一种改进的可分割任务调度算法 LBMR*)

王 君 李肯立 李仁发
(湖南大学计算机与通信学院 长沙 410082)

摘 要 可分割任务调度在科学和工程计算领域中具有重要的地位,其有效调度算法的设计对并行分布式处理的计算效率至关重要。UMR (Uniform Multi-Round)算法通过限定每次传输到工作节点块的大小,使各工作节点始终处于计算状态,不仅实现了计算资源的最大利用,而且可计算出整个任务调度的最优路数。但是:由于该算法设计中并未考虑网络带宽的有限性,因而难以满足实际计算环境的需求。为此,本文在 UMR 算法中引入网络带宽限制,对该算法在此条件下进行重新设计,提出一种改进的多路可分割任务调度算法 LBMR((limited bandwidth multi-round algorithm)。理论分析和基于 GridSim 的模拟实验结果表明:与 UMR、MI、EMI 等同类调度算法相比,本算法改进了其调度性能,且具有更好的实用性。

关键词 可分割任务,任务调度,单路算法,多路算法,跨度

LBMR : An Advanced Multi-Round Algorithms for Scheduling Divisible Loads

WANG Jun LI Ken-Li LI Ren-Fa
(College of Computer and Communication, Hunan University, Changsha 410082)

Abstract Task schedule problem arises in many fields of science and engineering. It can be parallelized in master-worker fashion and relevant scheduling strategies have been proposed to reduce application makespan. By imposing the restriction that equal sized chunks are sent to workers within a round, the UMR algorithm makes it possible to compute an optimal number of rounds while modeling resource latencies. This paper proposes a new multi-round task schedule algorithm LBMR(limited bandwidth multi-round algorithm). It uses the idea of UMR algorithm, but imports the condition of bandwidth, so it is more actual. Through plenty of simulated experiments by toolkit GridSim, we compare the results with UMR, MI, XMI algorithm, the performance of LBMR is better than pervious algorithms, it can be used widely.

Keywords Divisible task, Task schedule, One round algorithm, Multi-round algorithm, Make-span

1 引言

在众多大规模工程和科学计算的并行与分布式处理应用中,许多的任务可视为独立任务,其有效调度算法的研究对提高并行与分布式计算系统的处理效率具有重要意义,而且由于该问题的调度算法适于机群环境下以 master-worker 模式运行的并行处理,因而近年来备受关注[1~3]。然而,因为任务运行前时间花费的未知性,满足一般分布式系统的高效合理的调度算法的提出还存在实质性困难。这种前期的时间花费包括:(1)将应用程序的输入数据传输给工作节点以及将工作节点的计算结果返回给主机的时间花费;(2)初始化一次计算或通信的时间花费。

可分割任务是指负载可被分割成任意数目的独立子任务。可分割任务调度算法包含两类:单路任务调度算法[4~6]和多路任务调度算法[7,8]。单路指主机进行一次任务分配的工作节点序列。因此单路任务算法指主机将任务通过单路形式分配给工作节点;多路任务调度算法指通过多次的单路调用将任务分配给工作节点。但由于多路任务调度算法分析的复杂性,这方面的研究成果有限。

分割算法的选择是可分割任务的调度算法设计的关键问题。Rosenberg[6]提出了一种可行的单路任务调度算法,该算法将任务分割成与工作节点相同数目的子任务,并将子任务以单路的调度方式分配到工作节点上。该算法存在工作节点的计算与通信的交互性能差等缺陷。为提高工作节点的计算效率,Bharadwaj 等人提出一种多路调度算法 MI(multi-installment algorithm)[10],在调度过程中子任务的大小呈现递增的趋势,以减少调度的时间跨度,虽然算法在给定路数 M 条件下,实现了性能的优化,但该算法没有考虑调度前期的时间消耗,也没有给定求解最优路数 M 的方法,其可用性受到一定的限制。

