结合变邻域搜索的竞争 Hopfield 神经网络解决最大分散度问题

周雅兰[1] 王甲海[2] 闭 玮[2] 莫 斌[2] 李曙光[2]
(广东商学院信息学院 广州 510320)[1] (中山大学信息科学与技术学院 广州 510006)[2]

摘 要 提出一种结合变邻域搜索的离散竞争 Hopfield 神经网络,用于求解最大分散度问题。为了克服神经网络易陷入局部最小值的问题,将变邻域搜索的思想引入到离散竞争 Hopfield 神经网络中,一旦网络陷入局部最小值,变邻域搜索能帮助神经网络动态改变搜索邻域,从而跳出局部最小值去搜寻更优的解。最后,针对最大分散度问题的实验结果表明,提出的算法具有良好的性能。

关键词 变邻域搜索,Hopfield 神经网络,最大分散度问题
中图法分类号 TP18 **文献标识码** A

Competitive Hopfield Network Combined with Variable Neighborhood Search for Maximum Diversity Problems

ZHOU Ya-lan[1] WANG Jia-hai[2] BI Wei[2] MO Bin[2] LI Shu-guang[2]
(School of Information, Guangdong University of Business Studies, Guangzhou 510320, China)[1]
(School of Information Science and Technology, Sun Yat-sen University, Guangzhou 510006, China)[2]

Abstract A discrete competitive Hopfield neural network (DCHNN) combined with variable neighborhood search (VNS) was proposed for the maximum diversity problem. In order to overcome the local minima problem of DCHNN, the idea of VNS was introduced into DCHNN. Once the network is trapped in local minima, the VNS can generate a new search neighborhood for DCHNN. Thus, the proposed algorithm can escape from local minima and further search better results. Finally, simulation results on the maximum diversity problem show that the proposed algorithm has good performance.

Keywords Variable neighborhood search, Hopfield network, Maximum diversity problem

1 引言

最大分散度问题(maximum diversity problem, MDP)是一个 NP-难问题[1],该问题描述如下:假设一个集合有 n 个元素,两元素 i 和 j 之间的距离表示为 d_{ij} ($d_{ij}=d_{ji}$),如果 $j\neq i$ 则 $d_{ij}>0$;否则 $d_{ij}=0$。要求在该集合中选出含有 m 个元素的一个子集,使得该子集中元素之间的距离之和最大。如果元素 i 属于该子集,则 $v_i=1$,否则 $v_i=0$,求解 MDP 就是使如下的函数值最大:

$$f(V)=\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}d_{ij}v_iv_j \tag{1}$$

且满足约束条件:

$$\sum_{i=1}^{n}v_i=m \tag{2}$$

提出了很多元启发式算法,用于求解 MDP。Macambira[2]根据文献[3]的通用框架提出一种禁忌搜索(tabu search, TS),称为 MTS。Silva 等人[4]组合不同的初始解构造方法和后期解改进方法形成了一系列的贪婪随机自适应搜索方法(greedy randomized adaptive search procedure, GRASP),并把其中的一个 GRASP 算法与路径重链(path-relinking, PR)技术相结合形成称为 KLD + PR 的混合算法[5]。Aringhieri 等人[6]提出一种基于 GRASP 的 TS 算法,称为 GRASP-TS。Duarte 和 Martí[7]引入 TS 的记忆元件,提出了基于 TS 的算法,特别地,他们还提出了两种新的构造算法(一种基于 GRASP 构造,另外一种基于记忆元件构造),并把它们与 Ghosh[8]提出的简单局部搜索(local search, LS)、改进的 LS 和具有短暂记忆的 TS 这 3 种迭代改进方法相结合。其中与具有短暂记忆的 TS 相结合形成的称为 Tabu_D2+LS_TS 的混合算法求解 MDP 能获得最好的性能。最近,Palubeckis[9]提出一种求解 MDP 的迭代禁忌搜索(iterated TS, ITS)算法,实验显示 ITS 算法取得了迄今为止最好的结果。

