空间连接处理中提炼步骤的遗传优化

熊 伟, 廖 巍, 张 帆, 景 宁, 陈宏盛
(国防科技大学电子科学与工程学院, 湖南长沙 410073)

摘 要: 内务处理是用来优化空间连接处理提炼步骤的I/O代价, 属于NP-难的问题, 现有求解方法复杂度太高. 本文将页面聚簇和聚簇调度问题分别归约为图的k 划分和最长路径问题, 提出了改进的遗传算法和基于最大生成树的近似算法进行求解. 对经典遗传算法的进行了修正以满足页面聚簇划分约束. 近似算法的解大于最优解的一半. 理论分析和仿真实验验证了算法的可行性和有效性.

关键词: 空间连接; 提炼步骤; 内务处理; 遗传算法
中图分类号: TP392 文献标识码: A 文章编号: 0372-2112 (2006) 06-1069-05

Genetic Optimization to the Refinement Step of Spatial Join Processing

XIONG Wei, LIAO Wei, ZHANG Fan, JING Ning, CHENG Hong-sheng
(School of Electronic Science and Engineering, National University of Defense Technology, Changsha, Hunan 410073, China)

Abstract: The housekeeping processing is usually performed to optimize the I/O costs of spatial join refinement step. The problems in housekeeping phase have been shown NP-hard, and most methods solve them with high computation complexity. We reduce the page-clustering problem and cluster-sequencing problem to k-way partition and maximum length of graph respectively, and solve the problems with improved genetic algorithm and approximation algorithm based on maximum-span-tree. To satisfy the constraint of page cluster partition, we make some correction on the classical genetic algorithm. The solution of approximation algorithm is greater than half of optimal solution. Theoretical analysis and simulation experiment shows the feasibility and effectiveness.

Key words: spatial join; refinement step; housekeeping; genetic algorithm

1 引言

空间连接是空间数据库中最重要和最常用的操作, 由于空间对象的大规模和空间操作的高计算量, 造成了空间连接代价高昂. 为了提高处理效率, 空间连接处理通常分为过滤和提炼两个阶段. 近来, 很多方法通过在过滤和提炼阶段之间进行内务处理(Housekeeping)优化提炼阶段的I/O代价. 内务处理步骤通常可分为聚簇划分和聚簇存取顺序调度两个阶段. 聚簇划分可以将那些密切相关且可以被同时载入内存的对象聚集到一起; 聚簇存取顺序调度则负责调度聚簇的访问顺序, 以便通过降低页面重复读取的总数得到最小的存取开销.

存取顺序优化问题的正式名称为最优页面访问顺序问题. 针对这些问题, 人们提出了许多启发式解决方案, 这些解决方案大致上可以分为非对称单表聚簇方案和对称双表聚簇方案两个类型[1]. Abel等在深入分析影响空间连接性能的不同因素的基础上, 提出了一种混合非对称单表聚簇和对称双表聚簇的交换调度策略[2]. Shekhar等[1]提出的利用最小割图划分算法实现连接索引分组的思想, 将连接索引划分为若干组, 然后再对每组中的数据进行批处理. 通过对可以实现稀疏矩阵对角化排列的BEA(Bond Energy Algorithm)算法的适当修改, Xiao等[3,4]给出了一种更为高效的PCG 邻接矩阵分解方案. 在分组批处理策略中, 同一个对象可以被划分到不止一个组.

本文主要研究空间连接提炼步骤的优化策略, 通过将页面聚簇归约为页面连通图的k 划分问题, 提出采用遗传算法优化求解这一NP 难的问题, 通过全局搜索获得一种全局更优的解; 将页面聚簇的调度归约为最长路径问题, 利用近似算法求解, 获得接近最优的页面调度. 最后通过实验验证了本文的方法能够优化空间连接处理的提炼步骤.

