一种求解异构DAG调度问题的置换蚁群

邓 蓉 陈闵中 王 博 王小明 李 灿
(同济大学计算科学与技术系 上海 201804)

摘 要 减少分布式程序的执行时间,是网格调度系统需要解决的重要问题。因分布式程序常建模为DAG图,故该问题又称异构DAG调度问题。提出的置换调度蚁群PSACS(Permutation Scheduling Ant Colony System)将DAG调度方案表示为任务置换列表,使用标准蚁群搜索技术探索解空间。实验表明,该算法明显优于遗传算法和粒子群算法,能够一次求出大部分(65%)同构DAG调度问题的最优解并获得非常好的异构DAG调度方案。

关键词 网格,DAG调度,蚁群优化

Permutation Ant Colony System for Heterogeneous DAG Scheduling Problem

DENG Rong CHEN Hong-zhong WANG Bo WANG Xiao-ming LI Can
(Department of Computer Science and Technology, Tongji University, Shanghai 201804, China)

Abstract Reducing execution time of distributed program is a major issue of Grid Scheduling System. Because scheduled programs are modeled by DAG, this problem is also called Heterogeneous DAG scheduling problem. Permutation Scheduling Ant Colony System(PSACS) proposed by this paper presents solution of this problem as task permutation list and utilizes standard ACO searching technique to explore solution space. Experimental result indicates that PSACS outperforms GA and PSO substantially. It can get global optima for the majority(65%) of homogeneous DAG scheduling problems and pretty good solutions for heterogeneous DAG scheduling problems.

Keywords Grid, DAG scheduling, Ant colony optimization

在异构系统中运行着许多科学工作流或商业工作流,通常人们用有向无环图(Directed Acyclic Graph, DAG)为这些分布式程序建模。为了满足程序的性能要求,网格必须提供足够的资源管理和调度服务,以有效解决异构DAG图的调度问题。该问题以缩短整个程序的运行时间为优化目标,主要考虑为程序所包含的所有任务分配处理器资源并优化它们执行的先后次序。这是一个NP完全难题[1],不存在多项式时间复杂度的解。过去的十几年,研究者们从人类日常生活经验和生物群落行为中汲取灵感,针对此问题提出了许多启发式算法(如文献[2-4])和演化计算方法(如文献[5-7])。虽未能完满解决所研究的问题,但这些算法为后续研究奠定了坚实的基础。

蚁群优化(Ant Colony Optimization, ACO)是一类新兴的演化计算方法。1992年,意大利学者Marco Dorigo在其博士论文[8]中首次提出该方法,并成功将其用于求解旅行商问题(Traveling Salesman Problem, TSP)。该算法模仿蚁群找食过程中所体现的社会行为,使用正反馈(positive feedback)的方法将先前蚂蚁构造优秀解的步骤用信息素(pheromone)的方式存储在系统之中,后续蚂蚁在决策时将综合知识和先前蚂蚁的经验构造新的解。

经过近10年的研究,蚁群优化方法逐步成熟,成为继遗传算法之后求解组合优化难题的另一种有效途径。实验表明该方法在求解排序问题、开放车间调度问题、车辆寻径问题和分类等问题时,性能优于所有其它已知算法;求解其它组合优化难题时,ACO算法亦可与最优的其它算法相媲美[9]。受此结论鼓舞,笔者尝试用ACO方法求解异构DAG图调度问题,得到算法PSACS(Permutation Schedule Ant Colony System)。本算法将DAG调度方案表示为任务置换列表;使用蚁群系统(Ant Colony System,最有效的蚁群优化方法之一[9])的标准搜索技术对解逐步求精。实验表明,本算法性能优秀,能够一次求出大部分(约65%)同构DAG调度问题的最优解并获得较好的异构DAG调度方案(21%的异构问题经过PSACS的调度,makespan不到HEFT的90%,最高改善比可达27.59%)。

本文第2节对异构DAG调度问题进行形式化描述;第3节介绍PSACS算法;第4节对实验结果进行分析;最后总结全文。

1 异构DAG调度问题

分布式程序调度,又称异构DAG调度问题,是网格资源管理必须解决的重要问题。为了缩短整个程序的运行时间(或对开销等其它指标进行优化),调度器需要为程序所包含的所有任务分配处理器资源并优化它们执行的先后次序。

