中国科学: 信息科学 2016 年 第 46 卷 第 9 期: 1321–1338
SCIENTIA SINICA Informationis
《中国科学》杂志社
SCIENCE CHINA PRESS
论 文

一种基于从众和声粒子群算法的并行软硬件划分方法

鄢小虎[①②③], 何发智[①②*], 陈壹林[①②]

① 武汉大学软件工程国家重点实验室, 武汉 430072
② 武汉大学计算机学院, 武汉 430072
③ 国网电力科学研究院武汉南瑞有限责任公司, 武汉 430074
* 通信作者. E-mail: fzhe@whu.edu.cn

收稿日期: 2016–01–05; 接受日期: 2016–05–03; 网络出版日期: 2016–09–09
国家自然科学基金 (批准号: 61472289) 和湖北省自然科学基金 (批准号: 2015CFB254) 资助项目

摘要 软硬件划分是软硬件协同设计中关键步骤之一, 并且随着设计复杂度的增加, 逐步成为一个具有挑战性的优化问题. 提出一种基于从众和声粒子群算法 (conformity particle swarm optimization with harmony search, CPSO-HS) 的并行软硬件划分方法. 按生物行为学理论, 个体粒子具有从众行为, 趋向于靠近群体粒子聚集的安全地点, 以避免被捕食者袭击. CPSO-HS 算法通过模拟这种从众行为, 能够保持搜索种群的多样性, 以避免陷入局部最优, 有利于逼近全局最优点. 通过改进和声搜索算法 (harmony search, HS) 的初始化策略, 将 HS 集成到 CPSO-HS 中, 在当前全局最优解附近提高算法的搜索精度, 有利于提升解的质量. 以上两步的有机结合, 增强了 CPSO-HS 算法搜索的多样性和集中性. 进一步考虑软硬划分方法的特殊性, 其中最耗时的过程是计算软硬件的通讯代价, 因此在常用的 PC 平台上采用并行策略加速该过程, 以便在大规模的软硬件划分问题中有效减少整体运行时间. 最后, 通过基准任务测试集验证了本文方法的有效性.

关键词 软硬件划分 粒子群算法 和声搜索 背包问题 通讯代价 并行计算

1 引言

软硬件划分是软硬件协同设计中最关键的一个步骤, 它是在满足某些约束的条件下, 将系统模块 “最优地” 分配给软硬件部分执行. 软件部分执行灵活性好、代价低, 但是需要耗费更多的时间; 硬件部分执行速度快, 但是成本较高 [1]. 所以, 软硬件划分的主要目的是在一定的限制条件下, 将所有的计算任务合理地映射到软件或硬件上, 使得整个系统的性能最优化. 传统的软硬件划分是通过手工进行的, 但随着嵌入式系统协同设计复杂度的增加, 软硬件划分已成为一个具有挑战性的优化问题.

软硬件划分问题是一类 NP 完全问题, 目前国内外学者在软硬件划分问题方面做了大量的研究工作, 解决方法主要包括精确算法和启发式算法两类 [2]. 精确算法有动态规划算法 [3, 4]、分支限界法

引用格式: 鄢小虎, 何发智, 陈壹林. 一种基于从众和声粒子群算法的并行软硬件划分方法. 中国科学: 信息科学, 2016, 46: 1321–1338, doi: 10.1360/N112016-00006

(branch-and-bound, B&B) [5] 和整数线性规划法 (integer linear programming, ILP) [6] 等. 启发式算法主要包括遗传算法 (genetic algorithm, GA) [7,8]、模拟退火算法 (simulated annealing, SA) [9,10]、贪心算法 [11]、禁忌搜索算法 (tabu search, TS) [12,13]、蚁群算法 (ant colony optimization, ACO) [14] 和粒子群算法 (particle swarm optimization, PSO) [15,16] 等. 精确算法一般用于较小规模的划分问题, 当问题规模较大时, 需要采用启发式算法. 这些划分算法在各自的协同设计环境下取得了非常好的效果, 但是由于它们之间巨大的差异性, 很难比较这些算法的性能 [17].

近些年, 粒子群算法因其良好的性能, 已广泛应用于软硬件划分问题中. 利用 PSO 求解大规模的软硬件划分问题, 文献 [15] 发现粒子群算法在代价函数和处理时间上优于遗传算法. 为了避免早熟收敛, 文献 [16] 提出了一种改进的粒子群算法求解软硬件划分问题. 文献 [18] 提出一种简化的软硬件划分模型, 并验证了基于 PSO 的软硬件划分方法在性能上优于 ILP, GA 和 ACO. 文献 [19] 结合离散的粒子群算法和分支限界法求解软硬件划分问题, 离散的粒子群算法用于加速分支限界法. 文献 [20] 融合 PSO 和 TS 求解软硬件划分问题, 利用 GA 的复制和交叉操作避免 PSO 的早熟收敛问题. 这些改进的粒子群算法未从搜索的多样性和集中性两方面同时考虑, 本文提出的 CPSO-HS 通过从众行为增加搜索的多样性, 集成 HS 算法增强搜索的集中性, 改进的效果更加明显.

