(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 102819664 A
(43)申请公布日 2012.12.12

(21)申请号 201210248732.3

(22)申请日 2012.07.18

(71)申请人 中国人民解放军国防科学技术大学
地址 410073 湖南省长沙市开福区德雅路109号

(72)发明人 李姗姗 廖湘科 刘晓东 吴庆波 戴华东 彭绍亮 王蕾 付松龄 鲁晓佩 郑思

(74)专利代理机构 国防科技大学专利服务中心 43202
代理人 郭敏

(51)Int.Cl.
G06F 19/00(2011.01)

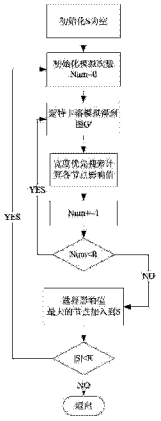
权利要求书 3 页 说明书 6 页 附图 1 页

(54)发明名称

一种基于图形处理单元的影响最大化并行加速方法

(57)摘要

本发明公开了一种基于图形处理单元的影响最大化并行加速方法，目的是提出一种新型基于GPU的影响最大化并行方法，利用GPU的并行计算能力加速算法执行、减少执行时间。其特征在于在每次蒙特卡洛模拟中，首先找到网络图中的强连通分量，将同一强连通分量中的所有节点合并为一个节点，其权重为该强连通分量中各节点权重之和；然后采用自底向上遍历的策略，并行计算各节点的影响值；利用GPU的并行计算能力，由各个GPU计算核心采用各自的线程对不同的节点并行计算影响值，得到K个最有影响的节点。本发明将图转化为有向无环图，可以显著减少影响值的计算量，同时通过最大程度地调度各节点在GPU的计算核心中并行计算，降低了整体运行时间。



CN 102819664 A

1. 一种基于图形处理单元的影响最大化并行加速方法，包括以下步骤：

第一步：初始化影响最大化节点集合 S 为空；

第二步：设定当前蒙特卡洛模拟次数 Num=0；

第三步：采用蒙特卡洛模拟方法对图进行选边，得到图 G′；

其特征在于还包括以下步骤：

第四步：采用 Tarian 算法，基于深度优先搜索寻找图 G′ 中所有的强连通分量 SCC_i，i 取值从 0 到 j-1，i 为图 G′ 中的强连通分量的个数；

第五步：根据图 G′ 的各强连通分量 SCC_i，将图 G′ 转变为有向无环图 G^*，方法是：

5.1：初始化 i=0；

5.2：将强连通分量 SCC_i 用新节点 v_{n+i} 代替，其中 n 是图 G′ 中的节点个数；

5.3：i=i+1，如果 i<j，转 5.2；如果 i ≤ j，执行第六步；

第六步：从出度为 0 的节点开始，自底向上遍历有向无环图 G^* 中所有节点，利用 GPU 不同线程计算所有节点的影响值，线程号为 p 的线程负责计算节点 v_p 的影响值，其中 0 ≤ p ≤ n-1，具体方法是：

6.1：定义和初始化变量，方法是：

6.1.1：使用布尔数组 Visited[] 记录各个节点是否已经被访问过，Visited[v_p] 等于 true 表示节点 v_p 已经被访问，Visited[v_p] 等于 false 表示节点 v_p 未被访问，将数组 Visited[] 全部初始化为 false，表示所有节点均未被访问；

6.1.2：使用整数数组 Count[] 记录各个节点已经被访问的子节点个数，其中 0 ≤ Count[v_x] ≤ outdegree[v_x]，0 ≤ x ≤ n-1，outdegree[v_x] 是节点 v_x 的出度；将数组 Count[] 全部初始化为 0，表示均未被访问；

6.1.3：使用整数数组 Inf[] 记录各个节点的影响值，其中 0 ≤，Inf[v_x] ≤ n，0 ≤ x ≤ n-1，将数组 Inf[] 全部初始化为 0；

6.1.4：使用字符串数组 Label[] 记录各个节点的标签，标签 Label[v_x] 标记了节点 v_x 可能同其他节点发生重叠的位置，其中节点 v_a 及节点 v_b 重叠于节点 v_c 当且仅当从节点 v_a 和 v_b 均存在至少一条路径可达节点 v_c，0 ≤ a，b，c ≤ n-1，将数组 Label[] 全部初始化为 NULL；

