维普资讯 http://www.cqvip.com

异构计算中一种图的非均衡划分算法*)

沈轶炜　曾国荪
(同济大学计算机科学技术系　上海 200092)
(国家高性能计算机工程技术中心同济分中心　上海 200092)

摘　要　现有的图的划分算法大多是均衡划分，要求划分块的权值相等，划分块之间的连接代价尽量最小。但是在异构计算环境中，不同的处理机的计算能力不尽相同，从而在并行任务调度时所分配的计算任务量也应随之不同。所以为了适应更广泛意义上的异构负载均衡，本文提出了异构计算中的一种任务图的非均衡划分算法。该算法根据任意给定的需求，使得划分好的各个子集权值不均等。其中划分子集的个数等于异构环境中处理机的个数，各子集的大小比例于不同处理机的计算能力。算法包括3步：粗化阶段、非均衡划分阶段以及精化还原阶段。本文通过用格林威治大学提供的系列开放图来测试该算法，实验结果表明算法是准确有效的。

关键词　异构计算，非均衡的图划分，任务图，分布向量

An Unbalanced Partitioning Scheme for Graphs in Heterogeneous Computing

SHEN Yi-Wei　ZENG Guo-Sun
(Department of Computer Science and Technology, Tongji University, Shanghai 200092)
(Tongji Branch, National Engineering & Technology Center of High Performance Computer, Shanghai 200092)

Abstract　Most existing graph partitioning algorithms produce good equivalent partitions. It means that the partitioned subsets have equal number of vertexes, and meanwhile, the edge-cuts are minimal. However, in the heterogeneous computing environment, the computing power of different processors or workstations is variant, so that the size of the tasks scheduled on them should not be the same as well. In order to meet the need of the load balance for heterogeneous computing, this paper presents a novel algorithm that partitions the original task graph into unbalanced subsets according to the arbitrarily given conditions. In usual case, the number of the partitions is equal to that of the processors and the size of each partition is set according to the computing power. The algorithm is consisted of three phases. Coarsen the original graph, then partition the coarsest graph and last project it back to the original graph and conduct refinement. We test the algorithm using Greenwich Graph Partitioning Archive and get good experiment results.

Keywords　Heterogeneous computing, Unbalanced partitioning, Task graph, Distribution vector

1　引言

超大规模集成电路设计、并行任务调度、城市建设规划等科学和工程问题归根到底是图的划分问题。图的划分问题是一种NP完全问题，从划分的节点子集大小或权值来讲，可分为均衡划分和非均衡划分两类。现有的研究大多集中在均衡划分方面，人们根据启发式规则提出了一系列算法解决了这个问题。例如谱图划分算法通过计算特征向量[5]，几何划分算法利用图的几何信息[6]，它们都能得到优良的划分结果。还有一种多级划分算法，它通过合并节点与边从而得到一个点边集合规模较小的图，在这较小的图上进行划分，最后还原到初始图，同样也能快速得到划分结果[3,4]。这些算法的共同点是强调划分好的子集大小或权值均衡。目前对于图的非均衡划分的研究不多，非均衡划分也不能简单地由均衡划分扩展而成。在许多应用领域中，图的划分块大小需要按不同需求进行分配，这也就需要非均衡划分技术的支持。特别在并行任务调度中，由于在异构环境中[1,2]处理机或工作站的处理能力各不相同，所以任务的粒度大小也应随之有所差异，从而保障在异构环境下的任务负载均衡，同时要满足各个任务间通信量尽可能均衡。这需要解决异构计算中的图的非均衡划分问题。

正是基于上述原因，按照应用需求，研究一种把图划分成任意给定数目、任意给定不同大小的节点子集，同时各子集相连代价也尽可能小的算法是十分必要的。下面将具体介绍基于多级划分算法提出的图的非均衡划分算法。

2　图的非均衡划分的问题描述

定义1(处理机的计算能力)　在异构环境中，单台处理机的计算能力CP(Computing Power)是一个综合量，包括I/O、内存读写、处理器、网络通信开销等。所以，一个处理机的计算能力可定义为：

$$CP=\frac{\frac{f_{i/o}}{t_{i/o}}+\frac{f_{r/w}}{t_{r/w}}+\frac{f_{comp}}{t_{comp}}+\frac{f_{comm}}{t_{comm}}}{f_{i/o}+f_{r/w}+f_{comp}+f_{comm}}$$

