一种基于图形处理器的压缩单纯形方法

白洪涛[1,2],欧阳丹彤[1,2],何丽莉[1,2],姜珊珊[3]
(1. 吉林大学计算机科学与技术学院,吉林长春 130012;
2. 吉林大学符号计算与知识工程教育部重点实验室,吉林长春 130012;
3. 理光软件研究所(北京)有限公司,北京 100044)

摘 要: 针对 GPU 通用计算环境 CTM 纹理资源的限制,研究了一种适于 CTM 的单纯形方法. 依据单纯形方法每次变换最多只增加一列非单位元向量和矩阵求逆运算的特征,给出 GPU 上系数矩阵、基逆矩阵等的压缩存储策略及在该策略下求解基逆矩阵、单纯形乘子和检验数等步骤新的计算规则. CPU 主要进行迭代控制;而计算密集类任务皆由 GPU 完成. 理论分析证明该方法比标准方法在时空复杂度上提高了一个数量级. 数值实验表明该方法不仅扩大了可求解问题的规模,且在获得正确优化结果的前提下,效率比 CPU 版本有数百倍的提高,甚至数倍领先于 MATLAB R2007a.

关键词: 单纯形方法; 图形处理器; CTM; 纹理; 像素程序
中图分类号: TP391 **文献标识码**: A **文章编号**: 0372-2112 (2009) 11-2574-05

A Compressed Simplex Method Based on GPU

BAI Hong-tao[1,2], OUYANG Dan-tong[1,2], HE Li-li[1,2], JIANG Shan-shan[3]
(1. *College of Computer Science and Technology, Jilin University, Changchun, Jilin* 130012, *China*;
2. *Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun, Jilin* 130012, *China*;
3. *Ricoh Software Research Center (Beijing) Co. Ltd., Beijing* 100044, *China*)

Abstract: We present a GPU-based algorithm for solving linear programming problems using simplex methods under CTM (close to the metal) environment. Matrix-packed storage strategy and correlative computation formulas such as revised basic matrix, simplex multiplier and departing variables are brought forward, according to the fact that at most one column vector produces in each vertex changing step and the feature of reverse matrix of simplex method. CPU takes charge of the iteration and all compute-intensive tasks are carried out by GPU. Theory analysis proves that both space complexity and time complexity are superior to traditional method. Numerical experiments on randomly generated problems demonstrate that this method can not only get correct optimal solution, but also reach as fast as hundred times of CPU-based version; and it runs several times faster than MATLAB R2007a.

Key words: simplex method; graphics processing unit; close to the metal; texture; pixel shader

1 引言

单纯形方法是求解线性规划问题的典型方法. 在优化计算领域,单纯形方法及其变种以其良好的结构和清晰的计算过程被商业系统如 MATLAB、CPLEX 和 LINDO/LINGO 等采用,创造了巨大的社会和经济效益. 近来,单纯形方法与遗传算法、进化计算等智能方法相结合,在一定程度上弥补了随机搜索的固有缺陷,取得了很好的效果[1,2].

如何快速处理大规模问题一直是优化计算研究和应用的一个难题. Paparrizos 等研究了避开可行解区域的外点单纯形方法、内点方法与单纯形相结合的新方法[3,4]. Luh 使用内点法替代单纯形阶段 I的方法[5]. Junior 提出了使用余弦迭代获得更优的初始可行解的方法[6]等等,这些方法大都致力于减少总的迭代次数,而大规模问题求解,单纯形方法的一次顶点变换即需较多的计算时间,只有提高迭代过程自身的效率,才能达到全面提升性能的目标.

