ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.12, December 2006, pp.2589−2600
DOI: 10.1360/jos172589

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

并行机间歇过程生产调度的遗传局部搜索算法*

苏 生+, 战德臣, 徐晓飞

(哈尔滨工业大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

A Genetic Local Search Algorithm for the Parallel Machine Batch Process Scheduling Problem

SU Sheng+, ZHAN De-Chen, XU Xiao-Fei

(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

+ Corresponding author: Phn: +86-451-86414906, E-mail: susheng@hit.edu.cn, http://www.hit.edu.cn

Su S, Zhan DC, Xue XF. A genetic local search algorithm for the parallel machine batch process scheduling problem. *Journal of Software*, 2006,17(12):2589−2600. http://www.jos.org.cn/1000-9825/17/2589.htm

Abstract: A parallel machine batch process scheduling problem (PBPSP) integrating batching decision is investigated. The problem is converted into the fixed charge transportation problem (FCTP). A genetic local search algorithm (GLSA) with intensification strategy of local search and escape strategy from local optimal solution is developed. The sorted edges attained by root-first search of spanning tree are used to encode spanning tree in the genetic local search algorithm. Efficient single point crossover operator appending edges to sub-tree is proposed. Network simplex method based local search is used to be the mutation of individual. To enhance the capacity of searching the global optimal solution, this paper presents an intensification strategy of local search that applies continuous random node local search to the current optimal solution and an escape strategy from local optimal solution based on random pivot mutation and random node local search. The results of computations demonstrate that the genetic local search algorithm is better than the permutation encoding genetic algorithm and the matrix encoding genetic algorithm on solution quality, and can find the optimal solution of all Benchmark problems. Moreover, the genetic local search algorithm is robust. Compared with the tabu heuristic search procedure, this algorithm can obtain more frequently the optimal solutions of the test problem instances.

Key words: batch process; scheduling; fixed charge transportation problem; spanning tree; genetic algorithm; local search

摘 要: 研究了一类集成分批的并行机间歇过程调度问题(parallel machine batch process scheduling problem,简称 PBPSP),将此问题转化为固定费用运输问题(fixed charge transportation problem,简称 FCTP)后,提出了具有集中邻域搜索机制和局部最优逃逸机制的遗传局部搜索算法(genetic local search algorithm,简称 GLSA).GLSA 算法用

* Supported by the National Natural Science Foundation of China under Grant No.60573086 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2003AA4Z3210 (国家高技术研究发展计划(863)); the National Research Foundation for the Doctoral Program of Ministry of Education of China under Grant No.20030213027 (国家教育部博士点基金)

Received 2005-08-28; Accepted 2006-03-07

先根遍历边排列模式编码生成树解,具有高效的子树补充式单点交叉操作.将基于网络单纯型方法的邻域搜索作为变异算子,并提出了连续随机节点邻域搜索的集中邻域搜索策略以及随机旋转变异与全局邻域搜索相结合的局部最优逃逸策略,极大地强化了遗传局部搜索算法的全局寻优能力.实验表明:GLSA 算法获得的解质量优于基于排列编码的遗传算法和基于矩阵编码的遗传算法,得到了所有 Benchmark 问题的最优解,且具有高鲁棒性.针对一定规模的 FCTP 问题,GLSA 算法比 Tabu 启发式搜索算法具有更高的获得最优解几率.

关键词: 间歇过程;调度;固定费用运输问题;生成树;遗传算法;局部搜索
中图法分类号: TP316 文献标识码: A

在化工、食品以及包装等多种生产环境中存在着大量间歇过程,间歇过程自身所具有的设备处理柔性、批量有限性和不可忽略的产品切换时间等特点,使得间歇过程生产调度比一般生产调度(如离散制造生产调度)更难以处理,它不仅需要确定任务与设备之间的分配关系以及任务在设备上的处理顺序,而且需要确定任务应该划分的批次数目以及每个批次的批量等[1].并行机间歇过程是间歇过程的一种,其生产环境由若干相互独立的并行间歇过程处理设备组成,每个设备能够完成多种任务.针对并行机间歇过程生产调度的研究较多[2],但这些研究存在两方面不足.一方面是将求解过程分两步进行:其中第 1 步是需求分批,即对大数额的产品计划需求分割或将小数额的产品计划需求合并为一系列可通过间歇设备一次性处理完成的多个批次[3,4];第 2 步是类似于作业族调度的批次调度,即对第 1 步产生的那些批次进行调度,确定每个批次通过系统的精确时间,如批次的设备资源分配和批次排序,其中每种产品的所有批次都可以看成一个作业族,一个作业族的所有批次可组合为若干批次组,一个批次组中的所有批次共享一次设备设置操作,调度的任务还需要确定批次组的个数和每个批次组包含的具体批次[3−6].虽然这种两步递阶的方法简化了问题求解难度,但其结果不精确甚至不正确.另一方面是求解方法多局限于分枝定界等精确解法,研究重点在于如何建立好的 MILP 或 MINLP 模型以削减模型的整数变量数目,虽然这些方法能够在一定程度上解决并行机间歇过程调度问题,但当问题规模增加时,这些方法花费的求解时间非常巨大,难以应用到实际生产过程中.

为克服现有缺点,我们尝试同时进行需求分批与批次调度,并采用 meta 启发式方法获取问题的最优解或近优解以求解大规模问题.本文考虑了一类并行机间歇过程生产调度问题(parallel machine batch process scheduling problem,简称 PBPSP),在分析 PBPSP 问题的几个性质后,将 PBPSP 问题转化为一个固定费用运输问题(fixed charge transportation problem,简称 FCTP),然后重点讨论求解固定费用运输问题的 meta 启发式方法.

固定费用运输问题是一个 NP-hard 问题[7],其最优解出现在由问题约束定义的凸集的某个极点上,即最优解是运输图的一棵生成树.求解固定费用运输问题最常用的方法是将其作为一个混合整数网络规划问题来求解.一些精确算法用于求解固定费用运输问题,如极点排列法[7]和分枝定界法[8]等,但由于这些方法没有利用固定费用运输问题特殊的网络结构,所以这些方法被证明既低效且计算耗时,只适合求解小规模问题.为了克服计算时间过长的问题,一些启发式方法用于求解固定费用运输问题,如 Lagrangian 松弛法、邻接极点法[9]等,虽然这些启发式方法计算得比较高效,但主要缺点是获得的解的质量较差,在实际应用中会耗用很多不必要的费用.

近年来,一些 meta 启发式方法用于求解固定费用运输问题:Michalewicz 提出了基于排列编码的遗传算法[10];Gen 提出了基于 Prüfer 数编码的遗传算法[11,12];Gottlieb 提出了基于矩阵编码的遗传算法[13];Sun 提出了 Tabu 启发式搜索方法[14].Prüfer 数可以非常简洁地表达生成树,用一个 n–2 位代表节点标号的数字组成的位串即可代表一棵唯一的 n 节点树,但文献[15]证明:在用 Prüfer 数编码生成树的遗传算法中,单点交叉的遗传性(heritability)很低,且其变异操作的区域性(locality)也很低,当问题规模较大时,Prüfer 数编码方法很难获得最优解或近似最优解,其求解质量随问题规模急剧下降.同时,Gottlieb 提出了基于矩阵编码的遗传算法,其求解质量和效率明显优于基于排列编码和基于 Prüfer 数编码的遗传算法.虽然基于矩阵编码的遗传算法相对有效,但 Sun 的 Tabu 启发式搜索方法被证明是目前求解 FCTP 问题综合性能最好的启发式算法.

