摘　要

Hopfield 神经网络是神经网络发展历史上的一个重要发展阶段，它成功地解决了*TSP* 计算难题。*Hopfield* 神经网络是一种反馈型神经网络，它的稳定形态比前向型网络要繁杂得多。*Hopfield* 神经网络分为离散型和连续型两种网络模型，*Hopfield* 神经网络模型具有高效性和稳定性，但是*Hopfield* 神经网络算法是一种贪心算法，是通过寻找局部最优解来达到全局解，但是这个全局解不一定为全局最优解，所以本文尝试对此进行改进，以达到最优解，避免不足之处。本文介绍了*Hopfield* 神经网络在数学建模中的应用，运用*Hopfield* 神经网络算法求解智能RGV 的动态调度优化问题。

关键词： *Hopfield* 神经网络算法；*TSP* 问题；智能 RGV 动态调度。

目　　录

1　引言……P1
2　*Hopfield* 神经网络的基本理论……P1
2.1 离散型 *Hopfield* 神经网络算法的定义及特性…… P1
2.2 连续型 *Hopfield* 神经网络算法的定义及特性…… P3
2.3 *Hopfield* 网络当前的研究成果…… P5
3　*Hopfield* 神经网络在数学建模中的应用…… P6
3.1 *Hopfield* 神经网络求解 *TSP* …… P6
3.2 应用举例：智能 RGV 的动态调度优化研究…… P7
3.2.1 问题重述…… P8
3.2.2 问题分析…… P9
3.2.3 问题假设…… P10
3.2.4 符号说明…… P10
3.2.5 模型的建立与求解…… P12
3.2.6 检验模型的实用性和算法有效性…… P16
3.2.7 模型的评价与推广…… P16
4　*Hopfield* 神经网络的发展展望…… P17
致谢……P17
参考文献……P18
附录 A　*Hopfield* 神经网络模型代码…… P19

Hopfield 神经网络算法在数学建模中的应用

1 引　　言

Hopfield 神经网络是神经网络发展历史上的一个重要的里程碑。1982 年，美国加州理工学院物理学家 J. J. *Hopfield* 教授对 *Hopfield* 神经网络的相关问题进行研究，研究结果显示 *Hopfield* 是一种单层反馈神经网络。除此之外，1984 年，*Hopfield* 教授设计并研制了网络模型的电路，并成功地解决了 *TSP* 计算难题。*Hopfield* 神经网络是一种反馈型神经网络，它的稳定形态比前向型网络要繁杂得多。*Hopfield* 神经网络分为离散型和连续型两种网络模型，分别记作 *DHNN*（*Discrete Hopfield Neural Network*） 和 *CHNN* （*Continues Hopfield Neural Network*）。

Hopfield 神经网络算法主要用于解决 *TSP* 问题，但是现在还有很多方面都可应用 *Hopfield* 神经网络算法，例如许多由 *TSP* 所衍生的问题都可用 *Hopfield* 神经网络算法解决，但是需要将 *Hopfield* 神经网络模型改进，从而得到问题的答案。*Hopfield* 神经网络模型具有高效性和稳定性。通过找出所有路径的组合之后，再进行比较从而找到最佳路径来解决 *TSP* 问题，但是这种传统穷举法的计量工作量会随着维数的增加而大幅度增加，用 *Hopfield* 神经网络算法来解决 *TSP* 问题就可以避免这种情况，但是 *Hopfield* 神经网络算法是一种贪心算法，通过寻找局部最优解来达到全局解，但是这个全局解不一定为全局最优解，所以本文改进约束条件能量函数，达到最优解，避免不足。

2 *Hopfield* 神经网络的基本理念

Hopfield 神经网络分为离散型和连续型两种网络模型，分别记作 *DHNN*（*Discrete Hopfield Neural Network*） 和 *CHNN* （*Continues Hopfield Neural Network*）。*DHNN* 与 *CHNN* 的主要差别在于：*CHNN* 神经元激活函数使用 sigmord 函数，而 *DHNN* 神经元激活函数使用了硬极限函数。

2.1 离散型 *Hopfield* 神经网络算法的定义及特性

离散 *Hopfield* 神经网络（*DHNN*）：神经元的输出取 1 代表其为激活形态，取 0 代表其为抑制形态，二值神经元的计算公式如下，$u_j = \sum w_{ij} y_i + x_j$ 其中 x_i 为外部输入，并且有：$\begin{cases} y_i = 1, \text{当} u_i \geq 0 \text{时} \\ y_i = 0, \text{当} u_i \leq 0 \text{时} \end{cases}$

