“云室效应”算法求解仓储系统的任务分配*

陈方泽　　张　钹

（清华大学计算机系　北京　100084）

摘要　AGV的任务规划问题属于复杂的组合优化问题，是NP完全问题.本文提出一种与常规的由任务节点到任务链的规划过程不同的方法，采用由核心路线到基于核心路线的聚类到任务链的新的规划方法，即“云室效应”算法.在得到接近最优任务链之时，提高了规划的实时性.

关键词　AGV，核心路线，聚类，分割，云室效应.

仓储调度问题的目标是把在二维欧氏平面上随机分布的任务节点（有待取或待放物品的货位）中的任务，在最短时间内完成.而这种任务规划问题，总可以认为是寻找一条最佳路径，使得AGV（automated guided vehicle）沿着这条路径执行任务节点的任务时（每个任务被执行1次，AGV为多负载且具有容量约束），系统的吞吐率最大，因此这种路径规划问题比TSP（traveling salesman problem）更难处理.[1]为便于考虑问题，本文AGV的吞吐率定义为物品任务量/物品运输距离，物品取放时间折算为运输距离.AGV基于有效任务节点集φ_i的目标函数如下：

$$\mathrm{Max}(Z_i)=\frac{\sum_{\varphi_i} W_{ij}}{2\sum_{\varphi_i} l_{ij}+l_i+\sum_{\varphi_i}\Gamma_i} \qquad s.t.\ \sum_{\varphi_i} W_{ij}\leqslant V_{AGV} \qquad i\in[1,K];K\in N$$

其中W_{ij}，l_{ij}，Γ_i分别为φ_i的第j个任务节点的任务量、该节点对主路线的转移距离以及该节点任务操作时间的折算距离；l_i为φ_i的主路线长度；V_{AGV}为AGV的任务容量.

大量的任务规划方法是直接针对各任务节点生成任务链，然后再在此基础上形成任务执行路径[2]，这种方法的不足是不易考虑路径的约束条件.为此本文提出了基于核心路线和路径规划相结合的算法，在使生成的任务链接近最优的同时，既便于考虑路径的堵塞与AGV的碰撞问题，又有利于提高系统的实时性.

对于动态仓储调度系统来说，实时性常常是一个关键问题，而过去大量的研究集中于静态规划问题.[3,4]本文通过对各初始任务节点（分子任务节点）的聚类分裂处理，以提高任务规划的实时性.在水果收获机器人的任务规划中，由于在NN（nearest neighbor）算法中采用了分组思想[5]，使得其规划的实时性得到提高.

---

* 作者陈方泽，1969年生，博士后，主要研究领域为人工智能，调度优化，计算机应用.张钹，1935年生，中国科学院院士，教授，主要研究领域为计算机应用，人工智能.

本文通讯联系人：陈方泽，北京 100084，清华大学化工系统工程教研组

本文1995－06－26收到修改稿

本算法的第 1 步生成核心路线集;第 2 步是对初始的任务节点进行聚类,然后围绕核心路线进行聚类,得到聚类节点任务链;最后通过对任务链中的聚类节点分裂、重组,并对路径进行调整,即得到了最后的任务链. 由于这一过程极类似于物理学中的云室效应,因此把这种算法称为"云室效应"算法 CREA(cloud room effect algorithm).

1 路线的生成算法 K－路径启发式双向并行搜索算法

由于 AGV 的耦合会引起路径的阻塞,因此在路径生成时,生成 K 个由起点到终点的最短路径,以便后序任务链生成时有一定的选择余地.[6,7] K 的大小取决于具体问题对实时性的要求. 经典的 K 个最短路径生成方法基于 Dijkstra 著名的最短路算法[8],文献[6,7]在此基础上分别提出了 LS(linear search)算法和 BH(binary heap)算法,其中 BH 算法的实时性优于 LS 实时性. 这 2 种算法的不足是除了生成由起点到终点的最短路径,而且生成由起点到路径网络中任意一个节点的最短路径. 另一个不足是其标号过程具有回溯的特点,因此算法的实时性不高.

