doi: 10.3969/j.issn.1006-1576.2011.10.008

流水线加工调度问题的神经网络算法

王新军，卿华，姚娇艳
（中国人民解放军空军第一航空学院航空修理工程系，河南 信阳 464000）

摘要：针对启发式算法求解调度问题时算法执行时间短，但计算结果较差的问题，提出一种基于目标增量的构造优化神经网络算法。通过引入一个加工时间为 0 的虚拟工作，利用相邻工件加工结束时间差矩阵，将求解无等待流水线加工调度问题的最小最大完工时间问题映射为 TSP 问题，建立构造优化神经网络模型，将流水线调度问题映射到神经网络上。实验结果证明：该算法在时间性能和结果的最优性方面较启发式算法 SA2、RAJ、GR 和目标增量法有较大提高，对于大规模问题该算法优势明显。

关键词：流水线；调度；构造优化神经网络
中图分类号：TP183 **文献标志码**：A

A Neural Network Algorithm for Flow Line Processing Scheduling Problem

Wang Xinjun, Qing Hua, Yao Jiaoyan
(Dept. of Aeronautical Repair Engineering, First Aeronautical Institute of PLA Air Force, Xinyang 464000, China)

Abstract: Aiming at algorithm execution time short but can not get good result problem of heuristic algorithm at solving scheduling, put forward a constructive-optimizer neural network (NN) algorithm based on object increment. Firstly, a dummy job with 0 processing time at any machine was introduced, a matrix denoting minimum finish times of adjacent jobs was created and the scheduling problem can then be transferred into a TSP problem, so the scheduling problem was mapped onto a constructive-optimizer neural network (NN). The simulation results based on the well known benchmarks show that the proposal outperforms the best know methods: SA2, RAJ, GR and objective increment method in most cases especially for large scale problems.

Keywords: flow shop; scheduling; constructive-optimizer neural network

0 引言

调度问题主要是根据产品制造需求合理分配产品制造资源，进而达到合理利用产品制造资源、提高企业经济效益的目的。无等待流水线调度问题是应用最为广泛的一类调度问题，理论上已经证明该问题是 NP（non-deterministic polynomial）难题[1]，要想用多项式时间复杂度的算法找到最优解是不可能的。因此，求解无等待流水线调度问题的近似最优解很有必要。目前的研究主要分为启发式方法和元启发式方法。目前最好的启发式算法有 NEH[2]算法、GR[3]算法和 RAJ[4]算法等。Aldowiasan 和 Allahverdi[5]提出了 6 个元启发式算法，其中最好的 2 个为基于模拟退火法的 SA2 和基于遗传算法的 GEN2 算法。李小平[6]等提出一种基于目标增量的新启发式算法，该算法在时间和最优性方面都优于 SA2 和 GEN2。另一方面，自从 Hopfield[7]等提出 Hopfield 神经网络（HNN）模型、Kohonen[8]提出自组织映射算法以来，神经网络算法已被认为是解决复杂组合优化问题的有效方法之一。M. Saadatmand-Tarzjan[9]等将 Hopfield 神经网络算法与 Kohonen 自组织映射算法相结合，提出一种新的神经网络算法：构造优化神经网络算法（constructive-optimizer neural network，CONN）用来解决 TSP 问题。因此，笔者利用李小平[6]等提出的相邻工件加工结束时间差矩阵，通过引入一个虚拟工作，将以最小化 MakeSpan 为目标的流水调度问题转化为等价的 TSP 问题，从而可以利用构造优化神经网络模型求解该无等待流水线调度问题。

文中用到了以下符号：n 为总工件个数；m 为当前已经排序的工件个数；J 为第 i 个工件；$D(J_i,J_j)$ 为工件 J_i 与 J_j 之间的加工结束时间差；x^l_i 为第 l 层第 i 个神经元的输出值；$x^l_{i,j}$ 为向量 x^l_i 第 j 个元素的值；Ψ_j 为当前已排序序列中的第 j 个工件；Q 为当前已排序的工件集合；R 为不包含在当前已排序序列中的工件。