收稿日期: 2005-05-24; 修回日期: 2005-11-18
基金项目: 国家高技术研究发展计划(863计划)(No. 2003AA135110)

2 问题描述

基于空间连接索引(SJI)所对应的页面连通图(PCG, Page Connectivity Graph),降低空间连接提炼步骤中冗余的 I/O 需要解决聚簇划分和聚簇存取顺序调度两个问题. 本节将给出这两种优化问题的形式化描述. 对称的页面聚簇方法在页面聚簇中包含了连接关系的两个数据集,因此只有割边(Cut)关联的节点可以用来减少冗余 I/O. Shekhar等将该问题形式化为求 PCG 的最小割划分问题[1]. 通过将 PCG 划分为不相连的子集,使得连接不同划分的边数(即割边)最少. 通过定义割边的概念,并证明割边对于页面聚簇优化的影响,我们进而将该问题归约为图的 k 划分问题.

定义 1 割边

给定页面连通图 $G=(V,E)$ 节点集的一个划分$\{V1, V2, \cdots, Vk\}$,割边 E 定义为$\{(vi,vj) \mid vi\in Vi, vj\in Vj, i\neq j, 1\leqslant i, j\leqslant k\}$,用 E_C 表示割边的集合. 基于割边, Shekhar 证明了下面这个定理[4].

引理 1 给定页面连通图 $G=(V,E)$ 的一个划分$\{V1, V2, \cdots, Vk\}$,E_C 为割边集合,假设$|Vi|$小于缓冲区的规模,则存在一个长度 $K\leqslant |E_C|+|V|$的页面访问序列. 当$|E_C|=0$时,可以保证获得最优的页面访问序列.

问题 1 PCG 的 k 划分问题

给定页面连通图 $G=(V,E)$, G 的节点数 $|V|=n$,缓冲区规模为 $B\geqslant 2$. 现将节点集 V 划分为k 个子集,分别为 $V1, V2, \cdots, Vk$,满足 $i\neq j$ 时,$Vi\cap Vj=\phi$ 并且$\cup_{1\leqslant i\leqslant n} Vi=V$,约束为$|Vi|\leqslant(B-1)$(这里表示留出了一个页面用于处理割边所关联的节点),划分数根据节点数和缓存规模计算为 $k=\lceil |V|/(B-1)\rceil$ 问题的目标是求使得$\Sigma|E_C|$最小的划分. 这样就将 PCG 的最小割划分问题归约为图的 k 划分问题.

引理 2 PCG 的 k 划分问题解满足划分中必定包含属于不同数据集的节点.

证明 通过对 PCG 划分中节点假设进行反证法可以很容易证明. 引理 2 证明了 PCG 的 k 划分问题求解是一种对称的页面聚簇方法.

问题 1 属于组合优化的范畴,是一个 NP-难问题[5]. 其计算复杂度随图的规模和划分数 k 的增大而急剧增加,传统的启发式搜索方法已无能为力. 遗传算法作为一种全局优化搜索算法,由于其本身所具有的全局收敛性和隐含的并行性,加之其简单易用、鲁棒性强,问题越复杂,它相对于其他算法的优越性越明显,故十分适合解决这类问题.

定义 2 加权聚簇图(WCG, Weighted Cluster Graph)及其路径

给定页面连通图 $G=(V,E)$ 节点集的一个划分$\{V1, V2, \cdots, Vk\}$,加权聚簇图 $G=(Vk, Ek, w)$ 是一个无向图,其中节点集 Vk 为表示页面聚簇的划分集合, $i\neq j$ 时,边$(Vi, Vj)\in Ek$,当且仅当 Vi, Vj 之间有割边,边(Vi, Vj)的权值 w 为Vi、Vj 上割边的数目和,记为 $w(Vi, Vj)$,如果 Vi Vi, Vj 之间没有割边,则 $w(Vi, Vj)=0$. 给定加权聚簇图 $G=(V, E, w)$, $V=\{V1, V2, \cdots, Vk\}$,则 G 中的一条路径为$P=(Vi1, Vi2, \cdots, Vik)$,其中 $im\neq in$, $1\leqslant im, in\leqslant k$, P 经过G 中每个节点一次且仅一次. $w(Vi, Vj)$为节点 Vi 到节点 Vj 的长度. 路径的长度为$\Sigma_{1\leqslant i, i\leqslant k, i\neq j} w(Vi, Vj)$,用$|P|$表示.

