

神经网络在计算系统软件抗衰重启技术中的应用研究

王 湛 郭成昊 刘凤玉 张 宏

(南京航空航天大学民航学院 南京 210094)

摘 要 将神经网络应用于计算系统的抗衰重启技术中,以实现细粒度的软件抗衰,可以更大程度地增强软件抗衰的智能化,提高抗衰效率及准确性,进一步降低抗衰开销,提高软件可靠性.判定模块重启相关性及模块可达集是实施细粒度软件抗衰策略的关键环节.文中结合神经网络工作原理,构建了判定模块间重启相关度及模块可达集的神经网络结构模型.该模型根据软件系统中模块间的控制、调用及数据访问关系,通过分析模块间的耦合程度和重启相关性的相关理论及其之间的关系,制定模块重启相关度和模块可达集的判定算法,最终完成系统模块间重启相关度及模块可达集的判定任务,从而为实现智能化细粒度软件抗衰提供支持.

关键词 软件抗衰;抗衰粒度;重启相关度;神经网络

中图法分类号 TP311

Research on the Application of Artificial Neural Network in the Fine-Grained Software Rejuvenation of Computing System

WANG Zhan GUO Cheng-Hao LIU Feng-Yu ZHANG Hong

(College of Civil Aviation, Nanjing University of Aeronautics and Astronautics, Nanjing 210094)

Abstract In order to improve the software availability and reliability, intelligentize the software rejuvenation and boost up its veracity and efficiency, the rejuvenation granularity would be finer and artificial neural network would be applied. The key step of fine-grained software rejuvenation is to determine restart dependence between the modules. This paper researches the principium of artificial neural network, puts forward the model of artificial neural network which determines the degree of restart dependence between modules and reachable set of each module finally. Based on the coupling relation between modules of software system, this model analyzes the connection between restart dependence and coupling relation, sets down the arithmetic to calculate the degree of restart dependence between modules and reachable set of each module; so that the intelligent software rejuvenation with fine rejuvenation granularity is supported.

Keywords software rejuvenation; rejuvenation granularity; degree of restart dependence; artificial neural network

1 引 言

软件老化是指在软件的长期不间断运行过程

中,由于系统内存占用和泄漏、未释放的文件锁、数据更新不及时、存储空间碎片以及舍入误差的累积等原因而导致的软件性能的衰退^[1-3].软件老化最终导致软件的失效.为了对抗软件老化,人们提出了软

件抗衰 (Software Rejuvenation, SR)^[2] 的思想. SR 是当软件性能衰退到一定程度时, 终止程序的继续运行, 重启系统以清理其内部状态 (如进行垃圾收集、刷新操作系统内核表、重新初始化内部数据结构等), 从而释放操作系统资源, 使软件性能得到恢复^[2-3].

软件抗衰是一种对软件老化现象的积极反应, 是在软件故障发生前采取的重启策略. 许多已知和未知的软件故障都可以通过软件整体或部分的重启排除. 加利福尼亚大学伯克利分校和斯坦福大学合作开展的面向恢复的计算 (Recovery Oriented Computing, ROC) 的研究中提出的递归重启 (Recursive Restartability, RR) 技术^[4] 和 Candea 提出的微重启 (Microreboot) 技术^[5] 是预先建立一棵重启树 (restart tree), 当执行重启操作时, 从重启树最底层的模块开始重启, 性能不能恢复则上推一级, 执行更大范围的重启. 这两种技术在软件开发之初即遵循了模块间松耦合的原则, 使得一个模块的重启不会影响其它模块的正常工作.

为了执行细粒度的 SR, 提高抗衰效率, 进一步提高软件抗衰技术的智能化, 本文借鉴文献^[4-5] 的思想, 首先制定了软件系统模块交互图的统一标准, 定义了模块耦合度, 并通过分析耦合度及重启相关性间的关系, 给出了系统任意非松耦合模块间重启相关度及模块可达集的判定方法; 同时结合神经网络的特性和工作原理, 给出了判定模块重启相关度及模块可达集的神经网络结构模型, 定义了各神经网络层及其输入输出向量, 给出了各层的作用及工作原理. 由此从根本上实现了智能化细粒度的软件抗衰, 全面提高了抗衰技术的效率及准确性, 有效地节约了抗衰成本, 提高了软件可靠性.

2 模块重启相关度及模块可达集的判定算法

模块是完成软件系统子功能的可管理的一段程序或算法. 在软件体系中, 它既可以表示传统结构化软件系统中的程序, 又可以表示面向对象软件系统中的对象, 以此来表示软件体系中各个组件的功能情况^[5].

