一种优化高维复杂函数的 PSO 算法*)

雷开友[1] 邱玉辉[1] 贺 一[1,2]

(西南大学计算机与信息科学学院 重庆 400715)[1] (重庆师范大学现代管理学院 重庆 400047)[2]

摘 要 对于高维复杂函数,一般粒子群优化算法收敛速度慢,易早熟收敛。本文重构一个适合高维复杂函数惯性权重函数,使粒子群算法寻优过程中的全局收搜能力和局部收搜能力良好平衡,以达到快速收敛,高效避免早熟问题,获得最优解。对典型高维复杂函数的仿真表明:算法在求解质量和求解速度两方面都得到了好的结果。

关键词 粒子群优化,惯性权重,早熟收敛问题

An Effective Particle Swarm Optimizer for Solving Complex Functions with High Dimensions

LEI Kai-You[1] QIU Yu-Hui[1] HE Yi[1,2]

(Faculty of Computer & Information Science, Southwest University, Chongqing 400715)[1]

(School of Management, Chongqing Normal University, Chongqing 400047)[2]

Abstract For complex functions with high dimensions, general particle swarm optimization methods are slow speed on convergence and easy to be trapped in local optima. This paper proposes an efective particle swarm optimizer, which can automatically select a preferably inertia weight curve for the complex functions according to the fitness change ratio of swarm, and to balance global and local search ability, fasten convergence speed, avoid premature problem, and obtain global optimum. Experimental results on several benchmark complex functions with high dimensions show that the algorithm can rapidly converge at high quality solutions.

Keywords Particle swarm optimization, Inertia weight, Premature problem

1 引言

20 世纪 90 年代中期,Eberhart 博士和 Kennedy 博士共同发明了一种新的群体智能计算技术,粒子群优化算法(Particle Swarm Optimization, PSO),其源于对鸟群和鱼群群体运动行为的研究。该算法已经广泛应用于函数优化、神经网络训练、模糊系统控制[1~3]等领域。

近年来的研究已越来越多地表明,粒子群优化算法对函数优化有强大的生命力,特别是对一些规模大、维数高、非线性、非凸和不可微等特性的函数进行优化,因为许多传统确定性优化算法及全局性的优化算法(如遗传算法等)易早熟收敛,对于高维复杂函数难以实现高效优化。本文以几种典型高维复杂函数为测试算例,提出了一种重构惯性权重函数粒子群优化算法,协调好全局收搜能力与局部收搜能力,快速收敛,高效避免早熟收敛问题,实验结果表明,解的质量好,验证该算法高效可行。

2 一种粒子群优化算法的自适应综合策略

2.1 粒子群优化算法及早熟收敛问题

粒子群优化标准 PSO 算法的进化方程为:

$$v = w * v + c_1 * rand * (pBest - Present) + c_2 * rand * (gBest - Present) \quad (1)$$

$$Present = Present + v \quad (2)$$

其中,v 是粒子速度,Present 是粒子当前位置,$pBest$ 表示一个粒子本身所找到的最优解,被称为个体极值,$gBest$ 表示整个粒子群目前找到的最优解,被称为全局极值;w 是惯性权重,一般在 0.9 到 0.1 之间取值;c_1,c_2 被称作学习因子,通常均为 2 的常数;$rand \in [0,1]$ 是均匀分布的随机数。算法首先初始化一群随机粒子,然后迭代运行,在每一次迭代中,粒子通过跟踪两个"极值"来更新自己,最终找到最优位置。如果该最优位置为一局部最优点,粒子群就无法在解空间内重新搜索,因此算法陷入局部最优,出现了与其它算法(如遗传算法)存在的早熟收敛现象,特别是复杂的高维多峰搜索问题,早熟收敛现象较普遍。