针对 LS 算法和 BH 算法的不足,本文提出了 K－路径启发式双向并行搜索算法. 本算法是由寻径的 2 个端点—起点和终点开始交替并行均匀搜索,这样就形成了 2 个相对的搜索树,当 2 个搜索树的的叶节点一致时,就得到了一条由起点到终点的路径,由于并行搜索增大了 2 树的"碰撞截面",显然比 LS 算法和 BH 算法采用的串行搜索效率高. 并行搜索的另一优点是避免了程序设计时的嵌套调用,有利于提高算法的实时性. 另外在搜索树的生长过程中,引入了分枝限界思想. 若正在生长的路径长度已超过了解集中最长路径的长度,则该路径的生成过程停止.

生成的路径放入一解集内,解集的容量为 K,解集内存放的是已找到的最短路径,当解集中的路径数达到 K 时,其中最长的路径将被更短的路径取代.

路线的生成算法基于以下假定:①任务链的起点和终点是已知的;②仓库平面内任意 2 个点是连通的;③已得到了路径的连接矩阵.

本算法的主要问题是确定路径节点、确定路径叶节点及构造路径树枝.

(1)确定路径节点

取任意路径节点链 $\Psi_i=\{ND_1,ND_2,\ldots,ND_j\}_i$,其中 ND_j 是 ND_{j-1}的子节点,若 ND_j 的子节点 P_j 不在节点链内,则找到了新的有效节点,节点链就生长到下一级,成为 $\Psi'_i=\{ND_1,ND_2,\ldots,ND_j,P_j\}_i$.

(2)确定路径叶节点

首先判断新节点是否是另 1 个搜索树的端节点,若是,则找到了 1 条有效路线,本节点链的生长结束.

当新节点的所有子节点已包含在节点链中,则该新节点就成为叶节点,该节点链的生长到此为止.

(3)构造路径树枝

这种算法要同时生长各个节点链,也就是同时生长各个树枝. 每确定 1 个新的节点,就完成了树枝的 1 次生长.

假定现有 n 个节点链,取任一节点链 $\Psi_i=\{ND_1,ND_2,\ldots,ND_j\}_i$,节点链已生长到第 j 级.现在节点链从端节点 ND_j 开始生长,ND_j 的相连路径节点有 m 个,在这些节点中,最后确定的有效节点为 l 个($l\leqslant(m-1)$):$\{P_1,P_2,\ldots,P_l\}$,这样原来的 1 个树枝就又分出 l 个分枝,这些分枝构成的路径节点链为:

$$\left.\begin{array}{l}\{ND_1,ND_2,\ldots,ND_j,R_1\}_i\\ \{ND_1,ND_2,\ldots,ND_j,R_2\}_i\\ \cdots\cdots\\ \{ND_1,ND_2,\ldots,ND_j,R_l\}_i\end{array}\right\}\ l\text{个}$$

这样每生长一级,就要生成新的分枝.

针对图 1 所示的模型仓库,本算法($K=12$)的最大寻径时间为 0.16s(路径的起始点和终止点随机分布),而在同样的仿真环境下,即使在采用限界启发信息后,LS 算法生成路径的时间仍为 0.88s($K=12$),本算法的效率大约是 LS 算法的 5 倍,而 BH 算法的效率大约是 LS 算法的 2 倍.

2 "云室效应"算法 CREA

2.1 初始任务节点预聚类

聚类的目的是为了提高问题求解的实时性,在此把路径作为聚类的核点,这里基于这样一个启发信息,即同一路径上的任务节点,其转移时间为最小的概率最大,而通常的转移时间为零,当然并不能排除不在同一路径上的任务节点的转移时间,小于同一路径上任务节点之间的转移时间,这种情况在任务链起点所在的路径上可能出现.

这样对于图 1 所示的模型仓库,通过预聚类之后,无论实际的任务节点有多少,得到的预聚类任务节点最多只有 26 个(在本仿真实例中只有 23 个),其中的小实心圆为初始任务节点,而大空心圆为预聚类节点.每个预聚类节点由其重心、任务量和路径来标识.

定义 1. (X_{ij},Y_{ij})为任务节点 TN_{ij}的欧氏坐标,该任务节点对应的路径为 R_j,W_{ij}为该节点的任务量.