1 问题分析

1.1 无等待流水线调度问题描述

设有 n 个工件 $\{J_1, J_2, \cdots, J_n\}$，要以相同顺序在 m 台机器上顺序加工，每个工件在某一时刻只能够在一台机床上加工，一台机床在某一时刻只能够加工一个工件。无等待条件要求工件在当前机器上完成

收稿日期：2011-06-01；修回日期：2011-07-01
作者简介：王新军（1978—），男，陕西人，硕士，讲师，从事人工神经网络、航空维修研究。

加工后马上进行下一个机器上的加工，为了满足这种无等待的约束条件，必要时要延迟某些工件在第一台机器上的开始加工时间。

1.2 相邻工件加工结束时间差矩阵的计算

李小平[6]等提出一种利用目标增量快速求解无等待流水调度问题的最小最大完工时间的启发式算法，该算法中使用了相邻工件加工结束时间差的最小值矩阵。设 S_{ij} 和 C_{ij} 分别表示工件 j 在机器 i 上的开始时间和结束时间，工件 i 在机器 j 上所需的加工时间为 t_{ij}。图 1 描述了某工作序列中相邻 2 个工件的加工结束时间差 D_{ij}。设 $\pi=(J_{[1]},J_{[2]},\cdots,J_{[n]})$ 表示一个加工序列，其中 $J[i]\in\{J_1,J_2,\cdots,J_n\}$ 表示序列中的第 i 个工件，虚线部分 J_j' 表示后续工件 J_j 在第一台机器上的开始时间等于前一工件 J_i 在最后一台机器上的结束时间的情形，显然这样安排必然能满足无等待的约束条件，但 Makespan 值太大。

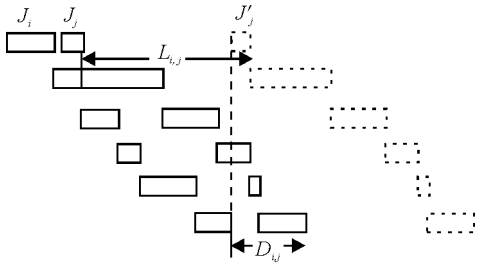

图 1 相邻工件的加工结束时间差

为了优化目标值 Makespan，该工件可以在保证满足无等待约束条件下向前移动。则相邻工件加工结束时间差对应的矩阵可计算如下：

$$D_{i,j}=C_{m,j}-C_{m,i}=C'_{m,j}-C_{m,i}-L_{i,j}=\sum_{h=1}^{m}t_{h,j}-L_{i,j}=\max_{k=1,\cdots,m}\{\sum_{h=k}^{m}(t_{h,j}-t_{k,i})+t_{k,j}\},\quad i\neq j,i\in\{1,2,\cdots,n\},j\in\{1,2,\cdots,n\} \tag{1}$$

$$C_{k,i}=S_{1,i}+\sum_{h=1}^{k}t_{h,i}(k=1,\cdots,m) \tag{2}$$

$$S'_{1,j}=C_{m,i}=S_{1,i}+\sum_{h=1}^{m}t_{h,i} \tag{3}$$

$$S'_{k,j}=S_{1,i}+\sum_{h=1}^{m}t_{h,i}+\sum_{h=1}^{k}t_{h,j}-t_{k,j} \tag{4}$$

$$L_{i,j}=\min_{k=1,\cdots,m}\{\sum_{h=k}^{m}t_{h,i}+\sum_{h=1}^{k}t_{h,j}-t_{k,i}-t_{k,j}\} \tag{5}$$

2 以最小化 Makespan 为目标的无等待流水线调度问题的神经网络算法

2.1 以最小化 Makespan 为目标的无等待流水线调度问题可转化为非对称 TSP 问题

证明 引入一个虚拟工作 J_{n+1} 作为工作序列的第一个或最后一个工件，该工件在任意机器上的加工时间都为 0，则工作序列可表示为 $\pi=(J_{[n+1]},J_{[1]},J_{[2]},\cdots,J_{[n]},J_{[n+1]})$，则无等待流水线调度问题的 Makespan 可表示为：

$$C_{\max}(\pi)=C_{Jn+1,m}=\sum_{i=1}^{n}D_{JiJi+1}$$

其中 D_{ij} 按照式 (1) 计算。显然，因为虚拟工件在任意机器上的加工时间为 0，故不会改变原问题的目标函数值。如果将工作序列中的每个工件看作空间中的一点，相邻工件加工结束时间差的最小值 D_{ij} 可看成两点之间的距离，则该问题变成一个非对称 TSP 问题。

