文章编号:1671 - 6833(2018)03 - 0034 - 06

求解柔性作业车间调度问题的细菌算法对比及改进

吴秀丽, 张志强

(北京科技大学 机械工程学院,北京 100083)

摘 要: 为充分探讨细菌系列算法求解离散优化问题的能力,针对柔性作业车间调度问题,采用细菌趋化算法、细菌群体趋化算法、细菌进化算法、细菌群游算法和细菌觅食优化算法进行求解. 首先建立了以完成时间为目标的柔性作业车间调度问题模型,然后用 5 种细菌算法进行求解,数值试验结果表明:细菌觅食算法的寻优能力最强. 接着,进一步对细菌觅食算法进行了改进,针对其关键操作设计了数十种算子,最终得到优化能力最强的算法结构和算子组合. 最后的数值实验表明,改进的细菌觅食算法寻优能力及稳定性大幅提升,体现出非常好的全局开发能力和局部搜索能力.

关键词: 柔性作业车间调度; 细菌趋化算法; 细菌群体趋药性算法; 细菌觅食算法; 细菌群游算法; 细菌进化算法

中图分类号: TP18 **文献标志码**: A doi:10. 13705/j. issn. 1671 - 6833. 2017. 06. 018

0 引言

生产调度对于提高制造企业运作效率、降低成本具有举足轻重的作用,该问题也是典型的优化难题,吸引了众多学者开展研究. 近年来,Brukcer 等[1]提出了柔性作业车间调度问题(flexible job shop scheduling problem, FJSP). FJSP 包含两个子问题:路由问题和调度问题,分别表示工序的机器分配问题和调度问题. FJSP 将机器柔性引入传统 job-shop 调度模型,更接近于实际生产调度环境,增加了调度灵活性,但问题复杂度也相应增加,传统数学方法难以在有限时间内求解,因此众多学者采用智能优化算法研究该问题[2-3].

细菌算法作为一个较为新颖的算法,受到了学者的广泛关注. 细菌算法都是基于细菌的生物特性来实现的,其一是基于细菌觅食过程;其二是基于细菌进化过程. 前者由于对细菌觅食过程的拆解和侧重,衍生出一系列其它算法,如细菌趋化算法、细菌群体趋化算法、细菌群游算法和细菌觅食算法等;后者发展出细菌进化算法.

细菌趋化算法最初由 Bremermann 等[4]提出,后经 Müller 等[5-6]进一步地研究与综合,提出了细菌趋化(bacterial chemotaxis, BC)算法. 该算法是优化领域的一种新的仿生学进化算法,其利用细菌对环境中引诱剂的应激反应行为来进行函数优化. 国内最早研究 BC 算法的是李威武等[7],他提出细菌群体趋药性算法(bacterial colony chemotaxis, BCC). 细菌觅食优化算法(bacterial foraging optimization algorithm, BFO)最初由 Passino[8]2002 年提出,借鉴了大肠杆菌在觅食过程中的趋化、群聚、繁殖和消除-扩散等运动特性,设计了优化算法. 在此基础上,Tang 等[9]提出细菌群游算法(bacterial swarming algorithm, BSA),Chu 等[10]提出快速细菌群游算法(fast bacterial swarming algorithm, FBSA). Nawa 等[11]提出了细菌进化算法(bacterial evolutionary algorithm, BEA).

笔者设计了多种算法结构并进行对比实验,以确定算法的不同特性对问题的影响,并将其中性能最佳的细菌觅食算法作为主要研究对象,在其基础上进行改进和集成,以得到一个寻优能力更强的改进细菌觅食算法(IBFO).

1 柔性作业车间调度问题建模

1.1 问题描述

FJSP 问题可以这样描述:n 个工件在 m 台机

收稿日期:2017 - 04 - 10;**修订日期**:2017 - 07 - 16

基金项目:国家自然科学基金青年科学项目(51305024,71301054)

作者简介:吴秀丽(1977—),女,河南鹤壁人,北京科技大学副教授,博士,主要从事制造过程智能优化调度算法研究,E-mail:wuxiuli@ ustb. edu. cn.

器上加工,每个工件有多道工序,工序顺序是预先确定的,每道工序可由多台机器加工,不同机器对同一工序的加工时间并不相同. 调度过程中主要解决两个子问题:一是机器分配问题;二是工件排序问题. 调度目标是通过机器分配和工序排序而优化某个(些)目标,如完工时间为优化目标.