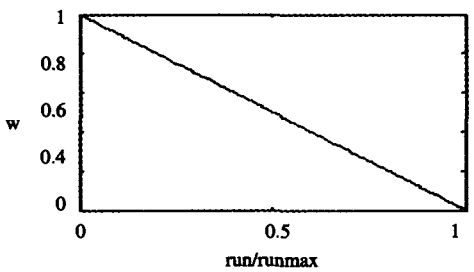

图 1 式(3)惯性权重曲线

为了加快收敛速度,避免早熟收敛,人们提出了多种方法,其中 Shi[4] 等提出了一种惯性权重的方法,设 w_{max} 为最大惯性权重,w_{min} 为最小惯性权重,run 为当前迭代次数,$runmax$ 为算法迭代总次数,则有惯性权重公式:

$$w = w_{max} - (w_{max} - w_{min}) * \frac{run}{run\ \max} \quad (3)$$

使惯性权重 w 随算法迭代的进行而线性减小,见图 1。该方

*)本文受到教育部科学技术重点项目(No. 104262)和重庆市科委基金项目(2003-7881)共同资助。雷开友 副教授,博士研究生,研究方向为神经网络与计算智能。邱玉辉 教授,博士生导师,长期从事非单调推理、近似推理、神经网络、机器学习和分布式人工智能的教学和研究工作。贺 一 副教授,博士研究生,研究方向为神经网络与计算智能。

法加快了收敛速度，提高了算法的性能。当待解问题很复杂时，该法使得 PSO 在迭代后期全局搜索能力不足，导致不能找到要求的最优解[5]。

2.2 一种重构惯性权重函数粒子群优化算法的提出

(1)重构惯性权重函数粒子群算法的基本思想

在式(1)中，第一项对应多样化(diversification)的特点，第二项、第三项对应于搜索过程的集中化(intensification)特点[5]，即当 w 越大，全局搜索能力越强；当 w 越小局部搜索能力越强；当 $w=0$ 时，第一项为零，全局搜索能力减为最弱，而局部搜索能力加到最强；在式(3)中，w 先是最大，后是由大变小，最后为最小，显然 w 最大最小所经历的时间较短，全局搜索和精细的局部搜索时间不够，用式(3)作 PSO 的惯性权重去优化复杂的高维函数找不到要求的最优解；文[4～9]通过大量实验证明，对所有优化对象而言：第一，较大惯性权重 w 可以加强 PSO 的全局搜索能力，即探索较大的区域，较快地定位最优解的大致位置，第二，较小的惯性权重 w 能加强 PSO 局部搜索能力，即粒子速度减慢，开始精细的局部搜索；第三，惯性权重 w 都是由大到小变化。

根据上面的分析，越是难优化的函数，越需要加强全局搜索能力，一但定位到最优解的大致位置，越需要加强局部搜索能力，即加强精细的局部搜索。因此，本文重构一个粒子群优化算法惯性权重函数，它由三段构成：w 先保持最大，后是由大变小过渡，最后保持近似零的最小，这样，协调好何时进行全局收搜，何时进行局部收搜，以及收搜时的强度，使全局收搜能力与局部收搜能力良好平衡，快速收敛得到最优解。这个重构的惯性权重函数为：

$$w=\exp(-30*(run/run\ \max)\wedge(s)) \tag{4}$$

其中 S 为大于 1 的整数，本文 S 取值范围[1,30]，run 为当前迭代次数，$runmax$ 为算法迭代总次数，S 不同时有不同条曲线，见图 2，给出了四条曲线；当 S 确定后，式(4)表达的曲线就确定了，令式(4)代入式(1)，粒子群优化算法就可以按某条惯性权重曲线进行运算了。这条惯性权重曲线由下面算法自动确定。

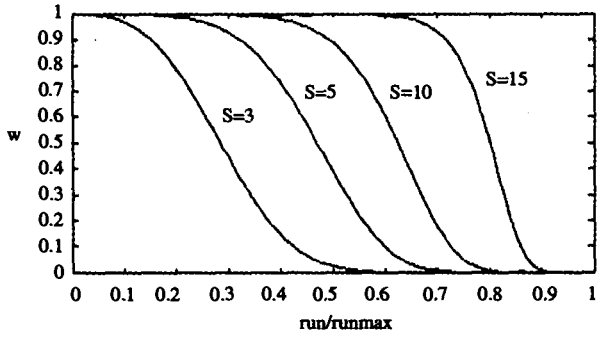

图 2 式(4)惯性权重曲线

对图 2 中每一条惯性权重曲线而言，式(3)的惯性权重直线类似它的中段，即式(3)的惯性权重直线为式(4)惯性权重曲线之一的特例。因此，用式(4)惯性权重曲线之一优化高维复杂函数会得到较好的效果。