为了进一步改善遗传算法性能,利用 FCTP 的最优解是运输图的一棵生成树的特殊属性,本文提出了一种新的遗传局部搜索算法(genetic local search algorithm,简称 GLSA).GLSA 算法用先根遍历边排列模式编码生成

树,采用随机 Prim 算法初始化个体,其高效的子树补充式单点交叉操作直接满足节点的能力约束或需求约束,不需要任何修补.基于网络单纯型方法的邻域搜索操作能够在解的邻域内集中搜索,加快解的改善.为了进一步改善已经获得的近优解,提出了连续随机节点邻域搜索的集中邻域搜索策略,以及随机旋转变异与全局邻域搜索相结合的局部最优逃逸策略,极大地强化了遗传算法的全局寻优能力.计算实验证明:GLSA 算法优于基于排列编码的遗传算法和基于矩阵编码的遗传算法;针对一定规模的 FCTP 问题,GLSA 算法优于 Tabu 启发式搜索算法,具有更高的获得最优解的几率.

1 并行机间歇过程调度问题

并行机间歇过程调度问题(PBPSP)描述:在计划区间 H 内需要生产 n 种产品,产品 i 的计划需求为 D_i (i=1,2,…,n);有 m 台按批次生产的非等同间歇过程设备,设备 j 的生产能力为 C_j (j=1,2,…,m),每台设备可生产多种产品,设备 j 处理产品 i 的最大允许批量为 $\max U_{i,j}$;设备 j 上不同产品的处理批次切换时具有与设备和产品种类相关、产品处理顺序无关的设备设置时间,即当设备 j 处理的第 1 个批次为产品 i 的某批次或从产品 i'转为处理产品 i 时具有设置时间 $su_{i,j}$;同一设备在顺序处理同种产品的多个批次时,批次切换导致的设备设置时间可以忽略不计;设备 j 处理产品 i 的单位产品加工时间为 $v_{i,j}$,现需要在满足产品计划需求约束和不违背设备生产能力约束的条件下,确定一个调度 S,安排 n 种产品在 m 台设备上的处理批次数目、每个批次的批量、每台设备上所有批次的处理顺序,使得所有产品的总处理时间之和最小.

定义 **1**. 在问题 PBPSP 中,如果一个调度 S 满足所有约束,并使得所有产品的总处理时间之和最小,则称调度 S 是最优调度 S_{opt}.

定理 **1**. 在问题 PBPSP 的最优调度 S_{opt} 中,任何一台设备上相邻两批次之间没有空闲时间间隔.

证明:与一般生产调度中相邻两作业之间没有空闲时间间隔一样,容易理解,详细证明略. □

定理 **2**. 在问题 PBPSP 的最优调度 S_{opt} 中,如果产品 i 在设备 j 上的处理量大于 0,则产品 i 在设备 j 上的所有批次都前后相连,顺序处理.

证明:假设在最优调度 S_{opt} 中,设备 j 的第 k 个批次 $b_{j,k}$ 与第 k+2 个批次 $b_{j,k+2}$ 均处理产品 i,第 k+1 个批次处理产品 i',设 $b_{j,k}$,$b_{j,k+1}$ 和 $b_{j,k+2}$ 的处理时间分别为 $t_{j,k}$,$t_{j,k+1}$ 和 $t_{j,k+2}$,产品 i 和 i'在设备 j 上的设置时间分别为 $su_{i,j}$ 和 $su'_{i,j}$,则 S_{opt} 中批次 $b_{j,k}$,$b_{j,k+1}$ 和 $b_{j,k+2}$ 的总处理时间为 $t_{j,k}+t_{j,k+1}+t_{j,k+2}+su_{i,j}+su'_{i,j}$;如果将 $b_{j,k}$ 与 $b_{j,k+2}$ 连续处理,并将 $b_{j,k+1}$ 放在 $b_{j,k+2}$ 之后处理,则批次 $b_{j,k}$,$b_{j,k+1}$ 和 $b_{j,k+2}$ 的总处理时间变为 $t_{j,k}+t_{j,k+1}+t_{j,k+2}+su'_{i,j}$,比顺序处理 $b_{j,k}$,$b_{j,k+1}$ 和 $b_{j,k+2}$ 所需时间减少 $su_{i,j}$,由于 $su_{i,j}>0$,故顺序处理 $b_{j,k}$,$b_{j,k+2}$ 和 $b_{j,k+1}$ 更好(如图 1 所示).依此类推,如果设备 j 上产品 i 的其他批次之间也存在其他产品的某个批次,那么经过类似的顺序变换,最终使得设备 j 上产品 i 的所有批次都前后相连,顺序处理. □

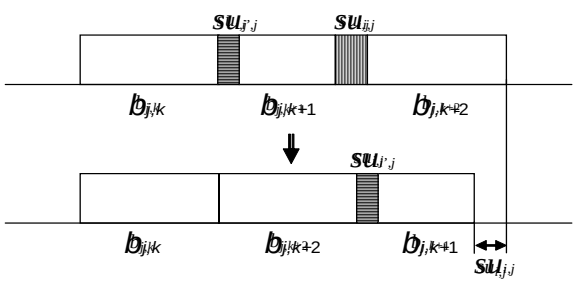

Fig.1 All batches of product i in machine j are processed sequentially

图 1 设备 j 顺序依次处理产品 i 的所有批次

根据问题描述,同一设备在顺序处理同种产品的多个批次时,批次切换导致的设备设置时间可以忽略不计,结合定理 2 可知:在已知产品 i 在设备 j 上处理总量 $x_{i,j}$ 的条件下,可以根据生产安排需要或市场需要将 $x_{i,j}$ 分割为任意数量批次(批量小于最大允许批量 $\max U_{i,j}$),只要维持这些批次都前后相连、顺序处理,那么任意排列这些

批次的处理顺序均不会对总处理时间发生影响,且由于同一设备上不同产品的处理批次切换时具有与产品处理顺序无关的设备设置时间,则任意排列设备 j 上所有产品的处理顺序也不会对总处理时间发生影响.故总结起来,只要知道每种产品在每个设备上的分摊数量就可以确定所有产品在所有设备上的总处理时间.至此,问题 PBPSP 简化为:在满足需求约束和能力约束的条件下,确定每种产品在各设备上的分配数量,使得所有产品的总处理时间之和最小.

