



(12) 发明专利

(10) 授权公告号 CN 102968173 B

(45) 授权公告日 2015. 08. 26

(21) 申请号 201210421845. 9

US 2012/0060142 A1, 2012. 03. 08,

(22) 申请日 2012. 10. 30

CN 101473301 A, 2009. 07. 01,

(66) 本国优先权数据

JP 特开平 9-179738 A, 1997. 07. 11,

201210367870. 3 2012. 09. 28 CN

于利前等. 《静动态结合的 Java 程序不变性分析方法》. 《计算机学报》. 2010, 第 33 卷 (第 4 期),

(73) 专利权人 北京航空航天大学

地址 100191 北京市海淀区学院路 37 号

审查员 罗湘

(72) 发明人 牛建伟 宋文芳

(74) 专利代理机构 北京永创新实专利事务所

11121

代理人 周长琪

(51) Int. Cl.

G06F 1/32(2006. 01)

G06F 9/44(2006. 01)

(56) 对比文件

US 2010/0131721 A1, 2010. 05. 27,

US 5579518 A, 1996. 11. 26,

权利要求书2页 说明书9页 附图12页

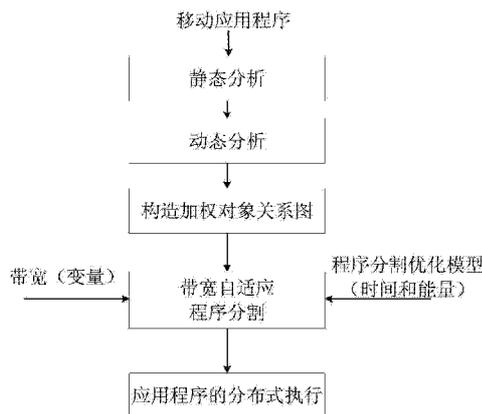
(54) 发明名称

一种基于带宽自适应代码迁移的移动设备节能方法

(57) 摘要

本发明提出一种基于带宽自适应代码迁移的移动设备节能方法,通过对移动应用程序进行静态和动态分析,构造出一种加权对象关系图,从而将程序分割问题转化为对象关系图分割问题;将移动环境中变化的带宽看作一个变量,提出基于运行时间和能量最优的目标模型;针对程序分割优化模型提出带宽自适应的分支定界程序分割法(BBAP)和基于最小割的贪心程序分割法(MCGAP);最后基于所得到的分割方案实现移动应用程序在移动设备和服务器间的代码迁移和分布式执行。

本发明方法能够有效缩短移动应用程序的运行时间,降低移动设备的电能消耗,并对带宽变化具有良好的适应性。



1. 一种基于带宽自适应代码迁移的移动设备节能方法,在移动环境带宽不断变化的情况下,其特征在于,该移动设备节能方法通过下面步骤实现:

步骤 1:构造加权对象关系图,具体是:首先,进行静态程序分析,根据程序的方法调用关系,构造程序初始的对象关系图,对象关系图中的各节点和边分别对应程序中的对象和对象间的调用关系;然后,通过动态程序分析获得对象关系图中各节点对象的运行时间以及每条边的通信数据量,得到加权对象关系图;

步骤 2:基于加权对象关系图,将带宽作为一个变量,建立基于运行时间和耗电量的程序分割优化模型,具体是:令  $x_i$  标识节点  $i$  的运行位置,  $x_i = 1$  和  $x_i = 0$  分别表示节点  $i$  在移动设备和服务器端运行,程序分割方案表示为  $X = \{x_1, x_2, \dots, x_n\}$ ,  $n$  为加权对象关系图中的节点个数,则程序分割优化模型的目标在于求得最优分割方案  $\bar{X}$ , 满足  $\bar{X} = \arg \min_{X \in X_p} W(X, b)$ ,  $X_p$  为模型求解空间,  $b$  表示当前带宽,  $W(X, b)$  表示当前带宽为  $b$  时程序分割方案  $X$  的优化模型,具体定义如下:

$$W(X, b) = w_t \times T(X, b) / T_{Local} + w_e \times E(X, b) / E_{Local}$$

其中,  $T_{Local}$  和  $E_{Local}$  分别表示程序全部运行在移动设备的运行时间和耗电量;  $w_t$  和  $w_e$  分别表示运行时间和耗电量的权重,且满足  $w_t + w_e = 1$ ;

$T(X, b)$  表示当前带宽为  $b$  时采用程序分割方案  $X$  的程序运行时间:

$$T(X, b) = \sum_{1 \leq i \leq n} (x_i \times t_{ni} + (1 - x_i) \times t_{nsi}) + \sum_{1 \leq i < j \leq n} |x_i - x_j| \times t_{ij}$$

其中,  $t_{ni}$  与  $t_{nsi}$  分别表示节点  $i$  在移动设备与服务器端的运行时间;  $t_{ni}$  为加权对象关系图中对应节点  $i$  的权重,设服务器计算速度是移动设备的  $k$  倍,则:  $t_{nsi} = t_{ni}/k$ ; 令  $W_{ij}$  表示移动设备与服务器端的通信数据量,则带宽为  $b$  时节点  $i, j$  间数据传输时间  $t_{ij}$  为:  $t_{ij} = W_{ij}/b$ ;

$E(X, b)$  表示当前带宽为  $b$  时采用程序分割方案  $X$  的程序耗电量:

$$E(X, b) = \sum_{1 \leq i \leq n} x_i \times E_i + \sum_{1 \leq i < j \leq n} |x_i - x_j| \times E_{ij}$$

其中,  $E_i$  表示节点  $i$  在移动设备上运行的耗电量,  $E_i = t_{ni} \times P_{cpu}$ ;  $E_{ij}$  表示节点  $i, j$  之间数据传输的耗电量,  $E_{ij} = t_{ij} \times P_w$ ;  $P_{cpu}$  表示移动设备的 CPU 功率,  $P_w$  表示移动设备 WiFi 接口的数据通信功率;

所述的模型求解空间  $X_p$  定义为:  $X_p = \{X | W(X, b_1) \leq W(X_{b_1}^{opt}, b_1) \times (1 + a)\}$ , 其中, 带宽阈值  $b_1$  为满足概率  $P(b \geq b_1) \geq P_c$  的最大带宽值,  $P_c$  为所设定的概率阈值;  $W(X_{b_1}^{opt}, b_1)$  为带宽  $b_1$  下最优程序分割方案  $X_{b_1}^{opt}$  对应的优化模型值; 经验常数  $a > 0$ ;

步骤 3:进行带宽自适应程序分割,在节点个数  $n$  不超过 200 时,采用分支定界程序分割方法寻找最优分割方案,在节点个数  $n$  大于 200 时,采用基于最小割的贪心方法来寻找最优分割方案;

所述的分支定界程序分割方法基于分支定界深度优先搜索法,深度搜索的约束条件包括:1) 程序分割方案在当前带宽  $b$  下对应的优化模型值小于当前保存的优化模型的最小值;2) 程序分割方案在带宽阈值  $b_1$  下的模型目标值不高于所设定模型上限值

$$W(X_{b_1}^{opt}, b_1) \times (1+a);$$

所述的基于最小割的贪心方法,利用 Stoer-Wagner 最小割算法求得静态带宽为  $b$  和  $b_1$  时对应的运行在移动设备的节点集合  $C_b$  和运行在服务器端的节点集合  $S_b$ , 令待分割节点  $V' = V - C_b - S_b$ ,  $V$  表示加权对象关系图中的节点集合,对  $V'$  中节点进行分割,采用 Stoer-Wagner 最小割算法中的最大邻接值排序算法对节点进行排序,然后基于贪心策略,同时根据与分支定界程序分割方法中相同的深度搜索的约束条件选取最优分割方案;

步骤 4:根据得到的最优分割方案,实现程序在移动设备和服务器间的代码迁移和分布式执行。

2. 根据权利要求 1 所述的一种基于带宽自适应代码迁移的移动设备节能方法,其特征在于,所述的概率阈值  $P_c$  设定为 0.85。

3. 根据权利要求 1 所述的一种基于带宽自适应代码迁移的移动设备节能方法,其特征在于,所述的经验常数  $a$  设定为 0.4。

## 一种基于带宽自适应代码迁移的移动设备节能方法

### 技术领域

[0001] 本发明属于移动计算领域,具体涉及一种基于带宽自适应代码迁移的移动设备节能方法。

### 背景技术

[0002] 移动设备因其轻巧便携、使用方便等优点,逐渐成为用户量最大的计算平台。随着移动用户的需求不断增长,移动设备的性能不断提升,电池容量已经成为移动设备进一步发展的瓶颈问题。近年来,基于程序分割的代码迁移技术逐渐应用于移动计算领域,成为缓解移动设备资源有限问题的一种有效方法。根据程序分割方案将应用程序的一部分代码迁移到服务器端运行,可以缩短应用程序的运行时间,降低移动设备的能量消耗。

[0003] 面向对象应用程序的复杂性和多样性为实现程序分割带来了挑战。国内外学者在这一领域进行了较为广泛的研究并提出了相应的方案。Eduardo 等人提出的 MAUI 系统是基于微软通用语言运行库 CLR 的代码迁移系统,需要程序员事先对程序源代码中可在远端服务器执行的类添加“Remotable”标识;Roelof 等人提出的 Cuckoo 系统通过扩展 Android 编译系统,为程序员提供了一种便捷的编程模型,可以辅助程序员完成一部分工作(例如远程方法调用的实现),但仍需程序员手动实现;Shumao 等人则是借助中间件实现移动设备上的程序分割和迁移。这些方案依赖于程序员或中间件工具,限制了其应用领域。