建模被调度程序为有向无环图:$\Omega(\Gamma, \Lambda)$:任务集 $\Gamma=$

到稿日期:2010-01-12 返修日期:2010-03-30 本文受863国家重点基金项目(No. 2009AA01Z141)资助。

邓 蓉(1973-),女,讲师,主要研究方向为操作系统、分布式计算等,E-mail:drong2003@gmail.com;陈闵中(1951-),男,博士生导师,主要研究方向为操作系统、分布式计算等;王 博,王小明,李 灿 男,硕士,主要研究方向为操作系统、分布式计算等。

$\{t_i\}, i\in[0,N-1]$，N 是任务数；边集 $\Lambda=\{e_{i,j}\}, i,j\in[0,N-1]$，$e_{i,j}$ 表示任务 t_i 和 t_j 之间的数据依赖关系：t_i 是 t_j 的直接前驱，t_j 是 t_i 的直接后继，边的权重为 t_i 运行结束后需要向 t_j 传送的数据量。后继任务在收到所有前驱发送的数据之前不能开始运行；若分配给任务 t_i 和 t_j（直接前驱后继任务）的处理器相同，数据传输时间为 0。称图中没有前驱的节点为入口节点，没有后继的节点为出口节点。下述的术语节点与任务等价。

程序的运行环境为一组全连接的处理器集合（连接在 Internet 上的每台处理器均可直接与其它处理器通信，故该假设适用于计算网格）。任务可以在所有处理器上运行，其执行代价用矩阵 ETC 表示。每台处理器空分、非剥夺，也就是说一旦任务开始便在某处理器上开始执行，独占该处理器直至运行结束。表 1 是本文所使用的记号及它们的含义。

表 1 符号和定义

符号	定义
$prec(i)$	任务 t_i 的直接前驱集合。
$succ(i)$	任务 t_i 的直接后继集合。
$proc(i)$	分配给任务 t_i 的处理器。
$C(i,j)$	任务 t_i 和 t_j 之间数据传输所需的时间。
$ready(t)$	解的构造过程中，调度步骤 t 考察的就绪任务集合，包括所有前驱均已接受调度的任务。$ready(0)=\{t_i \mid i\in[0,N-1] \text{ and } prec(i)=\emptyset\}$，含所有入口节点。
$ETC(i,j)$	Estimated Time to Complete，任务 t_i 在处理器 p_j 上的执行时间。
$MAT(j)$	Machine Available Time，处理器 p_j 在时刻 $MAT(j)$ 之前忙，新分配的任务只能在此时刻之后运行。
$DRT(i,j)$	Data Ready Time，任务 t_i 在处理器 p_j 处的数据就绪时刻，用式(1)计算。 if $prec(i)\neq\emptyset$ $DRT(i,j)=\max(\max\limits_{\substack{k\in prec(i)\\proc(k)\neq j}}(CT(k)+C(k,i)), \max\limits_{\substack{k\in prec(i)\\proc(k)=j}}(CT(k)))$ else $DRT(i,j)=0$ (1)
$EST(i,j)$	Earliest Start Time，任务 t_i 在处理器 p_j 处的最早开始执行时刻，用式(2)计算。 $EST(i,j)=\max(DRT(i,j),MAT(j))$ (2)
$ST(i)$	Start Time，任务 t_i 的开始执行时刻。$ST(i)=ST(i,proc(i))$，为任务 t_i 分配处理器后，其值可定。
$CT(i,j)$	Complete Time，任务 t_i 在处理器 p_j 处的最早结束时刻。 $CT(i,j)=EST(i,j)+ETC(i,j)$ (3)
$CT(i)$	Complete Time，任务 t_i 的结束时刻。$CT(i)=CT(i,proc(i))$，为任务 t_i 分配处理器后，其值可定并可用于计算任务 t_i 后继的最早开始执行时刻。
s	异构 DAG 调度问题的一个解（调度方案），含所有任务的处理器分配方案和开始、结束时刻。
$MK(s)$	MakeSpan，s 的调度长度。 $MK(s)=\max\limits_{i\in[0,N-1]}(CT(i))$ (4) 异构 DAG 调度问题的优化目标为： $\min\limits_{S\in\text{feasible solution set}}(MK(s))$。