在实际调度过程中,网络的传输速度和工作节点的计算速度都是动态变化的,因而在预测时存在一定的预测错误,对此问题,Torben Hagerup[11]提出的调度算法要求子任务的大小呈现递减的趋势,这样减少了工作节点性能预测的错误率,使得系统可在允许的预测错误范围内运行。文[14]中 Casanova 在 MI 算法的基础上提出了 XMI 算法(eXtended MI algorithm),与 MI 相比,算法中加入了调度前期的时间消耗,性能有了一定的提高,但是仍然没有提出求解最优路数 M 的方法。为使多路任务调度算法更有效,Yang 和 Casanova 提出了一种新颖的多路任务调度算法 UMR(Uniform Multi-Rounde)[8,9],该算法通过限定子任务的大小,给出了求解最优路数 M 的方法,使得整体的调度性能得到了大幅度的提

*)本文得到国家自然科学基金(60274026,30370356)与教育部重点项目(05128)资助。王 君 硕士生,研究方向为网格计算。李肯立 副教授,博士后,研究领域为并行处理、组合优化。李仁发 教授,博士生导师,研究领域为嵌入式计算,无线通讯与网络,数字媒体。

高，因而该算法和前述算法相比具有明显的优越性。

值得指出的是：尽管 UMR 算法具有较同类算法更好的理论性能，该算法却并未考虑实际网络中可用带宽的限制。显然，这一缺点将影响算法在实际应用过程中的实际调度性能。鉴此，本文提出一种 UMR 算法的改进算法 LBMR。本算法通过引入网络带宽限制，将其置于算法的约束条件中，并在新的约束条件下使用 UMR 算法的设计思想，对调度算法的各步骤进行新的理论分析和重新设计，以期使 LBMR 算法理论上既具有和 UMR 算法类似的优点，又可避免因未考虑实际网络中的可用带宽而带来的不足。将 LBMR 算法、UMR 算法及其它相关算法在 GridSim 上的仿真实验结果表明：本文的改进算法 LBMR 确具有更高的调度性能。

本文的其余部分组织如下：第 2 部分是对 UMR 算法的简单介绍，第 3 部分提出改进的调度算法 LBMR，模拟实验结果及算法性能分析与比较在第 4 部分给出，最后是对本文的总结。

2 UMR 算法

在 UMR 算法中，W 代表负载总数，W 被分割成调度运行所需的子任务 s。其 s_i 表示在工作节点 i 上运行的子任务的大小，$1 \leqslant i \leqslant N$。$s_i$ 在节点 i 上的计算所需时间 Tp_i 为[8]：

$$Tp_i=\alpha+\frac{s_i}{\omega} \tag{1}$$

其中 α 为计算开始前的固定时间消耗，ω 为节点的运算速度（单元数每秒钟），s_i 被传输到节点 i 上所需的时间 Tn_i 为[8]：

$$Tn_i=\beta+\frac{s_i}{c}+d_i \tag{2}$$

β 为主机向工作节点 i 初始化一次传输所消耗的时间，例如初始化 TCP 连接等，c 为主机传输任务到工作节点 i 的速率（单元数每秒），d_i 为主机发送完数据后到节点 i 接收到最后一个数据单元之间的时间间隔。假设在时间 $\beta+s_i/c$ 内，不存在计算与传输的交互，但在时间 d_i 内存在两者的交互。

图 1 显示了 UMR 任务调度算法分配任务的策略，它的思想与 MI 算法[10]相近，UMR 算法在每条单路中保持子任务的大小是固定的，在路与路之间子任务的大小呈现递增的趋势，这样可以减少 α、β 时间消耗，从而减少整个调度的时间跨度。

