

基于架构负载感知的虚拟机聚簇部署算法

王光波^{1*}, 马自堂¹, 孙磊¹, 吴乐²

(1. 信息工程大学, 郑州 450004; 2. 中国人民解放军 94353 部队, 河南 商丘 461000)

(* 通信作者电子邮箱 691759571@qq.com)

摘要: 针对云计算中虚拟机部署问题, 提出了一种基于架构负载感知的虚拟机聚簇部署算法。首先计算云数据中心各层架构的负载, 并在架构内对主机进行聚簇。虚拟机进行部署时, 先按照相应的规则进行虚拟机间的聚簇, 并优先选择负载较低的架构进行部署, 架构选择后, 进行虚拟机簇与主机簇的匹配以选择最优的主机簇进行部署。最后通过 CloudSim 进行仿真验证, 将其与贪婪算法及基于架构负载感知的基本部署算法进行比较, 证明了算法在部署时间方面有明显的优越性, 并提高了网络资源的利用率。

关键词: 云计算; 虚拟机; 架构负载感知; 聚簇; 可获得性限制

中图分类号: TP302 **文献标志码:** A

Deployment of virtual machines with clustering method based on frame load awareness

WANG Guangbo^{1*}, MA Zitang¹, SUN Lei¹, WU Le²

(1. Information Engineering University, Zhengzhou Henan 450004, China;

2. PLA Troops of 94353, Shangqiu Henan 461000, China)

Abstract: Concerning the deployment of virtual machines in the cloud computing, an algorithm on the deployment of virtual machines with clustering method based on frame load awareness was proposed. First of all, the load of each layer in datacenter was computed and the clustering of physical machines in each layer was constructed. In the process of deploying virtual machines, the clustering of virtual machines was first done according to some rules and then the frame with lower load was chosen preferentially. The last step was to match the virtual machines cluster and physical machines cluster in order to deploy the optimal physical machines cluster. The performance of the algorithm was validated with the experiments in CloudSim. The result was compared to that of the greedy algorithm and basic deployment algorithm with the frame load awareness, which shows that the algorithm proposed in this article has evident priority, and improves the utilization rate of network resources.

Key words: cloud computing; virtual machine; frame load awareness; clustering; availability restriction

0 引言

计算机软件与硬件技术的飞速发展导致了计算模型的不断演化。继分布式计算、并行计算和网格计算等模型之后, 计算机工业界提出了云计算模型^[1]。云计算通过互联网将超大规模的计算存储资源整合起来, 形成一个虚拟的计算资源池, 并以服务的形式按需提供给用户。

虚拟化技术^[2]是云计算的关键技术, 它通过将一台物理服务器分割为若干个相互隔离的虚拟服务器, 实现对于物理资源的动态分割, 提高系统的资源利用率, 降低管理难度, 同时屏蔽了底层硬件资源的异构性。虚拟化技术的应用使得云计算服务部署从复杂的构建服务器集群, 转换为更为快捷简便的对于虚拟机的部署。随着虚拟化成为了云计算的核心技术, 虚拟机部署问题也成为了云计算的一个关键问题。

随着云计算的日益成熟, 虚拟机的部署也从单个虚拟机的部署转变成虚拟机簇的部署, 虚拟机簇即为属于某个用户应用程序的具有通信需求和部署限制的多个虚拟机的集合。为了提高服务质量, 云基础设施提供者需要有效地部署用户的虚拟机簇, 使之既能满足可靠性要求, 又具有较高的应用性

能, 而且虚拟机的部署时间应该合理。这样的部署为 NP-hard^[3]问题, 目前还没有多项式最优解算法。

研究者大多采用启发式方法进行全局优化搜索, 解决虚拟机部署下的约束满足问题^[4-5]。文献[6]提出了基于图分解的启发式部署算法, 但是只考虑了在一台主机上部署一个虚拟机的情况; 文献[7]提出了一种基于聚簇的启发式算法, 但是只考虑了服务器无负载时的情况; 文献[8]提出了一种分解解决方案, 某些相互连接的物理资源首先被确认为部署的目标机器, 而不是在整个系统中进行搜索, 但是该方案只考虑了网络因素; 文献[9]研究了确定虚拟机部署的最优簇问题; 文献[10]则进行了实际的测试实施。

