

基于贪心算法和模拟退火算法的软硬件划分

张良*, 徐成, 田峥, 李涛

(湖南大学 信息科学与工程学院, 长沙 410082)

(*通信作者电子邮箱 zhangliangin2006@163.com)

摘要: 软硬件划分是嵌入式系统设计过程中一个关键环节, 已经被证明是一个 NP 问题。针对目前算法在进行大任务集下的软硬件划分时计算复杂度高、不能快速收敛, 且找到的全局最优解的质量不佳等问题, 提出一种基于贪心算法和模拟退火算法相融合的软硬件划分方法。首先将软硬件划分问题规约为变异的 0-1 背包问题, 在求解背包问题的算法基础上用贪心算法构造出初始划分; 然后, 对代价函数的解空间进行合理的区域划分, 并基于划分的区间设计新的代价函数, 采用改进的模拟退火算法对初始划分进行全局寻优。实验结果表明, 与目前已有的类似改进算法相比, 新算法在任务划分质量和算法运行时间两个方面的提升率最大可达到 8% 和 17% 左右, 具有高效性和实用性。

关键词: 软硬件划分; 启发式算法; 0-1 背包问题; 模拟退火; 代价函数

中图分类号: TP302.7 **文献标志码:** A

Hardware/software partitioning based on greedy algorithm and simulated annealing algorithm

ZHANG Liang*, XU Cheng, TIAN Zheng, LI Tao

(College of Information Science and Engineering, Hunan University, Changsha Hunan 410082, China)

Abstract: Hardware/Software (HW/SW) partitioning is one of the crucial steps in the co-design of embedded system, and it has been proven to be a NP problem. Considering that the latest work has slow convergence speed and poor solution quality, the authors proposed a HW/SW partitioning method based on greedy algorithm and Simulated Annealing (SA) algorithm. This method reduced the HW/SW partitioning problem to the extended 0-1 knapsack problem, and used the greedy algorithm to do the initial rapid partition; then divided the solution space reasonably and designed a new cost function and used the improved SA algorithm to search for the global optimal solution. Compared to the existing improved algorithms, the experimental results show that the new algorithm is more effective and practical in terms of the quality of partitioning and the running time, and the promotion proportions are 8% and 17% respectively.

Key words: Hardware/Software (HW/SW) partitioning; heuristic algorithm; 0-1 knapsack problem; Simulated Annealing (SA); cost function

0 引言

随着半导体工艺的发展, 由通用微处理器和硬件处理单元组成的新型嵌入式系统的应用越来越广泛。软硬件协同设计已成为嵌入式领域的主流技术。众所周知, 硬件的执行速度快, 但往往资源有限, 且成本较高; 而软件的执行速度较慢, 但操作灵活, 成本较低。因此, 能否合理地把任务划分到软硬件执行, 将对整个嵌入式系统的性能有着重要的影响^[1]。

软硬件划分是软硬件协同设计过程中的一个关键问题, 该问题已经被证明是一个典型的多目标优化问题^[2]。关于软硬件划分的问题目前国内外已经有了大量的研究, 主要集中在利用启发式算法寻求近似全局最优解。主流的算法有贪心算法^[3]、遗传算法^[4-5]和模拟退火算法^[6-9]。

使用贪心算法^[3]最大的特点是时间复杂度低, 但易陷入局部最优解。模拟退火算法具有能够跳出局部最优解, 寻求近似全局最优解的特点。理论上, 模拟退火算法可以找到全

局最优解, 但收敛速度很慢, 执行时间长。文献[6]提出了一种改进的模拟退火算法, 在代价函数和解空间方面都有所改进, 但模拟退火算法解的初始化比较耗时, 对大任务集来说, 收敛速度依旧缓慢。文献[10]和文献[11]把贪心算法和模拟退火算法融合在一起, 与本文的思想最接近, 对于大任务集有较好的效果, 但文献[10]中对代价函数中解空间的搜索方向的区间划分过于粗糙, 没能对特殊的解进行特殊处理。文献[11]提出了一种新的接受规则 (Improved Acceptance Criteria, IAC), 对解空间的搜索方向进行引导, 但新的 IAC 使用固定的接收概率, 舍弃了原有 IAC 的随机性, 增大了陷入局部最优解的可能性。本文在文献[10]和文献[11]的系统模型上, 借鉴求解 0-1 背包问题的策略, 提出了一种改进的基于贪心退火算法的软硬件划分方法: 采用贪心算法对初始解进行预划分, 然后采用模拟退火算法寻求最优解, 在模拟退火算法中, 本文对代价函数的解空间进行合理的区域划分, 并基于划分的区间设计新的代价函数。实验表明本文所提算法与

