网格原子化作业的二分图调度方法

李建勋[1] 郭建华[1] 李维乾[2] 曹茂生[1]
(西安理工大学经济与管理学院 西安 710048)[1] (西安工程大学计算机科学学院 西安 710048)[2]

摘 要 对于网格系统中计算力调度等问题,结合有向无环作业图DATG和无向节点图UNG,采用并行集APS建立了一种基于二分图的网格调度算法BGS,并在惩罚策略、负载均衡、复活机制的引导下,使系统的调度动态地逐步趋向优化。实验结果表明:该算法能够更加适应网格资源的变化,降低作业负载,提高作业的并行化程度,并能根据系统负载合理地利用节点资源。

关键词 网格,原子操作,网格调度
中图法分类号 TP319 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.3.048

Bipartite Graph Scheduling Method for Grid Task Atomized

LI Jian-xun[1] GUO Jian-hua[1] LI Wei-qian[2] CAO Mao-sheng[1]
(College of Economic and Management, Xi'an University of Technology, Xi'an 710048, China)[1]
(Computer Science School, Xi'an Polytechnic University, Xi'an 710048, China)[2]

Abstract Focusing on the scheduling problems in grid system, adopting directed acyclic task graph and undirected node graph, the paper proposed a grid scheduling algorithm based on bipartite graph (BGS) by using atom parallel set, and made the dynamic grid scheduling gradually optimized by introducing punishment strategies, load balancing and revived mechanism. The results show that the BGS algorithm can adapt to the changes in grid resources better, reduce the load of tasks, improve the degree of parallel operations and reasonably use the node resources in accordance with system load.

Keywords Grid, Atom operation, Grid scheduling

1 引言

当前,随着网格应用的逐步开展,人们迫切地需要更加灵活、高效的调度算法对复杂作业的执行予以支撑,实现作业内部逻辑的并行化,提供更细粒度的调度方案,并合理地调度网格节点资源为系统化应用服务。为了实现网格上的作业调度与实际应用相结合,文献[1]采用服务质量为工作流执行过程中成员服务的选择提供依据,使得网格中的资源更好地围绕用户的要求进行组织和分配;Robertazzi和Bharadwaj[2]则在计算和通信都不存在启动开销的假设下,提出了基于线性代价模型的可分任务理论;文献[3]研究了异构网络的可分任务调度问题,并指出优化作业调度应按照网络带宽从高到低的顺序执行;Beaumont[4]按照独立任务调度的思想给出了同构任务异构平台的优化调度算法。但此类算法主要适应于业务逻辑简单、计算耗时小的作业调度,还不能够满足复杂业务应用的需要,也未能较好解决网络通信、作业流程对调度算法的影响问题。为消除通信开销的影响,Rashmi[5]采用了复制技术,在较低的复杂度下得到问题的满意解,其需要较大的存储空间;Radulescu[6,7]在分析列表调度算法的复杂性的基础上,针对消息处理和任务选择问题提出了快速算法;文献[8]构造了两种动态的、需求驱动的任务分配启发式算法,讨论了在网格资源计算能力和网络通信速度异构的树型计算网格环境下独立任务的调度问题。鉴于网格和图之间的紧密关系,文献[9]考虑了任务之间的约束关系,提出了一个异构调度算法DLS,在有效利用计算力资源的基础上实现作业的优化调度;杜晓丽等[10]针对网格环境中任务调度规模庞大、分布异构和动态性等特点,提出一种基于模糊聚类的网格异构任务调度算法;金海等[11]还将DAG划分为若干个独立任务集合,然后采用Min-min等算法对各个独立任务集进行调度。上述算法虽然提高了作业执行速度,然而还存在一些冗余迭代、慢启动等问题,并且大多以元作业为研究对象,并未能实现作业的细粒度拆分,也未考虑节点资源能力的变化。另外,文献[12]给出了一个动态的关键路径调度算法DCP,但其将关键路径的计算看作是调度的核心,缺乏对作业整体性能和各节点计算性能的考虑,具有一定的片面性。

