第 33 卷第 10 期
2016 年 10 月

控 制 理 论 与 应 用
Control Theory & Applications

Vol. 33 No. 10
Oct. 2016

DOI: 10.7641/CTA.2016.50982

一种求解企业员工指派问题的离散状态转移算法

董天雪, 阳春华, 周晓君†, 桂卫华

(中南大学 信息科学与工程学院, 湖南 长沙 410083)

摘要: 员工指派问题是运筹学中的一类整数规划问题, 为了寻找最佳的员工指派方案, 使得完成所有任务的总成本代价最小, 本文研究了一种新的离散状态转移算法. 在一次状态转移的基础上提出了二次状态转移的概念, 从而扩大了候选解集的范围, 并提高候选解集的多样性. 为了克服算法在迭代后期更新缓慢的缺点, 提出了停滞回溯策略, 即当算法陷入局部最优解时进行回溯操作, 从历史停滞解中随机选择一个更新当前最优解. 通过与模拟退火算法进行测试比较实验, 证明了本文所提出算法的有效性, 同时该算法提高了求解员工指派问题的成功率与稳定性.

关键词: 指派问题; 离散状态转移算法; 二次状态转移; 停滞回溯; 整数规划

中图分类号: TP273　　**文献标识码:** A

A novel discrete state transition algorithm for staff assignment problem

DONG Tian-xue, YANG Chun-hua, ZHOU Xiao-jun†, GUI Wei-hua

(School of Information Science and Engineering, Central South University, Changsha Hunan 410083, China)

Abstract: The staff assignment problem is a kind of integer programming problem in operations research. In order to find the optimal staff assignment scheme with minimal total cost, this paper proposes a novel discrete state transition algorithm and puts forward the concept of second transition on the basis of first transition, which is helpful to expand the range of candidate solutions and improve the diversity of the candidates. To overcome the shortcomings of slow convergence of the algorithm at a later stage, stagnation backtracking strategy is proposed; that is to say, when the algorithm is stagnated into local minima, the backtracking operation is performed, and the current optimal solution is randomly selected from previously stagnant solutions. Finally, the experiments are verified and compared with the simulated annealing algorithm to prove the validity of these two strategies. Simulation results have showed the effectiveness of the improved method . Meanwhile, the method can improve the success rate and stability for this problem.

Key words: assignment problem; discrete state transition algorithm; second transition; stagnation backtracking; integer programming

1　问题描述(Problem description)

企业员工指派问题在人力资源规划中是必不可少的, 为了合理地安排员工完成任务, 企业必须权衡任务需求、员工技能和人工成本做出决策, 寻求到最佳的指派方案, 做到“因岗设职, 动态适应”的原则. 以企业指派员工进行产品售后服务为例: 某一大型设备生产企业需要为购买其设备的用户进行售后服务, 购买设备的用户分别位于不同的城市, 每个城市需指派一名员工进行设备维护. 一方面, 考虑到每个城市需要维护的设备不相同, 每个员工由于维修技能和任务性质的差异造成完成效率的不同, 因此指派员工去城市完成维修任务所需付出代价的不同. 另一方面, 位于不同城市的员工之间通常需要互相拨打电话讨论技术难点、及时沟通工作, 鉴于两两城市之间通话费率的不相同(暂不考虑全球通服务, 因为并不是每位员工均会办理该业务), 企业通常需要报销指派员工的通讯费用. 企业必须寻找最佳的员工指派方案, 从而使得完成所有任务的总成本代价最小.

上述员工指派问题的形式化描述如下: 企业抽调n个员工派送到n个城市进行售后服务与维修. 第i个员工派去第k个城市所要付出的代价为c_{ik}. 另一方面, 由于n个员工的技能不同, 当员工i处理位于k城市的任务时, 员工i会打电话向位于l城市的j员工求助, 他们的通话时间为t_{ij}, 而其中第k, l两个城市之

收稿日期: 2015−12−10; 录用日期: 2016−07−22.

†通信作者. E-mail: michael.x.zhou@csu.edu.cn; Tel.: +86 13787052648.

本文责任编委: 胡跃明.

国家自然科学基金项目(61533021, 61503416), 中南大学创新驱动计划项目(2015cx007), 中南大学探索项目(7131253)资助.

Supported by National Natural Science Foundation of China (61533021, 61503416), Innovation-driven Plan in Central South University (2015cx007) and Key Exploration Project (7131253).

间的通话费率为f_{kl}(假设接电话不需要花钱). 要求确定员工指派方案, 以最小的代价完成总任务, 同时必须满足以下任务条件: 每个城市只能分配一名员工, 每个员工也只能去一个城市.

该企业员工指派问题的数学模型为

$$\min Q(x) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} t_{ij}f_{kl}x_{ik}x_{jl} + \sum_{i=1}^{n}\sum_{k=1}^{n} c_{ik}x_{ik}, \tag{1}$$

$$\text{s.t.}\begin{cases} \sum\limits_{i=1}^{n} x_{ik} = 1,\ \forall k = 1,2,\cdots,n, \\ \sum\limits_{k=1}^{n} x_{ik} = 1,\ \forall k = 1,2,\cdots,n, \\ x_{ik} \in \{0,1\},\ \forall i,k. \end{cases}$$

该模型中各变量的含义见表1.

表 1 员工指派问题模型中的变量含义

Table 1 Variable meaning in staff assignment problem model

变量	含义
x_{ik}	决策变量: 表示是否派第i个员工去第k座城市, 即当$x_{ik}=0$表示不派第i个员工去第k座城市; 当$x_{ik}=1$表示派第i个员工去第k座城市;
n	员工指派问题的规模, 即城市数或员工数;
c_{ik}	指派员工i去城市k企业付出的代价;
t_{ij}	位于不同城市的员工i和员工j的通话时间;
f_{kl}	第k个城市与第l个城市两两之间的通话费率.

下面用图解的方式分析上述指派问题中员工与城市之间的企业需付代价关系, 如图1所示. 若问题规模为3, 即城市数与员工数均为3, 员工A, B, C可分别对应序号1, 2, 3, 城市北京、上海和深圳可分别对应序号1, 2, 3. 以员工A(序号1)为例, 其可去的城市有北京(序号1)、上海(序号2)和深圳(序号3), 则指派员工A去上述3个城市需付出的代价分别为C_{11}, C_{12}和C_{13}, 以此类推.

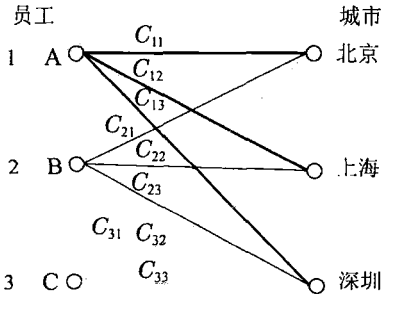

图 1 员工与城市之间的代价关系

Fig. 1 Cost relationship between the staff and the city

在员工指派方案中, 必须同时满足以下任务条件: 即每个城市只能分配一名员工, 每个员工也只能去一个城市. 根据式(1), 满足上述案例的其中一个可行解为

$$\begin{matrix} x_{11}=0,\ x_{12}=1,\ x_{13}=0 \\ x_{21}=0,\ x_{22}=0,\ x_{23}=1 \\ x_{31}=1,\ x_{32}=0,\ x_{33}=0 \end{matrix} \rightarrow \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}. \tag{2}$$

根据式(2)可知, 员工1被派去城市2, 即员工A被派去城市上海; 员工2被派去城市3, 即员工B被派去城市深圳; 员工3被派去城市1, 即员工C被派去城市北京. 由于位于不同城市的员工之间会互相沟通, 则其通讯时间与费用的代价关系如图2所示.