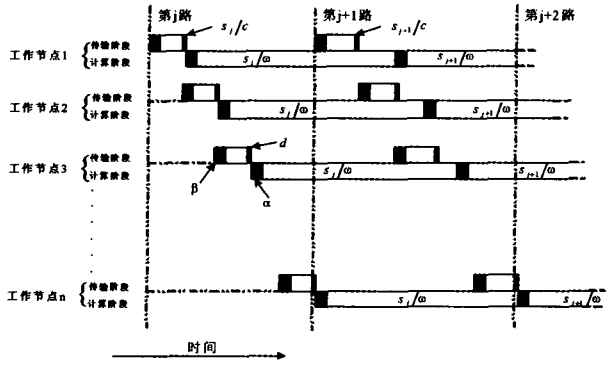

图 1 UMR 算法的任务分配

实验表明[8]：和此前的单路、多路任务调度算法相比，UMR 算法调度性能得到提高，并通过限制单路子任务的大小近似计算出最优路数 M^*。但是该算法假设网络带宽足够大，而实际网络带宽是有限的，并且在数据传输过程中，网络带宽是一个影响算法性能的主要因素。为此本文在 UMR 算法基础上引入网络带宽的限制，提出更加符合实际要求的 LBMR 算法。新的算法将根据网络带宽的实际情况，利用不同的调度策略完成可分割任务的调度，关于算法的详细介绍将在第 3 部分给出。

3 LBMR 算法

LBMR 算法采用星型的网络模型，如图 2 所示。星型网络包括一个中心节点 P_0 和 K 个子节点 $P_1,\cdots,P_k$。P_0 每次只能与一个孩子 $P_i(i=1,\cdots,k)$ 进行通信，为了简化计算，假设各个工作节点都具有相同的计算速度，即 $\omega_1=\omega 2=\cdots=\omega_k$。$P_0$ 不参与任务计算，即 $\omega_0=0$，该节点负责任务的调度。P_i 负责任务的计算，其中 $c_i(i=1,\cdots,k)$ 为主机 P_0 传输任务到节点 i 的速率（单元数每秒），c_{-1} 为 P_0 从父节点 P_{-1} 接收任务的速率（单元数每秒）。

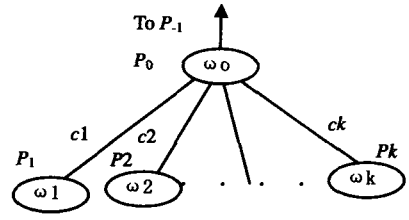

图 2 星型网络模型

LBMR 算法在 UMR 算法基础上引入了网络带宽这一约束条件，算法在执行调度策略之前须先判断网络带宽是否满足数据传输需求，因此将网络按照带宽的情况分为两类：充足带宽和受限带宽。前者可以保证所有的工作节点都参与任务的计算，并且各个节点不存在闲置状态；而后者导致部分的工作节点处于闲置状态，造成了计算资源的浪费。LBMR 为了满足网络实际情况的需要，针对以上两种情况，分别提出了充足带宽多路算法和受限带宽多路算法。

3.1 充足带宽多路算法

在充足带宽网络环境下，网络带宽满足数据传输的需要，所有的工作节点都参与任务计算，但是 UMR 算法假设网络的传输速度是相同的，即 $c_1=c_2=\cdots=c_k$，而在实际计算环境中，网络传输速度 c_k 是不同的，为此充足带宽多路算法引入 c_k 的差异性，关于子任务 s_j、最优路数 M 推导如下：

为了保证计算资源的高利用率，首先必须满足条件：第 j 条单路节点计算所用的时间等于第 $j+1$ 条单路 P_0 向所有工作节点分配任务所需时间，即：

$$\alpha+\frac{s_j}{\omega}=\frac{s_{j+1}}{c_1}+\beta_1+\frac{s_{j+1}}{c_2}+\beta_2+\cdots+\frac{s_{j+1}}{c_k}+\beta_k \tag{3}$$