收稿日期: 2013-01-28; 修回日期: 2013-01-28。

基金项目: 国家自然科学基金资助项目 (60973030); 湖南省科研条件创新专项 (2010TT1002)。

作者简介: 张良 (1987 -), 男, 河南南阳人, 硕士研究生, 主要研究方向: 嵌入式系统; 徐成 (1962 -), 男, 湖北蕲春人, 教授, 博士, CCF 会员, 主要研究方向: 嵌入式系统、数字视频处理、机械控制与自动化; 田峥 (1983 -), 男, 湖南长沙人, 博士研究生, CCF 会员, 主要研究方向: 计算机视觉、模式识别; 李涛 (1986 -), 男, 湖南郴州人, 硕士研究生, 主要研究方向: 嵌入式系统。

文献[11]中算法相比,在划分质量方面取得了更好的效果。

1 软硬件划分模型

1.1 基本假设

在实际应用中存在多种不同的嵌入式体系结构,例如单处理器结构、多处理器结构;不同处理单元之间的通信机制也有多种方式。为了研究方便,本文假设系统由一个处理器和一个可编程逻辑单元组成。处理器代表软件执行模块,可编程逻辑单元代表硬件执行模块。任务在处理器和可编程逻辑单元上运行所消耗的时间和面积是静态的,可事先计算出来。软件和硬件之间通过共享内存的方式通信。硬件受软件控制,并且软件和硬件不能同时运行^[12]。如果两个任务同时被映射到硬件(软件),则两个任务之间的通信代价可忽略不计;如果一个任务被映射到硬件(软件),另一个任务被映射到软件(硬件),那么它们之间的通信代价需要考虑。

1.2 计算模型

本文采用和文献[3]类似的有向无环图(Directed Acyclic Graph, DAG)来描述一个系统模型,记作 $G = \{V, E\}$, 其中 V 表示任务节点的集合, $v_i \in V (1 \leq i \leq n)$ 表示任务 i , 用一个三元组 $v_i = (T_i^w, T_i^{hw}, A_i)$ 来表示, 第一个参数 T_i^w 表示任务 i 在软件上执行所需要的时间, 第二个参数 T_i^{hw} 表示任务 i 在硬件上执行所需要的时间, 第三个参数 A_i 表示任务 i 在硬件上执行所消耗的硬件代价。 $E = \{e_{ij} \mid i, j \in N \text{ 且 } i < j\}$ 表示边的集合, e_{ij} 表示任务 i 调用了任务 j , 权值 $|e_{ij}|$ 表示通信代价。

硬件划分问题的实质就是找出一种合理的划分方法 P 把任务节点 V 划分为两个子集, 即 $P = \{V_h, V_s\}$, V_h 表示划分到软件执行的任务集, V_s 表示划分到硬件执行的任务集, 其中 $V_h \cup V_s = V$ 且 $V_h \cap V_s = \emptyset$ 。如果用 $x = (x_1, x_2, \dots, x_n)$ 表示一个划分 P , $x_i = \{0, 1 \mid i = 1, 2, \dots, n\}$, $x_i = 1$ 表示任务 i 划分到硬件执行, $x_i = 0$ 表示任务 i 划分到软件执行, 则系统的硬件代价 $H(X)$, 软件代价 $S(X)$ 以及通信代价 $C(X)$ 如式(1)~(3):

$$H(X) = \sum_{i=1}^n T_i^{hw} \cdot x_i \quad (1)$$

$$S(X) = \sum_{i=1}^n T_i^w \cdot (1 - x_i) \quad (2)$$

$$C(X) = \sum_{i=1}^n \sum_{j=1}^n |e_{ij}| \cdot |x_i - x_j| \quad (3)$$

系统总的时间开销可以表示为:

$$T(X) = H(X) + S(X) + C(X) \quad (4)$$

在实际的应用中, 硬件资源都是有限的, 假设可提供的最大硬件面积为 A_c , 那么划分到硬件上执行的任务的总的硬件开销满足:

$$\sum_{i=1}^n A_i \cdot x_i \leq A_c$$

由上可知, 软硬件划分问题 P 是一个多目标优化问题, 模型的形式化描述为:

$$P \begin{cases} \text{minimize} & T(X) \\ \text{subject to} & \sum_{i=1}^n A_i \cdot x_i \leq A_c \\ & x_i = \{0, 1 \mid i = 1, 2, \dots, n\} \end{cases}$$

