文章编号: 1001-0920 (2010) 06-0889-05

# 一种求解QAP问题的混合嵌套分区优化算法

武 维, 卫军胡, 管晓宏

(西安交通大学 a. 制造系统国家重点实验室, b. 智能网络与网络
安全教育部重点实验室, c. 系统工程研究所, 西安 710049)

**摘 要:** 提出一种基于嵌套分区算法 (NPM) 框架求解二次分配问题 (QAP) 的混合优化算法. 算法利用嵌套分区树来描述二次分配过程, 对可行域进行系统性分区, 采用禁忌抽样算子对分区进行抽样并评估各个分区的性能. 在每次迭代中, 算法重点跟踪和搜索优良解最有希望出现的分区, 并结合禁忌搜索算法来实现分区转移. 数值仿真实验表明, 引入更加有效的禁忌抽样算子后, NPM 算法具有更好的寻优能力.

**关键词:** 嵌套分区算法; 二次分配问题; 禁忌搜索算法; 组合优化

**中图分类号:** TP391 **文献标识码:** A

# A hybrid nested partitions optimization algorithm for the QAP problem

*WU Wei*, *WEI Jun-hu*, *GUAN Xiao-hong*

(a. State Key Laboratory of Manufacturing System Engineering, b. Key Laboratory for Intelligent Network Security of Ministry of Education, c. Systems Engineering Institute, Xi'an Jiaotong University, Xi'an 710049, China. Correspondent: WU Wei, E-mail: mr.allawn@gmail.com)

**Abstract:** This paper proposes a hybrid optimization algorithm based on nested partitions method(NPM) framework for solving quadratic assignment problem(QAP). In the algorithm, the QAP is described as a nested partitions tree. The algorithm systematically partitions the feasible region and uses tabu search sampling operator to evaluate the performance of each subregion. In the each iteration, the algorithm focused on the most promising region for searching the optimal solutions. And the tabu search is incorporated into the sampling procedure to make the subregion correct move. The results of numerical experiments show that the hybrid algorithm has better performance.

**Key words:** Nested partitions algorithm; Quadratic assignment problem; Tabu search; Combinatorial optimization

## 1 引 言

2000 年 Shi 和 Olafsson 等[1]提出了嵌套分区 (NPM) 算法, 该算法结合了全局优化和局部优化, 具有开放性、并行性和易操作性等突出优点. NPM 有着坚实的数学基础, 能够解决许多确定性和随机性的仿真优化问题, 而且具有很高的计算效率. 近几年, 嵌套分区算法被应用于柔性资源调度、机器分配、供应链管理、客户关系管理、医疗保健等大规模随机优化问题, 取得了显著效果[2-4]. Al-Shihabi 在嵌套分区算法抽样过程中结合蚂蚁算法, 较好地解决了 TSP (Traveling salesman problem) 问题. Olafsson 和 Shi[5] 在嵌套分区算法中应用序比较方法提高了算法的收敛速度. Chalermdamrichai 和 Dharmaraj 等将嵌套分区算法应用于 CNC (Computer numerical control) 工序优化问题, 取得了显著效果. 路晓伟等[6]将嵌套分区算法与模拟退火算法结合起来, 得到了比模拟退火算法更好的结果.

二次分配问题 (QAP) 是一个重要的理论和实际问题. 许多问题均可形式化为二次分配问题, 例如集成电路布线、作业调度问题等. 二次分配问题属 NP 难问题, 通常, 当问题规模 $n$ >20 时很难找到最优解. 本文针对二次分配问题, 在 NPM 的基础上提出一种 TSNP (Tabu search nested partitions) 算法. 该算法结合 NPM 算法的全局搜索能力和 Tabu search 算法的局部搜索能力, 提高了算法的优化效率. 目前, 国内关于嵌套分区算法的文献还较少, 本文将采用嵌套分区

**收稿日期:** 2009-06-09; **修回日期:** 2009-08-28.

**基金项目:** 国家自然科学基金项目(60736027, 60704033); 国家 863 计划项目(2007AA04Z154); 陕西省自然科学基础研究计划项目(2007F41).

**作者简介:** 武维(1982–), 男, 山西太谷人, 博士生, 从事复杂系统的建模、仿真与优化调度的研究; 管晓宏(1955–), 男, 四川江安人, 教授, 博士生导师, 从事系统优化调度、计算机网络信息安全等研究.

算法来求解QAP的优化调度问题. 最后通过实验证明算法的计算效率和可行性.