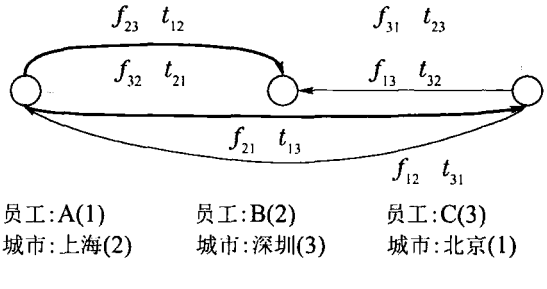

图 2 员工之间的通话时间和通话费用

Fig. 2 Time and call charge between staffs

由于只需考虑通话方所付的话费, 则员工A打电话给员工B所需费用为$f_{23} * t_{12}$, 员工B打电话给员工A所需费用为$f_{32} * t_{21}$, 按图以此类推. 其中, 员工之间拨打电话的时间矩阵t, 城市两两之间的通话费率矩阵f如式(3)所示:

$$t = \begin{bmatrix} 0 & t_{12} & t_{13} \\ t_{21} & 0 & t_{23} \\ t_{31} & t_{32} & 0 \end{bmatrix},\ f = \begin{bmatrix} 0 & f_{12} & f_{13} \\ f_{21} & 0 & f_{23} \\ f_{31} & f_{32} & 0 \end{bmatrix}. \tag{3}$$

结合式(1)–(3)可知, 上述例子的目标值为

$$\begin{aligned} Q = {} & t_{12}f_{23}x_{12}x_{23} + t_{13}f_{23}x_{12}x_{33} + \\ & t_{23}f_{31}x_{23}x_{31} + t_{21}f_{32}x_{23}x_{12} + \\ & t_{31}f_{12}x_{31}x_{12} + t_{32}f_{13}x_{31}x_{23} + \\ & C_{12} + C_{23} + C_{31}. \end{aligned} \tag{4}$$

当$n=3$时的其他可行解可依照上述例子得出其对应的目标值, 同时选择目标值最小的可行解, 即为最优解.

本文组织结构如下: 第2节给出现有指派问题研究算法的综述, 同时介绍了约束处理策略, 即如何将具有约束的员工指派问题化为无约束的员工指派问题; 第3节在已有离散状态转移算法的基础上, 提出了二次状态转移以及停滞回溯策略; 第4节进行了一系列的实验来测试本文所提新的离散状态转移算法的性能; 第5节总结全文.

2 相关研究工作(Related research work)

2.1 研究现状(Research status)

本文研究的企业员工任务指派问题是一个典型的混合指派问题, 其数学模型不仅包含线性的标准形式, 如式(5)所示, 而且还包括了二次指派问题的形式, 如式(6)所示:

$$\min Q(x)=\sum_{i=1}^{n}\sum_{k=1}^{n}c_{ik}x_{ik}, \tag{5}$$

$$\text{s.t.}\begin{cases}\sum\limits_{i=1}^{n}x_{ik}=1,\ \forall k=1,2,\cdots,n,\\ \sum\limits_{k=1}^{n}x_{ik}=1,\ \forall k=1,2,\cdots,n,\\ x_{ik}\in\{0,1\},\ \forall i,k,\end{cases}$$

$$\min Q(x)=\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}t_{ij}f_{kl}x_{ik}x_{jl}, \tag{6}$$

$$\text{s.t.}\begin{cases}\sum\limits_{i=1}^{n}x_{ik}=1,\ \forall k=1,2,\cdots,n,\\ \sum\limits_{k=1}^{n}x_{ik}=1,\ \forall k=1,2,\cdots,n,\\ x_{ik}\in\{0,1\},\ \forall i,k.\end{cases}$$

从求解方法角度来看, 员工指派问题属于NP–hard问题, 综合目前的研究情况, 求解标准形式指派问题比较经典的算法是匈牙利算法[1], 匈牙利法的最大特点就是能够科学的对人员和工作进行合理的分配, 做到“人事匹配”. 但匈牙利算法在求解大规模任务分配时运算效率不高, 并且其运算过程较繁琐, 如果计算过程中间稍有差错, 就可能导致最终结果的错误. 鉴于匈牙利算法不能求解大规模问题的缺点, 在文献[2]中提出了一种新的求解任务分配问题的方法—剪枝优化算法. 该算法通过逐步剔除已确定的部分分配方案对应代价矩阵元素, 逐次降低分配问题的规模, 从而实现快速求解全局任务分配问题, 但是该算法的适应范围仅用于求解代价矩阵中元素均为非负值的问题.

除此之外, D. P. Bertsekas等[3]提出的拍卖算法也可以用于求解标准形式的任务指派问题, 但是该算法只能终止于可行分配, 所以当问题没有可行解时, 由于用户不清楚问题是没有可行解还是难解, 拍卖算法将一直迭代下去.

综合目前的研究情况, 研究人员已经提出不同的确定性算法和启发式算法求解二次指派问题. 其中, 确定性算法仅适合求解小规模的二次指派问题, 而启发式算法可以在合理的优化时间内求解较大规模问题的近似最优解[4]. 在利用确定性算法求解二次指派问题的一个全局最优解时, 最常用的算法包括分支定界法、割平面法、分支切割混合法以及动态规划法. 分支定界法利用剪枝规则定义问题的下界, Gilmore[5], Land[6] 和Lawler[7]最初用分支定界法来剔除不被需要的解决方案, 文献[8]证明了分支定界法求解二次指派问题的可行性和有效性.

近年来, 利用分支定界法与并行化实现技术相结合的程序求解问题, 可得到最优解. 动态规划法用来求解特殊的二次指派问题, 其中, 该类问题的流程矩阵以树型的邻接矩阵表示, 文献[9]利用混合整数线性规划方法来松弛问题, 然后利用动态规划算法求解问题.

割平面法由Bazaraa和Sherali于1980年提出, 最初求解问题时并不能取得令人满意的结果, 后来他们利用混合整数线性规划和Benders分解法处理问题模型, 到目前为止, 该技术并没有被广泛应用, 仅能较好地处理一些二次指派问题, 鉴于其收敛速度慢, 所以只能用来处理很小规模的问题. 分支切割技术是由Padberg和Rinaldi[10]于1991年提出的一种另类的切割策略, 其主要切割由问题可行解定义的多面体的侧面, 其切割侧面的效果优于割平面法的切割效果, 因此可以加速其收敛到最优解的速度, 但由于二次指派问题多面体知识的缺乏导致了该算法不能广泛应用于求解该类问题.