1.3 0-1 背包问题规约

背包问题(Knapsack Problem)是一种组合优化的 NP 完

全问题。问题可以描述为: 给定一组物品, 每种物品都有自己的重量和价格, 在限定的总重量内, 如何选择才能使得物品的总价格最高。

0-1 背包问题是背包问题的一种特例, 即每一个物品只能被选择或不被选择, 问题可描述为:

$$0/1KP \begin{cases} \text{maximize} & \sum_{i=1}^n b_i x_i \\ \text{subject to} & \sum_{i=1}^n w_i x_i \leq K \\ & x_i = \{0, 1 \mid i = 1, 2, \dots, n\} \end{cases}$$

其中: b_i 和 w_i 分别表示物品 i 的价格和重量, $x_i = 1$ 表示物品 i 放到背包中, $x_i = 0$ 表示物品 i 没有放到背包中。

由式(1)~(4)可得:

$$T(X) = \sum_{i=1}^n T_i^{sw} - \sum_{i=1}^n (T_i^{sw} - T_i^{hw}) x_i + C(X)$$

本文中软硬件划分的粒度为粗粒度, 所以通信代价 $C(X)$ 和软硬件运行时间相比可以忽略。在任务集确定的情况下, $\sum_{i=1}^n T_i^{sw}$ 为固定常数, 因此, 要使得 $T(X)$ 最小, 只要使得

$\sum_{i=1}^n (T_i^{sw} - T_i^{hw}) x_i$ 最大即可。令 $\Delta T_i = T_i^{sw} - T_i^{hw}$, 通过等价变换, 软硬件划分模型可以转换为:

$$P \begin{cases} \text{maximize} & \sum_{i=1}^n \Delta T_i \cdot x_i \\ \text{subject to} & \sum_{i=1}^n A_i \cdot x_i \leq A_c \\ & x_i = \{0, 1 \mid i = 1, 2, \dots, n\} \end{cases}$$

令 ΔT_i 和 A_i 分别对应着 0-1 背包问题中的 b_i 和 w_i , 因此, 可以借用 0-1 背包问题的经典解决方案来处理软硬件划分问题。

2 改进的贪心退火算法

2.1 贪心算法的预划分

贪心算法是一种简单的启发式算法, 是解决 0-1 背包问题的常用算法, 通常的思路是: 先按照价格与重量的比值对物品进行排序, 如式(5)所示, 然后根据此顺序依次将物品放入背包, 直到达到约束条件的上限为止。

$$b_1/w_1 \geq b_2/w_2 \geq \dots \geq b_n/w_n \quad (5)$$

这样的算法计算复杂度低, 执行速度快, 文献[3]就是基于这样的贪心思路来搜索一维空间获得最优解。但这样的算法搜索的解空间有限, 容易陷入局部最优解, 难以找到全局最优解。

本文提出的算法首先利用文献[3]的贪心策略进行软硬件预划分, 即先按照 ΔT 与 A_i 的比值对任务集进行排序, 然后依次把任务放到硬件上去执行, 直到超过硬件资源约束值为止。算法的形式化描述如下:

Input: 任务图 G 和约束条件 A_c 。

Output: 预划分结果 $x_{\text{pre}} = (x_1, x_2, \dots, x_n)$ 。

Algorithm: Alg_Greedy。

1) 任务全部划分到软件上 $x_{\text{pre}} = (0, 0, \dots, 0)$ 。

2) 求出每个任务 V_i 的价格重量比 $q_i = \Delta T_i / A_i$, 并按照降序对任务集进行排列, 建立新任务集 $Q = V$ 。

3) $i = 1$; 剩余硬件面积 $A_{\text{res}} = A_c$ 。

4) 找出 Q 中价格重量比最大的任务 v_k , 如果 $A_k \leq A_{\text{res}}$,

则把任务 v_k 放到硬件执行, 即 $x_k = 1$, 把 v_k 从 Q 中移除, $A_{res} = A_c - A_k$; 否则, 执行空操作。

5) $i = i + 1$, 如果 $i < n$ 且 $A_{res} > 0$, 回到第4)步; 否则, 转到第6)步。

6) 输出预划分结果 $x_{pro} = (x_1, x_2, \dots, x_n)$ 。

算法 Alg_greedy 的时间复杂度主要集中在任务的排序(第2)步和任务的划分(第4)步, 当任务集的任务数为 n 时, 这两个操作的计算复杂度分别为 $O(n \log n)$ 和 $O(n)$, 所以, 算法 Alg_greedy 的时间复杂度为 $O(n \log n) + O(n)$, 即 $O(n)$ 。因为基于这样的策略进行预划分时没有考虑任务在软硬件之间的通信代价, 所以, 产生的解不是一个可行解, 需要对解空间进行优化。

2.2 模拟退火算法

2.2.1 算法描述及分析

模拟退火算法最早的思想由 Metropolis 等提出, Kirkpatrick 等将其应用于组合优化问题中, 与贪心算法相比, 该算法通过赋予搜索过程一种时变且最终趋于零的概率突跳性, 从而可有效避免陷入局部最优解并最终找到近似全局最优解。在预划分的基础上, 本文采用模拟退火算法对预划分的解进行优化, 搜索全局最优解。算法的形式化描述如下:

Input: 贪心算法的预划分结果 x_{pro} ; 任务图 G 和约束条件 A_c 。

Output: 最终划分结果 $x_{end} = (x_1, x_2, \dots, x_n)$ 。

Algorithm: Alg_SA。

1) 给定当前划分 $x_{cur} = x_{pro}$, 当前温度 $T_{cur} = T$ (初始值), 以及连续未接受次数 $m = 0$ 。

2) For $i = 1$ to K

根据扰动模型产生新划分 x_{new}

根据代价函数计算代价函数差 Δ

根据接收准则决定是否接收 x_{new}

如果 x_{new} 没有被接收, 则 $m = m + 1$;

否则, $m = 0, x_{cur} = x_{new}$

End For

3) T_{cur} 退火。

4) 如果 $T_{cur} > T_{td}$ (T_{td} 为 T_{cur} 的临界值) 且 $m < N$ (N 的临界值), 则令 $m = 0$, 并回到第2步; 否则, 转到第5)步。

5) 输出最终划分结果 x_{end} 。

模拟退火算法虽然能够得到近似全局最优解, 但其计算复杂度高, 收敛速度慢。由上可以看出算法由两层循环构成, 内层循环的执行次数往往是确定数值 K , 因此, 大多数的研究集中在通过修改算法, 进而影响外层循环的次数, 加快算法收敛速度。大量文献通过改动扰动模型和退火进度, 来提高算法收敛速度, 但往往对算法在解空间的搜索方向不加限制, 这会导致算法搜索具有很大的盲目性。文献[6]和[10]研究了这一问题, 文献[6]提出了一种新的代价函数计算方法, 该方法对解空间上的搜索方向进行引导, 达到了算法快速收敛的目的, 但该方法计算复杂。文献[10]简化了文献[6]提出的计算方法, 计算复杂度大大降低, 但由于对解空间的划分过于粗糙, 并不能针对特殊情况作出处理。文献[11]改进了接受规则 IAC。本文基于上述文献的研究成果, 提出了一种新的代价函数计算方法, 对算法的搜索空间进行引导, 以求提高算法收敛速度, 提高划分质量。

2.2.2 代价函数改进

通过扰动模型产生的新的划分与当前的划分相比, 在系

统时间和硬件消耗面积方面必然会有所不同。把当前的划分记为 P , 新的划分记为 $P1$, $P1$ 的系统时间的增量和硬件面积的增量分别记为 ΔT 和 ΔA , 以 P 为原点, ΔT 为横轴, ΔA 为纵轴, 建立坐标轴, 如图1所示, 则 $P1$ 必然会落在四个象限中的某一个象限, 若 $P1$ 处于图1中所示位置, 则表示新划分中总的硬件面积增加, 总的系统时间减少。 ΔA 和 ΔT 的单位不同, 必须要进行归一化处理。