## 2 二次分配问题描述

二次分配问题可以描述为将一个设备集合分配到一个节点集合的过程. 其中给定了节点之间的距离和设备之间的信息流, 要求按照某种规则把设备分配给节点集合, 使得信息流与距离乘积的总和最小[7].

QAP问题的数学描述如下: 令$n$为设备集合及节点集合的大小, $a_{ij}$为节点$i$与$j$之间的距离, $b_{rs}$为设备$r$与$s$之间的信息流量, 要求把每个设备分配给一个节点. 令$\theta$是一个分配方案, 表示各个节点所分配设备的一个排序. 定义目标函数为

$$f(\theta)=\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ij}b_{\theta(i)\theta(j)}, \tag{1}$$

其中$\theta(i)$和$\theta(j)$分别表示分配给节点$i$和$j$的设备. 系统优化的目标是找到一个置换排列$\theta$, 使目标函数值最小, 即$\min f(\theta)$.

## 3 嵌套分区算法

在NPM框架下, 上述优化问题可以描述为如下形式:

$$\min_{\theta\in\Theta}J(\theta). \tag{2}$$

其中: $\Theta$是可行域, $J:\Theta\to R$是系统的性能函数. 假设该优化有唯一最优解$\theta_{\text{opt}}\in\Theta$, 对于所有的$\theta\in\Theta\backslash\{\theta_{\text{opt}}\}$有$J(\theta_{\text{opt}})<J(\theta)$. 在复杂优化问题中, $J(\theta)$通常是对输入参数的性能估计. 但在某些情况下, 并没有解析表达式用以表达系统的性能函数. 此时, 采用仿真的方法$J(\theta)=E[L(\theta)]$来估计系统的性能. 其中$L(\theta)$是对系统性能的无偏估计, 即系统抽样的性能. NPM算法对系统性能函数的形式没有要求. NPM算法的基本思想如图1所示.

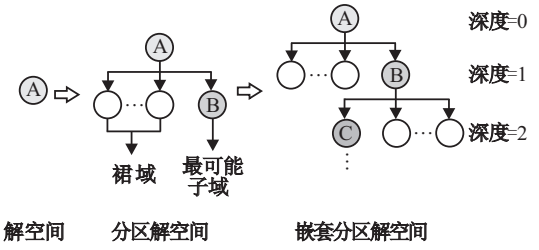


**图 1 NPM基本思想**

为了说明嵌套分区算法, 首先给出如下定义:

**定义 1**[8](单解域) 如果一个集合中所有解都是可行的, 则该集合是一个可行域. 所有可行域的集合表示为$\Sigma$. 只包含一个可行解的区域称为单解域. 单解域的集合表示为$\Sigma_0$, $\Sigma_0\subset\Sigma$.

**定义 2**[8](裙域) 在迭代过程中对可行域$\sigma(k)$进行分区, 并将$\sigma(k)$外的其他分区合并为一个区域$\Theta\backslash\sigma(k)$, 称为裙域.

**定义 3**[8](母域) 对可行域$\eta\in\Sigma$分区, 得到可行域$\sigma\in\Sigma$, 则称$\sigma$为$\eta$的子域, $\eta$为$\sigma$的母域. 定义母域映射函数$s:\Sigma\to\Sigma$, 即如果$\sigma\in\Sigma\backslash\{\Theta\}$, 则$s(\sigma)=\eta\in\Sigma$. 对于$\Theta$, 定义$s(\Theta)=\Theta$.