Hopfield 神经网络[10]自从提出以来,在组合优化问题的应用中取得了很大的成功[11-13]。本文提出一种离散竞争 Hopfield 神经网络(discrete competitive Hopfield neural network, DCHNN),用于求解 MDP。DCHNN 采用 N 中取 k(k-out-of-N)的竞争规则总是能够产生合理解,大大减少了搜索空间,而且没有参数需要调整。但是 Hopfield 神经网络常常

到稿日期:2009-09-28 返修日期:2009-12-18 本文受国家自然科学基金(60805026, 60905038),高等学校博士学科点专项科研基金(20070558052),教育部留学回国人员科研启动基金(教外司留[2007]1108号),广东商学院校级科研项目(08BS52001)资助。
周雅兰(1979—),女,博士,讲师,CCF 会员,主要研究方向为人工智能与数据挖掘,E-mail:zhouyalan@gdcc.edu.cn;王甲海(1977—),博士,副教授,主要研究方向为人工智能及其应用;闭 玮 男,主要研究方向为人工智能;莫 斌 男,主要研究方向为人工智能;李曙光 男,主要研究方向为人工智能。

陷入局部极小值，所以网络的稳定状态并不一定对应于问题的最优解[11-13]。因此，怎么脱离局部极小值，使得网络能够继续寻优，是 Hopfield 神经网络一个由来已久的关键问题。为了克服 DCHNN 的局部最小值问题，将变邻域搜索(variable neighborhood search，VNS)算法的思想引入到 DCHNN 中，提出了结合变邻域搜索的竞争 Hopfield 神经网络，用于解决 MDP，该网络简称为 DCHNN-VNS。

2 离散竞争 Hopfield 神经网络求解 MDP

给定一个 $n\times n$ 对称矩阵 $D=(d_{ij})$ 的 MDP 能被映射成带 n 个神经元的 HNN，设神经元 i 的输入为 u_i，输出为 v_i，如果元素 i 被选中作为子集中的元素，那么 $v_i=1$，否则 $v_i=0$。MDP 的能量函数表示如下：

$$E=A(\sum_{i=1}^{n}v_i-m)^2-\frac{B}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}d_{ij}v_iv_j \tag{3}$$

其中，A 和 B 是权重因子。

MDP 解的质量受权重因子 A 和 B 的值影响较大，为了减少参数对解的影响，提出离散竞争 Hopfield 神经网络(DCHNN)来处理式(3)中等式右边的第一项约束项。DCHNN 采用 k-out-of-N 的竞争规则更新神经元，使得网络在 N 个神经元中恰好有 k 个被激活，从而使网络达到一个平衡稳定状态，对应 MDP 的一个合理解。第 i 个神经元的输入输出函数表示为：

$$v_i(t+1)=\begin{cases}1 & \text{if } u_i(t)\geqslant u_{m\text{-}th}(t)\\ 0 & \text{otherwise}\end{cases} \tag{4}$$

其中，$u_{m\text{-}th}(t)$ 是 $\{u_1(t),\cdots,u_n(t)\}$ 中第 m 个最大的值。n 个神经元的输入 u_i 以递减的顺序排列，从这 n 个神经元中选出排在前面的 m 个神经元作为激活神经元，输出为 1，其余神经元的输出为 0。在迭代中，如果有多于 m 个神经元满足 $u_i(t)\geqslant u_{m\text{-}th}(t)$，那么优先选择前一次迭代输出为 0 的神经元作为激活神经元，这种激发机制可以让更多的神经元获得被激活的机会，使得神经元的输出状态能够随机变化，从而防止出现总是连续激活一个神经元而使网络停滞于一个固定状态的问题。式(4)表示的竞争神经元模型相当于一个动态阈值等于 $u_{m\text{-}th}(t)$ 的 MacCulloch-Pitts 神经元模型。