大多数研究在计算代价函数差 Δ 时使用的公式都是 $\Delta = N_i - P_i$, 其中 N_i 表示新划分的系统时间, P_i 表示当前划分的系统时间。从图1可以看出, 当新划分落在在二、三象限时, $\Delta < 0$, 根据 Metropolis 准则, 当 $\Delta < 0$ 时, 新的划分以概率1被接受。但是, 当新的划分处于接近纵轴的位置时, 例如图1中的 $P2$, 该划分对系统时间的改善非常有限, 但却占用了大量硬件资源, 以概率1接受 $P2$ 显然是不合理的, 需要降低它的接受概率。当新划分落在第四象限但处于接近纵轴的位置时, 例如图1中的 $P3$, $\Delta > 0$, 根据 Metropolis 准则, 新划分以一定概率被接受, 此时的划分虽然增加了系统时间, 但却大大节省了硬件资源, 为搜索最优解提供了较大的改进空间, 应该增加接受该划分的概率。

通过以上分析, 可知传统的解空间搜索方法具有很大的盲目性, 因此, 本文从搜索方向上对代价函数进行改进。作一条平分二、四象限的直线 l , l 把二、四象限分为四个区域, 记作区域1、区域2、区域3、区域4, 如图1所示。

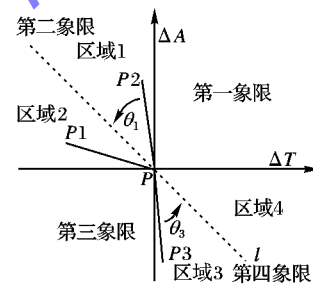


图1 节点扰动模型

对于第一、三象限以及区域2、4来说, 当新划分落在这些区域中时, 都采用 Metropolis 准则来接受。当新的划分落在区域1、3中时, 比较理想的方法(记作 continuous_way)是假设一个参考函数(Reference Function, RF) $f(\theta)$, θ 表示新的划分与直线 l 之间的夹角, 令 $N_i = N_i \cdot f(\theta)$ 。在区域1中, θ 记为 θ_1 , $f(\theta)$ 记为 $f_1(\theta_1)$, 函数 $f_1(\theta_1)$ 的坐标如图2(a)所示, ε_1 表示一个大小可调的参数, $\varepsilon_1 > 1$, 则 $1 \leq f_1(\theta_1) \leq \varepsilon_1$ 。随着划分越来越接近纵轴正半轴, N_i 的值越来越大, $\Delta < 0$ 的概率也就越来越小, 则新划分不再以概率1被接受, 而是和具体所处的位置有关。在区域3中, θ 记为 θ_3 , $f(\theta)$ 记为 $f_3(\theta_3)$, 函数 $f_3(\theta_3)$ 的坐标如图2(b)所示, ε_3 是一个可调的参数, $\varepsilon_3 < 1$, $\varepsilon_3 \leq f_3(\theta_3) \leq 1$, 即随着划分越来越接近纵轴负半轴, N_i 的值越来越小, $\Delta < 0$ 的概率也就越来越大, 划分被接受的概率越来越大。

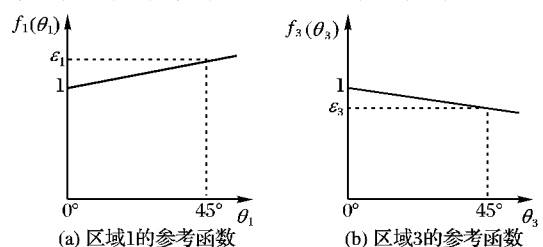
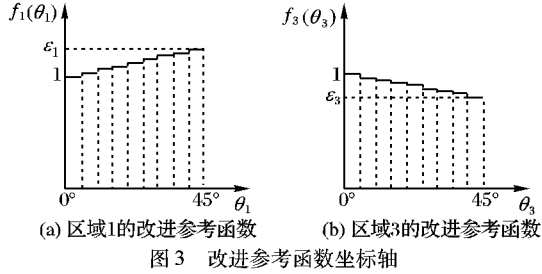


图2 参考函数坐标轴

Continuous_Way 定义了一个连续函数解决遇到的问题,

从应用数学的角度来看是最理想的方法,但在实际应用中存在着计算量大的缺点。在进行实际划分时,没有必要用连续函数对每一个 θ 值都进行赋值。参考微积分领域的相关研究方法,可以把连续函数量化成离散函数,减少运算量,如图3所示,该方法记作 Discrete_Way。



