

基于改进遗传算法的云计算任务调度算法

胡艳华¹, 唐新来^{1,2}

(1. 广西科技大学鹿山学院 电气与计算机工程系, 广西 柳州 545616;

2. 广西科技大学 教务处, 广西 柳州 545616)

摘要:任务调度是云计算的核心问题。云计算中的任务调度算法要求在提高系统吞吐量和最大跨度的同时又要兼顾资源的安全与负载均衡问题。传统遗传算法因具有强大的并行空间搜索能力而在云计算中得到广泛应用,但其亦存在明显不足,即随着计算机规模的不断扩大,收敛性逐渐降低,存在易早熟等不足,限制了其调度性能。而 Min-Min 和 Max-Min 算法简单易行,且具有较好的时间跨度,可以较好地弥补传统算法的不足。在传统遗传算法的基础上,结合 Min-Min 和 Max-Min 算法,提出了一种新的云计算任务调度算法,在产生初始化种群时引入 Min-Min 和 Max-Min 算法,并选取任务完成时间和负载均衡作为双适应度函数,提高了初始化种群的质量、算法搜索能力以及收敛速度。仿真结果表明,该算法优于传统遗传算法,是一种有效的云计算任务调度算法。

关键词:云计算;遗传算法;任务调度;Min-Min 算法;Max-Min 算法

中图分类号:TP393

文献标识码:A

文章编号:1673-629X(2016)10-0137-05

doi:10.3969/j.issn.1673-629X.2016.10.030

A Task Scheduling Algorithm Based on Improved Genetic Algorithm in Cloud Computing Environment

HU Yan-hua¹, TANG Xin-lai^{1,2}

(1. Department of Electrical and Computer Engineering, Lushan College of Guangxi University of Science and Technology, Liuzhou 545616, China;

2. Office of Academic Affairs, Guangxi University of Science and Technology, Liuzhou 545616, China)

Abstract: Task scheduling mechanism is one of the core issues in cloud computing. The task scheduling algorithm in cloud computing requires improvement of the system throughput and the largest span while considering resources security and load balancing problems. As a classical task scheduling algorithm with powerful and implicit parallel space search capability, genetic algorithm is widely used in cloud computing. However, it has many deficiencies, such as slow convergence and premature with the increasing calculation scale. Min-Min algorithm and Max-Min algorithm are simple and practicable with better makespan, which can well make up the deficiencies of traditional genetic algorithm. On this basis, an improved algorithm is put forward, which introduces Min-Min algorithm and Max-Min algorithm in the process of population initialization, and uses the minimizing makespan and the load balancing of resource as double-fitness function meanwhile. The simulation shows that this algorithm can elevate the quality of initial population, the search capability and the convergence rate, which is more efficient.

Key words: cloud computing; genetic algorithm; task scheduling; Min-Min algorithm; Max-Min algorithm

0 引言

作为一种新兴技术,云计算已成为当今计算机领域的一个研究热点。云计算是在分布式处理、并行处理和网格计算等技术的基础上^[1-2],整合了虚拟化、效用计算、IaaS、PaaS 及 SaaS 等诸多概念发展而来^[3]。所谓的“云”实际上是一个庞大的网络,该网络将用户

请求拆分成若干个子任务,交由至一个庞大服务器群构成的高效处理系统进行搜索并分析计算,然后再返回至用户。它所面向的用户群数量是巨大的,需要高效地处理海量的任务,故如何进行合理而高效的任务调度是云计算的重点与难点。

由于云计算环境中资源的动态异构性,大规模的

收稿日期:2015-12-22

修回日期:2016-04-12

网络出版时间:2016-09-19

基金项目:广西壮族自治区自然科学基金项目(2013GXNSFAA019347);广西科技大学鹿山学院科学基金项目(2015LSKY05)

作者简介:胡艳华(1980-),女,硕士研究生,讲师,研究方向为云计算和计算机网络;唐新来,博士,教授,研究方向为云计算。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20160919.0839.010.html>