[0004] 于是 Sinha 等研究者进一步提出了自动程序分割方案,这种方法无需借助额外工具,不依赖于程序员,可以透明地实现程序自动分割,适用于移动平台。近年来,面向对象程序的自动分割方案大都通过对程序进行纯静态分析或纯动态分析构造程序对象关系图,并基于对象关系图(Object Relation Graph, ORG)实现程序分割。Spiegel 等人和 Dahm 提出的 Doorastha 系统均采用纯静态程序分析方法构造对象关系图,不能够准确反映程序的运行态信息,因而无法获取最优的程序分割结果。而 Messer 等人和 Bialek 等人则采用纯动态程序分析方法构造对象关系图,相比纯静态分析,纯动态程序分析能够获取更为精确的程序运行时信息,但得到的对象关系图庞大复杂,导致程序分割开销过大,不适用于资源有限的移动平台。Wang Lei 等人通过构造两层对象关系图,提出了适用于不同目标模型的程序分割方案和基于 Java 的分布式系统框架,但并未对程序分割方法进行具体研究,没有考虑移动环境中带宽的变化。

[0005] 程序分割的关键是如何选择程序的最优分割方案,尽可能实现移动设备能耗最小和数据传输开销最小。由于数据传输开销与网络带宽直接相关,不同的带宽会对应不同的最优分割方案。以往的程序分割通常假设带宽固定不变,如 Diaconescu 等人即根据静态带宽计算网络传输开销以决策程序分割,并未考虑网络环境因素(如带宽)的变化。由于移动网络环境中带宽变化频繁,传统的静态程序分割技术通常会保守地根据较低带宽决策分割方案。然而这种分割方案在带宽较高情况下无法充分利用带宽资源,造成网络资源浪费。Byung-Gon 等人通过分析移动设备和网络环境的异构和动态变化特性,提出了动态程序分割,但频繁的程序分割任务势必为移动设备带来较大的额外开销负担。

## 发明内容

[0006] 针对移动设备能量资源有限问题,为了有效节省移动设备的能量消耗,并克服静态程序分割不适用于带宽变化的移动环境和动态程序分割额外开销较大的问题,本发明提出一种基于带宽自适应代码迁移的移动设备节能方法。

[0007] 本发明的一种基于带宽自适应代码迁移的移动设备节能方法,具体过程如下:

[0008] 步骤1,构造加权对象关系图,具体是:首先,对应用程序进行静态程序分析,根据应用程序的方法调用关系,获取应用程序中的对象和对象间的调用关系,构造应用程序初始的对象关系图(Initial Object Relation Graph, IORG),对象关系图的节点和边分别对应应用程序对象和对象间的调用关系;然后,通过动态程序分析获得对象关系图中各节点对象的运行时间以及对象间的调用关系(每条边)的通信数据量,得到加权对象关系图。

[0009] 步骤2,基于以上加权对象关系图,将带宽作为一个变量,建立基于运行时间和耗电量的程序分割优化模型。令  $x_i$  标识节点  $i$  的运行位置,  $x_i = 1$  和  $x_i = 0$  分别表示节点  $i$  在移动设备和服务器端运行,则程序分割方案可表示为  $X = \{x_1, x_2, \dots, x_n\}$ ,  $n$  为节点个数,令  $b$  表示当前带宽。程序分割优化模型的目标在于求得最优分割方案  $\bar{X}$ , 满足  $\bar{X} = \arg \min_{X \in X_p} W(X, b)$ ,  $X_p$  为模型求解空间,  $W(X, b)$  表示当前带宽为  $b$  时程序分割方案  $X$  的优化模型,具体定义如下:

$$[0010] \quad W(X, b) = w_t \times T(X, b) / T_{Local} + w_e \times E(X, b) / E_{Local}$$

[0011] 其中,  $T_{Local}$  和  $E_{Local}$  分别表示程序全部运行在移动设备时的运行时间和耗电量;  $w_t$  和  $w_e$  分别表示运行时间和耗电量的权重,且满足  $w_t + w_e = 1$ ;  $T(X, b)$  和  $E(X, b)$  分别表示当前带宽为  $b$  时采用程序分割方案  $X$  的程序运行时间和耗电量;模型求解空间  $X_p$  表示为:

$$[0012] \quad X_p = \left\{ X \mid W(X, b_1) \leq W(X_{b_1}^{opt}, b_1) \times (1+a) \right\}$$

[0013] 其中,带宽阈值  $b_1$  为满足条件  $P(b \geq b_1) \geq P_c$  的最大带宽值,  $P_c$  为所设定的概率阈值;  $W(X_{b_1}^{opt}, b_1)$  为带宽  $b_1$  下最优分割方案  $X_{b_1}^{opt}$  对应的模型值;  $a > 0$  为经验常数,用以设定  $W(X, b_1)$  的上限值为  $W(X_{b_1}^{opt}, b_1) \times (1+a)$ 。

[0014] 步骤3,进行带宽自适应程序分割,在节点个数  $n$  不超过200时,采用基于深度优先搜索策略的分支定界程序分割方法寻找最优分割方案,在节点个数  $n$  大于200时,采用基于最小割的贪心方法(MCGAP)来寻找最优分割方案。

[0015] 所述的分支定界程序分割方法基于分支定界深度优先搜索法,深度搜索的约束条件包括:1) 程序分割方案在当前带宽  $b$  下对应的优化模型值小于当前保存的优化模型的最小值;2) 程序分割方案在带宽阈值  $b_1$  下的模型目标值不高于所设定模型上限值  $W(X_{b_1}^{opt}, b_1) \times (1+a)$ 。

[0016] 所述的基于最小割的贪心方法,利用 Stoer-Wagner 最小割算法求得带宽为  $b$  和  $b_1$  时对应的运行在移动设备的节点集合  $C_b$  和运行在服务器端的节点集合  $S_{b_1}$ , 令待分割节点  $V' = V - C_b - S_{b_1}$ , 对  $V'$  中节点进行分割,采用 Stoer-Wagner 最小割算法中的最大邻接值排

序算法对节点进行排序,然后基于贪心策略,同时根据与分支定界程序分割方法中相同的深度搜索的约束条件选取最优分割方案。

[0017] 步骤 4,根据步骤 3 求得的最优分割方案,实现程序在移动设备和服务器间的代码迁移和分布式执行,有效降低应用程序的执行时间和移动设备的耗电量,延长移动设备的电池使用寿命。

[0018] 本发明的优点与积极效果在于:(1)采用静态与动态程序分析相结合的方法构造程序的加权对象关系图,所构建的对象关系图复杂度低,并能够准确地反映程序结构,有利于程序分割,适用于资源有限的移动平台;(2)将移动环境中带宽作为一个变量,建立基于运行时间和耗电量的程序分割优化模型,适用于带宽变化的移动环境;(3)基于加权对象关系图和程序分割优化模型,提出了两种带宽自适应的程序分割方法:分支定界程序分割法和基于最小割的贪心方法,前者能够求出小规模程序分割的最优解,后者可快速求得大规模程序分割的近似最优解;两种方法能够适应带宽变化,有效缩短程序运行时间,降低移动设备的能量消耗。

#### 附图说明

[0019] 图 1 为本发明的带宽自适应程序分割流程图;

[0020] 图 2 为本发明的示例程序图;

[0021] 图 3 为本发明的示例程序初始对象关系图;

[0022] 图 4 为本发明的示例程序加权对象关系图;

[0023] 图 5 为本发明的 BBAP 方法伪代码示意图;

[0024] 图 6 为本发明的 MCGAP 方法伪代码示意图;

[0025] 图 7 为采用三种方法构造加权对象关系图的大小比较示意图;

[0026] 图 8 为本发明实施例中程序实际运行时间与三种方法的预测运行时间的对比示意图;

[0027] 图 9 为本发明的仿真实验参数示意图;

[0028] 图 10 为本发明的仿真实验加权对象关系图信息;

[0029] 图 11 为本发明的 BBAP 方法对目标模型三种特殊情况的求解分割时间对比图;

[0030] 图 12 为本发明的 BBAP 方法的分割结果在不同带宽下对应的运行时间对比示意图;

[0031] 图 13 为本发明的 BBAP 方法的分割结果在不同带宽下对应的耗电量对比示意图;

[0032] 图 14 为本发明的 BBAP 方法的分割结果在不同带宽下对应的加权模型值对比示意图;

[0033] 图 15 为本发明的 MCGAP 方法对目标模型三种特殊情况的求解分割时间对比图;

[0034] 图 16 为本发明的 BBAP 与 MCGAP 运行时间最优模型求解结果对比图;

[0035] 图 17 为本发明的 MCGAP 方法的分割结果在不同带宽下对应的运行时间对比示意图;

[0036] 图 18 为本发明的 MCGAP 方法的分割结果在不同带宽下对应的耗电量对比示意图;

[0037] 图 19 为本发明的 MCGAP 方法的分割结果在不同带宽下对应的加权模型值对比示

意图；

[0038] 图 20 为本发明的 BBAP 与 MGCAP 的平均性能对比图。

### 具体实施方式

[0039] 下面将结合附图和实例对本发明作进一步的详细说明。

[0040] 本发明提出的一种基于带宽自适应代码迁移的移动端节能方法，首先，采用静态与动态分析相结合的方法分析移动应用程序，并构造其对应的加权对象关系图；其次，将移动环境中变化的带宽作为一个变量，建立基于运行时间和耗电量的程序分割优化模型；然后，基于对象关系图和程序分割模型实现带宽自适应的程序分割方法；最后，根据分割结果实现程序在移动设备和服务器间的代码迁移和分布式执行，从而降低应用程序的执行时间和移动设备的耗电量，延长移动设备电池的使用寿命，而且应用程序的运行时间和耗电量随带宽变化波动较小，对带宽变化具有一定的适应性。如图 1 所示，本发明的移动端节能方法，具体过程如下：