其中 $f_{i/o}$，$f_{r/w}$，f_{comp}，f_{comm} 分别为I/O、内存读写、计算、通信操作的频率。$\bar{t}_{i/o}$，$\bar{t}_{r/w}$，$\bar{t}_{comp}$，$\bar{t}_{comm}$ 为上述各类操作类型的平均执行时间，可以由benchmark程序得到。

*)国家自然科学基金资助项目(60173026)、上海科委重大项目(03DZ15029)、上海高校网络技术E-研究院资助(200301-1)。沈轶炜　硕士生，软件理论专业；曾国荪　博士、教授、博导，主要研究领域为异构计算，计算网络。

维普资讯 http://www.cqvip.com

定义 2(分布向量) 非均衡的分布向量定义为 $A=(\alpha_1,\alpha_2,\cdots,\alpha_k)$,其中 k 为对象的个数,且 $\sum_{i=1}^{k}\alpha_i=1$。特别当 A 为异构任务分布向量时,$\alpha_i=\frac{CP_i}{\sum_{j=1}^{k}CP_j}$,$CP_i$ 为第 i 台处理机的计算能力。而当 A 为图的划分分布向量时,α_i 为划分子集大小的比例。

定义 3(并行任务图) 并行任务图可定义为 DAG $=(V,E)$,其中 V 为任务节点,E 为连接任务节点的通路。任务节点的权值可视为计算量,通路的权值可视为通信代价。一般来说,控制流图 CFG 结合数据流图 DFG,可以构造并行任务图[2]。

定义 4(图的均衡划分的定义) 给定一个 $G=(V,E)$,V 为节点集合,E 为边集合,$|V|=n$。把 V 划分成 k 个子集 $V_1,V_2,\cdots,V_k$,$V_i\cap V_j=\phi,i\neq j,|V_i|=n/k,\bigcup_i V_i=V$,并且把连接不同子集的边的数量减至最小。

定义 5(图的非均衡划分的定义) 给定一个 $G=(V,E)$,分布向量 $A=(\alpha_1,\alpha_2,\cdots,\alpha_k)$,$|V|=n$,把 V 划分成 k 个子集 $V_1,V_2,\cdots,V_k$,$V_i\cap V_j=\phi,i\neq j,|V_1|:|V_2|:\cdots:|V_k|=\alpha_1:\alpha_2:\cdots:\alpha_k$,$\bigcup_i V_i=V$,并且把连接不同子集的边的数量减至最小。

比较二者的定义,其中最大区别在于:均衡划分中,$|V_i|=n/k$,即 $|V_i|:|V_j|=1:1$;而非均衡划分中,$|V_i|:|V_j|=\alpha_i:\alpha_j$。

本文用大小为 n 的一维数组 P 来表示节点的划分。P 实际上是一个节点到划分块的映射函数。$\forall v\in V,P[v]=p$,$p\in\{0,\cdots,k-1\}$。

定义 6(连接代价) 给定一个 P,P 的连接代价 EC_p(edge-cut)$=\sum_{P[v]\neq P[u]}w(v,u)$,其中 $v,u\in V,(v,u)\in E,w(v,u)$ 为边 (v,u) 的权值。

至此,本文给出了与所有非均衡划分有关的定义。给定一个并行任务图,需要在异构环境下进行调度运行。通过计算参与调度的处理机的计算能力,可以得到一个分布向量,也就是划分的分布向量。从而按照这个划分分布向量对并行任务图进行非均衡划分。下面具体介绍图的非均衡划分算法。

3 图的非均衡划分算法

图的非均衡算法的基本思想主要包括以下 3 步:粗化阶段,图 G_0 首先通过合并节点粗化为节点数较少的图 G_1,G_2,$\cdots,G_m$;非均衡划分阶段,在粗化好的图 G_m 上进行划分;精华还原阶段,进一步调整各节点子集大小,同时尽可能减小连接代价,然后借助 G_m,G_{m-1},$\cdots,G_1$ 把划分映射回最初的图 G_0。下面详细介绍每个阶段。

3.1 粗化阶段

在粗化阶段,主要是基于原图 G_0 构造一系列点集与边集的大小逐渐减小的图 $G_i=(V_i,E_i)$,其中 $|V_i|<|V_{i-1}|$,通常一部分 G_i 的点结合成为 G_{i+1} 的一个点。令 V_i^v 为 G_i 中组成节点 v 的节点的集合,$v\in G_{i+1}$。为保持原图的权值信息,节点 v 的权值等于 V_i^v 中节点的权值的总和。同样,与 v 相连的边是与 V_i^v 中节点相连的所有的边的并集。如果在 V_i^v 中有多条边连接同一个 G_{i+1} 中另一个点 u,边 (v,u) 的权值为这些边的权值的总和。这样就保证了粗化的图与原图的连接代价相同。