离散 *Hopfield* 神经网络是一个单层网络，有 n 个神经元节点，每个神经元节点的输出都能接到其它神经元节点的输入。各节点没有自反馈，每个节点都附有一个阀值。每个节点都可处于一种可能的状态（1 或－1），即当阀值比神经元所受的刺激大时，神经元就处于一种状态（比如-1），否则神经元就始终处于另一种状态（比如 1）。

一个 *DHNN* 的网络状态是输出神经元信息的集合。对于一个输出层是 n 个神经元的网络，其 t 时刻的状态为一个 n 维向量：$Y(t) = \left[y_1(t), y_2(t), ..., y_n(t) \right]^1$

因为 $y_i(t)$ 可以取值为 1 或 0，故 n 维向量 Y(t)有 $2n$ 种状态，即网络有 $2n$ 种状态。

如图所示：如果 *Hopfield* 神经网络是一个稳定网络，有 3 个神经元，则有 2^3 种状态。由图 2-1 可得：若在网络的其中一个端点上加上一个输入向量，则网络的状态会产生变化，即从正八面体的一个顶点转向另一个顶点，最终趋于稳定。

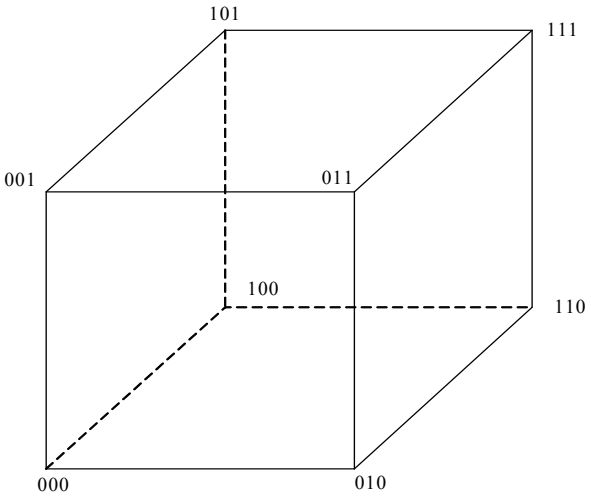

图 2-1　3 神经元 8 种状态的立方体模型

假设一个 *DHNN*，其状态为 $Y(t)$：

$Y(t) = \left[y_1(t), y_2(t), ..., y_n(t) \right]^1$

如果对于任何 $\forall t$，当神经网络从 $t = 0$ 开始，有初始状态 $Y(0)$。经过有限时刻 t，有：$Y(t + \forall t) = Y(t)$ 则称网络是稳定的。

Hopfield 神经网络稳定的充分条件：权系数矩阵 W 是对称矩阵，并且对角线元素为 0。无自反馈的权系数对称 *Hopfield* 神经网络是稳定的。

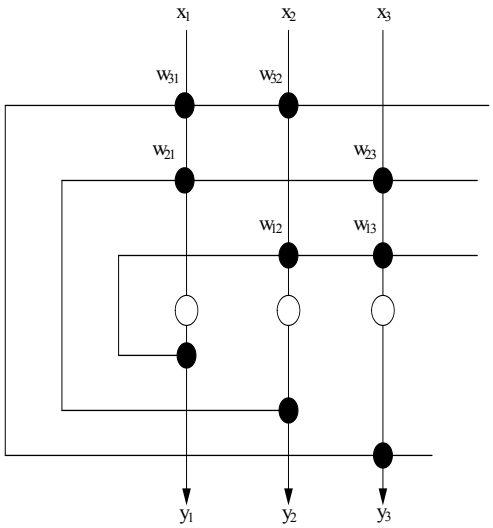

图 2-2　稳定的 *Hopfield* 神经网络

离散 *Hopfield* 神经网络有联想记忆的功能。对于 *Hopfield* 神经网络，用它作联想记忆时，先确定权系数的值，使得所记忆的信息在网络的 n 维正八面体的某一个顶角的能量最小。当网络的权系数确定之后，只要给网络加上输入向量，网络依旧可以完整输出所记忆的信息。

2.2　连续型 *Hopfield* 神经网络算法的定义及特性

连续 *Hopfield* 神经网络（*CHNN*）拓扑结构和 *DHNN* 的结构相同。不同之处在于其函数 g 不是阶跃函数，而是 S 形的连续函数。一般取 $G(u)=1/(1+e^{u})$