基于图形处理器(graphics processing unit, GPU)的图形处理及其通用计算近年来成为图形学及高性能计算领域的热点研究课题[7]. GPU 不仅在图形计算领域能够将计算和渲染结合在一起,且有效用于如小波变换、电磁加速等完全通用的计算,逐步成为独立的高性能计算平台[8-10]. GPU 所拥有的众多单指令流多数据流(single

收稿日期:2007-10-08;修回日期:2009-06-09
基金项目:国家自然科学基金重大项目(No. 60496320;No. 60496321);国家自然科学基金(No. 60773097,No. 60873148)

instruction multi data,SIMD)处理器能够并行计算单纯形方法中矩阵的各元素,且矩阵运算的内存访问模式也能发挥 GPU 高存储带宽的优势.尽管如此,单纯形方法在 GPU 上的并行化还有如下困难:其一,GPU 架构迥异,基于 GPU 的算法必须针对某一或一个系列的 GPU 特殊设计,本文所用的是 ATI(现并入 AMD)的通用计算环境 CTM(close to the metal)[11];其二,GPU 单一纹理尺寸为 4096×4096(Directx 9.0),使用标准单纯形方法,只能求解约 1400 个变量的问题,不能像用 CPU 计算那样受限于总内存大小.

本文依据单纯形方法顶点变换和基逆矩阵等运算的特征,结合现实问题系数矩阵多为稀疏的性质,提出了一种适于 CTM 的压缩存储策略并给出了该策略求解基逆矩阵、单纯形乘子和检验数等步骤新的计算规则.计算密集负载全部移植至 GPU.理论分析和实验均表明了本文方法大幅提升了求解效率,扩大了问题的求解规模.

2　压缩存储策略

在数据存储上,压缩存储系数矩阵,矩阵 A 按列存储原始系数矩阵中的非零元、矩阵 rowA 记录 A 中每一个元素在原始矩阵中的行号、矩阵 columnA 记录原始矩阵每一列中第一个非零元在 A 中出现的序号,这三个矩阵在 CPU 上初始化并传输到 GPU 的存储器纹理中.在 GPU 存储器上直接生成的矩阵 B^{-1}和 E,也进行压缩存储:初始 B^{-1}是一个单位矩阵,每次迭代(顶点变换)至多增加一列非单位元向量,增量存储换入换出所旋转出的列,并用一个辅助矩阵 BIcount 记录该列在原 B^{-1}中的列号,同样的策略应用于旋转矩阵 E.我们选取单纯形方法中若干关键运算步骤给出压缩存储策略下新的计算规则:

(1) 基逆矩阵

起始基 B_0 是由松弛变量和(或)人工变量所组成的单位矩阵 I_m,因而其基逆矩阵 B_0^{-1} 也为 I_m.公式 $B_{new}^{-1} = EB_{old}^{-1}$迭代计算基逆矩阵 B^{-1}.B^{-1}求解的特点是从 I_m 开始,每次迭代最多只增加一个非单位列向量.据此,只需对 E 和 B_{old}^{-1}的非单位元进行计算,即可以得到新的 B_{new}^{-1}.设 $E=(e_1,\cdots,e_{l-1},\xi,e_{l+1},\cdots,e_m)$,$\xi=(\xi_1,\xi_2,\cdots,\xi_m)'$,换出变量列序号为 l;$B_{old}^{-1}=[\lambda_{i,j}]_{m\times m}$,其非单位列向量序号存储在集合 U 中;$B_{new}^{-1}=[\lambda'_{i,j}]_{m\times m}$.新基逆矩阵 B_{new}^{-1}的计算公式如下:

(i) $l\notin U$

$$\forall i\in[0,m-1]:\forall j\in U\cup\{l\}:\quad \lambda'_{i,j}=\begin{cases}\xi_i\times\lambda_{i,j}+\lambda_{i,j}, & i\neq l\wedge j\neq l\\ \xi_i\times\lambda_{i,j}, & i=l\wedge j\neq l\\ \xi_i, & j=l\end{cases}\tag{1}$$

(ii) $l\in U$

$$\forall i\in[0,m-1]:\forall j\in U:\quad \lambda'_{i,j}=\begin{cases}\xi_i\times\lambda_{i,j}+\lambda_{i,j}, & j\neq l\\ \xi_i\times\lambda_{i,j}, & j=l\end{cases}\tag{2}$$

为节省纹理资源和加速计算,在 GPU 上开辟三个静态纹理空间分别存储 ξ、B^{-1}的非单位元和集合 U;换出变量序号 l 存储在寄存器中.