6.1.5：使用布尔变量 Stop 记录线程计算是否完成，Stop 等于 true 表示该次模拟中所有节点影响值计算已经完成，Stop 等于 false 表示未完成，Stop 为全局变量，所有 GPU 线程均可以修改其内容，初始化 Stop 为 false；

6.2：如果停止标志 Stop 为 false，转 6.3；如果 Stop 为 true，转第七步；

6.3：GPU 采用单指令流多数据流的执行方式，以多线程并行的方式计算节点的影响值；多线程并行的方式是指：GPU 为每个节点分配一个线程计算影响值，GPU 一次由 v 个线程并行计算 y 个节点的影响值，y 为 GPU 中的流处理器数目，当 GPU 当前 y 个节点的影响值计算完成后，若还有节点影响值未计算，则 GPU 经过 GPU 线程调度以多线程并行的方式计算剩余节点的影响值，直至所有节点的影响值计算完毕，GPU 的一个线程计算节点 v_p 影响值的方法是：

6.3.1：将停止标志 Stop 置为 true；

6.3.2：如果 Visited[v_p] 等于 false，执行 6.3.3；否则说明节点 v_p 已经被访问过，转

6.2；

6.3.3：如果 Count[v_p] 等于节点 v_p 的出度，则说明节点 v_p 的所有子节点均已被访问，执行 6.3.4 计算节点 v_p 的影响值；否则说明节点 v_p 的子节点中仍有未处理的节点，则将停止标志 Stop 置为 false，转 6.2；

6.3.4：计算节点 v_p 的所有子节点影响值的总和 sum，$sum = \sum_{v_q \in out[v_p]} Inf[v_q]$，其中 out[$v_p$] 是节点 v_p 所有子节点的集合；

6.3.5：计算节点 v_p 的标签 Label(v_p)，节点 v_p 的标签 label(v_p) 等于节点 v_p 的所有子节点对 v_p 贡献的并集，即 $Label(v_p) = \cup_{v_q \in out[v_p]} Con(v_q)$，其中 Con($v_q$) 是子节点 v_q 对节点 v_p 的贡献，子节点 v_q 对节点 v_p 的贡献是指：如果子节点 v_q 的入度大于 1，则 v_q 对节点 v_p 的贡献为节点 v_q 自身，即 Con(v_q) = v_q；如果子节点 v_q 的入度小于等于 1，v_q 的贡献为节点 v_q 的标签，即 Con(v_q) = Label(v_q)；

6.3.6：计算节点 v_p 所有子节点的集合 out[v_p] 重叠的影响值 Overlap(out[v_p])，方法是：

6.3.6.1：初始化 Overlap(out[v_p]) 为 0，初始化重叠范围集合 Range 为 out[v_p]；

6.3.6.2：对于任意节点 $v_a \in$ Range，如果存在节点 $v_b \in$ Range 且 $v_b \neq v_a$，并且从节点 v_a 存在路径可达节点 v_b，此时重叠发生在节点 v_b，故 Overlap(out[v_p]) = Overlap(out[v_p])+Inf[v_b]，同时将 v_b 从 Range 中删除，即 Range = Ranqe-v_b；

6.3.6.3：使用字符串数组Extra[]记录Range中的重叠项，将Extra[]初始化为空集∅；使用字符串数组Filter[]记录Range中除去重叠项剩余的单一项，将Filter[]初始化为空集∅；对于任意节点 $v_a \in$ range，对于任意元素 u ∈ Label(v_a)，如果元素 u 已经属于 Filter，则元素 u 为重叠项，将 u 加入 Extra 中并将其影响值 Inf[u] 加入到 Overlap(out[v_p])，即 Extra = Extra ∪ u，Overlap(out[v_p]) = Overlap(out[v_p])+Inf[u]；如果 u 不属于 Filter，则将 u 加入到 Filter 数组中，即 Filter = Filter ∪ u；

6.3.6.4：由于Extra[]和Filter[]数组中的元素仍然可能存在重复，故最终节点 v_p 所有子节点的重叠影响值 Overlap(out[v_p]) 需要加上两者重叠值之差，即 Overlap(out[v_p]) = Overlap(out[v_p])+(Overlap(Filter)-Overlap(Extra))；