连续型 *Hopfield* 神经网络（*CHNN*）是联接简易的电子线路形成的。每个神经元都拥有一个输出值，这个输出值有着连续地时间变化。模拟神经元的 S 型单调输入——输出关系，即 $v_i = f_i(u_i)$

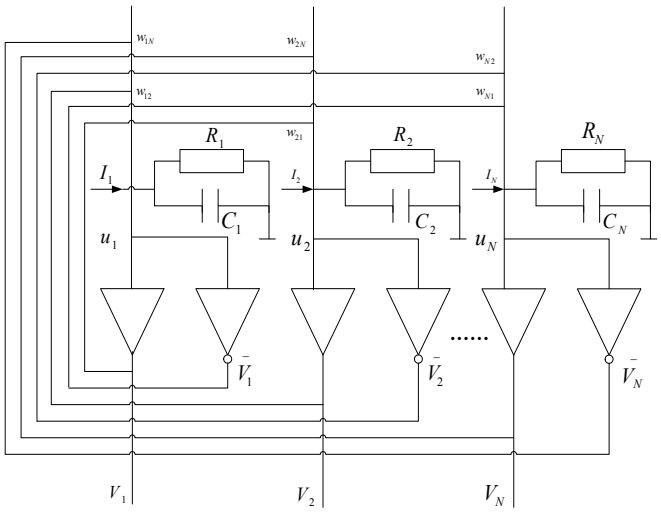

图 2-3　电子线路连接的连续 *Hopfield* 神经网络（a）

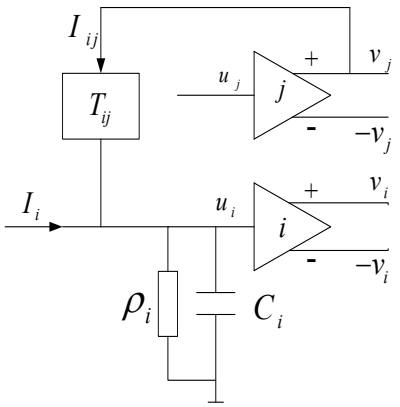

图 2-3　电子线路连接的连续 *Hopfield* 神经网络（b）

对于一个 N 节点的 *CHNN* 模型来说，其神经元状态变量的动态变化可用下述非线性微分方程组来描述

$$\begin{cases} C_i \dfrac{du_i}{dt} = \sum_{j=1}^{N} T_{ij} v_j - \dfrac{u_i}{R_i} + I_i \quad i = 1, 2, 3, \ldots, N \\ v_i = f_i(u_i) \end{cases}$$

能量函数定义为

$$E = -\frac{1}{2} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} T_{ij} v_i v_j - \sum_{i=1}^{N} v_i I_i + \sum_{i=1}^{N} \frac{1}{R_i} \int_0^{v_i} f^{-1}(v) dv$$

CHNN 的能量函数与物理意义上的能量函数不同，只是双方都表示网络状态的变化趋势。

定理：若作用函数 $f^{-1}(*)$ 是单调递增且连续的，则能量函数 E 是单调递减且有界的。

CHNN 用非线性微分方程描述，网络的稳定性通过其能量函数（又称 *Liapunov* 函数）的构造，并用 *Liapunov* 第二稳定性定理进行判断。

2.3 *Hopfield* 神经网络当前的研究成果

Hopfield 神经网络在解决 *TSP* 问题上颇有建树，1985年 *Hopfield* 和 *Tank* 用 *Hopfield* 神经网络求解 *TSP* 问题，并以此进行深入研究，最终确定了神经网络优化的新方法。

弱引力透镜具有对暗能量的状态方程严格限制的潜力。然而，这只有在剪切测量方法可达到精度所要求的水平的时候才能成为可能。通过将总的点扩散函数（PSF）利用数据的直接去卷积，采用表示 PSF 作为特普利茨矩阵的线性代数形式主义，使得我们可以通过应用 *Hopfield* 神经网络迭代方案来解决卷积方程。星系在去卷积图像中的椭圆率使用图像的自相关函数的二阶矩就能够得到测量。

在水质评价上，可以利用 *Hopfield* 神经网络的特性，并且在与其他水质评价方法对比之后， *Hopfield* 更有它的独到之处。在简化模型中，运用奇异设计值分解连接权重，使得运行效率提升。而且通过删除不重要的权重得到了更为简单的 *Hopfield* 神经网络结构。事实证明了该简化模型用于水质评价中的效率和可行性。