**定义 4**[8](深度) 定义$d:\Sigma\to N_0$为区域$\Sigma$的分区深度, 即由区域$\Theta$到区域$\Sigma$的分区层数. 例如, $\Theta$的深度为0, 即$d(\Theta)=0$, 则其子域深度为1, 依此类推. 单解域具有最大深度.

**定义 5**[8](希望指数) 设定希望指数$I:\Sigma\to R$, 表示$\theta_{\text{opt}}$出现在某个分区的可能性大小, 用来确定下一步要划分和搜索的区域. NPM具有开放性, 可以根据具体问题设定希望指数的形式.

NPM的基本思想是: 在算法的第$k$次迭代中, 如果认为$\sigma(k)\subseteq\Theta$是包含$\theta_{\text{opt}}$的可行域, 则将$\sigma(k)$划分为$M_{\sigma(k)}$个子域, 并将$\Theta\backslash\sigma(k)$定义为裙域. 这样便得到$M_{\sigma(k)}+1$个不相交的子域(初次分区除外). 在每个区域上先利用抽样方法估计该区域的希望指数, 确定最优解最有可能出现的区域; 然后再对该区域进一步分区, 直到得到单解域. NPM分为4个步骤: 分区、抽样、选区和回溯[9]. 具体如下:

1) 分区. 如果$\sigma(k)\notin\Sigma_0$, 则将$\sigma(k)$划分为$M_{\sigma(k)}$个子域$\sigma_1(k),\cdots,\sigma_{M_{\sigma(k)}}(k)$, 并将$\Theta\backslash\sigma(k)$合为一个区域$\sigma_{M_{\sigma(k)}+1}(k)$, 即产生$M_{\sigma(k)}+1$个区域. 但是, 如果$\sigma(k)=\Theta$, 则只分区产生$M_{\sigma(k)}$个区域.

2) 抽样. 对区域$\sigma_j(k)$选择$N_{\sigma_j(k)}$个抽样点$\theta^{j_1}$, $\theta^{j_2},\cdots,(\theta^{jN_{\sigma_j(k)}})$, 根据这些抽样点估计该分区的性能值分别为$L(\theta^{j_1}),L(\theta^{j_2}),\cdots,L(\theta^{jN_{\sigma_j(k)}})$. 算法可采用多种抽样方法, 只要满足区域中每个点被选择的概率大于0即可.

3) 选区. 根据希望指数$I:\Sigma\to R$确定下一步要分区的区域.

4) 回溯. 如果区域$\sigma_j(k)\subset\Theta\backslash\sigma(k)$是要被分区的区域, 则算法将要回溯. 目前有两种回溯策略: 1)回溯到当前域的母域, 即$\sigma(k+1)=s(\sigma(k))$; 2)回溯到初始$\Theta$域, 即$\sigma(k+1)=\Theta$.

## 4 NPM框架下的禁忌搜索策略

禁忌搜索(TS)算法属于一种启发式算法, 由Fred Glover在1986年提出. TS的基本思想是: 随机给出一个初始解作为当前解, 在当前解的邻域中确定若干个候选解. 若最佳候选解对应的目标值优于当前解, 则用其代替当前解, 并将其加入禁忌表, 同时更新禁忌表; 如果不存在上述解, 则选择候选解中的非禁忌的最佳状态为当前解, 同样将其加入禁忌表, 并更新禁忌表. 如此重复上述迭代搜索过程, 直到满足终

止条件.

本文在NPM框架下的禁忌搜索算法步骤如下[10,11]:

**Step 1**　随机生成一个开始抽样点 $\theta_j^0$. 初始化 $n=0$, 定义当前解为 $\theta^n$, 当前发现的最优解为 $\theta^{\text{best}}$, 并初始化 $\theta^{\text{best}}=\theta^n=\theta_j^0$. 定义禁忌列表为 $L(n)$, 并初始化为空, 即 $L(n)=\varnothing$.