算法中实现这种粗化的方法是秉承于"匹配"的思想[7]。图 G_i 的一个匹配 M_i(matching)为一个边的集合,$\forall e_1=(v_1,u_1),e_2=(v_2,u_2)\in M_i$,满足 $v_1\neq v_2,v_1\neq u_2,u_1\neq v_2,u_1\neq u_2$。算法构造图 G_i 的一个匹配,把已匹配的两个节点合并成 G_{i+1} 一个节点,未匹配的点就直接复制到 G_{i+1},从而减小 G_i 的大小。

3.1.1 最大权值边匹配算法

令 $W(E)$ 为 E 中所有的边的权值总和,显然 $W(E_{i+1})=W(E_i)-W(M_i)$。为了最小化图的连接代价,要找一个 M_i,使 $W(M_i)$ 尽量大。算法描述如下:

输入:图 $G_i=(V_i,E_i)$

输出:图 $G_{i+1}=(V_{i+1},E_{i+1})$

```
mapnum←0;
while not ∀v∈Vi,v is matched
  v←GetUnmatchedVertex( ); /*随机得到未匹配的节点 v*/
  if ∀u∈Adj(v) is matched /*Adj(v)为 v 的邻接点*/
    match[v]←v;
  else
    Find an unmatched u∈Adj(v) and w(v,u) is the largest;
                          /*在 v 的未匹配的邻接点中找 w(v,u) 最大的点 u*/
    match[v]←u; match[u]←v;
    map[u]←mapnum;
    Set u to be matched; /*把 u 标记为已匹配*/
  endelse
    map[v]←mapnum;
    Set v to be matched;
    mapnum←mapnum+1;
endwhile
```

不过对于那种所有的边的权值相同的图来说,没有所谓的最大权值的边。把算法改进一下,可以得到更好的匹配结果:从 v 的所有尚未匹配的邻接点中找到一个节点 u,令 W_{v-u} 为 u 连接到 v 的邻接点的边的权值的总和,即 $W_{v-u}=\sum_{(u,a),(v,a),(v,u)\in E}w(u,a)$,$u$ 满足 W_{v-u} 是所有未匹配中最大的。

算法在粗化阶段重复权重边匹配算法,直到粗化后图 G_i 中的节点数小于 ck,k 为需要划分的子集数,c 为一个常数。根据实验经验[4],取 $c=15$ 到 20。若相继两个图 G_i 和 G_{i+1} 的节点数相比,即 $|V_{i+1}|/|V_i|>0.8$,也结束粗化阶段。至此,原图被粗化为节点数和边数均大幅减小的图,在此基础上进行划分,将大大减少划分所用时间。

3.2 非均衡划分阶段

非均衡划分算法是个 k 步循环过程。在粗化阶段得到的 G_i 上划分 k 个指定大小的划分块,随后对划分产生的零碎块进行处理,得到一个初步划分 P_i。

3.2.1 非均衡划分增长算法

给定划分的分布向量 $A=(\alpha_1,\alpha_2,\cdots,\alpha_k)$,令 wgt 为 G_i 的节点的权值总和,算法从一点开始,按广度优先次序逐一将邻接点加入该划分,直到权值等于 $\alpha_i * wgt$,然后在余下的节点继续上述过程。由于同一划分中的节点应该是连通的,所以在划分过程中,难免会遇到这种情况。进行广度优先搜索邻接点时,队列中已无邻接的节点,但该划分的权值尚未达到所需大小。这称为划分的碎片问题。为了尽量避免碎片问题,同时使划分大小满足需求,本文给出如下算法描述:

输入:图 $G_i(V_i,E_i)$,分布向量 $A=(\alpha_1,\alpha_2,\cdots,\alpha_k)$,误差变量 ε

输出:划分映射数组 P

```
pnum←0;fragnum←k; /*pnum 为划分标识,fragnum 为碎片标识*/
while pnum<k and not ∀v∈Vi, v is partitioned
  pwgt←0; /*pwgt 为当前划分块的大小*/
  v←GetUnpartitionedVertex(); /*得到一个未划分的节点 v*/
  AddToTail (v,queue); /*将 v 加入队列尾*/
  while not empty(queue)
    v←GetFromHead(queue); /*从队列首取一点,赋值给 v*/
    pwgt←pwgt+w(v);
    P[v]←pnum;
    if pwgt between αpnum * wgt * (1−ε) and αpnum * wgt * (1+ε)
      pnum←pnum+1;
      SetEmpty(queue); /*将队列清空*/
    else if pwgt<αpnum * wgt * (1−ε)
      Add Adj(v) which is not partitioned into queue;
```

维普资讯 http://www.cqvip.com

```
                        /*将 v 的未划分且未加入队列的邻接点
                        加入队列*/
    else if pwgt>αpnum * wgt * (1+ε)
      roll back to the original state; /*回滚*/
    endif
  endwhile
  if pwgt<min(αi) * wgt * b /*b 为将划分定为碎片的临界值*/
    Set all P[v] which equal pnum to fragnum;
                    /*将属于划分 pnum 的点改为碎片块 fragnum*/
    fragnum←fragnum+1;
  endif
endwhile
```

在最外圈循环过程完毕后，若尚有 v 未划分，则进行碎片划分，$P[v]=fragnum$；若所有节点都已划分，但划分块不足 k 个，就从最大的碎片中补上。显然，碎片数加上已划分的块数大于等于 k。接下来的过程则是把碎片中的点逐一加入邻接的划分块中，并按照邻接划分块的权值与所需值之差由大到小的顺序加入。至此，算法产生了一个划分，划分块 i 的大小大致等于 $\alpha_i * wgt$，从而进入精化还原阶段。

3.3 精化还原阶段

精化还原阶段顾名思义包括两步：进一步调整各划分块的权值大小以及把划分 P 映射到初始图。还原过程十分简单，我们知道每个 G_{i+1} 中的节点 v 都是 G_i 中不相交的节点集合 V_i^v，还原过程所做的就是让 $P_i[u]=P_{i+1}[v]$，$u\in V_i^v$。然而我们发现，即使 P_{i+1} 中移动任意一个点，都不能改善各划分块的权值大小或减小连接代价。但还原到 G_i 后，便有进一步精化的可能，因此算法在还原的每一步都执行一遍精化过程。

精化有两点需要考虑：完善划分块的权值大小，同时尽可能减小连接代价。调整划分块大小主要就是把处在划分块边缘的节点移动到另一划分块，而减小连接代价则是基于 KL 算法的思想[8]，二者结合就是本文的精化算法。

3.3.1 精化算法

令 $Adj(v)$ 为 v 的邻接点的集合，$N(v)=\bigcup_{u\in Adj(v)}P_i[u]-\{P_i[v]\}$。显然，对于一个划分块内部的点，$N(v)=\phi$。对于每个 $p\in N(v)$，令 $ED(v)_p=\sum_{P_i[u]=p}w(v,u)$，$ID(v)=\sum_{P_i[v]=P_i[u]}w(v,u)$，$gain(v)_p=ED(v)_p-ID(v)$。若 $gain(v)_p$ 为正，则说明把 v 移到划分块 p 中能减少 $gain(v)_p$ 的连接代价。

下面给出算法描述：

输入：划分映射数组 P

输出：已精化的划分映射数组 P

```
while ∃ unrefined partitions
  p←GetUnrefinedPartition(); /*得到未精化的划分块的块号 p*/
  if |w(p)-W(p)| is in the range of error
          /*w(p)为 p 的实际权值，W(p)为 p 的所需权值*/
    Set p to be refined; /*将 p 设为已精化*/
  else if w(p)<W(p)
    Find unrefined q∈AdjPartition (p) whose w(q)-W(q) is the lar-
                                    gest;
                                    /*在 p 的未精化的邻接块中找 q 满
                                    足 w(q)-W(q)最大*/
    Find v∈BoundaryVeterx(q) which gain(v)p is the largest;
                                    /*在 q 的边界点中找一点 v 满
                                    足 gain(v)p最大*/
    P[v]←p;
    w(p)←w(p)+w(v);
    w(q)←w(q)-w(v);
  else if w(p)>W(p)
    Find unrefined q∈AdjPartition (p) whose w(q)-W(q) is the smal-
                                    lest;
                                    /*在 p 的未精化的邻接块中找 q 满
                                    足 w(q)-W(q)最小*/
    Find v∈BoundaryVeterx(p) which gain(v)q is the largest;
    P[v]←q;
    w(p)←w(p)-w(v);
    w(q)←w(q)+w(v);
  endif
endwhile
```