任务调度要求在尽量提高系统的吞吐量和最优跨度的同时也要考虑资源的安全与负载均衡等问题。目前应用较多的云计算调度算法主要有经典的遗传算法、Min-Min、Max-Min、Suffrage 及蚁群算法等。异构环境下的资源调度是个 NP 问题,在求解 NP 问题时,遗传算法能够得到很好的解,其在异构环境下任务调度方面的研究较多,与模拟退火、蚁群等算法相比,遗传算法能够得到更优的解。但传统的遗传算法亦存在明显的不足,即随着计算规模的不断扩大,其收敛性逐渐降低^[4]。Min-Min 和 Max-Min 这两种算法的优点是比较简单且容易实现,同时具有较好的时间跨度。因此,文中在传统遗传算法的基础上,结合 Min-Min 和 Max-Min 两种算法的优点,提出了将 Min-Min 算法与 Max-Min 算法和遗传算法(Genetic Algorithm, GA)相结合的改进遗传算法(MMGA 算法)。该算法主要对初始种群的生成进行了改进,并设计出了与传统自适应遗传算法不同的选择和交叉算子,从而提高了算法的搜索能力和收敛速度。在 CloudSim 平台下^[5]的仿真实验证明此算法明显优于传统遗传算法。

1 云环境中的任务调度问题描述

目前,云计算的编程模式大多采用 Google 提出的 Map/Reduce 模型^[6],其主要原理是将要执行的问题分解成 Map 和 Reduce,通过 Map 程序将数据分割成若干独立区域,再将其调度给大量的计算机处理,实现并行计算,然后通过 Reduce 程序汇总计算结果,最后输出用户需要的结果。显然,在此模型中,合理的任务调度至关重要。尽量缩短用户的响应时间,同时也要达到良好的资源动态负载均衡,这是文中要考虑的重点问题。传统 GA 算法虽然在异构环境下有着较好的调度性能,但其天生的缺点,即收敛速度缓慢、易早熟等;而 Min-Min 算法和 Max-Min 算法都较易实现,同时又具有较好的时间跨度、良好的调度性能。文中以经典的遗传算法为基础,引入 Min-Min 及 Max-Min 算法,提出了一种改进遗传算法,并考虑了响应时间和负载均衡问题,旨在建立一个更优的分配调度策略。

文中将云计算中的资源(包括处理器、存储器、网络等)统一视为计算资源,同时假定任务的输入是由若干较大任务分解成的一批子任务,子任务数量远远大于资源数量,各个子任务所需的计算时间已知,且各个子任务的运行时间差异不大。假如用 T 表示大型任务的数量, N 表示子任务的总数量, M 表示计算资源的数量,并用 $M * N$ 的 ETC (Expect Time to Complete) 矩阵来计算各个计算资源上任务队列完成所需时间,其中 $ETC(i, j)$ 表示第 j 个子任务在第 i 个计算资源上运行完成所需的时间。至此,将云计算中的任务调度

问题定义为如何合理地将子任务分配到各个计算资源,使得任务运行完成所需的时间最短、负载均衡最小。

2 云环境中遗传调度算法的改进

GA 是根据生物遗传和进化规律提出的一种用于复杂系统的自适应概率优化技术。由于 GA 具有全局解空间搜索及并行性等优势,该算法以及以该算法为基础的诸多算法在云计算任务调度中得到了广泛应用^[7]。文献[8]提出了一种双适应度的遗传算法(DF-GA),在考虑最小任务完成时间的同时,亦兼顾了子任务平均完成时间,然而该算法求解效率不高;另外,目前多数改进遗传算法尚未考虑资源的负载均衡情况^[9-11]。

文中参照 Map/Reduce 模型^[6],结合已有云计算环境下的改进遗传算法^[9,12],同时考虑到任务完成时间和负载均衡,将 Min-Min 与 Max-Min 算法引入到传统遗传算法中,提出了一种改进遗传算法,以提高云计算环境下的任务调度效率。

2.1 染色体编码与染色体解码

染色体的编码方式包括直接与间接编码。文中考虑到大规模任务处理的特性以及云计算环境的动态异构性等特点,采用间接编码的染色体编码方式,即任务-资源映射模式,对每个子任务所占资源进行编码,每条染色体的总长等于子任务的总数量,染色体中的每一位基因都为正整数,代表子任务编号,此位置上的值代表该子任务所占资源编号,如图 1 所示。

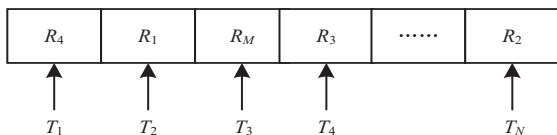


图 1 任务-资源编码

图中, T_i 表示任务编号; R_j 表示任务 T_i 执行时所占第 j 个资源编号。

若有 10 个子任务,3 个可用资源,则染色体长度为 10,每个基因取值为 1~3 之间的随机数,例如随机产生以下的染色体编码:

{2,3,2,3,1,1,2,3,1,2}

则此染色体代表第 2 个资源运行第 1 个子任务,第 3 个资源运行第 2 个子任务,以此类推。得到染色体后,还须解码,以得到各个资源上子任务分布的情况。上述染色体可解码为:

$R_1: \{T_5, T_6, T_9\}$

$R_2: \{T_1, T_3, T_7, T_{10}\}$

$R_3: \{T_2, T_4, T_8\}$

通过解码,求得每个计算资源上的子任务序列,然后通过 ETC 矩阵,计算出每个子任务序列所需的完成

时间,进而得到总任务完成时间函数为:

$$F_1(x) = \max_{i=1}^M \sum_{j=1}^n \text{time}(i,j) \tag{1}$$

其中, $\text{time}(i,j)$ 为被分配到计算资源 R_i 上第 j 个子任务执行所用的时间; n 为分配到该计算资源 R_i 上的子任务数量。

2.2 改进的初始种群生成

初始种群在遗传算法中具有极为重要的作用,其生成方式是首要解决的问题。初始种群的平均适应度越高,较优个体就能更快地引导种群向理想的方向发展而得到最优解,从而使其迭代过程变短、收敛性提高。获取一个具有高平均适应度值的初始种群,是文中的研究重点之一。

Min-Min 算法和 Max-Min 算法具有良好的调度性能,这两种算法能产生质量较优的个体,而以此为基础交叉或变异得到的个体质量亦较优,故文中在初始化种群的生成过程中引入 Min-Min 算法和 Max-Min 算法,对这两种算法分别产生的个体进行交叉,并对变异后的个体再次交叉,同时得到 6 个个体,充分发挥了上述两种算法在调度性能上的优势。为保持个体差异,防止近亲繁殖,避免局部最优,上述 6 个个体以外的初始种群个体均为随机产生,并采用海明距离 (Hamming) 进一步维持个体差异性,即当新个体与前个体之间的海明距离不足某一设定值时,将重新生成该个体,直到得到具有一定个体差异性的种群。如此反复,所得初始种群不仅具有较高的平均适应度值,又能保证种群个体的多样性;使算法具有较好的搜索空间能力,能有效避免局部最优,并具有较高的搜索速度。改进后的种群初始化方法如图 2 所示。

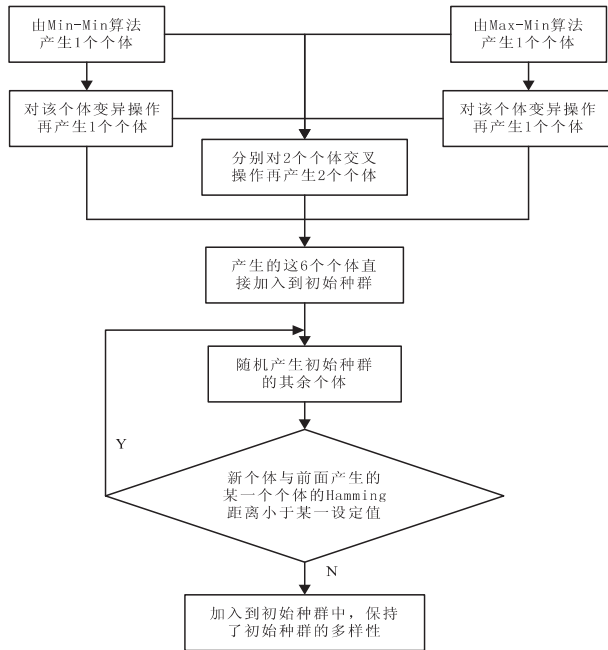


图2 改进后的种群初始化

2.3 适应度函数

在进化搜索中,遗传算法一般单纯使用适应度函数为依据,使用个体的适应度值进行搜索。所以适应度函数的选择至关重要,将直接影响到收敛速度以及最终能否找到最优解^[13]。最优跨度,即使任务的总执行时间 (makespan) 最小,是文中调度算法涉及的一个重要内容。故文中选用任务的总执行时间函数作为遗传算法的适应度函数之一:即

$$f_1(x) = \frac{1}{F_1(x)} \times u_{LB} \tag{2}$$

$$u_{LB} = \frac{\sum_{i=1}^M \sum_{j=1}^n \text{time}(i,j)}{M \times F_1(x)} \tag{3}$$