如果将设备看成具有有限能力的供应地点,产品看成具有一定需求数量的需求地点,产品在设备上的分摊数量看成供应地点向需求地点的运输量,设备生产产品所花费的时间看成运输费用,则上述问题可以看成一个运输问题,但由于产品在设备上的生产时间除了具有与生产数量成正比的变化处理时间以外,还具有顺序独立的产品切换时设备设置时间,故上述问题不是一个线性运输问题,而是一个具有固定费用的运输问题,固定费用即为产品切换时设备设置时间.问题 PBPSP 与等价的 FCTP 问题的数学模型如下:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{m} g_{ij}(x_{ij})$$

$$\text{st.} \quad \begin{cases} \sum_{j=1}^{m} x_{i,j} = D_i, & i=1,2,...,n \\ \sum_{i=1}^{n} x_{i,j} = C_j, & j=1,2,...,m \\ x_{i,j} \geq 0, & i=1,2,...,n;\ j=1,2,...,m \end{cases} \tag{1}$$

其中

$$g_{ij}(x_{ij}) = \begin{cases} 0, & x_{i,j}=0 \\ su_{i,j} + v_{i,j}x_{i,j}, & x_{i,j}>0 \end{cases} \tag{2}$$

式(1)为需求约束和能力约束,表明需求地点 i 接收所有供应地点的运输量总和等于其需求量 D_i,每个供应地点 j 向所有需求地点运输的数量总和等于其能力 C_j.式(2)表明运输成本由固定费用 $su_{i,j}$ 和线性费用 $v_{i,j}x_{i,j}$ 组成.固定费用运输问题是线性运输问题的推广,线性运输问题是一种特殊的线性规划问题,可以在多项式时间内获得最优解,但固定费用运输问题由于固定费用的出现,使得目标函数出现不连续性,导致固定费用运输问题成为一个 NP-hard 问题,且其最优解出现在由约束定义的凸集的某个极点上.对于具有 m 个供应地点和 n 个需求地点的 $m\times n$ 固定费用运输问题,固定费用运输问题的解有以下两个特征:(1) 有 $m+n-1$ 个基变量,每个基变量对应运输表中的一格;(2) 基必须是一棵运输树(运输图的一棵生成树),即在运输表的每一行和每一列中至少存在一个基本单位,且不构成回路.这两个特征是本文遗传局部搜索算法的基础.

2 遗传局部搜索算法

2.1 先根遍历边排列编码

给定生成树 $T=(V_T,E_T)$和起始节点(根节点)r,先根遍历边排列编码按照先根遍历模式访问树的每个节点 $v\in V_T-\{r\}$,每访问一个节点即访问一条不同的边 e,遍历一棵生成树的所有节点(边)共需$|V_T|(|E_T|)$次访问,按照访问顺序排列的边集合即作为生成树的基因型.

2.2 创建生成树

在本文的算法中,从运输图 $G=(V,E)$中创建或者从其他运输树等向一棵已存在的子树补充边并形成一棵满足节点能力约束或需求约束的新运输树是基本任务.Prim 算法是创建生成树的经典算法,它以贪心方式每次从图中选择一个成本最低的可行边加入到新创建的子树中,最终形成图的一棵生成树.在固定费用运输问题中,由于生成树必须满足节点能力约束或需求约束,应用 Prim 算法可能会产生不可行个体,且其贪心偏好使得产生的个体不能以统一概率分布在解空间中.下面给出创建固定费用运输问题生成树的基本算法——PrimRFCT(Prim random fixed charge transportation tree)算法.PrimRFCT 算法随机从运输图 G 的那些还没有被选择且符合条件

的边中选择一条边加入到已选择的边集中,并赋予最大可分配量给新加入的边.重复$|V|-1$次,PrimRFCT 函数将产生一棵生成树.用 T 表示需要生成的运输树,V_T 表示 T 的节点集合,E_T 表示 T 的边集合,节点 u 的需求用 $D(u)$ 表示(目的地或产品的需求为其需求量,源地或设备的需求为其可供应量,两者均为正值),P_T 表示 T 中需求未完全满足的节点集合.PrimRFCT 算法描述如下:

procedure $PrimRFCT(V_T,E_T,P_T,V,E)$

If $P_T \neq \varnothing$ Then

Select randomly a node u from the set P_T;

Else

Select randomly a node u from the set V_T;

End If

Select randomly an edge (u,v) from the set E, and $v \in \{V-V_T\}$;

$x(u,v) \leftarrow \min(D(u),D(v))$;

$D(u) \leftarrow D(u)-x(u,v)$;

$D(v) \leftarrow D(v)-x(u,v)$;

$V_T \leftarrow V_T \cup \{v\}$;

$E_T \leftarrow E_T \cup \{(u,v)\}$;

If $D(u)=0$ and $u \in P_T$ Then $P_T \leftarrow P_T-\{u\}$; End If

If $D(v) \neq 0$ Then $P_T \leftarrow P_T \cup \{v\}$; End If

2.3 子树补充式单点交叉

由于需要满足能力约束和需求约束,故针对运输生成树的交叉操作通常都需要进行大量的修补操作才能形成可行个体(如基于矩阵排列编码的遗传算法),修补过程不仅复杂,且比较耗时.利用先根遍历边排列编码策略,本文提出了特殊的子树补充式单点交叉操作:从父个体 $P1$ 和 $P2$ 中随机选择一个作为基本个体(如 $P1$),从基本个体 $P1$ 的边集 E_{P1} 中随机选择一个位置 $k(0<k<|E_{P1}|)$,其对应的边为 $e_k=(u_k,v_k)$;从位置 k 出发,向后顺序遍历边直到获得一棵以节点 v_k 为根的子树 $T(P1,k)=(V_{T(P1,k)},E_{T(P1,k)})$,其中$|E_{T(P1,k)}|<|E_{P1}|$,然后从 $P2$ 和 G 向子树 $T(P1,k) \cup \{e_k\}$添加$|E_{P1}|-|E_{T(P1,k)}|-1$ 条边,以形成完整的子个体 $S1$.在添加边的过程中,如果 $P2$ 中存在可行边 $e=(u',v')$,其中 $u' \in P_{T(P1,k)}, e \in E_{P2}-E_{T(P1,k)}$,则调用 $PrimRFCT(V_{T(P1,k)},E_{T(P1,k)},P_{T(P1,k)},V_{P2},E_{P2})$,从 $P2$ 中添加边 e;否则调用 $PrimRFCT(V_{T(P1,k)},E_{T(P1,k)},P_{T(P1,k)},V,E)$算法从原运输图 G 中补充一条可行边 $e \in E-E_{P2}-E_{T(P1,k)}$.

