遗传算法与蚂蚁算法动态融合的软硬件划分*

熊志辉[1,2+], 李思昆[1], 陈吉华[1]

[1](国防科学技术大学 计算机学院,湖南 长沙 410073)

[2](国防科学技术大学 信息系统与管理学院,湖南 长沙 410073)

Hardware/Software Partitioning Based on Dynamic Combination of Genetic Algorithm and Ant Algorithm

XIONG Zhi-Hui[1,2+], LI Si-Kun[1], CHEN Ji-Hua[1]

[1](School of Computer Science, National University of Defense Technology, Changsha 410073, China)

[2](School of Information System and Management, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: Phn: +86-731-4573560, Fax: +86-731-4575876, E-mail: xzhnudt@vip.sina.com, http://www.nudt.edu.cn

Received 2003-12-30; Accepted 2004-05-08

Xiong ZH, Li SK, Chen JH. Hardware/Software partitioning based on dynamic combination of genetic algorithm and ant algorithm. *Journal of Software*, 2005,16(4):503−512. DOI: 10.1360/jos160503

Abstract: Genetic algorithm can do colony global searching quickly and stochastically, but can't efficiently get to optimal results, since it slows down when solving to certain scope. On the other hand, ant algorithm gets to optimal results efficiently, but lacks initial pheromone at the beginning. To solve the hardware/software bi-partitioning problem in embedded system and system-on-a-chip design, the authors put forward a new algorithm based on dynamic combination of genetic algorithm and ant algorithm. The basic idea is: (1) using genetic algorithm to generate preliminary partitioning results, converting them into initial pheromone distribution for ant algorithm, and then using ant algorithm to search for optimal partitioning scheme; (2) while running genetic algorithm, dynamically determining the best combination time of genetic algorithm and ant algorithm to avoid too early or too late termination of the genetic algorithm. The algorithm utilizes the advantages of the two algorithms and overcomes their disadvantages, and it introduces a dynamic combination strategy between them. Experimental results show the algorithm excels genetic algorithm and ant algorithm in performance, and it is discovered that the bigger the partitioning problem is concerned, the better the algorithm performs.

Key words: genetic algorithm; ant algorithm; embedded system; hardware/software partitioning; pheromone

摘 要: 面向嵌入式系统和 SoC(system-on-a-chip)软硬件双路划分问题,提出遗传算法与蚂蚁算法动态融合的软硬件划分算法.基本思想是:(1) 利用遗传算法群体性、全局、随机、快速搜索的优势生成初始划分解,将其转化为

* Supported by the National Natural Science Foundation of China under Grant No.90207019 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2002AA1Z1480 (国家高技术研究发展计划(863))

作者简介: **熊志辉**(1976−),男,江西南昌人,博士,讲师,主要研究领域为 SoC 系统设计方法,电子设计自动化;**李思昆**(1941−),男,教授,博士生导师,主要研究领域为电子设计自动化,SoC 设计方法学,虚拟样机与分布式虚拟环境;**陈吉华**(1963−),男,教授,主要研究领域为电子设计自动化,VLSI 设计方法学.

蚂蚁算法所需的初始信息素分布,然后利用蚂蚁算法正反馈、高效6收敛的优势求取最优划分解;(2) 在遗传算法运行过程中动态确定遗传算法与蚂蚁算法的最佳融合时机,避免由于遗传算法过早或过晚结束而影响划分算法的整体性能.该算法既发挥了遗传算法与蚂蚁算法在寻优搜索中各自的优势,又克服了遗传算法在搜索到一定阶段时最优解搜索效率低以及蚂蚁算法初始信息素匮乏的不足,并且在算法中提出了遗传算法与蚂蚁算法动态融合的衔接策略.实验结果表明,该算法在性能上明显优于遗传算法和蚂蚁算法,并且划分问题规模越大,优势越明显.

关键词: 遗传算法;蚂蚁算法;嵌入式系统;软硬件划分;信息素

中图法分类号: TP18 **文献标识码**: A

软硬件划分是嵌入式系统和 SoC(system-on-a-chip)软硬件协同设计的关键步骤,其基本任务是:在满足某些约束的条件下,将系统功能行为"最优地"分配到一定的软硬件系统结构上.根据目标系统结构不同,软硬件划分问题可分为双路划分(bi-partitioning)和多路划分(multi-way partitioning).其中双路划分应用最广泛,也是软硬件划分问题的基础.在有些研究中,把划分作为软硬件综合的一部分[1,2].

软硬件划分是 NP 完全问题[3].1992 年,Gupta 等人开发了一种软硬件划分算法用于自动化设计空间探索过程[2].此后,人们研究了各具特色的划分算法.其中,应用效果较好的典型算法是启发式算法[4],如爬山法[5]、遗传算法[1,6,7]、模拟退火[8]、禁忌搜索[9]等,这些算法通过定义启发信息指导搜索过程逐步向最优解收敛.爬山法采用"以退为进"的策略寻优,但易于陷入局部最优;遗传算法吸取了生物进化和遗传变异论的研究成果,是一种群体性全局寻优方法,但算法执行到一定阶段后向最优解收敛速度缓慢.模拟退火算法模拟物质材料的冷却与结晶过程,通过退火温度控制搜索过程,但问题规模较大时,系统进入热平衡状态(对应于最优解)的时间较长.禁忌搜索法模拟人类智力过程,通过引入灵活的存储结构和相应禁忌准则来避免迂回搜索,并通过藐视准则赦免一些被禁忌的优良状态,具有较强的"爬山"能力,但数据存取操作频繁,影响了搜索速度.另外,较有特色的 GCLP/IBS 算法[10]通过定义软/硬件极端节点、软/硬件排斥节点和普通节点来体现进程节点在硬件和软件上执行时表现出来的性能特征,但该算法对目标系统结构的依赖性较大.文献[11]提出的约束驱动与松弛时间消除相结合的软硬件划分算法[11]在求解效率上改进了 GCLP/IBS 算法.