以区域1为例, θ 从 $0^\circ \sim 45^\circ$ 被划分为 n 个区间(zone),则 $f_1(\theta_1)$ 满足式子:

$$f_1(\theta_1) = \begin{cases} 1, & \theta_1 \in \text{zone 1} \\ 1 + 1 \times (\epsilon_1 - 1)/n, & \theta_1 \in \text{zone 2} \\ 1 + 2 \times (\epsilon_1 - 1)/n, & \theta_1 \in \text{zone 3} \\ \vdots \\ 1 + (n-1) \times (\epsilon_1 - 1)/n, & \theta_1 \in \text{zone } n \end{cases}$$

在具体应用的过程中,该方法实际上是在空间和时间之间做了一个折中,利用查表的方法快速确定每个区间的值,以达到减小计算量的目的。

量化的区间个数也是一个重要的参数,量化的区间过多,就陷入了计算量复杂的误区;区间个数过少,划分就过于粗糙,不能对特殊问题进行处理。为了确定划分的区间个数,本文对不同的区间个数下的算法性能进行了测试,取任务数为500,硬件约束比(Hardware Restraining Ratio, HRR)为0.8,划分区间的个数分别取1~8,通过多次实验得出:划分区间个数为4时,划分后的系统时间已接近收敛至最优,如图4所示。因此,本文最终确定划分的区间为4个。

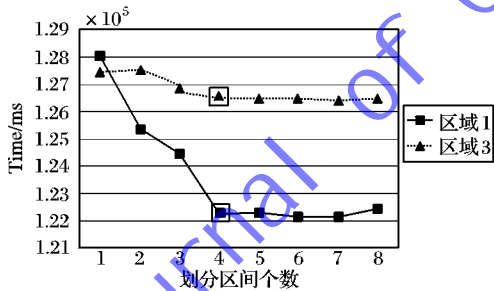


图4 不同区间个数下的系统划分时间

改进的代价函数的算法描述如下,其中 $\rho_1[i]$ 表示在区域1中 θ 对应的区间 i 的函数值, $\rho_3[i]$ 表示在区域3中 θ 对应的区间 i 的函数值:

Input: 扰动产生的新划分 x_{new}, N_t, P_t 。

Output: 代价函数差 Δ 。

Algorithm: costfunction_improved。

根据 x_{new} 的 ΔT 和 ΔA 判断 x_{new} 属于哪个区域

If $x_{\text{new}} \in \text{区域1}$ 和 $\theta \in \text{区间 } i$

Then $N_t = \rho_1[i] \cdot N_t$ ($N \leq \rho_1[i] \leq 1$)

Else if $x_{\text{new}} \in \text{区域3}$ 和 $\theta \in \text{区间 } i$

Then $N_t = \rho_3[i] \cdot N_t$ ($\rho_3[i] \geq 1$)

Else

Then $N_t = N_t$

EndIf

输出代价函数差 $\Delta = N_t - P_t$

3 测试实验与分析

本文采用 TGFF (Task Graphs For Free) 工具随机产生 DAG 大任务图^[13]。为了证明改进算法的有效性,对5个算法进行包括算法执行时间(Algorithm Running Time, ART),算法加速比(Algorithm Acceleration Ratio, AAR)等方面的测试和比较。其中,加速比表明了采用软硬件划分算法之后系统性能相对于在纯软件上执行的提升状况,表示为 $\eta = (1 - T_{\text{alg}}/T_{\text{sw}}) \times 100\%$, T_{alg} 表示算法划分后系统的执行时间, T_{sw} 表示任务在纯软件上执行的时间。这5个算法分别是文献[3]中的一维寻优算法、文献[6]中的改进模拟退火算法、文献[10]中的贪心模拟退火算法、文献[11]中的贪心模拟退火算法,以及本文提出的对解空间的引导规则进行改进的贪心模拟退火算法。在引言部分已经对前四种算法的优缺点进行了分析,引入它们是为了与本文算法进行对比。为方便起见,本文分别把这4个算法叫作 Alg_1D、Alg_ESA、Alg_GSA1、Alg_GSA2和 Alg_Mine。

本文所有的实验测试平台均为:CPU 奔腾双核 3.2 GHz, 2 GB 内存, Windows 7 操作系统。开发环境为 VS2010, 测试代码用 C++ 实现, 本仿真系统需要运用多线程技术和 GUI 界面, 因此选择使用 Qt 作为开发库。