2.4 基于网络单纯型方法的邻域搜索

网络单纯型方法是求解最小费用网络流问题的高效算法,其基本操作是通过添加一条入基变量边和删除一条出基变量边对当前生成树(基可行解)进行一次旋转以获得一个邻接生成树(基可行解),以初始解为起点,通过多次迭代旋转,可实现解的不断变换,最终获得最优解(线性费用情况).这种旋转过程具有邻域搜索性质.给定基可行解 T,其邻域由所有可通过对 T 进行一次旋转而获得的邻接基可行解的集合.考虑在基可行解 T 中引入非基变量边$(i,j) \notin T$,按如下方法确定出基变量边:在 T 中获取从节点 j 到节点 i 的路 P,然后在保持原始解可行的条件下选择分配量为$\alpha=\min\{x_{p,q}|(q,p) \in P\}$的基变量边作为出基变量边.如果$\alpha=0$,则选择一条具有最大固定费用的基变量边$(r,s) \in P$ 作为出基变量边.如果(i,j)被引入基可行解中,则其分配量 $x_{i,j}=\alpha$,其引起的目标函数的净改变可通过式(4)获得[13]:

$$\delta_{i,j}=\begin{cases}0, & \text{if } \alpha=0\\ \alpha(c_{i,j}-\pi_i+\pi_j)+f_{i,j}-\sum\limits_{(p,q)\in U_1}f_{p,q}+\sum\limits_{(p',q')\in U_2}f_{p',q'}, & \text{if } \alpha>0\end{cases} \tag{3}$$

式(3)中,$U_1=\{(p,q)|(q,p)\in P, x_{p,q}=\alpha\}$,$U_2=\{(p',q')|(p',q')\in P, x_{p',q'}=0\}$,$\pi_i$ 和 π_j 分别是节点 i 和节点 j 的对偶变量,对偶变量仅用变化成本定义. $\sum\limits_{(p,q)\in U_1}f_{p,q}$ 和 $\sum\limits_{(p',q')\in U_2}f_{p',q'}$ 分别表示离开生成树的所有基变量边(P 中分配量从某个正

值减少到 0 的边)所减少的固定成本总和以及基可行解中所有分配量增加的退化基变量边(P 中分配量从 0 增加到某个正值的边)所增加的固定成本总和.为实现邻域搜索,选择非基变量$(i,j)\notin T$,使得$\delta_{i,j}=\min\{\delta_{p,q}|(p,q)\notin T\}$作为入基变量.如果$\delta_{i,j}<0$,则非基变量$(i,j)\notin T$作为入基变量将改善当前解,否则当前解是一个局部最优解.

为避免在计算所有非基变量$(i,j)\notin T$的$\delta_{i,j}$时耗费过长时间,在本文的邻域搜索中,随机选择一个节点 i,以节点 i 的所有邻接非基变量边$(i,j)\notin T$作为当前基可行解的邻域进行搜索,并称之为随机节点邻域搜索,与之相对应,如果是针对 T 中所有节点,则称之为全局邻域搜索.

由于添加边和删除边的操作打破了 T 的先根遍历边排列状态,故需要对邻域搜索后 T 的子个体 S 的边集重新排列,恢复为先根遍历边排列状态.假定非基变量边$(i,j)\notin T$为入基变量边,T 的边集的第 k 条边 $e_k=(u_k,v_k)\in T$ 为出基变量边,从位置 $k+1$ 出发,向后顺序遍历边直到获得一棵以节点 v_k 为根的子树 $T(T,k)=(V_{T(T,k)},E_{T(T,k)})$,然后从生成树 T 中删除子树 $T(T,k)$,获得另一棵子树 $T-T(T,k)$,子树 $T(T,k)$和子树 $T-T(T,k)$通过边(i,j)相连.假定节点 i 位于子树 $T-T(T,k)$中,节点 j 位于子树 $T(T,k)$中,现将 $T(T,k)$从以节点 v_k 为根节点的子树变换为以节点 j 为根节点的子树,变换的同时将 $T(T,k)$的排列边 $E_{T(P1,k)}$重排为从节点 j 出发按先根遍历所得的排列边 $E'_{T(P1,k)}$即可恢复为先根遍历边排列状态.

2.5 随机旋转变异

基于网络单纯型方法的邻域搜索是在当前基可行解 T 的邻域内寻找一条使目标成本下降得最快的非基变量边$(i,j)\notin T$作为入基变量,另一种选择入基变量的策略是不经过邻域搜索比较,而是直接在 T 的所有非基变量边集合 $N=\{(i,j)|(i,j)\notin T\}$中随机选择一条边$(i,j)$作为入基变量,再按照与邻域搜索相同的方法确定一条出基变量$(r,s)\in P$,并相应减少或增加路 P 上各边的分配量.

与基于网络单纯型方法的邻域搜索一样,随机旋转变异在添加一条非基变量边和删除一条基变量边之后,其父个体的先根遍历边排列状态被打破,故也需要重排其边集,重排方法与第 2.4 节中基于网络单纯型方法的邻域搜索使用的重排方法相同.

2.6 集中邻域搜索

对遗传算法性能改进的研究往往只注重于从增加种群多样性的角度防止进化过程早熟收敛和陷入局部最优[16,17],而对挖掘既有解的邻域解策略考虑较少.虽然前文结合邻域搜索操作的遗传算法能够在一定程度上增强搜索邻域解空间的能力,但较小的变异概率抑制了邻域搜索能力的充分发挥,故需要在适当进化时机进一步加强邻域搜索.为此,本文提出了以下集中邻域搜索策略.

在遗传算法的演化计算过程中,如果连续η代不能改善当前最优解 S_0,则认为获得一个局部最优解,且在以后的演化计算中难以从此局部最优解跳出或其跳出进程会很缓慢,为进一步开采此局部最优解的邻近优化解,加快进化过程,对当前最优解 S_0 执行连续γ_0步随机节点邻域搜索操作,在连续γ_0步邻域搜索的过程中,如果某步搜索 k 获得的解 S_k 优于当前最优解 S_0,则停止搜索,用 S_k 代替 S_0,并以 S_0 所在的种群 $Popu_{s_0}$ 为起点,继续进行遗传进化,否则在执行完连续γ_0步遗传搜索后继续对种群 $Popu_{s_0}$ 进行遗传进化.

连续执行随机节点邻域搜索的步数γ_0是集中邻域搜索策略的关键参数.由于不能预先获知在距离当前局部最优解 S_0 多远的地方才有可能获得改善解 S_k,且这种距离随着 S_0 的变化和进化时期的变化而发生变化,故采用自适应参数调整法确定γ_0:用 LS 表示集中邻域搜索过程,用 EP 表示从初始种群到执行第一次 LS 过程以及两次 LE 过程之间的遗传进化过程,则在执行至少一次 LS 过程的遗传算法中,EP 过程和 LS 过程交替出现.用二元组(EP,LS)表示遗传进化过程与集中邻域搜索过程组合的进化过程,如果连续 k 次组合进化过程(EP,LS)没有改进当前最优解 S_0,则增加变换次数,令$\gamma_k=\varepsilon^{k-1}\gamma_0$,如果 k 增至一个阈值 $Threshold$(即 k 为正整数,且 $1<k\le Threshold$),则调用局部最优逃逸操作.在每个 EP 和 LS 过程中,只要获得改进解,则将γ_k置为初始值,即令$\gamma_k=\gamma_0$,并继续遗传进化.