此外,加工过程还要满足约束条件:①所有机器在 $t=0$ 时刻都可用;②所有工件在 $t=0$ 时刻都可被加工;③所有工件的工艺计划都是固定不变的;④工序在可用机器上的加工时间是确定的;⑤每个工件在固定时刻只能在一台机器上加工,且一旦开始加工不能中断;⑥加工是非抢占式的.

1.2　模型建立

建模所用变量如下:

n——工件总数;

m——机器总数;

n_i——工件 i 的工序总数;

i, h——工件号索引,i, $h=1, 2, \cdots, n$;

j, g——工序号索引,j, $g=1, 2, \cdots, n_i$;

k——机器号索引,$k=1, 2, \cdots, m$;

O_{ij}——工件 i 的第 j 道工序;

O_{ijk}——工序 O_{ij}选择在机器 k 上加工;

p_{ijk}——工序 O_{ij}选择在机器 k 上加工所耗费的时间;

S_{ij}——工序 O_{ij}的可用机器集合,$S_{ij} \subset \{1,2,\cdots,m\}$;

C_{ij}——工序 O_{ij}的完工时间;

$C_{\max}$——调度方案的最大完工时间 Makespan;

X_{ijk}——决策变量,若 $X_{ijk}=1$ 表示工序 O_{ij}选择在机器 k 上加工 O_{ijk},否则 $X_{ijk}=0$;

Y_{hgij}——决策变量,若 $Y_{hgij}=-1$ 表示工序 O_{hg}为 O_{ij}相邻的前一道工序;$Y_{hgij}=1$ 表示工序 O_{hg}为 O_{ij}相邻的后一道工序;$Y_{hgij}=0$ 表示 O_{hg}和 O_{ij}为不相邻的两道工序.

通常情况下,高效快速地完成生产任务是绝大多数生产企业追求的第一目标,因此设定优化目标函数是最大完工时间,即

$$\min C_{\max} = \min(\max(C_{ij})). \tag{1}$$

s. t.

$$C_{ij} - C_{i(j-1)} \geqslant p_{ijk}X_{ijk}, j = 2,3,\cdots,n_i; \tag{2}$$

$$(C_{ij} - C_{hg} - p_{ijk})X_{hgk}X_{ijk}\frac{Y_{hgij}}{2}(Y_{hgij} - 1) + (C_{hg} - C_{ij} - p_{hgk})X_{hgk}X_{ijk}\frac{Y_{hgij}}{2}(Y_{hgij} + 1) \geqslant 0, \forall (h,g),(i,j),k; \tag{3}$$

$$\sum X_{ijk} = 1, k \in S_{ij}, \forall i,j; \tag{4}$$

$$X_{ijk} \in \{0,1\}; \tag{5}$$

$$Y_{hgij} \in \{-1,0,1\}. \tag{6}$$

其中,式(1)是目标函数;式(2)是工艺约束;式(3)是机器约束;式(4)限定一道工序只能在一台机器上独立完成;式(5)和式(6)限定决策变量的取值范围.

2　细菌系列算法性能比较

为了求解 FJSP,必须建立该问题模型与算法的映射关系,采用基于工序的编码方式[12]来实现,并依据活动化调度方式生成具体的调度方案.

为了对比细菌系列算法与其他进化算法的性能,对 FJSP 的标准问题 Kacem[13]的 5 个标准算例和 Brandimarte[3]的 10 个标准算例进行实验,并与最新的研究成果进行对比分析. 平均离差率为一个衡量算法寻优能力的指标:

$$M = \frac{1}{n}\sum_{i=1}^{n}\frac{F_i - F_{io}}{F_{io}}, \tag{7}$$

式中:M 为平均离差率;n 为试验算例数目;i 为试验号;F_i 为试验 i 的试验最优值;F_{io} 为试验 i 的目前最优值.

这些算法具有不同的参数,笔者在保证各算法具有可比性的前提下综合考虑各个参数的经验值[4-11],得到细菌系列算法的参数见表 1.