为了便于 s_j 的推导，我们简化公式(3)，得到如下公式：

$$\alpha+\frac{s_j}{\omega}=s_{j+1}\times c+\beta \tag{4}$$

其中 $c=\frac{1}{c_1}+\frac{1}{c_2}+\cdots+\frac{1}{c_k}$，$\beta=\beta_1+\beta_2+\cdots+\beta_k$，由公式(4)推导出 s_j 计算公式：

$$\begin{cases} s_j=\left(\frac{1}{c\omega}\right)^j(s_0-\Delta)+\Delta, c\omega\neq 1 \text{ 其中 } \Delta=\frac{\omega(\alpha-\beta)}{c\omega-1} \\ s_j=s_0+j\omega(\alpha-\beta), c\omega=1 \end{cases} \tag{5}$$

利用 UMR 算法求解 s_0 的方法，计算最优路数 M，计算公式如下：

$$\begin{cases} K\Delta-\dfrac{W-KM\Delta}{1-\left(\frac{1}{c\omega}\right)^{M}}\times\left(\frac{1}{c\omega}\right)^{M}\ln\frac{1}{c\omega}-2\times\alpha\times\frac{K}{c}\times\dfrac{\left(\frac{1}{c\omega}\right)^{M}}{1-\frac{1}{c\omega}}=0, c\omega\neq 1 \\ M=\sqrt{\dfrac{2cW}{3\alpha+\beta}}, c\omega=1 \end{cases} \tag{6}$$

通过公式(6)将求解出 M 的值，但由于求解的复杂性，将 M 的近似值记为 M^*，s_0 初始值为：

$$\begin{cases} s_0=\dfrac{\left(1-\frac{1}{c\omega}\right)(W-KM^*\Delta)}{K\times\left(1-\left(\frac{1}{c\omega}\right)^{M^*}\right)}+\Delta, c\omega\neq 1 \\ s_0=c\times(2M^*\alpha-(M^*-\frac{1}{2})(\alpha-\beta)), c\omega=1 \end{cases} \tag{7}$$

通过以上理论分析，得到下述充足带宽多路算法描述：

算法 1 充足带宽多路算法（Abounded bandwidth multi-round algorithm）

```
begin
    using (6)calculate M*
    for 0≤j≤M*
    using (5)and (7)calculate s_j
        for all P_i where 1≤i≤k do
            receive s_j from P_0
            execute task s_j
        end for
    end for
end
```

3.2 受限带宽多路算法

在受限带宽的网络环境下，部分工作节点必然处在闲置状态，没有参与任务的计算。为了精确描述整个星型网络的计算速率，引入参数 R、ω_{tree}，其中 R 单位时间内网络执行的单元数，则 $\omega_{tree}=1/R$。提高网络性能的问题就转换最小化 ω_{tree} 的优化问题。

为取得工作节点的最大利用率，首先按照递减的顺序对 c_i 进行排序，p 为参加计算工作节点的最大下标，则 $\sum_{i=1}^{p}\frac{c_i}{\omega}\leqslant 1$ 并且 $\sum_{i=1}^{p+1}\frac{c_i}{\omega}>1$。时间 $T=lcm(\omega,c_{-1},c_{p+1})$（$lcm$ 为最小公倍数函数）为各工作节点进入稳定计算状态时间，即：从 0 到 $T-1$ 时间内，各工作节点始终处于接收任务状态。算法描述如下[12]：

算法 2 受限带宽多路算法（Scarce bandwidth multi-round algorithm）

```
begin
    sort c_i(1≤i≤k), that c_1≥c_2≥…≥c_k
    if p<k
        then ε=1-Σ_{i=1}^{p} c_i/ω
else ε=0
    for P_i where 1≤j≤p+1 do
        receive task from P_0
        execute task on P_i
    end for
end
```

通过算法 2 的调度分配，星型网络的计算速率为 $\omega_{tree}=\max\left(\frac{1}{c-1},\frac{1}{p\omega+c_{p+1}\times\varepsilon}\right)$，保证了参与运行计算的节点数为 $p+1$，即 $P_1,\cdots,P_p$ 节点在调度任务结束前始终处于计算状态，节点 P_{p+1} 存在部分的闲置，在带宽不足的情况下保证了计算资源的高利用率，并且 $k-(p+1)$ 个节点不参与该负载的计算，可参与其他负载的计算，保证了高效的计算稳定性。