2.7 局部最优逃逸

根据组合优化的进化过程和动态系统理论之间的相似性,Battiti 在其反馈 Tabu 搜索算法中给出了两种局

部最优吸引子[18]:一种是由最速下降搜索策略引起动态吸引子,这些吸引子事实上是稳定的不动点,直到引入某种跳坑策略才能进一步搜索.显然,在具有交叉操作和变异操作的遗传算法以及在基于节点随机邻域搜索的集中邻域搜索中,均难以存在这种动态吸引子;另一种是混沌吸引子,这种混沌吸引子具有区域收缩的特征,将从不同初始解出发的演化进程压缩到一个有限的部分解空间中,如果此部分解空间不包含问题的全局最优解,则进化过程将永远找不到全局最优解.在前文的遗传局部搜索中,集中邻域搜索策略从纵深方向挖掘邻域解,在增加改善解几率的同时,也增加了将进化过程推入局部最优的可能性.在 $Threshold$ 次组合进化过程(EP,LS)均不能改进当前局部最优解 S_0 的情况下,可以认为这个局部最优解是一个混沌吸引子.为了逃离混沌吸引子,引入 Battiti 的局部最优逃逸策略以对当前最优解实施逃逸变化.

局部最优逃逸的基本思想是对当前局部最优解 S_0 实施连续 δ_0 步随机旋转变异.鉴于在局部最优逃逸之前已经实施了 $\varepsilon^{Threshold-1}\gamma_0$ 步随机节点邻域搜索,选择 $\delta_0=2\varepsilon^{Threshold-1}\gamma_0$.设第 $k(1\le k<\delta_0)$ 次随机旋转变异获得的解为 S_k,入基变量边为(i,j),出基变量边为(r,s),用二元组$((i,j),(r,s))$表示此变异(旋转),为防止在后继的操作中立即返回到解 S_k,将$((i,j),(r,s))$禁忌(在禁忌期间,不允许进行$((r,s),(i,j))$旋转),然后对 S_k 实施全局邻域搜索,并更新 S_k.在逃逸变化过程中,如果某步逃逸 k 获得的解 S_k 优于当前最优解 S_0,则停止逃逸,用 S_k 代替 S_0,并以 S_0 所在的种群 $Popu_{s_0}$ 为起点,继续进行遗传进化,否则停止计算.

2.8 遗传局部搜索算法

用 $DepthMining(T,step)$表示集中邻域搜索函数,其中 T 表示需要进行集中邻域搜索的局部最优解;$step$ 表示需要随机节点邻域搜索的次数.用 $LocalOptEscape(T,step)$表示局部最优逃逸函数,其中 T 表示需要进行局部最优逃逸的局部最优解;$step$ 表示需要逃逸的步数.选择策略采用锦标赛选择,个体适应度直接使用目标函数值.局部最优逃逸遗传算法具体描述如下:

1. 初始化参数:种群规模 $N_{popu}=100$,最大遗传代数 $\max Gene=2000$,锦标赛选择压力 $Pressure=2$,交叉概率 $p_c=0.3$,变异概率 $p_m=0.1$,$\gamma_0=3000$,连续未改进组合进化过程数目 $nobetter=0$,$\eta=30$,$\varepsilon=1.5$,$Threshold=3$.

2. 调用 $PrimRFCT$ 算法从运输图 $G=(V,E)$产生 N_{popu} 个生成树个体,形成初始种群 $Popu_0$,在个体初始化过程中,首先随机选择一个结点 k 作为生成树的根,然后调用 $|V|-1$ 次 $PrimRFCT$ 算法产生一棵生成树个体;计算初始种群的个体目标函数值,获得种群 $Popu_0$ 的最佳个体 $Best_0$.

3. 对第 t 代种群 $Popu_t(0\le t<\max Gene)$实行 $N_{popu}/2$ 次遗传操作:

1) 利用锦标赛选择从第t代种群$Popu_t$中选择两个父个体$P1$和$P2$.

2) 对父个体$P1$和$P2$按概率p_c实行子树补充式单点交叉操作,产生两个子个体$S1$和$S2$,产生子个体$S1$时以$P1$为基本个体,用$P2$和G补充边;产生子个体$S2$时以$P2$为基本个体,用$P1$和G补充边.

3) 对父个体$P1$和$P2$($P1$和$P2$未交叉)或子个体$S1$和$S2$($P1$和$P2$已交叉),按概率p_m实行基于网络单纯型方法的邻域搜索操作,并将处理后的个体放入种群$Popu_{t+1}$.

4. 计算种群 $Popu_{t+1}$ 的个体目标函数值,获得种群 $Popu_{t+1}$ 的最佳个体 $Best_{t+1}$ 和最差个体 $Worst_{t+1}$,如果 $Best_{t+1}$ 劣于种群 $Popu_t$ 的最佳个体 $Best_t$,则采取保优策略,用 $Best_t$ 代替 $Worst_{t+1}$,即 $Best_{t+1}$ 变为 $Best_t$;如果 $Best_{t+1}$ 优于 $Best_t$,令 $step=\gamma_0$,$nobetter=0$.

5. 如果遗传代数等于 $\max Gene$,则停止计算;如果连续 η 代遗传操作没有改进最优解 $Best_{t+1-\eta}$,即 $Best_{t+1}=Best_{t+1-\eta}$,则调用集中邻域搜索函数 $DepthMining(Best_{t+1},step)$;否则,以种群 $Popu_{t+1}$ 为当前种群,转 3.

6. 如果 $DepthMining$ 函数返回值优于当前最优解 $Best_{t+1}$,则用此返回值代替 $Best_{t+1}$,令 $step=\gamma_0$,$nobetter=0$,以种群 $Popu_{t+1}$ 为当前种群,转 3;否则,$step=\varepsilon\gamma_0$,$nobetter$ 加 1.

7. 如果 $step=\varepsilon^{Threshold-1}\gamma_0$,则调用 $LocalOptEscape(Best_{t+1},step)$;否则,以种群 $Popu_{t+1}$ 为当前种群,转 3.

8. 如果 $LocalOptEscape(Best_{t+1},step)$改进了当前最优解 $Best_{t+1}$,则以种群 $Popu_{t+1}$ 为当前种群,转 3;否则停止计算.

3 算法分析

3.1 正确性分析

从运输图 $G=(V,E)$调用$|V|-1$次*PrimRFCT*算法一定能够创建可行生成树(基可行解).在创建生成树T之前,T为空,即$|V_T|=|E_T|=0$,每执行一次 *PrimRFCT* 算法,就向 T 中增加一个不包含在 V_T 中的节点和不包含 E_T 中的边.事实上,在执行第$|V|-1$次 *PrimRFCT* 算法之前,集合 P_T始终只存在一个元素,在执行第$|V|-1$次 *PrimRFCT* 算法之后,集合 P_T清空,所有节点需求都满足,$|V_T|=|V|$, $|E_T|=|V_T|-1$,同时,这$|V_T|$个节点一定是连通的.

对于子树补充式交叉操作,在子树 $T(P1,k)\cup\{e_k\}$的$|V_{T(P1,k)}|+1$ 个节点中,子树 $T(P1,k)$的所有节点的需求都是满足的(从父个体 $P1$ 继承),在边 e_k的父节点 u_k和子节点 v_k中,只有父节点 u_k的需求未被满足.在调用 *PrimRFCT* 算法补充$|E_{P1}|-|E_{T(P1,k)}|-1$ 条边的过程中,集合 P_T 也始终只存在一个元素,故子树补充式交叉操作一定能获得一个以 u_k为根节点的可行生成树个体.由于子树 $T(P1,k)\cup\{e_k\}$的排列边从父个体 $P1$ 继承,故这些边排列状态一定是从节点 u_k出发的先根遍历排列模式;用 *PrimRFCT* 算法补充的边组成另一棵以节点 u_k为根的子树,类似创建生成树过程,这个子树的边排列状态也为先根遍历模式,并附加在子树 $T(P1,k)\cup\{e_k\}$的排列边之后,故交叉后获得的整个子个体的边排列状态为先根遍历模式.