在求解大规模软硬件划分问题时, 启发式算法耗费的时间太长, 因此相关研究提出基于高性能服务器和集群的求解方案 [21]. 但是由于这些基于启发式算法的软硬件划分方法是串行程序, 很难有效地提升算法的运行速度. 为了减少运行时间, 文献 [20] 提出基于多 CPU 实现的并行混合粒子群算法求解软硬件划分问题, 文献 [22] 将基于 MPI 实现的并行遗传算法应用于软硬件划分问题, 文献 [23] 提出采用 GPU 并行加速软硬件划分过程. 这些方法都是对整个软硬件划分过程进行加速, 本文则通过多核并行加速软硬件划分方法中最耗时的通讯代价的计算过程, 更能减少运行时间.

本文讨论的软硬件划分问题基于文献 [2,11] 中相同的假设和模型, 将软硬件划分问题定义为一个扩展的 0/1 背包问题, 提出一种并行的从众和声粒子群算法求解这类问题. 为了避免早熟收敛和陷入局部最优, CPSO-HS 算法模拟粒子的从众行为, 个体粒子趋向于靠近群体粒子聚集的安全地点. 为了提高算法的搜索精度和提升解的质量, 改进和声搜索算法的初始化策略, 将 HS 集成到 CPSO-HS 中, 在当前全局最优解附近寻找更优的位置. 为了加速基于 CPSO-HS 的软硬件划分方法, 软硬件通讯代价在多核并行环境中计算.

与遗传算法不同, CPSO-HS 不是通过遗传算子进化, 而是通过个体间协作与竞争来寻找最优解. 在 CPSO-HS 的每次迭代中, 每个粒子根据自身经历的最优位置、全局最优位置和安全位置寻优, 没有遗传算法的编解码、选择、杂交和变异等复杂运算. 与 PSO 相比, CPSO-HS 通过模拟粒子的从众行为, 增加搜索的多样性; 通过集成 HS 算法, 增强搜索的集中性. 因此, CPSO-HS 收敛速度快、计算简单、全局搜索能力强、时间和空间复杂度低, 更加适合求解大规模的软硬件划分问题.

本文的组织结构如下: 第 2 部分描述相关的工作, 第 3 部分提出从众和声粒子群算法, 第 4 部分提出一种基于从众和声粒子群算法的并行软硬件划分方法, 第 5 部分通过求解基准任务对算法的性能进行分析, 第 6 部分对本文进行总结.

2 相关工作

2.1 问题定义

本文采用文献 [2,11] 中提出的软硬件划分模型, 利用无向图 $G = (V, E)$ 描述系统任务, 其中 V 和

E 分别是结点和边的集合, s_i, h_i 和 c_i 分别表示结点 v_i 的软件、硬件和通讯代价. 软硬件划分问题可以描述为最小化优化问题 P:

$$P\begin{cases}\min \quad \sum_{i=1}^{n} h_i(1-x_i),\\ \text{s.t. } \sum_{i=1}^{n} s_i x_i + C(\boldsymbol{x}) \leqslant R, \quad x_i \in \{0,1\}, \quad i=1,2,\ldots,n,\end{cases} \tag{1}$$

其中 $\boldsymbol{x}=(x_1,x_2,\ldots,x_n)$ 表示软硬件划分问题的一个解. x_i=1 (x_i=0) 表示该结点是软件 (硬件) 划分. $C(\boldsymbol{x})$ 表示划分 $\boldsymbol{x}$ 的通讯代价, R 是约束条件. 通过简单的变换, 最小化优化问题 P 可以转换为最大化优化问题 Q:

$$Q\begin{cases}\max \quad \sum_{i=1}^{n} h_i x_i,\\ \text{s.t. } \sum_{i=1}^{n} s_i x_i + C(\boldsymbol{x}) \leqslant R, \quad x_i \in \{0,1\}, \quad i=1,2,\ldots,n.\end{cases} \tag{2}$$

在文献 [11] 中, $C(\boldsymbol{x})$ 由 uR 代替, 其中 $0 \leqslant u \leqslant 1$, 从而问题 Q 被转换为问题 Q':

$$Q'\begin{cases}\max \quad \sum_{i=1}^{n} h_i x_i,\\ \text{s.t. } \sum_{i=1}^{n} s_i x_i \leqslant (1-u)R, \quad x_i \in \{0,1\}, \quad i=1,2,\ldots,n.\end{cases} \tag{3}$$

