基于精英多策略的货位分配优化方法

张贵军 姚 俊 周晓根 王 文
(浙江工业大学信息工程学院 杭州 310023)

摘 要 针对智能立体仓库货位分配问题,提出一种基于精英多策略的货位分配优化方法。首先,考虑货物重量、出入库频率和出入库时间等因素,以货架重心低、出入库频率高、货物离出入库口近等为原则建立货位分配优化模型;然后,提出一种精英多策略差分进化算法,通过提取部分精英个体的信息指导变异,并根据精英个体的拥挤度变化对不同的阶段使用不同的策略,从而产生高质量的解,同时加快算法的收敛速度;最后,通过10个经典测试函数验证了所提算法的有效性,并基于该方法对某智能制造企业的成品库进行了优化,得到了满意的货位分配方案。

关键词 智能立体仓库,差分进化,货位分配,精英个体
中图法分类号 TP391.9 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.01.048

Storage Location Assignment Optimization Method Based on Elite Multi-strategy

ZHANG Gui-jun YAO Jun ZHOU Xiao-gen WANG Wen
(College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract To address the problem of storage location assignment in the intelligent stereoscopic warehouse, a storage location assignment optimization method using elite multi-strategy was proposed. Firstly, by considering the factors of the weight, the frequency and time of import and export of goods, the storage location assignment optimization model was constructed based on the principle of low gravity center of goods shelf, high frequency of import and export and close distance between goods and import and export. Then, an elite multi-strategy-based differential evolution algorithm was designed to solve the constructed model. In this approach, the information of some elite individuals is extracted to guide the mutation and the mutation strategies for different search stages are selected according to the variation of the crowding degree of the elite individuals. Thus, the individuals with high quality are generated and the convergence speed is improved. Finally, the performance of the proposed algorithm was verified over ten classical benchmark functions, and the optimum storage location assignment scheme of the finished product warehouse of a company was obtained by the proposed method.

Keywords Intelligent stereoscopic warehouse, Differential evolution, Location allocation, Elite individual

1 引言

当今新型技术的高速发展以及新产品生命周期的不断缩短,导致传统企业正在逐渐被现代化、智能化的企业所替代,其中"智能化"已经成为创新的代名词。为了推动当代科技、产业和工业技术的快速发展,德国推出了工业4.0的概念,其更是被行业界称为第四次工业革命[1],其中智能工厂就是工业4.0研究的一个核心主题。然而,智能立体仓库作为智能工厂的组成部分,不仅使得货物在仓库内高效按需自动存取,而且使得可以与智能工厂中的生产环节进行有机连接,再通过计算机管理系统和搬运设备使得仓库成为生产物流中的一个重要环节。

立体仓库普遍采用扫描技术和射频数据通信技术,不仅可以提高信息的传输速度和准确性,而且使得数据的采集、处理和交换能够在搬运工具中与中央计算机之间快速进行,使物品的存取和信息的发送做到快速、实时、可靠和准确[2-3]。国内的智能立体仓库系统起步晚、发展缓慢,虽然近几年我国各相关高校、科研院所以及专业软件公司对自动化立体仓库以及仓储管理系统的研究取得了一定的进展,但是国内的智能立体仓库在运行效率、智能化和自动化方面与其他国家相比还存在着一定的差距。为了改善仓库管理,提高仓库作业效率,我国也正在大力研究和发展相关技术产业。

由于立体仓库采用大型仓储货架的拼装,同时自动化管理技术使得货物便于查找,因此智能立体仓库有占地面积小、空间利用率大和存取效率高等优点[2]。在发达国家,提高空间利用率已经成为系统合理性和先进性的重要考核指标,智能立体仓库可以形成先进的生产链,促进生产力的进步;智能立体仓库由于存取效率高,因此可以有效地连接仓库外的生产环节,在存储中形成自动化的物流系统,从而形成有计划、有编排的生产链,进一步促进生产力的发展。

到稿日期:2016-11-21 返修日期:2017-04-24 本文受国家自然科学基金(61773346,61573317),浙江省重中之重学科开放基金(20151008,20151015),浙江省大学生"新苗计划"(2016R403083)资助。

张贵军(1974—),男,博士,CCF会员,主要研究方向为智能信息处理、智能交通系统、优化理论及算法设计、生物信息学,E-mail:zgj@zjut.edu.cn(通信作者);姚 俊(1989—),男,硕士生,主要研究方向为智能信息处理;周晓根(1987—),男,博士生,CCF学生会员,主要研究方向为智能信息处理、优化理论及算法设计;王 文(1994—),男,硕士生,主要研究方向为智能信息处理。