定义 2. (X_j,Y_j)为 R_j 的预聚类节点 TNC_j 的重心坐标;W_j 为 TNC_j 的任务量;n_j 为 R_j 上的任务节点数,则

$$X_j=\frac{\sum_{i=1}^{n_j}X_{ij}W_{ij}}{\sum_{i=1}^{n_j}W_{ij}},\quad Y_j=\frac{\sum_{i=1}^{n_j}Y_{ij}W_{ij}}{\sum_{i=1}^{n_j}W_{ij}},\quad W_j=\sum_{i=1}^{n_j}W_{ij}.$$

2.2 合并准则算法 TNCA(Task Node Collecting Algorithm)

以核心路线为合并核心,将相关的预聚类任务节点合并,放入目标任务节点集中,重复这一过程,直到目标节点集中的任务节点满足以下 2 个条件之一:

①节点任务总量已超过了 AGV 的任务容量;

②任务集继续扩大使任务链的吞吐率下降;

这样就得到了基于预聚类任务节点的具有前趋约束的任务链.

定义 3. L_i 为核心路线 ML_i 的路线长;L_{ji}为 TNC_j 到 ML_i 的可行欧氏距离(考虑了货

架的障碍效应).

定义 4. n_i 为预聚类任务节点数,Γc_{ij}为 TNC_j 对 ML_i 的转移距离,SR_{ij}为 ML_i 的第 j 个子路径,即 $ML_i=SR_{i1}\cup SR_{i2}\cup\ldots\cup SR_{ij}\cup\ldots\cup SR_{in}$.

基于 ML_i 的目标函数为:$\mathrm{Max}(Z_i)=\dfrac{\sum_{j=1}^{n_j}\alpha_j W_j}{2\sum_{j=1}^{n_j}\alpha_j L_{ji}+L_i+\sum_{j=1}^{n_j}\alpha_j\Gamma c_{ij}}$,

其中 $\alpha_j=\begin{cases}0 \text{ if } TNC_j \text{ is not selected}\\ 1 \text{ if } TNC_j \text{ is selected}\end{cases}$; $\sum_{j=1}^{n_j}\alpha_j W_j\leqslant V_{\mathrm{AGV}}$.

在上式分母中预聚类节点的可行欧氏距离考虑了 AGV 的往返过程,因此其值为 $2\Sigma L_{ji}$. 由此就得到了基于核心路线的预聚类节点任务链.

下面是本算法的类 C 语言描述:

```
输入:核心路线集{MLi};
     聚类任务节点集{TNCj};
     AGV 的路线表{Line_Billm};
输出:聚类节点任务链{TNCn}final;
     N1=核心路线数;
     N2=预聚类任务节点数;
     for(c1=0;c1<N1;c1++){
          N3={MLi}包含的路径数;
          for(c2=0;c2<N3;c2++){
               for(c3=0;c3<N2;c3++){
                    bill[c2]=throughput(ROUT[c2],ND_collect[c3]);/*1*/
                    }
               }
          bill_final[c1]=re-arrange(bill[c2]);/*2*/
          V=0;
          L=L[c1];
          throughput.old=0;
          for(c4=0;c4<N2;c4++){
               V=V+bill_final[c4].volume
               L=L+2*bill_final[c4].L
               throughput.new=V/L;
               if(throughput.new<throughput.old){
                    goto Q;
                    }
               if(V>=AGV 容量){
                    goto Q;
                    }
               throughput.old=throughput.new;
               }
Q:        }
```

注1:$throughput(ROUT[c2],ND_collect[c3])$计算预聚类节点 $ND_collect[c3]$到路径 $ROUT[c2]$的转移吞吐率;$bill[c2]$是一个结构类型,保存 $ROUT[c2]$对各预聚类节点的转移吞吐率、任务量、转移距离,并且已经按吞吐率由大到小的顺序排序.

注2:由于前面得到了各预聚类节点对一个核心路线上的每一个子路径的转移吞吐率,因此会造成预聚类节点之间的冲突,$re-arrange(bill[c2])$子程序通过优化,消灭了冲突.

bill_final[*c*1]具有与 *bill*[*c*2])相同的类型结构,其中存放按吞吐率由大到小排序的无冲突预聚类节点.

对于基于任一核心路线生成的预聚类任务节点链来说,得到的任务链对该核心路线来说是最优的.

定理. 设 $\Omega=\{TNC(j,ML_i)\}$,Ω 是基于 ML_i,通过以上算法得到且已按转移吞吐率排序的预聚类节点集,则由该节点集中的前 β 个预聚类任务节点,按照 *TNCA* 结束2个条件得到的任务链是基于 ML_i 的最优预聚类任务链.