2.2 以最小化 Makespan 为目标的 CONN 模型

图 2 为当前序列中有 3 个元素总共 6 个元素的 CONN 结构：

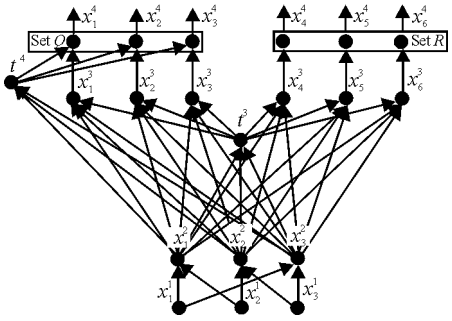

图 2 6 个元素的 CONN 结构

CONN 有 4 层神经元，其中，神经网络第 1 层的神经元记录当前已排序序列中的 m 个工件，故当前已排序序列可由第 1 层的神经元得到，即

$$S=[x^1{}_1,x^1{}_2,\cdots,x^1{}_m] \tag{6}$$

第 2 层的神经元记录由当前序列相邻工件构成的边，每个边用一个二维向量表示相邻的 2 个工件，可由第 1 层神经元值得到。由于添加了一个虚拟工件，整个序列可以构成一个闭合路径，故有 m 条边：

$$x^2{}_j=[x^2{}_{j1},x^2{}_{j2}]=[x^1{}_j,x^1{}_{j+1}] \tag{7}$$

第 3 层为边竞争层，每个神经元对应一个工件，记录第 2 层神经元等于该工件竞争后的获胜边。其中神经元 t^3 是一个阈值神经元，记录第 3 层其它神

经元对应的阈值构成的向量，阈值可由第 2 层的神经元得到：

$$t^3=[t^3_j]=[D(x^2_{j1},x^2_{j2})],\quad (j=1,2,\cdots,m) \tag{8}$$

每个神经元先计算 m 个输入所对应的激活值 v^3_{ij}：

$$v^3_{ij}=D(x^2_{j1},c_i)+D(c_i,x^2_{j2})-t^3_j,\quad (i=1,2,\cdots,n;\ j=1,2,\cdots,m)$$

最后竞争的获胜者为最小激活值对应的神经元。第 3 层神经元的输出值为一个二维向量，记录获胜的最小激活值以及获胜神经元：

$$x^3_i=[x^3_{i1},x^3_{i2}]=[\min_{j\in P_i}(v^3_{ij}),\arg(\min_{j\in P_i}(v^3_{ij}))],\quad (i=1,2,\cdots,n) \tag{9}$$

由于第 2 层神经元对应序列中的边，故获胜神经元对应当前序列中的一条边，P_i 表示不与 c_i 相邻的边：

$$P_i=\{j\mid j=1,2,\cdots,m;x^2_{j1}\neq c_i,\ x^2_{j2}\neq c_i\} \tag{10}$$

事实上，v^3_{ij} 表示工件 i 插入到 j 所对应的边造成当前序列长度的增加值。第 2 层神经元代表当前序列的所有边对神经元 i 竞争，获胜者对应将工件 i 插入该边所在位置后序列长度的增加值最小。

第 4 层为点竞争层，每个神经元对应一个工件。当 CONN 工作在构造状态时集合 R 对应的神经元工作，当 CONN 工作在优化状态时集合 Q 对应的神经元工作。其中神经元 t^4 是一个阈值神经元，记录第 4 层其它神经元对应的阈值构成的向量，阈值可由第 2 层的神经元得到：

$$t^4=\begin{cases}D(x^2_{j,1},x^2_{j,2})+D(x^2_{j+1,1},x^2_{j+1,2})-D(x^2_{j,1},x^2_{j+1,2}),\\ x^2_{j,2}=x^2_{j+1,1}=c_i\in Q\\ 0\qquad\qquad\qquad c_i\in R\end{cases} \tag{11}$$