针对智能立体的货位分配优化问题，国内外学者进行了大量研究。Muppani 和 Adil[4] 建立了货位分配优化的非线性整数规模模型，并利用分支定界法对模型进行求解，从而节省了仓储空间；Cui[5] 提出了一种基于改进粒子群优化算法的货位分配方法，通过置换概念来计算粒子的速度，并通过小生境技术来提高较差解的多样性；Li 等[6] 提出了一种基于改进遗传算法的货位分配优化算法，该算法中除了交叉和变异算子，还包括 Pareto 和小生境操作，从而提高了算法的性能；杨玮等[7] 提出了一种混合粒子群优化算法，通过建立区域划分、货位分配两阶段多目标货位分配决策模型来实现货位分配优化；张仰森等[8] 提出了一种货位分配综合优化算法，通过综合考虑重量均分步、货位控制时间和就近货位选择等因素来选择入库货位。

本文根据智能立体仓储系统的结构和特点，综合考虑货物重量、出入库频率和出入库时间等因素，建立货位分配优化数学模型；然后针对建立的模型设计了一种精英差分进化算法，通过利用部分局部精英个体的信息设计新的变异策略，并根据局部精英个体的拥挤度信息来自适应地选择变异策略，从而提高搜索效率和解的可靠性效果；最后利用所提算法对建立的模型进行优化，在给出最优结果的同时，提供两种备选方案，并将设计算法应用于实际系统中。

2 货位分配优化模型

智能立体仓库主要包括三大系统：1）存储系统，主要由分层立体货架和托盘组成；2）输送系统，主要由巷道堆垛机和 AGV 小车构成；3）控制系统，主要由自动控制系统和中央计算机管理系统组成。自动控制系统是驱动自动化立体库系统各设备的自动控制系统，主要采用现场总线方式作为控制模式[9]，如图 1 所示。

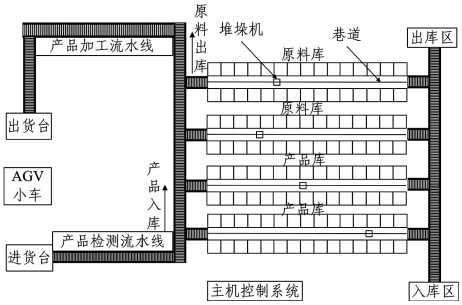

图 1 智能工厂立体仓储系统示意图

Fig. 1 Schematic diagram of the intelligent factory warehouse system

存储系统决定了整个智能立体仓库的运行效率。一个成功的货位分配策略可以缩短仓库堆垛机移动的距离，不仅可以节省能源、减少作业的时间，而且可以减小存储空间并减缓存储系统的磨损速度[9]。智能立体仓库中常见的货位分配策略有分类、随机、定位、共享和分类随机等存储方法[10]，本文以巷道式智能立体仓库为对象，利用随机存储策略进行研究。分配策略只是货物存储区规划的原则，必须结合货位分配优化原则才能决定智能立体仓库存储系统的运作模式。随着智能仓库的发展，货位分配的原则越来越多，本文主要考虑货物相关性原则、就近原则、稳定性原则和弹性原则[11]。

假设一个有 X 层 Y 列的智能立体仓库货架，设其货架的高度为 H_{max}，宽度为 L_{max}，如图 2 所示，每个货位的高度和宽度分别为 H 和 L，其中，将第 i 层第 j 列的货位记为 (i,j)，$i=1,2,3,\cdots,X$；$j=1,2,3,\cdots,Y$。设置离地面最近的一层为第 1 层，离出货台最近的一列为第 1 列。为了提高效率，采取就近原则，则出入库频率高的物料应该离出货台最近。设第 i 层第 j 列货位货物的出入频率为 P_{ij}，第 i 层第 j 列货位货物的出库距离为 S_{ij}，以 P_{ij} 和 S_{ij} 乘积之和 J_1 最小为目标，则得到第一个目标函数：

$$J_1=\sum_{i=1}^{X}\sum_{j=1}^{Y}P_{ij}\times S_{ij}=\sum_{i=1}^{X}\sum_{j=1}^{Y}P_{ij}\times(i\times H+j\times L)\times J_1$$
$$=\sum_{i=1}^{X}\sum_{j=1}^{Y}P_{ij}\times S_{ij}=\sum_{i=1}^{X}\sum_{j=1}^{Y}P_{ij}\times(i\times H+j\times L) \quad (1)$$