文献[12]提出应用蚂蚁算法求解软硬件系统任务双路划分问题的方法,试图在满足面积约束的条件下优化时间性能.与前述方法相比,该方法在求解时间和寻优精度上取得了更好的结果.但单纯的蚂蚁算法在运行初期缺乏信息素,这限制了算法搜索效率的进一步提高.我们研究表明,蚂蚁算法中约占整个运算过程65%的时间,被用于形成最优解上的信息素强度.

文献[13]提出了遗传算法与蚂蚁算法融合的一般性优化问题求解策略,在对 TSP(traveling salesman problem)问题的应用实验中取得了良好效果.该策略将遗传算法设置为运行固定的迭代次数,具有简单、易行的优点,但这样容易造成融合时机过早或过晚.文献[14]将蚂蚁算法与遗传算法反复交叉,利用蚂蚁算法不断生成种群个体,进行同步时序电路的初始化,这种优化方法能够尽可能多地初始化触发器,但每次种群进化都要经历几代,运行效率较低.

面向嵌入式系统和 SoC 软硬件双路划分问题,本文提出遗传算法与蚂蚁算法动态融合的软硬件划分算法(dynamic combination of genetic algorithm and ant algorithm,简称 DCG3A).基本思想是:(a) 先利用遗传算法快速随机的群体性全局搜索能力生成划分问题初始解,并将其转化为初始信息素分布,然后利用蚂蚁算法正反馈、高效收敛的优势寻求最优划分解.(b) 遗传算法迭代过程中统计子代群体的进化率,在给定的遗传迭代次数范围内,如果连续若干代,子代群体的进化率都低于预设的最小进化率,则终止遗传算法过程,进入蚂蚁算法,确保遗传算法和蚂蚁算法在最佳时机融合.

实验表明,与已有的基于遗传的划分算法[1]和基于蚂蚁的划分算法[12]相比,本文算法运行效率提高约 1 倍,而且划分问题规模越大,改进效果越明显.

1 遗传算法与蚂蚁算法动态融合

1.1 遗传算法简介

美国 Michigan 大学 J. Holland 教授于 1975 年提出的遗传算法,以达尔文生物进化理论“适者生存,优胜劣汰”和孟德尔遗传变异理论“生物遗传进化主要在染色体上,子代是父代遗传基因在染色体上的有序排列”为基础,模拟生物进化过程[15].此算法在自适应控制、组合优化、模式识别、机器学习、规划策略、信息处理等领域的应用中展示了明显的优越性.在软硬件划分领域,遗传算法也有应用[1,6,7].

遗传算法的主要优点是:a) 具有领域无关的群体性全局搜索能力,可避免陷入局部最优;b) 搜索使用评价函数启发,过程简单;c) 使用概率机制进行迭代,具有随机性;d) 可扩展性强,易于介入到已有模型中,且易于与其他优化技术结合.

主要缺点是:a) 没有充分利用系统反馈信息,使搜索具有盲目性;b) 算法求解到一定范围时往往做大量冗余迭代,向最优解收敛的速度大大降低,导致求最优解效率较低.

1.2 蚂蚁算法简介

意大利学者 M. Dorigo 在文献[16]中提出了蚂蚁算法,并因此获得 2003 年度玛丽-居里夫人杰出成就奖.研究表明,几乎不具备视觉的蚂蚁之所以能够找到蚁巢到食物之间的最短路径,是靠其在走过的路上释放一种挥发性分泌物(即信息素,pheromone)来实现的,后来的蚂蚁选择该路径的概率与当时这条路径上信息素的强度成正比.当某路径上通过的蚂蚁越来越多时,在此路径上留下的信息素也越来越强,使得后来的蚂蚁选择此路径的概率更高,从而更增加了此路径上的信息素强度,可吸引更多的蚂蚁选择此路径,这样就形成了一种正反馈机制,使蚂蚁最终可发现最短路径.蚂蚁算法在许多组合优化问题上取得了较好的效果,如二次分配问题.TSP 问题和车间作业调度问题等.

蚂蚁算法的主要优点是:a) 具有正反馈机制,通过信息素的不断更新高效收敛到最优解;b) 具有通用随机优化特性,并且在蚂蚁中融入人类智能;c) 是一种分布式优化方法,有利于并行计算;d) 是一种全局优化方法,既可求解单目标优化问题,也可求解多目标优化问题.

主要缺点是:初期信息素缺乏,导致搜索初期积累信息素占用的时间较长.

1.3 遗传算法与蚂蚁算法动态融合基本原理

通过对遗传算法与蚂蚁算法的研究与实验发现,它们在总体态势上呈现出如图 1 所示的速度-时间曲线.遗传算法在搜索的初期(t_0~t_a 时间段)具有较高向最优解收敛的速度,但 t_a 之后求最优解的效率显著降低.而蚂蚁算法在搜索的初期(t_0~t_a 时间段)由于缺乏信息素,使得搜索速度缓慢,但当信息素积累到一定的强度之后(t_a 时刻之后),向最优解收敛的速度迅速提高.遗传算法与蚂蚁算法动态融合的基本思想就是:在最佳点(a 点)之前采用遗传算法生成初始信息素分布,在最佳点之后采用蚂蚁算法求取最优解.