2.1 模块耦合性与重启相关性

软件系统中各模块间并非完全相互独立的, 它们之间连接的紧密程度称为耦合性. 模块之间的连接越紧密, 联系越多, 耦合程度越高. 根据模块间交换数据的情况, 通常将耦合性分为 4 类: 数据耦合、

控制耦合、公共耦合和内容耦合.

如果模块间通过接口的参数表交换信息数据, 称为数据耦合; 若上层模块调用下层模块完成指定功能; 若调用关系是通过参数传递来实现的, 称为控制耦合. 若模块间通过一个公共环境进行数据交换, 称为公共耦合. 这个公共环境可以是全局变量、全局数据结构、通信缓冲区和数据 (库) 文件等. 若模块直接进入另一个模块中存取数据或使用服务, 或存在双向调用关系, 称为内容耦合. 以上 4 种耦合的耦合程度依次增加, 其中内容耦合的适性最差. 如果模块间的联系只是通过上层模块的控制和调用来实现, 则视其为非直接耦合. 本文将采取直接交互方式而发生数据、控制、公共、内容 4 种耦合关系的模块称为直接耦合模块.

一个模块重启时, 可能导致其它模块的故障或错误, 称为模块间的重启相关性. 模块间重启相关性取决于它们的耦合程度. 对应于模块间的几种耦合性, 将重启相关性分为 4 类: 相互独立、功能相关、状态相关和功能双向相关. 定义如下.

定义 1. 若模块 A 调用模块 B , 则 A 与 B 功能相关, 记为 $A \rightarrow B$.

性质 1. 功能相关的传递性: 若 $A \rightarrow B, B \rightarrow C$, 则 $A \rightarrow C$.

定理 1. 若模块 A 与 B 功能相关, 则 A 重启, B 同时重启, 但 B 重启时, A 不一定重启.

假定 $A \rightarrow B$, 则模块 A 重启时, B 必须同时重启, 如若不然, 被调用的模块 B 不能与调用模块 A 的状态保持一致; 而当模块 B 重启时, A 则不一定重启, 因为 B 并不向其上层模块 A 传递控制和调用参数, 不会影响其状态.

定义 2. 若 $A \rightarrow B$, 且 $B \rightarrow A$, 则 A 与 B 功能双向相关, 记为 $A \leftrightarrow B$.

性质 2. 功能双向相关具有交换性和传递性. 若 $A \leftrightarrow B$, 则 $B \leftrightarrow A$; 若 $A \leftrightarrow B, B \leftrightarrow C$, 则 $A \leftrightarrow C$.

定理 2. 若模块 A 与 B 功能双向相关, 则其中一个模块重启, 另一模块同时重启. 即 A 与 B 总是同时重启.

假定 $A \leftrightarrow B$, 则 $A \rightarrow B$ 且 $B \rightarrow A$, 此时无论重启模块 A 还是 B , 由定理 1 知必同时重启另一模块, 否则不能保持状态一致.

定义 3. 若 A 与 B 有数据或状态共享, 则 A 与 B 状态相关, 记为 $A \wedge B$.

性质 3. 状态相关具有交换性和传递性. 若 $A \wedge B$, 则 $B \wedge A$; 若 $A \wedge B, B \wedge C$, 则 $A \wedge C$.

定理 3. 若模块 A 与 B 状态相关,则其中一个模块需重启,另一模块同时重启. 即 A 与 B 总是同时重启.

假定 $A \wedge B$,则模块 A, B 总是同时重启,因为 $A(B)$ 间接使用了 $B(A)$ 的数据,如果不同时重启,会导致数据不一致或数据冲突.

定义 4. 若模块 A 与模块 B 既非功能相关也非状态相关,则 A 与 B 相互独立,记为 $A \oslash B$.

性质 4. 相互独立的交换性:若 $A \oslash B$,则 $B \oslash A$.
在数据耦合和非直接耦合的情况下,相应的模块被认为相互独立. 相对独立不存在传递性.

当两模块非直接耦合时,判定它们的重启相关性需要考虑其间经过的所有模块之间的重启相关性.

2.2 重启相关度的判定

对于模块间重启相关性,需要一个确定的数值来对此衡量,由此引入重启相关度的概念.

定义 5. 一个模块 A 重启与另一模块 B 重启的相关程度称为重启相关度 $D[A \cdot B]$,取值为 $\{0, 1, -1, 2, 3, 4\}$. 具体地, $D[A \oslash B] = 0, D[A \rightarrow B] = 1, D[A \wedge B] = 2, D[A \leftrightarrow B] = 3$. 另外定义 $D[A \cdot A]$ 为模块 A 自身重启相关度,且 $D[A \cdot A] = 4$. 特别地,若 $A \rightarrow B$,则记 B 与 A 的重启相关度为 $D[B \cdot A] = -1$.