表 1　细菌算法的参数

Tab. 1　Parameters of bacterial algorithms

算法	参数	取值	算法	参数	取值
BC	迭代代数	5 000	BEA	迭代代数	100
	邻域个数	50		种群规模	20
BCC	迭代代数	100		克隆体个数	20
	种群规模	50		基因段个数	2
	趋化次数	50		感染次数	10
BSA	种群规模	50	BFO	种群规模	50
	趋向次数	100		趋向性次数	50
	转基因次数	50		复制次数	5
	转基因概率	0.7		驱散次数	20
				驱散概率	0.8

计算对比如表 2 和表 3 所示. 可以看出,对于 Kacem 算例,各算法达到最优值的成功率均为 100%;对于 Brandimarte 算例,BC、BEA 和 BSA 成功率为 40%,BCC 为 50%,BFO 为 60%. 并且对于 Brandimarte 算例,各个算法的离差率均较低,说明各算法性能都较好;BFO 的平均离差率最低,说明其性能最好,其次依次为 BCC、BSA、BEA、

BC. 考虑成功率和平均离差率的结果,5 种算法的性能排序为 BC < BEA < BSA < BCC < BFO.

表 2 Kacem 算例计算结果

Tab. 2 Results for Kacem benchmarks

(工件,机器)	最优[14]	BC	BCC	BEA	BSA	BFO
(4,5)	11	11	11	11	11	11
(8,8)	14	14	14	14	14	14
(10,7)	11	11	11	11	11	11
(10,10)	7	7	7	7	7	7
(15,10)	11	11	11	11	11	11
成功率	—	100%	100%	100%	100%	100%

表 3 Brandimarte 算例计算结果

Tab. 3 Results for Brandimarte benchmarks

(工件,机器)	最优[14]	BC	BCC	BEA	BSA	BFO
(10,6)	40	40	40	40	40	40
(10,6)	26	28	26	28	27	26
(15,8)	204	204	204	204	204	204
(15,8)	60	66	63	67	66	60
(15,4)	172	177	174	177	174	173
(10,15)	58	63	62	62	60	60
(20,5)	139	144	141	142	141	141
(20,10)	523	523	523	523	523	523
(20,10)	307	307	307	307	307	307
(20,15)	197	223	223	221	218	218
成功率/%	—	40.00	50.00	40.00	40.00	60.00
平均离差率/%	—	4.60	2.77	4.49	3.06	1.61

3 细菌觅食算法改进设计

由对比实验可知,细菌觅食算法性能最优,因此在细菌觅食算法的基础上,试验多种优化算子,以求得到性能更优的算法. 细菌觅食算法三大操作中,趋向性操作对算法影响的显著性最高,其次是驱散操作,而复制操作最低. 因此,笔者着重进行了趋向性操作算子和驱散操作算子的优化试验,通过改变其操作算子的操作方式,试验不同算子的寻优能力.

为保证试验的可靠性,遵循单一变量的试验原理,在每改变一个算子操作方式后均进行单独试验,其他试验条件不变. 试验过程中涉及主要参数设置如表 4 所示.

3.1 驱散操作改进

(1)随机驱散. 细菌觅食算法原始的驱散方式是以一定概率随机产生个体.

(2)基于工件驱散. 借鉴基于工件的编码方式,以工件进行编码,解码过程中优先排完一个工件所有工序才进入下一个工件的排序.

表 4 试验参数设置

Tab. 4 Parameters configuration

参数名称	参数值
种群规模	10
趋向性操作次数	30
复制操作次数	3
驱散(迁徙)操作次数	5
驱散(迁徙)概率	0.7
试验次数	10

(3)基于最长工件驱散. 优先加工总加工时间最长的工件. 由于 FJSP 存在机器选择的问题,无法确切知道每道工序在哪台机器上加工以及加工时间是多少. 因此,采用其可用机器的加工时间求均值的方式来衡量该工件的平均加工时间,通过比较每个工件每道工序平均加工时间的和,可以找到加工时间最长的工件,优先安排其所有工序.

(4)基于最短工件驱散. 与基于最长工件的驱散方式原理相同,仅将最长工件换成最短工件.

对 4 种不同的驱散方式进行试验,试验结果见表 5. 由表 5 可知,基于工件驱散方式表现最优.

表 5 驱散试验结果

Tab. 5 Results for disperation test

驱散操作方式	10 组试验均值
随机驱散	149.5
基于工件驱散	149.1
基于最长工件驱散	149.9
基于最短工件驱散	149.4

3.2 趋向性操作改进

BFO 的趋向性操作包括群体自身寻优和基于最优个体的群体寻优两个过程,两个过程分别体现了细菌的趋化特性和群聚特性.