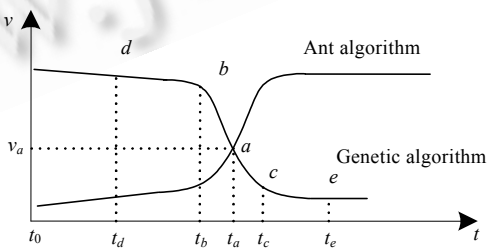

Fig.1 Speed-Time curve of genetic algorithm and ant algorithm

图 1 遗传算法与蚂蚁算法速度-时间曲线

文献[13]提出了遗传算法与蚂蚁算法融合的一般性优化问题求解策略,在 TSP 应用实验中取得了良好的优化效果.该策略将遗传算法设置为运行固定的迭代次数,这样会造成过早(如 t_d 时刻)或过晚(如 t_e 时刻)结束遗传算法过程,不能有效保证两者在最佳时机(t_a 时刻)的融合.

本文提出的动态融合策略可以确保遗传算法与蚂蚁算法在最佳时机融合.方法是:a) 设置最小遗传迭代次数 $Gene_{min}$(如 t_b 时刻)和最大遗传迭代次数 $Gene_{max}$(如 t_c 时刻).b) 遗传算法迭代过程中统计子代群体的进化率,并以此设置子代群体最小进化率 $Gene_{min\text{-}impro\text{-}ratio}$.c) 在设定的迭代次数范围内,如果连续 $Gene_{die}$ 代,子代群体的进化率都小于 $Gene_{min\text{-}impro\text{-}ratio}$,说明这时遗传算法优化速度较低,因此可终止遗传算法过程,进入蚂蚁算法.

2 软硬件划分问题与双着色模型

定义 1(k 路划分问题). 对于模块集合 $M=\{m_1,m_2,\ldots,m_n\}$,k 路划分问题就是寻找簇的集合 $P=\{p_1,p_2,\ldots,p_k\}$,满足:

$$\begin{cases} p_i \subseteq M, 1 \le i \le k \\ \bigcup_{i=1}^{k} p_i = M \\ p_i \cap p_j = \Phi, 1 \le i, j \le k, i \ne j \end{cases}.$$

k 路划分问题的求解通常是要在满足某些约束的条件下优化目标函数.当 k=2 时,称为双路划分问题;当 k>2 时,称为多路划分问题.

软硬件划分是嵌入式系统设计中的关键问题之一,目前研究和应用较多的是双路划分,即考虑系统中只有一个处理器再加上 ASIC 或其他硬件部件的情况.本文考虑嵌入式系统任务软硬件双路划分,并在论文叙述中简称为软硬件划分.

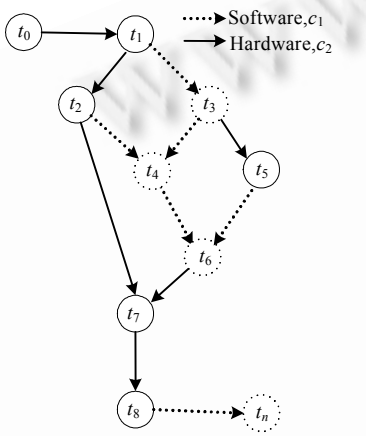

Fig.2 Bi-Coloring model
图 2 双着色模型

任务是粗粒度、接口定义清晰的一系列运算操作的集合,通常表现为高级语言中的一个算法过程.可以用任务图表达嵌入式系统的功能行为.

求解软硬件划分问题时,通常用有向无环图(DAG)描述任务图,$G=(T,E)$,其中,$T=(t_0,t_1,\ldots,t_n)$是任务节点集,E 是有向边集.节点代表功能行为(t_0 和 t_n 是虚拟任务节点,用于确保任务图中只有一个开始节点和一个结尾节点),有向边代表节点之间的控制/数据依赖关系与数据通信代价.

可以用图的双着色模型[12]来表示软硬件双路划分问题,如图 2 所示.把划分到软件部分的功能用颜色 c_1 表示,划分到硬件部分的功能用颜色 c_2 表示.划分的目标是寻找任务图中所有节点的最优着色方案,使得在不超过给定硬件面积的条件下系统的执行时间最短.由于 t_0 和 t_n 是虚拟任务节点,因此,最终优化方案与节点 t_0,t_n,边 e_{01} 以及边 e_{8n} 的颜色无关.着色过程中将边 e_{ij} 的颜色设置为该边目标节点(即 t_j)的颜色,并且每条边 e_{ij} 对应于两个全局启发信息 $\tau_{ij}(k)$,分别表示任务 t_j 被着色为颜色 c_k 的概率,其中 k=1,2.

3 基于 DCG3A 的软硬件划分算法

下面以软硬件划分的双着色模型为基础,介绍本文提出的基于 DCG3A 的软硬件划分算法.

3.1 DCG3A中的遗传算法规则

遗传编码:采用二进制编码方法[15],0 代表颜色 c_1(划分到软件),1 代表颜色 c_2(划分到硬件).

目标函数与适应值函数:本文讨论的软硬件划分是在硬件面积约束条件下使系统运行时间最短,因此,将目标函数定义为 s_{best}=arg min $time_s$,适应值函数定义为 $fitness(s)=1/time_s$.其中,$time_s$ 表示着色(划分)方案 s 的运行时间.

初始种群生成:采用随机方法生成初始种群.

选择算子:采用遗传算法中应用最广的转盘式选择策略(roulette wheel selection)[15].

交叉算子:采用均匀杂交(uniform crossover)策略[15].

变异算子:以变异概率 p_m 将所选个体位取反.

遗传控制参数设置:采用算子执行非重叠遗传过程,依据文献[15]给定的控制参数经验值范围,设定种群规模 N=50,杂交概率 p_c=0.6,变异概率 p_m=0.2.