由上文,根据重启相关类型可以确定重启相关度,同样地,由重启相关度可以确定重启相关类型. 即 $D[A \cdot B] = 0/1/2/3 \Leftrightarrow A$ 相对 B 独立/ A 与 B 功能相关/ A 与 B 状态相关/ A 与 B 功能双向相关.

约定 1. 当两个模块之间存在多种重启相关性时,取重启相关度最高的相关性. 当两个模块之间存在多种耦合时,取最高的耦合. 耦合程度越高,模块间连接关系越紧密,重启相关度也越高. 这样可以保证重启的完善性和有效性.

定义 6. 若模块 A 与 B 功能相关,认为从 A 可达 B ,但从 B 不可达 A ;若模块 A 与 B 状态相关或功能双向相关,认为 A 与 B 相互可达;若模块 A 与 B 相互独立,认为 A 与 B 相互不可达. 反之亦然.

定义 7. 若模块 A 与 K 非直接连接,但 A 经若干模块可达 K ,则认为从 A 到 K 间有一条可达路径 $R[A \sim K]$,经历的模块以“—”连接.

约定 2. 若从模块 A 到 K 存在可达路径 $R[A \sim K] = A - K_1 - K_2 - \dots - K_n - K$,则 A 与 K 的重启相关度为 $D[A \cdot K] = \min \{D[A \cdot K_n], D[K_n \cdot K]\}$.

推论 1. 若模块 A 与 K 间存在可达路径

$R[A \sim K]$,但不存在可达路径 $R[K \sim A]$,则 A 与 K 功能相关.

推论 2. 若模块 A 与 K 间既不存在可达路径 $R[A \sim K]$,也不存在可达路径 $R[K \sim A]$,则 A 与 K 相互独立.

推论 3. 若模块 A 与 K 间既存在可达路径 $R[A \sim K]$,又存在可达路径 $R[K \sim A]$,则 A 与 K 或状态相关或功能双向相关.

对推论 1 进行证明,用反证法. 若模块 A 与 K 非功能相关,则可能是其它 3 种情况:相互独立、状态相关、功能双向相关. 若模块 A 与 K 相互独立,按定义 6, A 与 K 相互不可达,即不存在可达路径 $R[A \sim K]$,与假设矛盾. 若模块 A 与 K 状态相关或功能双向相关,由定义 7,模块 A 与 K 相互可达,再由定义 7,两模块间必存在一条可达路径 $R[K \sim A]$,与假设矛盾. 因此模块 A 与 K 必为功能相关. 对于推论 2,3,可用类似方法推出.

2.3 重启可达集的判定

定义 8. 若从模块 A 到模块 K 存在可达路径 $R[A \sim K]$,则 K 为模块 A 的可达模块. A 的所有可达模块构成 A 的可达模块集,简称可达集. 用 $S[A]$ 表示. 执行重启操作时,可达集作为重启策略执行的根本依据,是执行软件抗衰的首要前提.

定理 4. 模块 A 与其任意可达模块 K 的重启相关度 $D[A \cdot K] > 0$.

若 K 为模块 A 的可达模块,则存在可达路径 $R[A \sim K]$,由推论 1、3 得模块 A 与 K 间至少为功能相关,据定义 5, $D[A \cdot K] > 0$.

- 由此要确定各模块的可达集,需以下步骤.
1. 确认初始模块和终结模块. 因为可达路径是单向的.
 2. 查找从初始模块到终结模块的所有可达路径.
 3. 按每一条可达路径求初始模块和终结模块间的重启相关度.
 4. 确定初始模块和终结模块的最终重启相关度及重启相关性.
 5. 查出与起始模块间的重启相关度大于零的模块,放入起始模块可达集中.

3 神经网络结构模型的制定

人工神经网络是模仿脑细胞结构和功能、脑神经结构以及思维处理问题等脑功能的新型信息处理系统. 目前神经网络在处理模式识别和最优化问题方面已趋于成熟.

3.1 神经网络在重启技术中的可用性

随着计算系统规模及复杂性的不断提升,系统模块间重启相关度和模块可达集的准确判定成为消耗抗衰时间及成本的主要方面,它会随着系统规模的增大而呈指数增长.为了节约抗衰成本及提高抗衰准确度,对模块重启相关度和模块可达集智能化判定策略的研究势在必行.

分析神经网络在处理模式识别问题和最优化问题方面的原理可确定其判定模块重启相关度和模块可达集中的高度可用性.

确定直接耦合模块间的耦合程度是判定任意模块间的重启相关度前提,而最直接清晰的表示模块间耦合关系的工具是软件体系结构图,故将其作为神经网络的输入向量,首先将其标准统一化:设两模块 A, B ,若 A 通过参数传递来调用 B 完成指定功能,则由图 1 表示;若 A 直接进入 B 中存取数据或使用服务,或存在双向调用关系,则由图 2 表示;若 A, B 分别通过一个公共环境进行数据交换,则由图 3 表示;如果 A, B 间通过接口的参数表交换信息数据,则由图 4 表示;如果 A, B 间未直接发生交互作用,则 A, B 无连接.