通过调整式(4)的竞争模型，式(2)描述的问题约束条件总是能被满足，那么 MDP 的能量函数表示如下：

$$E=-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}d_{ij}v_iv_j \tag{5}$$

根据离散 HNN 的更新规则[12]，求解 MDP 的神经元的输入计算如下：

$$\frac{du_i}{dt}=-\frac{\partial E}{\partial v_i}=\sum_{j=1}^{n}d_{ij}v_j \tag{6}$$

实际上，上式用一阶欧拉方程计算如下：

$$\Delta u_i=\sum_{j=1}^{n}d_{ij}v_j \tag{7}$$

于是，DCHNN 中的每个神经元的输入 u_i 迭代更新如下：

$$u_i(t+1)=u_i(t)+\Delta u_i \tag{8}$$

迭代更新一次表示 n 个神经元全部更新一次。

DCHNN 算法求解 MDP 的过程描述如下：

第一步　初始化 $u\in[-1,1]$；

第二步　根据式(4)计算 v，并初始化 t 为 0；

第三步　根据动态方程(8)计算 $u_i(t+1)$；

第四步　根据式(4)计算 $v_i(t+1)$；

第五步　如果 $t=t+1$ 小于 t_{max} 或者还未达到稳定终止条件，则转至第三步；否则算法终止。

Hopfield 神经网络由于采用梯度下降法，很容易使网络陷入局部极小值。在下一小节，将提出结合 VNS 的 DCHNN 来求解 MDP，以克服 DCHNN 易陷入局部极小值的问题。

3 结合 VNS 的 DCHNN 求解 MDP

DCHNN 在优化早期效果明显(也就是说，在迭代初期能量函数下降迅速)，经过几次迭代后优化效果慢慢变得不太明显，例如迭代 10 个循环之后，将陷入局部最小值，此时能量函数不再下降[13]。可以考虑当网络陷入局部最小值时，将 VNS 作为一个扰动操作引入到 DCHNN，通过 VNS 扩展新的搜索邻域为 DCHNN 的下一轮搜索产生新的起始点。下面首先简单介绍 VNS，然后提出结合 VNS 的 DCHNN 算法。

3.1 VNS

VNS 是 Hansen 等人提出的一种元启发式算法[14]，与传统方法采用单一邻域结构不同，VNS 定义了一系列邻域结构，在搜索过程中根据某种机制不断改变邻域结构来搜索最优解。VNS 包括抖动、局部搜索和替代三步，其基本思想是：1)不同的邻域结构，可能有不同的局部最优；2)所有可能邻域结构下的局部最优为全局最优。变邻域搜索算法简单有效，因而在组合优化领域应用广泛[15]。

VNS 的基本框架描述如下：

第一步　随机产生初始解 x_cur，初始化全局最优解 x_opt 为 x_cur；

第二步　定义 x_opt 的一系列邻域结构 $N_k(x\_opt)$，初始化 k 的值为 1；

第三步　抖动，从 x_opt 的邻域 $N_k(x\_opt)$ 中随机产生临时解 x_temp；

第四步　局部搜索，以 x_temp 为初始解，利用局部搜索方法得到新的当前解 x_cur；

第五步　替代，如果 x_cur 优于 x_opt，则用 x_cur 更新 x_opt 并重设 k 的值为 1，转至第三步；否则下一步；

第六步　如果 $k=k+1$ 小于 k_{max}，则转至第三步；否则算法终止，输出 x_opt 为问题的解。

3.2 DCHNN-VNS

我们将 VNS 与 DCHNN 相结合，由 DCHNN 执行局部搜索，VNS 执行扰动操作，以帮助 DCHNN 脱离出局部最优值。DCHNN-VNS 算法求解 MDP 的过程描述如下：