给定 PCG 如图 1(a),设缓冲区规模为 5,那么对应的某个聚簇划分为图 1(b),加权聚簇图为图 1(c),(C1, C2, C3)为 WCG 的一条路径,路径长度为 3.

引理 3 给定加权聚簇图 $WCG=(Vk, Ek, w)$,若 WCG 上的最长路径为 P(边的权值表示路径长度),WCG 边的权值和为 W,那么加权聚簇图 I/O 代价可以用页面访问序列的长度表示为 $K\leqslant|V|+|W|-|P|$.

证明 $|W|$是实际冗余的 I/O. 只要我们可以找到当前聚簇划分中的某个页面满足:

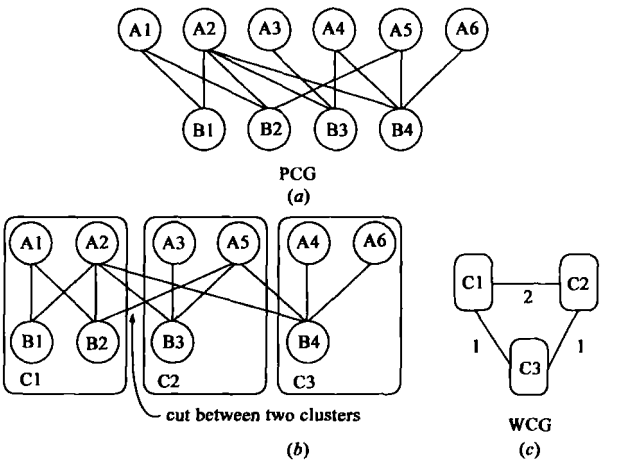

图 1 割边、加权聚簇图及加权聚簇图中的路径示意图

(1)无割边与之相连;

(2)已经经过连接处理,连接结果实体化.

那么该页面可以被页面访问序列中下一个聚簇划分中的页面替换掉,从而减少了冗余的 I/O,在页面载入的累积中,最大可以节约的 I/O 总数为$|P|$. 证毕. 基于引理 3,我们将 WCG 中的边权值定义为路径的长度,那么聚簇划分调度的问题归约为求 WCG 最大长度路径的问题. 问题描述如下.

问题 2 WCG 最大长度路径问题

给定加权聚簇图 $G=(V, E, w)$, $V=\{V1, V2, \cdots, Vk\}$,则问题的目标是求得 G 中的一条路径 $P=(Vi1, Vi2, \cdots, Vi3)$使得$|P|$的值为所有路径中最大.

问题 2 是一个典型的旅行商问题,属于 NP-难的问题[6]. 如果有效地划分聚簇后,在聚簇数目较小以及加权聚簇图中割边数量较小的情况下,聚簇调度相对来说对于 I/O 次数的影响较小,这时可以采用本文提出的近似算法予以求解,否

则, 也可以应用遗传算法求解, 类似编码方法和相关算法提出了很多[6].

3　算法描述

3.1　PCG 的 k 划分问题的遗传算法求解

PCG 的 k 划分问题有以下三个基本特点: 第一, 划分后所得到的任一子集都不能为空, 至少必须包含一个顶点, 至多包含由缓冲区规模限制的顶点数目. 经典遗传操作不能保证满足这一约束条件, 需要进行修正; 第二, PCG 的 k 划分问题是约束优化问题, 经典算法在产生非法解时丢弃, 当合法解产生概率较低时, 该方法将浪费大量 CPU 时间. 因此需要针对问题的约束条件构造算子和编码, 保证只产生合法解, 同时收敛速度较快. 第三, 必须考虑适当的变异操作, 减弱因选择操作造成的群体多样性减少的影响, 维持群体的多样性, 增强局部随机搜索能力.