图 1

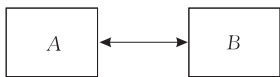


图 2

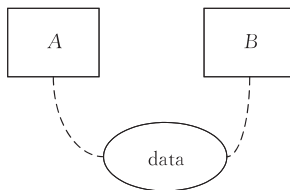


图 3

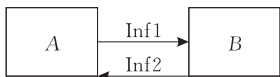


图 4

3.2 神经网络模型的确定

根据上文模块重启相关度及模块可达集的判定算法的相关步骤,分析各不同类型神经网络的特点,将网络结构构建为 5 层,以完成相应的计算任务.

第一层网络层选用概率神经网络(Probabilistic Neural Networks, PNN),用于判定软件系统直接耦合模块间的重启相关度.该类型网络训练容易、收敛速度快,从而非常适用于实时处理;当其用于检测和模式分类时,可以得到贝叶斯最优结果.由于随着系统运行状态的不同,系统直接耦合模块的判定过程具有很强的实时性,从而确定了 PNN 在该任务中的可用性.

此时将软件系统模块交互图作为输入向量,并将

交互图及图中所有交互类型编号,若两模块非直接耦合,交互类型设定为 0.此层网络结构如图 5 所示,其中 k 为输入向量类别数目, $a_i^1 = \text{radbas}(\| \mathbf{IW}_i^{1,1} - g \| b_i^1)$, a_i^1 为 \mathbf{a}^1 的第 i 个元素; $\mathbf{a}^2 = \text{compet}(\mathbf{LW}^{2,1} - \mathbf{a}^1)$; $\mathbf{IW}_i^{1,1}$ 为权值矩阵 $\mathbf{IW}^{1,1}$ 的第 i 行向量.假定交互图有 n 个模块,定义 P_n^2 组四维输入、输出向量 $\mathbf{P}_1, \mathbf{T}_1$,其元素为 $[f, s, d, r]$,其中 f 为交互图类型号,设定 s 为起始模块, d 为终止模块;在输入向量 \mathbf{P}_1 中设定 r 为 s, d 交互类型编号,对应的期望输出向量 \mathbf{T}_1 中 r 为 s 与 d 的重启相关度.由此得到交互图中直接耦合模块重启相关度.此时通过计算得到 $k=6$.

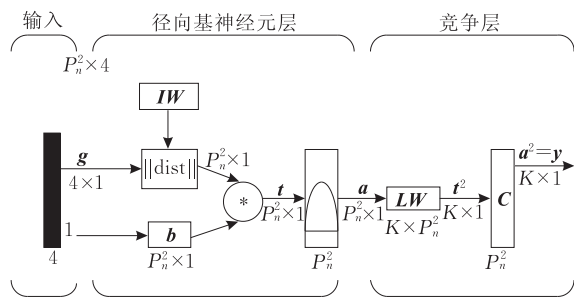


图 5

同样的原理,第 2 层仍选用 PNN 网络.此时将第 1 层中的 \mathbf{T}_1 作为输入向量 \mathbf{P}_2 输入第 2 层网络层,计算得知 k 取值为 3.此层将已得的直接耦合模块间的重启相关度加以对应改变,以方便利用推论 1~3 来判定非直接耦合模块间的重启相关度.特别地,在推论 3 中,若两模块相互可达,则为功能双向相关或状态相关,为确定一一对应的关系,必须利用该网络层将已得的直接耦合模块重启相关度作对应转换,完成初步判定工作.此时网络的输入向量 $\mathbf{P}_2 = \mathbf{T}_1$,此时期望输出向量 \mathbf{T}_2 与 \mathbf{P}_2 对应关系如下:

\mathbf{P}_2	\mathbf{T}_2
$(f, s, d, 0)$	$(f, s, d, 0)$
$(f, s, d, 1)$	$(f, s, d, 1)$
$(f, s, d, -1)$	$(f, s, d, 0)$
$(f, s, d, 2)$	$(f, s, d, 2)$
$(f, s, d, 3)$	$(f, s, d, 1)$
$(f, s, d, 4)$	$(f, s, d, 0)$

得到 \mathbf{T}_2 后,即得简单的模块有向图,将顶点编号为 (V_1, V_2, \dots, V_m) .

第 3 层网络层选用离散的 Hopfield 神经网络(Discrete Hopfield Neural Network, DHNN),用以计算非直接耦合模块间的重启相关度. DHNN 用于联想记忆有两个突出的特点:记忆是分布式的,联想是动态的.随着复杂性的增强,整个系统从整体上呈

现出分布式特性的,且随着系统运行状态的不同系统模块间的交互具有很强的动态性.由此可确定 DHNN 在该层的可用性.