3 *Hopfield* 神经网络在数学建模中的应用

3.1 *Hopfield* 神经网络求解 *TSP*

对于一个城市规模为 N 的 *TSP* 问题来说，需要用一组 $N \times N$ 个神经元的输出表示其旅行次序。假设当城市规模 N =6 时，则需要的神经元数为 36 个，

其神经元输出矩阵如图 3-1 所示。

	1	2	3	4	5
A	0	1	0	0	0
B	0	0	0	1	0
C	1	0	0	0	0
D	0	0	1	0	0
E	0	0	0	0	1

图 3-1　神经元输出矩阵

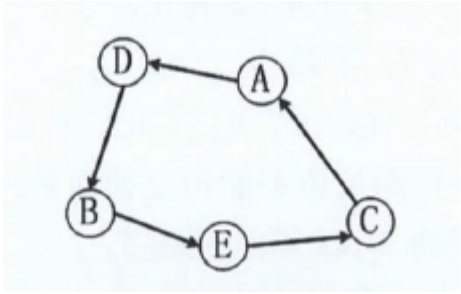

图 3-2　旅行路径

图 3-1 代表一组神经元的输出状态，城市在本次行程中的顺序用输出矩阵每一行中 1 的位置来表示，假设起点为 C，则图 3-1 表示的旅行顺序为 $C \to A \to D \to B \to E$，其旅行路径（见图 3-2）长度 $d = d_{CA} + d_{AD} + d_{DB} + d_{BE} + d_{EC}$，这里 d_{ij} 表示 i 市与 j 市间的距离。显然，在利用 *Hopfield* 神经网络求解 *TSP* 时，要求各神经元的输出矩阵在每一行每一列中有且仅有一个 1，便可确定唯一的一条旅行路线。

利用 *Hopfield* 神经网络求解 *TSP*，主要是将能量函数构造出来。此能量函数应包含两部分：一是对能量函数取得极小值时，各神经元输出矩阵在每一行每一列中有且仅有一个 1；二是将 *TSP* 旅行路径的长度表示出来。例如：

$$E=\frac{A}{2}\sum_{x=1}^{n}\sum_{i=1}^{n}\sum_{j=1,j\neq 1}^{n}V_{ij}V_{xj}+\frac{B}{2}\sum_{x=1}^{n}\sum_{i=1}^{n}\sum_{y=1,y=x}^{n}V_{xi}V_{yi}+$$

$$\frac{C}{2}\left[\sum_{x=1}^{n}\left(\sum_{i=1}^{n}V_{xi}-1\right)^2+\sum_{i=1}^{n}\left(\sum_{x=1}^{n}V_{xi}-1\right)^2\right]$$

$$\frac{D}{2}\sum_{x=1}^{n}\sum_{y=1,y\neq x}^{n}\sum_{i=1}^{n}d_{xy}V_{xi}\left(V_{y,i+1}+V_{y,i-1}\right)$$

式中 $A,B,C,D>0$,此时神经元(x,i)和(y,j)的连接权值 $w_{xi,yj}$ 为：

$$w_{xi,yj}=-A\delta_{xy}\left(1-\delta_{ij}\right)-B\delta_{ij}\left(1-\delta_{xy}\right)-C\left(\delta_{xy}-\delta_{ij}\right)-Dd_{xy}\left(\delta_{j,i+1}-\delta_{j,i-1}\right)$$

其中：

$$\delta_{xy}=\begin{cases}1, & \text{当i=j}\\ 0, & \text{其他}\end{cases}$$

权值给定时，随机设置 *Hopfield* 神经网络初始输入 u_i，网络将最终收敛于某个平衡状态。然而随着城市数量 N 的扩大，其 $N\times N$ 个神经元数量也对应扩大，此时 *Hopfield* 神经网络存在大量的平衡点,很难通过若干次计算得到全局最优解。而利用 *Hopfield* 神经网络求解 *TSP* 的优势在于：当城市数量适中时，*TSP* 的最优解一定在网络的平衡点中。所以，当 *TSP* 城市数量较多时，若可以先减少城市数量后再利用 *Hopfield* 神经网络求解，则能较快的得到原 *TSP* 问题的一个较优解。