多年来的分析与探索业已证明:网格环境中的作业调度,

到稿日期:2014-04-17 返修日期:2014-07-07 本文受"十二五"国家水体污染控制与治理重大专项课题:基于水环境风险防控的松花江水文过程调控技术及示范(2012ZX07201-006),国家自然科学基金:基于数字地球和复杂性理论的水污染模拟仿真研究(51109177),陕西省自然科学基础研究计划项目:3S下的水利工程移民辅助决策技术研究与实现(2014JM9365)资助。

李建勋(1977-),男,博士,副教授,主要研究方向为决策支持系统,E-mail:jxli@xaut.edu.cn;郭建华(1972-),男,硕士,讲师,主要研究方向为决策支持系统,E-mail:jhguo@xaut.edu.cn;李维乾(1980-),男,博士,讲师,主要研究方向为软件工程,E-mail:wqli@foxmail.com;曹茂生(1968-),男,硕士,讲师,主要研究方向为决策支持系统,E-mail:maoshengcao@xaut.edu.cn。

是结合当前的负载和资源情况，动态地安排应用程序运行的策略，是面向异构平台和节点内部自治的优化服务。它要求调度活动在不干涉网格节点内部的任务执行的同时能够适应节点资源或结构的动态变化。然而现有算法均没有完全地反映节点自治任务和计算力资源变化等因素，调度算法的适用性不高，也难于满足气象、水利、医药等行业复杂应用的需要。为此，本文首先讨论了作业的关键路径，通过动态作业关键路径来控制调度的优化方向；然后结合 DATG(Directed Acyclic Task Graph)和 UNG(Undirected Node Graph)，提出了一个 APS 上的二分图调度算法 BGS(Bipartite Graph Scheduling)；在信任模式下将调度转化为二分图的联接操作，并及时地响应节点属性变化，提高了调度算法的实用性、灵活性和高效性。

2 作业关键路径

在项目管理中，关键路径是指网络终端元素的序列，该序列具有最长的总工期并决定了整个项目的最短完成时间。网格中的作业与项目类似，在将网格作业拆分为原子操作后[13]，由于每个原子具有独立的运行机制、数据结构体、输入输出接口和消息传递方法，且通过数据流连接构成有向无环作业图 DATG，因此只有作业中最长或耗时最多的路线上的原子完成之后，作业才能结束，这条最长的活动路线称为作业关键路径(CTP：Critical Task Path)，组成关键路径的活动称为关键原子(CA：Critical Atom)。任何一个 CA 的延迟都会导致整个作业的延迟，而缩短非 CTP 上原子的执行时间并不对作业产生任何影响。本文正是在分析原子耗时的基础上，引入关键路径算法来避免作业调度服务的盲目性，调度的核心则是基于信任模型的二分图调度。

网格中原子的整个执行耗时来自于调度耗时、通信耗时、执行耗时 3 部分。各原子的调度耗时差别甚小，由于给所有路径增加等量的耗时并不能够影响关键路径，在计算时可以忽略；通信耗时主要取决于调度命令和数据交互的通信耗时，每个原子的调度命令通信耗时在执行时几乎相等，也可忽略，而原子的执行耗时和数据交互的通信耗时已经包含在 $N_{TimeCost}$ 内。据此，在 CTP 的初始生成过程中就可使用 $N_{TimeCost}$ 来代替原子的总耗时。

设 $ESA(a_i)$ 表示原子 a_i 执行的最早开始时间，$EFA(a_i)$ 表示原子 a_i 执行的最早完成时间，$LSA(a_i)$ 表示原子 a_i 执行的最晚开始时间，$LFT(a_i)$ 表示原子 a_i 执行的最晚完成时间，则 $ESA(a_i)=LSA(a_i)$ 的原子被称为关键原子。以下给出网格中关键路径的算法(为了便于表达在作业图中增加两个原子操作，一个是起始原子操作，一个终止原子操作，该两个原子操作并不产生耗时)：