第一步　初始化 u，$u\in[-1,1]$；

第二步　根据式(4)获得当前解 v_cur，初始化全局最优解 v_opt；

第三步　定义 v_cur 的一系列邻域结构 $N_k(v\_cur)$，初始化 k 值为 k_{min}；

第四步　运行时间小于 T_{max}，则转至下一步；否则转至第十步；

第五步　抖动，从当前解 v_cur 的邻域 $N_k(v\_cur)$ 中随机产生临时解 v_temp；

第六步　局部搜索，使用 DCHNN 求解 MDP(其中 $t_{max}=10$)，产生新的当前解 v_cur；

第七步　使用式(9)重新初始化 u;

第八步　替代,如果 v_cur 优于 v_opt,则用 v_cur 更新 v_opt,重置 k 值为 k_{min},转至第四步;否则下一步;

第九步　当 $k=k+1$ 小于 k_{max},则转至第四步;否则下一步;

第十步　算法终止,输出 v_opt 为问题的解。

上述算法中 v_cur 的一系列邻域结构 $N_k(v\_cur)$ 定义如下:MDP 的解对应到 DCHNN 就是 $v=\{v_1,v_2,\cdots,v_n\}$,其中有 m 个激活神经元,从这 m 个激活神经元中随机选出一个神经元 p,从 $n-m$ 个非激活神经元中随机选出一个神经元 q,两个神经元 p,q 进行值的互换操作,作为一次 swap(p,q) 操作,得到邻域 $N(v)$。连续做 k 次 swap(p,q) 操作,得到的邻域记为 $N_k(v)$,$k\leqslant\min(m,n-m)$。

DCHNN-VNS 算法过程中的局部搜索采用前面第 2 小节所描述的 DCHNN 算法。DCHNN-VNS 算法过程中的抖动不同于原始 VNS 算法每次采用最优解进行抖动,而是采用 DCHNN 进行局部搜索得到新的当前解进行抖动,这有利于保持解的多样性。为了防止神经元的输入 u_i 在演化过程中随着式(8)的更新变得过大或者过小,在下一轮 DCHNN 局部搜索之前,将每一个神经元的输入 u_i 重新设置为 0 或者 1,操作如下[12]:

$$u_i=\begin{cases}1 & \text{if } u_i>0\\ 0 & \text{otherwise}\end{cases} \tag{9}$$

在 DCHNN-VNS 算法中,DCHNN 的局部搜索和 VNS 的扰动操作循环反复进行,参数 k 的值达到 k_{max} 或者算法的运行时间大于 T_{max} 为算法的终止条件。

4　仿真实验

为了检验所提算法的性能,本文用 C 语言在 PC (Pentium 4,2.8 GHz)上进行了仿真实验。本小节首先描述用于实验仿真的 benchmark 数据,然后显示 DCHNN-VNS 在 benchmark 数据上的实验结果,同时与其他求解该问题的元启发式算法进行比较。实验中 DCHNN-VNS 的参数 k 设置为:$0.2m\leqslant k\leqslant\min(m,n-m)$(即 $k_{min}=0.2m$,$k_{max}=\min(m,n-m)$)。

4.1　benchmark 数据

以 2 个数据集共 18 个实例检验了所提算法的性能。实例的大小从 400(小规模)到 2000(大规模),表 1、表 2 显示了这 18 个实例的名称以及迄今所知的最优解。描述这 2 个数据集如下:

1)Silva 实例:从均匀分布的[0,9]区间中产生随机整数组成 $n\times n$ 的矩阵,$n\in[100,500]$,$m\in[0.1n,0.2n,0.3n,0.4n]$。Silva 等人[4]首次引入这些实例,它们可从在 http://www.ic.uff.br/~gsilva/instSilva.zip 下载。由于小规模的数据比较容易求解,因此本文取其中规模最大的两组,即 n 取 400,500 共 8 个实例来测试本文算法。DCHNN-VNS 在 $n=400$ 的实例上设置 $T_{max}=13$ 秒,在 $n=500$ 的实例上 $T_{max}=18$ 秒。