3.2 应用举例：智能 RGV 的动态调度优化研究

随着科技的进步，越来越多的智能系统代替了繁琐的人工操作。现有一种由 8 台计算机数控机床（CNC）、1 辆轨道式自动引导车（RGV）、1 条 RGV 直线轨道、1 条上料传送带、1 条下料传送带等附属设备组成的智能加工系统。其中，RGV 是一种无人驾驶、能在固定轨道上自由运行的智能车，它能根据指令自动控制移动方向和距离，并自带一个机械手臂、两只机械手爪和物料清洗槽，能够完成上下料及清洗物料等作业任务。由于 8 台机床都可同时作业，且都能完成生料到熟料的加工转换，但轨道式自动引导车 RGV 仅有一辆，为了提高工厂的工作效率，保证在最短时间内得到最大的工作效益，即每班次传送带输送出更多加工完成的成料。因此，我们通过分析 RGV 的机械手爪工作顺序以及移动距离、各台 CNC 的加工情况、传送带物料的运送等一系列问题，在一般情况和出现故障的特殊情况下分别讨论一道工序和两道工序时加工系统的工作效率。

对于任务：针对一般情况下，要求我们分别给出一道工序和两道工序时加工系统的 RGV 动态调度模型以及相应的算法。由于 RGV 小车要从初始位置在轨道上对 8 台 CNC 进行上下料作业且最终回到初始位置，对此我们受“*TSP* 组合优化问题”的启发，建立非线性规划模型，运用此模型，我们可以很自然想到运用 *Hopfield* 神经网络算法求解。

3.2.1 问题重述

（1）问题的背景：

在这个快节奏高效率的生活时代中，为了提高各个工厂物料生产的效率，满足供给需求，人们引进智能加工系统，通过 RGV 这一有轨穿梭小车的运转、CNC 加工台的加工以及上下料传送带三者的配合，进行物料的加工合成、清洗和传送。在这个智能加工系统中，RGV 起着纽带的作用，它通过自带的机械手臂和两个机械手爪负责将生料进行搬移至加工台上，完成加工、清洗、送出等一系列动作，从而完成生料到成料的转换。RGV 具有移动速度快、运行平稳、停车位置准确且容易控制位置、可靠性高等一系列优点，但是 RGV 运行的先后顺序又决定了其运行的时间长短以及物料生产的效率。

（2）问题的提出：

针对一般问题进行研究，给出 RGV 动态调度模型和相应的求解算法。这里的一般问题指的是每台 CNC 在加工作过程中都正常运行，均不发生故障的情况。换言之，就是研究在每台 CNC 均正常运行，一道工序的物料加工作业（每台 CNC 安装相同的刀具加工）情况和两道工序的物料加工作（每个物料的第一和第二道工序分别由两台不同的 CNC 依次加工）情况。

利用表 3-2 中系统作业参数的数据分别检验模型的实用性和算法的有效性，即运用表 3-2 中的数据，带入问题一中所建立的数学模型以及给出的离散型 *Hopfield* 神经网络算法进行验证。

3.2.2 问题分析

（1）智能系统的运行状况及物料的加工传送过程

从智能加工系统中，我们可以了解到，智能加工系统通电启动后，RGV 在 CNC1#和 CNC2#正中间的初始位置（如下图 3-3），所有 CNC 都处于空闲状态。由于 RGV 为偶数编号的 CNC 一次上下料所需时间要大于为奇数编号的 CNC 一次上下料所需时间，所以 RGV 在刚刚通电启动时，先收到 CNC1#发出上料需求信号，随后它会自行确定该 CNC 的上下料作业次序，并依次按顺序为其上下料作业。根据需求指令，RGV 运行至需要作业的某台 CNC 处，同时上料传送带将生料送到该 CNC 正前方，供 RGV 上料作业。此时，RGV 中的机械臂前端上方手爪抓住上料传送带上的生料 A，机械臂下方空置的手爪抓住 CNC 台上未清洗的但已加工完成的熟料 B，旋转手爪，将生料 A 对准放置刚刚加工熟料 B 的加工台位置，然后即刻抓取清洗槽中的成料 C 放于下料传送带送出系统。同时将熟料 B 放置于 RGV 中的物料清洗槽，这就完成了一个完整的物料加工过程。如此循

环直至一个班次（8 小时）作业结束时停止。考虑到 RGV 上下手爪抓取的物料顺序以及移动过程中所耗费的时间，我们需要根据不同的情况进行分类讨论，从而建立相应的数学模型。

