但容易出现在最优解的位置来回反复移动, 陷入局部最优的问题. 而采用“动态变化惯性权重法”优化的 PSO 算法, 具有比固定值更为良好的搜索结果, 使得迭代初期较大的惯性权重能够在全局搜索范围内得到较好的粒子, 且搜索后期较小的惯性权重能够在极值点的周围做精细的搜索[8], 有效提高了算法的求解精度. 本文采用的动态变化惯性权重法公式如下, 对应的曲线如图 1 所示.

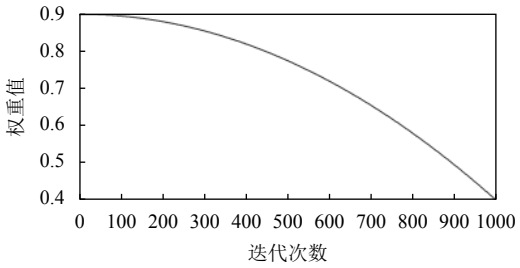

图 1 动态变化惯性权重

$$\omega(k)=\omega_{\text{start}}-(\omega_{\text{start}}-\omega_{\text{end}})\left(\frac{k}{T_{\max}}\right)^2 \tag{3}$$

其中, ω_{start}为 PSO 算法迭代初期时的惯性权重值; ω_{end}为最大迭代次数时的终止权重值; k为当前迭代数; $T_{\max}$为最大迭代数. 从图 1 可以看出, 迭代初期惯性权重变化较慢, 粒子可以进行较大范围的搜索, 随着迭代次数的增加, 惯性权重下降的速度也逐渐加快, 较好的局部搜索能力展现出来. 该策略下的粒子速度更新公式如下:

$$V_{im}^{k+1}=\omega V_{im}^{k}+c_1r_1(P_{im}^{k}-X_{im}^{k})+c_2r_2(P_{gm}^{k}-X_{im}^{k}) \tag{4}$$

2.2.2 模拟退火算法的引入

SA 算法, 主要源于热力学的理论, 通过研究热平衡问题, 将其应用在优化问题的求解过程中. 在模拟高温物体退火过程中寻找优化问题的全局或者近似全局最优解[9]. 因此如何有效的调整温度成为 SA 算法最重要的问题, 在算法中温度是一个很重要的参数, 一方面, 温度可以限制 SA 的搜索范围; 另一方面, 温度决定了算法在运行过程中以多大的概率接受劣于当前解的新解. 其中, SA 算法采用 Metepolis 准则作为接受新解的概率. 公式如下:

$$P(x=>x')=\begin{cases}1, & f(x')<f(x)\\ \exp\left[-\dfrac{f(x')-f(x)}{T}\right], & f(x')\geq f(x)\end{cases} \tag{5}$$

其中, x为当前解; x'为新解; $f(\cdot)$代表解的目标函数值; T 为温度.

本文提出的 NPSOSA 算法以 NPSO 算法为主要的流程, 是在继承 NPSO 算法优点的基础上, 引入 SA 算法. 该算法的思想是: 在更新个体和种群极值时加入 SA 算法, 对 NPSO 算法的适应度值和位置按照 Metropolis 准则接受最优解的同时以一定的概率接受恶化值. 该过程不断迭代进行, 开始时温度较高, 接受劣解的概率相对较高, 使得粒子有很高的机率跳出局部最优解, 随着温度的逐步下降, 能量降低, 粒子接受较差解的概率变小, 最终收敛至全局最优解, 避免早熟问题. p_i存储的是各个粒子的位置和适应值, P_g存储种群最优粒子的位置和适应值. 计算各粒子接受概率公式如下:

$$TF(P_i)=\frac{e^{-(f(p_i)-f(p_g))/t}}{\sum_{i=1}^{N}e^{-(f(p_i)-f(p_g)/t}} \tag{6}$$

2.3 构建 NPSOSA-BP 神经网络

NPSOSA-BP 神经网络其本质是利用 NPSOSA 算法寻优能力强, 收敛速度快的优势, 获得最优的初始权值和阈值, 并将最优值赋值给 BP 神经网络, 应用在软件老化预测趋势上. 实验表明, 利用 NPSOSA 算法优化 BP 神经网络能有效提高网络的性能, 预测精度也相对提升.

2.3.1 NPSOSA-BP 神经网络结构的确定

本文实验的输入层与输出层节点数分别为 3、10. 隐含层节点数按照传统的计算公式$l=\sqrt{m+n}+a$, $a\in(1,10)$, 得出适宜的节点数值范围为 3~12. 经过仿真实验, 验证得出当隐含层节点数为 10 的时候, 模型能达到最佳软件老化预测状态. 于是本文搭建了具有 3-10-1 结构的 NPSOSA-BP 神经网络预测模型. 实际上, 除了选取最佳隐含层神经元个数以外, 也可以通过增加隐含层数来提高神经网络的学习能力, 但后者方法使网络结构复杂化, 训练时间增长, 因此本文采用单隐含层 BP 神经网络.

