基于布谷鸟搜索的多处理器任务调度算法

杨辉华[1,2] 张晓凤[1] 谢谱模[1] 韦向远[1]
(桂林电子科技大学广西信息科学实验中心 桂林 541004)[1] (北京邮电大学自动化学院 北京 100876)[2]

摘 要 多处理器系统在高性能计算中扮演着重要角色。为提高系统的并行性能,基于布谷鸟搜索算法,提出一种新的多处理器任务调度算法。该算法以全部任务的最晚完成时间最小为目标,利用基于任务优先权的编码方式使连续的布谷鸟搜索算法适用于离散的多处理器任务调度问题。实验结果表明,所提算法不仅求解质量高,而且求解速度最快,与目前广泛采用的遗传算法和粒子群算法相比其执行时间缩短超过60%。

关键词 多处理器,任务调度,布谷鸟搜索算法
中图法分类号 TP301.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.1.020

Multiprocessor Task Scheduling Method Based on Cuckoo Search Algorithm

YANG Hui-hua[1,2] ZHANG Xiao-feng[1] XIE Pu-mo[1] WEI Xiang-yuan[1]
(Guangxi Experiment Center of Information Science, Guilin University of Electronic Technology, Guilin 541004, China)[1]
(Automation School, Beijing University of Posts and Telecommunications, Beijing 100876, China)[2]

Abstract Multiprocessor system plays a key role in high performance computing. In order to improve the parallelism of the system, we proposed a new multiprocessor task scheduling algorithm, which is based on Cuckoo search(CS) algorithm. The algorithm takes the latest completion time of all tasks as the goal, and encodes based on task priority method to make continuous CS algorithm suitable for discrete task scheduling problem probably. The experimental results show that CS algorithm not only can obtain shortest task completion time, but also has the fastest solving speed. Its computation time is 60% lower than the widely used genetic algorithm and particle swarm optimization algorithm.

Keywords Multiprocessor, Task scheduling, Cuckoo search algorithm

1 引言

随着人们对高性能计算需求的不断提高,多处理器任务调度技术在许多科学及工程领域得到了广泛应用。多处理器任务调度是把复杂的应用任务划分成多个具有一定约束关系的子任务,合理地调度到多处理器系统的各个处理器上并行执行,以使全部任务的最晚完成时间最小。调度算法的优劣将严重影响多处理器系统的性能,一个好的调度算法能大大减小全部任务的最晚完成时间,一个低劣的调度算法可能导致多处理器系统效率低下,甚至其性能不如单个处理器,严重的可能导致任务执行失败。因此,在多处理器系统中,提供一种高效的任务调度算法是提高此系统性能的关键所在。

多处理器任务调度问题是NP完全的组合优化问题,长期以来针对该问题的研究主要集中在满足时间复杂度的前提下获得近似最优解的启发式算法,其中比较典型的有表调度技术(List Scheduling)[1-4]、遗传算法(Genetic Algorithm, GA)[5-9]及粒子群算法(Particle Swarm Optimization, PSO)[10-14]。表调度技术尽管时间复杂度和空间复杂度都较低,但其缺点是无法保证解的质量。遗传算法和粒子群算法属于智能优化算法,通过对邻域的不断搜索和当前解的不断改进,能够在可接受的时间范围内以较大概率获得最优解或满意解,在求解多处理器任务调度问题中得到了广泛应用。但遗传算法具有操作复杂、难以实现、实时性较差等缺点;粒子群算法操作简便,但由于本身机制更新的限制,存在早熟收敛和后期振荡等缺点。

布谷鸟搜索算法(Cuckoo Search, CS)是由Yang等[15]在2009年提出的一种新的智能优化算法,该算法模拟了自然界中布谷鸟寄生孵育雏鸟的巢寄生行为。该算法具有模型简单、可调参数少及收敛速度快等优点,因此目前在函数优化领域[15,16]及组合优化领域[17]都得到了初步应用,但将该算法用于解决多处理器任务调度问题还未见文献报道。基于此,本文基于布谷鸟搜索算法提出了一种新的多处理器任务调度算法,并且为了验证算法的有效性,将其与遗传算法和粒子群算法进行了对比验证,实验结果表明该算法具有更高的调度性能。