[0041] 步骤 1，构造加权对象关系图。

[0042] 本发明实施例以一个 Java 程序为例来阐述加权对象关系图的构造过程，示例程序如图 2 所示，程序包含四个类：FacePreview、ImageCapture、FaceDetection 和 FaceDetectionLib，其中 FacePreview 类的 main 方法中创建了一个 ImageCapture 对象和两个 FaceDetection 对象，而两个 FaceDetection 对象均在执行自身 DetectFace 方法时调用了 FaceDetectionLib 类的 ProcessBlock 方法。首先，进行静态程序分析，利用指向分析 (Points-to) 技术分析 Java 程序的字节码文件，得到其方法调用关系图，并通过遍历调用关系图获取程序的所有对象及对象间的调用关系，从而构造出程序的初始对象关系图 (IORG)，其节点和边分别对应应用程序的对象和对象间的调用关系。图 3 表示为示例程序对应的初始对象关系图，S\_FacePreview 创建对象 D\_ImageCapture、D\_FaceDetection1 和 D\_FaceDetection2，对象 D\_FaceDetection1 和 D\_FaceDetection2 都调用 S\_FaceDetectionLib 的方法，其中 S\_ 和 D\_ 前缀分别表示静态对象和动态对象。由于节点间的交互主要以方法调用的方式体现，其通信数据主要表现为方法调用的参数及返回值，在动态程序分析阶段，利用 Java 字节码重写技术，可以获得程序 IORG 中每个节点对象的 CPU 运行时间以及每条边（对象间的调用关系）的通信数据量，即加权对象关系图中各节点与边的权值。图 4 表示为示例程序对应的加权对象关系图，例如节点 S\_FacePreview 所代表的对象的 CPU 运行时间为 50ms，节点 S\_FacePreview 与节点 D\_ImageCapture 之间的通信数据量为 9066bytes。

[0043] 步骤 2，建立程序分割优化模型。

[0044] 考虑到移动环境中带宽的变化，本发明将带宽作为一个变量，提出基于运行时间和耗电量的程序分割优化模型，令  $x_i$  标识节点  $i$  的运行位置， $x_i = 1$  和  $x_i = 0$  分别表示节点  $i$  在移动设备和服务器端运行，则程序分割方案可表示为  $X = \{x_1, x_2, \dots, x_n\}$ ， $b$  表示当前带宽， $\bar{X}$  表示最优程序分割方案。程序分割优化模型的目标在于求得最优分割方案  $\bar{X}$ ，满足  $\bar{X} = \arg \min_{X \in X_p} W(X, b)$ ， $X_p$  表示模型求解空间， $W(X, b)$  表示当前带宽为  $b$  时分割方案  $X$  的优化模型，具体定义如下：

[0045]  $W(X, b) = w_t \times T(X, b) / T_{Local} + w_e \times E(X, b) / E_{Local}$

[0046] 其中,  $T_{Local}$  和  $E_{Local}$  分别表示程序全部运行在移动设备时的运行时间和耗电量;  $w_t$  和  $w_e$  分别表示运行时间和耗电量的权重, 且满足  $w_t + w_e = 1$ ;  $T(X, b)$  和  $E(X, b)$  分别表示当前带宽为  $b$  时采用程序分割方案  $X$  的程序运行时间和耗电量; 模型求解空间  $X_p$  表示为:

[0047]  $X_p = \{X | W(X, b_1) \leq W(X_{b_1}^{opt}, b_1) \times (1+a)\}$

[0048] 其中, 带宽阈值  $b_1$  为满足概率  $P(b \geq b_1) \geq P_c$  的最大带宽值,  $P_c$  为概率阈值, 主要根据无线移动环境中的带宽变化情况具体设定, 本发明的实施实验中设定  $P_c$  为 0.85。 $W(X_{b_1}^{opt}, b_1)$  为带宽  $b_1$  下最优分割方案  $X_{b_1}^{opt}$  对应的优化模型值;  $a$  为经验常数, 用以设定  $W(X, b_1)$  的上限值为  $W(X_{b_1}^{opt}, b_1) \times (1+a)$ , 其中  $a$  的优选值与具体应用程序及其移动环境有关, 本发明设定的优选范围为  $0 < a < 1$ , 本发明中根据对具体应用程序的实验测试, 该参数的优选值为  $a = 0.4$ 。

[0049] 带宽为  $b$  时采用程序分割方案  $X$  的程序运行时间  $T(X, b)$  表示如下:

[0050]  $T(X, b) = \sum_{1 \leq i \leq n} (x_i \times t_{ni} + (1-x_i) \times t_{nsi}) + \sum_{1 \leq i < j \leq n} |x_i - x_j| \times t_{ij}$

[0051] 其中,  $t_{ni}$  与  $t_{nsi}$  分别表示节点  $i$  在移动设备与服务端端的运行时间, 节点  $i$  在移动设备的运行时间  $t_{ni}$  在动态分析阶段可以得到, 即节点  $i$  的权重, 设服务器计算速度是移动设备的  $k$  倍, 则:

[0052]  $t_{nsi} = t_{ni} / k$

[0053]  $W_{ij}$  表示在不同位置 (移动设备或服务端) 执行的节点  $i$  与节点  $j$  的通信数据量, 则带宽为  $b$  时节点  $i, j$  间数据传输时间  $t_{ij}$  为:

[0054]  $t_{ij} = W_{ij} / b$

[0055] 带宽为  $b$  时采用程序分割方案  $X$  的程序耗电量  $E(X, b)$  表示为:

[0056]  $E(X, b) = \sum_{1 \leq i \leq n} x_i \times E_i + \sum_{1 \leq i < j \leq n} |x_i - x_j| \times E_{ij}$

[0057] 其中,  $E_i$  表示节点  $i$  在移动设备上执行的耗电量,  $E_{ij}$  表示节点  $i, j$  之间数据通信的耗电量; 令  $P_{cpu}$  表示移动设备的 CPU 功率。一般情况下, 移动设备 WiFi 接口的接收功率与发送功率差别较小, 均表示为数据通信功率  $P_w$ , 则有:

[0058]  $E_i = t_{ni} \times P_{cpu}$

[0059]  $E_{ij} = t_{ij} \times P_w$

[0060] 由于  $\bar{X} \in X_p$ , 则  $W(\bar{X}, b_1) \leq W(X_{b_1}^{opt}, b_1) \times (1+a)$ 。若  $b \geq b_1$ , 则  $W(\bar{X}, b) \leq W(\bar{X}, b_1) \leq W(X_{b_1}^{opt}, b_1) \times (1+a)$ ;

又由于  $P(b \geq b_1) \geq P_c$ , 因此在目标模型中最优分割方案  $\bar{X}$  满足:

[0061]  $P(W(\bar{X}, b) \leq W(X_{b_1}^{opt}, b_1) \times (1+a)) \geq P_c$

[0062] 这说明所求最优分割方案在带宽变化情况下能够以概率  $P_c$  使所对应优化模型的值满足求解空间  $X_p$  的阈值限制条件。

[0063] 该模型中运行时间和耗电量的权重可以由用户根据移动设备剩余电量和对程序响应时间的需求灵活设定。特别地, 当  $w_t = 1$  和  $w_e = 0$  时, 即对应运行时间最优模型; 当  $w_t$

$= 0$  和  $w_e = 1$  时,即为耗电量最优模型。

[0064] 步骤 3,进行带宽自适应程序分割。

[0065] 在上述加权对象关系图和程序分割优化模型的基础上,首先,基于深度优先搜索策略,提出分支定界程序分割方法 (BBAP),深度搜索的约束条件不仅满足分割方案在当前带宽下对应的优化模型值小于当前保存的最小优化模型值,同时还需保证此分割在带宽阈值  $b_1$  下的模型值不高于所设定的优化模型的上限值,从而避免所选分割方案在带宽较低时开销过大,实现对带宽变化的适应性。但由于在最坏情况下方法时间复杂度为  $O(2^n)$  ( $n$  为节点个数),该方法不适用于对象 (节点数) 较多规模较大的应用程序,在本发明的实验平台下当移动应用程序对应的对象节点数大于 200 时,该方法计算程序分割的求解时间会明显增加。为此,本发明针对大规模应用程序,从贪心策略考虑提出启发式的基于最小割的贪心方法 (MCGAP)。MCGAP 方法采用最小割 Stoer-Wagner 方法中用到的最大邻接值排序方法对对象节点进行排序,以 BBAP 方法中的约束条件为限制,基于贪心策略快速解得程序分割方案。

[0066] 本发明从搜索策略和贪心策略考虑,提出分别适用于小规模 and 大规模应用程序分割的分支定界程序分割法 (BBAP) 和基于最小割的贪心方法 (MCGAP),其伪代码分别如图 5,图 6 所示,其中输入主要包括当前带宽  $b$ 、带宽阈值  $b_1$ ,以及程序对应的加权对象关系图  $WORG = (V, E, W_v, W_e)$ ,  $V$ 、 $E$ 、 $W_v$  和  $W_e$  分别表示节点集合、边集合、节点权重集合和边权重集合;输出为程序的最优分割方案和优化模型对应的值。