2.3.2 NPSOSA-BP 神经网络预测模型的训练与预测

NPSOSA-BP 神经网络预测模型的训练与预测流程如下:

① 数据预处理. 由于原始数据数量级差别较大, 对 BP 神经网络训练过程产生较大的影响, 加大训练难度. 因此为了提高神经网络的训练效率, 在对数据进行分析之前, 需要对数据进行归一化处理, 将输入层数据和输出层数据归一化到[–1, 1]之间.

② NPSOSA-BP 算法参数的设定. 本实验 NPSO 算法参数设定: 加速度因子$c_1 = c_2 = 1.494\,45$, $\omega_{start} = 0.9$, $\omega_{end} = 0.4$, 迭代次数为 1000 次, 种群规模为 60. SA 算法参数设置: 初始温度为 10 000, 温度下降速率采用公式: $T = T_0(0.9^{j-1})$, 其中T_0为初始温度, j为迭代次数. BP 算法初始参数设置: 学习速率为 0.05, 期望误差为 1×10^{-6}.

③ 初始化粒子群. NPSO 算法需要对 BP 神经网络中所有权值和阈值的集合进行优化. 粒子的速度和位置向量的维数等于 BP 神经网络中所有权值和阈值的数量之和, 即 3×10+10×1+10+1=51. 采用算法开发与分析工具 Matlab 2016a, 其内部函数 rands(1, 51) 随机初始化粒子的位置和速度.

④ 更新粒子的速度和位置. 将所有粒子放置在预测模型中进行训练, 通过适应度函数式 (7) 计算适应度值, 以此作为评价粒子的优劣, 适应值越小, 表示适应度越高. 此外根据式 (2) 和式 (4), 更新粒子的速度和位置, 本文采用 BP 神经网络的输出均方根平均误差和作为神经网络的适应度值, 公式如下:

$$F = \sum_{i=1}^{m}(y_i - \hat{y})^2 \tag{7}$$

⑤ 迭代输出最优值. 粒子在每次更新时, 采用 Metropolis 准则计算适应度值, 随着温度的降低, 收敛得到全局最优解.

⑥ NPSOSA-BP 神经网络模型训练. 获得最优解并确定 BP 神经网络的权值和阈值, 输入训练数据来训练模型. 用训练好的模型对软件老化趋势进行预测.

3 实验与结果分析

3.1 软件老化加速寿命实验

为研究与内存泄漏相关的软件老化现象, 本文以一个符合 TPC-W 基准测试规范的 Web 应用服务器为研究对象, 在其后台程序中注入关于内存泄漏的代码加速其老化, 从而设计出加速寿命测试实验[10], 并收集老化数据作为实验的训练集和测试集. TPC-W 的组成在逻辑上分为 3 层: 一个 Web 服务器, 一个数据库, 以及一定量的模拟浏览器. 三者之间的流程为: 数据库存储售书网站的数据; 模拟浏览器以会话的形式并发访问网站页面; 当请求发送给 Web 服务器查询相应网页和图像时, Web 服务器紧接着与数据库通信, 传输数据并动态生成响应页. 实验配置如表 1 所示, 三者间的工作原理如图 2 所示.

表 1 实验配置

	服务器	客户端	数据库
操作系统	Linux CentOS7 Kernel 3.10.0-693.17.1. el7	Linux CentOS7 Kernel 3.10.0-693.17.1. el7	Linux CentOS7 Kernel 3.10.0-693.17.1. el7
CPU	Intel® Xeon® CPU E5506@2.13 GHz	Intel® Core™2 Duo CPU E8400@3.00 GHz 3.00 GHz	Intel® Xeon® CPU E5506@2.13 GHz
JVM	JDK 1.8. 0		
软件	Apache Tomcat 8.5	TPC-W 客户端	MySQL 5.7