基于网络单纯型方法的邻域搜索和随机旋转变异操作添加一条边并删除一条边,保持生成树的边数不变,且通过调整路 P 上各边的分配量,各个节点的需求量也是满足的.通过重排后,其边排列状态也为先根遍历模式.

3.2 空间与时间复杂性分析

GLSA 算法在整个求解过程中用一个邻接表存储运输图 $G=(V,E)$,用二维数组存储图 G 上各边的变化成本和固定成本.用大小为$|V|-1$的一维数组存储个体,数组元素为边对象,边对象成员包括边的父节点和子节点以及边的分配量,可以看出,这种个体编码模式是简洁与节省存储空间的.

在 *PrimRFCT* 算法中,由于运输图 G 以邻接表的形式存储,故可以在 $O(|E|)$时间内获得一个节点的邻接节点.辅以链表存储生成树节点集合 V_T,对于判断一个节点是否存在于 V_T 中的操作,其空间与时间复杂度均为 $O(|V|)$.在单点交叉操作和重排生成树边操作中,均涉及获取某个节点的子树操作.根据先根遍历的边排列模式,从一个生成树个体 $T=(V_T,E_T)$中获取子树 $T(T,k)$,只需要顺序访问$|E_{T(P1,k)}|$条边即可,故耗费时间为 $O(|E_T|)$.在基于网络单纯型方法的邻域搜索和随机旋转变异操作中,获取个体 T 中从节点 j 到节点 i 的路 P 是一个基本操作,用堆栈临时存储非路 P 上的节点,用链表存储路 P,获取路 P 的空间与时间复杂度均为 $O(|V|)$.根据第 2.4 节中的描述,将子树从以节点 v_k为根节点按先根遍历模式排列的边集,重排为以节点 j 为根节点按先根遍历模式排列的边集需耗费 $O(|V|)$时间.

集中邻域搜索操作是高效率的,它只对当前种群的最优个体实施连续$\xi(1\le\xi\le step)$步变换,相当于对ξ个个体实施一次邻域搜索操作,一次集中邻域搜索耗费的时间不超过对规模大小为 N_{popu} 的种群实施$\lceil\xi/N_{popu}\rceil$次只有邻域搜索操作的遗传进化.当然,在实际计算过程中,不同的$\gamma_0,\varepsilon,Threshold$ 值对算法效率产生影响.局部最优逃逸操作也是高效的,它只使用随机旋转变异和全局邻域搜索作为基本操作,通常在远小于最大逃逸步数的地方就可以改进当前局部最优解,但由于在获得最终最优解之前,需要执行$\varepsilon^{Threshold-1}\gamma_0$ 次逃逸,故其总体耗费时间相对集中邻域搜索更长.

3.3 收敛性分析

定理 **3**. GLSA 算法获得最优解的几率大于等于标准遗传算法获得最优解的几率.

证明:在 GLSA 算法中,如果连续η代遗传操作没有改进当前最优解,则对当前最优解调用集中邻域搜索函数,如果找到更好的解,则返回更好的解到种群中并继续演化计算,否则返回.故集中邻域搜索策略或者保持、或者提高了算法获得全局最优解的几率.类似地,局部最优逃逸策略也或者保持、或者提高了算法获得全局最优解的几率.进一步,集中邻域搜索策略和局部最优逃逸策略的组合,同样或者保持、或者提高了算法获得全局最优解的几率. □

4 计算实验与分析

固定费用运输问题的求解难度随着问题规模的增大而增加,但对于同种规模问题,求解难度和固定成本与变化成本之间的比值呈强对应关系.为充分证明 GLSA 算法的优越性,使用两类测试数据:一类测试数据是随机生成数据,这主要是与 Sun 的 Tabu 启发式搜索算法(Sun'Tabu 算法)比较解的质量(Gottlieb 的基于矩阵的遗传算法没有使用这种测试).它包括 4 种不同规模的固定费用运输问题,其源地 m(设备)与目的地 n (产品)分别为 10×10,10×20,15×15,10×30.所有测试问题都是全连结的(运输图为完全二分图).GLSA 算法采用 Java 语言编程实现,运行环境为一台 PC 机,CPU 为 Pentium(R)4 2.50GHz,操作系统为 Windows XP.表 1 给出了每种规模固定费用运输问题的总需求量(总供应量).考虑 7 种不同固定费用范围(见表 2)的具体问题.所有问题的单位变化成本都为 3~8 的整数.对于每个规模的每种类型问题,随机生成 15 个实例,每个实例运行 5 遍.

Table 1 Total supply (demand) for different problem sizes

表 **1** 不同规模问题的总需求量(总供应量)

Problem size	Total supply
10×10	10 000
10×20	15 000
15×15	15 000
10×30	15 000

Table 2 Ranges of integer fixed costs for different types of test problems

表 **2** 不同类型具体问题的整数固定成本范围

Problem type	Range of fixed costs	
	Lower limit	Upper limit
A	50	200
B	100	400
C	200	800
D	400	1 600
E	800	3 200
F	1 600	6 400
G	3 200	12 800

另一类测试数据是 Benchmark 问题实例(从数学规划软件 GAMS 的性能世界论坛获得,http://www.gamsworld.org/performance/plib),这是 Gottlieb 使用的比较方法[19].

表 3 给出了 GLSA 算法和 Sun'Tabu 算法在求解每种规模和类型问题的 15 个实例中获得的最差值(worst)、平均值(average)和最好值(best)的比例.如果启发式算法获得的解是最优的,则其百分比为 100;如果是近优的,百分比大于 100,其值越小,越接近全局最优,即越好.另外,用 Optimal 表示在 15 个实例中获得最优解的实例个数.