遗传算法[11]、模拟退火算法[12]、神经网络算法[13]等都是已经成功应用于求解二次指派问题的启发式算法. 教学与优化算法是一种新型的群智能算法, 它分为两个阶段, 第1阶段是所有人的训练, 在第2阶段教师与同学们提高自己的知识水平. 文献[14]提出了一种基于混合算法来求解二次指派问题的教学与优化算法, 个体在利用重组操作训练后, 再使用禁忌搜索算法处理所有个体.

针对企业混合员工指派问题, 本文提出了一种新的基于状态转移思想[15]的算法来求解上述整数规划问题. 本文提出的离散状态转移算法利用特殊的约束处理策略, 将该具有约束的员工指派问题化为无约束指派问题.

2.2 约束处理策略(Constraint handling strategy)

根据第1节介绍的企业员工指派问题模型可知, 令

$$C=(c_{ik}),\ M=(x_{ik}),$$

则矩阵C称为任务分配问题的价值系数矩阵, 矩阵M称为分配问题解的匹配矩阵, 其表示形式如下:

$$M=(x_{ik})=$$

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\ x_{31} & x_{32} & x_{33} & \cdots & x_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \cdots & x_{nn} \end{bmatrix}. \tag{7}$$

由于每个城市只能分配一名员工, 每个员工也只能去一个城市, 所以矩阵解M中每一行和每一列的x_{ik}有且仅有一个值为1, 其余为零; 如此满足了上述约束条件.

鉴于文献[16]中旅行商问题的最优解以城市号的排列顺序表示, 如$\{1, 2, \cdots, n\}$. 则本文员工指派问题的最优解也可采取上述方法表示, 如此即可建立员工指派问题的矩阵解与向量排列之间的一一映射关系. 如式(1)中, 当$n = 5$, 即城市数或员工数为5, 满足约束条件的变量x_{ik}组成的最优解可以用式(8)中矩阵M表示(x_{ik}下标中的i代表第i个员工, k代表了第k座城市).

$$\begin{cases} M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \leftrightarrow \begin{pmatrix} x_{11}=1 \\ x_{24}=1 \\ x_{33}=1 \\ x_{42}=1 \\ x_{55}=1 \end{pmatrix} \leftrightarrow \begin{pmatrix} \text{第1个员工被派去第1座城市} \\ \text{第2个员工被派去第4座城市} \\ \text{第3个员工被派去第3座城市} \\ \text{第4个员工被派去第2座城市} \\ \text{第5个员工被派去第5座城市} \end{pmatrix} \leftrightarrow \begin{bmatrix} 1 \\ 4 \\ 3 \\ 2 \\ 5 \end{bmatrix}, \\ M = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \leftrightarrow \begin{pmatrix} x_{12}=1 \\ x_{23}=1 \\ x_{34}=1 \\ x_{45}=1 \\ x_{51}=1 \end{pmatrix} \leftrightarrow \begin{pmatrix} \text{第1个员工被派去第2座城市} \\ \text{第2个员工被派去第3座城市} \\ \text{第3个员工被派去第4座城市} \\ \text{第4个员工被派去第5座城市} \\ \text{第5个员工被派去第1座城市} \end{pmatrix} \leftrightarrow \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 1 \end{bmatrix}, \\ M = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \leftrightarrow \begin{pmatrix} x_{13}=1 \\ x_{21}=1 \\ x_{32}=1 \\ x_{45}=1 \\ x_{54}=1 \end{pmatrix} \leftrightarrow \begin{pmatrix} \text{第1个员工被派去第3座城市} \\ \text{第2个员工被派去第1座城市} \\ \text{第3个员工被派去第2座城市} \\ \text{第4个员工被派去第5座城市} \\ \text{第5个员工被派去第4座城市} \end{pmatrix} \leftrightarrow \begin{bmatrix} 3 \\ 1 \\ 2 \\ 5 \\ 4 \end{bmatrix}. \end{cases} \tag{8}$$

通过分析式(7)可知, 员工指派问题的最优解以矩阵解M的形式表示时, 它与矩阵列向量的排序存在一一对应关系. 因此, 满足员工指派问题约束条件的向量置换$[1\ 2\ 3\ \cdots\ n]^{\mathrm{T}}$就是该问题的最优解.

3 离散状态转移算法的原理与计算步骤 (Principles and computational procedures of discrete state transition algorithm)

状态转移算法[17–22]是近几年发展起来的一种新型随机性全局优化算法, 基本思想是将优化问题的一个可行解作为一种状态, 可行解的更新过程被视为从一种状态向另一种状态变化, 即状态转移. 借助于状态空间的概念, 离散状态转移算法产生候选解的基本框架可描述如下:

$$\begin{cases} x_{k+1} = A_k x_k \oplus B_k u_k, \\ y_{k+1} = f(x_{k+1}), \end{cases} \tag{9}$$

其中: $x_k \in \mathbb{Z}^n$代表当前的状态, 对应着优化问题的一个解; $A_k, B_k \in \mathbb{R}^{n\times n}$为状态转移矩阵, 可看成是优化算法中的算子; $\oplus$是运算符, 将两种状态联系在一起; $u_k \in \mathbb{Z}^n$是关于当前状态和历史状态的函数; $f(x)$是代价函数或评价函数.

3.1 状态变换算子(State transformation operators)

为了解决状态转移算法求解离散优化问题时状态转移过程的可控性、高效性, 设计了3种特殊的状态变换算子(见参考文献[16]).

1) 交换变换算子

$$x_{k+1} = A_k^{\mathrm{swap}} x_k, \tag{10}$$

其中$A_k^{\mathrm{swap}} \in \mathbb{R}^{n\times n}$称为交换变换矩阵, 是一个带有交换变换功能的随机0–1矩阵. 该算子具有交换x_k中随机两个位置元素的能力. 比如:

$$\begin{pmatrix} 1 \\ 5 \\ 3 \\ 4 \\ 2 \\ 6 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{pmatrix}, \tag{11}$$

在这里, 第2号和第5号位置上的元素发生交换.

2) 移动变换算子

$$x_{k+1} = A_k^{\text{shift}} x_k, \tag{12}$$

其中$A_k^{\text{shift}} \in \mathbb{R}^{n\times n}$称为移动变换矩阵, 是一个带有移动变换功能的随机0–1矩阵. 该算子具有将x_k中某个随机位置的元素移动到另一个随机位置后面的能力. 比如:

$$\begin{pmatrix}1\\2\\4\\5\\3\\6\end{pmatrix} = \begin{pmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\\0&0&1&0&0&0\\0&0&0&0&0&1\end{pmatrix} \times \begin{pmatrix}1\\2\\3\\4\\5\\6\end{pmatrix}. \tag{13}$$

在这里, 第3号元素移动到第5号元素后面.

3) 对称变换算子

$$x_{k+1} = A_k^{\text{sym}} x_k, \tag{14}$$

其中$A_k^{\text{sym}} \in \mathbb{R}^{n\times n}$称为对称变换矩阵, 是一个带有对称变换功能随机0–1矩阵. 该算子具有将x_k中某两个随机位置之间所有元素对称或倒置的能力. 比如:

$$\begin{pmatrix}1\\5\\4\\3\\2\\6\end{pmatrix} = \begin{pmatrix}1&0&0&0&0&0\\0&0&0&0&1&0\\0&0&0&1&0&0\\0&0&1&0&0&0\\0&1&0&0&0&0\\0&0&0&0&0&1\end{pmatrix} \times \begin{pmatrix}1\\2\\3\\4\\5\\6\end{pmatrix}, \tag{15}$$

在这里, 对第2, 3, 4, 5号元素进行对称变换, 其位置发生倒置.