**Step 2**　针对当前可行解生成邻域解 $N(\theta^n)$. 搜索所有可行的邻域解, 并计算目标函数找到最优的非禁忌解 $\tilde{\theta}^{\text{nontabu}}$ 和最优的禁忌解 $\tilde{\theta}^{\text{tabu}}$.

**Step 3**　根据不同情况, 进行如下处理:

1) 如果 $f(\theta^{\text{best}}) \leqslant \min\{f(\tilde{\theta}^{\text{nontabu}}), f(\tilde{\theta}^{\text{tabu}})\}$, 则令 $\theta^{n+1}=\tilde{\theta}^{\text{nontabu}}$, 并将 $\theta^n=\tilde{\theta}^{\text{nontabu}}$ 添加到禁忌列表中, 即

$$L(n+1)=L(n)\bigcup\{\tilde{\theta}^{\text{nontabu}}\to\theta^n\}.$$

2) 如果 $f(\tilde{\theta}^{\text{nontabu}}) \leqslant \min\{f(\theta^{\text{best}}), f(\tilde{\theta}^{\text{tabu}})\}$, 则令 $\theta^{n+1}=\tilde{\theta}^{\text{nontabu}}$, $\theta^{\text{best}}=\theta^{n+1}$, $f^{\text{best}}=f(\theta^{n+1})$. 并将 $\theta^n=\tilde{\theta}^{\text{nontabu}}$ 添加到禁忌列表中, 即

$$L(n+1)=L(n)\bigcup\{\tilde{\theta}^{\text{nontabu}}\to\theta^n\}.$$

3) 如果 $f(\tilde{\theta}^{\text{tabu}}) < \min\{f(\theta^{\text{best}}), f(\tilde{\theta}^{\text{nontabu}})\}$, 则令 $\theta^{n+1}=\tilde{\theta}^{\text{tabu}}$, $\theta^{\text{best}}=\theta^{n+1}$, $f^{\text{best}}=f(\theta^{n+1})$, 并将 $\theta^n=\tilde{\theta}^{\text{tabu}}$ 添加到禁忌列表中, 即

$$L(n+1)=L(n)\bigcup\{\tilde{\theta}^{\text{tabu}}\to\theta^n\}.$$

如果禁忌列表长度超过最大值, 则依次从禁忌列表中删除旧数据.

**Step 4**　如果停止准则满足, 则返回 $\theta^{\text{best}}$, 并作为下个可行抽样解; 否则, 令 $n=n+1$, 转 Step 2.

对每个子域 $\sigma_j(k)$ 重复禁忌搜索 $N_j$ 次后, 每个子域中有 $N_j$ 个抽样点, 即

$$\theta_1^j,\theta_2^j,\cdots,\theta_{N_j}^j,\ \ j=1,2,\cdots,M_{\sigma(k)}+1,$$

计算相应的希望指数

$$f(\theta_1^j),f(\theta_2^j),\cdots,f(\theta_{N_j}^j),\ \ j=1,2,\cdots,M_{\sigma(k)}+1.$$

最后根据相应的希望指数进行选区或回溯流程.

本文结合NPM的全局搜索能力和TS算法的局部搜索能力求解QAP, 并将TS算法用于NPM的分区和抽样算子中. 新算法用TSNP表示.

## 5　优化算法流程详细描述

求解QAP问题的TSNP算法整体流程如图2所示.

### 5.1　分区算子

用嵌套分区算法求解QAP优化问题的过程就是构造分区树的过程, 在分区树每一层分配一个设备到一个节点. 这样在深度为 $d$ 的分区上有 $d$ 个设备已经被分配, $n-d$ 个设备没有被分配. 一个完整的分配方案可以表示为 $\theta=(U)=\{u_1,u_2,\cdots,u_n\}$. 图3表明了设备数 $n=5$ 的分区树构建过程.