算法 1 CTP 及 CA 生成算法

输入：DATG

输出：CTP、CA 列表

(1)令 $EFA(a_i)=0$，并初始化一个链表 CAList；

(2)从起始原子 a_0 出发，按拓扑有序序列求其余各原子的可能最早发生时间 EFA，并开始向前逐步递推：$EFA(a_k)=\max\{EFA(a_j)+a_k.N_{TimeCost}\}$，$a_j\in pred(a_k)$；

(3)如果得到的拓扑有序序列中顶点的个数小于网中顶点个数 n，说明网中有环，不能求出关键路径，算法结束；

(4)从完成顶点 a_n 出发，令 $LFT(a_n)=EFA(a_n)$，按逆拓扑有序求其余各顶点的允许的最晚发生时间，开始向后递推：$LFT(a_j)=\min\{LFT(a_k)-a_j.N_{TimeCost}\}$，$a_k\in succ(a_j)$；

(5)求每一原子 $a_i(1\leqslant i\leqslant n)$ 的最早开始时间 $ESA(a_i)=EFA(a_j)$ 和最晚开始时间：$LSA(a_i)=LFT(a_k)-a_j.N_{TimeCost}$；

(6)若某条弧满足 $ESA(a_i)=LSA(a_i)$，则它是 CA，将其依序加入到 CAList 中。最终 CAList 的链路关系即为 CTP，CAList 的每个元素即为 CA。

在作业调度的过程中，由于网格的自治性和异构性，若将彼此间存在通信代价的原子调度到同一节点上或将同一原子调度到不同节点上，都将引起路径长度的改变，并影响 CTP 和 CA 的生成。因此在调度过程中还需要给节点附加记忆功能，在 V_i 中记录节点执行原子时的历史过程中的通信数据总量，以该值作为近似值代替 CTP 计算时的通信量，对 CTP 和 CA 进行动态的调整，从而尽可能动态地、准确地获取作业图的 CTP，并对各 CA 优先安排高效能资源，压缩作业执行所需时间。

3 二分图调度算法

在网格中，作业是一个有向无环图，是原子之间的一种逻辑依赖关系。资源图是一个抽象拓扑图，它在一定程度上抽象地表示各节点之间的连接关系。根据作业图的 APS(Atom Parallel Set，按优先执行顺序 $A_i\prec A_{i+1}$ 建立偏序关系集合，其中集合 A_i 中的任一原子必须在 A_{i+1} 之前执行)，我们可以将作业中的所有原子划分为一系列集合，每个集合内部的原子都相互独立，可以并行地执行。从而调度问题被转化为：如何在信任模型下将 APS 与特定的资源节点相互连接起来(简称为关联)，如图 1 所示，关联的建立次序就是原子被调度的时序。

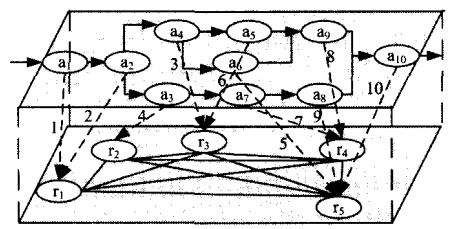


图 1 并行集上的二分图调度

设各节点执行原子操作时在无缓冲池模式下按照线程方式执行，且各计算节点上的线程之间互不干扰，则定义调度的算法框架如下：给定一个并行集 A_i、前驱并行集 $pred(A_i)$ 和关键路径 CTP，首先构造一个被初始化为空集的调度池集合 M；然后从 A_i 中提取一个原子 a，计算 a 的前驱集 $pred(a)$，在前驱并行集 $pred(A_i)$ 中判断 $pred(a)$ 是否执行完成，如果是则将 a 加入到调度池 M 中；其后计算 M 中各节点的信任度，形成满足置信区间的节点集合 H；从 H 中依次提取节点的属性状态对其计算力使用 Q 进行评价，并优先选择 CTP 上的原子以尽量缩短关键原子的执行时间；再后，将信任度超过阈值 θ 的最优节点当作目标节点 r，调度原子 a 到节点 r 上执行(调度时通信协议依据该原子的 $F_{Serialize}$ 和 $F_{Deserialize}$ 建立)，同时连接 DATG 的 a 和无向节点图 UNG 的 r 生成一条边，在边上表示调度花费的时间；最后，根据本次调度后 UNG 节点之间的关系对 CTP 进行修正，把节点之间的通信代价加入到关键路径的计算中。由此，调度过程被简化为一个二分图的逐步建立过程(根据子图 DATG 和 UNG 生成一个二分图 DATG_UNG)，而我们依靠 APS 的偏序结构调配原子到节点进行执行就可得到网格的调度算法。