其中 AdjPartition(p)返回 p 的邻接划分块，BoundaryVeterx(p)返回 p 的边界点。精化过程完毕后，根据精化好的 P 执行还原过程。上述算法还有需要注意的是：若前一次 p 权值小于所需但加入一点又大于所需或相反情况，算法也终止 p 的精化过程。至此，经过一步步精化还原，最终达到了划分的目标。

3.4 算法分析

本文提出的异构计算中图的非均衡划分算法借鉴了多级划分的思想[3,4]。粗化阶段的目的是合并原图节点从而减小节点数，因此能大大减少划分时间。最大权值边匹配算法的复杂度为 $O(|E|)$。非均衡划分阶段的划分增长算法是广度优先搜索算法的一种变形，它对碎片问题进行了处理，能保证划分出 k 个划分块。其复杂度也是 $O(|E|)$。精化过程要得到每个点的邻接点情况，所以复杂度同样是 $O(|E|)$。因此，整个算法的复杂度与边数有着线性关系。

与均衡划分相比，非均衡划分算法更注重每个划分块严格按照分布向量来划分大小，其次是连接代价，均衡划分研究重点则是如何减少连接代价。二者最大的区别就是在对划分块大小的控制上，非均衡划分实现时，在划分块大小条件的判断上更为复杂。

4 算法实验与测试

本文实验所用的图均来自格林威治大学 Chris Walshaw 教授所收集的一系列开放的测试用图，表 1 描述了各图的属性，包括节点数、边数以及图的相关描述信息[9]。可以从如下网址获得详细信息：http://staffweb.cms.gre.ac.uk/~c.walshaw/partition。

表 1 测试用图属性

图名	节点数	边数	描述
3elt	4720	13722	2D Finite element mesh
4elt	15606	45878	2D Finite element mesh
add20	2395	7462	20-bit adder
add32	4960	9462	32-bit adder
bcsstk33	8738	291583	3D Stiffness matrix
brack2	62631	366559	3D Finite element mesh
crack	10240	30380	2D nodal graph
cs4	22499	43858	3D dual graph
cti	16840	48232	3D semi-structured graph
data	2851	15093	Map data
fe_sphere	16386	49152	Sphere graph
memplus	17758	54196	Memory circuit
uk	4824	6837	2D dual graph
whitaker3	9800	28989	2D Finite element mesh
Shmap	14007	21753	Map of Shanghai

本文以 uk 和 data 为例，给出划分后各个划分块的权值大小的实验数据。我们把这两个图划分成 4 个划分块，分布向量 $A=(0.1, 0.2, 0.3, 0.4)$，误差值 $\varepsilon=0.02$。data 的总权值为 2851，各个划分块的大小分别为 290，581，862，1118。uk 的总权值为 4824，各个划分块的大小分别为 492，946，1476，1910。实验表明，划分算法所得到的结果是正确的，各划分块大小均在误差范围之内。

下面，本文给出该算法在余下的图上所产生的连接代价与均衡划分的连接代价的比较，均衡划分的连接代价的数据来自文[3,4,9]。目前为止，这是该领域所知晓的最好的划分结果。比较结果如表 2 所示。其中，非均衡划分采用的分布向量如下：

$A_1=(0.3, 0.7)$；

维普资讯 http://www.cqvip.com

$A_2=(0.2,0.2,0.3,0.3)$；

$A_3=(0.05,0.05,0.1,0.1,0.15,0.15,0.2,0.2)$；

$A_4=(0.025,0.025,0.025,0.025,0.05,0.05,0.05,0.05,0.075,0.075,0.075,0.075,0.1,0.1,0.1,0.1)$；

A_5 为32维分布向量，每个元素均相等，值为0.03125。

本文引用的均衡划分的连接代价数据均是在把图划分成32块的条件下所得的。

表2 均衡与非均衡划分的连接代价比较，EC为连接代价

图名	边数	非均衡划分 EC					均衡划分 EC
		A_1	A_2	A_3	A_4	A_5	划分块数32
3elt	13722	175	373	647	1092	1424	972
4elt	45878	267	676	1288	2489	3088	1606
add20	7462	1469	1945	2163	2734	2801	2765
add32	9462	12	27	96	161	304	234
brack2	291583	7113	17071	19239	33120	32568	22451
bcsstk33	366559	19413	45300	61986	82264	100756	79121
crack	30380	374	788	1067	1868	2680	1768
cs4	43858	750	2436	3411	5331	5668	3110
cti	48232	1940	3909	4562	6291	7248	4460
fe_sphere	49152	736	1595	1963	3728	3903	2567
memplus	54196	5492	6028	12732	15284	15694	14830
whitaker3	28989	180	685	1068	1581	2466	1719
Shmap	21753	121	246	412	787	942	无