6.3.7：计算节点 v_p 的影响值 Inf[v_p]，Inf[v_p]=sum+weight(v_p)-overlap(out[v_p])，其中 weight(v_p) 是节点 v_p 的权重；Totallnf[v_p]=Totallnf[v_p]+Inf[v_p]，其中 Totallnf[v_p] 是 R 次蒙特卡洛模拟节点 v_p 的总影响值；R 是总模拟次数，R 为正整数；

6.3.8：如果节点 v_p 无父节点，则转 6.3.8；否则，对于节点 v_p 的任意父节点 v_s，将其已访问子节点个数 Count[v_s] 加 1，即 Count[v_s]=Count[v_s]+1，并将停止标志 Stop 置为 false；

6.3.9：将节点 v_p 标记为已经被访问，即 Visited[v_p]=true，转 6.2；

第七步：将蒙特卡洛模拟次数 Num 加 1，判断 Num 是否小于 R，如果 Num<R，转第三步，否则执行第八步；

第八步：对集合 V—S 中所有节点，选择 Totallnf[] 最大的节点 v 加入到集合 S 中；

第九步：如果集合 S 的节点个数 |S|<K，转第二步，否则说明已经选定 K 个最有影响的节点，结束。

2. 如权利要求 1 所述的一种基于图形处理单元的影响最大化并行加速方法，其特征在

于所述第五步中将强连通分量 SCC_i 用新节点 v_{n+i} 代替的方法是：

5.2.1：对于强连通分量 SCC_i，新增节点 v_{n+i}，节点 v_{n+i} 的入边集合置为 SCC_i 中所有节点入边集合的并集，出边集合为 SCC_i 中所有节点出边集合的并集，权重为该强连通分量中各节点权重之和；

5.2.2：将强连通分量 SCC_i 中所有节点的入边集合和出边集合置空，权重置零，方法是：

5.2.2.1：初始化整数变量 l 为 0；

5.2.2.2：对于强连通分量 SCC_i 中节点 v_l，将节点 v_l 的入边集合和出边集合置为空集∅，权重置为 0；

5.2.2.3：l=l+1；如果 $l<n_i$，其中 n_i 是强连通分量 SCC_i 的节点个数，则转 5.2.2.2；如果 $l \geqslant n_i$，结束。

3. 如权利要求 1 所述的一种基于图形处理单元的影响最大化并行加速方法，其特征在于所述总模拟次数 R 为 20000。

一种基于图形处理单元的影响最大化并行加速方法

技术领域

[0001] 本发明涉及海量数据挖掘领域中社会网络影响最大化问题的解决方法，尤其指针对大规模社会网络的海量用户挖掘，提出的一种基于图形处理单元 GPU 的并行加速方法。

背景技术

[0002] Web2.0 技术的快速发展推动了社会媒体的蓬勃发展。各类社交网站不断涌现，例如国外的 Facebook、Twitter 以及国内的人人网、新浪微博等网站用户数量增长十分迅速，当前 Facebook 的活跃用户已经超过了 8.5 亿。社交网站不仅是人们沟通和交流的桥梁，同时还成了信息传播和扩散的重要媒介。研究表明，68% 的顾客会在购买产品之前询问其家人、朋友的意见。病毒式营销(Viral Marketing)正是利用了用户之间口碑传播的原理，进行品牌推广等网络传销方法。而且随着社会网络用户的持续快速增长，病毒式营销已经成为一种十分高效的信息传播方式。

[0003] 影响最大化问题是社会网络分析中关于影响传播的经典问题。设想如下场景：一家公司要进行新产品推广，其推广策略是：选择 K 名顾客免费试用新产品，之后利用这 K 名顾客对产品的宣传推广和影响传播吸引更多的顾客购买新产品，从而达到利益最优的目的。影响最大化问题可以形式化描述为：对于社会网络图 G = (V, E, W)，其中 V = $\{v_0, v_1, \ldots, v_{n-1}\}$ 是节点集合，V 中节点个数为 n；E 是节点集合 V 中节点之间的有向边集合，即 $E \subset V \times V$，E 中有向边的条数为 m；W 是 G 中节点权重的集合，表征了各节点的影响力(初始值设定为 1，即仅能影响节点自身)。给定网络图 G 和初始活跃节点集合中的节点个数 K，影响最大化问题是从节点集合 V 中选择最佳的 K 个节点作为初始活跃节点集合 S，通过影响传递，使得影响扩散的最终范围最大。影响最大化问题的核心在于如何定位网络中最有影响力的 K 名成员，即网络中的意见领袖，从而通过病毒式营销使得最终被影响的用户数目最大。影响最大化问题的研究不仅对市场营销有着十分重要的现实意义，同时还对舆情预警、疫情发现等方面有着十分重要的应用。自从 Pedro Domingos 和 Matt Richardson 于 2001 年 ACM SIGKDD 会议公布的文章 Mining the network value ofcustomers 中提出影响最大化问题后，该问题受到了越来越多研究者的关注。David Kempe 等人在 2003 年 ACMSIGKDD 会议公布的文章 Maximizing tte Spread of Influence through a Social Network 中证明了影响最大化问题隶属于 NP-Hard 问题，并且提出了一种爬山贪心算法来获得近似最优解。虽然爬山贪心算法可以达到 1-1 / e 的最优逼近(e 是自然对数底)，但是由于 David Kempe 采用多次的蒙特卡洛模拟(例如 20000 次)来计算各个节点的影响值，因此需要消耗大量时间，而且无法扩展应用到大规模的网络中。