到稿日期:2013-12-26 返修日期:2014-03-15 本文受国家自然科学基金项目(21365008,61105004),广西自然科学基金资助项目(2012GXNSFAA053230,2013GXNSFBA019279),广西信息科学实验中心重点基金项目(2012-02),广西高等学校优秀人才资助计划项目(桂教人[2011]40号)资助。

杨辉华(1972—),男,博士,教授,博士生导师,CCF高级会员,主要研究方向为机器学习、高性能计算、近红外光谱分析,E-mail:13718680586@139.com;**张晓凤**(1988—),女,硕士生,主要研究方向为高性能计算、机器学习;**谢谱模**(1989—),男,硕士生,主要研究方向为高性能计算;**韦向远**(1988—),男,硕士生,主要研究方向为高性能计算、图像处理。

2 任务调度问题描述

调度问题通常用任务图来描述。这种任务图是一种加权有向无环图(DAG),节点表示任务,边反映了任务间的数据依赖[18]。图1给出一个具有9个任务的DAG用例图,其中节点圆圈内的数据表示任务编号,旁边数据表示任务处理时间,有向边上的数据表示通信开销。例如,T1是T2的先序任务,它们的处理时间分别为2和3,若它们不在同一个处理器上处理,则通信开销为4,否则为0。

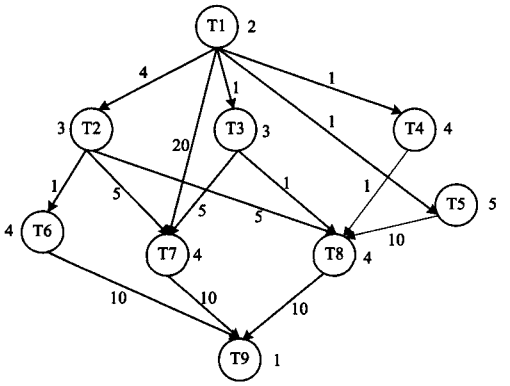

图1 具有9个任务的DAG用例图

本文研究的多处理器任务调度问题可描述为:在 m 个同构处理器上处理 n 个任务,寻找最优的任务调度序列,使全部任务的最晚完成时间最小,并且满足如下约束条件:①同一时刻每个处理器只能处理一个任务;②每个任务只能由一个处理器处理;③n 个任务之间要满足依赖关系,即一个任务若存在先序任务,则必须待其先序任务全部完成之后才可能开始被处理器调度处理,且如果它与其先序任务不在同一个处理器上处理,还需考虑通信开销。多处理器任务调度的直观描述如图2所示,其数学描述见式(1)。

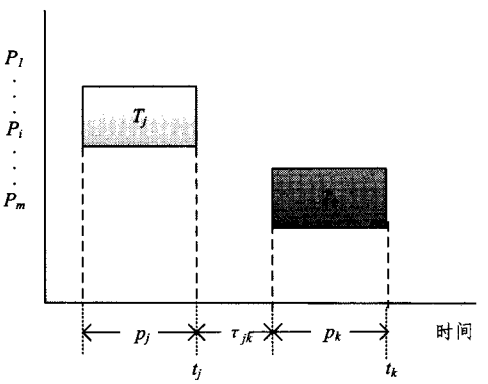

图2 DAG时间图表

$$\begin{aligned}
&\min\ f=\max_{\forall j}\{t_j\}\\
&\text{s.t.}\sum_{i=1}^{m}\sum_{r}^{R}x_{ijr}=1,\forall j\\
&\quad\sum_{j=1}^{n}x_{ijr}\leqslant 1,\forall i,r\\
&\quad t_k-p_k-d_{jk}\geqslant t_j,T_j\succ T_k,\forall k,\forall j\in pre(k)\\
&\quad t_j\geqslant 0,\forall j
\end{aligned}\tag{1}$$

其中,i 为处理器下标,$i=1,2,\cdots,m$;j,k 为任务下标,$j,k=1,2,\cdots,n$;f 为全部任务的完成时间;t_j 为决策变量,表示任务 T_j 的完成时间;R 为时间段的集合,$r\in R$;$x_{ijr}=\begin{cases}1, & \text{如果处理器 } P_i \text{ 在第 } r \text{ 个时间段处理任务 } T_j\\ 0, & \text{否则}\end{cases}$;$p_j$ 为任务 T_j 的处理时间;τ_{jk} 为任务 T_j 与任务 T_k 之间的通信时间;$\succ$ 为代表任务之间的依赖关系,$T_j\succ T_k$ 表示 T_j 是 T_k 的前序;$pre(j)$ 为任务 T_j 的前序集合;$d_{jk}=\begin{cases}\tau_{jk}, & \text{如果 } T_j \text{ 和 } T_k \text{ 在不同的处理器上处理}\\ 0, & \text{否则}\end{cases}$。