算法的设计分为以下 5 个部分:

(1)编码方法

本文采取自然数编码方式. 设 PCG 有 n 个节点, 根据需要欲划分为 k 个聚簇, 则染色体(即问题的一个解) 可如下表示: $x = (x_1, x_2, \cdots, x_n)$, 其中 $x_i \in \{1, \cdots, k\}$, $i \in \{1, 2, \cdots, n\}$

在种群初始化的时候, 需要满足每个划分中节点数目小于缓冲区规模的约束. 因此初始化的时候, 可以按照划分次序, 重复随机选取 k 个未归入划分的节点分别归入到 k 个划分, 直到所有节点划分完毕. 这样虽然对于靠前的划分有所偏好, 但是通过后面的遗传操作仍然可以保证全局搜索; 也可以在初始化种群后, 修改某些偏好的划分来保证搜索空间的全局性.

设 $1 \leqslant i, j \leqslant n$, 则顶点 xi 与 xj 的连接关系表示为 xij, 即:

$$xij = \begin{cases} 1, & xi \text{ is connected with } xj \\ 0, & \text{otherwise} \end{cases},$$

得到一个染色体 x 后, 对于 PCG 中的边 (x_i, x_j), 定义割边函数如下:

$$cut(xij) = \begin{cases} 1, & xi \neq xj \text{ and } i \neq j \\ 0, & \text{otherwise} \end{cases}$$

(2)适应度函数定义

根据问题 1, 目标函数是求割边集合元素数目最小值的问题, 因此我们直接将目标函数作为适应度函数. 假设 x 表示问题的一个解, 则它的适应度函数 f(x) 定义为:

$$f(x) = \sum_{1 \leqslant i, j \leqslant n} cut(xij) * xij.$$

(3)选择操作

用适应度函数计算每个解的适应度, 选择适应度高的解生成下一代. 可采用轮盘赌选择法, 即在每一代运算过程中, 先计算出个体的相对适应值 $\frac{f(x)_i}{\sum f(x)_i}$, 记为 P_i, $i \in \{1, 2, \cdots, npop\}$. $f(x)_i$ 表示种群中第 i 个个体的适应度函数值. npop 为设定的种群个数. 个体被选中的概率与其在群体中的相对适应度成正比.

(4)交叉操作

交叉算子的作用是实现个体结构的重组, 拓展全局搜索空间. 但个体结构重组的幅度不宜过大, 否则会过多地破坏父体的信息, 使遗传算法退化为随机搜索算法. 这里使用一种对应基因单点交叉算子, 随机选取一个节点的位置, 然后交换两个染色体中对应的部分, 从而产生新染色体. 具体说明如图 2 所示.

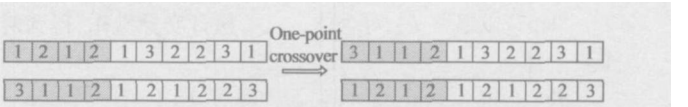

图 2　单点交叉操作示意图

为满足划分约束, 既要保证交叉产生的新染色体仍可以表示有效的划分, 不存在空划分, 也要保证每个划分中的节点总和少于缓冲区规模. 因此在交叉后我们通过两种修正, 来保证解的可行性. 第一种情况是当交叉后出现了空划分, 那么就检查是否有划分超过缓冲区规模, 如果超过, 则随机选取一个节点将其归入空划分. 如果没有划分超过缓冲区规模, 那么随机选取交叉部分中的一个划分归入空划分, 这种修正称为空划分修正; 第二种情况继续修正超过缓冲区规模的划分, 不断随机选取该划分中的节点归入到小于缓冲区规模的划分中, 直到所有划分中的节点数目小于缓冲区规模, 称为溢出划分修正. 修正执行示意见图 3.