[0004] 很多研究人员都致力于设计新的方法来解决影响最大化的效率问题。爬山贪心算法中的核心问题在于需要多次蒙特卡洛模拟以计算所有节点的影响值。为了解决该问题，Jure Leskovec 等人在 ACM SIGKDD2007 中公布的文章 Cost-effective Outbreak Detection in Networks 中根据影响扩散函数的半模特性设计了新的优化方法 CELF，可以很大程度地降低蒙特卡洛模拟的计算量，从而减少了计算时间。之后，Wei Chen 等人在 ACM

SIGKDD2009 中公布文章 Efficient Influence Maximization in Social Networks，文章中提出了目前最优的贪心算法 MixGreedy。该算法的改进在于在每次蒙特卡洛模拟时为网络中所有节点计算影响值，因而进一步降低了算法的复杂度。同时 MixGreedy 整合了 CELF 算法，大大降低了算法执行时间。然而由于影响最大化计算复杂度很高，即使目前最优的 MixGreedy 算法在处理大规模社会网络时仍然十分耗时；例如从 37154 个社会网络节点中选择 50 个最有影响用户就需要 2 个小时以上。因此，如何从大规模社会网络海量用户中快速挖掘最有影响用户成为了亟待解决的问题。

[0005] 另一方面，图形处理单元(Graphics Processing Unit，GPU)的多核多线程高带宽的体系结构使得 GPU 具有超强的并行计算能力，被广泛应用于通用计算中。许多图论算法，例如宽度优先搜索、最小生成树等，都可以利用 GPU 的并行能力加速执行。如何充分利用 GPU 的并行计算能力，挖掘影响最大化问题的并发执行潜力，设计出基于 GPU 体系结构的影响最大化并行加速方法是解决大规模社会网络中影响最大化问题的可行方案。

[0006] 综上所述，影响最大化问题的效率问题是社会网络分析中广泛关注的问题，目前的计算方法无法在合理的时间内准确定位出最有影响力用户，并且具有很差的可扩展性，无法适用于大规模社会网络。因此，研究高效并且具有良好扩展性的影响最大化解决方法是本领域技术人员极为关注的技术问题。现有的影响最大化问题研究中没有公开文献涉及利用 GPU 的并行计算能力来减少运行时间的方法。

发明内容

[0007] 本发明要解决的技术问题是：针对社会网络中的影响最大化问题，提出一种新型基于 GPU 的影响最大化并行方法，充分挖掘贪心算法中的可并行部分并利用 GPU 的并行计算能力，以达到加速算法执行、减少执行时间的目的。

[0008] 为了解决上述技术问题，本发明技术方案是：在每次蒙特卡洛模拟中，首先找到网络图中的强连通分量，由于同一个强连通分量中各个节点的影响值相同，故将同一强连通分量中的所有节点合并为一个节点，其权重为该强连通分量中各节点权重之和；然后采用自底向上遍历的策略，并行计算各节点的影响值。利用 GPU 的并行计算能力，由各个 GPU 计算核心采用各自的线程对不同的节点并行计算影响值。通过最大程度地调度各节点在 GPU 的计算核心中并行计算，降低整体运行时间。

[0009] 具体的技术方案是：

[0010] 第一步：初始化影响最大化节点集合 S 为空。

[0011] 第二步：设定当前蒙特卡洛模拟次数 Num ＝ 0。

[0012] 第三步：采用 Wei Chen 等人在 ACM SIGKDD2009 公布的文章 Efficient Influence Maximization in Social Networks 中的蒙特卡洛模拟方法对图进行选边，得到图 G′ 。