[0067] 如图 5 所示, BBAP 方法的第 1 和 2 行伪代码首先采用 Stoer-Wagner 静态方法求解带宽为  $b_1$  时的最优分割方案对应的优化模型的值  $\min V_{SW}$ , 并设定  $b_1$  带宽下优化模型值的上限值  $\min V$ 。第 3 行初始化所要求解的优化模型最优值  $\min Value$  为一个所能表示的最大整数。 $N_L$  表示必须在本地移动设备上执行的节点集合,如与用户交互的接口对象等节点,第 4-8 行 BBAP 方法将必须运行在本地移动设备的节点标记为“ $x_i = 1$ ”,  $V$  中不属于  $N_L$  的节点都属于待分割节点,待分割节点标记为“ $x_i = -1$ ”。第 9 行 BBSearch 函数基于基本的分支定界法,采用深度优先搜索法求得模型的最优解,每次迭代中均将当前的解空间分成两个子空间 ( $x_i = 0$  表示该节点在服务器执行;  $x_i = 1$  表示该节点在本地移动设备上执行),并采用剪枝策略进行方法优化; BBSearch 中的约束条件不仅满足分割方案在当前带宽下对应的优化模型的值小于当前保存的模型最小值 ( $\min Value$ ),同时还需保证此分割在带宽阈值  $b_1$  下的优化模型的值不高于所设定的模型阈值 ( $\min V$ ),从而避免分割方案在带宽较低时开销过大,保证对带宽变化的适应性。最后 BBAP 方法的第 10 行返回程序的最优分割方案  $X_{\min}$  和优化模型对应的最优值  $\min Value$ 。最坏情况下 BBAP 方法时间复杂度为  $O(2^n)$ , 当应用程序对象节点数较多时,计算复杂度较高,因此该方法不适用于规模较大应用程序的分割。

[0068] 由于带宽越高,数据传输开销越低,因此,带宽较高情况下运行于移动设备的节点在带宽降低时将仍运行于移动设备,而带宽较低情况下运行于服务器的节点在带宽升高时将仍运行于服务器。记  $X_b = (C_b, S_b)$  表示带宽  $b$  下对应的程序最优分割方案,  $C_b$  和  $S_b$  分别表示带宽为  $b$  时运行在移动设备和服务器端的节点集合,对于某个带宽  $B$  下对应的程序最优分割方案  $X_B = (C_B, S_B)$ , 有: (1)  $B > b \Rightarrow C_B \subseteq C_b$ , (2)  $B > b \Rightarrow S_b \subseteq S_B$ 。MCGAP 方法中,考虑在  $b$  与  $b_1$  间变化的带宽,设  $\bar{\delta} = (C_{\min}, S_{\min})$  为模型最优解,则有  $C_b \subseteq C_{\min}$ ,  $S_b \subseteq S_{\min}$ 。

[0069] 如图 6 所示, MCGAP 方法首先在第 1 行代码中将所求解的优化模型最优值  $\text{minValue}$  初始化为一个所能表示的最大整数。第 2-4 行根据 Stoer-Wagner 方法分别求得静态带宽  $b$  和  $b_1$  下的最优程序分割方案  $(C_b, S_b)$  和  $(C_{b_1}, S_{b_1})$ , 并根据  $(C_{b_1}, S_{b_1})$  对应的优化模型最优值设定带宽  $b_1$  下优化模型值的上限值  $W_{b_1}$ 。MCGAP 方法中第 5-6 行的 Adjust 函数根据 Stoer-Wagner 方法所求得的  $C_b$  和  $S_b$  调整程序的加权对象关系图, 并确定待分割节点  $V'$ ,  $V' = V - C_b - S_b$ , 即当带宽在  $b_1$  和  $b$  间变化时只需决策  $V'$  中节点的分割, 以减小计算量。第 7-18 行即基于最小割求解程序的最优分割方案, 其中第 8 行 MA\_Order 函数采用最小割 Stoer-Wagner 方法中的最大邻接值排序方法对节点进行排序; 第 9-11 行基于贪心策略确定当前选择的分割方案  $(C, S)$ ,  $C$  和  $S$  分别表示当前分割中运行在本地移动设备和服务器的节点集合, 并计算此方案在带宽  $b$  和  $b_1$  下对应的优化模型值  $W_b(C, S)$  和  $W_{b_1}(C, S)$ ; 第 12-15 行根据与 BBAP 中相同的两个约束条件:  $W_{b_1}(C, S) < W_{b_1}$  和  $W_b(C, S) < \text{minValue}$ , 其中  $W_{b_1}$  表示带宽  $b_1$  下程序分割优化模型的上限值, 更新当前最优分割方案  $(C_{\text{min}}, S_{\text{min}})$ ; 第 16-17 行 Contract 函数是将  $v_{m-1}$  和  $v_m$  两个节点合并, 并从节点集合  $V'$  中删除节点  $v_m$ , 其中  $A_3 = \text{Contract}(A_1, A_2)$  具体表示删除两个节点  $A_1$  和  $A_2$ , 生成一个新节点  $A_3$ , 并对于图中其他任意节点  $v$ , 建立新边  $e = (A_3, v)$ , 更新其权重:  $w(A_3, v) = w(A_1, v) + w(A_2, v)$ 。最后 MCGAP 方法的第 19 行返回程序的最终最优分割方案  $(C_{\text{min}}, S_{\text{min}})$  和优化模型对应的最优值  $\text{minValue}$ 。

[0070] 步骤 4, 分布式执行程序。

[0071] 最后根据程序分割方法解得的最优分割方案, 实现程序在移动设备和服务器间的代码迁移和分布式执行, 可以有效降低应用程序的执行时间和移动设备的耗电量, 延长移动设备的电池使用寿命。

[0072] 下面将结合附图说明本发明提出的基于带宽自适应程序分割的移动设备节能方法相比其他方法的改进效果。首先为了分析本发明的加权对象关系图构造方法性能, 实施平台采用内存为 3.0GB, CPU 频率为 2.53GHz, 运行 Ubuntu 10.10 操作系统的 ThinkPad X201i 笔记本。利用 Java 字节码分析与优化框架 Soot 工具和字节码重写技术, 针对 Dacapo Benchmarks 程序, 将本发明的加权对象关系图构造方法 (CSDPAC) 与纯静态分析方法 (PSPAC)、纯动态分析方法 (PDPAC) 进行对比。图 7 和图 8 都是执行 8 个示例程序 (Programs) 来进行对比, 分别是: avrora, h2, luindex, lusearch, pmd, sunflow, tomcat 和 xalan。

[0073] 图 7 表示三种方法所获取对象关系图的大小比较。从图 7 可以看出, 与 PDPAC 相比, CSDPAC 与 PSPAC 所构造的加权对象关系图规模较小。这是由于 CSDPAC 与 PSPAC 均利用程序的静态信息删除系统库对象, 对初始程序对象关系图进行裁剪, 而 PDPAC 记录了包括系统库对象及其关系等所有运行时信息, 对象关系图规模较大。

[0074] 图 8 表示程序实际运行时间与三种方法下预测运行时间的对比。实验根据程序总字节码数 (Bytecode Instruction Count, 简称 BIC) 来预测程序的运行时间。预测运行时间 (Predicted Execution Time, 简称 PET) 与实际运行时间 (Actual Execution Time, 简称 AET) 越接近, 表明所构造对象关系图越准确。从图 8 可以看出, CSDPAC 和 PDPAC 的预测运行时间都比较接近于程序实际运行时间, 表明动态分析能够获得较为准确的程序运行时信息。而 PSPAC 只采用静态程序分析, 无法精确获取程序运行时信息, 因此预测运行时间偏小, 表明 PSPAC 所获取对象关系图不够准确。由图 7 和图 8 可以看出, CSDPAC 结合了 PDPAC

和 PSPAC 的优点,既能获得较准确的程序信息,又能避免对象关系图规模较大,减轻移动应用程序分割的计算负担。

[0075] 为评价本发明的 BBAP 与 MCGAP 方法的性能,实施平台采用内存为 3.0GB, CPU 频率为 2.53GHz, 运行 Windows 7 操作系统的 ThinkPad X201i 笔记本。具体实施参数设置如图 9 所示,图 10 为应用程序实例对应的对象关系图中节点个数和边个数。仿真实验针对程序分割模型的三种特殊情况:(1)  $w_t = 1, w_e = 0$  (运行时间最优);(2)  $w_t = 0, w_e = 1$  (能量最优);(3)  $w_t = 0.5, w_e = 0.5$ , 分别实施如下步骤:

[0076] (1) 用 Stoer-Wagner 方法和 BBAP/MCGAP 方法分别求解带宽 100kb/s 下的最优分割  $Cut(W, S\_W, 100)$  和  $Cut(W, BBAP, 100)$  (或  $Cut(W, MCGAP, 100)$ ); 其中  $S\_W$  是 Stoer-Wagner 方法的简称,  $W$  表示优化模型;

[0077] (2) 分别计算以上两个分割  $Cut(W, S\_W, 100)$  与  $Cut(W, BBAP, 100)$  (或  $Cut(W, MCGAP, 100)$ ) 在不同带宽  $b$  下对应的模型值  $W\{Cut(W, S\_W, 100), b\}$ 、 $W\{Cut(W, BBAP, 100), b\}$  (或  $W\{Cut(W, MCGAP, 100), b\}$ );

[0078] (3) 用 Stoer-Wagner 方法依次计算不同带宽  $b$  下对应的模型最优值  $Wmin(b)$ ;