3.2.1 最优个体自身寻优

个体表现优秀是因为其对各个工件的排列顺序更加合理,而每个工件的第一道工序又起到决定性作用. 因此,最优个体自身寻优方式为保留不同工件的第一道工序的相对位置信息,其他工序随机重排.

3.2.2 个体自身寻优

①交换变异:随机产生两个位置,交换位置信息. ②移码变异:随机产生一个位置和一个距离 s,将该位置信息向右移动距离 s;如果超过最后一个位置则转到第一个位置继续. ③反转变异:随机产生两个位置,将两个位置之间的信息逆序排列.

④插入变异:随机产生两个位置,将第一个位置的信息插入到第二个位置之后. ⑤位移变异:随机产生 3 个位置,将前两个位置之间的信息插入到第 3 个位置之后.

3.2.3　基于最优个体寻优

(1)次序交叉:随机产生两个交叉位置,将最优个体交叉点之间的信息片段复制给普通个体 a,并剔除普通个体的交叉点之外与最优个体交叉点之间冲突的信息,得到 b;复制普通个体交叉点之间的位置信息,得到 c;将 c 中的位置信息从第二交叉位置开始依次填入 b 中的空缺位置得到最终个体 e.

(2)基于位置的交叉:随机产生多个位置点,将最优个体位置点的信息复制给普通个体 a,得到 b;依次剔除普通个体 a 中与最优个体位置点的信息冲突的部分,得到 c;将 c 中的位置信息依次填入 b 个体中的空缺位置得到最终个体 e.

(3)基于顺序的交叉:随机产生一个二进制表,将最优个体与二进制表中为 1 的对应位置的位置信息复制给普通个体 a,得到 b;依次剔除普通个体 a 中与最优个体位置点的信息冲突的部分,得到 c;将 c 中的位置信息依次填入 b 中的空缺位置得到最终个体 e.

(4)线性次序交叉:随机产生两个交叉位置,将最优个体交叉点之间的信息片段复制给普通个体 a,得到 b;依次剔除普通个体 a 中与最优个体信息片段冲突的位置部分,得到 c;将 c 中的位置信息依次填入 b 中的空缺位置得到最终个体 e.

(5)局部调度交换交叉:在最优个体和普通个体 a 中分别随机产生两个交叉位置(交叉位置间长度可不同),将最优个体交叉点之间的信息片段填充到普通个体交叉点之间,保留普通个体交叉点外位置信息,得到 b;在交叉点外,依次剔除(或增补)b 中与交叉位置信息片段冲突(多或者少位置信息)的位置部分得到 c;整理即得到最终个体 e.

(6)优先级保存交叉:产生一个由 0 和 1 组成的随机序列,0 对应的位置取最优染色体的信息,1 对应的位置取普通个体 a 的信息,每个位置信息选取的时候保证取到信息不与之前信息冲突;最后得到最终个体为 b.

(7)优先操作交叉:随机选择一个优先操作集合,将最优个体的优先操作集合的位置信息复制给普通个体 a,得到 b;剔除普通个体中该优先操作集合,将剩余位置信息依次填入 b 空缺位置,得到最终个体 c.

3.2.4　改进设计数值实验.

(1)是否增加最优个体自身寻优试验. 对增加最优个体寻优过程的 BFO 进行 10 组试验,平均值为 148.9,没有增加该过程的试验值为 149.5. 经过比较可见,增加最优个体自身寻优过程,有利于增加算法的寻优能力.

(2)个体自身寻优试验. 对细菌觅食算法的个体自身寻优过程进行 10 组试验,由表 6 中第 2 列可知,个体自身寻优过程中采用反转变异,有利于增加算法的寻优能力.

表 6　10 组群体自身寻优试验试验均值

Tab.6　Results for 10 tests of the population optimization

寻优方式	个体自身寻优均值	最优个体自身寻优均值
原变异方式	150.0	149.8
交换变异	147.7	148.0
移码变异	148.2	148.6
反转变异	146.9	148.3
插入变异	149.0	148.9
位移变异	148.0	148.8

(3)最优个体自身寻优试验. 对 BFO 的最优个体自身寻优过程进行 10 组试验,由表 6 中第 3 列可知,最优个体自身寻优过程中采用交换变异有利于增加算法的寻优能力.