[0013] 第四步：寻找图 G′ 中的强连通分量。在有向图中，如果两个节点 v_e 和 v_f 间既存在一条从 v_e 到 v_f 的有向路径，同时又存在一条从 v_f 到 v_e 的有向路径，则称 v_e 和 v_f 强连通。如果有向图中每两个节点都强连通，则该图是一个强连通图。采用 Robert Tarian 等于 1972 年 SIAM Journal on Computing 杂志公布的文章 Depth-first search and linear graph algorithm 中提出的 Tarjan 算法，基于深度优先搜索寻找图 G′ 中所有的强连通分量 SCC_i，i 取值从 0 到 j-1，j 为图 G′ 中的强连通分量的个数。

[0014] 第五步：根据图 G′ 的各强连通分量 SCC_i，将图 G′ 转变为有向无环图 G^*，方法是：

[0015] 5.1：初始化 i = 0。

[0016] 5.2：将强连通分量 SCC_i 用新节点 v_{n+i} 代替，其中 n 是图 G′ 中的节点个数。

[0017] 具体方法是：

[0018] 5.2.1：对于强连通分量 SCC_i，新增节点 v_{n+i}。节点 v_{n+i} 的入边集合置为 SCC_i 中所有节点入边集合的并集，出边集合为 SCC_i 中所有节点出边集合的并集，权重为该强连通分量中各节点权重之和。

[0019] 5.2.2：将强连通分量 SCC_i 中所有节点的入边集合和出边集合置空，权重置零。方法是：

[0020] 5.2.2.1：初始化整数变量 l 为 0。

[0021] 5.2.2.2：对于强连通分量 SCC_i 中节点 v_l，将节点 v_l 的入边集合和出边集合置为空集∅，权重置为 0。

[0022] 5.2.2.3：l = l+1。如果 $l<n_i$，其中 n_i 是强连通分量 SCC_i 的节点个数，则转 5.2.2.2。如果 $l \geqslant n_i$，转 5.3。

[0023] 5.3：i = i+1。如果 i<j，转 5.2。如果 i ≥ j，则说明所有强连通分量均已被新节点代替，此时图 G 转变为了有向无环图 G^*，执行第六步。

[0024] 第六步：从出度为 0 的节点开始，自底向上遍历有向无环图 G^* 中所有节点，利用 GPU 计算所有节点的影响值。具体方法是：

[0025] 6.1：变量的定义和初始化。方法是：

[0026] 6.1.1：使用布尔数组 Visited[] 记录各个节点是否已经被访问过，$Visited[v_p]$ 等于 true 表示节点 v_p 已经被访问，$Visited[v_p]$ 等于 fajse 表示节点 v_p 未被访问，其中 $0 \leqslant p \leqslant n-1$。将数组 Visited[] 全部初始化为 false，表示所有节点均未被访问；

[0027] 6.1.2：使用整数数组 Count[] 记录各个节点已经被访问的子节点个数，其中 $0 \leqslant Count[v_x] \leqslant outdegree[v_x]$，$0 \leqslant x \leqslant n-1$，$outdegree[v_x]$ 是节点 v_x 的出度。将数组 Count[] 全部初始化为 0，表示均未被访问；

[0028] 6.1.3：使用整数数组 Inf[] 记录各个节点的影响值，其中 $0 \leqslant Inf[v_x] \leqslant n$，$0 \leqslant x \leqslant n-1$。将数组 Inf[] 全部初始化为 0；

[0029] 6.1.4：使用字符串数组 Label[] 记录各个节点的标签，标签 $Label[v_x]$ 标记了节点 v_x 可能同其他节点发生重叠的位置，其中节点 v_a 及节点 v_b 重叠于节点 v_c 当且仅当从节点 v_a 和 v_b 均存在至少一条路径可达节点 v_c，$0 \leqslant a,b,c \leqslant n-1$。将数组 Label[] 全部初始化为 NULL。

[0030] 6.1.5：使用布尔变量 Stop 记录线程计算是否完成，Stop 等于 true 表示该次模拟中所有节点影响值计算已经完成，Stop 等于 false 表示未完成。Stop 为全局变量，所有 GPU 线程均可以修改其内容。初始化 Stop 为 false。