该阈值表示当前序列中删除某个点后序列长度的减小值。第 4 层其它神经元的激活值 v^4_i 按式 (12) 计算：

$$v^4_i=x^3_{i,1}-t^4_i,\ i=1,2,\cdots,n \tag{12}$$

该激活值表示将工件 i 从当前位置删除并插入 $x^3_{i,1}$ 所指的边序列长度的增加值。由于 $x^3_{i,1}$ 所指的边是工件 i 在当前序列 m 条边中的最佳插入位置，故 v^4_i 给出将工件 i 从当前位置删除并插入序列最佳位置后序列长度的增加值。

当 CONN 工作在优化状态时，集合 Q 中的神经元之间相互竞争，获胜者为激活值最小的神经元：

$$\omega_{opt}=\arg(\min_{i\in Q}(v^4_i)) \tag{13}$$

此时神经元的输出值由下式给出：

$$x^4_i=\begin{cases}0, & i\neq\omega_{opt}\\ x^3_{i,2}\varphi(v^4_i), & i=\omega_{opt}\end{cases} \tag{14}$$

显然最多只有一个神经元输出值不为 0，不为 0 的输出值指出获胜神经元对应的工件应插入序列中哪条边所在的位置。若所有输出都为 0，此时 CONN 的一次优化运算结束，根据集合 R 是否为空决定 CONN 结束程序还是切换到构造状态运行。

当 CONN 工作在构造状态时，集合 R 中的神经元之间相互竞争，获胜者为激活值最小的神经元：

$$\omega_{cns}=\arg(\min_{i\in R}(v^4_i)) \tag{15}$$

此时神经元的输出值由下式给出：

$$x^4_i=\begin{cases}0, & i\neq\omega_{cns}\\ x^3_{i,2} & i=\omega_{cns}\end{cases} \tag{16}$$

2.3　以最小化 Makespan 为目标的构造优化神经网络 (CONN) 算法流程

构造优化神经网络模型包括构造和优化 2 种工作状态。应用网络优化算法之前，先用一定方法构造一个长度为 γ 的初始序列 S，然后将 S 交给 CONN 开始网络优化算法。本文中初始序列的构造采用 NEH[2] 初始序列的构造方法，根据每个工件在所有机器上的总加工时间降序排列。每个工件 i 可看作序列中的一个点，相邻两点构成一条边，该边的长度对应两工件的加工结束时间差。首先 CONN 工作在构造状态，向 S 中逐一添加工件扩展 S 中的工件个数 m，直到 $m=\gamma+\lambda$，λ 为 CONN 扩展部分执行一次要添加的工作数。然后 CONN 切换到优化状态，对序列 S 中的 m 个工件的位置进行优化。优化过程结束后，CONN 根据当前序列 S 是否已经包含所有工件决定是否结束网络优化程序。如果 $m<n$ 则 CONN 更新 $\gamma=m$ 并切换到构造状态，向 S 中逐一添加工件扩展 S 中的工件个数 m，开始新的循环。算法流程如图 3。

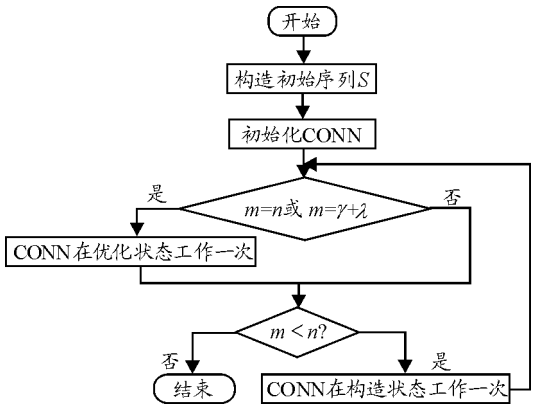

图 3　算法流程图

3　实验结果分析

在文献[6]中作者将所提出的启发式算法实验结果与目前最好的几个快速算法：SA2、RAJ 和 GR 相比较，文献[6]所提出的目标增量法在时间性能和结果的最优性方面均优于其它算法，故笔者仅仅将 CONN 算法与文献[6]所提出的目标增量法做比较。2 个算法都应用 Matlab 编写程序代码，在同一台 PC 机器上运行标准数据库 Taillard[10]中的 120 个实验数据。2 种算法对于前 60 个小规模问题和后 60 个大规模问题运行结果比较如下：