遗传算法结束条件:在本文算法中,遗传算法结束条件实际上就是判断遗传算法与蚂蚁算法的最佳融合时机.我们设置最小遗传迭代次数 $Gene_{\min}$=15,最大遗传迭代次数 $Gene_{\max}$=50,最小进化率 $Gene_{\text{min-impro-ratio}}$=3%,$Gene_{\text{die}}$=3.

3.2 DCG3A中的蚂蚁算法规则

目标函数与适应值函数:同遗传算法.

信息素设置与更新规则:以 Thomas Stuzle 在 MMAS(max-min ant system)算法[17]中提出的信息素设置与更新策略为基础.因此,信息素 $\tau_{ij}(k)$的更新方程为

$$\tau_{ij}(k)=\begin{cases}(1-\rho)*\tau_{ij}(k)+\Delta\tau_{ij}(k)_{\text{best}} & \text{且}\\ \tau_{ij}(k)_{\max},\text{if } \tau_{ij}(k)>\tau_{ij}(k)_{\max} & \text{且}\\ \tau_{ij}(k)_{\min},\text{if } \tau_{ij}(k)<\tau_{ij}(k)_{\min}\end{cases} \tag{1}$$

其中,ρ是信息素挥发率,$\tau_{ij}(k)_{\max}$和$\tau_{ij}(k)_{\min}$分别是边 e_{ij} 上 c_k 信息素的最大值和最小值(本文分别设定为无限大和60),$\Delta\tau_{ij}(k)_{\text{best}}$ 是本圈最佳蚂蚁对边 e_{ij} 上 c_k 信息素的增加量,定义为

$$\Delta\tau_{ij}(k)_{\text{best}}=\begin{cases}Q/time_{s_{\text{best}}}, & s_{\text{best}}\text{中}e_{ij}\text{着色为}c_k\\ 0, & \text{其他情况}\end{cases} \tag{2}$$

其中,Q 是常数.

蚂蚁圈模型各因子定义:对于除 t_n 以外的每个节点 t_i,蚂蚁试图确定 t_i 的每个后继 t_j 的颜色,它主要依据边 e_{ij} 上的全局启发信息(即信息素 $\tau_{ij}(k)$)和节点 t_j 上的局部启发信息(运行时间与面积的综合代价)完成此工作.准确地说,节点 t_i 上的蚂蚁,以如下概率猜测后继节点 t_j 将被着色为 c_k:

$$p_{ij}(k)=\frac{\tau_{ij}(k)^{\alpha}\eta_j(k)^{\beta}}{\sum_{l=1,2}\tau_{ij}(l)^{\alpha}\eta_j(l)^{\beta}} \tag{3}$$

其中,$\tau_{ij}(k)$是边 e_{ij} 上的信息素,α和β是全局启发信息和局部启发信息相对重要性的控制因子,$\eta_j(k)$是 t_j 被着色为 c_k 的局部启发信息,定义为

$$\eta_j(k)=1/((w_t\times time_t(k))+(w_a\times area_j(k))),$$

其中,w_t 和 w_a 是运行时间和所需硬件面积之间的归一化权重因子,$time_j(k)$和 $area_j(k)$分别表示节点 t_j 被着色为 c_k 时的运行时间和所需面积.

当蚂蚁进入节点 t_i 时,将综合考虑 t_i 所有前驱节点对 t_i 着色猜测的结果,猜测 t_i 在本次蚂蚁迭代中的颜色.它以如下概率猜测 t_i 被着色为 c_k:

$$p_i(k)=\frac{\text{猜测}t_i\text{着色为}c_k\text{的前驱个数}}{\text{节点}t_i\text{的前驱个数}} \tag{4}$$

蚂蚁算法控制参数设置:α=β=1,w_t=1,w_a=2,信息素挥发率ρ=0.2,常数 Q=1000.

蚂蚁数设置:蚂蚁数 m 的设置有 2 种选择:a) m 等于任务图平均分支数;b) m 等于任务图最大分支数.本文将 m 设置为任务图平均分支数.

蚂蚁算法结束条件:当满足以下条件之一时,蚂蚁算法终止:a) 蚂蚁算法迭代代数达到 $Ant_{\max}$;b) 迭代中连续 Ant_{die} 代,子代优化解改进率都小于 $Ant_{\text{min-improv-ratio}}$.本文设置 $Ant_{\max}$=100,Ant_{die}=3,$Ant_{\text{min-improv-ratio}}$=0.5%.

3.3 DCG3A中遗传算法与蚂蚁算法的衔接

DCG3A 算法前期执行遗传算法过程,后期执行蚂蚁算法过程,因此两者的衔接是很重要的.主要包括这些问题:

动态融合时机设置:同前述“遗传算法结束条件”.

蚂蚁算法信息素初值设置:MMAS 算法把各路径信息素初值设定为最大值 $t_{\max}$,文献[12]中所提算法把初值固定为 τ_0=100.我们利用遗传算法得到初始信息素分布,将各边 e_{ij} 的信息素初值设定为

$$\tau_{ij}^{S}(k)=\tau_{ij}^{C}(k)+\tau_{ij}^{G}(k) \tag{5}$$

其中,$\tau_{ij}^{C}(k)$ 是 e_{ij} 上 c_k 信息素常数,相当于 MMAS 中的 $t_{\min}$,$\tau_{ij}^{G}(k)$是从遗传算法求解结果转换而来的 e_{ij} 上的 c_k 信息素值.本文设置 $\tau_{ij}^{C}(k)=\tau_{ij}(k)_{\min}$=60,$0\le i,j\le n$,$k$=1,2.