算法 2　二分图调度算法 BGS

输入：DATG 和 UNG

输出：DATG_UNG

(1)Create APS{$A_1, A_2, \cdots, A_n$}, $A_i \prec A_{i+1}$ from DATG=(A,F,U); SchedulingSet M=Φ;DATG_UNG=null;

(2)copy(DATG) to DATG_UNG;copy(UNG) to DATG_UNG;

(3)fireEvent(DATG. a_0. event. TASK_START);

(4)On (event. ATOM_FINISHED or event. TASK_START) and (eventObj instanceOf Atom) do

(5)　If event. TASK_START then {init(UNG. P);init (UNG. V);}

(6)　If event. ATOM_FINISHED then {update(UNG. P); update (UNG. V);}

(7)　A_i=DATG. getAPS((Atom)eventObj);

(8)　For a in A_i do {If pred(a). finished and a. notstart then M. push (a);}

(9)　If M≠Φ and ! DATG. aborted then

(10)　Compute dynamic CTP by Algorithm 1;

(11)　UNG. refresh();//if any node online or offline

(12)　For a in M do {If a is CA then M. setPos(a,0);}

(13)　For a in M do

(14)　　Create Set H =Φ;

(15)　　For r in UNG do

(16)　　Compute C=[CMin,CMax]=[$B_R^{a,x,y}(T), P_R^{a,x,y}(T)$];

(17)　　If r>CMin and r<CMax and r. Q>θ then H. push(r);

(18)　　End for

(19)　　Sort H by Q;

(20)　　For r in H do

(21)　　If r=max(H) then

(22)　　Connect(DATG_UNG. r, DATG_UNG. a); a. IP=a. U (pred(a). OP);

(23)　　Event=callRPC(r. a. Execute());

(24)　　If Event = ATOM_FINISHED and succ(a) = Φ then {fireEvent(a. event. TASK_FINISHED);return a. OP;}

(25)　　Else If Event=ATOM_FINISHED then {fireEvent(a. event. ATOM_FINISHED);}

(26)　　Else If Event=ATOM_EXECUTE_ERROR or EventFlag =ATOM_EXECUTE_EXPIRE then

(27)　　a. ErrorCount++;

(28)　　If a. ErrorCount>a. $N_{MaxErrorNum}$ then

(29)　　DATG. abort();break;

(30)　　Else

(31)　　fireEvent(pred(a). event. ATOM_FINISHED);// to execute atom "a" repeatedly

(32)　　End if

(33)　　Else

(34)　　sleep(random(10));fireEvent(pred(a). event. ATOM_FINISHED);

(35)　　End if

(36)　　End for

(37)　　End for

(38)　End if

(39)End on

(40)On Timer(Interval as 100ms) do

(41)　If Cost(DATG)>sum(DATG. a. $N_{ExpireTime}$) then {DATG. abort();update(UNG. P);update(UNG. V);}

(42)End On

在调度算法的第(26)－(32)步，给出了原子运行时出错或超时的处理方法。该方法虽然在原子执行失败且未超过 $N_{MaxErrorNum}$ 时能够调度原子再次尝试执行，但在节点加载大量负载或频繁脱离网格时，网格的稳定性急剧下降，导致作业拥塞而无法正常提供服务。在多次实验分析后，本文提出了惩罚策略、负载均衡、复活机制 3 方面的改进措施，并将其加入到调度算法的第(15)－(19)步中。

(1)惩罚策略：扩展 UNG 中的节点属性 V，记录下节点在执行原子操作时的失败次数 n 和超时次数 m，以 $1/2n+1/m$ 作为惩罚系数，在计算力测评时，将惩罚系数乘到评测结果上。