[0079] (4) 计算程序全部在移动设备上执行对应的模型值  $WLocal$ 。

[0080] 特殊地,为了更加直观,对于 (1)  $w_t = 1, w_e = 0$  (运行时间最优) 和 (2)  $w_t = 0, w_e = 1$  (能量最优) 两种情况,实验结果将模型值等价转化为程序运行时间和耗电量,图中符号表示分别用 ‘T’ (Time) 和 ‘E’ (Energy) 代替相应的 ‘W’ (Weight), 即  $W\{Cut(W, S\_W, 100), b\}$ 、 $W\{Cut(W, BBAP, 100), b\}$  (或  $W\{Cut(W, MCGAP, 100), b\}$ )、 $Wmin(b)$  和  $WLocal$  在 (1)  $w_t = 1, w_e = 0$  情况下分别对应表示为  $T\{Cut(T, S\_W, 100), b\}$ 、 $T\{Cut(T, BBAP, 100), b\}$  (或  $T\{Cut(T, MCGAP, 100), b\}$ )、 $Tmin(b)$  和  $TLocal$ , 在 (2)  $w_t = 0, w_e = 1$  情况下分别对应表示为  $E\{Cut(E, S\_W, 100), b\}$ 、 $E\{Cut(E, BBAP, 100), b\}$  (或  $E\{Cut(E, MCGAP, 100), b\}$ )、 $Emin(b)$  和  $ELocal$ 。

[0081] 图 11 表示 BBAP 方法对目标模型三种特殊情况下的求解时间。图中 Model\_T、Model\_E 和 Model\_W 分别对应 (1)  $w_t = 1$  和  $w_e = 0$ ; (2)  $w_t = 0$  和  $w_e = 1$ ; (3)  $w_t = 0.5$  和  $w_e = 0.5$  三种情况。可以看出,对于节点个数较少的图, BBAP 能够在较短时间内求出最优解;对于节点个数较多的图,其求解时间较长。这说明 BBAP 适用于规模较小的图,而对于节点数较多的图,如对于 lusearch 程序, BBAP 的计算开销明显增大。

[0082] 图 12、图 13 和图 14 分别表示为 BBAP 与其他方法在带宽变化情况下对应的程序运行时间 ( $w_t = 1, w_e = 0$ )、耗电量 ( $w_t = 0, w_e = 1$ ) 和最优模型值 ( $w_t = 0.5, w_e = 0.5$ ) 的性能对比图。图 12 ~ 14 中分别给出了在四个程序的运行对比,四个程序分别是:(A) RandomGraph1 程序, (B) RandomGraph2 程序, (C) lusearch 程序, (D) avrora 程序。在图 12 中,  $T\{Cut(T, BBAP, 100), b\}$  和  $T\{Cut(T, S\_W, 100), b\}$  分别表示带宽为 100kb/s 时用 BBAP 和 Stoer-Wagner 方法得到的最优分割方案  $Cut(T, BBAP, 100)$  和  $Cut(T, S\_W, 100)$  在不同带宽  $b$  下对应的程序运行时间。 $Tmin(b)$  表示在每个带宽  $b$  下多次用 Stoer-Wagner 方法求得的最优分割方案对应的最短程序运行时间。 $TLocal$  表示不采用程序分割,即程序全部在移动设备本地执行的时间。可以看出,  $T\{Cut(T, BBAP, 100), b\}$  与  $Tmin(b)$  在多数情况下均小于  $TLocal$ , 而  $T\{Cut(T, S\_W, 100), b\}$  在带宽较大情况下也小于  $TLocal$ , 这说明利用程序分割实现代码迁移,能够有效降低应用程序的运行时间。当带宽大幅度下降时,静态程序最

优分割  $\text{Cut}(T, S_W, 100)$  对应的运行时间  $T\{\text{Cut}(T, S_W, 100), b\}$  急剧增长, 随带宽变化波动较大。而  $T\{\text{Cut}(T, \text{BBAP}, 100), b\}$  较接近于不同带宽下的程序最短运行时间  $T_{\min}(b)$ , 且波动较为平缓。图 13 和图 14 对应的程序耗电量和最优模型值具有类似结论。这说明相比静态 Stoer-Wagner 方法, BBAP 能够较好地适应带宽变化, 减少应用程序运行时间和耗电量; 而与每个不同带宽下均需重新计算分割的动态方法相比, BBAP 仅需分割一次, 降低了分割开销。

[0083] 图 15 为 MCGAP 方法对目标模型三种特殊情况下的求解时间。Model\_T、Model\_E 和 Model\_W 分别对应 (1)  $w_t = 1$  和  $w_e = 0$  (运行时间最优); (2)  $w_t = 0$  和  $w_e = 1$  (能量最优); (3)  $w_t = 0.5$  和  $w_e = 0.5$ 。与图 11 比较, 可以看出, 图节点个数较少时 MCGAP 方法的求解时间与 BBAP 方法相差不大; 而节点个数较多时 MCGAP 方法的求解时间明显减少。

[0084] 图 16 为 MCGAP 方法与 BBAP 方法在不同带宽下求解的程序最短运行时间对比图。其中  $b$  表示带宽,  $B$ 、 $M$  和  $P$  分别表示 BBAP、MCGAP 方法和 MCGAP 近似解的精确度。设  $\bar{X}$  为 BBAP 解得的模型最优解,  $X$  为采用 MCGAP 方法求得的近似最优解, 则  $W(X, b)/W(\bar{X}, b)$  为 MCGAP 方法近似解的精确度。从图 16 可以看出, MCGAP 求解的最短程序运行时间较接近于 BBAP 对应的最优解, 精确度较高, 可用于较快地求得规模较大对象关系图的近似最优解。

[0085] 图 17、图 18 和图 19 分别为 MCGAP 与其他方法在带宽变化情况下对应的程序运行时间 ( $w_t = 1, w_e = 0$ )、耗电量 ( $w_t = 0, w_e = 1$ ) 和最优模型值 ( $w_t = 0.5, w_e = 0.5$ ) 的性能对比图。图 17 ~ 19 中分别给出了在四个程序的运行对比, 四个程序分别是: (a) avrora 程序, (b) lusearch 程序, (c) luindex 程序, (d) h2 程序。由图 17, 图 18 和图 19 可以看出, 与程序全部运行在移动设备相比, 利用程序分割和代码迁移可以降低移动设备的耗电量, 缩短应用程序的运行时间; 而与静态程序分割 Stoer-Wagner 方法相比, MCGAP 对应的模型目标值较接近于不同带宽下的最小值, 随带宽波动较为平缓。可见, 随着环境中带宽的变化, MCGAP 能够避免模型目标值因带宽降低而急剧增加, 有效缩短程序运行时间, 降低移动设备的耗电量。虽然图中动态方法的性能较好, 但 BBAP 和 MCGAP 均避免了动态方法在不同带宽下多次计算分割的较大开销。如果考虑动态方法多次分割的计算开销, 其性能将明显下降。

[0086] 图 20 为 BBAP 与 MCGAP 方法的平均性能对比图。TR1 和 ER1 分别表示与采用 Stoer-Wagner 方法相比, 采用 BBAP/MCGAP 方法时程序平均运行时间和耗电量降低的百分比; TR2 和 ER2 分别表示与程序全部运行在移动设备相比, 采用 BBAP 或 MCGAP 方法时程序平均运行时间和耗电量降低的百分比。可以看出, 与程序全部运行在移动设备相比, 采用 BBAP 方法对应的程序平均运行时间和耗电量分别降低了 44.17% 和 34.47%, 采用 MCGAP 方法的程序平均运行时间和耗电量分别降低了 37.44% 和 46.09%; 而与静态程序分割 Stoer-Wagner 方法相比, BBAP 对应的平均运行时间和耗电量分别降低了 26.38% 和 25.88%, MCGAP 对应的程序平均运行时间和耗电量分别降低了 26.99% 和 30.58%。

[0087] 通过以上具体实施实验验证, 可以得出, 在带宽不断变化的情况下, 本发明的基于带宽自适应程序分割和代码迁移的移动设备节能方法, 可以有效降低应用程序的执行时间和移动设备的耗电量, 从而延长移动设备的电池使用寿命。