本文对 B^{-1}的非单位列向量未使用压缩存储,这基于以下两点:存储上,即使基矩阵 B 是稀疏的,基逆矩阵 B^{-1}也不能保持稀疏性;计算上,GPU 的像素子程序(pixel shader)以并行模式运行,渲染顺序由 GPU 自身调度,渲染位置固定,若以压缩方式存储列元素,渲染就需要在纹理间多次寻址,不利于并行存储器访问.

(2) 单纯形乘子

单纯形乘子为 $Y=C_BB^{-1}$,其中

$Y=(y_1,y_2,\cdots,y_m)$,$C_B=(CB_1,CB_2,\cdots,CB_m)$,

Y 的计算公式如下:

$$y_i=\begin{cases}CB_i, & i\notin U\\ \sum_{j=1}^{m}CB_j\times\lambda_{i,j}, & i\in U\end{cases}\tag{3}$$

式(3)中,当 $i\in U$ 时,由独立的像素点渲染,而不是继续分解为两个元素的乘积和求和子程序.

(3) 检验数

标准单纯形方法计算非基变量 x_N 的检验数 σ_N 时,需生成非基变量系数矩阵 N.在 GPU 上会造成频繁寻址,并多消耗一个纹理.为此,本文计算所有变量的检验数,$\sigma_A=C_BB^{-1}A-C$,$\sigma_A=(\sigma_1,\sigma_2,\cdots,\sigma_n)$,$A=[a_{i,j}]_{m\times n}$.$\sigma_A$ 计算公式如下:

$$\sigma_j=-c_j+\sum_{i=1}^{m}y_j\times a_{i,j}\tag{4}$$

(4) 新基解

新基解 $x_B=B^{-1}b$,设 $x_B=(x_{B1},x_{B2},\cdots,x_{Bm})$,$B^{-1}=[\lambda_{i,j}]_{m\times m}$,$b=(b_1,b_2,\cdots,b_m)$,则 x_B 计算公式如下:

$$x_{Bj}=\begin{cases}\sum_{\forall j\in U}\lambda_{i,j}\times b_j+b_i, & i\notin U\\ \sum_{\forall j\in U}\lambda_{i,j}\times b_j, & i\in U\end{cases}\tag{5}$$

3　算法流程

在上述压缩存储策略和相应运算规则基础上,我们给出算法 GCRM(compressed revised simplex method based on GPU)的流程,如图 1 所示.CPU 和 GPU 工作于同步模式,即 CPU 调用 GPU 像素程序时,等待直到结果返回.