(4)基于最优个体的群体寻优试验. 对 BFO 基于最优个体的群体寻优试验过程进行 10 组试验,由表 7 可知,基于最优个体的群体寻优过程中,PBX、LOX 及 POX 有利于增加算法寻优能力.

表 7　基于最优个体的群体寻优试验

Tab.7　Results for population optimization based on the best chromosome

寻优方式	10 组试验均值
原交叉方式	148.9
次序交叉(OX)	149.4
基于位置的交叉(PBX)	148.5
基于顺序的交叉(OBX)	149.0
线性次序交叉(LOX)	148.5
局部调度交换交叉(PSXX)	149.8
优先级保存交叉(PPX)	150.0
优先操作交叉(POX)	148.6

综合考虑趋向性操作改进的试验结果,可以得出结论:增加最优个体的寻优过程,有利于算法寻优;在寻优初期,适宜进行反转变异;待群体染色体达到一定的优秀程度,适宜进行交换变异;对于交叉操作,适宜进行 PBX、LOX 和 POX.

3.3 算法最佳配置及参数

根据以上试验数据和BFO的参数配置,确定IBFO的配置:①基于工件驱散方式,②增加最优个体自身寻优,③普通个体自身寻优方式为反转变异,④最优个体自身寻优方式为交换变异,⑤基于最优个体的寻优方式为基于位置的交叉,⑥种群规模为50,趋向性操作次数 Nc 为50,复制操作次数 Nre 为5,驱散(迁徙)操作次数 Ned 为20,驱散(迁徙)概率 Ped 为0.7.

3.4 数值实验

为了对比IBFO与其他进化算法的性能,应用IBFO对Kacem[13]算例和Brandimarte[3]系列进行实验,结果见表8和9,可以看出IBFO以90%的成功率找到最优解,且平均离差率最低,证明IBFO算法的优越性能.因此,6种算法排名为BC < BEA < BSA < BCC < BFO < IBFO.

表8 Kacem算例计算结果

Tab.8 Results for Kacem Benchmarks

算例	(工件,机器)	IBFO	成功率/%
Kacem 1	(4,5)	11	
Kacem 2	(8,8)	14	
Kacem 3	(10,7)	11	100
Kacem 4	(10,10)	7	
Kacem 5	(15,10)	11	

表9 Brandimarte算例计算结果

Tab.9 Results for Brandimarte Benchmarks

算例	(工件,机器)	(LB,UB)	IBFO	成功率/%	平均离差率/%
MK01	(10,6)	(36,42)	40		
MK02	(10,6)	(24,32)	26		
MK03	(15,8)	(204,211)	204		
MK04	(15,8)	(48,81)	60		
MK05	(15,4)	(168,186)	172	90	0.76
MK06	(10,15)	(33,86)	58		
MK07	(20,5)	(133,157)	139		
MK08	(20,10)	(523,523)	523		
MK09	(20,10)	(299,369)	307		
MK10	(20,15)	(165,296)	212		

4 结论

针对FJSP,建立了优化模型,提出了BC、BEA、BSA、BCC和BFO,并进行了数值实验和分析.结果表明,5种算法稳定性、改进空间和寻优能力存在一定差异.其中,BFO表现最为优异,因此对其进行了改进,针对其关键操作设计了多种优化算子,最终得到优化能力最强的算法结构和算子组合.数值实验表明,改进的细菌觅食算法的寻优能力及稳定性大幅提升,体现出非常好的全局开发能力和局部搜索能力,能高效求解FJSP.

参考文献:

[1] BRUCKER P, SCHLIE R. Job-shop scheduling with multi-purpose machines[J]. Computing, 1990, 45(4): 369-375.

[2] 王书锋,肖小城,冯冬青.求解Job-shop问题的改进混合离散粒子群优化算法[J].郑州大学学报(工学版), 2010,31(4):44-47.

[3] BRANDIMARTE P. Routing and scheduling in a flexible job shop by tabu search[J]. Annals of operations research, 1993, 41(3): 157-183.

[4] BREMERMANN H. Chemotaxis and optimization[J]. Journal of the franklin institute, 1974, 297(5): 397-404.

[5] MÜLLER S, AIRAGHI S, MARCHETTO J, et al. Optimization algorithms based on a model of bacterial chemotaxis[C]// Proceedings of the 6th International Conference Simulation of Adaptive Behavior: From Animals to Animats. Paris, France: Springer, 2000: 375-384.