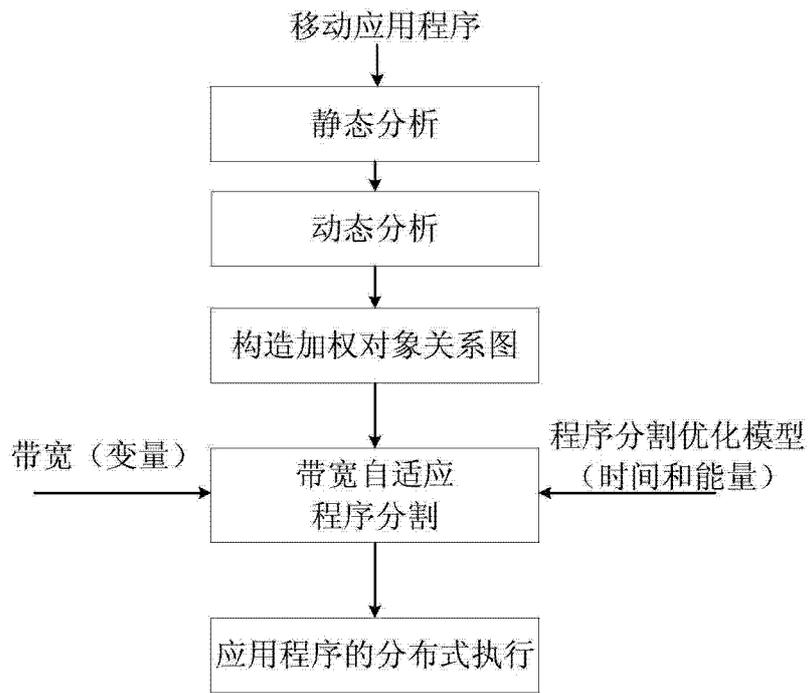


图 1

```
public class FacePreview{
    public static void main(String args[]){
        ImageCapture imageCapture = new ImageCapture(80,80);
        FaceDetection faceDetection1 = new FaceDetection();
        FaceDetection faceDetection2 = new FaceDetection();
        byte[] image = new type[9066];
        ImageCapture.CaptureImage(image);
        for(int i=0; i<10; i++){
            face1 = faceDetection1.DetectFace(image);
        }
        for(int i=0; i<5; i++){
            face2 = faceDetection2.DetectFace(image);
        }
    }
}
class ImageCapture{
    public void CaptureImage(byte[] image){
        .....
    }
}
class FaceDetection{
    public void DetectFace(byte[] image){
        byte[] imageblock = new type[16];
        FaceDetectionLib.ProcessBlock(imageblock);
    }
}
class FaceDetectionLib{
    public static void ProcessBlock(byte[] imageblock){
        .....
    }
}
```

图 2

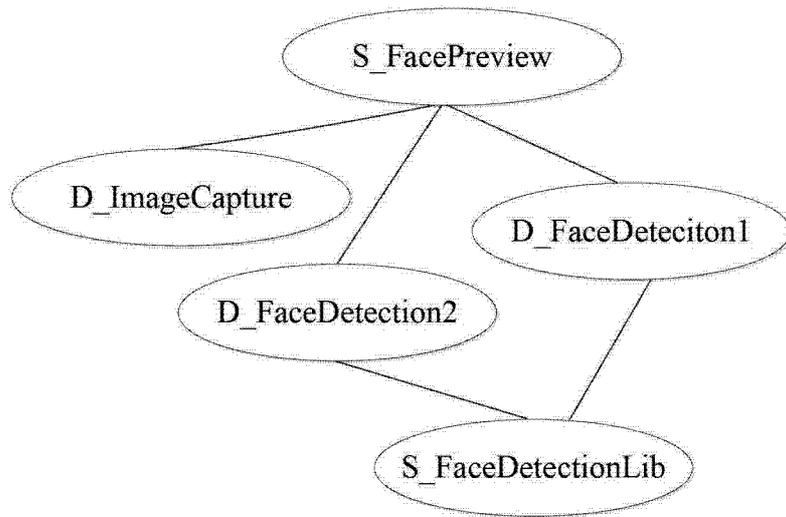


图 3

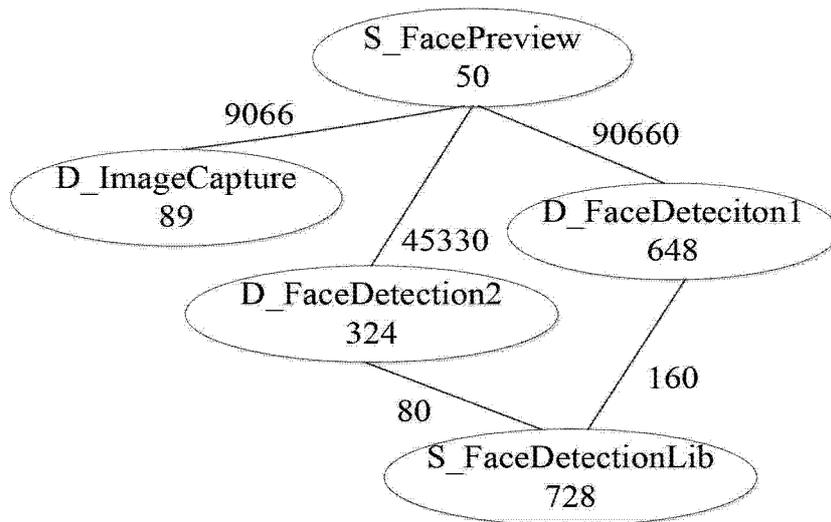


图 4

**算法 1. BBAP**

输入:  $WORG=(V, E, W_v, W_e), b, b_l, N_L, a$

输出:  $Xmin, minValue$

符号定义:

$a$ : coefficient limiting model value upper bound  $minV$

$minV$ : model value upper bound under bandwidth  $b_l$

$N_L$ : node set which must run locally on mobile devices

$Xmin$ : the optimal partitioning of BBAP

$minValue$ : the model value with partitioning  $Xmin$

$minV_{SW}$ : optimal model value under bandwidth  $b_l$

```

1   compute  $minV_{SW}$  using Stoer-Wagner
2    $minV = minV_{SW} * (1+a)$ 
3    $minValue = MAXINT$ 
4   for all  $v_i$  in  $V$ 
5       if  $v_i$  in  $N_L$  then  $x_i = 1$ 
6       else  $x_i = -1$ 
7       end if
8   end for
9   BBSearch(1,  $minV, WORG, Xmin, minValue, b$ )
10  return  $\{Xmin, minValue\}$ 

```

图 5

**算法 2. MCGAP**

输入:  $WORG = \{V, E, W_v, W_e\}, b, b_l, a$

输出:  $(C_{min}, S_{min}), minValue$

符号定义:

$(C_{min}, S_{min})$ : the optimal partitioning of MCGAP

$minValue$ : model value with partitioning  $(C_{min}, S_{min})$

$(C_b, S_b)$ : the optimal partitioning under bandwidth  $b$

$(C_{b_l}, S_{b_l})$ : the optimal partitioning under bandwidth  $b_l$

$W_{b_l}(C_{b_l}, S_{b_l})$ : model value with  $(C_{b_l}, S_{b_l})$  under  $b_l$

$W_{b_l}$ : the upper bound of model value under  $b_l$

$W_b(C, S)$ : model value with partitioning  $(C, S)$  under  $b$

```

1   $minValue = MAXINT$ 
2  compute  $(C_b, S_b)$  using Stoer-Wagner
3  compute  $(C_{b_l}, S_{b_l}), W_{b_l}(C_{b_l}, S_{b_l})$  using Stoer-Wagner
4   $W_{b_l} = W_{b_l}(C_{b_l}, S_{b_l}) * (1+a)$ 
5   $G = (V_G, E_G) = Adjust(WORG, C_b, S_b)$ 
6   $V' = V_G$ 
7  while  $(m = |V'|) > 1$ 
8       $\{v_1, v_2, v_3, \dots, v_m\} = MA\_Order(V', G)$ 
9       $S = \{v_m\} \cup S_{b_l}$ 
10      $C = C_b \cup \{v_1, v_2, \dots, v_{m-1}\}$ 
11     Compute  $W_{b_l}(C, S)$  and  $W_b(C, S)$ 
12     if  $(W_{b_l}(C, S) < W_{b_l} \ \&\& \ W_b(C, S) < minValue)$  then
13          $minValue = W_b(C, S)$ 
14          $C_{min} = C, S_{min} = S$ 
15     end if
16      $v_{m-1} = Contract(v_{m-1}, v_m)$ 
17      $V' = V' - v_m$ 
18 end while
19 return  $(C_{min}, S_{min}), minValue$ 

```

图 6

Programs	Classes	Methods	CSDPAC		PSPAC		PDPAC	
			Nodes	Edges	Nodes	Edges	Nodes	Edges
avroa	8812	22240	200	5430	208	5430	10098	25289
h2	10264	14494	4900	11088	4912	11088	13257	16237
luindex	6530	8199	1027	6199	1027	6199	6974	10299
lusearch	7812	10148	1238	7148	1238	7148	7976	10278
pmd	12673	17360	2312	9360	2320	9360	13879	18002
sunflow	9796	12506	2615	8506	2615	8506	9820	12623
tomcat	9760	12446	2718	8246	2718	8246	9772	12453
xalan	11074	14933	1896	9347	1896	9347	11076	14935

图 7

Programs	AET(ms)	CSDPAC		PSPAC		PDPAC	
		BIC(byte)	PET(ms)	BIC(byte)	PET(ms)	BIC(byte)	PET(ms)
avroa	6414	360846	6014	325800	5430	390780	6513
h2	34964	2054820	34247	665280	11088	2146866	35781
luindex	7266	421082	7018	371940	6199	454983	7583
lusearch	1887	115208	1920	52560	876	118505	1975
pmd	5125	323465	5391	230400	3840	321660	5361
sunflow	9733	580381	9673	404460	6741	617584	10293
tomcat	7350	443164	7386	319260	5321	454980	7583
xalan	1443	76680	1278	31380	523	98826	1647

图 8

Parameter	Variable	Value
CPU 功率(w)	$P_{cpu}$	7
Wi-Fi 功率(mw)	$P_w$	88
当前带宽(kb/s)	$b$	100
计算速度比例	$k$	3

图 9

Graph Name	Nodes	Edges
RandomGraph1	15	60
RandomGraph2	30	350
RandomGraph3	50	948
RandomGraph4	100	3238
avroa	200	5430
h2	4900	11088
luindex	1027	6199
lusearch	1238	7148

图 10

Graph Name	Model_T (s)	Model_E (s)	Model_W (s)
RandomGraph1	0.022	0.032	0.028
RandomGraph2	0.056	0.049	0.058
RandomGraph3	0.375	0.389	0.392
RandomGraph4	0.897	1.003	0.974
avrora	2.062	2.232	2.182
lusearch	200	217	209