[0031] 6.2：如果停止标志 Stop 为 false，说明此次模拟影响值计算仍未完成，则转 6.3 使用 GPU 多线程并行计算；如果 Stop 为 true，则说明此次模拟中所有节点的影响值均已计算完毕，转第七步。

[0032] 6.3：GPU 采用单指令流多数据流的执行方式，以多线程并行的方式计算节点的影响值；多线程并行的方式是指：GPU 为每个节点分配一个线程计算影响值，GPU 一次由 y 个

线程并行计算 y 个节点的影响值，y 为 GPU 中的流处理器数目(因 GPU 型号不同而流处理器个数不同)，当 GPU 当前 y 个节点的影响值计算完成后，若还有节点影响值未计算，则 GPU 经过 GPU 线程调度以多线程并行的方式计算剩余节点的影响值，直至所有节点的影响值计算完毕，GPU 采用单指令流多数据流的执行方式，同一个流处理器单元中的所有流处理器共用同一取指部件，指令按顺序发射，没有分支预测，即不同线程执行同一条指令，但处理不同数据，从而达到并行计算。GPU 的一个线程计算节点 v_p 影响值的方法是：

[0033] 6.3.1：将停止标志 Stop 置为 true。

[0034] 6.3.2：如果 Visited[v_p] 等于 false，执行 6.3.3；否则说明节点 v_p 已经被访问过，转 6.2。

[0035] 6.3.3：如果 Count[v_p] 等于节点 v_p 的出度，则说明节点 v_p 的所有子节点均已被访问，执行 6.3.4 计算节点 v_p 的影响值；否则说明节点 v_p 的子节点中仍有未处理的节点，则将停止标志 Stop 置为 false，转 6.2。

[0036] 6.3.4：计算节点 v_p 的所有子节点影响值的总和 sum，$sum = \sum_{v_q \in out[v_p]} Inf[v_q]$，其中 out[$v_p$] 是节点 v_p 所有子节点的集合。

[0037] 6.3.5：计算节点 v_p 的标签 Label(v_p)。节点 v_p 的标签 label(v_p) 等于节点 v_p 的所有子节点对 v_p 贡献的并集，即 $Label(v_p) = \cup_{v_q \in out[v_p]} Con(v_q)$，其中 Con($v_q$) 是子节点 v_q 对节点 v_p 的贡献。子节点 v_q 对节点 v_p 的贡献是指：如果子节点 v_q 的入度大于 1，则重叠可能发生于 v_q，此时 v_q 对节点 v_p 的贡献为节点 v_q 自身，即 Con(v_q) = v_q；如果子节点 v_q 的入度小于等于 1，重叠不可能发生于 v_q，此时 v_q 的贡献为节点 v_q 的标签，即 Con(v_q) = Label(v_q)。

[0038] 6.3.6：计算节点 v_p 所有子节点的集合 out[v_p] 重叠的影响值 Overlap(out[v_p])，方法是：

[0039] 6.3.6.1：初始化 Overlap(out[v_p]) 为 0，初始化重叠范围集合 Range 为 out[v_p]。

[0040] 6.3.6.2：对于任意节点 $v_a \in$ Range，如果存在节点 $v_b \in$ Range 且 $v_b \neq v_a$，并且从节点 v_a 存在路径可达节点 v_b，此时重叠发生在节点 v_b，故 Overlap(out[v_p]) = Overlap(out[v_p]+Inf[v_b]，同时将 v_b 从 Range 中删除，即 Range = Range-v_b。

[0041] 6.3.6.3：使用字符串数组 Extra[] 记录 Range 中的重叠项，将 Extra[] 初始即 Overlap(out[v_p]) = Overlap(out[v_p])+(Overlap(Filter)-Overlap(Extra))。

[0042] 6.3.7：计算节点 v_p 的影响值 Inf[v_p]，Inf[v_p] = sum+weight(v_p)-overlap(out[v_p])，其中 weight(v_p) 是节点 v_p 的权重。Totallnf[v_p] = Totallnf[v_p]+Inf[v_p]，其中 TotalInf[v_p] 是 R 次蒙特卡洛模拟节点 v_p 的总影响值。R 是总模拟次数，一般设定为 20000。

[0043] 6.3.8：如果节点 v_p 无父节点，则转 6.3.9；否则，对于节点 v_p 的任意父节点 v_s，将其已访问子节点个数 Count[v_s] 加 1，即 Count[v_s] = Count[v_s]+1，并将停止标志 Stop 置为 false。