2) 随机类型实例:文献[7]首次引入这些实例,它们可从 http://www.uv.es/~rmarti/paper/mdp.html 下载。同样,由于小规模的数据比较容易求解,我们取其中最大规模的 10 个实例 Type1_22.1－Type1_22.10($n=2000$,$m=200$)来测试本文算法,这些实例是从均匀分布的(0,10)区间中产生的实数矩阵。DCHNN-VNS 在这些实例上设置 $T_{max}=360$ 秒。

表 1　DCHNN-VNS 与其他元启发式算法求解 Silva 实例的实验结果比较

实例名称	最优解	DCHNN-VNS			ITS		MTS		Tabu_D2+LS_TS	KDL+PR	GRASP-TS
		Best	Av.	Std.	Best	Av.	Best	Av.	Best	Best	Best
Silva_400_40	4658	0	3	3.38	0	0	27	52.6	3	0	0
Silva_400_80	16956	0	10	6.57	0	0	24	78.7	8	11	0
Silva_400_120	36317	0	14	6.27	0	0	0	101.6	19	11	0
Silva_400_160	62487	0	19	18.66	0	7.0	10	88.1	31	0	0
Silva_500_50	7141	0	14	7.63	0	0	24	74.3	8	11	0
Silva_500_100	26258	0	7	11.43	0	0	15	96.8	4	4	0
Silva_500_150	56572	0	1	1.6	0	0	0	110.0	0	0	0
Silva_500_200	97344	0	23	10.78	0	0	18	108.7	0	0	0

表 2　DCHNN-VNS 与其他元启发式算法求解 Type1_22 实例的实验结果比较($n=2000$,$m=200$)

实例名称	最优解	DCHNN-VNS			ITS		MTS		Tabu_D2+LS_TS
		Best	Av.	Std.	Best	Av.	Best	Av.	Best
Type1_22.1	114271	49	261.6	101.94	110	280.1	634	943.2	431
Type1_22.2	114327	84	288.67	85.95	75	293.2	652	1117.1	491
Type1_22.3	114195	181	327.37	70.04	69	242.1	709	957.1	664
Type1_22.4	114093	107	243.57	78.98	93	200.9	685	936.7	668
Type1_22.5	114196	232	345.67	68.23	171	362.7	652	822.9	780
Type1_22.6	114265	59	217.77	77.88	105	249.8	586	869.9	304
Type1_22.7	114361	50	205.33	95.04	33	208.3	721	988.9	660
Type1_22.8	114327	19	364.7	116.39	25	271.7	711	1039.4	599
Type1_22.9	114199	117	258.83	79.01	9	200.4	551	878.4	501
Type1_22.10	114229	41	260.83	91.56	166	266.9	618	829.7	658

4.2　与其他元启发式算法比较

为了显示 DCHNN-VNS 算法的良好性能,将其与其他的元启发式算法,包括 ITS[9],MTS[2],Tabu_D2+LS_TS[7],GRASP-TS[6] 和 KLD+PR[5] 进行了比较。这些算法在求解 MDP 时都有良好表现,特别是 ITS 在 69 个测试问题上求得了最新的最优解,是迄今为止求解 MDP 的最好算法,因此,ITS 作为基准算法用于评估 DCHNN-VNS 算法的性能。

与文献[9]类似,给出的实验结果是相对解,"Best"是算法运行 30 次得到的最好解与迄今所知的最优解之间的偏差,"Av."是算法运行 30 次得到的平均解与迄今所知的最优解之间的偏差。表中用于比较算法的实验结果直接来自相关文献。