(2)负载均衡：若原子在同一节点上执行多次的耗时相差甚远，则通常该节点用户执行了自治任务，占用了部分计算力资源。为了避免将原子调度到负载较多的节点上，在计算力测评时先获取 V 中的状态数据，并计算历史上执行该节点之间时差的绝对值和，将该值的倒数作为衰减系数累乘到计算力评测函数上。

(3)复活机制：如果节点脱离网格次数频繁或执行原子超时、失败频度过高，节点的信任度将迅速下降，且远小于阈值 θ 而被搁置，形成死节点(DN：Dead Node)。特别是 DN 在性能改良或软故障排除后若依然不能够被网格重新利用，势必引起节点资源逐步减少，导致网格服务因计算力不足而瘫痪。为了确保 DN 修复后的再次利用，本文采用一个复活算子 $revive(r_{DN})$ 对 DN 进行评价，如果 DN 满足复活条件则可将原子调度到该节点上执行。简单的复活算子可以在节点性能恢复后直接将信任度置为 0.5(不确定信任)，而复杂的复活算子则需要根据应用需求并依靠节点的历史失败次数、超时次数、在线时长、下线时长以及执行效率比来确定。限于篇幅，此处不再赘述。

4　实验与分析

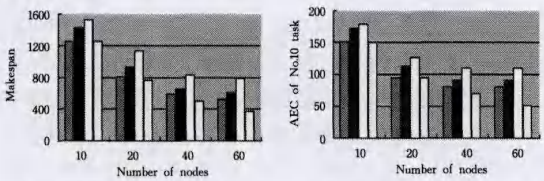
为了验证本文算法的有效性和实用性，我们分别在仿真环境下和复杂应用环境下进行实验。仿真实验采用 SimGrid 建立不同数量级节点的网格环境，实验方案分为基准测试集仿真分析和大规模仿真分析两个类别。基准测试集仿真分析采用基准测试集 PSGs 中的 11 个标准作业[14,15](Ahmad、Al-Maasarani、Ai-Mouhamed、Shirazi、Colin、Gerasoulis、Kruatrachue、McCreary、Chung、Wu、Yang)作为测试作业；大规模仿真分析则首先设计一个随机作业生成器和作业拆分器，然后对不同数量和类型的作业进行实验分析，其中随机作业生成器根据用户自定义的耗时大小、循环个数、分支个数、嵌套程度等参数产生相应的作业，作业拆分器对作业的内部代码进行分析，按照粒度大小、原子参数数量、原子数量等参数将作业拆分成原子。

另外，由于不同的调度算法，所应用的环境和问题假设也不尽相同，因此无法将所有调度算法放置在一种模型下进行评测。鉴于 DLS 算法在异构环境问题方面与本文的问题假设基本相同，DCP 算法则与本文算法较为接近，Min-Min 算法通常作为调度算法评测基准，因此我们将本文算法与 DLS、DCP、Min-Min 算法进行分析比较。

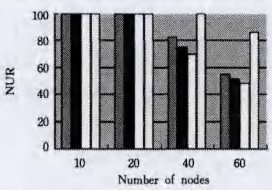
(1)测试集仿真分析

在给定 11 个系列标准作业作为测试基准的情况下，分别

构建不同数量级的网格节点，评测 DCP 、DLS、Min-Min 算法和本文算法 BGS 的 makespan、平均执行耗时(AEC：Average Executing Cost)、节点使用率(NUR：Node Used Rate)。实验在 10、20、40、60 个不同数量的网格节点下，通信/计算比 CCR=1 时，执行 11 个标准系列作业的实验结果如图 2 所示，由图可知，当节点数量逐步增大时 4 种算法的 makespan 和 AEC 均逐步下降，而由于 BGS 采用细粒度的调度思想将标准作业拆分成了原子，因此 makespan 相交更小，当节点数量为 60 个时，BGS 的 makespan 为 DCP 的 69%、DLS 的 59%、Min-Min 的 47%；实验中当节点数量小于 40 个时，4 种算法的 NUR 均可达到 100%，即各个节点均被利用到执行作业任务中，当节点数量超过 40 个时，DCP、DLS 和 Min-Min 算法的 NUR 逐步下降，仅能利用大约 33 个节点，并且其 makespan 和 AEC 变化也不再显著，从而说明此时这 3 种算法已不能够利用剩余的计算力资源，而 BGS 仍能够借助信任度评价实现对计算力资源的有效分析，NUR 依旧保持较高水平，为 87%，并将 makespan 和 AEC 减少 26%左右，这表明 BGS 算法能够更加合理地调度空闲节点，有利于进一步的负载均衡。