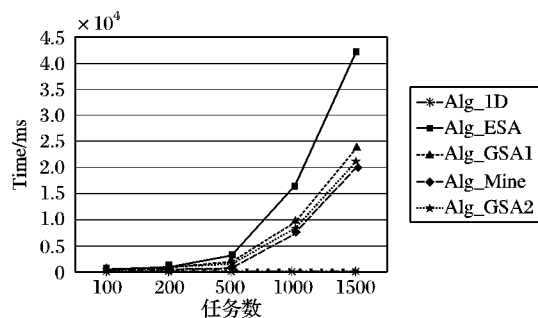
图5(a)~(b)表示的是在硬件约束比为0.3,任务数分别取100,200,500,1000,1500的情况下,五种算法的算法运行时间和加速比。而图5(c)~(d)表示的是在任务集为1000,硬件约束比分别取0.1,0.3,0.4,0.6,0.8的情况下,四种算法的算法运行时间和加速比。

通过对图5的综合对比分析可知:算法 Alg_1D 执行时间最短,比其他四个算法低3~4个数量级,但该算法的加速比最低。这是由于该算法搜索空间小,容易陷入局部最优解从而使得算法提前结束。在算法 Alg_ESA 和本文算法的对比中,不论是在不同任务集下还是在不同的硬件约束比下,本文的算法都大大优于算法 Alg_ESA,算法执行时间提高率为53.1%,加速比提高率为13.6%。说明本算法用贪心算法作预划分,并对代价函数进行改进可以提高算法收敛速度,优化划分解。本文算法与算法 Alg_GSA1 和算法 Alg_GSA2 都采用贪心算法进行预划分,加速了收敛时间,但本文在这两种算法的基础上,对启发式算法中的解空间的搜索区间进行了科学合理的划分,即最大限度地减少了算法搜索的盲目性,提高了搜索到全局最优解的速度;同时,采用 Metropolis 准则作为接受准则,与算法 Alg_GSA2 采用的固定接受概率相比,具有更大的扰动随机性,提高划分质量。本文算法与 Alg_GSA1 相比,算法执行时间提高率为17.7%,加速比提高率为8.2%;与 Alg_GSA2 相比,算法执行时间提高率为5.9%,加速比提高率为7.6%。同时,从图5(b)中的线段趋势可以看出,本文算法在任务集越大的情况下,优势越明显。

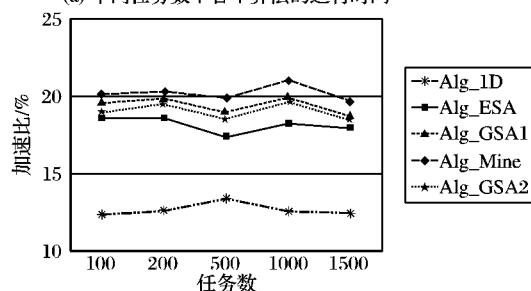
4 结语

本文提出了一种贪心算法和模拟退火算法相结合的软硬件划分算法。该算法主要有两大特点:1)用贪心算法对解空间进行预划分,再用模拟退火算法对预划分的结果进行优化,相比于单纯的模拟退火算法,算法运行速度提高了近一倍。2)对模拟退火算法的代价函数进行改进,合理地引导了对解

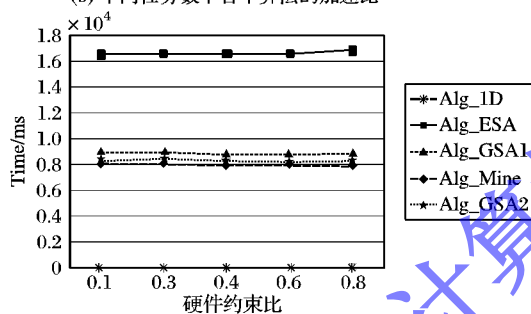
空间的搜索方向,提高算法收敛速度,同时,增加了找到最优解的概率。



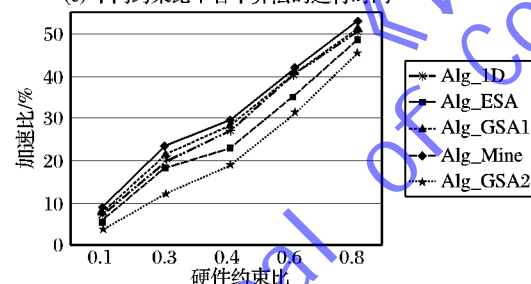
(a) 不同任务数下各个算法的运行时间



(b) 不同任务数下各个算法的加速比



(c) 不同约束比下各个算法的运行时间



(d) 不同约束比下各个算法的加速比

图5 实验结果

但是,该算法仍然有先验参数过多的缺陷,下一步工作的重点是找到一种自适应的参数机制,解决由于先验参数过多造成的算法随机性。同时,研究该算法在多路软硬件划分以及多处理器下的适用性。