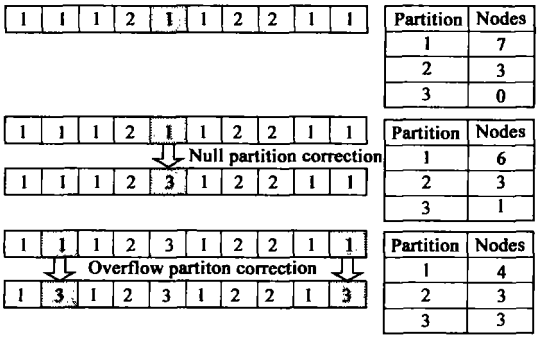

图 3　交叉结果的空划分与溢出划分修正

(5)变异操作

变异操作执行时随机在选择染色体中两个对应不同划分的节点, 然后交换这两个节点, 新产生的染色体中能够保证划分仍然能够满足每个划分中节点总和小于缓冲区规模, 但是对不同划分中的节点进行了重新分配.

基于上述设计思想, 我们给出求解 PCG 图 k 划分问题的遗传算法 KPP(K-way PCG Partition) 算法, 描述如下:

Step1: 产生初始种群 $X(0) = \{x_1(0), x_2(0), \cdots, x_{npop}(0)\}$, npop 为种群规模, 进化代数 t 初值为 0;

Step2: 根据适应度函数定义, 计算第 t 代种群 X(t) 中

每个个体的适应度值 $f(x_i(t))$, 为保证算法收敛到全局最优解, 必须保存适应度值最大的个体 $x_{max}(t)$;

Step3: 计算种群 $X(t)$ 中各个体的适应度值比例, 确定选择概率 P_i;

Step4: 根据选择概率 P_i 和轮盘赌选择策略在 $X(t)$ 中选择两个个体 $x_a(t)$, $x_b(t)$;

Step5: 以交叉概率 P_c 对个体 $x_a(t)$, $x_b(t)$ 执行交叉操作, 得到两个新个体 x_{ac}, x_{bc}, 这里通常 P_c 可以设置较大;

Step6: 对新个体 x_{ac}, x_{bc} 进行空划分修正和溢出划分修正;

Step7: 以概率 P_m 对个体 x_{ac}, x_{bc} 执行变异操作, 得到两个新个体 x_{acm}, x_{bcm};

Step8: 将 x_{acm}, x_{bcm} 插入到新种群 $X(t+1)$ 中;

Step9: 若新种群 $X(t+1)$ 中个体数目大于等于 npop, 则用 $X(t)$ 中 $x_{max}(t)$ 替换 $X(t+1)$ 中适应度值最小的 $x_{min}(t+1)$; 否则转 Step4, 继续从上代群体中生成新个体;

Step10: 设 $f_{max}(X(t))$, $f_{min}(X(t))$ 分别表示 t 代中适应度最大值和最小值, 若 $f_{max}(X(t)) - f_{min}(X(t)) < \varepsilon$, 其中 ε 取值远小于适应度值, 则算法终止, 输出 $X(t)$, 否则转 Step2.

我们使用有限状态马尔可夫链的一般遗传算法收敛性证明方法对 KPP 算法的收敛性进行证明.

引理 4 对应有限状态齐次 Markov 链的一般遗传算法, 在算法操作中, 若每代群体在选择操作前进行最优解的保持, 则遗传算法以概率 1 收敛到全局最优解[6].

定理 1 KPP 算法以概率 1 收敛到全局最优解.

证明 在 KPP 算法中, 一旦染色体长度 n 和种群规模 npop 给定且有限时, 则所有个体形成的个体空间和所有种群形成的种群状态空间均为有限的; 由于算法中使用的参数不随时间变化, 且算法中每一代状态的转移依赖于选择、交叉和变异操作, 且与进化代数无关, 因此 KPP 算法可以视为一个有限状态的齐次 Markov 链, 这里的证明从略. 表征 Markov 链的状态转移矩阵为交叉、变异和选择操作所决定的状态转移矩阵的乘积. 为了保证算法的优化性和收敛性, 在选择操作前保存了适应度值最大的个体. 由引理 4 可知, KPP 算法以概率 1 收敛到全局最优解.

3.2 WCG 最大长度路径问题的近似求解

本节将 Xiao 提出的聚簇最大覆盖路径算法[3] 引入 WCG 最大长度路径问题的求解, 在给出 WCG 最大长度路径近似求解算法 MPW (Maximum-length Path of WCG) 算法前, 我们先给出 WCG 最大生成树的定义.