表1显示了6种算法求解Silva实例运行30次后得到的最好解(Best),同时列出了所提算法DCHNN-VNS的平均解(Av.)和标准偏差(Std.),以及ITS和MTS的平均解(Av.)。从表中可以看出提出的算法DCHNN-VNS,ITS以及GRASP-TS在所有的8个实例上都能求得最优解,只是DCHNN-VNS求得的平均解比ITS稍差。

表2显示了DCHNN-VNS,ITS和MTS在Type1_22实例上的最好解和平均解,以及Tabu_D2+LS_TS的最好解,从表中可见DCHNN-VNS和ITS能获得比其他两种算法更好的解,其中在Type1_22.1,Type1_22.6和Type1_22.10上DCHNN-VNS能获得比ITS更好的最好解和平均解,在Type1_22.8上DCHNN-VNS能获得比ITS更好的最好解,但是平均解稍差,在Type1_22.2,Type1_22.5和Type1_22.7上DCHNN-VNS能获得比ITS更好的平均解,但是最好解稍差。

由于不同算法的运行时间是在不同计算机上测试而定的,因此时间难以直接比较。本文与文献[16,17]一样,采用Dongarra[18]的Mflop/s方法求出用于评估相对运行时间的Dongarra's因子,然后将运行时间做一个标准化的转换。表3列出了不同性能计算机采用Dongarra文献中的方法求出的可用于标准化运行时间的Dongarra's因子。采用Dongarra's因子进行标准化转换得出的运行时间虽然也不一定精准[16],但可以提供粗略的比较。

表3 计算机性能评估

计算机类型	Performance estimated from Dongarra's paper (Mflop/s)	Estimated Dongarra's factor
AMD Athlon 1.4 GHz	704	0.5135
Pentium 4 3GHz	1571	1.1459
Pentium 4 Mobile 2.8GHz	-	1
Pentium M 1733 MHz notebook	-	1
Pentium 4 2.8GHz	1371	1

表4列出了6种算法求解Silva实例在不同计算机上的实际运行时间和经过转换的标准化运行时间"NT"。可以看出提出的算法DCHNN-VNS与Tabu_D2+LS_TS的运行时间最短,只需要十几秒,而KLD+PR与GRASP-TS所花费的时间最长,高达几小时甚至十几个小时,从它们在小规模的数据上都要花费如此大量的时间来看,这两种算法没有求解大规模数据的能力,因此在表5中求解大规模数据时没有再使用这两种算法进行比较。

表4 各算法求解Silva实例的运行时间比较

	KLD+PR		GRASP-TS	DCHNN-VNS	ITS MTS	Tabu_D2+LS_TS	
运行环境	C,PC 1.4 GHz AMD Athlon with 250 MB RAM.	NT	C,PC Intel Pentium 4 Mobile 2.8GHz 512MB RAM	C,PC Pentium 4 2.8GHz 512MB	C,Pentium M 1733MHz notebook	Borland Builder 5.0,PC Pentium 4 3GHz with 1GB RAM	NT
Silva_400_40	1300.6	667.85	45.20	13	20	10	11.5
Silva_400_80	5957.1	3058.97	128.28				
Silva_400_120	13850.7	7112.33	208.22				
Silva_400_160	20456.7	10504.52	295.32				
Silva_500_50	3072.5	1577.73	94.33	18			
Silva_500_100	15857.2	8142.67	272.45				
Silva_500_150	34661.4	17798.63	446.43				
Silva_500_200	52497.4	26957.41	627.11				

表5是4种算法求解大规模实例Type1_22在不同计算机上的实际运行时间和经过转换的标准化运行时间"NT",其中DCHNN-VNS运行的时间稍长,但DCHNN-VNS是一种神经网络方法,可以用硬件FPGA来加快速度[19]。

表5 4种算法求解Type1_22实例的运行时间比较

	DCHNN-VNS	ITS,MTS	Tabu_D2+LS_TS	
运行环境	C,PC Pentium 4 2.8GHz 512MB	C,Pentium M 1733MHz notebook	Borland Builder 5.0, PC Pentium 4 3GHz with 1GB RAM	NT
Type1_22 (n=2000, m=200)	360	20	10	11.5