参考文献:

- [1] WU J G, SRIKANTHAN T, JIAO T. Efficient heuristics for functional partitioning and scheduling in hardware/software co-design [J]. Design Automation for Embedded Systems, 2008, 12(4): 345 - 375.
- [2] GAREY M R, JOHNSON D S. Computer and intractability: a guide to the theory of NP-completeness [M]. New York: W. H. Freeman Company, 1979.
- [3] WU J G, SRIKANTHAN T. Algorithmic aspects of hardware/software partitioning: 1D search algorithms [J]. IEEE Transactions on Computers, 2010, 59(4): 532 - 544.
- [4] CHEHIDA K B, AUGUIN M. HW/SW partitioning approach for reconfigurable system design [C]//Proceedings of the 2002 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems. New York: ACM Press, 2002: 247 - 251.
- [5] WANGTONG T, CHEUNG P, LUK W. Comparing three heuristic search methods for functional partitioning in hardware-software co-design [J]. Journal of Design Automation for Embedded Systems, 2002, 6(4): 425 - 449.
- [6] BANERJEE S, DUTT N. Efficient search space exploration for HW-SW partitioning [C]//The IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis. New York: ACM Press, 2004: 122 - 127.
- [7] 邢冀鹏, 邹雪城, 刘政林, 等. 一种基于改进模拟退火算法的软硬件划分技术[J]. 微电子学与计算机, 2006, 23(5): 31 - 37.
- [8] ELES P, PENG Z, KUCHCINSKI K, et al. System level hardware/software partitioning based on simulated annealing and Tabu search [J]. Design Automation for Embedded Systems, 1997, 2(1): 5 - 32.
- [9] KORYFIDIS I. Power aware HW/SW partitioning for DVB-H receiver module [D]. Julianalan, Netherlands: Delft University of Technology, 2006.
- [10] 肖平. 基于模拟退火算法的可重构计算系统软硬件划分方法研究[D]. 长沙: 湖南大学, 2011.
- [11] 湖南大学. 一种基于贪心模拟退火算法的软硬件划分的方法: 中国, CN102508721A [P]. 2012 - 06 - 20.
- [12] BASSIRI M M, SHAHMOSEINI H S. On-line HW/SW partitioning and co-scheduling in reconfigurable computing systems [C]//Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology. Piscataway: IEEE Press, 2009: 557 - 562.
- [13] VAKIL-BAGHMISHEH M T, NAVARBAF A. A modified very fast simulated annealing algorithm [C]//2008 International Symposium on Telecommunications. Piscataway: IEEE Press, 2008: 61 - 66.
- [14] Task Graphs for Free (TGFF v3.0) [EB/OL]. [2012 - 11 - 20]. <http://ziyang.eecs.umich.edu/~dickrp/tgff/>.

(上接第1884页)

- [8] GOYAL V, JAIN A, PANDEY O, et al. Bounded ciphertext policy attribute-based encryption [C]//ICALP2008: Proceedings of the 35th International Colloquium on Automata, Languages and Programming, LNCS 5126. Berlin: Springer-Verlag, 2008: 579 - 591.
- [9] WATERS B. Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization [C]//Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography Conference on Public Key Cryptography. Berlin: Springer-Verlag, 2011: 53 - 70.

- [10] CHASE M. Multi-authority attribute based encryption [C]//TCC2007: Proceedings of the 4th Conference on Theory of Cryptography, LNCS 4392. Berlin: Springer-Verlag, 2007: 515 - 534.
- [11] DIMAKIS A G, PRABHAKARAN V, RAMCHANDRAN K. Decentralized erasure codes for distributed networked storage [J]. IEEE/ACM Transactions on Networking-Special Issue on Networking and Information Theory, 2006, 14(SI): 2809 - 2816.
- [12] SHAMIR A. How to share a secret [J]. Communications of the ACM, 1979, 22(11): 612 - 613.