在状态转移算法中, 随机性主要体现在状态变换算子中, 离散状态算法中有3个功能算子, 包括交换变换、平移变换和对称变换, 这3个算子中的交换变换矩阵、平移变换矩阵、对称变换矩阵都是带有特殊功能的随机矩阵. 其中交换变换和平移变换影响局部搜索性能, 对称变换影响全局搜索性能, 更为详细的介绍见参考文献[22–23]. 此外, 以交换变换算子为例, 其交换变换矩阵的MATLAB实现如下:

function $y = \text{swap\_matrix}(n)$

$y = \text{eye}(n)$;

$R = \text{randperm}(n)$;

$T = R(1:2)$;

$S = \text{fliplr}(T)$;

$y(T,:) = y(S,:)$.

通过设计如上的交换变换矩阵, 对于给定的当前解, 在交换变换算子的作用下, 会产生具有交换两个随机位置元素功能的候选解. 其他状态变换矩阵的实现及基本离散状态转移算法的MATLAB代码见下载链接[24].

3.2 一次状态转移(First transition)

若利用上述3种搜索算子针对候选解只做一次状态转移操作, 则称为一次状态转移[22].

$$\{1\ \ 3\ \ 2\ \ 5\ \ 4\} \to \begin{pmatrix}\{3&1&2&5&4\}\\\{2&3&1&5&4\}\\\{5&3&2&1&4\}\\\{4&3&2&5&1\}\\\{1&2&3&5&4\}\\\{1&5&2&3&4\}\\\{1&4&2&5&3\}\\\{1&3&5&2&4\}\\\{1&3&4&5&2\}\\\{1&3&2&4&5\}\end{pmatrix}. \tag{16}$$

以候选解$\{1,3,2,5,4\}$为例, 利用交换变换算子对该候选解进行一次状态转移操作, 则所有可能候选解方案如式(16)所示.

根据式(16)可知, 利用交换变换算子进行一次状态操作, 有C_5^2种可能候选解. 若存在一个n维的离散优化问题, 则该问题的一个可行解经过一次状态转移后, 会有C_n^2 种可能候选解.

3.3 二次状态转移(Second transition)

为了提高候选解的多样性, 本文基于一次状态转移的概念提出了二次状态转移. 经过一次状态转移得到的候选解称之为一次候选解集, 在一次候选解集的基础上, 利用相同的搜索算子进行状态转移变换, 得到二次候选解集, 上述过程即二次状态转移的概念. 可将其视为在一次状态转移的基础上再进行一次状态转移, 如此有助于扩大候选解集的范围.

以候选解$\{1,3,2,5,4\}$为例, 利用交换变换算子进行二次状态转移, 得到所有可能的候选解如式(17)所示:

$$
\begin{aligned}
\{1,3,2,5,4\}\rightarrow\ & \{3,1,2,5,4\}\Rightarrow\begin{pmatrix}\{1,3,2,5,4\} & \{2,1,3,5,4\} & \{5,1,2,3,4\} & \{4,1,2,5,3\} & \{3,2,1,5,4\}\\ \{3,5,2,1,4\} & \{3,4,2,5,1\} & \{3,1,5,2,4\} & \{3,1,4,5,2\} & \{3,1,2,4,5\}\end{pmatrix},\\
& \{2,3,1,5,4\}\Rightarrow\begin{pmatrix}\{3,2,1,5,4\} & \{1,3,2,5,4\} & \{5,3,1,2,4\} & \{4,3,1,5,2\} & \{2,1,3,5,4\}\\ \{2,5,1,3,4\} & \{2,4,1,5,3\} & \{2,3,5,1,4\} & \{2,3,1,5,4\} & \{2,3,1,4,5\}\end{pmatrix},\\
& \{5,3,2,1,4\}\Rightarrow\begin{pmatrix}\{3,5,2,1,4\} & \{2,3,5,1,4\} & \{1,3,2,5,4\} & \{4,3,2,1,5\} & \{5,2,3,1,4\}\\ \{5,1,2,3,4\} & \{5,4,2,1,3\} & \{5,3,1,2,4\} & \{5,3,4,1,2\} & \{5,3,2,4,1\}\end{pmatrix},\\
& \{4,3,2,5,1\}\Rightarrow\begin{pmatrix}\{3,4,2,5,1\} & \{2,3,4,5,1\} & \{5,3,2,4,1\} & \{1,3,2,5,4\} & \{4,2,3,5,1\}\\ \{4,5,2,3,1\} & \{4,1,2,5,3\} & \{4,3,5,2,1\} & \{4,3,1,5,2\} & \{4,3,2,1,5\}\end{pmatrix},\\
& \{1,2,3,5,4\}\Rightarrow\begin{pmatrix}\{3,4,2,5,1\} & \{2,3,4,5,1\} & \{5,3,2,4,1\} & \{1,3,2,5,4\} & \{4,2,3,5,1\}\\ \{4,5,2,3,1\} & \{4,1,2,5,3\} & \{4,3,5,2,1\} & \{4,3,1,5,2\} & \{4,3,2,1,5\}\end{pmatrix},\\
& \{1,5,2,3,4\}\Rightarrow\begin{pmatrix}\{2,1,3,5,4\} & \{3,2,1,5,4\} & \{5,2,3,1,4\} & \{4,2,3,5,1\} & \{1,3,2,5,4\}\\ \{1,5,3,2,4\} & \{1,4,3,5,2\} & \{1,2,5,3,4\} & \{1,2,4,5,3\} & \{1,2,3,4,5\}\end{pmatrix},\\
& \{1,4,2,5,3\}\Rightarrow\begin{pmatrix}\{4,1,2,5,3\} & \{2,4,1,5,3\} & \{5,4,2,1,3\} & \{3,4,2,5,1\} & \{1,2,4,5,3\}\\ \{1,5,2,4,3\} & \{1,3,2,5,4\} & \{1,4,5,2,3\} & \{1,4,3,5,2\} & \{1,4,2,3,5\}\end{pmatrix},\\
& \{1,3,5,2,4\}\Rightarrow\begin{pmatrix}\{3,1,5,2,4\} & \{5,3,1,2,4\} & \{2,3,5,1,4\} & \{4,3,5,2,1\} & \{1,5,3,2,4\}\\ \{1,2,5,3,4\} & \{1,4,5,2,3\} & \{1,3,2,5,4\} & \{1,3,4,2,5\} & \{1,3,5,4,2\}\end{pmatrix},\\
& \{1,3,4,5,2\}\Rightarrow\begin{pmatrix}\{3,1,4,5,2\} & \{4,3,1,5,2\} & \{5,3,4,1,2\} & \{2,3,4,5,1\} & \{1,4,3,5,2\}\\ \{1,5,4,3,2\} & \{1,2,4,5,3\} & \{1,3,5,4,2\} & \{1,3,2,5,4\} & \{1,3,4,2,5\}\end{pmatrix},\\
& \{1,3,2,4,5\}\Rightarrow\begin{pmatrix}\{3,1,2,4,5\} & \{2,3,1,4,5\} & \{4,3,2,1,5\} & \{5,3,2,4,1\} & \{1,2,3,4,5\}\\ \{1,4,2,3,5\} & \{1,5,2,4,3\} & \{1,3,4,2,5\} & \{1,3,5,4,2\} & \{1,3,2,5,4\}\end{pmatrix}.
\end{aligned}
\tag{17}
$$

将式(16)与式(17)对比可知, 经过二次状态转移获得的候选解超过经过一次状态转移获得的候选解数目. 忽略式(17)与式(16)重复的候选解, 则有$C_5^2 * C_5^2$种可能候选解. 若存在一个N维的离散优化问题, 则该问题的一个可行解经过一次状态转移后, 会有$C_n^2 * C_n^2$种可能候选解. 由此可知, 二次状态转移可以扩大搜索空间.