2 PSACS

旅行商问题是 ACO 成功解决的第一个问题[9]。求解该问题时，蚂蚁给出的每个解$[c_1,c_2,\cdots,c_n]$都是城市号码的一个置换（c_i 表示旅行商访问的第 i 个城市）。因此任何可转化为置换调度的问题均适合用 ACO 方法求解。PSACS 将解 s 表示为任务的置换列表 $l=\{t_0,t_1,\cdots,t_i,\cdots,t_{N-1}\}$。任意置换列表 l 可如下展开为唯一的调度方案：按任务在列表中的出现次序，依次将它们调度至可令其最早结束的处理器，若有多台处理器均可令当前任务最早结束，取下标最小者。Algorithm1 列出了蚁群 PSACS 求解 DAG 调度问题的主要步骤。算法使用 10 只蚂蚁（num-ants=10），终止条件为循环次数超过 1000。在循环的每一轮，所有蚂蚁均需使用 Procedure1 得到问题的一个解（含局部更新操作）；单轮循环结束后，PSACS 根据式(9)用精英蚂蚁（elite）的解对信息素矩阵进行全局更新。

Algorithm1 PSACS

```
Step1  读入 DAG 和 ECT 矩阵，利用式(7)计算每个结点的
       rank 值(优先数)；
Step2  取 ACS 参数，初始化信息素矩阵；
Step3  While stopping criterion not satisfied Do
         Repeat for num-ants times
         solution=SolutionConstruction( );
           if(MK(solution)<MK(elite))
             elite=solution;
         EndRepeat;
         GlobalUpdate(elite);
       EndWhile //循环结束后，elite 为 PSACS 的解
```

2.1 解的构造

PSACS 采用表调度思想构造解，见 Procedure1。表调度算法通过缩短关键路径的方法减小调度长度，是解决 DAG 调度问题的有效手段。该类方法首先为任务赋优先级并按优先级递减的次序形成任务调度列表；之后依任务在列表中的出现次序为其分配处理器[2]。Procedure1 使用两个工作集合：$list$ 和 $ready$。$list$ 保存蚁群构造的任务置换列表，其中依次记录所有已经被调度的任务；初始化时所有任务均未调度，$list$ 为空。$ready(t)$ 表示调度步骤 t 考虑的就绪任务集合；初始化时 $ready(0)$ 包含所有入口任务。算法循环 N 次完成解的构造，在每个调度步骤 t：蚂蚁按状态迁移规则（式(5)，式(6)）从 $ready(t)$ 中随机选取任务 n，运用 EFT（Earliest Finish Time，最早完成时间）规则为其分配处理器 p，使之尽早结束（$p=\arg(\min(CT(n,k)))$，$k\in available(t_n)$）；之后蚂蚁登记任务 n 的完成时刻：finishTime[n]$=CT(n,p)$，将其放入置换列表：$list[t]=n$ 并重新计算 $ready$ 集合，使之包含 n 的就绪后继。所有任务调度后，$list$ 即所构造的解，max(finishTime) 为解 $list$ 的调度长度。

Procedure1 PSACS's SolutionConstuction( )

```
Step 1  t=0;
        ready(0) = readyInitialize( );
Step 2  Repeat for N times
        n=Select-Node(ready(t));
        p=Select-Best-Processor(n,t);
        finishTime[n]=CT(n,p);
        list[t]=n;
        Local_Update(τ(t,n));
        t=t+1;
        Recalculate(ready(t));
        EndRepeat;
Step 3  Return (list and max(finishTime) );
```

在蚂蚁构造解的每个调度步骤，$ready$ 中所有元素均有被选中的可能，这意味着低优先级的任务可能先于高优先级任务被调度；这种随机引入的延时操作使 PSACS 得以避免陷入局部最优，从而求得优于启发式算法的解。

2.2 状态迁移规则

人工蚂蚁使用状态迁移规则决定调度步骤 i 被选中的任

务。PSACS使用伪随机概率原则进行状态迁移，见式(5)、式(6)。

$$t_i=\arg\max_{k\in ready(i)}\{[\tau(i,k)][\eta(i,k)]^\beta\},\text{if } q<q_0 \quad (5)$$

$$p_i(j)=\begin{cases}\dfrac{[\tau(i,j)][\eta(i,j)]^\beta}{\sum_{k\in ready(i)}[\tau(i,k)][\eta(i,k)]^\beta}, & \text{if } j\in ready(i)\\ 0, & \text{otherwise}\end{cases} \quad (6)$$