本文主要解决云计算数据中心中虚拟机簇的部署问题, 对物理主机和虚拟机进行聚簇, 提高部署效率和对资源的利用率, 同时考虑用户的可获得性要求。

1 相关研究

1.1 虚拟机簇

虚拟机簇是由多个具有紧密联系的虚拟机组成的集合, 它通常构成某个应用程序的部署单元。在云计算环境下, 应

收稿日期: 2012-11-05; 修回日期: 2012-12-17。 基金项目: 武器装备预研重点基金资助项目(9140A15060311JB5201)。

作者简介: 王光波(1987-), 男, 山东青岛人, 硕士研究生, 主要研究方向: 云计算、系统优化; 马自堂(1962-), 男(回族), 安徽肥西人, 教授, 主要研究方向: 信息安全、密码系统工程; 孙磊(1973-), 男, 江苏靖江人, 副研究员, 博士, 主要研究方向: 云计算基础设施可信增强、可信虚拟化技术; 吴乐(1986-), 男, 江苏徐州人, 主要研究方向: 云计算。

小带宽。

因此定义变量 RH_i 代表主机的负载和网络特性,即

$$RH_i = \alpha L_{PM_i} / N_{PM_i}$$

其中: α 为调和参数,且 $\alpha > 0$ 。将主机按 RH_i 的大小升序排列。

2.2.2 聚簇

按照排序后的主机列表进行聚簇,每产生一个簇都进行主机列表更新,直到所有主机都加入到某个簇中,簇中主机的个数至少为1。定义变量 $Chance_{PM_i}$ 来决定主机 PM_i 是否允许加入到已存在的簇中。

$$Chance_{PM_i} = L_i + w \sqrt{H_i^2 + B_i^2}$$

其中:

$$L_i = \frac{\sum_{PM_j \in cluster_{PM_k}} |L_{PM_j} - L_{PM_i}|}{|numbers(cluster_{PM_k})|}$$

$$H_i = \frac{\sum_{PM_j \in cluster_{PM_k}} \frac{hops(PM_i, PM_j) - 1}{hops(PM_i, PM_j) + 1}}{|numbers(cluster_{PM_k})|}$$

$$B_i = \frac{\sum_{PM_j \in cluster_{PM_k}} \left(1 - \frac{\min_{LK_k \in path(PM_i, PM_j)} ba_k}{bc_k}\right)}{|numbers(cluster_{PM_k})|}$$

L_i 考虑的是负载因素,而 H_i 和 B_i 考虑的是网络特性因素, $|numbers(cluster_{PM_k})|$ 表示簇 $cluster_{PM_k}$ 中主机的数量, w 是调和参数。给定一个阈值 $threshold_{PM}$, 若 $Chance_{PM_i} \leq threshold_{PM}$ 则主机 PM_i 加入到簇 $cluster_{PM_k}$ 中。聚簇算法流程如图4所示。

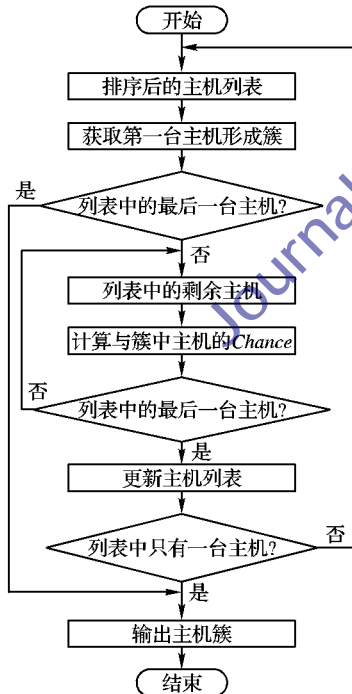


图4 聚簇过程流程

聚簇算法具体步骤如下:

- 1) 获取排序后的主机列表;
- 2) 取出主机列表中的第一台主机形成一个簇;
- 3) 判断其是否为主机列表中的最后一台主机,若是则跳转到7);
- 4) 依次取出主机列表中的剩余主机,计算其与簇中主机的 $chance$,若 $chance \leq threshold_{PM}$, 主机加入到簇中;

5) 更新主机列表,更新后的列表仍为降序排列;

6) 判断更新后的主机列表是否只有一台主机,若是则自成一簇,否则跳转到2);

7) 退出循环,输出主机簇,而且各主机簇已按照负载和网络因素降序排列。

2.3 架构负载计算

架构负载计算模块根据全局监控器提供的主机负载按组织架构计算每一层的负载。首先计算每个刀片数据中心的负载,定义刀片数据中心的负载如下:

$$L_{bladecenter_k} = \frac{\sum_{PM_i \in bladecenter_k} L_{PM_i}}{|numbers(bladecenter_k)|}$$

其中: $L_{bladecenter_k}$ 表示第 k 个刀片数据中心的负载, $|numbers(bladecenter_k)|$ 表示刀片数据中心包含主机的数量。接着计算每个机架的负载,定义机架的负载如下:

$$L_{rack_k} = \sum_{PM_i \in rack_k} L_{PM_i} / |numbers(rack_k)|$$

其中: L_{rack_k} 表示第 k 个机架的负载, $|numbers(rack_k)|$ 表示机架包含主机的数量。在进行虚拟机部署时为了改善整个系统的负载均衡状况,每层架构的负载按升序排列,即负载越低,优先权越高。

2.4 虚拟机簇部署

本节主要研究如何得到架构内的部署簇以及选择合适的主机簇来部署虚拟机簇。部署过程主要分为以下两个阶段。

2.4.1 虚拟机聚簇

本节主要根据虚拟机簇的部署限制,包括资源限制和可获得性限制,将虚拟机簇按架构进行划分,形成各个架构簇。虚拟机簇优先选择负载低的机架和刀片数据中心进行部署,这样做可以改善整个云计算数据中心的负载状况。架构簇形成后,在各个架构内划分部署簇,部署簇是进行整体部署的单元,可最终与主机簇进行匹配,部署在合适的主机簇上。

首先对虚拟机簇中的虚拟机进行排序,定义公式如下:

$$Rank_i = n_{link_i} * (linkbw_i + Resdemand_i) / 2$$

其中: n_{link_i} 表示簇中与虚拟机 i 有通信需求的虚拟机个数, $linkbw_i$ 表示其通信要求的带宽之和, $Resdemand_i$ 则表示虚拟机 i 的资源需求,有

$$linkbw_i = \sum_{VM_j \in links(VM_i)} bw_{i,j}$$

$$Resdemand_i = \sum_{r=1}^{n_r} k_i r_i$$

其中: $bw_{i,j}$ 表示虚拟机 i 和 j 之间的带宽需求; $Rank$ 值最高的虚拟机作为划分的入口点,根据部署限制矩阵 VC 及虚拟机之间的带宽需求进行划分。其设计的具体算法如下:

算法1 虚拟机聚簇算法。

```
frame_cluster (VC[ ][ ], BW[ ][ ], VMs[ ], load_frame) {
//VC[ ][ ] 代表可获得性限制矩阵
//VMs[ ] 代表排序后的虚拟机
//BW[ ][ ] 代表虚拟机之间的带宽需求
//load_frame 代表架构(rack, bladecenter) 的负载
for each VM ∈ VMs[ ] {
if (frame_cluster == ∅) {
j ← min_load (rack);
//min_load (rack) 函数找出负载最低的机架
k ← min_load (j, bladecenter);
//min_load (j, bladecenter) 函数找出机架j中负载最低的
//刀片数据中心
```



```

add VM to frame_cluster[j][k];
VMs.delete(VM); //更新虚拟机列表
else {
for each VMcurrent ∈ frame_cluster {
if (VC[VM][VMcurrent] == 1) {
j ← min_load(VMcurrent)
//min_load(rack - j) 函数找出
//排除虚拟机 VMcurrent 所在机架后负载最低的机架
k ← min_load(j, blade_center);
add VM to frame_cluster[j][k];
VMs.delete(VM);
return;
}
if (VC[VM][VMcurrent] == 2) {
k ← min_load(VMcurrent, blade_center - k);
//min_load(j, blade_center - k) 函数找出虚拟机 VMcurrent
//所在机架中排除刀片数据中心 k 后负载最低的
//刀片数据中心
add VM to frame_cluster[j][k];
VMs.delete(VM[i]);
return;
}
}
for each VM ∈ VMs[ ] {
m, n ← findMaxcomputeBW(VM, cluster)
//findMaxcomputeBW(VM, cluster) 函数找出虚拟机 i 和
//各簇中虚拟机带宽需求之和最大的机架和刀片数据中心
add VM to frame_cluster[m][n];
}
return frame_cluster;
}

```

算法得到在各机架的刀片数据中心中需要部署的虚拟机簇,接下来再划分部署簇,在每个架构簇内实施如下算法:

算法2 虚拟机部署簇算法。

```

deployment_cluster
(VC[ ][ ], BW[ ][ ], frame_cluster, thresholdvm) {
//frame_cluster 代表该架构内的虚拟机簇,而且已经按
//Rank 值进行了降序排列,thresholdvm 代表阈值
j ← 0;
for each VMcurrent ∈ frame_cluster {
if (deployment_cluster == ∅) {
add VMcurrent to deployment_cluster[j];
}
else {
for each VM ∈ deployment_cluster {
if (VC[VM][VMcurrent] = -∞) {
add VMcurrent to deployment_cluster[VM];
//将虚拟机 VMcurrent 加入到 VM 所在的部署簇内
return;
}
value ← maxcbw(VMcurrent, deployment_cluster)
//maxcbw 函数计算当前虚拟机和各部署簇中虚拟机的
//带宽需求之和,找出最大值 value
if (value < thresholdvm) {
add VMcurrent to deployment_cluster[j];
}
else {
add VMcurrent to deployment_cluster[value];
}
//将虚拟机 VMcurrent 加入到带宽需求之和为最大值 value
//所在的部署簇中
}
}
return deployment_cluster;
}

```

算法输出的是架构内的部署簇。

2.4.2 主机簇匹配部署

主机簇匹配即选择合适的主机簇部署虚拟机簇,为了改善系统的负载,提高服务质量,虚拟机簇优先选择负载低和网络特性好的主机簇进行部署,即按照主机聚簇步骤输出的主机簇列表进行选择。

首先考虑虚拟机簇对主机簇中主机数的要求, $n_{PM_{cluster}}$ 代

表主机簇中主机的个数, $n_{VM_{cluster}}$ 代表虚拟机簇需要的最少主机数量,若 $n_{PM_{cluster}} < n_{VM_{cluster}}$, 则排除该主机簇。确定变量 $n_{VM_{cluster}}$ 的数值时,需要考虑可获得性限制矩阵 VC , $vc_{ij} = +\infty$ 表示虚拟机必须部署在不同的主机上;若 $vc_{ij} = -\infty$ 则表示虚拟机必须部署在同一台主机上,则将虚拟机 vm_i 和 vm_j 作为一个单元来看待,其对资源的需求为二者之和。

匹配部署算法的具体步骤如下:

- 1) 获取排序后的主机簇列表;
- 2) 依次取主机簇列表中的主机簇;
- 3) 对主机簇中的主机按照负载高低进行降序排列;
- 4) 将虚拟机簇中的虚拟机按照对资源的需求进行降序排列;
- 5) 依次取出排序后虚拟机簇中的虚拟机,若无虚拟机可取,则转到步骤8);
- 6) 依次取出排序后主机簇中的主机,若无主机可取,则清空部署列表,转到步骤2);
- 7) 将虚拟机与主机依次进行匹配,确定主机及主机间的链路带宽是否满足虚拟机的性能需求,若满足则将其加入部署列表,并更新主机资源和虚拟簇,转到步骤3);若不满足则转到步骤6);
- 8) 退出循环,输出部署列表。

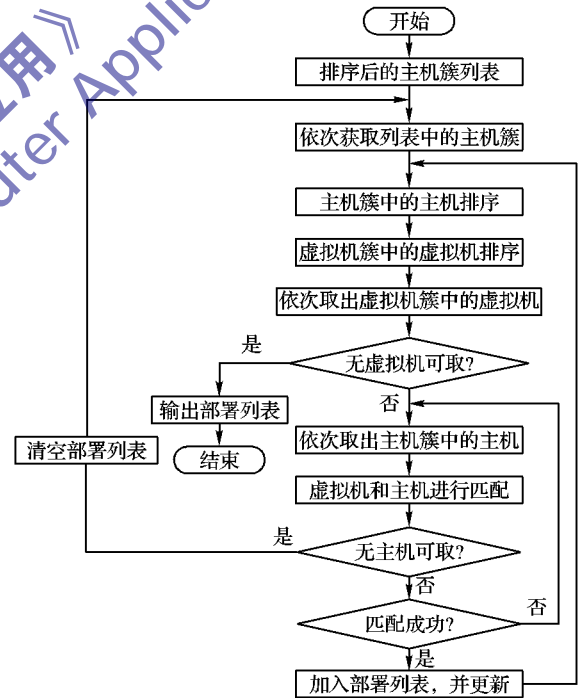


图5 主机簇匹配部署流程

3 仿真实验与分析

3.1 实验环境

本文采用澳大利亚墨尔本大学的网络实验室和 Gridbus 项目提出的云仿真平台 CloudSim^[13] 作为实验仿真工具,使用版本为 CloudSim-2.1.1。

首先建立一个 Datacenter,其规模由机架个数决定,每个机架包含4个刀片数据中心 (blade_center),每个刀片数据中心包含16台主机 (Host),每台主机包含32个处理单元 (Pe, 其处理能力为200 MIPS),32 GB 的内存 (RAM),1 TB 的硬盘 (storage),10 000 Mb/s 的带宽 (BW),每条网络链路的通信能力为1 024 Mb/s。

完成数据中心的初始化工作后,提交如图6所示的虚拟

机簇 VMC , 边上的数字代表虚拟机间的带宽需求。

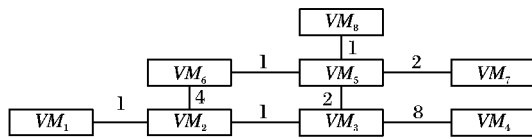


图6 虚拟机簇 VMC

各虚拟机的资源需求如表1所示。

表1 虚拟机资源需求列表

虚拟机	Pe	内存/GB	硬盘/GB	带宽/(Mb · s ⁻¹)
VM_1	1	2	100	1 000
VM_2	1	2	100	1 000
VM_3	4	5	400	1 500
VM_4	4	5	400	2 000
VM_5	4	5	400	1 500
VM_6	4	5	400	1 000
VM_7	8	10	800	1 500
VM_8	8	10	800	1 500

虚拟机间的可获得性矩阵 VC 为:

$$VC = \begin{matrix} & VM_1 & VM_2 & VM_3 & VM_4 & VM_5 & VM_6 & VM_7 & VM_8 \\ \begin{matrix} VM_1 \\ VM_2 \\ VM_3 \\ VM_4 \\ VM_5 \\ VM_6 \\ VM_7 \\ VM_8 \end{matrix} & \begin{bmatrix} -\infty & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -\infty & 0 & 0 & +\infty & -\infty & 0 & 0 \\ 0 & 0 & -\infty & 0 & +\infty & 0 & 0 & 0 \\ 1 & 0 & 0 & -\infty & 0 & 0 & 0 & 0 \\ 0 & +\infty & +\infty & 0 & -\infty & 0 & 0 & 0 \\ 0 & -\infty & 0 & 0 & 0 & -\infty & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\infty & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\infty \end{bmatrix} \end{matrix}$$