从图 1 中可以看出,GPU 的工作是完成单纯形乘子、逆矩阵及检验数和 θ 比值等复杂的计算,这些计算都适合 GPU 的流模型;而迭代过程中的结束判断、换入

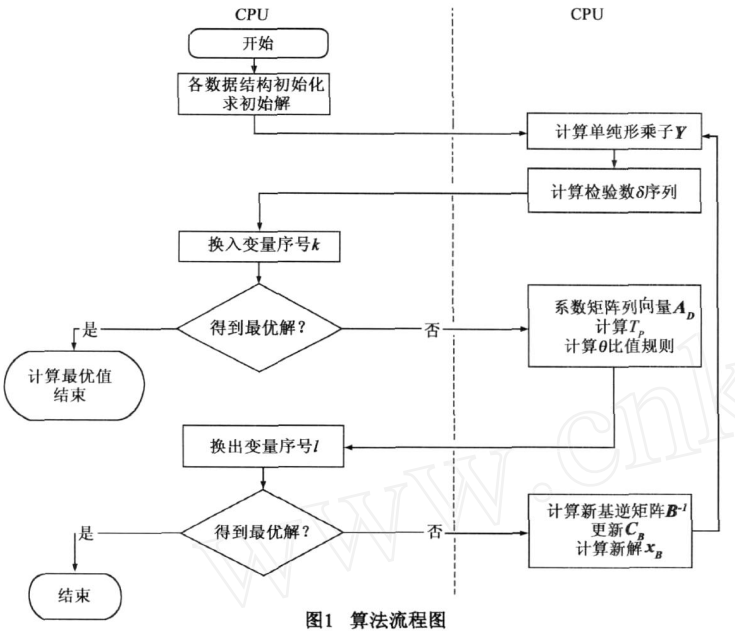

图1 算法流程图

和换出变量选择等操作都被分配给 CPU 执行. 在数据存储上,CPU 以动态数组形式存储,GPU 以静态纹理形式存储,这样的存储方式简化了编程接口,降低了虚拟地址和物理地址之间互相转换带来的复杂性. CPU 内存和 GPU 纹理间的数据传输是 GPU 通用计算带来的负面影响,我们尽量减少传输发生的次数和降低单次传输的数据量:迭代过程中的数据传输仅包括检验数和 θ 比值序列,这两者都是一维向量,具有 $O(n)$ 的空间复杂度;具有空间复杂度 $O(n^2)$ 的矩阵,如系数矩阵,都只在初始化或得到最优解后一次性地进行数据传输.

4 性能分析

4.1 复杂度分析

定理 4.1 若系数矩阵 A 的秩为 m,则 GCRM 单次迭代的时间复杂度为 $O(mk)$,其中 $k=|U|$.

证明 GCRM 的主要计算量是计算基逆矩阵,该步骤的时间复杂度为单次迭代的时间复杂度. 若 $l\notin U$,根据式(1),计算基逆矩阵需要 $m(k+1)$ 次乘法运算和 $(m-1)k$ 次加法运算;否则,$l\in U$,根据式(2),需要 mk 次乘法运算和 $m(k-1)$ 次加法运算. 综上,整个算法单次迭代的时间复杂度是 $O(mk)$.

若采用直接方法计算逆矩阵需要 $O(m^3)$ 次算术运算,GCRM 即使在 k 增长到 m 后,时间复杂度也能够降低 m 的一次幂.

定理 4.2 若系数矩阵 A 的稀疏度为 α^*,则 GCRM 的空间复杂度为 $O(\alpha mn)$.

证明 系数矩阵 A 的存储决定了整个算法的空间复杂度,本算法只存储非零元,占用 αmn 个浮点数存储单元. 此外,使用 $2n$ 个整数记录 A 的每一个非零元的位置. 因而,算法的空间复杂度为 $O(\alpha mn)$.

算法计算过程中,基逆矩阵 B^{-1} 和初等矩阵 E 也采用了压缩存储策略,节省了存储空间.

4.2 数值实验

本节给出压缩存储策略下算法的 CPU 版本、GPU 版本(GCRM)及 MATLAB R2007a(以下简称 MATLAB)求解线性规划问题的效果和性能比较. 测试样本为随机赋值为[0,1]上的浮点数,问题规模为 n,即 A 为 $n\times 2n$ 的矩阵、b 为 $n\times 1$,c 为 $1\times 2n$. 实验是在配有 Pentiun Ⅳ 3.4GHz CPU(主存 2GB)和 ATI FireStream 2U GPU(显存 1GB)的 PC 机上进行的.

设系数矩阵的稀疏度为 α,表 1 的系数矩阵 A 分为不稀疏($\alpha=0.0$)、20 %稀疏($\alpha=0.2$)、40 %稀疏($\alpha=0.4$)和 60 %稀疏($\alpha=0.6$)四种情况. 表中分别给出了每一规模在同样的测试用例下的最终优化值和运行时间. 由于 CPU 和 GPU 对浮点数的处理精度不同[11],优化值取小数点后 3 位,运行时间以秒为单位. CPU 版本和 GPU 版本中还给出了算法迭代次数,以求证明 GPU 版本算法的正确性;而 MATLAB 会根据输入问题的性质自动选择一个较优算法而不仅使用单纯形方法**,迭代次数不作比较. 每一问题规模生成 10 个计算实例,取其各项指标的均值.