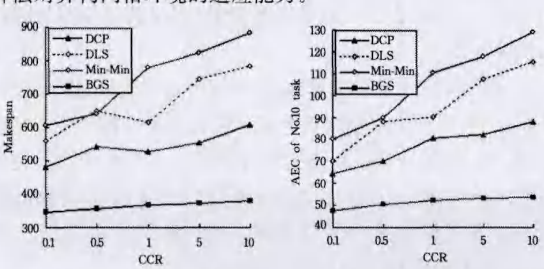
(a)4 种算法的 makespan 比较　(b)4 种算法的 10 号作业 AEC 比较



(c)4 种算法的 NUR 比较

图 2　测试集下不同数量节点时的仿真分析

在节点数量保持 100 个不变而 CCR 不同的情况下，实验结果如图 3 所示，4 种算法的 makespan 和 AEC 随着 CCR 的增长，大致呈增长趋势，DLS、Min-Min 的最大增长率已达到 46%，而 BGS 和 DCP 算法相对其它两种算法增长缓慢，最大增长率分别为 27%和 9%；特别是 CCR 分别为 0.1、0.5、1、5、10 时 BGS 的 makespan 仅为 DLS 的 48%，为 Min-Min 的 42%，BGS 的 AEC 则仅为 DLS 的 46%，为 Min-Min 的 41%。这是由于 BGS 和 DCP 算法能够根据关键路径，在不同的通信和计算力状况下动态地安排关键任务进行执行，从而提高算法对异构网格环境的适应能力。



(a)4 种算法的 makespan 比较　(b)4 种算法的 10 号作业 AEC 比较

图 3　测试集下的不同 CCR 时的仿真分析

(2)大规模仿真分析

为了使仿真环境与真实的网格环境更接近，我们扩大实验规模，将网格节点数量和作业数量扩大到 10^3 数量级。在给定耗时大小为[6,14]分钟、循环个数 15 个、分支个数 15 个、嵌套程度 3 层的参数下生成 1000 个作业，采用 DCP、DLS 和 Min-Min 算法直接对该 1000 个作业进行调度，而 BGS 算法则对拆分后(参数为粒度大小<4 分钟、原子参数数量<5 个、原子数量>8 个)形成的 7260 个原子进行调度。当 CCR=1.0，网格节点分别为 1000、1500、2000 个时，4 种算法的调度结果如图 4 所示，BGS 的 makespan 是 DCP 的 89%～93%，是 DLS 的 68%～80%，是 Min-Min 的 56%～58%，可见本算法在大规模作业任务的需求下，明显要优于 DLS 和 Min-Min 算法，且比 DCP 算法具有更少的作业执行时间。在保持网格节点数量为 1000 个不变，作业生成器参数也随机给定的情况下，让作业数量逐步增大时，各算法的调度结果如图 5 所示，由于作业数量的逐步增加，各算法的 MS 和 AEC 必然呈上升趋势，而 BGS 中基于信任度的计算力评价模型则能够快速地响应节点的负载变化，并结合并行集更加合理地调度原子的执行，因此增加程度十分缓慢，尤其是当作业数量达到 4000 时 BGS 的 makespan 仅为 388，而其它 3 种算法均在 530 以上。