3 布谷鸟搜索算法

3.1 布谷鸟繁殖行为

布谷鸟是一种巢寄生鸟类,它将卵产在其它鸟的鸟巢中,由宿主代为孵化和育雏。若被宿主发现,则可能将外来卵移走或直接放弃自己的鸟巢,寻找其他地方重新筑巢。布谷鸟为了增加繁殖成功率,在繁殖期会寻找与孵化期和育雏期相似、雏鸟食性基本相同、卵形与颜色易仿的宿主;它每飞到一个鸟巢,巢里只产一个卵,而且在产卵前,它会将宿主的一个卵移走或全部推出鸟巢。同时一旦布谷鸟的雏鸟孵出,它有将寄主的雏鸟推出巢外的习性,从而独享宿主抚育。

3.2 布谷鸟搜索算法描述

布谷鸟搜索算法是将自然界中布谷鸟寄生孵育雏鸟的生物行为与一些鸟类和果蝇的莱维飞行行为相结合构造出的随机搜索算法。在自然界中,布谷鸟寻找宿主鸟巢位置是随机的或类似随机的方式。为了模拟布谷鸟寻找鸟巢的方式,首先假设以下3个理想状态[19]:

1)布谷鸟一次只产一个卵,并随机选择鸟巢来孵化它;

2)在随机选择的一组鸟巢中,最好的鸟巢将会被保留到下一代;

3)可利用的鸟巢数量 n 是固定的,一个鸟巢主人能发现外来鸟蛋的概率为 $p_a\in[0,1]$,在这种情况下,宿主会将"外来者"移走或遗弃该鸟巢。

基于以上3个理想状态,布谷鸟寻找宿主鸟巢的位置更新公式如下:

$$x_i^{t+1}=x_i^t+\alpha\oplus Lévy(\lambda),1<\lambda\leqslant 3\tag{2}$$

其中,x_i^t 表示第 i 个布谷鸟在第 t 次搜索时宿主鸟巢的空间位置,$\alpha(\alpha>0)$为步长因子,$\oplus$表示点乘,$Lévy(\lambda)$为搜索路径,且服从列维分布。通过式(2)对位置更新后,用产生的随机数 $k\in[0,1]$与 p_a 进行比较,若 $k>p_a$,对 x_i^{t+1} 进行随机更新,否则不变。

布谷鸟搜索算法描述如下:

布谷鸟搜索算法

1)目标函数 f(x),$x=(x_1,\cdots,x_d)^T$

2)初始产生 n 个鸟巢 $x_i(i=1,\cdots,n)$,并根据适应度函数评估鸟巢位置优劣,记录当前最优鸟巢

3)While(未满足搜索精度或达到最大搜索次数)do

4)根据式(2)更新鸟巢位置

5)评估鸟巢位置优劣,若更优则接受4)中更新后的位置,否则保持位置不变

6) if($k>p_a$)then

7) 随机更新鸟巢位置

8)end if

9)评估鸟巢位置优劣,若更优则接受7)中更新后的位置,否则保持位置不变

10) 记录全局最优鸟巢

11)end while

12)输出最优值

4 基于布谷鸟搜索的任务调度算法

4.1 编码方案

多处理器任务调度问题属于组合优化问题。布谷鸟搜索算法是一类求解连续全局优化问题的启发式算法，采用实数的编码方式，解中的每个个体用矢量表示，解的维数由实际问题而定。而任务调度问题的解是一个任务调度序列，属于离散范畴。因此，在应用布谷鸟搜索算法求解任务调度问题时，首先需要构造合理的编码方式来表示调度问题的解，从而将表示每个鸟巢位置的矢量信息转化为满足一定拓扑关系的任务调度序列表。根据多处理器任务调度问题的特点，本文采用基于任务优先权的编码方案，对任务 i 赋予 $1\sim n$ 之间的随机整数，表示任务 i 的优先权值 p_i，从而将布谷鸟选择宿主鸟巢的一个连续位置向量 $x_i=[x_{i,1},x_{i,2},\cdots,x_{i,n}]$ 转化为一个任务优先权序列 $\pi=(p_1,p_2,\cdots,p_n)$。这种编码方式可以保证得到的解都是可行解，同时无需修改布谷鸟搜索算法的进化操作。表 1 是 2 个处理器处理 9 个任务(见图 1)时的编码方案。