其中, u_{LB} 为文中定义的平衡任务负载因子,代表各个计算资源的利用率情况。 u_{LB} 的值越大,表示计算资源的利用率越高,那么 $F_1(x)$ 的值就相对越小。

资源负载均衡问题是资源调度中要考虑的另外一个重要方面,它能大大提高资源的利用效率。在设计适应度函数时也应考虑到资源负载均衡问题。文中参考文献[12],采用染色体上资源节点任务分配数标准差来衡量资源负载均衡。种群初始化后,设种群大小为 Scale,子任务总个数为 N ,计算资源数即 worker 的个数为 M ,则每个染色体的资源节点平均分配任务数 $AT = N/M$ 。对于任一个染色体,基于资源节点任务分配数标准差的适应度函数为:

$$\sigma(j) = \sqrt{\frac{\sum_{i=1}^M (AT'(j,i) - AT)^2}{M}}, 1 \leq j \leq \text{Scale} \tag{4}$$

其中, $AT'(j,i)$ 为第 j 个染色体第 i 个资源节点所分配到的任务数。

由于在选择复制阶段必须选择任务标准差较小的染色体,因此设定基于任务分配数标准差的适应度函数为:

$$f_2(j) = \frac{1}{\sigma(j)}, 1 \leq j \leq \text{Scale} \tag{5}$$

2.4 遗传操作

2.4.1 选择操作

选择操作是指根据“优胜劣汰”的法则,在种群中不断选取适应度较强的个体,逐渐用以产生新种群的过程。个体的适应度越高,被筛选到下一代的概率就越大;反之亦然。如此反复,得到种群中个体的适应度值的最优解。

假设文中算法的种群大小为 Scale,首先选择父代中的最优个体和 Min-Min 算法、Max-Min 算法产生的个体,以保留较优个体,其他的 Scale-3 个体则采用轮盘赌方式作为选择操作算子,通过式(2)、(5)得出

个体选择概率为:

$$P_1(i) = \frac{f_1(i)}{\sum_{j=1}^{Scale} f_1(j)} \tag{6}$$

$$P_2(i) = \frac{f_2(j)}{\sum_{j=1}^{Scale} f_2(j)} \tag{7}$$

选择下一代个体时,先以 c_1 和 c_2 的概率分别选择 P_1 和 P_2 (其中 $0 < c_1 < 1, c_1 + c_2 = 1$), 选取其一作为个体选择概率^[14]。这样,种群中既有总任务完成时间较短的个体,又有负载较为均衡的个体,为种群进化奠定了优良的基因基础。

2.4.2 交叉与变异操作

普通自适应算法中,当个体适应度值趋向最大适应度值时,交叉概率与变异概率减小。这对于种群进化后期较为有利,但不利于初期进化,因其能增加进化走向局部最优的几率。因此,需要做进一步修改,修改后的公式为:

$$P_c = \begin{cases} P_{c1} - \frac{(P_{c1} - P_{c2})(f' - f_{avg})}{f_{max} - f_{avg}}, & f' \geq f_{avg} \\ P_{c1}, & f' < f_{avg} \end{cases} \tag{8}$$

$$P_m = \begin{cases} P_{m1} - \frac{(P_{m1} - P_{m2})(f_{max} - f)}{f_{max} - f_{avg}}, & f \geq f_{avg} \\ P_{m1}, & f < f_{avg} \end{cases} \tag{9}$$

当 $f' = f_{max}$ 时, $P_c = P_{c2} > 0$; 当 $f = f_{max}$ 时, $P_m = P_{m2} > 0$ 。

这样,种群中的较优个体之间拥有更高的交叉与变异概率。同时采用最优保存策略,将每一代的最优个体直接复制到下一代中而保证其不被破坏。上述公式中,一般取 $P_{c1} = 0.9; P_{c2} = 0.6; P_{m1} = 0.1; P_{m2} = 0.001$ 。