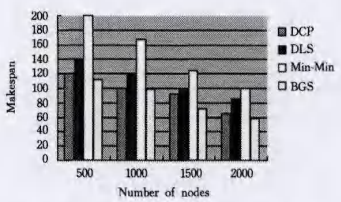


图 4　不同数量节点下的大规模仿真

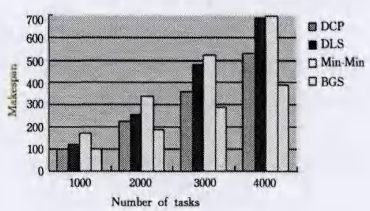


图 5　不同数量作业下的大规模仿真

结束语　本文在探讨现有网格调度算法的基础上，结合 DATG、UNG、APS 和 CTP 的构造，给出了一个具有较强稳定性、适应性、灵活性的二分图作业调度算法。该算法进一步增强了复杂作业的并行化程度，提高了作业执行的效率，并能够更好地与领域内的应用服务需求相结合，为网格环境下的服务组合、工作流、SOA、SaaS 等技术的应用提供了支撑和保障。下一步的工作重点是：(1)建立原子检查点机制，在节点服务终止后根据检查点恢复原子以前的执行状态，降低原子操作执行失败时的重复运行耗时；(2)丰富计算力评测模型和复活机制，并探讨节点数量与加速比的关系，以更加充分地利用网格资源为各领域内的复杂应用服务。

参 考 文 献

[1]　王勇，胡春明，杜宗霞. 服务质量感知的网格工作流调度[J]. 软件学报，2006，17(11)：2341-2351

(下转第 251 页)

网络图的基础上，提出了单点扇形子图的概念。利用对单点扇形子图组局部特征的研究，递归到 $C_{10^m} \times P_{m_{10}}$ 整个图形，进而给出了当 $m \equiv 1,3 \pmod 6$ 时，$C_{10^m} \times P_{m_{10}}$ 边-平衡指数集以及构造方法。当 n 为偶次幂时，本方法比 n 为奇数时标号函数的设计方法有所提高，为其他偶数次幂的无限嵌套图的研究提供了很好的思路借鉴，所证结果为图论和编码理论提供了重要的公式和数据。望近期能够用更有效的方法研究并完成全部幂圈嵌套图的边-平衡指数集，给出一个更加通用的计算公式。

参 考 文 献

[1] Kong M, Lee S M. On edge-balanced graphs[J]. Graph Theory, Combinatoric and Algorithms, 1995, 1: 711-722

[2] Chen B L, Huang K C, Lee S M, et al. On edge-balanced multigraphs[J]. Journal of Combinatorial Mathematics and Combinatorial Computing, 2002, 42: 177-185

[3] Lee A T, Lee S M, Ng H K. On balance index sets of graphs[J]. Journal of combinatorial mathematics and combinatorial computing, 2008, 66: 135-150

[4] Kong M, Lee S M, Ng H K. On friendly index sets of 2-regular graphs[J]. Discrete Mathematics, 2008, 308(23): 5522-5532

[5] Kim S R, Lee S M, Ng H K. On Balancedness of some graph constructions[J]. Journal of Combinatorial Mathematics and Combinatorial Computing, 2008, 66: 3-16

[6] Chopra D, LeeE S M, Su H H. On edge-balance index sets of wheels[J]. International Journal of Contemporary Mathematical Sciences, 2010, 5(53): 2605-2620

[7] Chou C C, Galiardi M, Kong M, et al. On edge-balance index of L-product of cycles with stars, part 1[J]. JCMCC, 2011, 78: 195-211

[8] Lu J, Zheng Y G. On the edge-balance index sets of B(n)[J]. Proceedings of the Jangjeon Mathematical Society, 2009, 12(1): 37-44

[9] Zheng Y G, Lu J, Lee S M, et al. On the perfect index sets of the Chain-Sum Graphs of the first kind of K_4-e[C]// Second International Conference on Intelligent Computation Technology and Automation, 2009. IEEE. Zhangjiajie, China: IEEE Press, 2009: 586-589

[10] Wang Y, Zheng Y G, Adiga C, et al. On the edge-balance index sets of N cycles three nested graph($N=0,1,2 \pmod 6$)[J]. Advanced Studied in Contemporary Mathematics, 2011, 21(1): 85-93