图 11

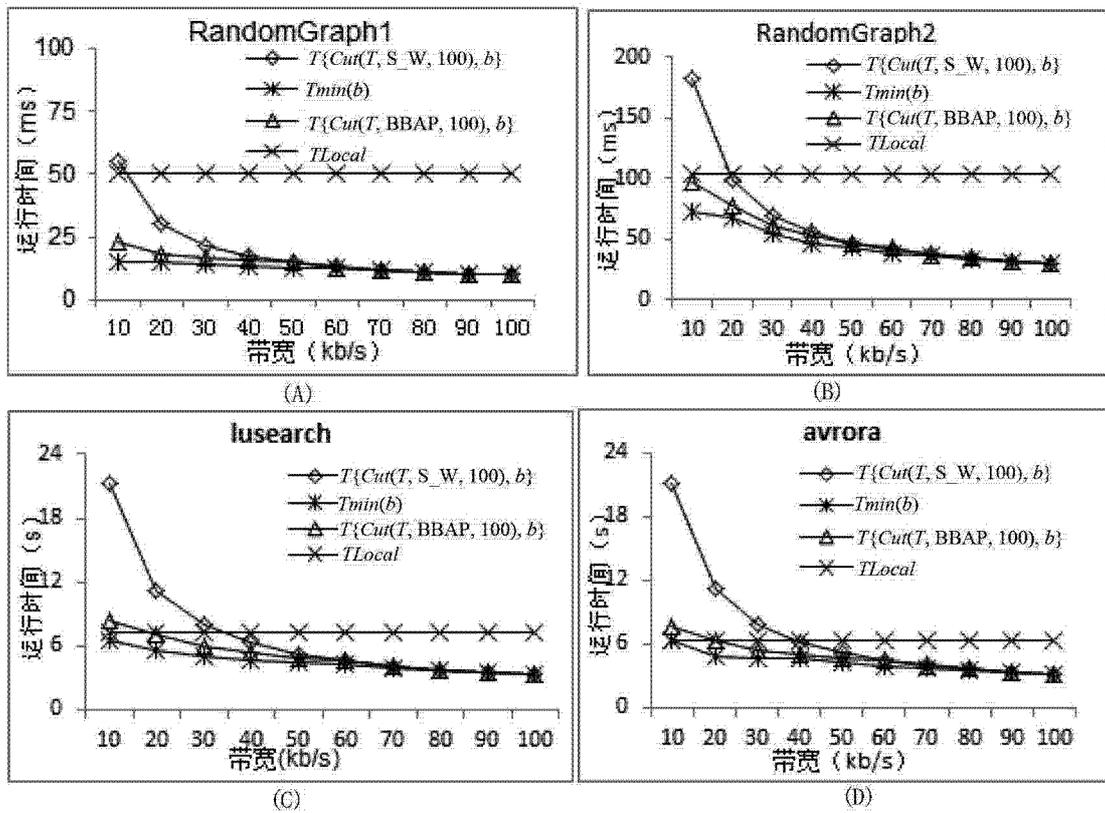


图 12

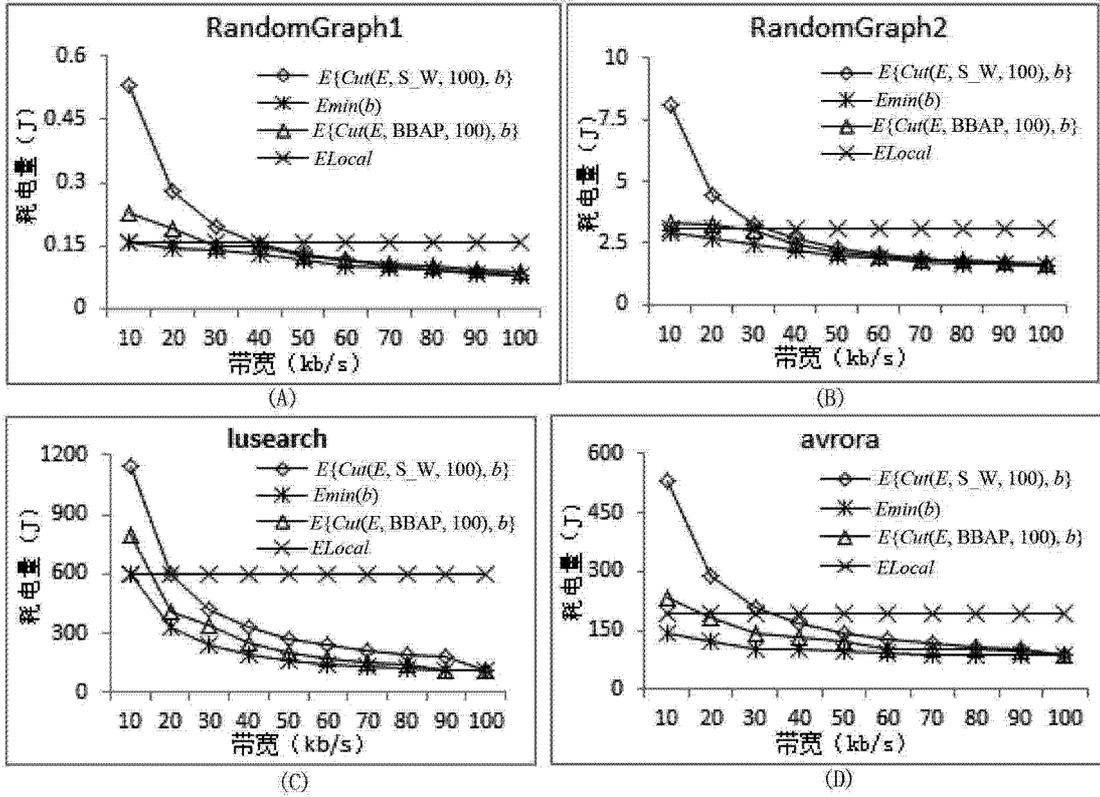


图 13

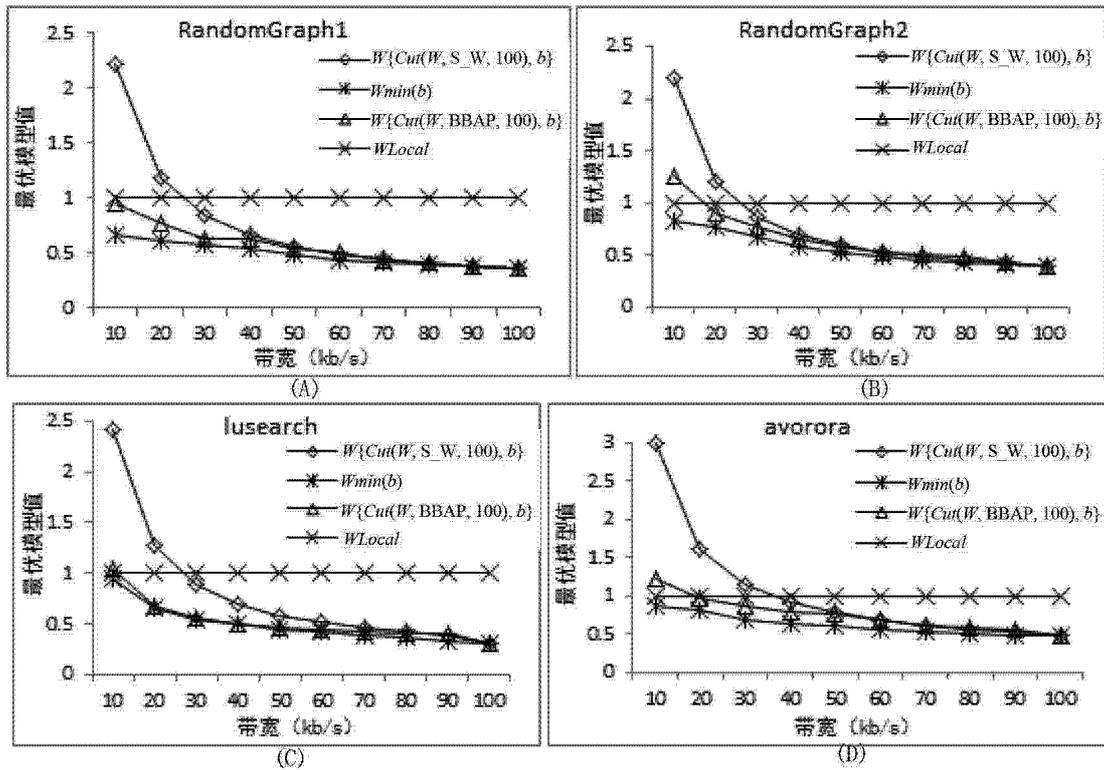


图 14

Graph Name	Model_T (s)	Model_E (s)	Model_W (s)
RandomGraph1	0.019	0.022	0.023
RandomGraph2	0.054	0.051	0.055
RandomGraph3	0.392	0.396	0.398
RandomGraph4	0.913	0.915	0.908
avrora	1.662	1.583	1.592
h2	52.28	51.30	54.29
lusearch	20.52	20.45	20.79
luindex	20.42	20.32	20.61