3 算法仿真结果与分析

文中采用云仿真器 CloudSim^[5] 对上述提出的 MMGA 进行验证和分析。

同时,在相同的环境条件下,对 GA 和 MMGA 进行了比较实验,主要参数如表 1 所示。

算法初始条件:Scale 取值为 100, M 取值为 5 ~ 50, N 取值为 1 000 ~ 5 000。

算法终止条件:文中设最大进化代数为 200,当算法连续 50 代没有找到更好的解,认定算法为基本收敛,将终止算法。

(1)若取 $M = 20$, N 取 1 000 ~ 5 000,实验过程中多次跟踪任务完成时间的完成时间,结果如图 3 所示。

表 1 实验参数设置

| 算法 | 主要参数 | 取值 |
|------|----------|-------|
| GA | k_1 | 0.39 |
| | k_2 | 0.85 |
| | k_3 | 0.096 |
| | k_4 | 0.056 |
| MMGA | P_{c1} | 0.9 |
| | P_{c2} | 0.6 |
| | P_{m1} | 0.1 |
| | P_{m2} | 0.001 |
| | c_1 | 0.7 |
| | c_2 | 0.3 |

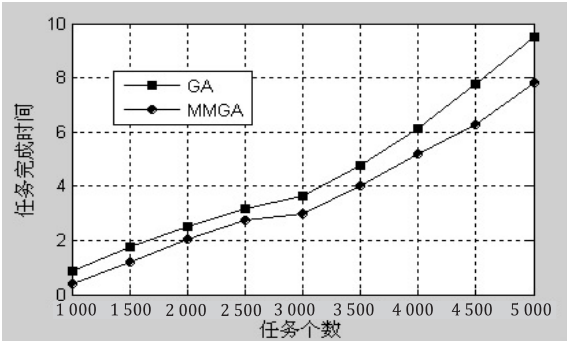


图 3 任务的完成时间曲线(时间单位:10⁴ ms)

由图 3 可以看出,与 GA 相比,MMGA 的任务完成时间明显较短。

(2)若 $N = 4\ 000, M = 20$,考察算法的收敛迭代情况,如图 4 所示。

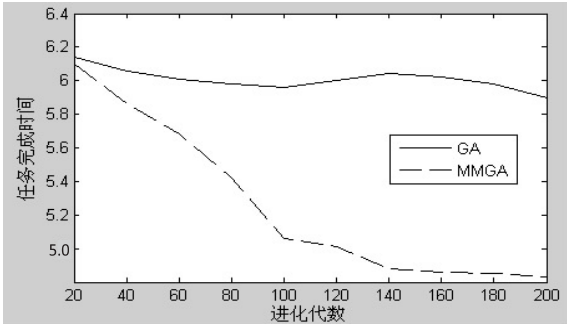


图 4 算法的收敛结果比较(时间单位:10⁴ ms)

由图 4 可知,在算法进化初期,GA 和 MMGA 得出的任务完成时间相差不大,但 MMGA 算法在初始化种群产生时引入 Min-Min 算法和 Max-Min 算法,较好地提高了初始化种群的质量,使收敛速度加快,缩短了寻找最优解时间,MMGA 在 140 次就开始收敛;随着进一步的进化,在 GA 初始阶段出现的超常个体误导了种群的进化方向,因而陷入了局部收敛,而 MMGA 随着进一步进化,同任务完成时间明显低于 GA,得到的最优解更好。

(3)若 $N = 5\ 000$,将任务分配到 $M = 5$ 个资源节点

(R_1, R_2, R_3, R_4, R_5) 上,若其处理能力为 (120, 200, 150, 340, 500) MFLOPS,实验过程中分别记录 5 个资源节点上的资源负载情况,其结果如图 5 所示。

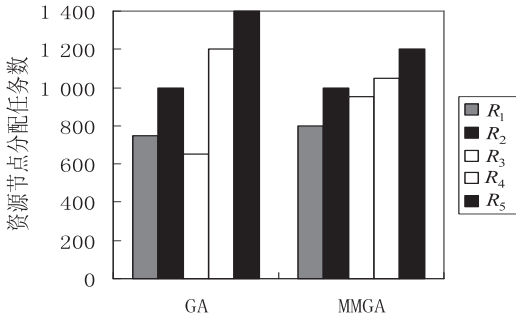


图 5 多任务情况下节点的资源负载情况

由图 5 可知,在任务量大、资源节点运算能力差异较大的情况下,MMGA 的资源负载均衡程度明显好于 GA。

综上所述,文中提出的 MMGA 比 GA 收敛速度更快,且可以使得任务执行时间较短,资源负载较均衡,能较好地应用在云计算资源环境中。

4 结束语

文中在充分考虑大规模任务处理特性和云计算环境动态异构性的基础上,提出了一种基于传统遗传算法的改进任务调度算法-MMGA。该算法既可以保证种群具有较高的平均适应度,又可以维护种群个体的多样性;同时,算法中采用任务执行时间和负载均衡作为双适应度函数,使得在提高收敛速度的同时,兼顾资源均衡。仿真结果表明,该改进算法收敛性能较好、资源负载较均衡,具有良好的效率,能更有效地解决云计算环境下的任务调度问题。