需要补充的是, 二次状态转移与一次状态转移的区别在于产生的候选解集不同, 且二次状态转移产生的候选解集包含一次状态转移产生的候选解集. 以交换变换为例, 假如问题的规模是n, 通过一次状态转移, 理论上将会产生C_n^2个候选解, 而通过二次状态转移, 理论上将会产生$C_n^2 * C_n^2$个候选解, 但是考虑到搜索力度SE是固定的, 从候选解集中采样到的样本数目是固定的, 即样本大小固定为SE, 考虑到产生候选解的时间复杂度是很低的, 算法的耗时主要用在样本的优劣比较上, 由于样本大小相同, 故两种状态转移的耗时是近似的, 因而不能认为二次状态转移的运行时间是一次状态转移的两倍. 此外, 从计算效果上看, 从二次状态转移候选解集中采样的SE个样本含有比一次状态转移更丰富的状态信息, 或者说二次状态转移具有更大的搜索空间, 故不能认为将一次状态转移的迭代次数加倍可以取得与二次状态转移相同的效果, 这个从下面的表5中两种状态转移的比较可以看出.

3.4 停滞回溯策略(Stagnation backtracking strategy)

离散状态转移算法是一种迭代算法, 利用其固有的操作算子求解问题的当前最优解. 迭代前期, 利用3种操作算子寻优过程中, 当前最优值下降迅速; 进入迭代后期, 当前最优值不仅下降缓慢, 而且当前最优解出现多次重复. 若多次迭代后, 仍未跳出当前最优解, 两种原因可以解释这种情况: 原因1, 算法求解问题过程中陷入了局部最优, 出现停滞现象; 原因2, 全局最优解处在孤岛上, 难以成功搜索到.

针对上述问题, 为了跳出局部最优, 本文提出了停滞回溯策略. 该策略描述如下: 若迭代一定次数后, 利用操作算子求解的结果陷入某个极值且不更新到一定的次数, 累计连续出现此极值的次数, 若超过一定的次数, 则将该极值对应的重复最优解称为停滞解. 其中, 累计出现的重复次数标记为flag. 同时, 将所有满足条件的停滞解均存储起来. 若当前停滞解的重复次数为m*flag($m \in$Integer), 此停滞解可以称之为陷入解. 为了摆脱重复最优的情况, 从历史停滞解中随机选择一个作为当前最优解进入

下一次迭代计算, 最后利用贪婪准则, 选择接收所有由操作算子产生的目标函数值中最小的候选解.

综上所述, 该更新策略包括两部分: 第1部分是存储满足条件的停滞解; 第2部分是更新局部解. 如图3所示, 该图显示的是一个$n=3$的问题的求解过程, 且$m=3$. 其中, 解{1,3,2}、解{2,3,1}、解{3,2,1}和解{3,1,2}的重复次数均大于flag, 则上述4个解均称之为停滞解, 其组成了一个历史停滞解集. 鉴于最后一个停滞解的重复次数为3*flag, 则最后一个停滞解是陷入解, 为了摆脱重复最优的情况, 需随机选择$\{1,3,2\}$, $\{2,3,1\}$和$\{3,2,1\}$中的一个作为当前最优解, 进入下一次迭代计算.

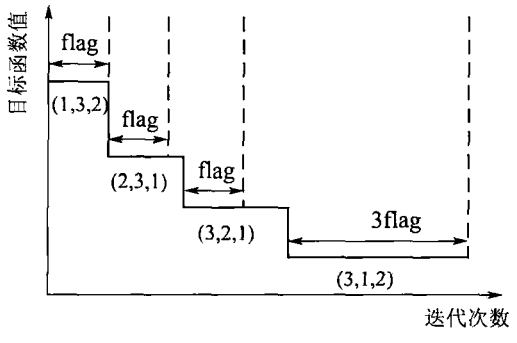

图 3 停滞解示意图

Fig. 3 Illustration of stagnation solution

3.5 离散状态转移算法的计算步骤(Computation steps of discrete state transition algorithm)

离散状态转移算法主要由初始化候选解、产生候选解和接收候选解3部分组成, 鉴于第2.2节介绍的约束处理策略, 该算法产生的所有候选解均为可行解. 离散状态转移算法的计算步骤如下所示:

1) 初始化候选解: 随机生成多组形如$[1\ 2\ 3\cdots n]^{\mathrm{T}}$的候选解. 其中$[1\ 2\ 3\cdots n]^{\mathrm{T}}$的数字序列随机生成;

2) 产生候选解: 利用二次状态转移概念生成多组不同的候选解; 由于本文利用了3种搜索算子, 所以有基于交换变换算子的二次状态转移、基于移动变换算子的二次状态转移和基于对称变换算子的二次状态转移.

3) 接收候选解: 在产生多组候选解后, 利用贪婪准则和停滞回溯策略, 选择接收目标函数值小的候选解.

如第2部分所述, 任务指派问题的一个矩阵解M对应着一个独一无二的矩阵列向量排列, 因此, 一个矩阵列向量排列也对应着唯一的一个矩阵解M.

本文提出的离散状态转移算法可以有效地处理向量置换, 避免了处理指派问题时的约束条件. 针对企业员工指派问题所提出的离散状态转移算法的流程如下所示:

算法 离散状态转移算法.

Best: 随机生成初始解

fBest $\leftarrow$ fitness(Best, funfcn): 计算目标函数适应值

$k \leftarrow 0$: k为迭代次数

flag $\leftarrow 0$

Repeat: 继续进行

[Best, fBest, flag] $\rightarrow$ swap(funfcn, Best, fBest, SE, flag)

[Best, fBest, flag] $\rightarrow$ shift(funfcn, Best, fBest, SE, flag)

[Best, fBest, flag] $\rightarrow$ symmetry(funfcn, Best, fBest, SE, flag)

[Best, fBest, flag] $\rightarrow$ update(funfcn, hisBest, Best, fBest, SE, flag)

$k \rightarrow k+1$

直到达到最大迭代次数

Matrix $X \leftarrow$ 将最优解的向量形式化为相应的矩阵形式

在上述伪代码中, 离散状态转移算法中集合的大小即搜索力度, 它类似于基于种群搜索算法的种群的大小, 会影响求解时间, 在相同的迭代次数下, 搜索力度越大, 求解时间越长, 本文中搜索力度$SE=20$取的是经验值. 此外, 在离散状态转移算法中, 始终保留搜索过程中寻找到的当前最优解Best, 并在当前最优解的基础上进行交换、平移和对称变换, 更新当前最优解也是采用贪婪准则, 这保证了当前最优解是历史搜索到的最优解.