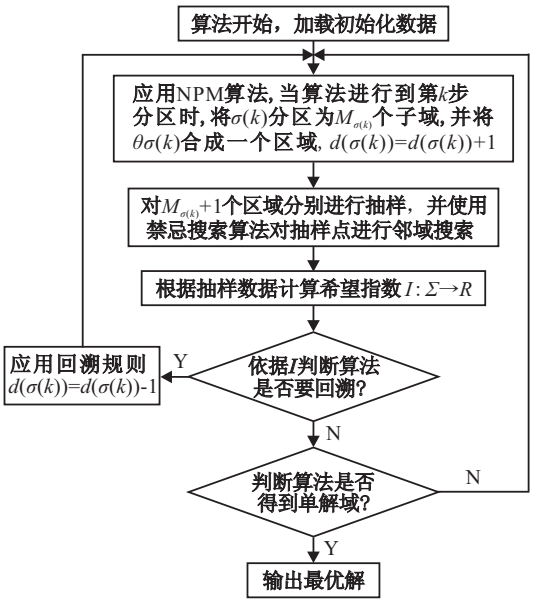


**图 2　TSNP 算法流程图**

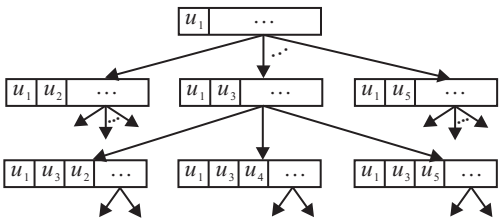

**图 3　设备数 $n=5$ 的分区树构建过程**

分区算子在分区树的每个深度分配一个设备, 当设备数为 $n$ 时, $d=n$. 深度为0的子域定义为

$$\{(u_1),(\cdot),\cdots,(\cdot)\},$$
$$\{(u_2),(\cdot),\cdots,(\cdot)\},$$
$$\vdots$$
$$\{(u_n),(\cdot),\cdots,(\cdot)\}.$$

深度为 $d$ 的子域与深度为0的子域表示相类似, 区别仅在于前者的分区已经固定了 $d$ 个设备. 分区树上子域数量随着深度的增加而逐渐减少, 其数学表达式为 $\left|N_j^U\right|$.

### 5.2　禁忌抽样算子

QAP抽样算子用来确定下一步要被分配的设备. 一个有效的抽样算子应保证分区中的每个设备都有被选中的可能. 如果在区域 $\sigma$ 中抽样, 已有 $d(\sigma)$ 个设备被分配, 则需要抽样剩下的 $n-d(\sigma)$ 个设备, 并生成抽样点 $\theta=\{u_1,u_2,\cdots,u_n\}$.

本文使用禁忌搜索算法作为抽样算子. 禁忌搜索抽样算子涉及到邻域、禁忌表、禁忌长度、特赦规则等具体操作.

1) 邻域. 设 $\theta=(U)=\{u_1,u_2,\cdots,u_n\}$ 是区域 $\sigma$ 中的一个抽样点. 交换分区 $\sigma$ 中相邻两个节点上的设备 $\theta_r$ 和 $\theta_s$, 并保证 $d(\sigma)\leqslant d(\theta_r)<d(\theta_s)\leqslant n$, 这样就得到了 $\theta$ 的一个邻域解 $\pi$.

数学描述如下:

$$\begin{aligned}&\pi(k)=\theta(k),\ \ \forall k\neq r,s;\\&\pi(r)=\theta(s);\\&\pi(s)=\theta(r).\end{aligned}\tag{3}$$

如果距离矩阵 $a$ 及信息矩阵 $b$ 为对称阵且对角线元素为 0, 则在交换设备 $r$ 及 $s$ 后所产生的目标函数的差值 $\Delta(\theta,r,s)$ 可以在 $O(n)$ 时间内得到, 即

$$\begin{aligned}&\Delta(\theta,r,s)=\\&\sum_{i=1}^{n}\sum_{j=1}^{n}(a_{ij}b_{\theta(i)\theta(j)}-a_{ij}b_{\pi(i)\pi(j)})=\\&2\sum_{k\neq r,s}(a_{sk}-a_{rk})(b_{\theta(s)\theta(k)}-b_{\theta(r)\theta(k)}).\end{aligned}\tag{4}$$