此时将判定非直接耦合模块间重启相关度问题转化为有向图最短路径计算问题.该层神经元数量为 n^2 ,其网络模型结构如图 6 所示,其中 $\mathbf{a}^1(k) = \text{satlins}(\mathbf{LW}^{1,1}\mathbf{a}^1(k-1) + \mathbf{b}^1)$;且当 $k=0$ 时, $\mathbf{a}^1(0) = \mathbf{p}$,其中 \mathbf{p} 为网络的初始条件.对于有向图 $G=(V, E)$,模块数为 n ,边数为 m ,模块节点与支路间的关联用一 $n \times m$ 阶增广矩阵表示: $\mathbf{A}_a = [a_{ij}]$; a_{ij} 可取值 1, -1, 0, 取值为 1 时,表示支路 j 与节点 i 关联,且它的方向背离该节点;取值为 -1 时,表示支路 j 与节点 i 关联,且它的方向指向该节点;取值为 0 时,表示支路 j 与节点 i 不关联.设定 $P_{sd} = V_1, V_2, \dots, V_m$,表示起点为模块 s 、终点为模块 d 的一条路径,其中各决策变量 V_i 的取值为 0 或 1,当 $V_i=0$ 时,表示该支路不在此路径上,当 $V_i=1$,表示该支路在此路径上.由此可得,结构中 \mathbf{p} 为

$$\min: Z = \sum_{i=1}^m \mathbf{C}_i V_i = \mathbf{C} \mathbf{V}^T \quad (1)$$

其中 $\mathbf{C} = [C_1, C_2, \dots, C_m]$ 为 m 阶行向量,各元素代表各支路长度,即上一网络层输出的 r 值.

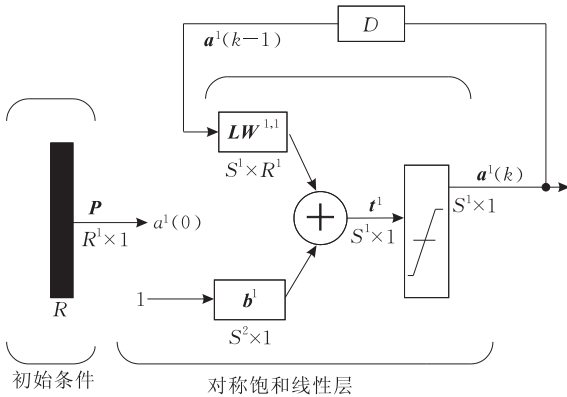


图 6

$$\text{s. t. } \sum_{i=1}^m a_{si} v_i = 1 \quad (2)$$

$$\sum_{i=1}^m a_{di} v_i = -1 \quad (3)$$

$$V_i \in 0, 1, i = 1, 2, \dots, m \quad (4)$$

由此其目标函数为

$$I = \rho_1 \sum_{i=1}^m C_i V_i + \rho_2 \sum_{i=1}^m V_i^2 (V_i - 1)^2 + \rho_3 \left(\sum_{i=1}^m a_{si} v_i - 1 \right)^2 + \rho_4 \left(\sum_{i=1}^m a_{di} v_i + 1 \right)^2$$

可得网络动态方程为

$$\left\{ \begin{aligned} \Delta u_i &= -k_i \frac{\partial E(v_1, v_2, \dots, v_m)}{\partial v_i} = \\ &= \sum_{i=1}^m \rho_1 c_i + \sum_{i=1}^m 2\rho_2 (2V_i^3 - 3V_i^2 + V_i) + \\ &+ 2\rho_3 \sum_{i=1}^m a_{si} \left[\left(\sum_{i=1}^m a_{si} V_i - 1 \right) \right] + \\ &+ 2\rho_4 \sum_{i=1}^m a_{di} \left[\left(\sum_{i=1}^m a_{di} V_i + 1 \right) \right], \\ V_i &= f(u_i) = \frac{1}{2} \left[1 + \tanh \left(\frac{u_i}{u_0} \right) \right]. \end{aligned} \right.$$

选择适当的参数 $(\rho_1, \rho_2, \rho_3, \rho_4)$ 和初值 u_0 , 按上式代入直到收敛为止. 给定起始模块 s 和终结模块 d 后, 此网络层计算其最短路径及长度, 将其作为输出向量 \mathbf{T}_3 .