定义 4 WCG 最大生成树

给定加权聚簇图 G, 则 G 的所有生成树中权值总和最大的生成树为最大生成树. 称权值总和为最大成生树的长度.

这样, 我们描述 MPW 算法如下:

Step1: 构造 WCG 的最大生成树 T, 可以使用如 Kruskal 或 Prim 算法;

Step2: 随机选取一个节点作为树根, 深度优先遍历 T 得到顶点序列 S;

Step3: 按照遍历顺序 S, 构造 WCG 的最大长度路径 P.

显然在第二步中选择不同起始节点遍历 T, 会得到不同的路径. 设 MPW 得到的路径长度为 $|P|$, 最优路径的长度为 $|T|$, 将 WCG 的路径长度看成最大覆盖路径权值和, 则 Xiao 已经证明了 $|T| \leqslant 2|P|$, 即路径长度最坏情况下也大于最优路径长度的一半. 根据 Xiao 的证明很容易得到定理 2.

定理 2 MPW 算法的解大于 WCG 最大长度路径问题最优解的一半.

为了选择较好的起始节点和聚簇中节点页面的替换时, 可以采用以下启发式策略:

(1) 聚簇之间有割边的连续装载; (2) 已在内存中相同节点的割边对应的节点连续处理; (3) 节点连接割边多的聚簇先载入内存.

4 实验仿真

实验环境为 Celeron 1.7GHz 处理器, 256MB DDR 内存, 40GB 7200RPM IDE 硬盘, 操作系统为 Windows 2000, 算法用 Visual C++ 实现. 实验数据使用的是来自 Census TIGER ®2000 的纽约公路和典型地物 MBR 数据集, 两个数据集的对象数分别为 8356、354. 实验仿真参数设置点数据规模为 8 Bytes, 在不考虑空间对象复杂度的情况下, 每个空间对象用其 MBR 表示, 每个 MBR 平均包含 32 个点, 从而每个页面存放的空间对象 MBR 数目相同. 实验处理由两个数据集的距离连接后得到连接索引. 表 1 根据连接距离与空间对象 MBR 平均边长的比值, 给出了连接索引的基数, 页面分别设定为 16KBytes、4 KBytes 和 2 KBytes 时 PCG 节点规模. 这样基本满足大规模问题求解所需的节点规模.

表 1 不同距离连接得到的 PCG 节点规模

距离/平均边长	连接索引基数	PCG 节点规模		
		页面 16KBytes	页面 4KBytes	页面 2KBytes
0.2	9841	154	616	1231
0.6	9987	157	625	1249
1	11205	176	701	1401
1.4	13109	205	820	1639
1.8	19038	298	1190	2380

将算法得到的目标函数值规格化, 设引入空划分和溢出划分后的目标函数值为 1. 图 4 考察进化代数为 100 时, PCG 节点规模增加时修正操作对算法性能的影响, 实验结果表明引入修正操作后虽然算法收敛时间有所增加, 但优化率有明显的增加. 而且可以看出溢出划分修正操作相比空划分修正更加耗时, 优化率也更高, 这是因为在溢出划分中对划分进行再分配, 增强了算法全局搜索能力, 有利于搜索节点划分的不同组合, 从而算法性能得以改进. 为了使修正操作不会陷入局部搜索, 可以以一个小概率允许不修正, 保留一部分不合法的解来进行迭代.

图 5 左缓冲区规模设定为 512KBytes, 页面规模为 16KBytes. 实验结果表明, 因为连接距离的增加, 结果集中 PCG 相关联的边的数目也增多, 对于没有优化的页面调度, 无疑产生大量可能冗余的I/O次数, 可以看出, 随结果集基数的增加 I/O 代价急剧增加. 同时可以看到, 采用遗传算法优化在 PCG 规模较小的情况下, 效果并不比最小割方法好, 遗传算法的优势主要体现在问题规模很大的情况. 图 5 右页面设定为 4KBytes, 距离连接结果集基数设定为 11205. 实验结果说明, 缓冲区规模较大时, 优化方法效果都不明显, 因为此时聚簇相对缓冲区规模比较小, 一次可以载入多个页面处理, MPW 算法的调度优势无法体现. 缓冲区规模较小的情况下, 我们采用的页面调度方法效率相对最小割方法有所提高.