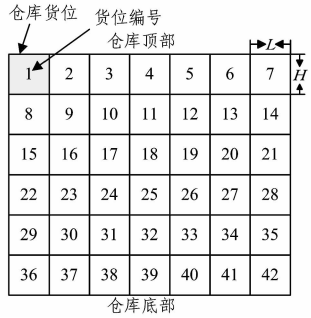

图 2 货位分配示意图

Fig. 2 Schematic diagram of storage location assignment

为了保持智能立体仓库的系统稳定性，按照上轻下重、尽量降低重心的原则，以第 i 层第 j 列货位货物的重量 G_{ij} 与其所在层数 i 的乘积之和最小为目标，即以重心最小为目标，可得到第二个目标函数 J_2：

$$J_2=\sum_{i=1}^{X}\sum_{j=1}^{Y}(G_{ij}\times i) \quad (2)$$

以上给出了两个目标函数。为了方便求解，本文通过设置式(1)和式(2)的目标函数的权重，将多目标问题转化成单目标问题。假设目标函数 J_1 和目标函数 J_2 的权重分别为 ω_1 和 ω_2，从而得到一个单一目标函数 J：

$$J=\omega_1\times J_1+\omega_2\times J_2$$
$$=\omega_1\times\sum_{i=1}^{X}\sum_{j=1}^{Y}P_{ij}\times(i\times H+j\times L)+\omega_2\times\sum_{i=1}^{X}\sum_{j=1}^{Y}(G_{ij}\times i) \quad (3)$$

3 精英差分进化模型求解方法

差分进化算法(DE)[12] 作为一种简单、有效的群体进化算法，被广泛用于各种实际问题的求解过程中。与其他进化算法一样，DE 算法也是一种基于自然选择和遗传等生物进化机制的随机优化算法，通过种群中各个体间的竞争与合作产生的群体智能来引导算法搜索。差分进化算法结构简单，容易实现，不仅鲁棒性强，而且收敛速度快[13]，因此常被用于各类问题的求解过程中[13-16]，如蛋白质结构预测、传感网络定位和流水作业调度等。

DE 算法[15] 首先在搜索空间中随机生成一个包含多个个体的种群，然后通过一种变异策略将每一个父代个体和随机选择的种群个体进行组合而生成测试个体，最后根据个体适应度值从父代个体和测试个体中选择较优的个体来更新种群。在上述操作中，变异策略至关重要，其决定了产生的后代解的质量，同时又可以维持种群的多样性，防止出现早熟收敛现象[17-18]。近年来，为了提高 DE 算法的性能，国内外学者相继设计出了各种变异策略，然而，由于不同的问题具有不同的数学特性，一种变异策略不可能适用于所有的问题[16]。因此，如何从众多的变异策略中选择一个适用于特定问题的策略是一项具有挑战性的工作。本文提出一种精英多策略差分

进化算法(EDE)，通过提取局部精英个体的信息指导算法变异，同时利用局部精英个体的拥挤度信息来估计当前种群的进化状态，从而在各阶段选择合适的变异策略，在加快算法收敛速度的同时，防止算法陷入局部最优而出现早熟现象，从而提高算法的可靠性和搜索效率。

3.1　精英个体中心变异策略

上述货位分配优化模型曲面复杂，如果利用基本 DE 算法进行搜索，不仅可能使其陷入局部最优而无法找到全局最优解，而且搜索效率较低。为了提高解的可靠性，同时加快算法的收敛速度，本文基于局部精英个体信息设计了一种精英个体中心变异策略，通过提取部分精英个体的信息来指导算法搜索。首先根据各个体的适应度信息对当前种群进行升序排列，然后选出前 N 个计算中心个体，最后根据中心个体指导各个体进行变异。假设在第 g 代种群中所选的 N 个精英个体为 $x_{elite,t}$，$t=1,2,\cdots,N$，则中心个体 x_{center} 可以由如下公式计算得到：

$$x_{center}^{j}=\frac{\sum_{t=1}^{N}x_{elite,t}^{j}}{N} \tag{4}$$

其中，$j=1,2,\cdots,D$，x_{center}^{j} 表示中心个体的第 j 维。利用中心个体替换“DE/current-to-best/1”策略[15]中的 best 个体，即可得到精英个体中心变异策略“DE/current-to-center/1”：

$$v_i=x_i+F(x_{center}-x_i)+F(x_a-x_b) \tag{5}$$

其中，F 为增益常数，v_i 表示变异个体，x_i 表示第 i 个目标个体，x_a 和 x_b 为从当前种群中随机选择的个体，且 $a\neq b\neq i$。

传统的“DE/current-to-best/1”变异策略利用当前种群的最优个体信息来引导种群搜索，虽然收敛速度快，但是容易导致算法陷入局部最优，从而无法求得问题的全局最优解；而在上述设计的精英个体中心变异策略中，根据适应度信息选取多个精英个体，从而根据这些精英个体的计算得到中心个体，并用其来代替全局最优个体引导算法搜索，既能防止算法早熟收敛，又能加快算法的收敛速度，进而提高算法的性能。