证明:设 Q 为原任务链的任务量,L 为其总的转移距离与 ML_i 长度之和,ΔQ 为 Ω 中刚好被放弃的关键聚类任务节点 TNC_{KEY},即 $TNC_{(\beta+1)}$的任务量,ΔL 为该 TNC_{KEY}的转移距离;若算法是因 $TNC_{(\beta+1)}$加入使吞吐率下降,即使得:$(Q+\Delta Q)/(L+\Delta L)=Q/L$ 而结束的,则可知:$\Delta Q/\Delta L<Q/L$,而 *TNCA* 已保证了对 TNC_{KEY}其后的任一 TNC',必有

$$\Delta Q/\Delta L>\Delta Q'/\Delta L',$$

因而有

$$\frac{Q+\Delta Q'}{L+\Delta L'}<\frac{Q}{L};$$

若算法是由于 AGV 的容量约束而结束的,则可知 TNC_{KEY}及其后的 $TNC_j(j\geqslant\beta+2)$,其转移吞吐率必不大于入选的 *TNC*,因而同理可知,节点的替换并不能使任务链的吞吐率增加. □

2.3 分裂准则算法 TNSA(task node separating algorithm)

由于前面节点的合并是针对预聚类任务节点,因此这里的第一步分裂是有效预聚类节点集中预聚类任务节点的自身分裂,即把预聚类任务节点分裂为初始任务节点(分子任务节点),然后进一步分裂有效初始任务节点,使任务链中的任务容量满足 AGV 的容量约束.

对任务链整体评价函数值贡献最小的分子节点首先被分裂出来,重复这一过程,直到再去掉一个分子节点后,任务链的任务量$<V_{AGV}$,这样就得到了有效分子任务节点集.

最后需进一步分裂有效分子节点集中的分子节点.分裂的原则是使得任务链的吞吐率最大,这一过程是一个切分过程,因此属于 *NP* 完全问题.在此采用了单点切分算法以达到小的时间代价,换取吞吐率的进一步提高.

TNSA 的算法步骤如下:

(1)分裂有效预聚类任务节点集$\{TNC_j\}_j$中的每个 TNC_j 为初始任务节点,由此得到了基于 ML_i 的初始任务节点集$\{TN_{ij}\}_{ML}$,计算该任务节点集中 TN_{ij}的转移吞吐率,并按照其转移吞吐率由大到小排序,得到新的有效初始任务节点集$\{TN_{ij}\}'_{ML}$;

(2)从$\{TN_{ij}\}'_{ML}$的最后一个节点开始,若将该节点从$\{TN_{ij}\}'_{ML}$中分裂出去,剩余节点的容量$>V_{AGV}$,则分裂该节点,重新执行第(2)步骤;否则,得到新的任务节点集$\{TN_{ij}\}''_{ML}$,执行步骤(3);

(3)以 ML_i 为线索,由$\{TN_{ij}\}''_{ML}$的任务节点构造一个任务链.通常的节点排序问题的复杂度为 $O(n!)$,但是这里从 ML_i 出发,而$\{TN_{ij}\}''_{ML}$中的 TN_{ij}已建立了与 ML_i 中子路径的对应关系,因此,其复杂度大大降低;

(4)计算每个节点的切分损失量,选择切分损失最小的节点,采用单点法进行调整,由此就得到了最后的任务链$\{TN_{ij}\}_{ML_final}$.

以上算法的计算量主要在第(3)步，但是本算法是建立在 *Novell* 网络的 *Btrieve* 数据库之上的，因此可利用本数据库的关键字特性，直接建立基于任务节点位置的排序，这就使得排序算法复杂度由 $n!\rightarrow n$.

最后根据 AGV 的任务时序表(AGV 的时序规划不在本文讨论范围内)生成该 AGV 的 K 个任务时序序列，得到避免与其它 AGV 碰撞或阻塞(AGV 碰撞或阻塞的细节可参看文献[1])的 K 个任务节点链的吞吐率，从而得到最后决策.

本算法的算法复杂度是 $O(KNM)$，K，N，M 分别为核心路线数、预聚类任务节点数和核心路线中的子路径数.

3 “云室效应”算法仿真

任务库中随机生成了64个任务节点. 仿真是在微机局域网上实现的，网络操作系统为 Netware V3.11，任务库、路径连接矩阵和 AGV 的任务时序表放在网络的文件服务器上，编程语言为 C 语言.