综上分析,从总体来看DCHNN-VNS与迄今求解MDP的最好算法ITS的性能相当。

结束语 本文提出了一种结合变邻域搜索的离散竞争Hopfield神经网络算法,用于求解MDP。提出的算法为了克服DCHNN容易陷入局部最小值的缺点,在每次DCHNN陷入局部最小值时,由VNS引导DCHNN到另一个邻域进行搜索,使得DCHNN能够脱出局部最小值。为了显示所提算法的性能,与其他的进化算法进行了比较,如:ITS,MTS,Tabu_D2+LS_TS,GRASP-TS和KLD+PR,显示了所提算法的良好性能。

参考文献

[1] Kuo C C, Glover F, Dhir K S. Analyzing and modeling the maximum diversity problem by zero-one programming [J]. Decis. Sci., 1993, 24: 1171-1185

[2] Macambira E M. An application of tabu search heuristic for the maximum edge-weighted subgraph problem [J]. Annals of Operations Research, 2002, 117: 175-190

[3] Glover F. Tabu search-part I [J]. ORSA Journal on Computing, 1989, 1: 190-206

[4] Silva G C, Ochi L S, Martins S L. Experimental comparison of greedy randomized adaptive search procedures for the maximum diversity problem [J]. Lecture Notes in Computer Science, 2004, 3059: 498-512

[5] Silva G C, Andrade M, Ochi L S, et al. New heuristics for the maximum diversity problem [J]. Journal of Heuristics, 2007, 13: 315-336

[6] Aringhieri R, Cordone R, Melzani Y. Tabu search versus GRASP for the maximum diversity problem [J]. 4OR: A Quarterly Journal of Operations Research, 2008, 6(1): 45-60

(下转第252页)

统计了各个算法生成的软件扇出树的一些静态信息。表1显示了软件扇出树的平均高度，软件扇出树的高度用从根节点到任意一个叶子节点的路径上所插入的最大的MOV指令条数来表示。为了最小化关键路径长度，Hartley和Casavant算法产生了最不平衡的软件扇出树。从表中可以看出，各个算法所生成的软件扇出树的平均高度比较接近，基于剖析信息和关键路径长度的软件扇出树生成算法生成的软件扇出树的平均高度略小于其它几种算法生成的软件扇出树的平均高度。各个算法生成的软件扇出树的最大高度也比较接近。但是各个算法所生成的软件扇出树的内部结构不同，因而导致了性能的差异。在基于剖析信息和关键路径长度的软件扇出树生成算法生成的软件扇出树中，执行概率大的和关键路径长的目标指令距离根节点较近，可以优先获得指令的计算结果，而在其它算法所生成的软件扇出树中却没有区分执行概率大的和关键路径长的目标指令，因而导致了性能的差异。以后的优化工作可以着重去优化软件扇出树的内部结构。

表1 各扇出树生成算法生成的扇出树的平均高度

	Average fanout tree height			
	Balanced	Huffman	H/C	Profile
ammp_1	3.04	3.04	3.04	3.04
ammp_2	3.03	3.03	3.03	3.03
art_1	3.25	3.24	3.25	3.22
art_2	2.81	2.81	2.83	2.76
art_3	3.23	3.25	3.24	3.22
bzip2_1	3.18	3.18	3.18	3.18
bzip2_2	3.07	3.07	3.07	3.07
bzip2_3	3.14	3.14	3.14	3.14
doppler	3.10	3.08	3.08	3.08