3.2　策略自适应机制

本文根据部分精英个体之间的距离信息来衡量所选精英个体的拥挤度，从而估计种群进化所达到的状态，进而根据估计状态达到自适应选择变异策略的效果。

首先，根据当前个体的适应度信息选取 N 个精英个体，并计算各精英个体之间的平均距离 d_g：

$$d_g=\frac{\sum_{t=1}^{N}\sum_{s=t+1}^{N}\sqrt{\sum_{j=1}^{D}(x_{center,t}^{j}-x_{center,s}^{j})^2}}{N} \tag{6}$$

其中，d_g 表示第 g 代种群中各精英个体之间的平均距离，本文选用欧氏距离。

然后，在每一代算法开始前，根据式(7)计算当前种群中 N 个精英个体之间的平均距离，并根据当前平均距离与上一代种群精英个体之间的距离之比来衡量种群拥挤变化因子 δ，即：

$$\delta=\frac{d_g}{d_{g-1}},\ g>0 \tag{7}$$

其中，初始种群拥挤变化因子为 1。进而根据种群拥挤变化因子判断当前种群的状态，即：

$$\psi=\begin{cases}S_1, & \text{if } rand(0,1)<\delta\\ S_2, & \text{otherwise}\end{cases} \tag{8}$$

其中，$rand(0,1)$表示 0 到 1 之间的随机小数，S_1 表示当前种群处于全局探测状态，S_2 表示当前种群处于局部搜索状态。当某一代的精英个体的平均距离与上一代精英个体的平均距离之比较大时，说明当前种群中的精英个体发生了较大变化，可判定当前种群处于全局探测状态；当某一代的精英个体的平均距离与上一代精英个体的平均距离之比较小时，说明当前种群中的精英个体变化较小，可判定当前个体处于局部搜索状态。

最后，根据上述估计得到的搜索状态自适应地选择变异策略。假设 M_i 为目标个体 x_i 的变异策略，则：

$$M_i=\begin{cases}\text{DE/rand/1}, & \text{if } \psi=S_1\\ \text{DE/current-to-center/1}, & \text{if } \psi=S_2\end{cases} \tag{9}$$

其中，“DE/rand/1”的具体表达式参见文献[19]。上述公式表明，当种群处于全局探测状态 S_1 时，种群中的大部分个体使用全局探测能力较强的“DE/rand/1”进行全局搜索最优解所在的区域，同时为了提高对已搜索到的区域进行局部搜索的能力，其他小部分个体则使用本文所设计的局部搜索能力较强的“DE/current-to-center/1”策略进行搜索；当整个种群处于局部搜索状态时，大部分个体使用“DE/current-to-center/1”策略进行搜索，另外，为了防止陷入局部最优并且搜索更优的区域，对其他小部分个体使用“DE/rand/1”策略继续进行全局探测。

通过上述设计，利用种群中部分精英个体的拥挤度变化来估计种群的搜索状态，从而对不同的状态自适应地使用不同的变异策略，进而生成高质量的后代个体，从整体上提高了算法的性能。

4　数值仿真

为了验证本文所提方法的性能，首先利用测试函数验证了所提 EDE 算法的有效性，然后针对某公司的智能立体仓库进行了优化，最后对分配方案进行了系统实现。

4.1　EDE 算法验证

为了验证所提 EDE 算法的性能，选用 10 个具有代表性的测试函数进行实验。表 1 列出了这 10 个测试函数的名称、维数、最优解和搜索域，具体数学表达式参见文献[19-21]。在这些函数中，不同的函数具有不同特性的函数曲面，函数 F_1-F_5 为单模函数，函数 F_6-F_{10} 为多模函数，其局部最优解的数量随着维数的升高而指数增加。

表 1　测试函数

Table 1　Benchmark functions

函数	维数(D)	最优值	搜索域
F_1:Sphere	30	0	(−100,100)
F_2:SumSquares	30	0	(−10,10)
F_3:Schwefel 2.22	30	0	(−10,10)
F_4:Exponential	30	0	(−1,1)
F_5:Tablet	30	0	(−100,100)
F_6:Griewank	30	0	(−600,600)
F_7:Schaffer 2	30	0	(−100,100)
F_8:Schwefel 2.26	30	−418.983D	(−500,500)
F_9:Ackley	30	0	(−30,30)
F_{10}:Rastrigin	30	0	(−5.12,5.12)

实验中，选取 SaDE[22]，JADE[23]和 CoDE[24] 3 种先进的改进 DE 算法与本文所提 EDE 算法进行比较。SaDE 算法是一种策略自适应 DE 算法，算法基于前期的成功经验从策略池中自适应地选取变异策略，同时更新参数。JADE 算法为基于最优存档的 DE 算法，算法通过随机选取较优个体的信息来设计新的策略以指导算法变异。CoDE 算法是一种具有