参照前面分析的 PNN 网络的特点, 第 4 层网络层选用 PNN 网络, 将最终判定系统任意模块的重启相关度. 设定输入向量 \mathbf{P}_4 为 \mathbf{P}_n^2 组 $2m+5$ 维向量, 其元素表达为 $(f, s, d, V_{s1}, V_{s2}, \dots, V_{sm}, Z_1, V_{d1}, V_{d2}, \dots, V_{dm}, Z_2)$, s, d 为模块号; $V_{s1}, V_{s2}, \dots, V_{sm}$ 表示从 s 到 d 的最短路径, $V_{d1}, V_{d2}, \dots, V_{dm}$ 表示从 d 到 s 的最短路径. Z_1, Z_2 则分别表示这两条路径的长度, 此时若: 1) $Z_1 = Z_2 = 0$, 则 $D[s \cdot d] = 4$; 此时 $s = d$; 2) Z_1 有非零解且 Z_2 无解, 则 $D[s \cdot d] = 1$; 3) Z_2 有非零解且 Z_1 无解, 则 $D[s \cdot d] = -1$; 4) Z_1, Z_2 有非零解, 此时若 $Z_1 = Z_2 = \sum_{si=1}^m V_{si} = \sum_{di=1}^m V_{di}$, 则 $D[s \cdot d] = 3$; 否则 $D[s \cdot d] = 2$. 输出向量向量 \mathbf{T}_4 为 \mathbf{P}_n^2 组四维向量 (f, s, d, r) , 其中 r 为模块 s 与 d 的重启相关度, 计算得 k 取值为 6.

第 5 层网络层选用 LVQ 网络. LVQ 网具有结构简单、学习速度快、分类稳定可靠且容错性好的优点, 采用 LVQ 网络进行分类决策及表达是合适的. 完成系统任意模块间的重启相关度的判定任务后, 模块可达集的判定可被认为是一项机械分类过程, 结合上述 LVQ 网络的特点, 可确定其在该层的可用性.

此时将 \mathbf{T}_4 作为第 5 层的输入向量 \mathbf{P}_5 , 其网络结构如图 7, 其中

$$\begin{aligned} t_i^1 &= -\| \mathbf{I} \mathbf{W}^{1,1} - \mathbf{P} \|; \mathbf{a}^1 = \text{compet}(t^1); \\ \mathbf{a}^2 &= \text{purelin}(\mathbf{LW}^{2,1} \mathbf{a}^1). \end{aligned}$$

此时竞争层按起始模块不同将输入向量分为 n 类. 再通过线性层按 r 是否大于零, 最终可得到系统各模块可达集.

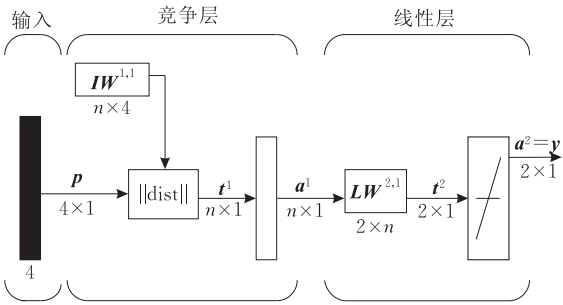


图 7

4 实验分析

实验环境:OS 为 Windows XP,CPU 为 P4,主频为 2.8GHz,主存为 512MB.以 Matlab 为仿真语言,结合 Visual C++实现神经网络模型.由于此模型为多层网络,故采用无导师训练算法与有导师训练算法结合的方法来对模型进行训练,用有导师训练解决输出层按系统要求给出指定的输出结果的问题,用无导师训练解决网络隐藏层的理想输出未知问题.图 8 是银行 ATM 系统现金提取用例对象图,其中 Object a: BankCard Reader, Object b: Customer Panel, Object c: Cash Dispenser, Object d: Cash Container, Object e: Receipt Printer, Object f: Card Transaction, Object g: Cash Withdrawal, Object h: Bank System,Actor A: Bank customer, Actor B: Bank System.

首先对图 8 制定消息类型编号:1 为调用消息;