表 1 运算结果比较

(1) $\alpha=0.0$

问题规模	MATLAB		CPU 版本			GPU 版本		
	优化值	时间	迭代次数	优化值	时间	迭代次数	优化值	时间
200	0.001	0.96	5	0.001	1.63	5	0.001	0.85
500	0.026	15.05	7	0.026	27.53	7	0.026	5.14
800	0.014	19.62	10	0.014	171.68	10	0.014	6.77
1000	0.019	25.33	21	0.019	707.24	21	0.019	7.22
1500	0.004	84.86	12	0.004	1584.88	12	0.004	10.77
2000	0.011	241.52	18	0.011	5664.62	17	0.011	19.66

(2) $\alpha=0.2$

问题规模	MATLAB		CPU 版本			GPU 版本		
	优化值	时间	迭代次数	优化值	时间	迭代次数	优化值	时间
200	0.065	3.02	16	0.065	3.80	16	0.065	1.54
500	0.032	13.92	15	0.032	54.46	15	0.032	2.34
800	0.020	19.44	22	0.020	365.00	22	0.020	5.64
1000	0.013	25.38	29	0.013	959.61	29	0.013	8.31
1500	0.003	55.75	22	0.003	2772.36	23	0.003	11.12
2000	0.010	151.63	30	0.010	9062.94	29	0.010	34.09
2500	0.008	178.97	30	0.008	13688.66	31	0.009	51.25

* 矩阵中零元素所占比例.

** MATLAB limprog 对大型优化问题采用 LIPSOL,而中型问题使用投影法.

(3) $\alpha=0.4$

问题规模	MATLAB		CPU 版本			GPU 版本		
	优化值	时间	迭代次数	优化值	时间	迭代次数	优化值	时间
200	0.091	1.53	12	0.091	2.89	12	0.091	1.23
500	0.031	15.47	9	0.031	33.06	9	0.031	1.73
800	0.025	11.78	23	0.025	82.91	23	0.025	2.77
1000	0.017	35.06	44	0.017	1452.68	51	0.017	10.43
1500	0.005	116.63	23	0.005	2896.52	23	0.005	10.74
2000	0.009	113.94	26	0.009	6008.78	26	0.010	17.78
2500	0.006	164.03	-	-	-	21	0.005	26.90
3000	0.009	393.64	-	-	-	26	0.009	69.08

(4) $\alpha=0.6$

问题规模	MATLAB		CPU 版本			GPU 版本		
	优化值	时间	迭代次数	优化值	时间	迭代次数	优化值	时间
200	0.120	1.44	30	0.120	6.91	30	0.120	2.53
500	0.080	15.22	37	0.080	132.72	37	0.080	4.45
800	0.057	20.53	54	0.057	891.64	54	0.057	9.96
1000	0.032	32.67	121	0.032	3986.67	121	0.032	21.78
1500	0.048	126.75	-	-	-	84	0.048	15.32
2000	0.035	199.16	-	-	-	114	0.035	34.70
3000	0.030	679.08	-	-	-	70	0.032	42.88
4000	/	/	-	-	-	/	/	/

表 1 中(1)所测试的问题,系数矩阵 A 是非稀疏的,最多可处理 2000 个变量的问题,较非压缩策略多近 600 个变量*. (2)～(4)根据稀疏比例的不同,*GPU* 版本能够求解更大规模的问题.'-'代表 5 个小时(18000 秒)以上没有计算出来结果,'/'代表没有可行解.

当问题规模在 1000 以下时,三者得到了截取到小数点后 3 位的相同的优化值,而当超过一定的规模,*GPU* 和 *CPU* 的迭代次数和优化值有时会出现一些偏差,这是由于目前的 *GPU* 对浮点数的运算还不符合 *IEEE*-754 标准[12]. 当达到 4000 变量时,*MATLAB* 和 *GPU* 版本都未能得到解,这证明了 *GPU* 版本不仅在计算最优值上是有效的,在发现问题无解上也是正确的.