特别在局部搜索中, 利用前一次置换信息, 特定交换产生的目标函数差值可以被更快地计算出来. 设 $\pi$ 为交换解 $\theta$ 中设备 $r$ 和 $s$ 得到的解, 则交换解 $\pi$ 中设备 $u$ 和 $v(\{u,v\}\cap\{r,s\}=\varnothing)$ 所产生的目标函数差值可在常数时间内得到, 即

$$\begin{aligned}&\Delta(\pi,u,v)=\\&\Delta(\theta,u,v)+2(a_{ru}-a_{rv}+a_{sv}-\\&a_{su})(b_{\pi(s)\pi(u)}-b_{\pi(s)\pi(v)}+\\&b_{\pi(r)\pi(v)}-b_{\pi(r)\pi(u)}).\end{aligned}\tag{5}$$

这样循环扫描一遍邻域只需要 $o(n^2)$ 时间, 大大提高了邻域搜索的效率.

2) 禁忌列表. 禁忌表使用二维数组构建, 表示为 $\mathrm{TA}=[\mathrm{TA}_{ij}]$, $i=1,2,\cdots,n$, $j=1,2,\cdots,n$. 定义 $\mathrm{TA}_{ij}$ 为上次节点 $i$ 和 $j$ 发生置换后, 最后的迭代数; $s$ 为禁忌列表的长度; $k$ 为当前迭代数. 如果在当前迭代中节点 $i$ 和 $j$ 发生置换后, $\mathrm{TA}_{ij}+s>k$, 则该置换被禁忌; 否则, 通过置换计算目标函数差值 $\Delta$, 修改禁忌表 $\mathrm{TA}_{ij}=k$. 表 1 给出了节点数为 5 的禁忌列表.

**表 1 节点数为 5 的禁忌列表**

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | | 1 | 5 | 7 | 3 |
| 2 | | | 6 | 10 | 8 |
| 3 | | | | 15 | 9 |
| 4 | | | | | 4 |
| 5 | | | | | |

禁忌列表是对称阵, 因此只需要记录上对角阵, 其中单元格数据 $\mathrm{TA}_{ij}(j>i)$ 表示节点 $i$ 和 $j$ 发生置换后, 最后的迭代数.

3) 特赦规则. 为防止算法进入局部搜索, 算法中采用了特赦规则. 如果候选解 $\pi$ 比当前解 $\theta$ 的目标函数值小, 但它是被禁忌的, 则将其解禁出来.

4) 禁忌长度. 禁忌长度 $s$ 的选择非常重要. $s$ 太大, 则大量候选解被禁忌, 需要大量迭代寻找希望的候选解, 从而大大加长了抽样时间. 随着 NPM 分区的不断进行, 分区大小逐渐缩小, 禁忌长度 $s$ 也应随着改变[12]. 定义

$$\begin{aligned}&s\in[s_{\min},s_{\max}],\\&s_{\min}=[0.9(n-d)],\\&s_{\max}=[1.1(n-d)].\end{aligned}\tag{6}$$

其中: $n$ 为节点数目, $d$ 为分区深度. 当分区深度发生变化时, 禁忌长度 $s$ 也相应调整, 因而提高了算法的搜索效率.

### 5.3 选区算子

用希望指数 $I:\Sigma\to R$ 来评定下一步要选择的分区. 在本例中希望指数与目标函数一致. 对于每个分区 $\sigma_j$, $j=1,2,\cdots,M_{\sigma(k)}+1$, 计算相应分区的性能指标, 即

$$\begin{aligned}&I(\sigma_j)=\min_{i=1,2,\cdots,N_j}f(x_i^j),\\&j=1,2,\cdots,M_{\sigma(k)}+1.\end{aligned}\tag{7}$$

选择希望指数最小的分区作为下一步要分区的区域. 定义 $N_k(\sigma)$ 为每个分区中单解域被抽样到的次数. 于是在每次迭代中将生成平稳的分布

$$\mu_k(\sigma)=\frac{N_k(\sigma)}{k},\ \ \sigma\in\Sigma.\tag{8}$$

该分布过程是指数收敛的. 文献 [5] 已经证明, 在一定条件下 NPM 算法是 Markov 收敛的.