(2)算法设计

算法设计的目的是为高维复杂函数自动寻找一条自己较好的惯性权重下降曲线。方法是：把有 n 个粒子的粒子群均分为 h 个子粒子群，h 为惯性权重曲线总数，如图 1，有 4 个 S 值(3,5,8,15)，就有 4 条惯性权重曲线，即 $h=4$；若粒子群的粒子数为 40 个，则 10 个粒子组成一个子粒子群，共 4 个子粒子群；算法开始运行时，每个子粒子群按各自的惯性权重曲线独立运行，即第一个子粒子群按 $S=3$ 的这条曲线运行，第二个子粒子群按 $S=5$ 的这条曲线运行，依此类推，每间隔一定的迭代步数，测出每个子粒子群的全局极值比较，取有最好极值的子粒子群替代有最差极值的子粒子群，最差子粒子群的惯性权重曲线就去掉了，最差子粒子群按最好子粒子群的惯性权重曲线运行，此时有 3 个子粒子群按 3 条惯性权重曲线独立运行，如此运行下去，直到满足算法收敛准则。这种方法被称为重构惯性权重函数粒子群算法。

(3)算法的优点

第一，根据图 1 函数曲线分析，式(4)是对式(3)的扩展，协调好了算法的开始阶段搜索较大的解空间的时间及最后阶段精细搜索时间；第二，一个优化对象在算法运行过程中，以最大概率自动寻找一条自己较好的惯性权重下降曲线。

2.3 算法步骤

其算法流程如下：

1)算法参数设定，其中有一组 S 参数，$S1\cdots Si\cdots Sh$，共 h 个；

2)随机初始化粒子群中的位置与速度；

3)将整个粒子群均分为 h 个子粒子群；

4)对所有子粒子群根据式(1)、式(2)分别按典型粒子群算法开始运行，其中第一个子粒子群按 $S1$ 参数的惯性权重曲线运行，第 i 个子粒子群按 Si 参数的惯性权重曲线运行；

5)判断算法收敛准则是否满足，如果满足，转向 8；否则，执行 6；

6)每间隔一定的迭代步数，检测每个子粒子群的全局极值比较，把有最差极值子粒子群用有最好极值的子粒子群替代，整个粒子群变为 $h-1$ 个子粒子群，$h-1$ 个子粒子群继续按 $h-1$ 惯性权重曲线运行；

7)判断算法收敛准则是否满足，如果满足，执行 8；否则，转向 6；

8)输出 Si 及相应的 $gBest$，算法运行结束。

其中初始化粒子群中的位置与速度，用均匀分布随机数在函数的定义域范围产生。

3 计算机仿真实验

3.1 测试算例

本文测试算例为四个典型函数优化问题(求解最小值)[9,10]，其中 Sphere 函数是简单的单峰函数，Rosenbrock 函数是极难最小值非凸的病态函数，Generalized Griewank 函数和 Ackley 函数是多峰函数，极难找到全局最优解。

Sphere 函数：

$$f_1(x)=\sum_{i=1}^{n}x_i,\ -100\leqslant x_i\leqslant+100 \tag{7}$$

Rosenbrock 函数：

$$f_2(x)=\sum_{i=1}^{n}[100*(x_i^2-x_{i+1})^2+(1-x_i)^2],\ -30\leqslant x_i\leqslant+30 \tag{8}$$

Generalized Griewank 函数：

$$f_3(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1\quad -600\leqslant x_i\leqslant+600 \tag{9}$$

Ackley 函数：

$$f_4(x)=-20\exp\left(-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i 2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e \quad -30\leqslant x_i\leqslant +30 \qquad (10)$$

3.2 算法参数设置

本文算法参数设置是：学习因子 $c1=c2=2$（其中 $f_2(x)$ 的 $c1=c2=1$）；迭代总次数 $runmax=1000$，间隔迭代步数 $m=20$（$f_2(x)$ 的 $m=60$）；粒子群总粒子数在 30 维时 200 个，在 60、90 维时 400 个；10 条惯性权重曲线，即式(4)中 S 为 3、5、7、8、9、10、11、13、17、20 时的曲线；速度、位置的最大值限制在函数的定义域范围内；取 $f_1(x)$、$f_2(x)$、$f_3(x)$、$f_4(x)$ 函数表达式为适应度函数。

3.3 两种惯性权重下的算例仿真结果比较

对于每个算例按 10 维，用 MATLAB6.5 编程，在 PC (PentiumIV-2.8GHzCPU，256M RAM，WinXP OS) 上均运行 20 次，得到 20 个结果，全部计算结果的统计如表 1、表 2 所示。