图1为预聚类节点任务链，图2为预聚类任务链分裂为初始任务节点后得到的任务链，图3是生成的最终任务链，实际上这一任务链是一最优任务链，图4为核心路线数与任务链的生成时间曲线.

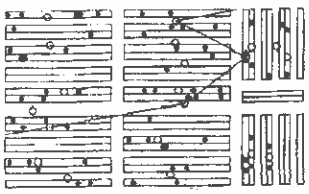

图1 预聚类节点任务链

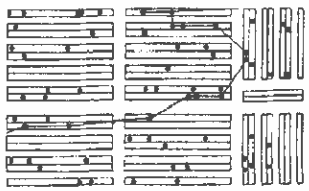

图2 节点任务链

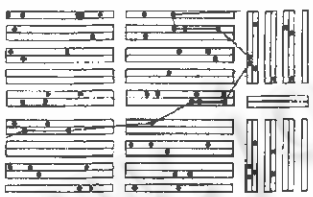

图3 最终任务链

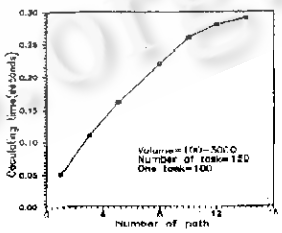

图4 核心路线数与计算时间曲线

另外在保持 AGV 容量、起始位置和终止位置不变的前提下，对任务库中的任务节点数从50～300时计算时间做了测定，结果表明计算时间是一致的，这就是本算法的优越性所在. 它不仅可得到满意的优化结果，更重要的是它的实时性得到提高，而且在仓储系统确定后，实时性与任务库内的任务节点数无关.

另外，在固定其它条件不变，而使 AGV 的容量发生变化，结果同样表明，计算时间没有发生变化，以上计算时间的测量精度为10ms.

4 结束语

由于仓储的任务规划问题实质上是寻求一条执行任务的路线,使得 AGV 在该路线上移动时,执行各相关任务的代价最小.因此,本文从聚类分割技术出发,以关键路线为聚类分割的核心,提出了"云室效应"算法 CREA,使得任务链在趋于最优的同时,实时性提高到一个新的层次,特别是本算法与任务库内的任务节点数无关,而通常算法计算复杂度至少是与任务节点数成正比,因此本算法具有良好的时间鲁棒性.本算法可进一步用于求解 DTRP (dynamic traveling repairman problem),PDPTM(pickup and delivery problem with time windows)等问题.

参考文献

1 Chen C L, Lee C S G, Mcgillem C D. Task assignment and load balancing of autonomous vehicles in a flexible manufacturing system. IEEE Journal of Robotics and Automation, 1987, **Ra-3(6)**: 659~670.

2 Martine Labbe, Gilbert Laporte, Helene Mercure. Capacited vehicle routing on trees. Operation Research, 1991, **39(4)**: 616~621.

3 Savelsbergh M W P. An efficient implementation of local search algorithms for constrained routing problems. European Journal of Operational Research, 1990, **47**: 75~85.

4 Li C L, Levi D S, Desrochers M. On the distance constrained vehicle routing problem. Operation Research, 1992, **40(4)**: 790~799.

5 Edan Y, Flash T. Near-minimum-time task planning for fruit-picking robots. IEEE Transaction on Robotics and Automation, 1991, **7(1)**: 48~55.

6 Sedgewick R, Vitter J S. Shortest paths in euclidean graphs. Algorithmica, 1986, **1**: 31~48.

7 Skiscim C C, Golden B L. Computing K-shortest path lengths in euclidean networks. Networks, 1987, **17**: 341~352.

8 运筹学. 北京:清华大学出版社,1990.

"CLOUD EFFECTIVE" ALGORITHMS FOR TASK ASSIGNMENT OF WAREHOUSE SYSTEM

Chen Fangze　Zhang Bo

(*Department of Computer Science and Technology　Tsinghua University　Beijing*　100084)

Abstract　The task assignment of AGV is a sophisticated combinational optimal problem. It is NP complete. A new method, which is different from the general searching from task node to task sequence, is given in this paper. The new planning process goes from key route to the clustering based on key route then to task sequence. So the method is called "cloud room effect" algorithms. As a satisfied solution is given, the ability of real time of the algorithms is improved.

Key words　AGV, key line, collection, cutting, cloud room effect.