3.3 LBMR 算法描述

LBMR 算法根据网络带宽的实际情况，针对充足带宽和受限带宽两种情况，分别调用充足带宽多路算法与受限带宽多路算法，算法描述如下：

算法 3 有限带宽多路算法（limited bandwidth muti-round algorithm）

```
begin
  sort c_i(1≤i≤k), that c_1>c_2≥…≥c_k
  if c_{-1}≥c_1
     then Perform Abounded bandwidth multi-round algorithm
  else Perform Scarce bandwidth multi-round algorithm
end
```

LBMR 算法在带宽满足数据传输的需要时，执行充足带宽多路算法，所有的工作节点都一直处于计算的稳定状态，实现了调度算法时间跨度的最优化；在带宽不能满足数据传输的需要时，此时存在工作节点处于闲置状态，则运行受限带宽多路算法，其中高性能的 $p+1$ 个计算节点将参与任务的计算，保证了计算资源的高效利用率。

4 模拟实验

本文使用 GridSim[13] 进行模拟实验，GridSim 是在 SimJava[15] 的基础上开发的，它提供丰富的函数库以支持模拟网格环境中的异构资源（时间共享和空间共享）、用户、应用程序、用户代理和调度器，可以快速地搭建模拟实验平台。算法结果与同类算法 UMR[8,9]、MI[10]、XMI[14] 进行性能对比，但由于 MI、XMI 算法不像 UMR 算法那样可以求解出最优路数 M，因此须对 MI、XMI 算法的循环路数进行赋值，给定 MI-x，XMI-x(x=1,…,3)。

实验所需的参数见表 1。表中“单元”为计算、传输的最小记数，即计算、传输的速率都为“单元”的整数倍。为了简化模拟的过程，假设 $\omega=1$。

表 1 实验参数值

参数	值
处理器的数目	N=10,15,20,…,50
总负载(单元总数)	$W_{total}=1000$
计算速率(单元数/秒)	$\omega=1$
传输速率(单元数/秒)	$c_i=(1,2,1.3,\cdots,2)\times N$
计算延时(秒)	α_i=0.0,0.1,…,1
传输延时(秒)	β_i=0.0,0.1,…,1

为了体现实验结果的整体效果，表 2 显示了在所有的模拟实验中，LBMR 算法的性能优于与其相比较的算法所占的比例，这些数据结果显示，LBMR 算法在至少 60%的实验中性能是明显高于其余算法的，甚至达到了 90%。为了进一步量化的表示 LBMR 算法与其他三类算法性能上的差异，将给出表 3，表 3 中的数据显示了在所进行的模拟实验中 LBMR 算法在性能上优于其他算法 15%的实验所占的比例。数据显示，在该种条件下，LBMR 算法至少 50%到 80%的实验性能优于对比算法。

表 2 LBMR 算法性能优于其他算法的实验占总实验数的比例

算法	UMR	MI-1	MI-2	MI-3	XMI-1	XMI-2	XMI-3
比例数	68.59	82.56	89.73	91.86	78.64	86.49	90.27

表 3 LBMR 算法性能优于其他算法 15%的实验占总实验数的比例

算法	UMR	MI-1	MI-2	MI-3	XMI-1	XMI-2	XMI-3
比例数	53.17	62.85	74.34	81.06	58.47	67.08	77.87

由表2、表3数据分析可得：LBMR算法与UMR、MI、XMI同类算法相比，60%以上实验结果性能得到明显提高。同UMR算法相比，LBMR算法可根据网络带宽实际情况，选择不同的多路调度算法来完成任务分配、计算，更加符合实际计算环境需求；与MI、XMI算法相比，除算法的自适应性更强以外，LBMR算法可以近似求解调度所需的最优路数M，而MI、XMI算法只能人为指定循环路数大小，任务调度性能不能得到保证。