[6] MÜLLER S D, MARCHETTO J, AIRAGHI S, et al. Optimization based on bacterial chemotaxis[J]. IEEE transactions on evolutionary computation, 2002, 6(1): 16-29.

[7] LI W, WANG H, ZOU Z, et al. Function optimization method based on bacterial colony chemotaxis[J]. Journal of circuits and systems, 2005, 10(1): 58-63.

[8] PASSINO K M. Biomimicry of bacterial foraging for distributed optimization and control[J]. Control systems IEEE, 2002, 22(3): 52-67.

[9] TANG W J, WU Q H, SAUNDERS J R. A bacterial swarming algorithm for global optimization[C]// 2007 IEEE Congress on Evolutionary Computation (CEC 2007). Singapore: IEEE, 2007: 1207-1212.

[10] CHU Y, MI H, LIAO H, et al. A fast bacterial swarming algorithm for high-dimensional function optimization[C]// 2008 IEEE Congress on Evolutionary Computation (CEC 2008). Hong Kong, China: IEEE, 2008: 3135-3140.

[11] NAWA N E, FURUHASHI T. Fuzzy system parameters discovery by bacterial evolutionary algorithm [J]. IEEE transactions on fuzzy systems, 1999, 7(5): 608-616.

[12] 吴秀丽,孙树栋,余建军,等.多目标柔性作业车间调度优化研究[J].计算机集成制造系统, 2006, 12(5): 731-736.

[13] KACEM I, HAMMADI S, BORNE P. Approach by localization and multi-objective evolutionary optimization for flexible job-shop scheduling problems [J]. IEEE transactions on systems, man and cybernetics

part C:application & reviews, 2002, 32:408 – 19.
[14] LI J Q, PAN Q K, GAO K Z. Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems[J]. International journal of advanced manufacturing technology, 2012, 55:1159 – 1169.

The Comparison and Improvement of Bacterial Algorithms for Flexible Job Scheduling Problem

WU Xiuli, ZHANG Zhiqiang

(School of Mechanic Engineering, University of Science and Technology Beijing, Beijing 100083, China)

Abstract: The article aimed to fully explore the ability of bacterial algorithin and its varieties for solving the discrete optimization problems. The bacterial chemotaxis algorithm(BC), bacterial colony chemotaxis algorithm(BCC), bacterial evolutionary algorithm(BEA), bacterial swarming algorithm(BSA) and bacterial foraging optimization algorithm(BFO) are designed to solve the flexible job scheduling problem. Firstly, the model of the flexible job scheduling problem was formulated. Then the five algorithms were designed to solve the benchmark was instances. The results showed that the BFO outperformed the others. Furthermove, a strategy to improve the BFO was proposed. More than ten optimization operators were designed and compared. Finally, the best structure of the improved BFO was built. The numerical experiments showed that the proposed BFO balanced the exploration and the exploitation very well and could solve FJSP effectively.

Key words: flexible job scheduling problem; bacterial chemotaxis algorithm; bacterial colony chemotaxis algorithm; bacterial evolutionary algorithm; bacterial swarming algorithm and bacterial foraging optimization algorithm

---

(上接第 33 页)

Detection of Human Behavior Anomaly Based on the Optical Flow Co-occurrence Matrix

ZENG Qingshan, SONG Qingxiang, FAN Mingli

(School of Electrical Engineering, Zhengzhou University, Zhengzhou 450001, China)

Abstract: The traditional anomaly detection algorithm for human beharior that based on image texture features tended to describe the change of human image texture rather than the actual situation of human motion behavior. Its detection porformance was not so good. In this paper, a method of feature extraction was proposed to reflect the real situation of human motion behavior. Firstly, the optical flow information was extracted by Lucas-Kanade optical flow algorithm, and the co-occurrence matrix and optical flow direction co-occurrence matrix were established. Then, the characteristics of two order distance, contrast, entropy and similarity are extracted by the co-occurrence matrix, and then combined them with the mean value of optical flow to form a feature vector to train the support vector machine (SVM). Finally, this algorithm was used to determine whether the crowd had abnormal behavior. The simulation results showed that the feature extraction method in this paper had more in depth processing of the crowd motion information provided by the optical flow method. Compared with the mainstream algorithm, it has a better recognition performance.

Key words: crowd behavior anomaly detection; optical flow; optical flow co-occurrence matrix; support vector machine