出相关算法的平均值、标准差和最优解来进行检验。结果表明,改进算法在收敛精度和收敛速度上均优于比较对比算法。在解决超大规模优化问题时,MSCA 算法并未出现维数灾难,保持了较好的收敛精度和稳定性。而原算法 SCA 随着维数的增加出现了不同程度的维数灾难,维数越高越不稳定。除此之外,还将改进后的 MSCA 应用在城市公交排班中,通过实验仿真得到改进的 MSCA 的最优目标函数值优于原有算法 SCA,以及公交排班符合客流的实际分布。也充分反映了公交运营商和乘客的共同利益。在下一步的研究中,将考虑将改进后的 MSCA 应用到图像处理等实际问题中。

参考文献:

[1] Kennedy J, Eberhart R. Particle swarm optimization [C]. In: Proceeding of International Conference on Neural Networks. Perth, Australia: IEEE, 1995: 1942-1948.

[2] Mirjalili S, Lewis A. The Whale Optimization Algorithm [J]. Advances in Engineering Software, 2016, 95: 51-67.

[3] Arora S, Singh S. Butterfly optimization algorithm: a novel approach for global optimization [J]. Soft Computing, 2019, 23(3): 715-734.

[4] Mirjalili S. Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm [J]. Knowledge-Based Systems, 2015, 89(11): 228-249.

[5] Wang Gaige, Deb S, Cui Zhihua. Monarch butterfly optimization [J]. Neural Computing and Applications, 2019, 31(7): 1-20.

[6] Wang Gaige, Suash D, Santos C L D. Earthworm optimization algo-rithm: a bio-inspired metaheuristic algorithm for global optimization problems [J]. International Journal of Bio-Inspired Computation, 2018, 12(1): 1-22.

[7] Wang Gaige, Deb S, Coelho L D S. Elephant herding optimization [C]. In: Proceeding of International Symposium on Computational and Business Intelligence. Bali, Indonesia: IEEE, 2015: 1-5.

[8] Seyedali Mirjalili. SCA: A Sine Cosine Algorithm for solving optimization problems [J]. Systems, 2016, 96: 120-133.

[9] 徐松金,龙文. 求解高维优化问题的改进正弦余弦

算法[J]. 计算机应用研究,2018,35(9):20-23.
[10] 张校非,白艳萍,郝岩,等. 改进的正弦余弦算法在函数优化问题中的研究[J]. 重庆理工大学学报(自然科学版),2017,31(2):146-152.
[11] Long W, Wu T, Liang X, et al. Solving high-dimensional global optimization problems using an improved sine cosine algorithm[J]. Expert Systems with Applications,2018,123:108-126.
[12] Mirjalili,Seyedali. The Ant Lion Optimizer[J]. Advances in Engineering Software,2015,83:80-98.
[13] Mirjalili S, Mirjalili S M, Lewis A. Grey Wolf Optimizer[J]. Advances in Engineering Software, 2014,69:46-61.
[14] Mirjalili S, Dong J S, Lewis A. Ant Colony Optimizer: Theory, Literature Review, and Application in AUV Path Planning: Methods and Applications [M]. Nature-Inspired Optimizers,2020.
[15] Kroon J D, Laan P V D. A generalization of Friedman´s rank statistic[J]. Statistica Neerlandica, 2010,37(1):1-14.
[16] Van Elteren P. On the combination of independent two-sample test of Wilcoxon[J]. Chemosphere,1960,37(3):81-87.
[17] Goldberg D E. Genetic Algorithm in Search Optimization and Machine Learning[J]. Addison Wesley,1989,xiii(7):2104-2116.
[18] Rudolph G. Local convergence rates of simple evolutionary algorithms with Cauchy mutations[J]. Evolutionary Computation IEEE Transactions on, 1998,1(4):249-258.
[19] Emad N. A modified flower pollination algorithm for global optimiza-tion[J]. Expert Systems with Applications,2016,57(9):192-203.
[20] Yongjun S, Xilu W, Yahuan C, et al. A modified whale optimization algorithm for large-scale global optimization problems[J]. Expert Systems with Applications,2018,27(8):563-577.
[21] 杨磊,刘卫朋,周磊. 基于改进的随机公交调度问题的数学模型[J]. 河北工业大学学报, 2010,39(1):74-78.

Application of Improved Sines and Cosines Optimization Algorithm in Bus Scheduling Model

HUANG Yan, WU Zezhong
(College of Applied Mathematics, Chengdu University of Information Technology, Chengdu 610225, China)

Abstract: Aiming at the shortcomings of the sine-cosine algorithm in the search process, such as low convergence accuracy and easy to fall into local optimum, this paper proposes an improved sine-cosine algorithm (MSCA) based on normal mutation adaptation. First of all, the mutation operation is introduced into the SCA algorithm to initialize the population; Secondly, the inertial weights are introduced to modify the position update equation to retain more information and generate promising candidate solutions. A non-linear conversion parameter decline strategy for balanced exploration and development is proposed, which jumps out of the local optimal solution state during the search process. Finally, the location update strategy based on the normal mutation operator is improved, and the local search space is increased. Then 26 international standard test functions are seleted to test the improved algorithm. The results show that the MSCA algorithm is better than other improved algorithms in terms of convergence accuracy, convergence speed and convergence stability. In addition, the improved MSCA is also used in the practical application of urban bus scheduling. The optimal objective function value of the improved MSCA obtained through experimental simulation is better than the original algorithm SCA, and the bus scheduling is in line with the actual distribution of passenger flow. It fully reflects the common interests of bus operators and passengers.

Keywords: sine cosine algorithm; normal variation factor; large-scale optimization problem; bus scheduling