图 4 和图 5 的柱状图中，横轴表示小规模问题 ta001−ta060 和大规模问题 ta061−ta120 的实验数据序号，纵轴表示 CONN 算法得到的 Makespan 结果减去目标增量法得到的 Makespan 结果的差值。位于横轴上部的柱状图表示目标增量法优于 CONN 算法的问题，位于横轴下部的柱状图表示 CONN 算法优于目标增量法的问题。类似的图 6 和图 7 的柱状图中，横轴表示小规模问题 ta001−ta060 和大规模问题 ta061−ta120 的实验数据序号，纵轴表示 CONN 算法所耗费的时间减去目标增量法所耗费的时间的差值。所以横轴上部的柱状图表示目标增量法在时间性能上优于 CONN 算法，横轴下部的柱状图表示 CONN 算法在时间性能上优于目标增量法的问题。

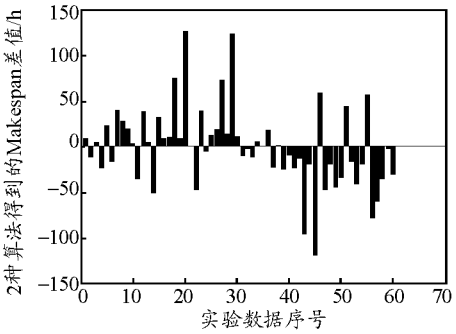

图 4　小规模问题 2 种算法结果的差值

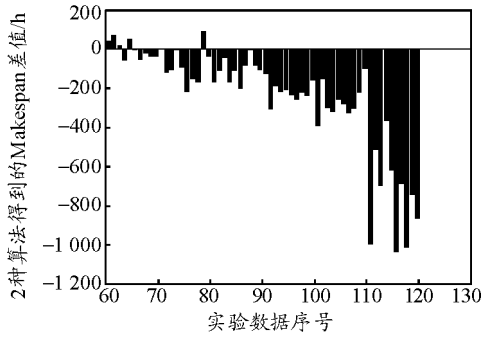

图 5　大规模问题 2 种算法结果的差值

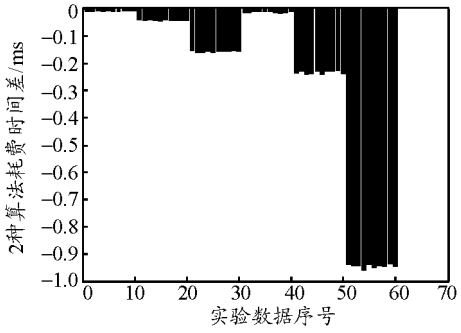

图 6　小规模问题 2 种算法耗费时间的差值

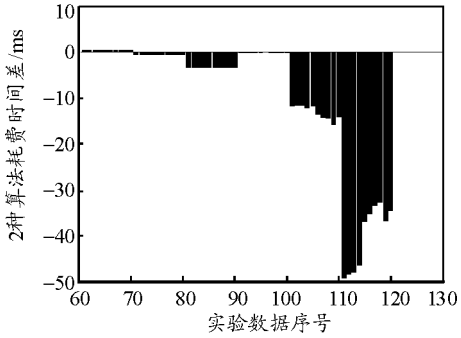

图 7　大规模问题 2 种算法耗费时间的差值

从图 4 可以看出，对于前面的小规模问题，2 种算法所得到的结果差异不大。从图 5 可以看出，柱状图的绝大部分都在横轴的下方，也就是说对于后 60 个大规模问题 CONN 算法明显优于目标增量法。图 6 和图 7 中柱状图的绝大部分都在横轴的下方，也就是说对于标准数据库中的绝大多数模问题，CONN 算法在时间性能上明显优于目标增量法。

4　小结

实验结果证明：对于绝大多数实验数据，该算法在时间性能和结果的最优性方面优于文献[6]中提出的启发式算法。笔者对于初始序列采用了类似 NEH 构造初始序列的方法，CONN 的训练算法和参数设置使用了文献[2]中的方法和数据，应用其它方法构造初始序列，以及改进 CONN 的训练算法和参数设置还有待于进一步研究。

（下转第 38 页）