式中，$\tau(i,k)$是信息素矩阵中的元素，表示根据蚁群的经验，调度决策$\langle i,k\rangle$的质量。$\tau(i,k)$值越大，后代蚂蚁采用$\langle i,k\rangle$元素构造解(于步骤i调度任务k)的概率就越大。

η为蚂蚁的调度操作提供启发式知识，能促使算法更快发现最优解，减少求解的盲目性。PSACS将HEFT[2]算法使用的启发式知识植入η。HEFT认为与出口节点相距最远的任务最重要，应当首先接受调度，故η可定义为任务的静态优先级$rank(j)$：

$$\eta(i,j)=\eta(t_j)=rank(j)=$$

$$\begin{cases}\overline{ETC_{j,p}}, & p\in available(j)\wedge succ(j)=\varnothing\\ \overline{ETC_{j,p}}+\max\limits_{k\in succ(i)}(C(j,k)+rank_k), & p\in available(j)\wedge succ(j)\neq\varnothing\end{cases} \quad (7)$$

式中，$available(j)$是可以运行任务t_j的处理器集合。PSACS初始化时计算每个任务的$rank$值，之后该信息作为所有蚂蚁共有的知识，诱导它们在每个调度步骤中选择就绪集合中$rank$值高的任务。

综合考虑蚁群的经验和知识，蚂蚁在构造解时认为就绪集合中$[\tau(i,k)][\eta(i,k)]^\beta$分值最高者为“最佳”任务。参数$\beta$调节知识(常识)和经验的相对重要程度。本文$\beta$取值为1.2(与文献[10]相同)，表示蚂蚁在调度时受启发式知识的影响更大。

回到式(5)、式(6)，蚂蚁在每次状态迁移(调度步骤)时均会生成一个[0,1]之间的随机数q；若q小于q_0，蚂蚁将选择就绪集合中“最佳”的任务式(5)；否则就绪集合中的每个任务均有被选中的机会，蚂蚁利用式(6)计算它们被选中的概率。q_0是蚁群系统的另一个重要参数，负责调节系统利用(exploitation)和探索(exploration)的比例。较大的q_0意味着蚁群偏爱利用知识和已有经验，在绝大多数情况下选择“最佳”任务；否则系统颇具探索精神，经常对“最佳”任务实施延时操作。PSACS中q_0取值0.9。

2.3 解空间搜索技术

2.3.1 局部更新原则(Local Update Rule)

局部更新原则用于防止本代中的后续蚂蚁得到与先前蚂蚁相同的解(式(8))。在构造解的每一步，蚂蚁挥发当前调度步骤$\langle t,n\rangle$所对应的信息素$\tau(t,n)$：

$$\tau(t,n)=\tau(t,n)*(1-\rho)+\tau_0*\rho \quad (8)$$

式中，挥发因子$0\leqslant\rho\leqslant1$，是ACS的又一个重要参数，默认值为0.1。$\tau_0=0.001$，是信息素矩阵元素初值。

2.3.2 全局更新原则(Global Update Rule)

在本轮(iteration)所有蚂蚁完成解的构造之后，系统对信息素矩阵进行全局更新：挥发所有元素之后，对精英蚂蚁(蚁群找到的最优解)的调度步骤进行增强。

$$\tau(t,n)=\tau(t,n)*(1-\delta)+\Delta\tau(t,n)*\delta \quad (9)$$

$$\Delta\tau(t,n)=$$

$$\begin{cases}\dfrac{1+\max(0,makespan_{gbold}-makespan_{interation\ best})}{\min(makespan_{interation\ best},makespan_{gbold})}, & (t,n)\in\text{gb new}\\ 0, & (t,n)\notin\text{gb new}\end{cases} \quad (10)$$

式中，gb old是本轮开始之前系统找到的最优解，iteration best是本轮找到的最优解，gb new是它们中的较好者。挥发因子$0\leqslant\delta\leqslant1$，取值0.1。

3 实验与分析

本文在Udo Hoenig[11]的标准测试集上对PSACS进行了全面测试。在拥有最佳调度方案的所有测试用例中，本文挑选了300个规模最大的问题(位于TB-Optimal-V130406 \ TB-Optimal目录下，每张DAG图拥有19～24个节点)，用测试集给出的最优解衡量所有算法的性能。