系综参数和策略的 DE 算法，通过从策略池和参数池中随机选择并组合生成多个测试个体，选出适应值最佳的个体为后代个体。上述 3 种算法的参数设置与各原文保持一致。EDE 算法设置：种群规模 $NP=50$，交叉概率 $CR=0.5$，增益常数 $F=0.5$。为了公平比较，所有算法的终止条件均相同，且各算法对于各函数均独立运行 30 次，并对结果取平均值。

选取函数评价次数(*FEs*)和成功率(*SR*)两个指标来验证所提 EDE 算法的收敛速度和可靠性。在给定的最大函数评价次数内，如果函数误差值($F(x)-F(x^*)$)小于设定的阈值，则表示算法成功运行。记录此时的目标函数评价次数，其中$F(x)$为算法所求得的最优解，$F(x^*)$为测试函数的全局最优值(见表 1)，*SR* 为成功运行次数与总运行次数之比。此实验中，设置最大函数评价次数为 300000，函数误差阈值精度为 0.00001。

表 2 列出了 SaDE，JADE，CoDE 和 EDE 算法的函数评价次数和成功率，其中最优结果通过加粗标出。从表 2 中可以看出，所提算法 EDE 对于大部分函数的收敛速度和可靠性总是优于其他 3 种对比算法。具体来讲，在 *FEs* 方面，SaDE 算法在函数 F_7 和 F_{10} 上优于其他算法；JADE 和 CoDE 算法没有在任何函数上优于其他算法；而本文所提 EDE 算法在所有单模函数上都优于其他 3 种算法；在 5 个多模函数中，EDE 算法在 3 个函数(F_6，F_8 和 F_9)上优于其他算法。在 *SR* 方面，SaDE 算法可以对 7 个测试函数以 100%的成功率求解；JADE 和 CoDE 算法可以对 9 个函数以 100%的成功率求解；而本文算法可以对所有函数以 100%的成功率求解。另外，从表中最后一行的平均值可以看出，EDE 算法对所有函数的平均目标函数评价次数最低为 27584，SaDE，JADE 和 CoDE 算法的目标函数评价次数分别为 31776，37633 和 66036，即 EDE 算法达到设定的函数误差精度所需的目标函数评价次数相对于 SaDE，JADE 和 CoDE 算法节省了 13.2%，26.7%和 58.2%；而且 EDE 算法的成功率为 100%，SaDE，JADE 和 CoDE 算法的成功率分别为 95.9%，98.3%和 99.7%。

表 2 函数评价次数和成功率

Table 2 Function evaluations and success rate

函数	SaDE		JADE		CoDE		EDE	
	FEs	*SR*	*FEs*	*SR*	*FEs*	*SR*	*FEs*	*SR*
F_1	20297	1.00	23557	1.00	40155	1.00	**13810**	1.00
F_2	18410	1.00	21600	1.00	36270	1.00	**12190**	1.00
F_3	24368	1.00	35787	1.00	56421	1.00	**19580**	1.00
F_4	10993	1.00	13377	1.00	22362	1.00	**7800**	1.00
F_5	21117	1.00	27007	1.00	41286	1.00	**14790**	1.00
F_6	21740	0.73	17581	0.83	43878	1.00	**10700**	1.00
F_7	**71833**	1.00	113058	1.00	172776	1.00	83210	1.00
F_8	36050	1.00	37098	1.00	59982	1.00	**21327**	1.00
F_9	30453	0.93	23242	1.00	56826	1.00	**19320**	1.00
F_{10}	**62508**	0.93	64025	1.00	130409	0.97	73121	1.00

平均值	31776	0.959	37633	0.983	66036	0.997	27584	1.00

图 3 给出了 4 个具有代表性的函数(2 个单模函数和 2 个多模函数)的平均收敛曲线。可以看出,所提 EDE 算法的收敛速度明显优于其他 3 种算法,对于函数 F_1 和函数 F_2,SaDE 算法仅次于 EDE 算法;而对于函数 F_6 和函数 F_7,JADE 算法仅次于 EDE 算法。