遗传算法求解结果向信息素值转换:我们选取遗传算法终止时种群中适应值最好的前 10%(即 0.1×N=0.1×50=5)个体作为遗传优化解集合,记为 $S_{10\%\text{better}}^{gene}$.开始时,设置 $\tau_{ij}^{G}(k)$ 为 0,$0\le i,j\le n$,k=1,2.对于 $S_{10\%\text{better}}^{gene}$ 中的每个解 s,如果边 e_{ij} 被着色为 c_k,则 $\tau_{ij}^{G}(k)$ 自加 20.

3.4 基于DCG3A的软硬件划分算法过程

首先将问题描述为双着色模型,然后执行遗传算法过程,直到到达最佳融合时机为止,然后,按式(5)将遗传算法求解结果转换为蚂蚁算法信息素初值,最后执行蚂蚁算法直到结束.

下面详细描述 DCG3A 中遗传算法、蚂蚁算法以及两者衔接部分的执行过程.

//遗传算法部分:

1.初始化遗传算法控制参数(种群规模 N=50,杂交概率 p_c=0.6,变异概率 p_m=0.2);

2.设置遗传算法结束条件($Gene_{\min}$=15,$Gene_{\max}$=50,$Gene_{\text{min-impro-ratio}}$=3%,$Gene_{\text{die}}$=3);

3.随机生成初始种群 $P(0)$,g=0; //g 是遗传代数

4.计算 $P(0)$中个体的适应值;

5.反复执行下列操作,直到满足遗传算法结束条件:

a) 根据个体适应值及转盘式选择策略确定 $P(g)$内每个个体的选择概率 p_i;

b) for(k=0;k<N;k=k+2)

{

i. 根据概率 p_i 在 $P(g)$内选择两个父体;

ii. r=*random*[0,1];

iii. if($r\le p_m$),对所选 2 个父体执行变异操作,并将所得 2 个后代插入新群体 $P(g+1)$中;

else if($r\le p_m+p_c$),执行杂交操作,并将所得 2 个后代插入新群体 $P(g+1)$中;

else,执行繁殖操作,将 2 个父体不变地插入新群体 $P(g+1)$中;

}//end for(k=0;k<N;k=k+2)

c) 计算 $P(g+1)$中个体的适应值,g=g+1;

//遗传算法与蚂蚁算法衔接部分:

6.从 $P(g)$中选择适应能力强的前 10%个体(5 个),放入集合 $S_{10\%\text{better}}^{gene}$ 中,作为优化解集合;

7.对于 $S_{10\%\text{better}}^{gene}$ 中的每个优化解 s,按前面描述的"遗传算法求解结果向信息素值转换"策略、"蚂蚁算法信息素初值设置"策略以及式(5),计算任务图中各边 e_{ij} 关于 c_k 的信息素初值 $\tau_{ij}^{S}(k)$,k=1,2;

//蚂蚁算法部分:

8.初始化蚂蚁算法控制参数(α=β=1,w_t=1,w_a=2,信息素挥发率ρ=0.2,常数 Q=1000);

9.设置蚂蚁算法结束条件($Ant_{\max}$=100,Ant_{die}=3,$Ant_{\text{min-improv-ratio}}$=0.5%);

10.在节点 t_0 处放置 m 只蚂蚁;//t_0 是唯一的起点,m 取为任务图的平均分支数

11.每只蚂蚁按任务图所对应 DAG 中边的方向遍历所有的边和节点,最终到达节点 t_n.每只蚂蚁在遍历过程中,都要依据式(3)和式(4)计算的概率,猜测所达节点的颜色,并最终确定一个可行的着色方案 s_i,i=1,…,m; //t_n 是唯一的终点

12.计算所得 m 种着色方案的适应值,并从中选择最佳方案 s_{best}=arg min $time_s$; //即满足面积约束并且运行时间最小的着色方案

13.依据式(1)和式(2)更新任务图中所有边的信息素值;

14.若满足蚂蚁算法结束条件,报告最佳着色方案 s_{best} 并退出;否则,继续执行步骤 10.

3.5 算法分析与讨论

(1) 复杂性分析

DCG3A 算法的运行时间 T_{DCG3A} 主要由遗传算法部分运行时间 T_{GA}、蚂蚁算法部分运行时间 T_{AA} 和衔接部分运行时间 T_{JOINT} 组成,可表示为 $T_{DCG3A}=T_{GA}+T_{AA}+T_{JOINT}$.DCG3A 中的遗传算法与蚂蚁算法过程采用相同的适应值计算规则,因此我们可以使用一致的时间值 $T_{cal\text{-}fitness}$ 表示这两个算法中计算每个划分方案适应值所需的时间.下面分别分析 T_{GA},T_{AA} 与 T_{JOINT} 的大小:

• T_{GA} 包括遗传算法初始设置时间 $T_{GA\text{-}init}$(步骤 1~步骤 3)、终止条件判断时间 $T_{GA\text{-}termi}$ 和种群进化迭代时间(步骤 4、步骤 5).假设遗传算法进化了 I_{GA} 代,种群规模为 N,种群中每个个体完成一次遗传操作(步骤 5 中的过程 a,b)所需的时间为 $T_{GA\text{-}operation}$,那么,种群迭代进化时间就是 $I_{GA}\times N\times(T_{GA\text{-}operation}+T_{cal\text{-}fitness})$.这样,$T_{GA}=T_{GA\text{-}init}+T_{GA\text{-}termi}+I_{GA}\times N\times(T_{GA\text{-}operation}+T_{cal\text{-}fitness})$,而 $T_{GA\text{-}init}+T_{GA\text{-}termi}$ 远小于 $I_{GA}\times N\times(T_{GA\text{-}operation}+T_{cal\text{-}fitness})$,因此,$T_{GA}\approx I_{GA}\times N\times(T_{GA\text{-}operation}+T_{cal\text{-}fitness})$.