参考文献:

[1] Chien A, Calder B, Elbert S, et al. Entropia: architecture and performance of an enterprise desktop grid system[J]. Journal of Parallel and Distributed Computing, 2003, 63(5): 597-

610.

[2] Kim J S, Nam B, Marsh M, et al. Creating a robust desktop grid using peer-to-peerservices [EB/OL]. [2009-10-16]. ftp://ftp.cs.umd.edu/pub/hpsl/papers/papers-pdf/ngs07.pdf.

[3] Armbrust M, Fox A, Griffith R, et al. A view of cloud computing[J]. Communications of the ACM, 2009, 53(4): 50-58.

[4] Carretero J, Xhafa F. Use genetic algorithms for scheduling jobs in large scale grid applications[J]. Technologies and Economic Development of Economy, 2006, 12(1): 11-17.

[5] Buyya R, Ranjan R, Calheiros R N. Modeling and simulation of scalable cloud computing environments and the CloudSim Toolkit: challenges and opportunities[C]//Proceedings of the seventh high performance computing and simulation conference. New York, USA: IEEE Press, 2009: 21-24.

[6] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[C]//Proceedings of the 6th symposium on operating system design and implementation. New York: ACM, 2004: 137-150.

[7] 王小平,曹立明. 遗传算法[M]. 西安: 西安交通大学出版社, 2002.

[8] 李建锋,彭 舰. 云计算环境下基于改进遗传算法的任务调度算法[J]. 计算机应用, 2011, 31(1): 184-186.

[9] 朱宗斌,杜中军. 基于改进 GA 的云计算任务调度算法[J]. 计算机工程与应用, 2013, 49(5): 77-80.

[10] 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.

[11] 王春莲. 基于改进遗传算法的网格任务调度算法[D]. 济南: 山东大学, 2009.

[12] 刘 愉,赵志文,李小兰,等. 云计算环境中优化遗传算法的资源调度策略[J]. 北京师范大学学报: 自然科学版, 2012, 48(4): 378-384.

[13] 段卫军,付学良,王 芳,等. 云计算环境下融合遗传算法和蚁群算法 QoS 约束任务调度[J]. 计算机应用, 2014, 34(S2): 66-69.

[14] 邹伟明,于 炯. 云计算环境下基于用户满意度的遗传算法[J]. 计算机应用研究, 2014, 31(1): 85-88.

(上接第 136 页)

[7] Eckel B. Java 编程思想[M]. 陈昊鹏,译. 北京: 机械工业出版社, 2007.

[8] 张海燕. Java 多线程技术在手机联网中的应用[J]. 农业网络信息, 2008(3): 97-98.

[9] Kraus J D, Marhefka R J. Antennas: for all applications[M]. New York: New McGraw-Hill, 2001.

[10] Bridge W M. Cross coupling in a five horn monopulse tracking system[J]. IEEE Transactions on Antennas and Propagation, 1972, 20(4): 436-442.

[11] Weaver A C, Luo Jihao, Zhang Xinyuan. Monitoring and control using the Internet and Java industrial electronics society [C]//Proc of IECON'99. [s.l.]: IEEE, 1999: 1154-1158.

[12] 代 敏. 基于 Android 平台下手机定位程序的设计及实现[J]. 计算机与数字工程, 2012, 40(4): 143-145.

[13] 耿东久,索 岳,陈 渝,等. 基于 Android 手机的远程访问和控制系统[J]. 计算机应用, 2011, 31(2): 559-561.

[14] 王朝华,陈德艳,黄国宏,等. 基于 Android 的智能家居系统的研究与实现[J]. 计算机技术与发展, 2012, 22(6): 225-228.

[15] 艾国祥,施浒立,吴海涛,等. 基于通信卫星的定位系统原理[J]. 中国科学: G 辑, 2008, 38(12): 1615-1633.