表 1 图 1 中 9 个任务的编码方案

任务编号	T1	T2	T3	T4	T5	T6	T7	T8	T9
优先权	7	3	9	1	4	6	7	8	2

4.2 解码方案

在得到鸟巢的位置信息后，要对其优劣进行评估，以确定未来布谷鸟寻找最优鸟巢位置的方向。本文采用调度长度，即全部任务的最晚完成时间作为适应度函数。在计算调度长度时首先需要得到任务的调度序列及其在处理器上的分配情况，这就涉及到对基于任务优先权的编码方案进行解码。本文根据任务之间的依赖关系与任务优先权值的大小来确定任务的调度序列，以保证任务只有在前序任务完成之后才能被调度，且在满足依赖关系的情况下，具有较高优先级的任务优先被调度；对任务优先权值与处理器个数进行取余操作，得到任务在处理器上的分配情况。表 1 中的编码方案对应的解码方案如表 2 所列。

表 2 表 1 中编码方案对应的解码方案

调度序列	T1	T3	T5	T2	T7	T6	T4	T8	T9
处理器	P1	P1	P2	P1	P1	P2	P1	P2	P2

4.3 CS 多处理器任务调度算法流程

综上所述，基于布谷鸟搜索的任务调度算法流程如下：

1)设置 CS 算法相关参数，初始产生任务调度方案，并根据适应度函数 f 评估方案优劣，找出当前最优调度方案；

2)保留上一代最优调度方案，根据式(2)对其余方案更新，评估更新后的方案，若更优则接受更新，否则保持原始方案不变；

3)产生随机数 k，若 $k>p_a$，随机产生新的调度方案；

4)评估 3)中新产生的调度方案，若更优则接受更新，否则保持原始方案不变；

5)记录全局最优调度方案；

6)当满足搜索精度或达到最大搜索次数时转 7)，否则转 2)进行下一轮搜索；

7)输出最优调度方案。

5 实验仿真与分析

本文在 Intel Core i5@1.8GHz CPU，2GB RAM，Windows XP，MATLAB 2013a 的计算环境下，对 CS 算法、文献[9]中的 GA 算法及文献[14]中的 PSO 算法进行性能对比验证。对比标准选择调度长度和算法求解时间。

通过实验选择算法最优参数如下：布谷鸟算法，PopSize=50，MaxGen=1000，$p_a=0.9$，$\alpha=1$；遗传算法，PopSize=100，MaxGen=1000，$p_C=0.7$，$p_M=0.3$；粒子群算法，PopSize=200，MaxGen=500。为了减小数据随机性对测试结果的影响和更准确地反映算法性能，每种算法在测试中都重复运行 20 次，统计出最优解及平均解，算法求解时间取 20 次测试中得到最优解时的平均求解时间。

实验在任务数分别为 18、50、100、210、300 及 500 时，选定不同处理器数目的情况下，在 MATLAB 仿真平台上分别执行以上 3 种算法。其中任务数为 18 的用例图来源于文献[9]中的图 12，任务数为 50、100、210、300 及 500 的用例图来源于 Standard Task Graphs Set [20]，并随机加上 0～20 的通信开销。实验结果见表 3－表 8。

表 3 18 个任务的情况下的对比结果

算法	PSO	GA	CS	PSO	GA	CS
处理器个数	2	2	**2**	4	4	**4**
最优调度长度	440	440	**440**	460	470	**440**
平均调度长度	465.71	473.33	**442.85**	466.19	504.28	**449.52**
平均求解时间(s)	123.45	127.65	**40.60**	126.92	131.44	**40.76**