图 15

Graph Name	$b=20\text{kb/s}$			$b=60\text{kb/s}$			$b=80\text{kb/s}$		
	B(ms)	M(ms)	P	B(ms)	M(ms)	P	B(ms)	M(ms)	P
RandomGraph1	17	17	1.000	12	13	1.083	11	12	1.091
RandomGraph2	71	72	1.014	40	41	1.025	33	35	1.061
RandomGraph3	105	107	1.019	70	72	1.029	62	62	1.000
RandomGraph4	167	181	1.084	113	114	1.009	97	98.6	1.016
avrora	4900	4950	1.010	3712	3716	1.001	3300	3324	1.007
h2	31410	31771	1.011	20112	20143	1.002	18352	18553	1.011
lusearch	5693	5712	1.003	4392	4424	1.007	4087	4213	1.031
luindex	1904	1995	1.048	1119	1135	1.014	1030	1072	1.041

图 16

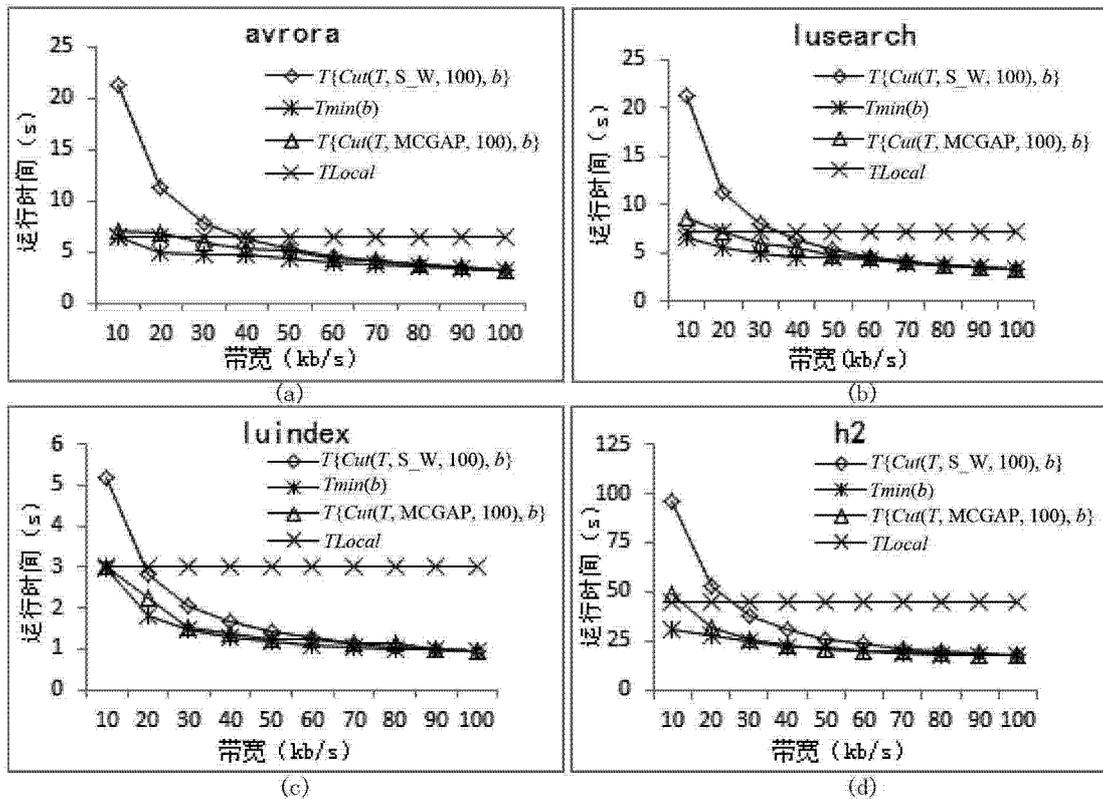


图 17

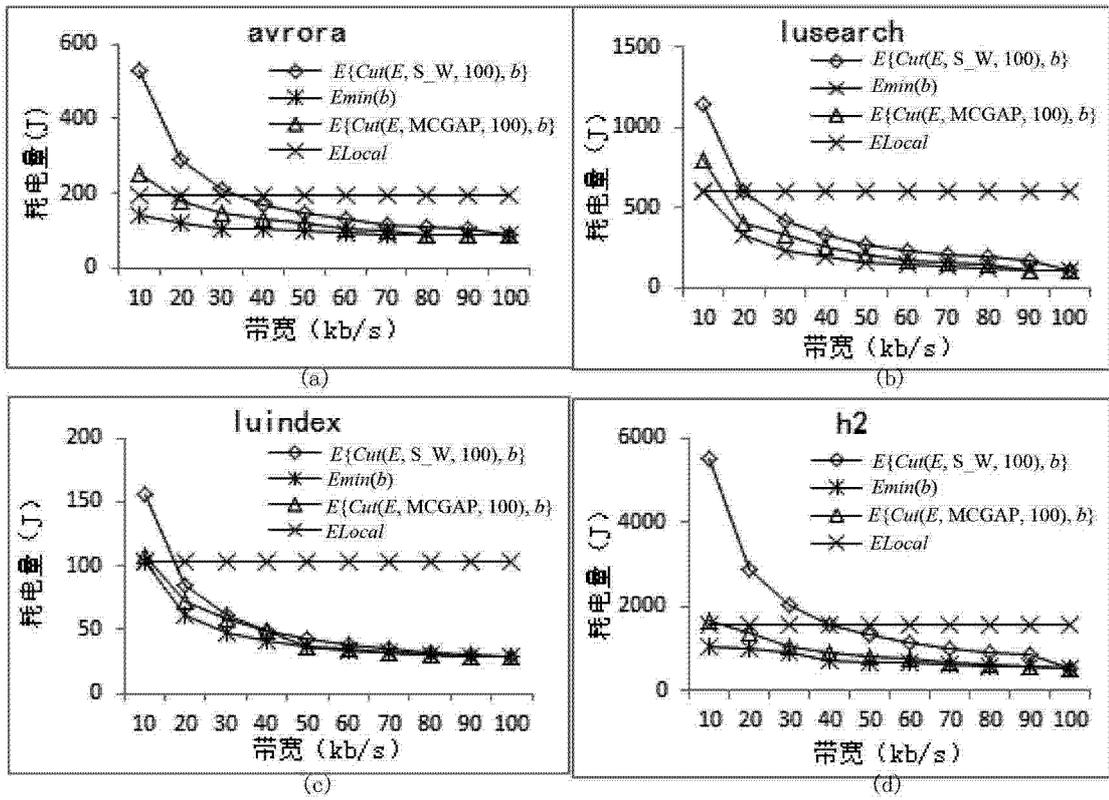


图 18

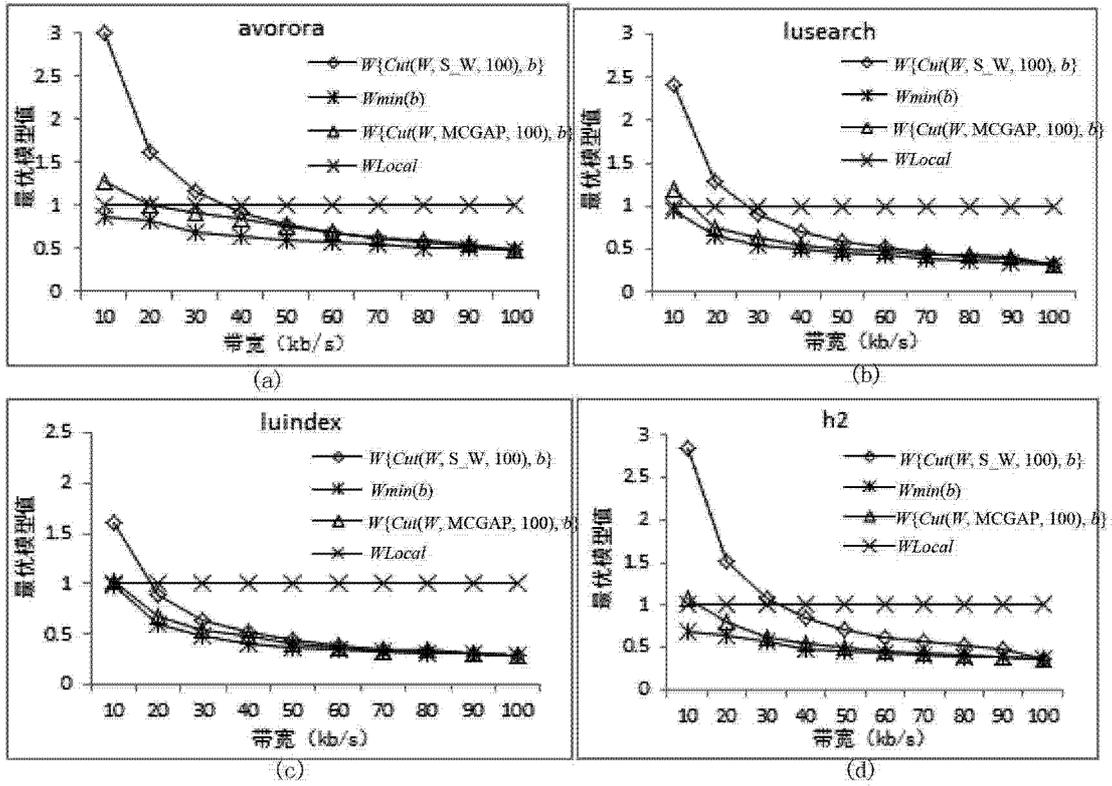


图 19

Algorithm	Execution Time		Energy Consumption	
	TR1(%)	TR2(%)	ER1(%)	ER2(%)
BBAP	26.38	44.17	25.88	34.47
MCGAP	26.99	37.44	30.58	46.09

图 20