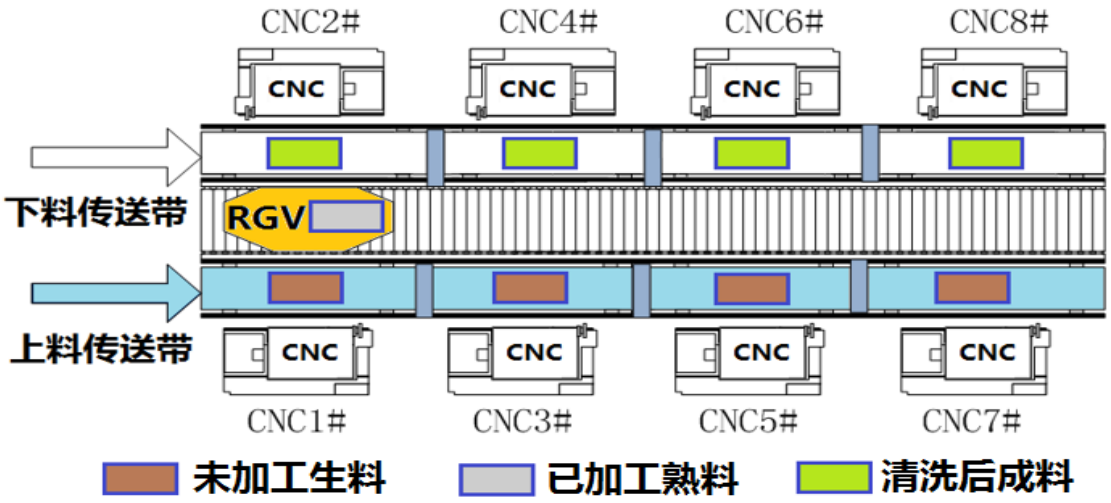

图 3-3　智能加工系统

（2）计算机数控机床 CNC 在加工过程中没有发生故障的情况

智能加工系统在通电后，先收到 CNC1#的上料需求信号，完成上料后，立即移动到相应位置给其它需要上料的 CNC 机器上料，我们对此情况进行模拟，发现当 RGV 给所有需要上料的机器 CNC 上料后并返回到初始位置（第一台上料的 CNC）时，RGV 仍需要一定的时间等待第一台 CNC 加工完成，由此联系到图论中的 *TSP* 优化问题，针对一道工序和两道工序，我们可以运用同一个模型来带入求解。同时，针对物料的状态（生料、熟料、成料）我们也可以看作是两道工序分别讨论。因此根据已知量可以建立相应的非线性规划模型。对于算法，我们先求局部的最优解，再通过各局部的最优解进行相加得到全局的最优解，因此，我们联想到应用 *Hopfield* 神经网络算法来解决问题。

（3）RGV 的调度策略和系统作业的效率

由于机器故障完全未知，而机器故障的概率已预知（故障发生概率为 1%），于是本文主要考虑在预调度的每个工序之前加入一定长度的故障排除时间，即我们选用的 RGV 的调度策略是基于机器故障概率分布的预调度算法。

系统的作业效率也可以相当于系统在实际过程中加工的熟料个数与原计划（理想状态）加工的物料个数的比值，因此，我们得考虑在可能发生故障的情况下，一个班次内正常运行的 CNC 机器数量以及它能产生的熟料个数。换句话说，

我们要考虑正常的一个班次内所有不能按理想状态生产熟料的机器数量，即机器的故障数量和因故障维修所耽误的机器数量。

3.2.3 问题假设

1.一般情况下，在人工智能系统作业的八小时之内不出现故障；

2.一般情况下，RGV 在智能加工系统通电启动后，先给 CNC1#上料，然后再给其它有发出需求信号的 CNC 加工台上料，并返回初始位置，且还需要等待 CNC1#加工成熟。

3.在使用 RGV 机械手臂旋转的过程中，没有消耗时间。

4.在可能发生故障的情况下，维修 CNC 加工台的时间均为 20 分钟。

5.RGV 为 CNC 一次上料和一次下料花费的时间相同。

3.2.4 符号说明

表 3-1　符号说明

符号	系统作业参数
i	物料
q	计算机数控机床 CNC
k，l	第k、l道工序
S_{ikq}	第i个物料的第k道工序在第q台 CNC 上起始加工时间
T_{ikq}	第i个物料的第k道工序在第q台 CNC 上加工的时间
C	8 个生料成料总花费时间
S_{ilq}	第i个物料的第l道工序在第q台 CNC 上起始加工时间