表 4 50 个任务的情况下的对比结果

算法	PSO	GA	CS	PSO	GA	CS
处理器个数	4	4	**4**	8	8	**8**
最优调度长度	163	158	**155**	119	110	**107**
平均调度长度	174.04	163.76	**158.42**	125.33	113.04	**109.66**
平均求解时间(s)	266.13	268.94	**95.70**	261.89	267.66	**98.56**

表 5 100 个任务的情况下的对比结果

算 法	PSO	GA	CS	PSO	GA	CS
处理器个数	5	5	**5**	10	10	**10**
最优调度长度	279	177	**170**	233	161	**160**
平均调度长度	302.57	190.47	**189.90**	246.85	178.61	**174.66**
平均求解时间(s)	607.95	584.38	**223.59**	613.81	595.98	**221.74**

表 6 210 个任务的情况下的对比结果

算法	PSO	GA	CS	PSO	GA	CS
处理器个数	6	6	**6**	12	12	**12**
最优调度长度	905	502	**484**	619	393	**382**
平均调度长度	932.52	523.52	**509.76**	680.42	426.47	**414.33**
平均求解时间(s)	1415.76	1407.59	**519.25**	1435.11	1407.87	**524.09**

表 7 300 个任务的情况下的对比结果

算法	PSO	GA	CS	PSO	GA	CS
处理器个数	6	6	**6**	12	12	**12**
最优调度长度	1284	717	**680**	1110	709	**668**
平均调度长度	1309.70	757.80	**710.15**	1148.30	742.20	**704.40**
平均求解时间(s)	2467.66	2550.81	**903.84**	2491.81	2476.56	**925.24**

表 8 500 个任务的情况下的对比结果

算法	PSO	GA	CS	PSO	GA	CS
处理器个数	10	10	**10**	20	20	**20**
最优调度长度	2637	1468	**1236**	1945	1238	**1148**
平均调度长度	2682.38	1528.24	**1321.57**	2047.24	1274.05	**1218.76**
平均求解时间(s)	5467.96	5886.67	**1957.81**	5487.82	5380.15	**1982.52**

从表中可明显看出，对于调度长度，CS 算法求得的最优

解及平均解均优于GA算法和PSO算法，而且任务规模越大越明显。GA算法和PSO算法容易出现早熟收敛，陷入局部最优。而CS算法稳定性更好，收敛精度高；对于求解时间，CS算法也明显快于两种对比算法，收敛速度更快，执行时间较两种对比算法缩短超过60%。

由此可知，虽然3种算法均为随机搜索算法，但是CS算法的稳定性更好，求解质量更高，执行速度更快，体现了布谷鸟搜索算法优良的进化机制。

结束语 为了提高多处理器系统中的任务调度效率，本文基于布谷鸟搜索算法提出了一种新的任务调度算法。该算法能充分利用布谷鸟搜索算法求解的精确性与高效性特点，以全部任务的最晚完成时间最小为目标，利用基于任务优先权的编码方案使连续的布谷鸟搜索算法适用于离散的多处理器任务调度问题。通过与遗传算法及粒子群算法性能测试比较结果可知，CS算法的求解质量更高且执行速度更快，能有效缩短任务的完成时间，提高多处理器系统的性能。

参 考 文 献

[1] Kwok Y K, Ahmad I. Static scheduling algorithms for allocating directed task graphs to multiprocessors[J]. ACM Computing Surveys (CSUR), 1999, 31(4): 406-471

[2] Daoud M I, Kharma N. A high performance algorithm for static task scheduling in heterogeneous distributed computing systems [J]. Journal of Parallel and Distributed Computing, 2008, 68 (4): 399-409

[3] 陈华平，黄刘生. 启发式任务调度中的处理器选择策略[J]. 软件学报，1999，10(11)：1194-1198

[4] 石威，郑纬民. 相关任务图的均衡动态关键路径调度算法[J]. 计算机学报，2001，24(9)：991-997

[5] Corrêa R C, Ferreira A, Rebreyend P. Scheduling multiprocessor tasks with genetic algorithms[J]. IEEE Transactions on Parallel and Distributed Systems, 1999, 10(8): 825-837

[6] Gupta S, Agarwal G, Kumar V. An Efficient and Robust Genetic Algorithm for Multiprocessor Task Scheduling[J]. International Journal of Computer Theory and Engineering, 2013, 5(2): 377-382