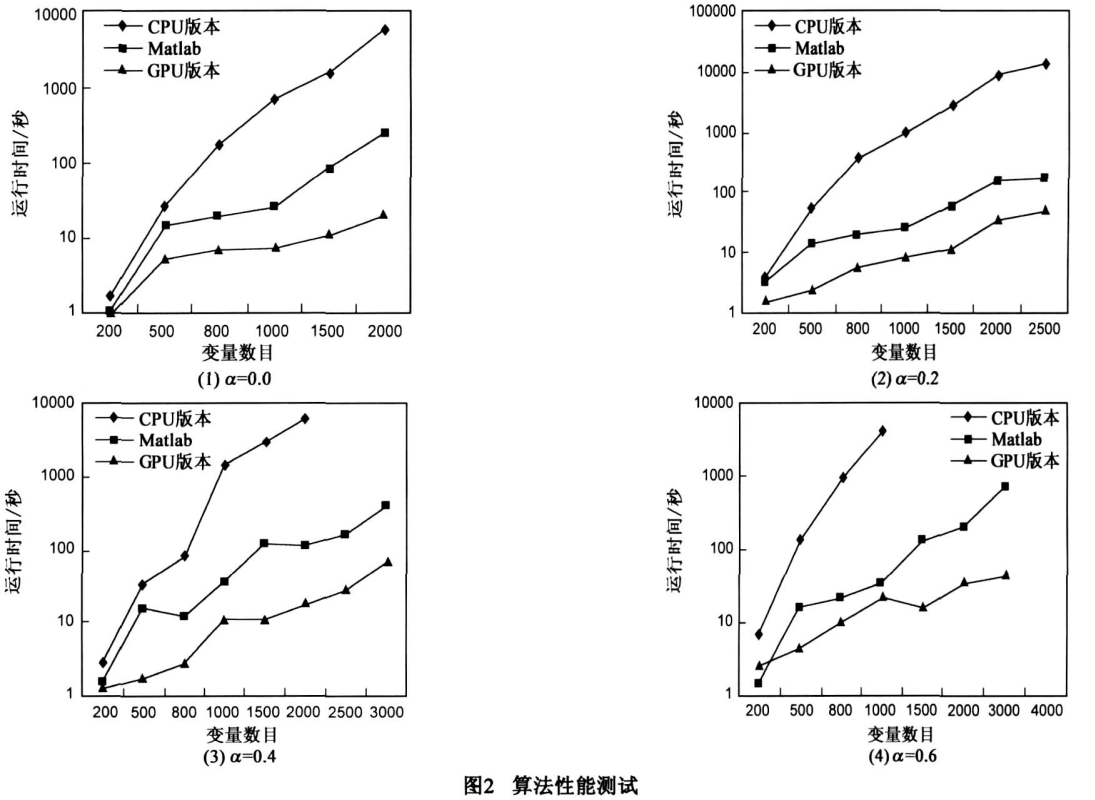

图2　算法性能测试

上表实验结果中 *MATLAB*、*CPU* 版本和 *GPU* 版本的性能对比如图 2 所示. 由于运算效率相差悬殊,比较使用了对数图. *GPU* 版本在所有的测试样例下,明显优于 *CPU* 版本. 尽管 *MATLAB* 针对 *CPU* 进行了大量的底层优化,如 *JIT* 即时编译和向量化技术等等,使其比 *C* 标准写的 *CPU* 版本代码快很多,但是当问题规模达到 2000 时,*GPU* 版本比 *MATLAB* 也快了 5 倍以上.

图 3 对比了 *GPU* 上非压缩策略和压缩策略的求解性能,测试样例均为系数矩阵 A 不稀疏. 显然即使同样解决不稀疏问题,压缩策略在性能上也是占优的.

5　结语

本文以研究了一种适于 GPU 的压缩单纯形方法求解线性规划问题,文[13]完成了 GPU 上的内点法,也取得了性能的提升,不同的是该文使用的是下三角矩阵,

*每个单精度浮点数占 4 字节,2000 ×(2000 + 1) ×4≈4096 ×4096.

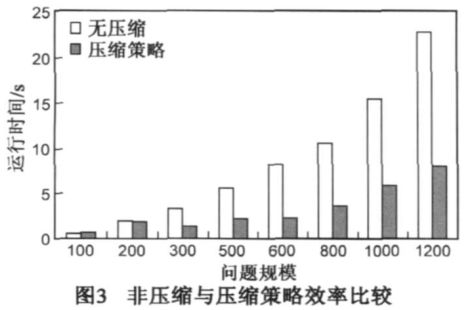

图3 非压缩与压缩策略效率比较

没有针对稀疏问题进行优化.我们下一步工作是利用分支定界方法,研究基于 GPU 的整数规划问题求解;单纯形方法在另一 GPU 通用计算平台——NVIDIA 的统一计算架构[14]上的实现并与之比较也是有意义的.随着 Stream 1.3 的推出,AMD 的 GPU 和 CPU 将走向融合(fusion),以 GPU 为主的多核计算展现出光明的前景.