为引用方便，按测试用例所属目录对其分组。命名如下：

2pe random：2TPE \ GS19 _ 24 \ NoSRand \ EL _ Rand \ RNodeR Edge

4pe random：4TPE\GS19_24\(同上) \RNodeREdge

8pe random：8TPE\GS19_24\(同上) \RNodeREdge

16pe random：16TPE\GS19_24\(同上) \RNodeREdge

32pe random：32TPE\GS19_24\(同上)\RNodeREdge

8pe high：8TPE\GS19_24\NoSHigh\EL_Rand\RnodeREdge

组名中的整数表示分布式环境中处理器的数量，即n pe表示DAG的运行环境包含n个相同的处理器。目录名中的其它标记刻画DAG图的其它属性，请参阅文献[11]。

3.1 同构调度问题

第一组实验测试PSACS搜索同构调度问题最优解的能力。在Udo Hoeing的标准测试集中，每个目录有两种文件：XXX. stg记录调度问题，同名文件XXX. ssf(若存在)是该问题的最优解。将300个问题的PSACS解与XXX. ssf中记录的最优解相比较，得表2。从统计结果可以看出，PSACS性能明显优于纯遗传算法[5]。对于大多数的问题，PSACS可一次得出问题的最优解，其它非最优解的质量亦较高(与最优解的平均差异仅1%左右)。

表2 PSACS搜索同构调度问题最优解的能力

(随机一次试验的统计结果)

	得到最优解的次数		与最优解的平均差异		与最优解的最大差异	
	PSACS	纯GA	PSACS	纯GA	PSACS	纯GA
2pe random	52.0%	6.0%	1.5%	14.7%	9.2%	33.0%
4pe random	66.7%	9.5%	0.7%	13.9%	8.3%	48.1%
8pe random	68.0%	8.0%	0.8%	11.0%	8.3%	42.0%
16pe random	65.9%	7.3%	0.8%	12.5%	8.3%	40.7%
32pe random	68.0%	10.0%	0.6%	14.0%	8.3%	58.0%
8pe high	68.0%	8.0%	0.5%	12.3%	3.4%	33.7%

蚁群优化方法和遗传算法均为随机搜索算法。为了增强实验数据的说服力，我们对分组8pe random中的每个问题进行了10次调度。结果表明PSACS稳定性极佳：平均而言同一张DAG图的10次调度结果标准差不超过调度长度的1.1%，其中90%的问题在所有实验中求得相同的解。遗传算法不稳定：同一张DAG图的10次调度结果标准差为5.3%且没有任何一张图的10次调度结果相同。此外，我们发现10次GA调度结果中最优的makespan逊于PSACS的最

差结果(平均差异可达 6.4%)。故表 2(及表 3)所列数据虽为随机一次实验的结果,但足以代表相关算法的普遍性能。

3.2 异构调度问题

第二组实验测试 PSACS 求解异构调度问题的性能。就笔者所知,目前没有异构 DAG 调度问题的标准测试集,因此没有最优解可供参照。因 PSACS 嵌入了 HEFT 使用的启发式知识,本文使用 HEFT 作为基准比较其它算法的性能。异构 DAG 图由本文 3.1 节所使用的测试用例转化而来,方法如下:保持 DAG 图结构不变,XXX.stg 文件中登记的任务执行时间 $weight_i$ 为任务 t_i 在异构处理器上运行时间的均值,$ETC_{i,j}$ 为[0,2 * $weight_i$]区间内均匀分布的随机变量。

实验结果(表 3)为:随机对所有 300 个被测试的异构 DAG 调度问题进行一次实验,21%的 PSACS 调度结果优于 HEFT 10%以上,最高改善比可达 27.59%。纯遗传算法的性能远远差于 PSACS:300 个问题中仅有 2 个问题的纯遗传算法的调度结果优于 HEFT 10%(最高改善比 11%)。此结论表明,PSACS 性能不仅优于纯遗传算法,而且优于最近的研究成果(DAG 调度问题的粒子群优化算法[12]),后者取 10 次实验中的最优值,对 HEFT 的改善比最高仅为 18.3%。

表 3 PSACS 求解异构调度问题的能力
(随机一次试验的统计结果)

	优于 HEFT(10%)的次数		对 HEFT 的最优改善比例	
	PSACS	纯 GA	PSACS	纯 GA
2pe random	3	0	12.8%	0.7%
4pe random	5	0	16.2%	−4%
8pe random	14	0	20.3%	1%
16pe random	17	0	24.4%	−29.3%
32pe random	9	0	27.6%	−29.2%
8pe high	14	2	27.4%	10.6%