表 1 用式(3)作惯性权重的标准算法仿真结果统计

函数	理论最优解	误差	最优解	平均最优解	平均收验代数	平均收验率%
$f_1(x)$	0	10^{-7}	0	0	34.28	100
$f_2(x)$	0	10^{-7}	0	0	20.56	100
$f_3(x)$	0	10^{-7}	0	4.5450	943.72	10
$f_4(x)$	0	10^{-7}	0	1.32993	563.88	40

表 2 用式(4)作惯性权重的本文算法仿真结果统计

函数	理论最优解	误差	最优解	平均最优解	平均收验代数	平均收验率%
$f_1(x)$	0	10^{-7}	0	0	32.35	100
$f_2(x)$	0	10^{-7}	0	0	16.34	100
$f_3(x)$	0	10^{-7}	0	0	632.27	100
$f_4(x)$	0	10^{-7}	0	0	61.72	100

从表 1、表 2 得出：用式(4)作惯性权重的本文算法好于用式(3)作惯性权重的标准算法仿真结果。

3.4 本文算法应用于高维算例仿真结果

对于每个算例按 30 维，60 维，100 维，在上述 pc 上均运行 50 次，得到 50 个结果，全部计算结果的统计如表 3、表 4、表 5 所示（表中输出 S 值为出现概率最大的 S 值）。

表 3 30 维仿真结果统计

函数	理论最优解	误差	最优解	平均最优解	平均收验代数	平均收验率%	输出 S 值
$f_1(x)$	0	$1*10^{-5}$	0	0	13.725	100	8
$f_2(x)$	0	$1*10^{-5}$	0	0	670.33	100	10
$f_3(x)$	0	$1*10^{-5}$	0	0	18.9	100	9
$f_4(x)$	0	$1*10^{-5}$	0	0	15.3	100	8

表 4 60 维仿真结果统计

函数	理论最优解	误差	最优解	平均最优解	平均收验代数	平均收验率%	输出 S 值
$f_1(x)$	0	$1*10^{-5}$	0	0	18.457	100	8
$f_2(x)$	0	0.1	2.7242e−5	0.041418	444.02	100	10
$f_3(x)$	0	$1*10^{-5}$	0	0	27.625	100	9
$f_4(x)$	0	$1*10^{-5}$	8.8818e−16	5.9793e−12	19.75	100	9

表 5 100 维仿真结果统计

函数	理论最优解	误差	最优解	平均最优解	平均收验代数	平均收验率%	输出 S 值
$f_1(x)$	0	$1*10^{-5}$	0	0	27.2	100	9
$f_2(x)$	0	0.1	9.7965e−6	0.05095	494.2	96	10
$f_3(x)$	0	$1*10^{-5}$	0	1.1102e−16	29.2	100	10
$f_4(x)$	0	$1*10^{-5}$	8.8818e−16	5.5369e−11	24.5	100	10

从表 3、表 4、表 5 可以得出：(1)本文算法对 $f_1(x)$、$f_2(x)$、$f_3(x)$、$f_4(x)$ 高维复杂函数优化效果好；(2)输出 S 值为 8、9 或 10 等的惯性权重曲线对高维复杂函数优化最佳。

3.5 S 参数对应的惯性权重曲线选择规律的仿真结果

为了进一步确定 S 参数的选择规律，即确定惯性权重曲线的选择规律，本文又对 S 参数为 1 到 30 的 30 条惯性权重曲线，在其它条件不变的情况下，把每条惯性权重曲线分别代入式(1)作仿真实验，平均运行 50 次，得到一个平均收验代数，列出典型 $f_1(x)$、$f_2(x)$ 在 S 参数为 1 到 15 的 30、60、100 维平均收验代数的统计结果，见表 6 所示（表中 $f_1(30)$ 指 30 维时平均收验代数，其它依此类推）。

从表 6 可以得出：(1)进一步验证了本文重构适合高维复杂函数惯性权重 PSO 算法思想的正确性；(2)选 S 参数大于 4 的任意一条惯性权重曲线对高维复杂函数优化均能得到较好的效果；(3)只要 S 参数代入式(4)，再把式(4)代入式(1)就可以运用典型粒子群算法对高维复杂函数优化，保持了 PSO 算法简单、易实现的优点。