### 5.4 回溯策略

当算法经过 $k$ 次分区后, 如果当前分区 $\sigma(k)$ 以外的整个区域希望指数是最小的, 则回溯到 $\sigma(k)$ 的母域 $s(\sigma(k))$, 对 $s(\sigma(k))$ 进行分区、抽样和选区的操作. 回溯时, 将已分区节点数减 1, 深度减 1 即可.

### 5.5 禁忌抽样算子对 NPM 算法的作用

典型的优化方法包括两个阶段: 搜索阶段和迭代阶段. 将足够的时间花在初始搜索阶段, 可减少迭代过程所需时间. 特别在复杂 DEDS 的仿真过程中, 由于事件的离散性、参数的不确定性以及事件发生的随机性, 精确求解是很难的. 将 NP 算法应用于这类问题时, 由于可行域较大, 需要经过较多的分区、抽样、回溯才能得到优解. 特别是回溯步骤, 会对仿真效率造成较大影响. 当算法进行到第 $k$ 步时, 如果裙域 $\Theta\backslash\sigma(k)$ 是最可能域, 则要回溯到 $\sigma(k)$ 的母域 $s(\sigma(k))$, 这样需要多抽样 $2k(M_{\sigma(k)}+1)$ 个点 ($k$ 为分区下抽样的点数). 随着可行域深度的不断增加, 分

区及回溯次数也将不断增加, 从而影响了算法的收敛速度. 利用禁忌抽样算子, 可保证得到新可行域以一致性概率包含最优解或次优解. 这样 NP 算法的回溯过程大大减少, 提高了算法的收敛速度.

为了避免算法陷入局部最优解, 算法引入回溯概念. 在算法的第 $k$ 次迭代中, 判断裙域是否为下一步要被分区的区域, 如果是, 则算法回溯. 但随着分区深度不断增加, 裙域变得越来越大. 若对裙域抽样较少, 则可能发现不了裙域中的优解, 从而算法有可能陷入局部最优; 若对裙域抽样过多, 则仿真计算消耗太大. 而通过禁忌抽样算子, 则可有效地将裙域中已经抽样的点禁忌, 从而减少仿真计算时间, 提高算法的收敛效率.

## 6　仿真实验及结果分析

为了表明嵌套分区算法对 QAP 问题的寻优能力, 本文从通用 QAPLIB 中选用典型的 Benchmark[13]实例进行数值仿真计算. 设计两组实验(实验 1 和实验 2), 分别在 Chr 和 Tai 标准测试问题集上对算法进行验证. 仿真程序使用 VC++.net 2005 编写, 以 Intel Celeron 1.6 Hz 计算机为运行环境进行实验, 仿真实验运行 5 次, 结果取平均值. 仿真实验策略见表 2.

**表 2　实验策略**

| 版本 | 算法 | 描　述 |
|---|---|---|
| A | NPM | 原始NPM算法框架 |
| B | TS | 禁忌搜索算法 |
| C | TSNP | NPM算法加入禁忌搜索抽样算子 |

**表 3　实验 1 的结果**

| 问题 | Opt* | RE | | | Time/s |
|---|---|---|---|---|---|
| | | NPM | TS | TSNP | |
| Chr12a | 9 552 | 55.50 | 0 | 0 | 0.2 |
| Chr15a | 9 896 | 48.50 | 0 | 0 | 0.5 |
| Chr18a | 11 098 | 16.39 | 0 | 0 | 1 |
| Chr20a | 2 192 | 124.08 | 1.368 6 | 0 | 2 |
| Chr22a | 6 156 | 66.95 | 0 | 0 | 3 |
| Chr25a | 3 796 | 38.04 | 0.684 9 | 0 | 4 |