• T_{AA} 包括蚂蚁算法初始设置时间 $T_{AA\text{-}init}$(步骤 8、步骤 9)、终止条件判断时间 $T_{AA\text{-}termi}$ 和蚂蚁算法迭代时间(步骤 10~步骤 14).假设蚂蚁算法迭代了 I_{AA} 次,每次迭代中使用了 m 只蚂蚁,每只蚂蚁完成一次任务图着色(即求取一个解)的时间为 $T_{AA\text{-}operation}$,那么,蚂蚁算法的迭代时间就是 $I_{AA}\times m\times(T_{AA\text{-}operation}+T_{cal\text{-}fitness})$.这样,$T_{AA}=T_{AA\text{-}init}+T_{AA\text{-}termi}+I_{AA}\times m\times(T_{AA\text{-}operation}+T_{cal\text{-}fitness})$,而 $T_{AA\text{-}init}+T_{AA\text{-}termi}$ 远小于 $I_{AA}\times m\times(T_{AA\text{-}operation}+T_{cal\text{-}fitness})$,因此,$T_{AA}\approx I_{AA}\times m\times(T_{AA\text{-}operation}+T_{cal\text{-}fitness})$.

• T_{JOINT} 包括构造优化解集合(步骤 6)所需的时间 $T_{JOINT\text{-}10\%better}$ 与计算信息素初值(步骤 7)所需的时间 $T_{JOINT\text{-}initial\text{-}pheromone}$.假设采用逐个顺序比较的方法构造优化解集合,那么 $T_{JOINT\text{-}10\%better}\approx N\times N\times 10\%\times T_{float\text{-}compare}$(其中,$T_{float\text{-}compare}$ 为完成一次浮点数比较所需的时间;10%表示选择了适应值高的 10%个体).进一步假设任务图节点数为 K,边数为 H,每次计算一条边上的信息素值的时间为 $T_{JOINT\text{-}edge\text{-}pheromone}$,那么,信息素初值计算时间 $T_{JOINT\text{-}initial\text{-}pheromone}$ 就是 $N\times 10\%\times H\times T_{JOINT\text{-}edge\text{-}pheromone}$.这样,$T_{JOINT}\approx N\times N\times 10\%\times T_{float\text{-}compare}+N\times 10\%\times H\times T_{JOINT\text{-}edge\text{-}pheromone}$.由上述分析可知,$T_{DCG3A}\approx I_{GA}\times N\times(T_{GA\text{-}operation}+T_{cal\text{-}fitness})+I_{AA}\times m\times(T_{AA\text{-}operation}+T_{cal\text{-}fitness})+N\times N\times 10\%\times T_{float\text{-}compare}+N\times 10\%\times H\times T_{JOINT\text{-}edge\text{-}pheromone}$.实际运算中,我们发现 $T_{cal\text{-}fitness}$ 的复杂度为 $O(K\times H)$,远大于 $T_{GA\text{-}operation}$,$T_{AA\text{-}operation}$ 以及 $N\times N\times 10\%\times T_{float\text{-}compare}+N\times 10\%\times H\times T_{JOINT\text{-}edge\text{-}pheromone}$.因此,可以将 T_{DCG3A} 简化为 $T_{DCG3A}\approx I_{GA}\times N\times T_{cal\text{-}fitness}+I_{AA}\times m\times T_{cal\text{-}fitness}=(I_{GA}\times N+I_{AA}\times m)\times T_{cal\text{-}fitness}$.

在运行过程中,遗传算法部分所需的存储空间约为 $2\times N\times O(K)$,蚂蚁算法部分所需的存储空间约为 $m\times O(K+H)$,算法衔接部分构造优化解集合所需的存储空间为 $10\%\times N\times O(K)$.当然,不同的程序实现,对算法运算时间与存储空间也有一定的影响.

(2) 算法运行效率探讨

上述复杂性分析表明,DCG3A 算法的运行时间 T_{DCG3A} 主要取决于 $I_{GA}\times N+I_{AA}\times m$ 与 $T_{cal\text{-}fitness}$ 的乘积.其中,$T_{cal\text{-}fitness}$ 是对某个划分解进行评价(求适应值)所需的时间,与所用的优化算法无关.对于特定的组合优化问题,$T_{cal\text{-}fitness}$ 本质上是评价问题空间中一个解所需的时间.因此,DCG3A 算法效率改进的关键在于最小化 $I_{GA}\times N+I_{AA}\times m$,其中 N,m 是控制参数,分别表示遗传算法的种群规模和蚂蚁算法的蚂蚁数,在算法中一般是固定的值.极端情况下,当遗传算法迭代次数 I_{GA} 为 0 时,DCG3A 算法就退化为蚂蚁算法;当蚂蚁算法迭代次数 I_{AA} 为 0 时,DCG3A 算法则退化为遗传算法.

DCG3A 在遗传算法运行过程中加入了子代优化解改进效率检测机制,使得当遗传算法对优化解改进效率降低到一定程度时,及时终止遗传过程,从而避免 I_{GA} 无谓的增长.此外,由于有了较为准确的初始信息素,使蚂蚁算法大大降低了用于形成最优解上信息素所需的迭代次数.这时,利用蚂蚁算法正反馈、高效收敛的优势,可以在 I_{AA} 很小的情况下,迅速找到最优解.因此,DCG3A 算法通过抑制 I_{GA} 与 I_{AA} 无谓的增长,从而最小化 $I_{GA}\times N+I_{AA}\times m$ 来达到提高搜索效率的目的.