4 实验结果与讨论(Experimental results and discussion)

为充分测试本文所提离散状态转移算法的性能和特点, 引入基于不同维数的矩阵系数T, F, C生成的员工指派问题来测试算法的寻优性能.

4.1 测试1(Test 1)

在该部分测试中, 分别以5维和10维的指派问题为例, 比较一次状态转移和二次状态转移效果的不同. 表2列举了5维问题和10维问题员工指派问题模型中每个t_{ij}的取值, 即系数矩阵T; 每个f_{kl}的取值, 即系数矩阵F和每个c_{ik}的取值, 即系数矩阵C. 上述3个系数矩阵均是随机生成的数据, 其中t_{ij}和c_{ik}的取值为1~100的随机整数, f_{kl}为0~1的随机小数.

表 2 员工指派问题的2个仿真实例

Table 2 Two simulation examples of staff assignment problem

编号	问题维数	参　数	最优解
k1	5	$T=[0,91,47,13,46;\ 38,0,13,1,5;\ 44,88,0,39,70;\ 11,58,89,0,79;\ 36,28,86,17,0];$ $F=[0,0.8416,0.0448,0.4068,0.4039;\ 0.7794,0,0.3054,0.9248,0.5331;\ 0.2204,0.7745,$ $0,0.9617,0.6658;\ 0.9215,0.3221,0.2478,0,0.9853;\ 0.5734,0.9833,0.2685,0.3772,0];$ $C=[90,7,93,51,17;\ 21,9,2,51,17;\ 21,9,2,49,41;\ 25,2,26,79,71;\ 4,41,26,30,70;$ $98,97,55,60,21];$	573.830
k2	10	$T=[0,41,10,64,47,48,20,64,77;\ 48,0,27,96,85,43,16,29,95,36;\ 70,72,0,25,35,$ $47,35,10,22,67;\ 70,97,29,0,78,77,61,58,71,42;\ 64,54,45,30,0,33,20,69,24,84;$ $4,33,53,68,2,0,74,55,13,83;\ 8,11,46,70,61,48,0,43,61,26;\ 33,61,88,8,39,5,$ $92,0,46,62;\ 54,78,52,26,92,18,28,65,0,59;\ 66,43,94,23,1,72,77,68,67,0];$ $F=[0,0.45,0.11,0.43,0.85,0.42,0.78,0.23,0.55,0.93;\ 0.79,0,0.96,0.91,0.62,0.05,0.39,$ $0.35,0.30,0.78;\ 0.31,0.23,0,0.18,0.35,0.90,0.24,0.82,0.74,0.49;\ 0.53,0.91,0.77,0,$ $0.51,0.94,0.40,0.02,0.19,0.44;\ 0.17,0.15,0.82,0.15,0,0.49,0.10,0.04,0.69,0.45;\ 0.60,$ $0.83,0.87,0.14,0.12,0.08,0,0.13,0.17,0.18,0.31;\ 0.26,0.54,0.08,0.87,0.24,0.34,0,$ $0.65,0.37,0.51;\ 0.65,1.00,0.40,0.58,0.90,0.96,0,0.63,0.51;\ 0.69,0.08,0.26,0.55,0.18,$ $0.37,0.58,0.65,0,0.82;\ 0.75,0.44,0.80,0.14,0.24,0.11,0.06,0.45,0.08,0];$ $C=[5,97,83,3,6,67,88,20,62,82;\ 8,65,81,98,74,54,67,43,27,27;\ 53,80,7,18,28,$ $70,20,49,83,60;\ 11,46,41,12,43,67,38,13,98,3;\ 82,44,53,38,55,19,47,59,73,43;$ $82,83,42,21,65,26,94,14,98,23,35,32;\ 73,9,66,49,42,100,16,39,59,17;\ 16,14,63,$ $35,98,18,86,59,12,19;\ 66,18,30,95,31,4,91,43;\ 52,40,44,92,70,57,38,30,88,10].$	2047.06

利用枚举法, 可以求得上述5维问题的最优值为573.8300, 对应的最优解为[2, 3, 4, 1, 5]; 10维问题的最优值为2047.0600, 对应的最优解为[4, 8, 6, 3, 1, 9, 7, 2, 5, 10].

该部分主要是测试一次状态转移和二次状态转移分别作用于k1, k2问题的的效果, 分别独立运行20次, 如表3所示, 从最好值、最差值、平均值等多方面对算法进行评估, 其中可行率是指在算法独立20次运行后, 获得可行解的次数占总运行次数的比率, 其旨在强调本文所提算法处理员工指派问题约束时的有效性, 本文在第2.2节提出了一种针对员工指派问题的特殊约束处理策略, 即下标表示法, 它将含有等式约束的员工指派问题变成了无约束优化问题, 从而确保每次迭代过程中生成的问题候选解都是可行解, 所以也就可同时保证每次迭代过程中约束处理策略的可行率为100%, 故而不会出现不可行解的情况. 另外, 成功率为运行20次运行后, 获得最优解的次数占总运行次数的比率.

表 3 一次状态转移与二次状态转移效果分析

Table 3 Effect analysis of first transition and second transition

问题维数	方 法	最好值	平均值	最差值	可行率/ %	成功率/ %
5	一次状态转移	573.8300	610.5835	825.7800	100	75
	二次状态转移	573.8300	573.8300	573.8300	100	100
10	一次状态转移	2047.0600	2209.7000	2144.3000	100	50
	二次状态转移	2047.0600	2080.4000	2101.3000	100	75

由表3可知, 利用一次状态转移和二次状态转移都具有较好的寻优效果, 都能得到已知5维和10维问题的全局最优解. 除此之外, 无论对于5维问题还是10维问题, 利用二次状态转移策略的平均值小于利用一次状态转移求取的平均值, 从而说明前者的算法稳定性优于后者.

根据第3.5节所述, 该问题的矩阵最优解如式(18)所示:

$$\begin{pmatrix}2\\3\\4\\1\\5\end{pmatrix}\rightarrow\begin{pmatrix}x_{12}\\x_{23}\\x_{34}\\x_{41}\\x_{55}\end{pmatrix}\rightarrow\begin{bmatrix}0&1&0&0&0\\0&1&0&0&0\\0&0&0&1&0\\1&0&0&0&0\\0&0&0&0&1\end{bmatrix}. \tag{18}$$

由于表3中的成功率代表了在利用算法计算20次的过程中,得到已知最优解的比率,故而可知求解过程中击中最优解的次数. 根据表3可知,无论是5维问题还是10维问题,利用一次状态转移求解问题的成功率均小于利用二次状态转移求解问题的成功率. 换而言之,采用二次状态转移策略获得全局最优解的次数多于利用一次状态转移求得全局最优解的次数,这也表明了二次状态转移的全局搜索能力优于一次状态转移.