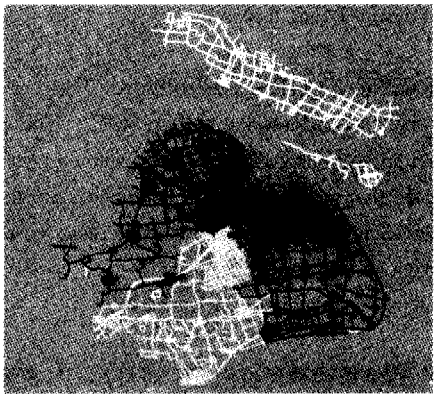

图1 上海地图非均衡划分

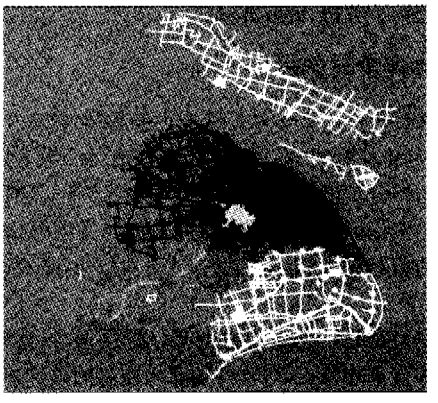

图2 上海地图均衡划分

为清楚展示划分结果，本文以上海地图为例，对其分别进行非均衡和均衡划分，并以不同颜色标识不同的划分块，结果如图1和图2所示。二者均为8划分，图1的分布向量为 A_3。在图1中，白、黑、蓝、青、黄、绿、紫红、红依次对应 A_3 中的8个分量。我们知道城区与郊区节点的稠密度差别较大，二图基本反映了各划分块的大小关系。

实验结果发现，当非均衡划分采用分布向量 A_5 与均衡划分32个划分块相比较，非均衡划分的连接代价要更大一点。主要由于算法侧重点不同，在非均衡划分与还原精化时，为了保证划分块大小，节点的gain不是首先考虑的因素所导致。如何更多减少连接代价也是今后所要研究的重点。

结语 本文提出的异构计算中图的非均衡划分算法在多级划分算法的基础上给出了一种按照分布向量来划分各划分块的大小或权值的方法。这种算法可以运用在异构计算中并行任务调度的负载平衡以及相应的应用问题中，为异构环境中的并行任务调度提供了一定支持，对于类似问题的其它领域应用也具有一定的参考价值。针对该算法，我们做了大量实验来检验其实效性。实验结果表明，这是正确且行之有效的。该算法尚有不足，在如何保证划分块大小符合需求的情况下进一步减小连接代价，是今后研究改进的重点。

参考文献

1 Freund R F, Siegel H J. Heterogeneous processing. Computer, 1993, 26(6): 18～27

2 曾国荪，陆鑫达. 异构计算中的负载共享. 软件学报，2000，11(4)：551～556

3 Karypis G, Kumar V. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. SIAM J Sci Comput, 1998, 20(1): 359～392

4 Karypis G, Kumar V. Multilevel k-way Partitioning Scheme for Irregular Graphs. J Parallel Distrib Comput, 1998, 48(1): 96～129

5 Hendrickson B, Leland R. An improved spectral graph partitioning algorithm for mapping parallel computations. SIAM J Sci Comput, 1995, 16(2): 452～469

6 Miller G L, Teng Shang-Hua, Vavasis S A. A unified geometric approach to graph separators. In: Proceedings of 31st Annual Symposium on Foundations of Computer Science, 1991. 538～547

7 Hendrickson B, Leland R. A multilevel algorithm for partitioning graphs: [Technical Report]. SAND93-1301, Sandia National Laboratories, 1993

8 Kernighan B W, Lin S. An efficient heuristic procedure for partitioning graphs. The Bell System Technical Journal, 1970, 49(2): 291～307

9 http://staffweb.cms.gre.ac.uk/~c.walshaw/partition