**表 4　实验 2 的结果**

| 问题 | Opt* | RE | | | Time/s |
|---|---|---|---|---|---|
| | | NPM | TS | TSNP | |
| Tai12a | 224 416 | 25.41 | 0 | 0 | 0.2 |
| Tai15a | 388 214 | 14.68 | 0 | 0 | 0.6 |
| Tai20a | 700 384 | 18.88 | 0.747 8 | 0.442 3 | 5 |
| Tai25a | 1 167 256 | 15.34 | 0.868 0 | 0 | 10 |
| Tai30a | 1 818 146 | 11.81 | 0.176 8 | 0 | 13 |
| Tai35a | 2 422 022 | 12.50 | 0.081 1 | 0 | 19 |
| Tai40a | 3 139 370 | 14.17 | 0.713 3 | 0.406 7 | 25 |
| Tai50a | 4 941 410 | 13.70 | 1.349 0 | 0.992 1 | 40 |
| Tai60a | 7 208 572 | 13.32 | 1.902 7 | 1.032 4 | 59 |
| Tai80a | 13 557 864 | 11.56 | 1.069 5 | 0.799 0 | 106 |
| Tai100a | 21 125 314 | 6.621 | 1.380 1 | 0.886 6 | 175 |

表 3, 表 4 给出了实验 1 和实验 2 的仿真结果. 在表 3 和表 4 中, Opt* 为该问题的已知最优解, Time 为计算时间 (单位为 s), RE 表示优化结果与 Opt* 差值的平均相对百分比.

在表 3 和表 4 的实验结果中, 原始 NPM 算法的 RE 结果远远偏离最优解. 可见, 如果在 NPM 框架中不引入较好的局部搜索策略, 则很难收敛到最优解.

图 4, 图 5 是在不同条件下采用 NP 算法寻优效率的对比. 其中: 图 4 为实验 1 的结果比较, 图 5 为实验 2 的结果比较.

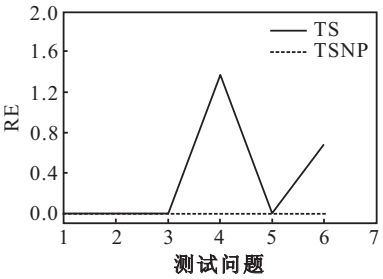


**图 4　实验 1 的结果比较**

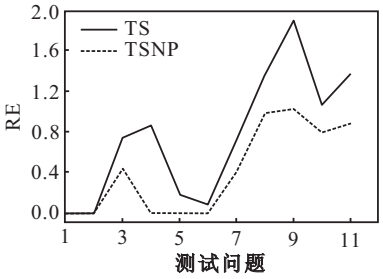


**图 5　实验 2 的结果比较**

## 7　结　论

本文在嵌套分区 (NPM) 算法框架下结合禁忌搜索算法, 提出了一种解决二次分配问题的新型算法. 实验结果表明, 在原始 NPM 算法中加入禁忌搜索抽样算子后, NPM 算法的收敛能力有了大幅提升, 对于一些小规模问题可以求得最优解. 新算法结合了 NPM 的全局搜索能力和 TS 算法的局部搜索能力, 获得了比 NPM 算法和 TS 算法更优的解. TSNP 算法可以方便地应用于 NP 难的组合优化问题的求解.

### 参考文献(References)

[1] Shi L, Olafsson S. Nested partitions method for global optimization[J]. Operations Research, 2000, 48(3): 309-407.

[2] Olafsson S, Shi L. A method for scheduling in parallel manufacturing systems with flexible resources[J]. IIE Trans Research, 2000, 32(1): 135-146.

[3] Shi L, Olafsson S, Chen Q. An optimization framework for product design[J]. Management Science, 2001, 47(2): 1681-1692.

(下转第898页)