结论 针对高维复杂函数的优化问题，本文提出的重构一个适合高维复杂函数惯性权重函数的粒子群算法效率高、优化性能好；另本算法也适合一般函数优化；如何对 PSO 算法进行理论分析，以及与其它方法相结合，并用之求解实际问题，将是进一步的研究工作。

参 考 文 献

1 Kennedy J, Eberhart R C. Particle Swarm Optimization. In: Proc. IEEE in-ternational Conf. on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, IV: 1995. 1942～1948

2 Shi Y H, Eberhart R C, Chen Y B. Design of Evolutionary Fuzzy Expert system [A]. In: Proc. of 1997 Artificial Neural Networks in Engineering Conf. St. Louis, Nov. 1997

3 Fukuyama Y. Fundamental of particle swarm techniques. Lee K Y, El-Sharkaei MA. Mo-dern Heuristic Optimization Techniques With Applications to Powr Systems. IEEE Power Engineering Society, 2002. 45～51

4 Shi Y, Eberhart R C. Empirical study of particle swarm optimization. In: Proc. of the Congress on Evolutionary Computation. Seoul, Korea, 2001

5 杨维，李歧强. 粒子群优化算法综述. 中国工程科学，2004，6(5)：87～93

6 Shi Y H, Eberhart R C. A Modified Particle Swarm Optimizer. In: IEEE In-ternational Conf. on Evolutionary Computation, Anchorage, Alaska, May 1998

7 Angeline P. Using Selection to Improve Particle Swarm Optimization. In: Proc. of IJCNN'99. Washington, USA, 1999. 84～89

8 Eberhart R C, Kennedy J. A New Optimizer Using Particles Swarm Theory. In: Proc. Sixth International Symposium on Micro Machine and Human Science (Nagoya, Japan), IEEE Service

Center, Piscataway, NJ, 1995. 39～43
9 Lei Kaiyou, Wang Fang, Qiu Yuhui, He Yi. An Adaptive Inertia Weight Strategy for Particle Swarm Optimizer [A]. In: The 3rd Intl. Conf. on Mechatronics and information technology [C]. Chongqing, China, Sep. 2005
10 吕振肃,侯志荣. 自适应变异的粒子群优化算法. 电子学报,2004,32(3):416～420
11 李炳宇,萧蕴诗,汪镭. 一种求解高维复杂函数优化问题的混合粒子群优化算法. 信息与控制,2004,33(1):27～30

表 6 $f_1(x)$、$f_2(x)$ 仿真结果统计

S参数	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$f_1(30)$	1000	17	17	16	17	17	17	14	15	15	15	15	15	15	15
$f_1(60)$	1000	22	17	18	18	17	17	15	16	16	16	18	17	18	18
$f_1(100)$	1000	141	123	23	23	23	23	25	23	22	21	20	23	27	28
$f_2(30)$	1000	1000	932	815	783	673	656	772	757	768	725	718	891	805	853
$f_2(60)$	1000	1000	1000	1000	745	818	784	696	773	697	755	801	805	809	854
$f_2(100)$	1000	1000	1000	1000	809	643	642	715	758	658	753	806	740	828	850

(上接第 172 页)

事务流(更新操作分别为 96837, 96571, 97217, 97575, 97011 和 97167 次),其平均 I/O 次数如图 2 所示,可见,ETPR-tree 略优于 TPR-tree,但 BiR-tree 明显优于其他两种索引结构(平均 I/O 次数为 6 次)。

执行 2400 次窗口查询的平均 I/O 次数如图 3 所示,可见,BiR-tree 和 TPR-tree 的查询性能相当,但 ETPR-tree 明显劣于其他两种索引结构。这是因为 ETPR-tree 非叶子结点占用存储空间较多,其扇出低于其他两种索引结构,同时插入、分裂等操作中对 IdBR 的处理使得内部结点的 TPBR 重叠增加。