T_{ilq}	第i个物料的第l道工序在第q台CNC上加工时间
X_{i}	神经元
V_{xi}	神经元的状态
q_{xi}	第q台CNC的神经元
$v_{xi}, v_{xj}, v_{yi}, v_{yj}$	神经元的位置
l_{xy}	有效路径
Z_4	网络能量函数
Z	总能量
$\delta_{i,j}$， $\delta_{x,y}$，	激励函数
u_{xi}	神经元输入
v_{xi}	神经元输出
a_1	RGV移动一个单位所需时间
a_2	RGV移动两个单位所需时间
a_3	RGV移动三个单位所需时间
b	CNC加工完成一个一道工序的物料所需时间
d_1	CNC加工完成一个两道工序物料的第一道工序所需时间
d_2	CNC加工完成一个两道工序物料的第二道工序所需时间

t_1	RGV 为 CNC1#，3#，5#，7# 一次上下料所需时间

t_2	RGV 为 CNC2#，4#，6#，8# 一次上下料所需时间
c	RGV 完成一个物料的清洗作业所需时间

3.2.5 模型的建立与求解

（1）根据上述问题的假设，在一般情况下，我们设物料为i，此时的物料包括在上料传送带上的生料以及下料传送带上的成料；计算机数控机床 CNC 为q台，且$M=\{1,2,3,4,5,6,7,8\}$，$q\in M$；工序为 k 道或 1道，且$N=\{1,2\}$，$k\in N$且$l\in N$；第i个物料的第 k 道工序在第q台 CNC 上起始加工时间为S_{ikq}；第i个物料的第 k 道工序在第q台 CNC 上加工的时间为T_{ikq}；8 个生料成料总花费时间为C；第i个物料的第1道工序在第q台 CNC 上起始加工时间为S_{ilq}；第i个物料的第1道工序在第q台 CNC 上加工时间为T_{ilq}。建立如下非线性规划模型：

$$\min C=\max_{i\in N}\sum_{q=1}^{8}\sum_{k=1}^{n}(S_{ikq}+T_{ikq})$$

约束条件为：

当$l>k$时，$S_{ilq}-S_{ikq}\geq T_{ikq}$；

当$l<k$时，$S_{ikq}-S_{ilq}\geq T_{ilq}$；

当$j>i$时，$S_{jlq}-S_{ikq}\geq T_{ikq}$；

当$j<i$时，$S_{ikq}-S_{jlq}\geq T_{jlq}$；

此时$i,j\in N,q\in M$

（2）由上述问题假设得，当给一道或者两道工序的物料加工时，RGV 在给第一台 CNC 机器上料后，立即移动到相应位置给其它需要上料的 CNC 上料，

我们对此情况进行模拟，发现当 RGV 给所有需要上料的机器 CNC 上料后并返回到初始位置（第一台上料的 CNC）时，RGV 仍需要一定的时间等待第一台 CNC

加工完成，由此联系到图论中的TSP优化问题。而我们发现，问题的解决需要做的是优化时间，让时间达到最小值，若想优化时间就必须考虑 RGV 的有效路径。因此应用$Hopfield$神经网络算法给出相应的解答，当路径达到最小值时，总能量最小，时间也就得到最优解。

“*Hopfield* 神经网络”是指一种递归神经网络，同时又是一个单层网络，每个神经元节点的输出都能接到其它神经元节点的输入。各节点没有自反馈，在一般情况下，根据它的特点，我们可以先算出局部能量最小，通过各部分的累加从而保证全局能量的最小值。因此设神经元为X_i，神经元的状态为V_{xi}，满足

$$V_{xi}=\begin{cases}1, q_{X_i}\text{在第}i\text{个位置出现}\\ 0, q_{X_i}\text{在第}i\text{个位置不出现}\end{cases}；$$

$Z_1=\frac{A}{2}\sum_{x=1}^{\infty}\sum_{i=1}^{\infty}\sum_{j=1,j\neq i}^{\infty}V_{xi}V_{xj}, A>0$为常数, Z_1保证矩阵 V 的每行只有一个 1；

$Z_2=\frac{B}{2}\sum_{i=1}^{\infty}\sum_{x=1}^{\infty}\sum_{y=1,y\neq x}^{\infty}V_{xi}V_{yi}, B>0$为常数, Z_2保证矩阵 V 的每列只有一个 1；