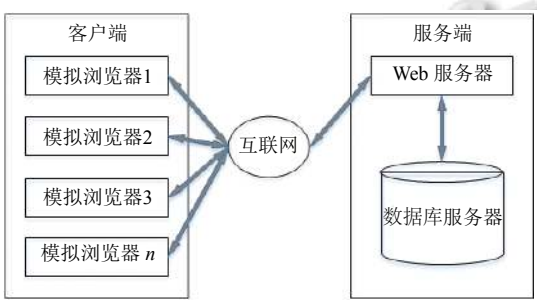

图 2 三者间工作原理

本文选取了内存消耗速率作为加速寿命测试中的加速压力, 对 Java 虚拟机的内存管理进行研究. 由于所有的 Servlet 运行在同一个 JVM 中, 因此所有对象的数据都存储在 JVM 堆中[11]. 为了模拟软件老化现象, 实验修改了购书网站中的 Search Request Page 这个页面, 对应 TPCW_search_request_servlet 这个 Servlet, 为其注入一段内存泄漏代码. 由于 JVM 有垃圾回收器 (GC), 程序在运行时, 对象的堆内存由 GC 的自动内存管理系统回收, 当 JVM 内存使用量逐渐增加并达到最大值时, GC 将自动启动并释放内存. GC 回收任何不再被引用的对象, 其占用的内存也会被释放. 为了模拟软件老化现象, 本文增加一个 HeapLeak 类, 使服务器整个生命周期都保持对该类对象的引用, 且 HeapLeak 类对象在程序运行期间不会被 GC 回收, 当创建的实例数量达到最大堆容量时, 会产生 JVM 堆溢出. 运行客户端, 每隔 1 s 采集一次 JVM 内存使用量, 共持续 14 400 s, 采集样本 14 281 个. 每隔 120 s 取一次均值, 实验数据

图如图 3 所示.

3.2 实验分析

为了证明模型的有效性, 使用 Matlab 2016a 编程实现了 NPSOSA 算法优化的 BP 神经网络模型, 以及与其对比分析的传统 PSOSA-BP、PSO-BP, BP 预测模型. 神经网络结构采用 3-10-1, 每组实验的工作负载为 100 个客户端. 实验共收集到 14 281 个数据, 取前 7000 个作为训练集, 后 7281 个作为预测集. 不同的预测模型在实验训练过程中采用的是相同的老化数据, 实验结束后分别得到 4 个模型的预测与期望输出的对比曲线, 如图 4 至图 7 所示. 本文选取 MAE (平均绝对误差), MSE (均方误差) 作为评价标准, 公式如下. 4 种不同预测模型的评价指标对比如表 2 所示.

$$MAE = \frac{1}{m}\sum_{i=1}^{m}\left|(y_i - \hat{y})\right| \tag{8}$$

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y})^2 \tag{9}$$

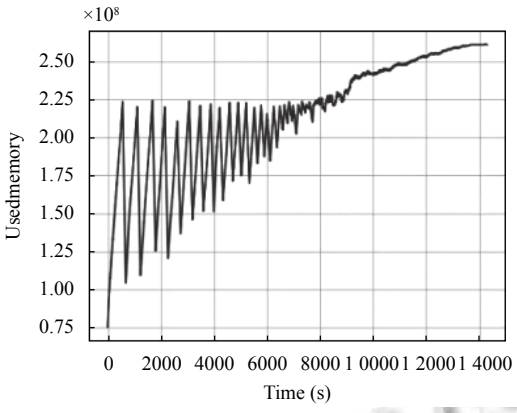

图 3 JVM 在 120 s 内平均使用内存变化趋势

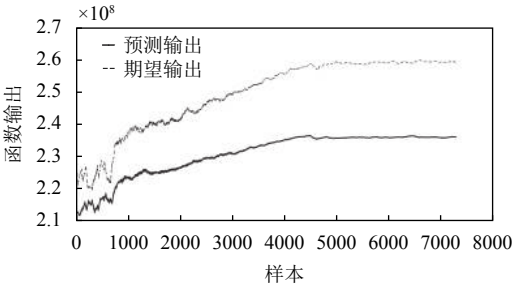

图 4 BP 神经网络预测

从图 4 到图 7 可以看出本文提出的 NPSOSA-BP 神经网络预测模型, 经引入动态变化惯性权重法, SA 算法的优化后, 预测输出曲线与期望输出曲线的拟合效果越来越好, 预测精度也明显提高. 从表 2 可以得出 NPSOSABP 神经网络模型相比 PSOSA-BP 神经网络模型预测精度提高了 8%, 相比 PSOBP 神经网络预测模型提高了 10%, 相比 BP 神经网络模型提高了 30%. 由此可以得出, 本文提出的基于新型粒子群退火算法优化的 BP 神经网络模型相比于其他优化算法在预测精度、拟合效果方面存在明显的优势.