4 实 验

采用与文献[12]相似的实验模型和方法,与基于遗传算法[1]、基于蚂蚁算法[12]的软硬件划分进行比较.

4.1 目标系统结构

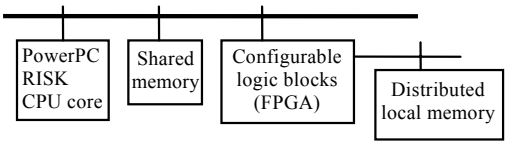

Fig.3 Abstract model of target architecture
图 3 目标系统结构抽象模型

本文考虑软硬件系统任务双路划分问题,目标系统中有 1 个处理器和 1 个硬件可编程部件(如 FPGA).使用 Xilinx Virtex II Pro Platform FPGA 作为参考模型,它最多可包含 4 个处理器核,13 404 个可编程逻辑块.实验中,我们限制使用 1 个处理器和最多 1 232 个可编程逻辑块.目标系统结构抽象模型如图 3 所示.

4.2 实验假设

为简化实验,作如下合理的假设:

a) 任务在处理器和在可编程部件上实现所需的运行时间和占用的硬件面积是静态的,可预先计算出来;

b) 任务间通信开销不变并已经包含在任务的开销(时间、面积)中;

c) 不考虑硬件实现的并行性.

4.3 实验环境与实例

目前,在嵌入式系统软硬件划分领域,国际上还没有公认的测试集[12].常见做法是随机生成有向无环图,并对节点与边赋以属性.我们采用如下方法构造测试集:a) 用 GVF 工具包随机生成指定节点数、指定平均分支数的若干个有向无环图,作为任务图;b) 将每个任务与一个函数(或算法过程)相关联,并以此函数的开销(运行时间、硬件面积、通信代价等)作为任务开销;c) 从 MediaBench 基准程序包中选择与任务相关联的函数.

利用上述方法生成 7 组 DAG(分别记为 DAG_{20},DAG_{30},…,DAG_{80}),每组包含 30 个 DAG 样本.7 组 DAG 的节点数分别为 20,30,…,80,平均分支数依次为 5,5,7,7,9,9,11.以各组 30 个样本的性能平均值作为该组的最终性能结果.

实验环境为:a) AMD Duron 700MHz 处理器,128MB 内存,b) Linux 7.3 操作系统,c) KDevelop 2.1 编程环境.

4.4 实验结果与分析

表 1 给出了本文 DCG3A 算法与遗传算法[1]、蚂蚁算法[12]进行软硬件划分得到的实验数据.其中,遗传算法与蚂蚁算法中控制参数的设置与 DCG3A 中相应的设置相同.我们使这 3 种算法求解的最优解与理论最优解之间的误差基本一致,以保证公平性.图 4 描述了表 1 中 3 种算法求解时间与节点规模的关系.

Table 1 Comparison of genetic algorithm (GA), ant algorithm (AA) and DCG3A algorithm

表 1 遗传算法、蚂蚁算法与 DCG3A 算法对比

TotalDAGNodes	GA[1]		AA[12]		DCG3A	
	Time (ms)	Iterations	Time (ms)	Iterations	Time (ms)	IterationsGA+AA
20	258	72.3	332	43.2	207	23.1+14.6
30	579	65.1	659	36.8	513	31.2+12.4
40	1 296	83.6	1 463	47.3	742	19.4+21.6
50	2 694	89.2	2 235	44.7	1 662	27.7+18.3
60	4 833	92.7	3 280	68.7	2 038	26.8+21.8
70	8 436	68.0	5 762	51.3	2 679	21.4+19.4
80	15 623	97.5	9 368	55.1	3 651	27.2+16.3

当任务图节点数较少(20~30)时,DCG3A 算法略优于遗传算法和蚂蚁算法,但效果不很明显.而当节点数超过 40 时,DCG3A 算法明显优于遗传算法和蚂蚁算法.随着任务图规模的增大,性能改进的效果越显著.

算法控制参数对实验结果也有一定的影响,本实验使用了文献[1,12]中相同的控制参数,以增强可比性.对于特定的组合优化问题,可分别使用相应问题中已验证的优化控制参数,也可以使用典型控制参数设置值[15,16].控制参数 $Gene_{\text{min-impro-ratio}}$ 决定了遗传算法的终止时机,此值若过大,则遗传算法在更接近 t_b(如图 1 所示)的时刻

结束;若过小,遗传算法就将在靠近 t_c 的时刻结束.实验发现,当 $Gene_{\text{min-impro-ratio}}$=3%时,遗传算法在 t_b 与 t_c 之间中间点位置(靠近最佳融合时机 t_a)结束,因此,本实验将 $Gene_{\text{min-impro-ratio}}$ 设置为 3%.

在所使用的目标体系结构下,本实验在一定的硬件面积约束下优化系统运行时间.实际上,在嵌入式系统与 SoC 软硬件划分中,其他性能指标(如功耗)的约束与优化也是非常重要的方面,本文算法同样适用于求解这类划分问题.另外,对于具有强实时性需求的嵌入式系统与 SoC 软硬件划分问题,需要多个嵌入式处理器协同工作,这时,除了运用本文算法进行划分求解外,还需要在划分中考虑嵌入式处理器之间的任务分配与负载平衡,我们将继续对这个问题进行深入的研究.