[0044] 6.3.9：将节点 v_p 标记为已经被访问，即 Visited[v_p] = true，转 6.2；

[0045] 第七步：将蒙特卡洛模拟次数 Num 加 1。判断 Num 是否小于 R，如果 Num<R，转第三步，否则执行第八步。

[0046] 第八步：对集合 V-S 中所有节点，选择 TotallnF[] 最大的节点 v 加入到集合 S 中。

[0047] 第九步：如果集合 S 的节点个数 |S| ＜ K，转第二步，否则说明已经选定 K 个最有影响的节点，结束。

[0048] 与现有技术相比，采用本发明能达到以下有益效果：

[0049] 1. 本发明第四步计算图中的强连通分量，由于强连通分量中所有节点的影响值完全相同，因此在第五步中将各强连通分量替换为单个节点，从而将图转化为有向无环图，可以显著减少影响值的计算量。

[0050] 2. 本发明第六步采用了自底向上的遍历方法为各个节点计算影响值。由于父节点的影响值直接依赖于其所有子节点的影响值，因此通过方法可以仅用一次全图遍历即可得到所有节点的影响值，降低了杂度。

[0051] 3. 本发明充分挖掘了原贪心算法的并行性和 GPU 的并行计算能力，尤其是为每一个节点分配一个 GPU 线程进行计算。利用 GPU 线程之间的并行执行，大幅度减少了程序的执行时间，因而可以对更大规模的社会网络进行处理，具有良好的可扩展性。