Table 3 Heuristic solution objective values as a percentage of the optimal objective values

表 **3** GLSA 算法与 Sun'Tabu 算法获得最优解的几率比较

Problem		GLSA				Sun'Tabu			
Size	Type	Worst	Average	Best	Optimal	Worst	Average	Best	Optimal
10×10	A	100.000	100.000	100.000	15	100.002	1000.000	100.000	14
	B	100.000	100.000	100.000	15	100.132	100.013	100.000	13
	C	100.000	100.000	100.000	15	100.198	100.019	100.000	12
	D	100.000	100.000	100.000	15	100.195	100.040	100.000	9
	E	100.000	100.000	100.000	15	100.589	100.143	100.000	7
	F	100.000	100.000	100.000	15	100.924	100.209	100.000	5
	G	100.103	100.010	100.000	14	101.878	100.342	100.000	9
10×20	A	100.000	100.000	100.000	15	100.059	100.008	100.000	13
	B	100.000	100.000	100.000	15	100.059	100.006	100.000	12
	C	100.000	100.000	100.000	15	100.119	100.014	100.000	12
15×15	A	100.000	100.000	100.000	15	100.021	100.003	100.000	12
	B	100.000	100.000	100.000	15	100.104	100.016	100.000	11
	C	100.346	100.136	100.000	10	101.454	100.084	100.000	6
10×30	A	100.000	100.000	100.000	15	100.153	100.019	100.000	10

从表 3 可以看出:在计算实例上,虽然 GLSA 算法和 Sun'Tabu 算法都可能获得最优解,但 GLSA 算法获得最优解的几率远高于 Sun'Tabu 算法,除了问题 10×10 的类型 G 和问题 15×15 的类型 C 外,GLSA 算法获得了其他所有问题类型的 15 个实例的最优解(optimal 等于 15),对于 Optimal 小于 15 的上述两个问题类型,其获得最优解的几率也比 Sun'Tabu 算法要高,且 Worst 指标和 Average 指标都小于 Sun'Tabu 算法.实际上,对于其他问题类型(如问题 10×20 的类型 D,E,F 等),GLSA 算法也具有相当高的获最优解的几率.由于 Sun'Tabu 算法没有给出这些问题类型的获最优解情况,故表 3 也没有列出 GLSA 算法对这些类型的获最优解结果.

Sun'Tabu 算法使用的邻域和 GLSA 算法使用的邻域一致,并通过边出现的频率和次数以实现集中搜索和分散搜索.然而,Sun'Tabu 算法只是对单个解实现邻域搜索,不具有 GLSA 算法具有解种群协同进化优势.另外,GLSA 算法的集中邻域搜索策略和局部最优逃逸策略极大地增强了其搜寻全局最优解的能力.

表 4 给出了 GLSA 算法、基于排列编码遗传算法(表中用 permutation 表示,简称 PE 算法)和基于矩阵编码的遗传算法(表中用 matrix 表示,简称 MA 算法)这 3 种遗传算法在求解 Benchmark 问题时获得的近优解与已知最优解的差距 gap=(value/opt−1)×100,其中,opt 是已知最优解,value 区分为 min,avg 和 max 这 3 种类型,分别对应 $gap_{\min}$,gap_{avg},$gap_{\max}$.从表 4 可以看出,MA 算法优于 PE 算法,而 GLSA 算法又明显优于 PE 算法和 MA 算法,获得了所有 Benchmark 问题的最优解,佐证了 GLSA 高的获得最优解几率,且问题 ran10×10a~ran10×10c, ran10×12~ran17×17 以及 ran12×21 的 min,avg 和 max 均等于 0.对于 avg 和 max 大于 0 的问题实例,GLSA 算法在所有实例上都比其他两种算法 gap_{avg},$gap_{\max}$ 要小.

从表 4 还可以看出:GLSA 算法的 $gap_{\min}$,gap_{avg},$gap_{\max}$ 之间的差距非常小,这证明 GLSA 算法具有很好的鲁棒性,而这是 PE 算法和 MA 算法所不具备的优点.

Table 4 Comparison of GLSA and the permutation encoding GA and the matrix encoding GA

表 4 GLSA 算法与基于排列编码的 GA 和基于矩阵编码的 GA 的解质量比较

Problem		Permutation			Matrix			GLSA		
Name	Opt	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
Ran10×10a	1 499	0	0.60	4.74	0	0	0	0	0	0
Ran10×10b	3 073	0	0.29	4.07	0	0.54	4.07	0	0	0
Ran10×10c	13 007	0	0.10	0.91	0	0.28	0.89	0	0	0
Ran10×12	2 714	0	2.69	5.64	0	0.70	5.64	0	0	0
Ran12×12	2 291	0	2.71	6.37	0	1.51	3.88	0	0	0
Ran13×13	3 252	0.98	0.98	3.26	0.98	3.06	6.98	0	0	0
Ran17×17	1 373	2.62	8.67	15.88	0.58	5.40	10.27	0	0	0
Ran4×64	9 711	0.58	3.00	5.21	0	0.67	1.70	0	0.32	0.94
Ran6×43	6 330	0.87	5.30	10.03	0	1.28	3.98	0	0.20	1.15
Ran8×32	5 247	3.14	6.04	8.61	0.55	3.29	7.62	0	1.28	2.27
Ran10×26	4 270	1.55	7.03	11.69	1.85	7.03	11.92	0	0.15	0.41
Ran12×21	3 664	3.87	8.32	12.69	1.26	4.53	8.98	0	0	0
Ran14×18	3 712	0.45	5.67	9.38	0	5.01	8.89	0	0.16	0.39
Ran16×16	3 823	2.27	5.89	9.36	0.92	4.67	8.53	0	0.33	0.76

GLSA 算法和 PE 算法与 MA 算法的区别之一是应用了具有邻域搜索性质的变异操作,这是增强 GLSA 算法全局寻优能力的一个方面.然而,最重要的差别是 PE 算法与 MA 算法没有实现诸如 GLSA 算法具有的集中邻域搜索和局部最优逃逸操作.为获知集中邻域搜索(DM)和局部最优逃逸(LE)以及用随机节点邻域搜索(LS)作为变异操作和用随机旋转(RP)作为变异操作对求解质量的影响,进行如下验证实验.

随机选择问题 10×10,10×20,15×15,10×30 类型 C 的某个实例,然后应用 4 种操作的不同组合情况求解问题实例(运行 15 遍).图 2 给出了应用 LS 作为变异算子时不采用 DM 和 LE、只采用 DM、只采用 LE、采用 DM 和 LE 这 4 种策略求解问题实例所获得平均解与最优解之间的差距.从图 2 可以看出:不采用 DM 和 LE 只能获得问题 10×10 和 10×20 的最优解;随着问题规模的进一步增加,这种策略获得的解的质量急速下降.在加入 DM 或 LE 后,求解质量得到大大改善,平均解非常接近最优解,且 LE 获得的解的质量稍好于 DM.在 DM 与 LE 均加入后,每个实例的每次运行都获得了最优解,证实了 DM 与 LE 组合所具有的全局搜索能力.图 3 比较了用随机节点邻域搜索(LS)作为变异操作和用随机旋转(RP)作为变异操作对求解质量的影响.从图 3 可以看出:使用 RP

作为变异操作且不加入 DM 和 LE 获得的解的质量最差,获得的每个问题实例的平均解均高于最优解;随着问题规模的增加,这种策略获得的解的质量也急速下降.用 LS 代替 RP,使求解质量大为提高,证明了遗传局部搜索的优异性.从图 3 还可以看出:使用 RP 作为变异操作且加入 DM 和 LE,获得了比应用 LS 作为变异操作且不加入 DM 和 LE 获得的解的质量更好,这也证实了 DM 与 LE 组合所具有的全局搜索能力.