结束语 本文给出蚁群系统 PSACS,用以解决异构 DAG 调度问题。因恰当地使用了启发式知识,算法性能明显优于遗传算法和粒子群算法。值得注意的是,本文并未使用任何复杂的演化技术,仅使用 ACS 默认参数和解空间搜索技术,PSACS 的性能便已相当可观。此结论表明蚁群优化方法适合求解异构 DAG 调度问题。

参 考 文 献

[1] UllMan J D. NP-complete scheduling problems[J]. J. of Computer and Systems Sciences, 1975, 10: 384-393

[2] Topcuoglu H, Hariri S, Wu Min-you. Performance effective and low-complexity task scheduling for heterogeneous computing [J]. IEEE Transactions on Parallel and Distributed Systems, 2002, 13(3): 260-274

[3] Bajaj R, Agrawal D P. Improving Scheduling of Tasks in A Heterogeneous Environment[J]. IEEE Transactions on Parallel and Distributed Systems, 2004, 15(2): 107-118

[4] Cirou B, Jeannot E. Triplet: A clustering scheduling algorithm for heterogeneous systems[C]// International Conference on Parallel Processing Workshop. 2001: 231-236

[5] Hou E S, Ansari N, Ren H. A Genetic Algorithm for Multiprocessor Scheduling[J]. IEEE Transactions on Parallel and Distributed Systems, 1994, 5(2): 113-120

[6] Correa R C, Ferreira A, Rebreyend P. Scheduling Multiprocessor Tasks with Genetic Algorithms[J]. IEEE Transactions on Parallel and Distributed Systems, 1999, 10(8): 825-837

[7] Dhodhi M K, Ahmad I, Yatama A, et al. An Integrated Technique for Task Matching and Scheduling onto Distributed Heterogeneous Computing Systems[J]. Parallel and Distributed Computing, 2002, 62(9): 1338-1361

[8] Dorigo M. Optimization, Learning and Natural Algorithms[D]. DEI, Polytecnico di Milano, Milan, Italy(in Italian), 1992

[9] Marco D, Mauro B, Thomas S. Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique[J]. IEEE Computational Intelligence Magazine, 2006(11): 28-39

[10] Chen Wei-neng, Zhang Jun. An Ant Colony Optimization Approach to a Grid Workflow Scheduling Problem with Various QoS Requirements[J]. IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews, 2009, 39(1): 29-43

[11] Udo H, Wolfram S. A Comprehensive Test Bench for the Evaluation of Scheduling Heuristics[C]// Proc. of the 16th International Conference on Parallel and Distributed Computing and Systems (PDCS'04). Cambridge, USA: ACTA Press, 2004: 840-845

[12] Xiao H K, Jun S, Wen B X. Permutation-based particle swarm algorithm for tasks scheduling in heterogeneous systems with communication delays[J]. International Journal of Computational Intelligence Research, 2008, 4(1): 61-70

(上接第 174 页)

[14] Gupta P, Doermann D, DeMenthon D. Beam search for feature selection in automatic SVM defect classification[C]// the 16th International Conference on Pattern Recognition. 2002: 212-215

[15] Guyon I, Weston J, Barnhill S, et al. Gene selection for cancer classification using support vector machines[J]. Machine Learning, 2002, 46: 389-422

[16] Hedenfalk I, Duggan D, Chen Y, et al. Gene-expression profiles in hereditary breast cancer[J]. N Engl J Med, 2001, 344: 539-548

[17] Alon U, Barkai N, Notterman D A. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays[J]. Proc Natl Acad Sci USA, 1999, 96: 6745-6750

[18] Alizadeh A, Eisen M B, Davis R E, et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling [J]. Nature, 2000, 403: 503-511

[19] van't Veer L J, Dai H, van de Vijver M J, et al. Gene expression profiling predicts clinical outcome of breast cancer[J]. Nature, 2002, 415(6871): 530-536

[20] Tourassi G D, Frederick E D, Markey M K, et al. Application of the mutual information criterion for feature selection in computer-aided diagnosis[J]. Med Phys, 2001, 28: 2394-2402

[21] Weston J, Mukherjee S, Chapelle O, et al. Feature selection for SVMs[M]. Advances in Neural Information Processing Systems. vol. 13. Cambridge, Massachusetts: MIT Press, 2000