参考文献:

[1] G M Wang, Z P Wan, X J Wang, et al. Genetic algorithm based on simplex method for solving linear-quadratic bilevel program-ming problem [J]. Computers & Mathematics with Applications, 2008, 56(10): 2550 - 2555.

[2] 王金根,龚沈光.基于运动标量磁强计的磁性目标定位问题研究[J].电子学报. 2002, 30(7): 1057 - 1060.
Wang Jin-gen, Gong Shen-guang. Research on the Problem of Localizing Magnetic Target Based on Motion Scalar Magnetometer [J]. Acta Electronical Sinica, 2002, 30(7): 1057 - 1060. (in Chinese)

[3] K Paparrizos. An infeasible exterior point simplex algorithm for assignment problems [J]. Mathematical Programming, 1991, 51(1 - 3): 45 - 54.

[4] K Paparrizos, N Samaras, G Stephanides. A new efficient primal Dual Simplex algorithm [J]. Computers and Operations Research, 2003, 30(9): 1383 - 1399.

[5] H Luh, R Tsaih. An efficient search direction for linear programming problems [J]. Computers and Operations Research, 2002, 29(2): 195 - 203.

[6] H B Junior, M P E Lins. An improved initial basis for the simplex algorithm [J]. Computers and Operations Research, 2005, 32(8): 1983 - 1993.

[7] 吴恩华.图形处理器用于通用计算的技术、现状及其挑战[J].软件学报, 2004, 15(10): 1493 - 1503.
Wu En-hua. State of the Art and Future Challenge on General Purpose Computation by Graphics Processing Unit [J]. Journal of Software, 2004, 15(10): 1493 - 1503. (in Chinese)

[8] 袁友伟.采用 GPU 加速的三维实体模型绘制[J].电子学报, 2008, 36(12A): 144 - 146.
Yuan You-wei. 3D Solid Models Rendering Based on GPU Acceleration [J]. Acta Electronical Sinica, 2008, 36(12A): 144 - 146. (in Chinese)

[9] T T Wong, C S leung, P A Henq, et al. Discrete wavelet transform on consumer-level graphics hardware [J]. IEEE Transactions on Multimedia, 2007, 9(3): 668 - 673.

[10] 杨正龙,金林,李蔚清.基于 GPU 的图形电磁计算加速算法[J].电子学报, 2007, 35(6): 597 - 601.
Yang Zheng-long, JiN Lin, Li Wei-qing. Accelerated GRECO Basedon GPU [J]. Acta Electronical Sinica, 2007, 35(6): 597 - 601. (in Chinese)

[11] P Mark, S Mark, G Derek. A Performance-Oriented Data Parallel Virtual Machine for GPUs [C]. In Proccedings of SIGGRAPH 2006, 2006: 184 - es.

[12] Ati Research. The Radeon X1x00 Programming Guide. www.ati.com, 2006.

[13] J H Jung, D P. O'leary. Implementing an Interior Point Method for Linear Programs on a CPU-GPU System [J]. Electronic Transactions on Numerical Analysis. 28, 2008, 174 - 189.

[14] Nvidia. NVIDIA CUDA Compute Unified Device Architecture-Programming Guide. 2008, http://developer.download.nvidia.com/compute/cuda.

作者简介:

白洪涛 男,1975 年生于吉林榆树市,在读博士生,主要研究方向为决策支持系统,高性能计算.
E-mail: baihongtao@263.net

欧阳丹彤 女,1968 年生于吉林长春市,教授,博士生导师,主要研究方向为基于模型的诊断和定理机器证明.
E-mail: ouyangdantong@163.com

何丽莉 女,1976 年生于吉林洮南市,博士,主要研究方向为无线传感器网络,高性能计算.
E-mail: helili@jlu.edu.cn