结 论 UMR算法是一种高效可分割任务调度算法，该算法实现了任务调度时间跨度最优化，给定了求解最优路数M的方法。由于实际网络带宽的有限性，大大限制了UMR算法的应用范围，本文在分析UMR算法基础上，引入网络带宽约束条件，提出了适用于有限网络带宽的可分割任务调度算法LBMR。该算法将结点的处理能力和网络流情况协同考虑，分别针对充足带宽和受限带宽提出相应调度算法，相比单纯的充足带宽多路算法和受限带宽多路算法，本文所提算法避免了纯充足带宽多路算法在处理任务时不能满足可适应性问题，又克服了纯受限带宽多路算法在任务处理过程中工作节点利用率低等问题的缺点。实验表明LBMR算法较先前提出的可分割任务调度算法，调度性能明显提高，该算法易于调度性强，具有更高的应用价值。

参 考 文 献

1 Davis T, Chalmers A, Jensen H W. Practical parallel processing for realistic rendering. ACM SIGGRAPH, 2000
2 Lee C, Hamdi M. Parallel Image Processing Applications on a Network of Workstations. Parallel Computing, 1995, 21: 137～160
3 Altilar D, Paker Y. An Optimal Scheduling Algorithm for Parallel Video Processing. In: Proceedings of the IEEE International Conference on Multimedia Computing and Systems, 1998
4 Blazewicz J, Drozdowski M, Markiewicz M. Divisible Task Scheduling - Concept and Verification. Parallel Computing, 1999, 25: 87～98
5 Drozdowski M, Wolniewicz P. Experiments with Scheduling Divisible Tasks in Clusters of Workstations. In: Proceedings of Europar'2000, 2000. 311～319
6 Rosenberg A L. Sharing Partitionable Workloads in Heterogeneous NOWs: Greedier Is Not Better. In: Proceedings of the 3rd IEEE International Conference on Cluster Computing, 2001. 124～131
7 Hummel S F. Factoring: a Method for Scheduling Parallel Loops. Communications of the ACM, 1992, 35(8): 90～101
8 Yang Y, Casanova H. UMR: A Multi-Round Algorithm for Scheduling Divisible Workloads. In: Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03), 2003
9 Yang Y, Casanova H. Multi-Round Algorithm for Scheduling Divisible Workload Applications: Analysis and Experimental Evaluation: [Technical Report CS2002-0721]. Dept. of Computer Science and Engineering, 2002
10 Bharadwaj V, Ghose D, Mani V, Robertazzi T G. Scheduling Divisible Loads in Parallel and Distributed Systems, chapter 10. IEEE Computer Society Press, 1996
11 Hagerup T. Allocating Independent Tasks to Parallel Processors: An Experimental Study. Journal of Parallel and Distributed Computing, 1997, 47: 185～197
12 Beaumont O, Carter L, Ferrante J, Legrand A, Robert Y. Bandwidth-centric allocation of independent tasks on heterogeneous platforms. In: Proc. Int'l Parallel and Distributed Processing Symp., 2002
13 Buyya l R, Murshed M. GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing. Concerrency and computation: Practice and experience, 2002, 14: 1175～1220
14 Yang Y, Casanova H. Extensions to the Multi-Installment Algorithm: Affine Costs and Output Data Transfers: [Technical Report CS2003-0754], 2003
15 Howell F, McNab R. SimJava: A Discrete Event Simulation Package For Java With Applications In Computer Systems Modelling. In: First International Conference on Web-based Modelling and Simulation, Society for Computer Simulation, 1998

(上接第273页)

综上可知，对一个Min-CVCB问题实例，如果存在一个受约束的最小点覆盖，则通过AACI算法一定可在$O(2^{2/\epsilon})$时间内找到一个近似率为$1+\epsilon$的受约束点覆盖。