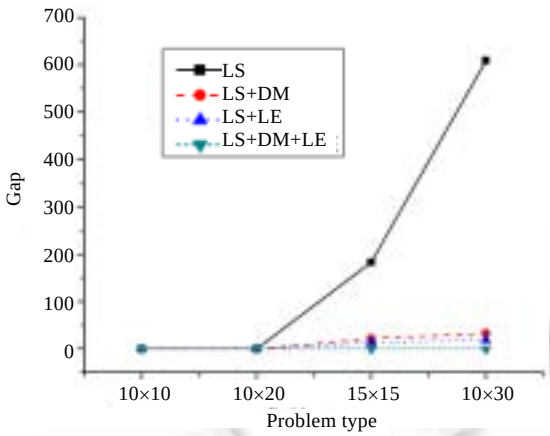

Fig.2 Influence of DM and LE on solution quality

图 2 DM 和 LE 对求解质量的影响

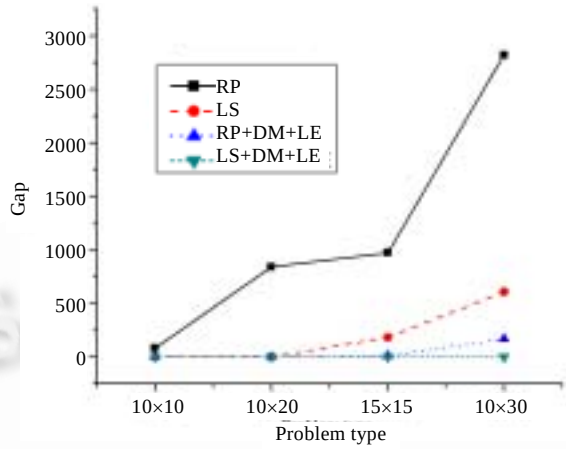

Fig.3 Influence of LS on solution quality

图 3 LS 对求解质量的影响

5 结 论

本文的遗传局部搜索算法有效地解决了 PBPSP 问题和 FCTP 问题,其提供的集中搜索策略和局部最优逃逸策略对其他遗传局部搜索算法有一定的指导意义.通过少许修改,也可将 GLSA 用于解决最优网络设计、最小费用网络流等其他网络优化问题.

另外,虽然本文研究的 PBPSP 问题是一个 NP-hard 问题,但其约束条件还相对较弱,在后续研究中,我们将考虑更为复杂的约束条件,如强的批次排序约束、订单交货期约束等.我们也期望本文能够提供用 meta 启发式方法同时进行需求分批和批次调度的并行机间歇过程调度问题解决模式,为解决实际生产中的大规模间歇过程生产调度问题提供一种手段.

References:

[1] Kondili E, Pantilides CC, Sargent, RWH. A general algorithm for short-term scheduling of batch operations-I, MILP formulation. Computers and Chemical Engineering, 1993,17(2):211−227.

[2] Floudas CA, Lin X. Continuous-Time versus discrete-time approaches for scheduling of chemical processes: A review. Computers and Chemical Engineering, 2004,28:2109−2129.

[3] Cerda J, Henning GP, Grossmann IE. A mixed-integer linear programming model for short-term scheduling of single-stage multiproduct batch plants with parallel lines. Industrial and Engineering Chemistry Research, 1997,36(5):1695−1707.

[4] Mendez CA, Hennning GP, Cerda J. Optimal scheduling of batch plants satisfying multiple product orders with different due-dates. Computers and Chemical Engineering, 2000,24:2223−2245.

[5] Hui CW, Gupta A. A bi-index continuous-time mixed-integer linear programming model for single-stage batch scheduling with parallel units. Industrial and Engineering Chemistry Research, 2001,40(25):5960−5967.

[6] Wu JY, He XR, Chen BZ, Qiu T. A new continuous-time MILP model for scheduling of multi-product batch plants. Journal of Chemical Industry and Engineering, 2003,54(9):1251−1256 (in Chinese with English abstract).

[7] Murty KG. Solving the fixed charge problem by ranking the extreme points. Operations Research, 1968,16:268−279.

[8] Palekar US, Karwan MK, Zionts S. A branch-and-bound method for the fixed charge transportation problem. Management Science, 1990,36(9):1092−1105.

[9] Walker WE. A heuristic adjacent extreme point algorithm for the fixed charge problem. Management Science, 1976,22(5):587−596.

[10] Vignaux GA, Michalewicz Z. A genetic algorithm for the linear transportation problem. IEEE Trans. on Systems, Man, and Cybernetics, 1991,21(2):445−452.

[11] Gen M, Li YZ. Spanning tree-based genetic algorithm for bicriteria fixed charge transportation problem. In: Proc. of the Congress on Evolutionary Computation. Washington: IEEE Press, 1999. 2265−2271.

[12] Xuan GN, Cheng RW. Genetic Algorithms and Engineering Optimization. Beijing: Tsinghua University Press, 2004. 226−254 (in Chinese).

[13] Gottlieb J, Paulmann L. Genetic algorithms for the fixed charge transportation problem. In: Fogel DB, *et al*., eds. Proc. of the 1998 IEEE Int'l Conf. on Evolutionary Computation. Piscataway: IEEE Press, 1998. 330−335.

[14] Sun MH, Aronson JE, Mckeown PG, Drinka D. A tabu search heuristic procedure for the fixed charge transportation problem. European Journal of Operational Research, 1998,106:441−456.

[15] Gottlieb J, Julstrom BA, Rothlauf F, Raidl GR. Prüfer numbers: A poor representation of spanning trees for evolutionary search. In: Spector L, *et al*., eds. Proc. of the 2001 Genetic and Evolutionary Computation Conf. San Francisco: Morgan Kaufmann Publishers, 2001. 343−350.

[16] Su XH, Yang B, Wang YD. A genetic algorithm based on evolutionarily stable strategy. Journal of Software, 2003,14(11): 1863−1868 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1863.htm

[17] Guo GQ, Yu SY. Genetic drift analysis of recombination. Journal of Software, 2003,14(11):1875−1881 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1875.htm

[18] Battiti R, Tecchiolli G. The reactive tabu search. ORSA Journal on Computing, 1994,6(2):126−140.

附中文参考文献:

[6] 吴建昱,何小荣,陈丙珍,邱彤.新的多产品间歇生产调度的 MILP 模型.化工学报,2003,54(9):1251−1256.

[12] 玄光南,程润伟.遗传算法与工程优化.北京:清华大学出版社.2004.226−254.

[16] 苏小红,杨博,王亚东.基于进化稳定策略的遗传算法.软件学报,2003,14(11):1863−1868. http://www.jos.org.cn/1000-9825/14/1863.htm

[17] 郭观七,喻寿益.重组的遗传漂移分析.软件学报,2003,14(11):1875−1881. http://www.jos.org.cn/1000-9825/14/1875.htm

苏生(1978－),男,四川南充人,博士生,主要研究领域为 ERP,SCM,生产计划与调度.

战德臣(1965－),男,博士,教授,博士生导师,主要研究领域为现代企业管理,数据与知识工程,企业资源计划与供应链管理技术,电子商务与商务智能,软件重构与重用.

徐晓飞(1962－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为企业智能计算,管理与决策信息系统,数据库,企业资源计划与供应链管理技术,电子商务与商务智能,知识工程及应用.