附图说明

[0052] 图 1 是最优的贪心算法 MixGreedy 流程图；

[0053] 图 2 是本发明总体流程图。

具体实施方式

[0054] 图 1 是最优的贪心算法 MixGreedy 流程图。

[0055] 第一步：初始化节点集合 S 为空。

[0056] 第二步：设定当前蒙特卡洛模拟次数 Num ＝ 0。

[0057] 第三步：采用蒙特卡洛模拟方法对图进行选边，得到图 G′ 。

[0058] 第四步：为各个节点进行宽度优先搜索，计算各节点的影响值。

[0059] 第五步：将蒙特卡洛模拟次数 Num 加 1。判断 Num 是否小于 R，如果 Num ＜ R，则转第三步，否则执行第六步。

[0060] 第六步：选择集合 V-S 中 TotallnF[] 影响值最大的节点 v 加入到集合 S 中。

[0061] 第七步：如果集合 S 的节点个数 |S| ＜ K，则转第二步，否则说明已经选定 K 个最有影响的节点，程序结束退出。

[0062] 图 2 是本发明总体流程图。

[0063] 第一步：初始化节点集合 S 为空。

[0064] 第二步：设定当前蒙特卡洛模拟次数 Num ＝ 0。

[0065] 第三步：采用蒙特卡洛模拟方法对图进行选边，得到图 G′ 。

[0066] 第四步：寻找图 G′ 中的强连通分量。

[0067] 第五步：根据各强连通分量将图 G′ 转变为有向无环图 G^*。

[0068] 第六步：从出度为 0 的节点开始，自底向上遍历有向无环图 G^* 中所有节点，利用 GPU 不同线程并行计算所有节点的影响值

[0069] 第七步：将蒙特卡洛模拟次数 Num 加 1。判断 Num 是否小于 R，如果 Num ＜ R，则转第三步，否则执行第八步。

[0070] 第八步：选择集合 V-S 中 TotallnF[] 影响值最大的节点 v 加入到集合 S 中。

[0071] 第九步：如果集合 S 的节点个数 |S| ＜ K，则转第二步，否则说明已经选定 K 个最有影响的节点，程序结束退出。

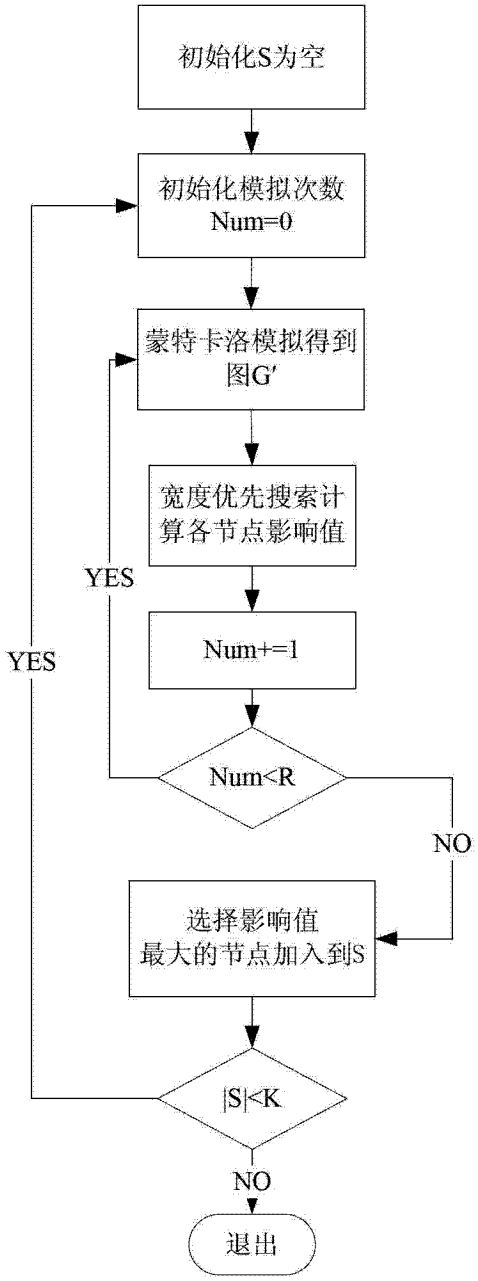

初始化S为空
初始化模拟次数
Num=0
蒙特卡洛模拟得到
图G'
宽度优先搜索计
算各节点影响值
Num+=1
Num<R
YES
NO
选择影响值
最大的节点加入到S
|S|<K
YES
NO
退出

图 1

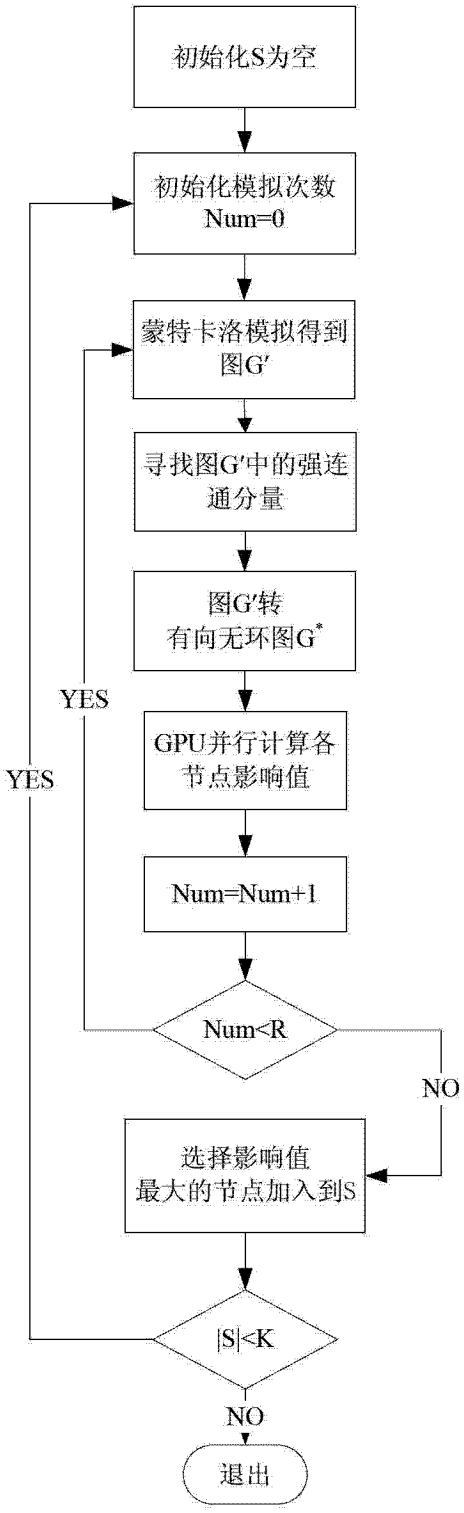

初始化S为空
初始化模拟次数
Num=0
蒙特卡洛模拟得到
图G'
寻找图G'中的强连
通分量
图G'转
有向无环图G*
GPU并行计算各
节点影响值
Num=Num+1
Num<R
YES
NO
选择影响值
最大的节点加入到S
|S|<K
YES
NO
退出

图 2