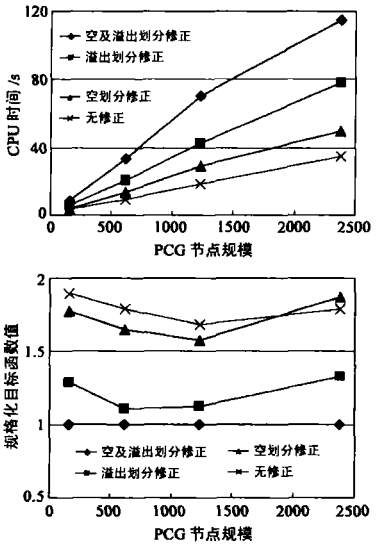

图 4　修正操作对算法性能的影响结果比较

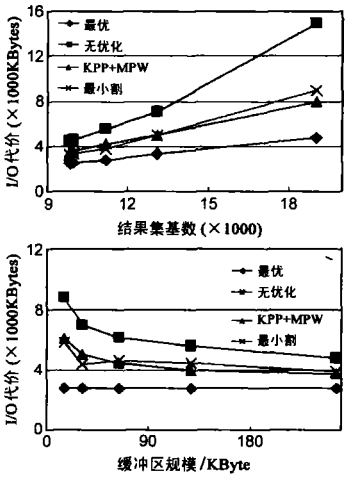

图 5　I/O 代价与结果集基数和缓冲区规模的关系

5　结论

本文提出的遗传算法和最大生成树算法求解思路, 获得了内务处理问题全局更优的解. 以空间查询中常见的距离连接为例, 通过实验仿真验证了本文方法的可行性和有效性. 我们下一步将考察如何处理非均匀的空间对象, 这时每个页面上存放的空间对象数目将不再相等, 需要引入新的方法. 在聚簇数目较大以及加权聚簇图中割边数量较多的情况下, 聚簇调度同样可以考虑采用遗传算法来求解 WCG 最大长度路径.

参考文献:

[1] S Shekhar, C T Lu, S Chawla, S Ravada. Efficient join-index-based spatial-join processing: A clustering approach [J]. IEEE Trans on Knowledge and Data Engineering, 2002, 14(6): 1400– 1421.

[2] Abel D J, Gaede V, Power R A, Zhou X. Caching strategies for spatial joins[J]. GeoInformatica, 1999, 3(1): 33 – 59.

[3] Xiao J, Zhang Y, Jia X. Clustering non-uniform-sized spatial objects to reduce I/O cost for spatial-join processing[J]. The Computer Journal, 2001, 44(5): 384– 397.

[4] Ravada S, Shekhar S, et al. Optimizing join index based spatial join processing: a graph partitioning approach, University of Minnesota[A]. Proceedings of 17th Symposium on Reliable Distributed Systems[C]. Indiana: IEEE Computer Society 1998. 302– 308.

[5] G Poirier. Spatially Augmented Greedy Heuristic for Efficient I/O in Spatial-join Processing of Non-uniform-sized Spatial Objects[R]. Technical Report(CS-741). Waterloo: University of Waterloo, 2002. 1– 25.

[6] 王凌. 智能优化算法及其应用[M]. 北京: 清华大学出版社, 2001. 40– 47.

作者简介:

熊　伟　男, 1976 年 11 月生于湖南长沙, 博士生, 主要研究方向: 主动数据库, 空间数据库. E-mail: xiongw@21cn.com

廖　巍　男, 1980 年 2 月生于湖北襄樊, 博士生, 主要研究方向: 时空数据库, 空间数据库. E-mail: liaowei_2000@163.com