此外,图4显示了一次状态转移和二次状态转移运行20次运行结果的箱形图. 根据图4可知,一次状态转移的箱形比二次状态转移的箱形要宽很多. 由此可见,二次状态转移的收敛性优于一次状态转移.

4.2 测试2(Test 2)

在该部分测试中,依旧以5维和10维的指派问题为例,测试停滞回溯策略的的效果,比较利用停滞回溯策略和无停滞回溯策略的实验结果,分别运算20遍. 将所得实验结果及分析汇聚于表4. 由表4可知,利用停滞回溯策略的算法在20次计算中每次都能得到已知最优解,充分显示停滞回溯策略在离散空间中有较强的全局寻优能力和计算鲁棒性.

表 4 无停滞回溯策略的离散状态转移算法与有停滞回溯策略的离散状态转移算法效果分析

Table 4 Effect analysis of discrete transition algorithm with the stagnation backtracking strategy

问题维数	方 法	最好值	平均值	最差值	可行率/ %	成功率/ %
5	无停滞回溯策略	573.8300	573.8300	573.8300	100	100
	有停滞回溯策略	573.8300	573.8300	573.8300	100	100
10	无停滞回溯策略	2047.0600	2080.4000	2144.3000	100	75
	有停滞回溯策略	2047.0600	2047.0600	2047.0600	100	100

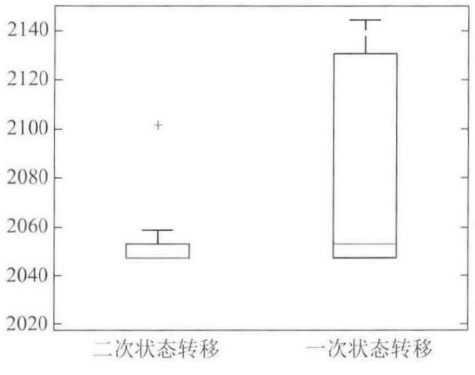

图 4 一次状态转移与二次状态求解10维问题结果比较图

Fig. 4 Comparison between first transition and second transition in 10-dimension problem

4.3 测试3(Test 3)

在该部分测试中,以10, 15, 20, 30, 40和50维的指派问题为例,测试了本文提出的离散状态转移算法的性能,同时比较离散状态转移算法与模拟退火算法[23]的实验结果,其中文献[23]采用的是最广泛采用的2–交换邻域函数,保留其设计的模拟退火算法思路用来求解本文所提的员工指派问题. 将算法分别独立运行20次,如图5至图7所示,其绘制的是最优解的平均收敛曲线,可以看出离散状态转移算法在收敛性和全局搜索能力上均有优势.

由表5可知,利用本文提出的离散状态转移算法求解上述4个问题时,其求得的最好值均优于利用模拟退火算法求得的结果,体现了本文提出的算法的有效性. 除此之外,离散状态转移算法求解每个问题所需时间均少于模拟退火算法所需时间,表明了算法计算的快速性. 同时,采用CPLEX进行求解10维问题时,得到的确定性解与离散状态转移算法的解相同,证明了离散状态转移算法求解能力. 在求解效率方面,CPLEX的运行时间为775.23 s,而离散状态转移算法仅需11.74 s,验证了本文所提出算法的快速性. 由于CPLEX的运行时间随着变量个数的增加呈指数增长,在求解15维及以上问题时需耗费大量的时间(变量个数为225,运行时间大于8个小时),因此本文只对10维问题进行精确求解. 此外,当忽略整数约束,将混合员工指派问题松弛成的二次规划问题时,由于其对角元素为0,故仍为非凸连续优化问题,很难求得其下界,故本文算法的效果只能通过与模拟退火算法比较中看出.

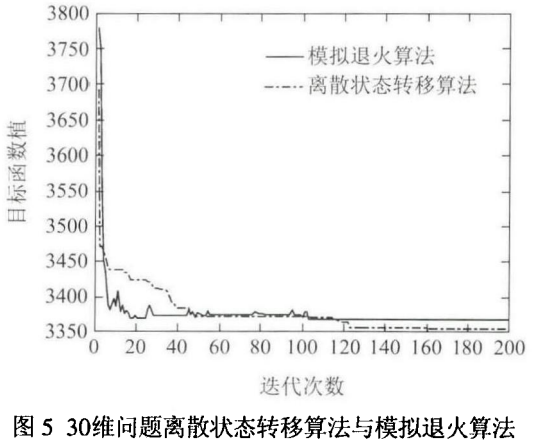

图 5 30维问题离散状态转移算法与模拟退火算法效果比较图

Fig. 5 Effect comparison between discrete state transition algorithm and simulated annealing in 30-dimension problem

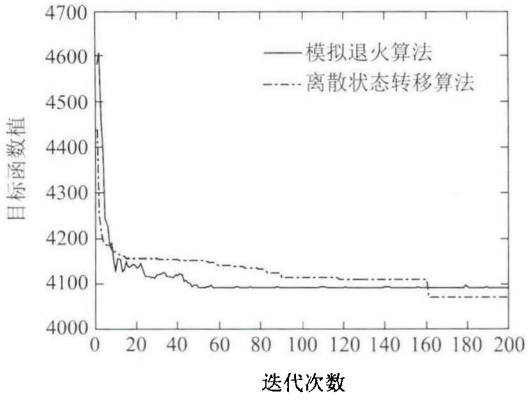

图 6 40维问题离散状态转移算法与模拟退火算法效果比较图

Fig. 6 Effect comparison between discrete state transition algorithm and simulated annealing in 40-dimension problem

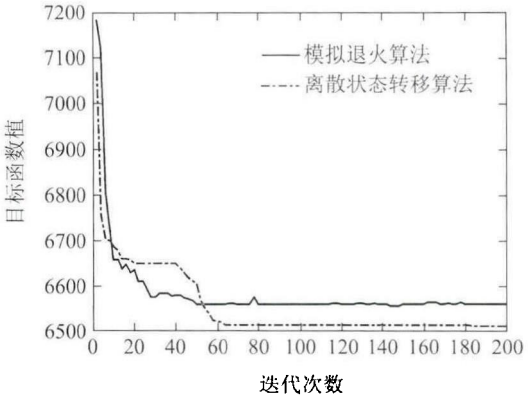

图 7 50维问题离散状态转移算法与模拟退火算法效果比较图

Fig. 7 Effect comparison between discrete state transition algorithm and simulated annealing in 50-dimension problem

表 5 离散状态转移算法与模拟退火算法效果分析

Table 5 Effect analysis of discrete transition algorithm and simulated annealing

问题维数	方 法	最好值	平均值	最差值	可行率/ %	运行时间/ s
10	DSTA	2047.0600	2047.0600	2047.0600	100	11.74
	SA	2066.1000	2071.7000	2081.3000	100	18.67
	CPLEX	2047.0600	2047.0600	2047.0600	100	775.23
15	DSTA	2531.5000	2531.5000	2531.500	100	12.23
	SA	2559.3000	2687.3000	2599.5000	100	19.88
	CPLEX	—	—	—	—	>28800
20	DSTA	1929.5000	1941.3000	1951.8000	100	13.34
	SA	1946.4000	1947.8000	1956.2000	100	21.58
	CPLEX	—	—	—	—	—
30	DSTA	3354.8000	3365.7000	33978.7000	100	15.06
	SA	3369.0000	3373.8000	3396.1000	100	26.82
	CPLEX	—	—	—	—	—
40	DSTA	4070.0000	4095.0000	4120.6000	100	17.47
	SA	4076.4000	4096.8000	4130.6000	100	33.62
	CPLEX	—	—	—	—	—
50	DSTA	6509.2000	6551.3000	6576.3000	100	19.04
	SA	6557.8000	6558.2000	6600.8000	100	41.89
	CPLEX	—	—	—	—	—