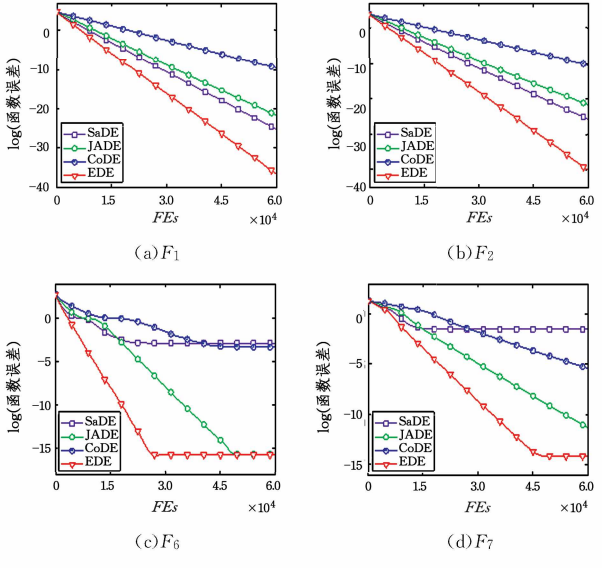

图 3 平均收敛曲线

Fig. 3 Comparison of average convergence

为了验证算法所求得的解的质量,选用平均函数误差(Mean Error)和标准偏差来衡量给定的目标函数评价次数内的解。另外,为了验证所提算法与其他对比算法之间是否有显著性优势,采用 Wilcoxon test[20,25] 对各算法运行 30 次所求得的结果进行非参数检验,设置显著性水平为 0.05,并用"+"表示所提算法明显优于对比算法,用"−"表示对比算法明显优于所提算法,用"≈"表示所提算法与对比算法之间没有显著性差异。

表 3 列出了各算法对各测试函数所求得的解的平均值和标准偏差。从表 3 中的结果可以看出,EDE 算法对大部分函数的表现明显优于其他 3 种对比算法。具体来说,EDE 算法显著优于 SaDE 算法 4 个单模函数(F_1 − F_3 和 F_5)和 2 个多模函数(F_6 和 F_9),并且对于函数 F_4 和 F_8 获得了与 EDE 相同的结果,而 SaDE 算法仅显著优于 EDE 算法 2 个函数(F_7 和 F_{10});与 JADE 算法相比,EDE 算法在 8 个函数上表现出了明显优势,并且对于 2 个函数(F_4 和 F_6),两种算法没有显著性差异,JADE 算法没有显著优于 EDE 算法的任何函数;EDE 算法明显优于 CoDE 算法 9 个测试函数,并且对于另一个函数其获得了与 CoDE 算法相似的结果,而 EDE 算法没有显著优于 CoDE 算法的任何函数。

表 3　各算法最优结果的平均值和标准偏差

Table 3　Mean value and standard deviation of the optimal results of the given algorithms

函数	SaDE	JADE	CoDE	EDE
	Mean Error(Std Dev)	Mean Error (Std Dev)	Mean Error (Std Dev)	Mean Error (Std Dev)
F_1	1.29E−23(3.07E−23)＋	3.01E−20(8.74E−20)＋	2.16E−10(1.73E−10)＋	**2.47E−37(3.16E−37)**
F_2	6.59E−25(8.39E−25)＋	1.16E−21(1.59E−21)＋	2.83E−11(2.61E−11)＋	**7.46E−41(9.24E−41)**
F_3	8.70E−16(5.17E−16)＋	3.11E−10(3.36E−10)＋	3.86E−06(1.32E−06)＋	**8.97E−20(1.76E−19)**
F_4	**0.00E+00(0.00E+00)**≈	**0.00E+00(0.00E+00)**≈	1.30E−14(1.13E−14)＋	**0.00E+00(0.00E+00)**
F_5	8.85E−23(4.35E−22)＋	1.83E−18(6.07E−18)＋	4.04E−10(3.04E−10)＋	**6.88E−37(1.23E−36)**
F_6	2.22E−03(4.73E−03)＋	9.70E−15(5.25E−14)≈	2.47E−07(7.34E−07)＋	**1.78E−16(9.93E−17)**
F_7	**7.51E−04(7.06E−04)**−	3.24E+00(1.23E+00)＋	1.97E+00(4.24E−01)＋	9.93E−02(5.43E−02)
F_8	**0.00E+00(0.00E+00)**≈	1.07E+01(2.20E+01)＋	1.66E−02(3.13E−02)＋	**0.00E+00(0.00E+00)**
F_9	2.40E−01(4.45E−01)＋	4.19E−11(4.49E−11)＋	3.75E−06(1.88E−06)＋	**7.55E−15(0.00E+00)**
F_{10}	**4.11E−02(1.82E−01)**−	4.76E+00(8.78E−01)＋	3.30E+01(5.81E+00)≈	2.38E+00(9.22E−01)

上述结果和分析表明，所提 EDE 算法针对大部分测试函数的性能优于对比算法，不仅收敛速度快、可靠性高，而且解的质量较高。

4.2　案例分析

遵循 J2EE 规范，以 SSH 架构为后端开发框架，并以 AngularJS 为前端开发框架，设计并开发了柔性制造系统。系统包含了本文所研究的智能立体仓库模块，该模块通过 PLC 技术、OPC 服务器、Socket 通信和 WebSocket 通信等技术实现了对智能立体仓库的实时监控以及相应命令的下达。