3.2 部署时间分析

为了验证聚簇步骤的重要性,将其与贪婪算法、架构感知的基本部署算法进行对比。贪婪算法部署流程为从数据中心中选出负载最低的第一台主机,并尝试在其上创建虚拟机;如果失败了就排除当前选择的这台主机,重返部署队列接着选择负载最低的主机进行部署,直到所有虚拟机部署完成。

在仿真实验中,首先随机部署虚拟机簇使整个系统的负载达到40%,然后再运用算法分别部署图6中的虚拟机簇,取参数 $threshold_{host} = 0.45$, $threshold_{vm} = 0.35$,对比其部署时间的差异,如图7所示。可以发现采用聚簇技术的算法部署时间明显减少,而且随着数据中心规模的增大基本不变,具有很好的可扩展性。

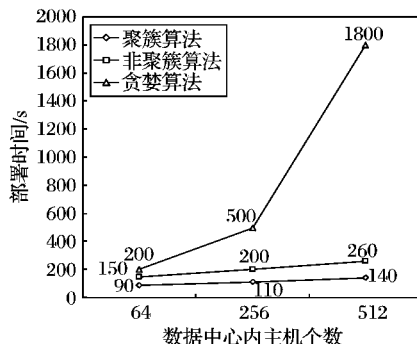


图7 部署时间对比

3.3 网络性能分析

为了进一步验证聚簇技术对网络特性的影响,定义测试

变量 $NetTest$ 如下:

$$NetTest = \sum_{VM_i, VM_j \in VMC} BW(VM_i, VM_j) * hops(VM_i, VM_j)$$

其中: $BW(VM_i, VM_j)$ 代表虚拟机 VM_i 和 VM_j 之间的带宽需求, $hops(VM_i, VM_j)$ 则代表虚拟机之间的跳数。 $NetTest$ 表示虚拟机通信占用的整个系统的网络带宽,实验结果如图8所示。可以看出本文算法优先其他算法,这主要得益于聚簇技术将相互通信比较大的虚拟机放在同一主机簇中,提高了网络的利用率。

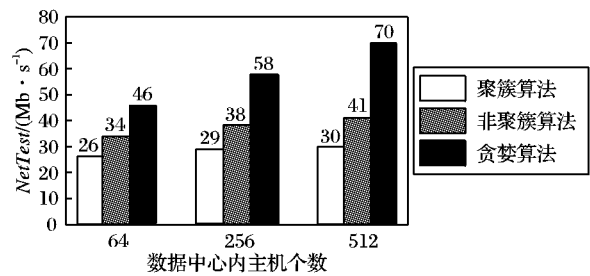


图8 网络性能对比

4 结语

本文使用基于架构负载感知的虚拟机聚簇部署算法进行虚拟机批量部署,虚拟机进行部署时,将虚拟机进行聚簇,将其与主机簇进行匹配。本文算法同时考虑了系统的负载均衡和网络特性因素,并通过 Cloudsim 进行仿真验证,将其与贪婪算法及基于架构负载感知的基本部署算法进行比较,证明了本算法的优势。

参考文献:

- [1] ARMBURST M, FOX A, GRIFFITH R, et al. A view of cloud computing[J]. Communications of the ACM, 2010, 53(4): 50-58.
- [2] 董耀祖,周正伟. 基于 X86 架构的系统虚拟化技术与应用[J]. 计算机工程, 2006, 32(13): 71-73.
- [3] BENGOTEXEA E. Inexact graph matching using estimation distribution algorithm[D]. Pairs: Ecole Nationale Supérieure des Telecommunications, 2003.
- [4] LI B, LI J-X, HUANG J-P, et al. EnaCloud: an energy-saving application live placement approach for cloud computing environments [C]// 2009 IEEE International Conference on Cloud Computing. Piscataway: IEEE, 2009: 17-24.
- [5] GUPTA R, BOSE S. K, SUNDARRAJAN S, et al. A two stage heuristic algorithm for solving the server consolidation problem with item-item and bin-item incompatibility constraints[C]// SCC'08: Proceedings of the 2008 IEEE International Conference on Services Computing. Piscataway: IEEE, 2008: 39-46.
- [6] MENG X, PAPPAS V, ZHANG L. Improving the scalability of data center networks with traffic-aware virtual machine placement[C]// 2010 Proceedings of IEEE INFOCOM. Piscataway: IEEE, 2010: 1-9.
- [7] JAYASINGHE D, PU C, EILAM T, et al. Improving performance and availability of services hosted on IaaS clouds with structural constraint-aware virtual machine placement[C]// 2011 IEEE International Conference on Services Computing. Piscataway: IEEE, 2011: 72-79.
- [8] ZHU Y, AMMAR M. Algorithms for assigning substrate network resources to virtual network components[C]// The 25th IEEE International Conference on Computer Communications. Piscataway: IEEE, 2006: 1-12.

(下转第 1288 页)

3.3 与其他经典方法的比较

为了验证本文基因选择方法的有效性,将本文方法与其他经典的基因选择方法进行比较。表3中列出了相关方法所获取的最小基因子集及相应的最大分类准确率。从表3中可以看出,对于白血病数据集,算法在达到100%分类率的情况下,选择的基因子集数目最少。对于结肠癌数据集,在选择同样数目的基因子集的情况下,样本的分类率高于基因排序算法^[21]。该实验结果进一步验证了本文基因选择方法能够选出更少冗余的基因,同时能够保证高的样本分类率。

表3 几种算法的分类准确率(最小基因子集规模)

基因选择方法	白血病		结肠癌	
	准确 率/%	基因子 集规模	准确 率/%	基因子 集规模
本文方法	100	3	93.53	3
最大相关最小冗余 ^[15]	100	11	89.06	28
基因排序 ^[21]	100	8	91.90	3
信息增益和微粒群优化 ^[16]	92.42	79	81.31	42
信噪比 ^[17]	100	5	—	—

4 结语

为了降低基因冗余度,本文提出一种基于聚类和微粒群优化的基因选择算法,应用于研究微阵列数据的基因选择。因为聚类算法可以根据基因的功能将具有相同功能的基因聚成一簇,不同功能的基因聚在不同的簇,通过预处理,含有大量噪声信息的聚类将会被移除,而具有高贡献的基因簇的基因子集组成基因库作为PSO的搜索空间。从实验结果可以看出,混合聚类和PSO的算法能成功选择较少数目但有较高分类率的基因子集。

参考文献:

- ANTONOV A V, TETKO I V, MADER M T, *et al.* Optimization models for cancer classification: extracting gene interaction information from microarray expression data [J]. *Bioinformatics*, 2004, 20(5): 644–652.
- VARMA S, SIMON R. Iterative class discovery and feature selection using minimal spanning trees [J]. *BMC Bioinformatics*, 2004, 5: 126.
- YAO F Z, COQUERY J, LÉCAO K-A. Independent principal component analysis for biologically meaningful dimension reduction of large biological data sets [J]. *BMC Bioinformatics*, 2012, 13: 24–38.
- 于化龙, 顾国昌, 刘海波, 等. 基于相关性分析的微阵列数据集分类研究[J]. *计算机研究与发展*, 2010, 47(2): 328–335.
- 黄德双. 基因表达谱数据挖掘方法研究[M]. 北京: 科学出版社, 2009.
- SAEYS Y, INZA I, LARRANAGA P. A review of feature selection techniques in bioinformatics [J]. *Bioinformatics*, 2007, 23(19): 2507–2517.
- MALDONADO S, WEBER R. A wrapper method for feature selection using support vector machines [J]. *Information Sciences*, 2009, 179: 2208–2217.
- GUYON I, GUNN S, NIKRAVESH M, *et al.* Feature extraction, foundations and applications: studies in fuzziness and soft computing [M]. Berlin: Springer, 2006: 29–203.
- GOLDBERG D E. Genetic algorithms in search, optimization and machine learning[M]. [S. l.]: Addison-Wesley Publishing, 1989.
- KENNEDY J, EBERHART R. Particle swarm optimization [C]// *Proceedings of 1995 IEEE International Conference on Neural Networks*. Piscataway: IEEE, 1995, 4: 1942–1948.
- SHI Y, EBERHART R. A modified particle swarm optimizer [C]// *The 1998 IEEE International Conference on Evolutionary Computation*. Piscataway: IEEE, 1998: 69–73.
- RANA S, JASOLA S, KUMAR R. A hybrid sequential approach for data clustering using K-means and particle swarm optimization algorithm [J]. *International Journal of Engineering, Science and Technology*, 2010, 2(6): 167–176.
- ZALIK K R. An efficient *k*-means clustering algorithm [J]. *Pattern Recognition Letters*, 2008, 29(9): 1385–1391.
- HUANG G B, ZHUO Y, SIEW C K. Extreme learning machine: theory and applications [J]. *Neurocomputing*, 2006, 70(1/2/3): 489–501.
- HUANG G B, DING X, ZHOU H. Optimization method based extreme learning machine for classification [J]. *Neurocomputing*, 2010, 74(1/2/3): 155–163.
- ZHAO W, WANG G, WANG H, *et al.* A novel framework for gene selection [J]. *International Journal of Advancements in Computing Technology*, 2011, 3: 184–191.
- MISHRA D, SAHU B. Feature selection for cancer classification: a signal-to-noise ratio approach [J]. *International Journal of Scientific & Engineering Research*, 2011, 2(4): 1–7.
- GOLUB T R, SLONIM D K, TAMAYO P, *et al.* Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring [J]. *Science*, 1999, 286(5439): 531–536.
- ALON U, BARKAI N, NOTTERMAN D A, *et al.* Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays [J]. *Proceedings of the National Academy of Sciences USA*, 1999, 96(12): 6745–6750.
- TONG D L. Hybridising Genetic Algorithm - Neural Network (GANN) in marker genes detection [C]// *Proceedings of the Eighth International Conference on Machine Learning and Cybernetics*. Piscataway: IEEE, 2009: 1082–1087.
- WANG Y, MAKEDON F, FORD J C, *et al.* HykGene: A hybrid approach for selecting marker genes for phenotype classification using microarray gene expression data [J]. *Bioinformatics*, 2005, 21(8): 1530–1537.
- MACROPOL K, SINGH A. Scalable discovery of best clusters on large graphs[EB/OL]. [2012-10-01]. <http://www.vldb.org/pvldb/vldb2010/papers/R62.pdf>.
- RICCI R, ALFELD C, LEPREAU J. A solver for the network testbed mapping problem[J]. *SIGCOMM Computer Communications Review*, 2003, 33(2): 65–81.
- NATHUJI R, KANSAL A. Q - clouds : Managing performance interference effects for QoS-aware clouds[C]// *Proceedings of the 5th European conference on Computer Systems*. New York: ACM, 2010: 237–250.
- GRAEPEL T. Statistical physics of clustering algorithms, Technical Report 171822[R]. Köln: Universität zu Köln, Institut für theoretische Physik, 1998.
- RODRIGO N, CALHEIROS, RAJIV RANJAN, *et al.* CloudSim: A novel framework for modeling and simulation of cloud computing infrastructures and services, GRIDS-TR-2009-1[R]. Melbourne, Australia: The University of Melbourne, Grid Computing and Distributed Systems Laboratory, 2009.

(上接第1275页)