显然，AACI算法满足前面定义4，故它是Min-CVCB问题的一个运行时间为$O(2^{2/\epsilon})$的PTAS算法。

结论 本文主要是研究在VLSI芯片生产中有着广泛应用的Min-CVCB问题。由于传统的启发式算法和精确算法都不能满足实际应用的需要，本文从一个全新角度出发，提出了一个近似率为$1+\epsilon$的近似算法。它首先运用参数计算理论中降核心阶技术处理问题实例，然后利用二分图的经典匹配理论进一步深入分析二分图结构，最后充分利用链暗示技术通过穷举寻找到受约束的近似点覆盖。具体为：当Min-CVCB问题实例存在受约束最小点覆盖时，对于任意给定的$\delta=1+\epsilon>1$，一定可以在$O(2^{2/\epsilon})$时间内找到一个近似率为$1+\epsilon$的受约束近似点覆盖。同时，它也是Min-CVCB问题的一个运行时间复杂度为$O(2^{2/\epsilon})$的PTAS算法。显然，当近似率$1+\epsilon$大小一定时，随着问题规模增大，AACI算法的时间复杂度不变；而当问题实例的规模一定时，随着近似率$1+\epsilon$的减小，AACI算法的时间复杂度越高。

由AACI-DAG D算法中的step2中case1和case3中判定过程可知：DAG D中每一个块都是仅取其U-部点或L-部点之一到受约束近似点覆盖中，故最后找到的近似点覆盖也是一个最小点覆盖，即$k_u^*+k_l^*=k_u+k_l$。它与原来受约束最小点覆盖(k_u,k_l)的区别是：(k_u^*,k_l^*)的取值范围比(k_u,k_l)要大，其中$k_u^*\leqslant(1+\epsilon)k_u$，$k_u^*\leqslant(1+\epsilon)k_u$，但时间复杂度却由指数时间降为多项式时间。目前，我们正在通过进一步深入运用本文中AACI算法，来处理Min-CVCB问题实例不存在受约束最小点覆盖时，如何快速寻找到Min-CVCB问题受约束近似解。

参 考 文 献

1 Downey R G, Fellows M R. Parameterized Complexity. Springer, New York, 1999
2 Kuo S-Y, Fuchs W. Efficient spare allocation for reconfigurable arrays. IEEE Design and Test, 1987, 4: 24～31
3 Hasan N, Liu C L. Minimum Fault Coverage in Reconfigurable Arrays. In: Proceedings of the 18th International Sysmposium on Fault-Tolerant Computing (FTCS'88), IEEE Computer Society Press, Los Alamitos, CA, 1988. 348～353
4 Chen J, Kanj I A. Constrained minimum vertex cover in bipartite graphs: complexity and parameterized algorithms. Journal of Computer and System Science, 2003, 67: 833～847
5 Fernau H, Niedermeier R. An efficient exact algorithm for constraint bipartite vertex cover. Algorithms, 2001, 38: 374～410
6 Lovasz L, Plummer MD. Matching Theory. Annals of Discrete Mathematics, Vol 29. North-Holland, Amsterdam, 1986
7 Blough D M. On the reconfigurable of memory arrays containing clustered faults. In Fault Tolerant Computing, pages, Los Alamitos, Ca., USA, IEEE Computer Society Press, June 1991. 444～451
8 Blough D M, Pelc A. Complexity of fault diagnosis in comparison models. IEEE Trans Comput, 1992, 41(3): 318～323
9 Hasan N, Liu C L. Fault covers in reconfigurable PLAs [A]. In: 20th Int Symp on Fault-Tloerant Computing (FTCS'90), IEEE Computer society Press, 1990. 166～173
10 Low C P, Leong H W. A new class of efficient algorithms for reconfiguration of memery arrays. IEEE Trans Comput, 1996, 45 (5): 614～618
11 Smith M D, Mazumder P. Generation of minimal vertex cover for row/column allocation in self-repairable arrays [J]. IEEE Trans Comput, 1996, 45: 109～115
12 Cormen T H, Leiserson C E, Rovest R L. Introduction to Algorithms. New York: McGraw-Hill Book Company, 1992