针对某智能制造企业的智能立体仓库，建立式(3)所表达的模型，实际设备如图 4 所示，利用本文设计的 EDE 优化算法进行计算。该智能立体仓库分为原料库和成品库，两个仓库均是 5 层 7 列的货架，每个货架有 35 个货位，一个货位至多存放一个货物。

图 4　智能立体仓库实物图

Fig. 4　The entity of intelligent tiered warehouse facility

不同的应用场景、不同的时间段和不同的地点对于各货位分配原则的侧重点都有所不同。层次分析法[26-28]能够对复杂、较为模糊的问题做出决策方案。综合考虑各货位分配原则对本案例的影响程度及数值实验，选取权重 $\omega_1=0.7790$，$\omega_2=0.2210$。

假设成品库的初始货位如图 5 所示，图中的每个方块表示一个货位，方块上面标记的数字表示货位分配优化之前所存放物品的重量，数字为 0 表示该货位不存放货物。图中每个货位的高度 H 均为 2，宽度 L 均为 3，货架行数 X 为 5，货架列数 Y 为 7，因此货架的货位数为 $5\times7=35$ 个，待入库的货物的数量也为 35 个，各货位的货物出货频率和相应的初始位置如表 4 所列。

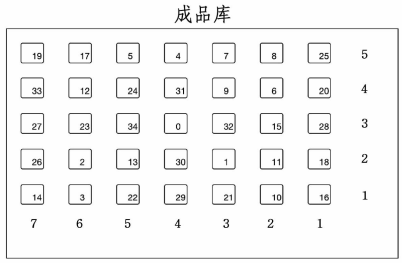

图 5　初始货物分布图

Fig. 5　Initial distribution of goods

表 4　优化前的货物情况表

Table 4　List of goods before being optimized

位置(i,j)	频率 P_{ij}	重量 G_{ij}	位置(i,j)	频率 P_{ij}	重量 G_{ij}
(1,1)	0.01	54	(3,5)	0.03	65
(1,2)	0.02	76	(3,6)	0.03	0
(1,3)	0.04	88	(3,7)	0.01	21
(1,4)	0.02	87	(4,1)	0.06	83
(1,5)	0.03	16	(4,2)	0.04	90
(1,6)	0.02	15	(4,3)	0.03	64
(1,7)	0.06	24	(4,4)	0.03	21
(2,1)	0.02	11	(4,5)	0.05	0
(2,2)	0.01	0	(4,6)	0.06	55
(2,3)	0.03	28	(4,7)	0.04	43
(2,4)	0.04	87	(5,1)	0.01	21
(2,5)	0.02	61	(5,2)	0.02	87
(2,6)	0.01	17	(5,3)	0.01	63
(2,7)	0.04	76	(5,4)	0.03	45
(3,1)	0.03	90	(5,5)	0.01	38
(3,2)	0.04	54	(5,6)	0.02	23
(3,3)	0.03	93	(5,7)	0.03	30
(3,4)	0.02	0			

利用本文所提 EDE 算法对上述成品库的货位进行优化，算法的终止条件为函数评价次数 $FEs=100000$，种群规模为 50，增益常数 F 为 0.5，交叉概率 CR 为 0.5，算法独立运行 30 次所得到的最优结果、次优结果 1 和次优结果 2 分别如图 7—图 9 所示。其中，次优结果 1 和次优结果 2 为两个备选方案，最优结果的值为 1253.57，次优结果 1 的值为 1256.42，次优结果 2 的值为 1259.85。本文只对成品库进行了优化，原料库的货位优化与成品库相同。

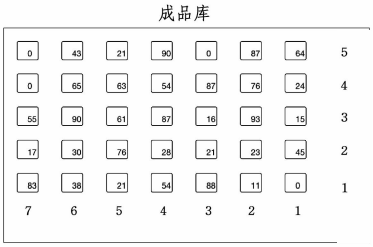

图 6 最优结果

Fig. 6 The optimal result

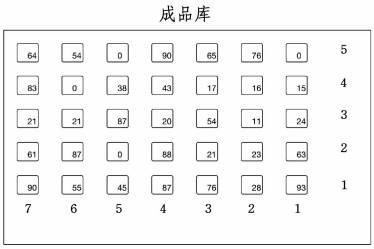

图 7 次优结果 1

Fig. 7 The sub-optimal result 1

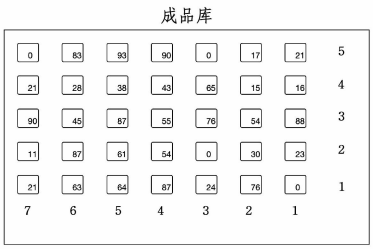

图 8 次优结果 2

Fig. 8 The sub-optimal result 2

为了验证所提 EDE 算法的优势，分别利用 SaDE[22]，JADE[23]，CoDE[24] 和 GL-25[29] 算法对上述成品库的货位进行优化，再进行比较分析。所有算法的终止条件均为函数评价次数达到 100000，各算法均独立运行 30 次，SaDE，JADE，CoDE 和 GL-25 算法的参数设置均与原文相同。