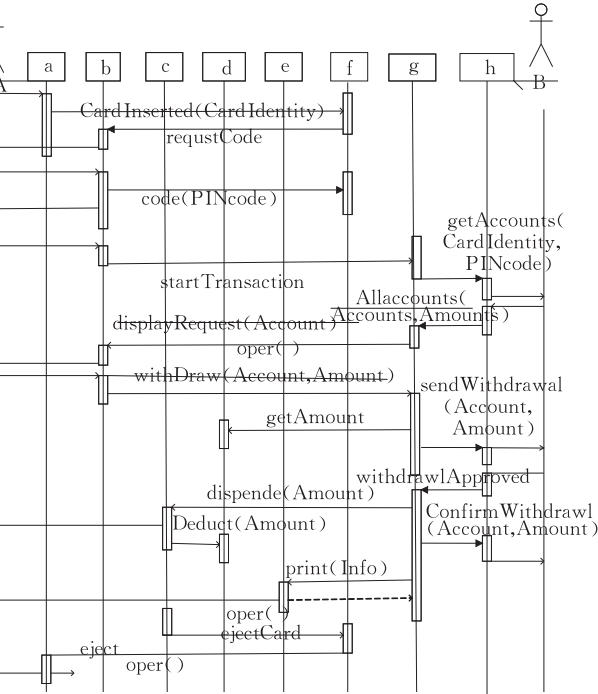


图 8 现金提取用例对象顺序图

2 为异步消息;3 为返回消息;4 为阻止消息;5 为超时消息;特别地,当 P_1 中 $s=d$ 时,设定其类型编号为 6,由此可参照 3.2 节得到神经网络的输入.最终得到预想的实验结果,如表 1~表 3 所示.

表 2 中的输出 NaN 表示解不存在.表 1 各单元数组中,第 1 层所示为网络输入向量中的 r 值,第 2 层所示为第 1 网络层和第 2 网络层的输出 r 值.

表 1 第 1、2 网络层中的输出 r 值

r								
	a	b	c	d	e	f	g	h
a	6,4,0	0,0,0	0,0,0	0,0,0	0,0,0	2,1,1	0,0,0	0,0,0
b	0,0,0	6,4,0	0,0,0	0,0,0	0,0,0	1,3,1	2,1,1	0,0,0
c	0,0,0	0,0,0	6,4,0	2,1,1	0,0,0	0,0,0	0,-1,0	0,0,0
d	0,0,0	0,0,0	0,-1,0	6,4,0	0,0,0	0,0,0	0,-1,0	0,0,0
e	0,0,0	0,0,0	0,0,0	0,0,0	6,4,0	0,0,0	3,2,2	0,0,0
f	0,-1,0	1,3,1	0,0,0	0,0,0	0,0,0	6,4,0	0,0,0	0,0,0
g	0,0,0	5,-1,0	2,1,1	2,1,1	3,2,2	0,0,0	6,4,0	1,3,1
h	0,0,0	0,0,0	0,0,0	0,0,0	0,0,0	1,3,1	6,4,0	

表 2 第 3 网络层的输出

$(\sum_{i=1}^m V_i, Z)$								
	a	b	c	d	e	f	g	h
a	(0,0)	(2,2)	(4,4)	(4,4)	(3,4)	(1,1)	(3,3)	(4,4)
b	NaN	(0,0)	(2,2)	(2,2)	(2,3)	(1,1)	(1,1)	(2,2)
c	NaN	NaN	(0,0)	(1,1)	NaN	NaN	NaN	NaN
d	NaN	NaN	NaN	(0,0)	NaN	NaN	NaN	NaN
e	NaN	NaN	(2,3)	(3,4)	(0,0)	NaN	(1,2)	(2,3)
f	(1,1)	(1,1)	(3,3)	(3,3)	(3,4)	(0,0)	(2,2)	(3,3)
g	NaN	NaN	(1,1)	(1,1)	(1,2)	NaN	(0,0)	(1,1)
h	NaN	NaN	(2,2)	(2,2)	(2,3)	NaN	(1,1)	(0,0)

表 3 第 4 网络层的输出:任意模块间重启相关度

重启相关度								
	a	b	c	d	e	f	g	h
a	4	1	1	1	1	1	1	1
b	-1	4	1	1	1	3	1	1
c	-1	-1	4	1	-1	-1	-1	-1
d	-1	-1	-1	4	-1	-1	-1	-1
e	-1	-1	1	1	4	-1	2	2
f	-1	3	1	1	1	4	1	1
g	-1	-1	1	1	2	-1	4	3
h	-1	-1	1	1	2	-1	3	4

第 5 网络层的输出:各模块可达集.

- S[a]={a,b,c,d,e,f,g,h};
- S[b]={ b,c,d,e,f,g,h};
- S[c]={c,d};
- S[d]={d};
- S[e]={c,d,e,g,h};
- S[f]={b,c,d,e,f,g,h};
- S[g]={c,d,e,g,h};
- S[h]={c,d,e,g,h}.

5 总 结

要执行细粒度软件抗衰策略,进一步提高软件抗衰的效率和准确度,更大程度地节省抗衰成本,从根本上实现软件抗衰的智能化,以此提高软件的可靠性,必须具备 4 个前提条件:(1)将软件系统划分到更细的粒度;(2)软件系统各级模块运行情况可监控;(3)给出合理的模块重启相关度及模块可达集的判定方法及原理;(4)构造神经网络模型实现抗衰重启;一般情况下,软件系统都满足第 1 和第 2 个条件。

第 3 个条件是实现第 4 个条件的前提和理论基础,本文根据软件系统中模块间的控制调用及数据访问关系,定义了模块间的耦合程度,并通过分析模块耦合度及重启相关性的关系,得出了模块重启相关度和可达集的判定算法理论。