[11] Yao J, Zheng Y G. On the quick construction of all edge-balance index sets of the graph C n× P 5[C]// 2011 International Conference on Consumer Electronics, Communications and Networks (CEC Net). IEEE. Xianning, China: IEEE Press, 2011: 4227-4230

[12] Zheng Y G, YAO J. On the quick construction of all edge-balance index sets of the graph C n× P 11[C]// 2011 International Conference on Consumer Electronics, Communications and Networks (CEC Net). IEEE. Xianning, China: IEEE Press, 2011: 4231-4234

[13] 郑玉歌，姚景景. 无限路等圈嵌套图边-平衡指数集的完全确定(1)[J]. 上海交通大学学报：自然版，2013，47(7)：1160-1163

[14] Zheng Y, Tian H. On the Edge-Balance Index Sets of the Power Circle Nested Graph C2m Pm2 (m 0 (mod 2))[J]. Advanced Science Letters, 2012, 7(1): 534-536

[15] Kang B, Kelarev A, Sale A, et al. A new model for classifying DNA code inspired by neural networks and FSA[M]// Advances in Knowledge Acquisition and Management. Springer Berlin Heidelberg, 2006: 187-198

[16] Kelarev A V. Algorithms for computing parameters of graph-based extensions of BCH codes[J]. Journal of Discrete Algorithms, 2007, 5(3): 553-563

(上接第 236 页)

[2] Bharadwaj V, Ghose D, Robertazzi T G. Divisible Load Theory: A New Paradigm for Load Scheduling in Distributed Systems [J]. Cluster Computing, 2003, 6(1): 7-17

[3] Beaumont O, Legrand A, Robert Y. Scheduling divisible workloads on heterogeneous platforms[J]. Parallel Computing, 2003, 29(9): 1121-1152

[4] Beaumont O, Boudet V, Petitet A, et al. A proposal for a heterogeneous cluster Scalapack (dense linear solvers) [J]. IEEE Transaction on Computers, 2001, 50(10): 1050-1070

[5] 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

[6] Radulescu A, Gemund A J C van. On the complexity of list scheduling algorithms for distributed-memory systems [C]// Proceeding ICS '99 Proceedings of the 13th international conference on Supercomputing, 1999. New York: ACM, 1999: 68-75

[7] Radulescu A, Arjan J C. Low-cost task scheduling for distributed-memory machines [J]. IEEE Transactions on Parallel and Distributed Systems, 2002, 13(6): 648-658

[8] 林伟伟，齐德昱，李拥军，等. 树型网格计算环境下的独立任务调度[J]. 软件学报，2006，17(11)：2352-2361

[9] Sih G C, Lee E A. A compile-time scheduling heuristic for interconnection constrained heterogeneous processor architectures [J]. IEEE Trans. on Parallel and Distributed Systems, 1993, 4(2): 75-87

[10] 杜晓丽，蒋昌俊，徐国荣，等. 一种基于模糊聚类的网格 DAG 任务图调度算法[J]. 软件学报，2006，17(11)：2277-2288

[11] Cao Hai-jun, Jin Hai, Wu Xiao-xin, et al. DAGMap: Efficient scheduling for DAG grid workflow job[C]// Proceedings of the 2008 9th IEEE/ACM International Conference on Grid Computing, 2008. Tsukuba, IEEE, 2008: 17-24

[12] Mourad H, Franck B. Dynamic Critical Path Scheduling Parallel Programs onto MultiProcessors [C]// IEEE Computer Society Proceeding of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), 2005. Washington, 2005: 203-209

[13] 李建勋，解建仓，陈田庆，等. 计算网格下的高性能 MODIS 蒸散发反演研究[J]. 西安交通大学学报，2010，44(10)：36-41

[14] Kwok Y K, Ahmad I. Benchmarking the task graph scheduling algorithms[J]. Journal of Parallel and Distributed Computing, 1999, 59: 381-422

[15] Freund R F, Siegel H J. Heterogeneous processing[J]. IEEE Computer, 1993, 26(6): 13-17