表 5 列出了各算法对上述仓位的优化结果。从表中数据可以看出，EDE 算法在最大值、最小值以及平均值方面均优于其他 3 种算法，SaDE 算法仅次于 EDE 算法，而且 EDE 算法的标准偏差最小，说明 EDE 算法最稳定；表中最后一列给出了 Wilcoxon test 的显著性结果，可以看出，所提 EDE 算法显著优于其他 4 种算法。另外，各算法的平均收敛曲线如图 9 所示，可以看出，所提 EDE 算法的收敛速度显著优于其他 4 种对比算法，SaDE 算法的收敛速度仅次于 EDE 算法，GL-25 算法的收敛速度相对较慢。

表 5 各种算法的优化结果比较

Table 5 Results comparison of optimization algorithms

算法	最大值	最小值	平均值	标准偏差	Sig
SaDE	1283.42	1259.43	1271.82	11.68	+
JADE	1284.32	1263.26	1273.27	8.73	+
CoDE	1321.59	1281.53	1296.14	17.86	+
GL-25	1343.65	1279.41	1305.92	17.97	+
EDE	1259.85	1251.97	1255.45	3.46	

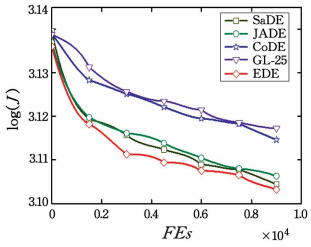

图 9 平均收敛曲线图

Fig. 9 Average convergence curves

总体来说，EDE 算法不仅搜索效率高、收敛速度快，而且较稳定，解的质量较高。

结束语 本文以巷道式智能立体仓库为研究对象，对立体仓库的货位分配方案进行了研究。首先，通过分析货物重量对货架稳定性的影响，以及货物的出入库频率和到达货位所需的时间对出入库作业效率的影响，建立了货位分配优化模型；然后，针对所建立的模型，提出了一种精英多策略差分进化算法，并通过利用种群中精英个体的信息来指导优化，从而快速地获得高质量的解；最后通过 10 个经典的测试函数验证了所提算法的有效性，同时针对某公司的某一成品仓库，利用所提方法进行优化求解，给出了最优方案和两种备选方案，并将所得方案在设计的智能仓库系统中进行了实现。实验结果表明，所提货位分配优化方法在确保立体仓库稳定性的前提下，能达到智能立体仓储系统存取货物效率的最大化。

参 考 文 献

[1] DU P S. Intelligent Plant —— Germany's first step in advancing the 4 strategy of industry(upper)[J]. Automation Panorama, 2014, 1(1): 22-25. (in Chinese)
杜品圣. 智能工厂——德国推进工业 4.0 战略的第一步(上)[J]. 自动化博览, 2014, 1(1): 22-25.

[2] HE T D, ZHANG F Z, MOU J. The Application of Automatic Storage & Retrieval System(AS/RS) in Management of Airplane Sheet Metal Mould[J]. Chinese Manufacturing Informatization, 2012, 41(5): 67-70. (in Chinese)
何腾达，张方哲，牟菊. 自动化立体仓库系统在飞机钣金模具管理中的应用[J]. 中国制造业信息化, 2012, 41(5): 67-70.

[3] GU J X, GOETSCHALCKX M, MCGINNIS L F. Research on warehouse operation: A comprehensive review[J]. European Journal of Operational Research, 2007, 177(1): 1-21.

[4] MUPPANI V R, ADIL G K. A branch and bound algorithm for class based storage location assignment[J]. European Journal of Operational Research, 2008, 189(2): 492-507.

[5] CUI Y. Location assignment optimization of AS/RS based on improved particle swarm optimization[J]. Computer Engineering & Applications, 2008, 44(11): 1-4.

[6] LI M, CHEN X, LIU C. Pareto and niche genetic algorithm for storage location assignment optimization problem[C]// 3rd International Conference on Innovative Computing Information and Control. 2008: 465-465.

[7] YANG W, ZHANG W Y, CHANG Y B, et al. Optimization of location assignment in AS/RS[J]. Modern Manufacturing Engineering, 2014(12): 134-140. (in Chinses)