第 4 个条件是实现智能化细粒度软件抗衰的关键。本文首先分析了不同神经网络的工作原理和应用领域,并在此基础上结合模块重启相关度及模块可达集的判定原理,首先确定了神经网络在软件抗衰中的可用性。同时结合判定算法的执行步骤及特点,构造出了基本的网络结构模型,定义了模型中的各个网络层及各层的输入输出向量,给出了各网络层的作用和 workflow,从而利用神经网络判定出模块重启相关度及模块可达集,最终为执行细粒度的软件抗衰策略奠定了坚实的基础,从根本上实现了软件抗衰的智能化,提高抗衰技术的执行效率和准确性,并更大程度地节约了抗衰成本,提高了软件可靠性。



WANG Zhan, born in 1982, Ph. D. candidate. Her current research interests focus on software rejuvenation.

Background

The Software Rejuvenation is a new technique of software fault tolerance which involves occasionally stopping the executing software, "cleaning" the "internal state" and re-

参 考 文 献

- [1] Garg S, van Moorsel A, Vaidyanathan K, et al. A methodology for detection and estimation of software aging//Proceedings of the 9th International Symposium on Software Reliability Engineering. Los Alamitos, CA, USA, 1998: 283-292
 - [2] Castelli V, Harper R E, Heidelberger P. Proactive management of software aging. IBM JRD, 2001, 45(2): 311-332
 - [3] Hong Y, Chen D, Li L et al. Closed loop design for software rejuvenation//Proceedings of the Workshop on Self-Healing, Adaptive and Self-Managed Systems. New York, USA, 2002
 - [4] Patterson D, Brown A, Broadwell P. Recovery oriented computing (ROC): Motivation, definition, techniques, and case studies. UC Berkeley: Computer Science Technical Report UCB/CSD-02-1175, 2002
 - [5] Candea G, Cutler J, Fox A. Improving availability with recursive microreboots: A soft-state system case study. Performance Evaluation Journal, 2004, 56(1-3): 213-248
 - [6] IBM servers. <http://www-03.ibm.com/servers/Systems/under Varying Workload>//Proceedings of the 10th International Pacific Rim Dependable Computing Symposium (PRDC 2004). Papeete, Tahiti, 2004: 122-129
 - [7] You Jing, Xu Jian, Zhao Xue-long, Liu Feng-Yu. Modeling and cost analysis of nested software rejuvenation policy//LNCS 3612. Springer-Verlag, 2005: 1280-1289
 - [8] You Jing, Xu Jian, Zhao Xue-long, Liu Feng-Yu. Modeling and availability analysis of nested software rejuvenation policy//Proceedings of IEEE SMC, 2005: 34-38
 - [9] Zhou Jing-Quan, Zhang Shun-Yi. The shortest path computation with neural network based on independent variables. Journal of Circuits and Systems, 2005, 10(4): 61-65(in Chinese)
- (周井泉,张顺颐. 基于独立变量的神经网络的最短路径计算. 电路与系统学报, 2005, 10(4): 61-65)

GUO Cheng-Hao, born in 1981, Ph. D. candidate. His current research interests focus on software rejuvenation.

LIU Feng-Yu, born in 1943, Ph. D., professor. Her current research interests include artificial intelligence and information security.

ZHANG Hong, born in 1956, Ph. D., professor. His current research interests include artificial intelligence and information security.

starting. This cleanup is done at desirable times during execution on a preventive basis so that unplanned failures, which result in higher costs compared to planned stopping, are

avoid.

So far, the detection of software fault as precondition of rejuvenation has been mature. The rejuvenation granularity has been fined to application layer, and the two-level-nested software rejuvenation policy has been researched. IBM has been pioneering software rejuvenation in conjunction with Duke University to improve reliability in both server and telecommunication environments. Now, the research on software rejuvenation focuses on there aspect: the thread-process-level rejuvenation policy, the intelligent rejuvenation policy, and the real -time rejuvenation policy.

This paper puts forward the model of artificial neural network which determines the degree of restart dependence between modules and reachable set of each module, the model uses the calculation arithmetic of degree of restart dependence and reachable set by analysing the connection between

restart dependence and coupling relation, so that the intelligent fine-grained thread-process layer software rejuvenation policy is supported. Then the software rejuvenation improves its veracity and efficiency, the software availability and reliability can be enhanced further.

This work is supported by the Nation Natural Science Foundation of China under grants No. 60273035. This project is conducted around the theories and principles in software performance maintenance.

This research team has focus on the research of software rejuvenation for over four years. They have published over 15 papers in highly-ranked international conferences and journals in the field of software fault- detection mechanisms, the rejuvenation policy mechanisms, the architecture of software rejuvenation mechanisms, etc.