$Z_3=\frac{C}{2}\left|\sum_{x=1}^{\infty}\sum_{i=1}^{\infty}V_{xi}-n\right|^2$，$C>0$为常数, Z_3保证矩阵 V 中为 1 的恰好n个，此时Z_3达到最小值$Z_3=Z_{3\min}=0$；

且Z_1、Z_2为局部最优解，Z_3表示全局最优解；

又因为利用*Hopfield*神经网络算法是为了考虑路径的合理性，我们引进神经网络能量函数$Z_4=\frac{D}{2}\sum_{x=1}^{\infty}\sum_{i=1}^{\infty}\sum_{y=1,y\neq x}^{\infty}l_{xy}v_{xi}\left(v_{y,i+1}+v_{y,i-1}\right), D>0$为常数, l_{xy}为有效路径的长度信息，若路径最佳时，Z_4达到最小值；若路径较好时，Z_4达到极小值，所以，构造总能量关于路径的函数关系式：

$$Z = Z_4 + Z_1 + Z_2 + Z_3$$

$$= \frac{D}{2}\sum_{x=1}^{\infty}\sum_{i=1}^{\infty}\sum_{y=1,y\neq x}^{\infty} l_{xy}v_{xi}(v_{y,i+1}+v_{y,i-1}) + \frac{A}{2}\sum_{X=1}^{\infty}\sum_{i=1}^{\infty}\sum_{j=1,j\neq i}^{\infty} V_{xi}V_{xj} + \frac{B}{2}\sum_{i=1}^{\infty}\sum_{x=1}^{\infty}\sum_{y=1,y\neq x}^{\infty} V_{xi}V_{yi} + \frac{C}{2}\left|\sum_{x=1}^{\infty}\sum_{i=1}^{\infty}V_{xi} - n\right|^2$$

将上式代入 *Hopfield* 神经网络电路标准能量函数公式得：

$$Z(\mathrm{t}) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}W_{ji}V_j(\mathrm{t}) - \sum_{i=1}^{n}V_i(\mathrm{t})I_i + \sum_{i=1}^{\mathrm{n}}\frac{1}{R_i}\int_0^{V_i(\mathrm{t})} g^{-1}(V)\,dV$$

且神经元 X_i 与 y_j 之间的导纳值为：

$$T_{xi,yj} = -A\delta_{x,y}(1-\delta_{i,j}) - B\delta_{i,j}(1-\delta_{x,y}) - C - Dl_{xy}(\delta_{j,j+1}+\delta_{j,j-1})(1-\delta_{x,y}) \quad \cdots\cdots①,$$

其中激励函数 $\delta_{i,j}$ 和 $\delta_{x,y}$ 定义为：$\delta_{i,j} = \begin{cases} 1, i = j \\ 0, \text{其它} \end{cases}$ ……②，

外激励函数为 I_{xi}=CN ……③，

将①②③分别代入网络状态方程得：

$$\begin{cases} C_{\mathrm{xi}}\dfrac{\mathrm{d}U_{\mathrm{xi}}}{\mathrm{dt}} = \dfrac{U_{\mathrm{xi}}}{R_{\mathrm{x}i}} - A\sum_{\mathrm{j}\neq\mathrm{i}}\mathrm{v_{xj}} - B\sum_{\mathrm{y}\neq\mathrm{x}}\mathrm{V_{xj}} - C\left(\sum_{\mathrm{x}}\sum_{\mathrm{y}} - N\right) - D\sum_{\mathrm{y}\neq\mathrm{x}}\mathrm{l_{xy}}(\mathrm{V_{y,\ i+1}} + \mathrm{V_{y,\ i-1}}) \\ V_{\mathrm{xi}} = \mathrm{g}(\mathrm{u}_{\mathrm{x}i}) \end{cases}$$

其中 $\mathrm{u}_{\mathrm{x}i}$ 为神经元的输入，$\mathrm{v}_{\mathrm{x}i}$ 为神经元的输出，两者关系为：

i） 当网络节点输出为连续型时，$\mathrm{V_{xi}} = \mathrm{f}(\mathrm{u_{xi}}) = \dfrac{1}{1+\exp(\frac{-2\mathrm{u_{xi}}}{\mathrm{u}_0})}$；

ii） 当网络节点输出为离散型时，$\mathrm{V}_{\mathrm{x}i} = \begin{cases} 1, & \mathrm{u}_{\mathrm{x}i} \geq 0 \\ 0, & \mathrm{u}_{\mathrm{x}i} < 0 \end{cases}$

iii）