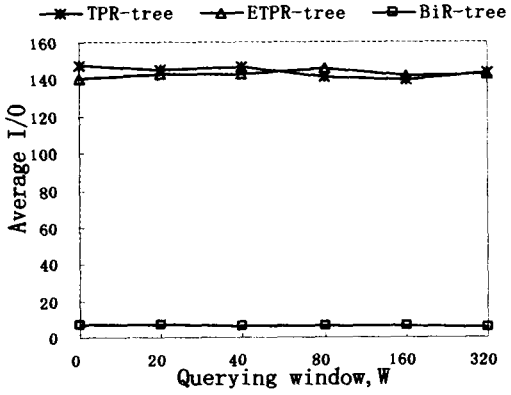

图 2 更新性能比较

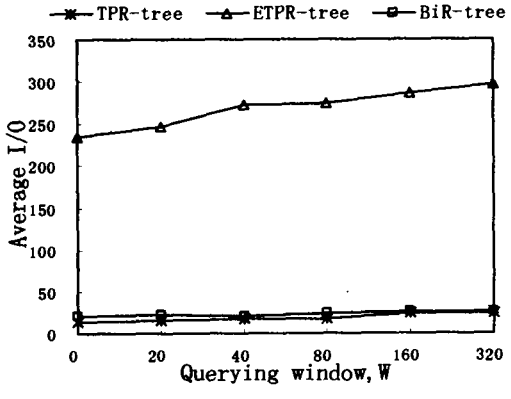

图 3 查询性能比较

实验结果表明,与 TPR-tree 相比,ETPR-tree 在更新方面性能略有提高,但其查询性能大幅度下降;BiR-tree 由于采用辅助索引结构,不仅具有良好的更新性能,同时保持原有 TPR-tree 的查询性能。因此,BiR-tree 索引结构是一种对 TPR-tree 有效的改进方法。

结论 在当前已提出的移动对象索引方法中,基本上都不支持对移动对象标识符的查询,由此导致索引结构的更新存在一定的缺陷。我们以 TPR-tree 为例,给出了两种改进方案(ETPR-tree 和 BiR-tree)。通过实验结果对比,BiR-tree 结构能在保证查询性能的前提下,有效地解决 TPR-tree 的更新缺陷。由于 BiR-tree 并不改变 TPR-tree 原有的算法和结构(对对象标识建立 B^+-tree 索引),因此这种建立辅助索引的方法可以应用到现有的其他索引结构中,解决其存在的更新缺陷。

参 考 文 献

1 Guttman A. R-trees: A dynamic index structure for spatial searching. In: Proc of SIGMOD, 1984. 47～57
2 张明波,陆锋,等. R 树家族的演变和发展. 计算机学报, 2005, 28(3): 289～300
3 Tao Y, Papadias D, Sun J. The TPR *-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries. In: Proc of VLDB, 2003. 790～801
4 Beckmann N, Kriegel H P, Schneider R, et al. The R *-tree: An Efficient and Robust Access Method for Points and Rectangles. In: Proc. of SIGMOD, 1990. 322～331
5 Hadjieleftheriou M, Kollios G, Tsotras V J, et al. Efficient Indexing of Spatiotemporal Objects. In: Proc of EDBT, 2002. 251～268
6 Kollios G, Tsotras V J, Gunopulos D, et al. Indexing Animated Objects Using Spatiotemporal Access Methods. TKDE, 2001, 13 (5): 758～777
7 Porkaew K, Lasaridis I, Mehrotra S. Querying Mobile Objects in Spatio-Temporal Databases. In: Proc. of SSTD, 2001. 59～78
8 Tao Y, Papadias D. MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries. In: Proc. of VLDB, 2001. 431～440
9 Agarwal P K, Arge L, Erickson J. Indexing Moving Points In: Proc. of PODS, 2000. 175～186
10 Jensen C S, Lin D, Ooi B C. Query and Update Efficient B+-Tree Based Indexing of Moving Objects. In: Proc. of VLDB, 2004. 768～779
11 Patel J M, Chen Y, Chakka V P. STRIPES: An Efficient Index for Predicted Trajectories. In: Proc. of SIGMOD, 2004. 637～646
12 Procopiuc C M, Agarwal P K, Har-Peled S. STAR-Tree: An Efficient Self-Adjusting Index for Moving Objects. In: Proc. of the Intl Workshop on Algorithm Engineering and Experiments, 2002. 178～193
13 Saltenis S, Jensen C S, Leutenegger S T, et al. Indexing the Positions of Continuously Moving Objects. In: Proc of SIGMOD, 2000. 331～342
14 Saltenis S, Jensen C S. Indexing of Moving Objects for Location-Based Services. In: Proc of ICDE, 2002. 463～472
15 TPR-tree. http://www.cs.auc.dk/TimeCenter/software.htm