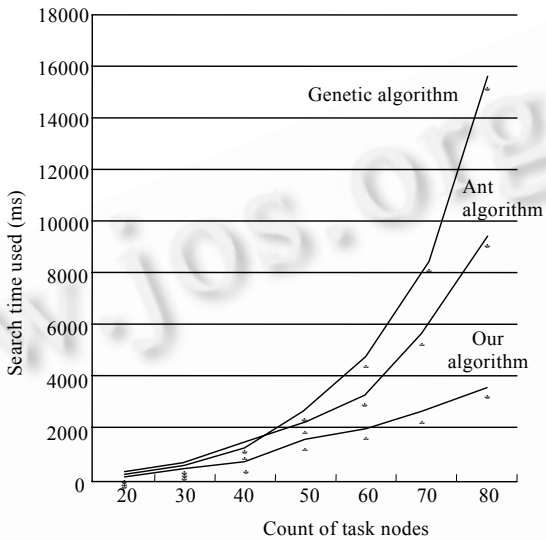

Fig.4 Running-Time/nodes curve

图 4 运算时间-节点规模曲线

5 结 语

DCG3A 算法吸取了遗传算法与蚂蚁算法在寻优问题上各自的优势,克服了它们的不足.在嵌入式系统和 SoC 软硬件划分中取得了很好的效果,并且随着问题规模的增大,性能改进更显著.

下一步我们将研究不同控制参数值对 DCG3A 算法性能的影响,寻找适合 DCG3A 软硬件划分算法控制参数的一组经验值,并将 DCG3A 算法应用于软硬件多路划分问题.

References:

[1] Zhang LF. Research on techniques of hardware/software co-synthesis and virtual microprocessor [Ph.D. Thesis]. Changsha: National University of Defense Technology, 2002 (in Chinese with English abstract).

[2] Gupta RK, Micheli GD. System-Level synthesis using re-programmable components. In: Hugo DM, Herman B, eds. Proc. of the European Conf. on Design Automation (EDAC). Brussels: IEEE Computer Society Press, 1992. 2−7.

[3] Garey MR, Johnson DS. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H.Freeman Company, 1979.

[4] Kastner R. Synthesis techniques and optimizations for reconfigurable systems [Ph.D. Thesis]. Los Angeles: University of California, 2002.

[5] Ernst R, Henkel J, Benner T. Hardware-Software cosynthesis for microcontrollers. IEEE Design & Test of Computers, 1993,10(4): 64−75.

[6] Saha D, Mitra RS, Basu A. Hardware software partitioning using genetic algorithm. In: Agrawal V, Mahabala HN, eds. Proc. of the 10th Int'l Conf. on VLSI Design. Hyderabad: IEEE Computer Society Press, 1997. 155−160.

[7] Guo XD, Liu JR, Wen H. A method for hardware/software partitioning using genetic algorithm. Journal of Computer-Aided Design & Computer Graphics, 2001,13(1):24−27 (in Chinese with English abstract).

[8] Peng Z, Kuchcinski K. An algorithm for partitioning of application specific systems. In: Courtois B, eds. Proc. of the European Conf. on Design Automation (EDAC). Paris: IEEE Computer Society Press, 1993. 316−321.

[9] Else P, Peng Z, Kuchcinski K, Doboli A. System level hardware/software partitioning based on simulated annealing and tabu search. Design Automation of Embedded Systems, 1997,2(1):5−32.

[10] Kalavade A, Lee EA. The extended partitioning problem: hardware/software mapping, scheduling, and implementation-bin selection. Design Automation of Embedded Systems, 1997,2(1):125−163.

[11] Cheng GD, Peng CL. An algorithm for hardware/software partitioning using constraint-driven and elimination of slack time. Journal of Computer Research and Development, 2003,40(6):889−896 (in Chinese with English abstract).

[12] Wang G, Gong WR, Kastner R. A new approach for task level computational resource bi-partitioning. In: Gonzalez TF eds. Proc. of the IASTED Int'l Conf. on Parallel and Distributed Computing and Systems (PDCS). ACTA Press, 2003. 434−444.

[13] Ding JL, Chen ZQ, Yuan ZZ. On the combination of genetic algorithm and ant algorithm. Journal of Computer Research and Development, 2003,40(9):1351−1356 (in Chinese with English abstract).

[14] Li Z, Xu CP, Mo W, Chen G. Initialization for synchronous sequential circuits based on ant algorithm & genetic algorithm. Acta Electronica Sinica, 2003,31(8):1276−1280 (in Chinese with English abstract).

[15] Pan ZJ, Kang LS, Chen YP. Evolutionary Computation. Beijing: Tsinghua University Press, 1998 (in Chinese).

[16] Dorigo M, Maniezzo V, Colorni A. The ant system: Optimization by a colony of cooperating agents. IEEE Trans. on Systems, Man and Cybernetics, Part-B, 1996,26(1):29−41.

[17] Stutzle T, Hoos HH. MAX-MIN ant system. Future Generation Computer System, 2000,16(8):889−914.

附中文参考文献:

[1] 张鲁峰.软硬件协同综合及虚拟微处理器技术研究[博士学位论文].长沙:国防科学技术大学,2002.

[7] 郭晓东,刘积仁,文晖.一种基于遗传算法的硬件/软件划分方法.计算机辅助设计与图形学学报,2001,13(1):24−27.

[11] 程国达,彭澄廉.约束驱动与松弛时间消除相结合的硬/软件划分算法.计算机研究与发展,2003,40(6):889−896.

[13] 丁建立,陈增强,袁著祉.遗传算法与蚂蚁算法的融合.计算机研究与发展,2003,40(9):1351−1356.

[14] 李智,许川佩,莫玮,陈光.基于蚂蚁算法和遗传算法的同步时序电路初始化.电子学报,2003,31(8):1276−1280.

[15] 潘正君,康立山,陈毓屏.演化计算.北京:清华大学出版社,1998.