[7] 叶春晓，陆杰. 基于改进遗传算法的网格任务调度研究[J]. 计算机科学，2010，37(7)：233-235

[8] Omara F A, Arafa M M. Genetic algorithms for task scheduling problem[J]. Journal of Parallel and Distributed Computing, 2010, 70(1): 13-22

[9] Hwang R, Gen M, Katayama H. A comparison of multiprocessor task scheduling algorithms with communication costs[J]. Computers & Operations Research, 2008, 35(3): 976-993

[10] 王成昌，陈闵中，方钰，等. 基于混合粒子群算法的网格任务调度[J]. 计算机科学，2012，39(2)：18-21

[11] Al Badawi A, Shatnawi A. Static scheduling of directed acyclic data flow graphs onto multiprocessors using particle swarm optimization[J]. Computers & Operations Research, 2013, 40 (10): 2322-2328

[12] Yin P Y, Yu S S, Wang P P, et al. A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems[J]. Computer Standards & Interfaces, 2006, 28(4): 441-450

[13] Sivanandam S N, Visalakshi P, Bhuvaneswari A. Multiprocessor Scheduling Using Hybrid Particle Swarm Optimization with Dynamically Varying Inertia[J]. IJCSA, 2007, 4(3): 95-106

[14] 高尚，杨静宇. 多处理机调度问题的粒子群优化算法[J]. 计算机工程与应用，2005，41(27)：72-73

[15] Yang X S, Deb S. Engineering optimisation by cuckoo search [J]. International Journal of Mathematical Modeling and Numerical Optimisation, 2010, 1(4): 330-343

[16] Gandomi A H, Yang X S, Alavi A H. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems[J]. Engineering with computers, 2013, 29(1): 17-35

[17] Ouaarab A, Ahiod B, Yang X S. Discrete cuckoo search algorithm for the travelling salesman problem[J]. Neural Computing and Applications, 2014, 24(7/8): 1659-1669

[18] 邱卫东，陈燕，李洁萍，等. 一种实时异构嵌入式系统的任务调度算法[J]. 软件学报，2004，15(4)：504-511

[19] Yang X S, Deb S. Cuckoo search via Lévy flights[C]// World Congress on Nature & Biologically Inspired Computing, 2009 (NaBIC 2009). IEEE, 2009: 210-214

[20] Standard Task Graph Set [OL]. http://www.kasahara.elec.waseda.ac.jp/schedule/index.html

(上接第81页)

参 考 文 献

[1] López J, Dahab R. Fast Multiplication on Elliptic Curves over GF(2^m) without Precomputation[C]//CHES. 1999: 316-327

[2] Kim C H, Kwon S, Hong C P. FPGA Implementation of High Performance Elliptic Curve Cryptographic processor over GF (2^{163})[J]. Journal of Systems Architecture - Embedded Systems Design, 2008, 54(10): 893-900

[3] Azarderakhsh R, Reyhani-Masoleh A. Efficient FPGA Implementations of Point Multiplication on Binary Edwards and Generalized Hessian Curves Using Gaussian Normal Basis[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2011, 19(1): 1453-1466

[4] Rebeiro C, Roy S S, Mukhopadhyay D. Pushing the Limits of High-Speed GF (2^m) Elliptic Curve Scalar Multiplication on FPGAs[C]//CHES. 2012: 494-511

[5] Rebeiro C, Mukhopadhyay D. Power Attack Resistant Efficient FPGA Architecture for Karatsuba Multiplier[C]// VLSID. 2008: 706-711

[6] Ansari B, Anwar M. High performance architecture of elliptic curve scalar multiplication[J]. IEEE Transactions on Computers, 2008, 57(11): 1443-1453

[7] Reyhani-Masoleh A. Efficient Algorithms and Architectures for Field Multiplication Using Gaussian Normal Bases[J]. IEEE Transactions on Computers, 2006, 55(1): 34-47

[8] Chelton N, Benaissa M. Fast Elliptic Curve Cryptography on FPGA[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2008, 16(2): 198-205

[9] Lai Y, Hung Y, Yang H, et al. High-Performance Architecture for Elliptic Curve Cryptography over Binary Field[C]//ISCAS. 2010: 3933-3936

[10] Shu C. Hardware Architectures of Elliptic Curve Based Cryptosystems over Binary Fields[D]. George Mason University, 2008