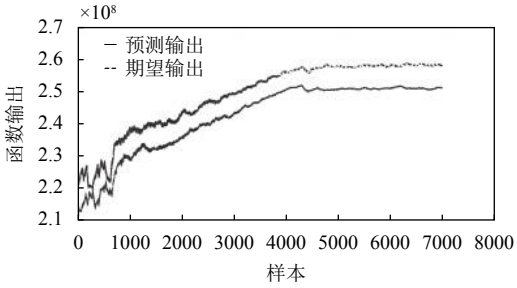

图 5 PSO-BP 神经网络预测

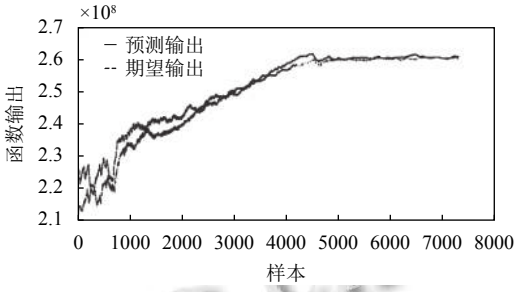

图 6 PSOSA-BP 神经网络预测

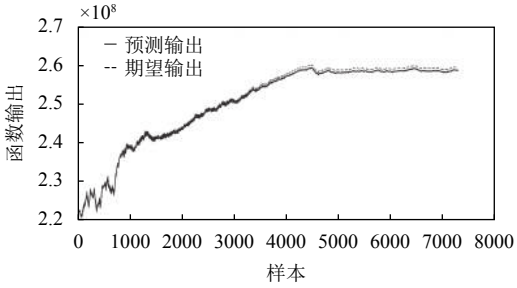

图 7 NPSOSA-BP 神经网络预测

表 2 神经网络模型误差对比图

类别	NPSOSA-BP	PSOSA-BP	PSO-BP	BP
MAE	8.0135e–05	2.0391e–06	7.1104e–06	1.4582e–07
MSE	7.2251e–11	9.2574e–12	5.0317e–13	2.5438e–14

4 结束语

软件老化是影响软件系统性能的重要因素, 本文

通过建立一个软件老化测试平台, 为其注入一段内存泄漏代码加速老化, 收集系统老化数据, 根据获取的老化数据构造 NPSOSA-BP 神经网络预测模型. NPSOSA 算法不仅解决了 BP 神经网络对初始权值和阈值依赖性的问题, 而且克服了 PSO 算法全局与局部搜索不平衡、早熟收敛相关问题, 提高了神经网络的预测精度. 使得 BP 神经网络拥有良好的学习能力和泛化能力, 能够很好的应用在软件老化趋势预测方面.

参考文献

1 Yan YQ. Variance analysis of software ageing problems. IET Software, 2018, 12(1): 41–48. [doi: 10.1049/iet-sen.2016.0290]

2 Zhao J, Trivedi KS, Grottke M, *et al*. Ensuring the performance of apache HTTP server affected by aging. IEEE Transactions on Dependable and Secure Computing, 2014, 11(2): 130–141. [doi: 10.1109/TDSC.2013.38]

3 Bovenzi A, Cotroneo D, Pietrantuono R, *et al*. On the Aging Effects due to concurrency bugs: A case study on MySQL. 2012 IEEE 23rd International Symposium on Software Reliability Engineering. Dallas, TX, USA. 2012. 211–220.

4 Machida F, Kim DS, Trivedi KS. Modeling and analysis of software rejuvenation in a server virtualized system with live VM migration. Performance Evaluation, 2013, 70(3): 212–230. [doi: 10.1016/j.peva.2012.09.003]

5 Huang DM, He SQ, He XH, *et al*. Prediction of wind loads on high-rise building using a BP neural network combined with POD. Journal of Wind Engineering and Industrial Aerodynamics, 2017, 170: 1–17. [doi: 10.1016/j.jweia.2017.07.021]

6 Chen X, Wang T, Liang W. General aircraft material demand forecast based on PSO-BP neural network. International Journal of Control and Automation, 2016, 9(5): 407–418. [doi: 10.14257/ijca.2016.9.5.39]

7 Lu HY, Sriyanyong P, Song YH, *et al*. Experimental study of a new hybrid PSO with mutation for economic dispatch with non-smooth cost function. International Journal of Electrical Power & Energy Systems, 2010, 32(9): 921–935.

8 Zhang YD, Wang SH, Phillips P, *et al*. Binary PSO with mutation operator for feature selection using decision tree applied to spam detection. Knowledge-Based Systems, 2014, 64: 22–31. [doi: 10.1016/j.knosys.2014.03.015]

9 蒋美云. 遗传模拟退火算法在云调度中的应用研究. 计算机与现代化, 2013, (6): 38–41. [doi: 10.3969/j.issn.1006-2475.2013.06.011]

10 Yin YC, Coolen FPA, Coolen-Maturi T. An imprecise statistical method for accelerated life testing using the power-Weibull model. Reliability Engineering and System Safety, 2017, 167: 158–167. [doi: 10.1016/j.ress.2017.05.045]

11 刘飞. Java 虚拟机内存管理与内存泄漏. 信息与电脑, 2017, (6): 69–71. [doi: 10.3969/j.issn.1003-9767.2017.06.025]