5 结论(Conclusions)

本文在离散状态转移算法基础上, 引入了二次状态转移和停滞回溯策略, 提出了一种面向员工指派问题的离散状态转移算法. 通过7个实例的仿真实验对比研究, 表明离散状态转移算法具有相对较好的稳定性和全局搜索能力, 优于经典的模拟退火算法. 将来可进一步分析停滞回溯策略中的部分参数效应, 并将其应用于工业生产的实际优化问题中, 比如有色冶金生产过程中的调度问题[25].

参考文献(References):(References):

[1] KUHN H W. The Hungarian method for the assignment problem [J]. *Naval Research Logistics*, 2005, 52(1): 7 – 21.

[2] MA Yunhong, JING Ze, ZHOU Deyun. A fast pruning optimization algorithm for task assignment problem [J]. *Journal of Northwestern Polytechnical University*, 2013, 31(1): 40 – 43.
(马云红, 井哲, 周德云. 一种任务分配问题的快速剪枝优化算法 [J]. 西北工业大学学报, 2013, 31(1): 40 – 43.)

[3] POLYMENAKOS L C, BERTSEKAS D P. Parallel shortest path auction algorithms [J]. *Parallel Computing*, 1994, 20(9): 1221 – 1247.

[4] ABBIW-JACKON R, GOLDEN B, RAGHAVAN S, et al. A divide-

and-conquer local search heuristic for data visualization [J]. *Computers and Operations Research*, 2006, 33(11): 3070 – 3087.

[5] GILMORE P C. Optimal and suboptimal algorithms for the quadratic assignment problem [J]. *SIAM Journal on Applied Mathematics*, 1962, 10(2): 305 – 313.

[6] LAND A M. A problem of assignment with interrelated costs [J]. *Operational Research Quarterly*, 1963, 14(2): 185 – 198.

[7] LAWLER E L. The quadratic assignment problem [J]. *Management Science*, 1963, 9(4): 586 – 599.

[8] BRIXIUS N W, ANSTERICHER K M. Solving quadratic assignment problems using convex quadratic programming relaxations [J]. *Optimization Methods and Software*, 2001, 16(1/2/3/4): 49 – 68.

[9] CHRISTOFIDES N, BENAVENT E. An exact algorithm for the quadratic assignment problem [J]. *Operations Research*, 1989, 37(5): 760 – 768.

[10] PADBERG M W, RINALDI G. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems [J]. *SIAM Review*, 1991, 33(1): 60 – 100.

[11] UMUT T. A new recombination operator for the genetic algorithm solution of the quadratic assignment problem [J]. *Procedia Computer Science*, 2014, 32: 29 – 36.

[12] MOHAMED S H, THOMAS S. Tabu search vs Simulated annealing as a function of the size of quadratic assignment problem instances [J]. *Computers & Operations Research*, 2014, 43: 286 – 291.

[13] UWATE Y, NISHIO Y, USHIDA A. Markov chain modeling of intermittency chaos and its application to Hopfield NN [J]. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, 2004, 87(4): 774 – 779.

[14] TANSEL D. Hybrid teaching-learning-based optimization algorithms for the quadratic assignment problem [J]. *Computers & Industrial Engineering*, 2015, 85: 86 – 101.

[15] ZHOU X J, YANG C H, GUI W H. Initial version of state transition algorithm [C] *//International Conference on Digital Manufacturing and Automation (ICDMA)*. Zhangjiajie, China: IEEE, 2011: 644 – 647.

[16] YANG Chunhua, TANG Xiaolin, ZHOU Xiaojun, et al. A discrete state transition algorithm for traveling salesman problem [J]. *Control Theory & Applications*, 2013, 30(8): 1040 – 1046.
(阳春华, 唐小林, 周晓君, 等. 一种求解旅行商问题的离散状态转移算法 [J]. 控制理论与应用, 2013, 30(8): 1040 – 1046.)

[17] ZHOU X J, YANG C H, GUI W H. State transition algorithm [J]. *Journal of Industrial and Management Optimization*, 2012, 8(4): 1039 – 1056.

[18] ZHOU X J, GAO D Y, YANG C H. A Comparative study of state transition algorithm with harmony search and artificial bee colony [J]. *Advances in Intelligent Systems and Computing*, 2013, 212: 651 – 659.

[19] ZHOU X J, YANG C H, GUI W H. Nonlinear system identification and control using state transition algorithm [J]. *Applied Mathematics and Computation*, 2014, 226: 169 – 179.

[20] DONG T X, ZHOU X J, YANG C H, et al. A discrete state transition algorithm for the task assignment problem [C] *//The 34th Chinese Control Conference (CCC)*. Hangzhou, China: IEEE, 2015: 2692 – 2697.

[21] ZHOU X J, GAO D Y, YANG C H, et al. Discrete state transition algorithm for unconstrained integer optimization problems [J]. *Neurocomputing*, 2016, 173: 864 – 874.

[22] ZHOU X J, GAO D Y, SIMPSON A R. Optimal design of water distribution networks by discrete state transition algorithm [J]. *Engineering Optimization*, 2016, 48(4): 603 – 628.

[23] MISEVIČIU A. A modified simulated annealing algorithm for the quadratic assignment problem [J]. *Informatica*, 2003, 14(4): 497 – 514.

[24] http://www.mathworks.com/matlabcentral/fileexchange/52499-discrete-state-transition-algorithm-for-traveling-salesman-problem/

[25] ZHOU Xiaojun, YANG Chunhua, GUI Weihua. Modeling and control of nonferrous metallurgical processes on the perspective of global optimization [J]. *Control Theory & Applications*, 2015, 32(9): 1158 – 1169.
(周晓君, 阳春华, 桂卫华. 全局优化视角下的有色冶金过程建模与控制 [J]. 控制理论与应用, 2015, 32(9): 1158 – 1169.)

作者简介:

董天雪 (1992–), 女, 硕士, 主要研究方向为优化算法及其应用, E-mail: tianxue@csu.edu.cn;

阳春华 (1965–), 女, 教授, 博士生导师, 主要研究方向为复杂工业建模与优化控制、在线检测技术及自动化装置, E-mail: ychh@csu.edu.cn.

周晓君 (1986–), 男, 博士, 讲师, 主要研究方向为复杂工业过程建模、优化与控制、对偶理论、最优化理论、算法及其应用, E-mail: michael.x.zhou@csu.edu.cn.

桂卫华 (1950–), 男, 中国工程院院士, 教授, 博士生导师, 主要研究方向为工业大系统递阶和分散控制理论及应用、复杂工业过程建模、优化与控制, E-mail: gwh@csu.edu.cn.