结束语 本文提出了一种基于剖析信息和关键路径长度的软件扇出树生成算法。本算法利用了目标指令的执行概率和关键路径长度等信息来构造一棵软件扇出树，以便把指令的计算结果播送给目标指令。根据目标指令的执行概率和关键路径长度信息，本算法优先把计算结果播送给执行频率较高的和关键路径较长的目标指令。实验结果证实了本算法较传统的平衡扇出算法、Huffman算法和Hartley与Casavant算法在性能上有所提高。但是该算法还有待进一步的研究，即着重提高剖析信息和关键路径长度的计算精度，赋给剖析信息和关键路径长度不同的权重等。

参考文献

[1] Ronen R, Mendelson A, Lai K, et al. Coming Challenges in Microarchitecture and Architecture [C] // Proceedings of The IEEE. 2001:325-340

[2] Agarwal V, Hrishikesh M S, Keckler S W, et al. Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures [C]// Proceedings of the 27th Annual International Symposium on Computer Architecture. July 2000:248-259

[3] Albonesi D H. Dynamic opc/block rate optimization [C]// Proceedings of the 20th Annual International Symposium on Computer Architecture. June 1998:282-292

[4] Amruthur B S, Horowitz M A. Speed and power scaling of SRAMs [J]. IEEE Journal of Solid State Circuits, 2000, 35(2): 175-185

[5] Burger D, Keckler S W, Dahlin M, et al. Scaling to the End of Silicon with EDGE Architecture. Computer [J]. IEEE Computer Society, 2004, 37(7):44-55

[6] Huffman D A. A method for the contruction of minimum-redundancy codes[C]// Proceedings of the IRE. September 1952, 40: 1098-1101

[7] Hartley R, Casavant A. Tree-height minimization in pipelined architectures[J]. IEEE Transactions on Computer Aided Design, 1989, 8:112-115

(上接第211页)

[7] Duarte A, Martí R. Tabu search and GRASP for the maximum diversity problem [J]. European Journal of Operational Research, 2007, 178:71-84

[8] Ghosh J B. Computational aspects of the maximum diversity problem [J]. Operations Research Letters, 1996, 19:175-181

[9] Palubeckis G. Iterated tabu search for the maximum diversity problem [J]. Applied Mathematics and Computation, 2007, 189: 371-383

[10] Hopfield J J, Tank D W. Neural computation of decisions in optimization problems [J]. Biological Cybernetics, 1985, 52: 141-152

[11] 旷章辉，王甲海，周雅兰. 用改进的竞争Hopfield神经网络求解多边形近似问题[J]. 计算机科学，2009，36(3)：179-182

[12] Smith K A, Abramson D, Duke D. Hopfield neural networks for timetabling: formulations, methods, and comparative results [J]. Computers & Industrial Engineering, 2003, 44: 283-305

[13] He H, Sykora O, Makinen E. An improved neural network model for the two-page crossing number problem [J]. IEEE Trans. Neural Network, 2006, 17(6): 1642-1646

[14] Hansen P, Mladenovic N, Moreno-perez J A. Variable neighborhood search [J]. European Journal of Operational Research, 2008, 191: 593-595

[15] 潘全科，王文宏，朱剑英，等. 基于粒子群优化和变邻域搜索的混合调度算法[J]. 计算机集成制造系统，2007，13(2)：323-328

[16] Lan G, DePuy G W. On the effectiveness of incorporating randomness and memory into a multi-start metaheuristic with application to the Set Covering Problem [J]. Computers & Industrial Engineering, 2006, 51: 362-374

[17] Créput J C, Koukam A. A memetic neural network for the Euclidean traveling salesman problem [J]. Neurocomputing, 2009, 72(4-6): 1250-1264

[18] Dongarra J J. Performance of various computers using standard linear equations software [R]. CS-89-85. Department of Computer Science, University of Tennessee, USA. http://www.netlib.org/benchmark/performance.ps, 2007

[19] Saif S M, Abbas H M, Nassar S M. An FPGA implementation of a competitive Hopfield neural network for use in histogram equalization [C]// Proceeding of 2006 International Joint Conference on Neural Networks. 2006: 2815-2822