背包问题 KP 和软硬件划分问题 Q 非常相似, 在背包问题中, 二进制 x_i=1 (x_i=0) 代表物品 i 在 (不在) 背包中. 背包中的每个物品拥有一个重量 w_i 和一个收益 b_i, 背包问题的描述如下:

$$\text{KP}\begin{cases}\max \quad \sum_{i=1}^{n} b_i x_i,\\ \text{s.t. } \sum_{i=1}^{n} w_i x_i \leqslant K, \quad x_i \in \{0,1\}, \quad i=1,2,\ldots,n,\end{cases} \tag{4}$$

其中 K 是背包的容量, 问题 Q 中 h_i, s_i 和 R 分别对应于背包问题 KP 中的 b_i, w_i 和 K. 两个问题的唯一差别在于问题 Q 中有通讯代价 $C(\boldsymbol{x})$, 所以本文中软硬件划分问题为一个扩展的 0/1 背包问题.

2.2 启发式算法

目前, 很多启发式算法被用于求解软硬件划分的问题. 文献 [11] 将软硬件划分问题定义为一类扩展的 0/1 背包问题, 提出了 3 种启发式的一维搜索算法求解这类问题. 在这 3 种算法中, Alg-new3 的性能最佳, 本文将其命名为 Base. 算法 Base 的时间复杂度为 $O(n\log n + d(n+m))$, 其中 n 是结点的数目, m 是边的数目. 文献 [13] 提出了一种启发式算法 Heur, 其时间复杂度为 $O(n\log n + m)$. 在算法 Heur 中, 为了得到可行解, Adjust_Out 用于移出背包中 $\frac{h_i}{s_i+c_i}$ 值最小的物品. 为了优化当前的解, Adjust_In 用于移入背包中 $\frac{h_i}{s_i+c_i}$ 值最大的物品. 在文献 [21] 中, NodeRank 算法用于求解软硬件划分问题, NodeRank 算法的时间复杂度为 $O(\text{iter\_num}(n+m)\log n)$, 其中 iter_num 为最大迭代次数.

2.3 标准粒子群算法

粒子群优化算法是由 Kennedy 和 Eberhart 通过对鸟群和鱼群某些行为的观察研究, 提出的一种新颖的进化算法 [24,25]. 由于 PSO 设置参数少, 容易实现, 已广泛应用于求解数据交换、链路调度和图像分割 [26,27] 等诸多优化问题.

在标准粒子群算法中, xSize 个粒子在 n 维目标搜索空间中寻优, 每个粒子有如下属性: 搜索空间

中当前的位置 X_i, 速度 V_i 和自身经历的最优位置 pbest_i, 则

$$X_i = (X_{i1}, \ldots, X_{id}, \ldots, X_{in}), \tag{5}$$

$$V_i = (V_{i1}, \ldots, V_{id}, \ldots, V_{in}), \tag{6}$$

$$\mathrm{pbest}_i = (\mathrm{pbest}_{i1}, \ldots, \mathrm{pbest}_{id}, \ldots, \mathrm{pbest}_{in}), \tag{7}$$

其中 $1 \leqslant i \leqslant \mathrm{xSize}$, $1 \leqslant d \leqslant n$. $\mathrm{gbest} = (\mathrm{gbest}_1, \mathrm{gbest}_2, \ldots, \mathrm{gbest}_n)$ 是当前群体发现的全局最优位置 [28], 则粒子群算法的进化方程为

$$V_{id}^{k+1} = wV_{id}^k + c_1 r_1(\mathrm{pbest}_{id}^k - X_{id}^k) + c_2 r_2(\mathrm{gbest}_d^k - X_{id}^k), \tag{8}$$

$$X_{id}^{k+1} = X_{id}^k + V_{id}^{k+1}, \tag{9}$$

其中 w 是用于平衡局部搜索和全局搜索的惯性权值, c_1 和 c_2 是学习因子, r_1 和 r_2 是在区间 $[0,1]$ 内均匀分布的随机数, k 为迭代次数. 式 (8) 由 3 部分组成, 其中第 1 部分为粒子当前的速度; 第 2 部分为认知部分, 表示粒子本身的思考; 第 3 部分为社会影响, 表示粒子间的社会信息共享 [29].

3 从众和声粒子群算法

标准粒子群算法有很多优势, 但在求解复杂的多维优化问题时, 粒子群算法存在早熟收敛和容易陷入局部最优的问题 [30]. 为了寻找全局最优解, 本文从搜索的多样性和集中性两方面考虑, 提出从众和声粒子群算法. 通过模拟粒子的从众行为, 增加寻求最优解时的多样性; 通过 HS 算法在当前全局最优解附近寻找更优的位置, 增强搜索的集中性.

3.1 从众粒子群算法

生物学家发现社会关系对于动物应对周围的环境十分重要, 动物在寻找食物时, 群体间会进行信息的交流 [31]. 按生物行为学理论, 个体粒子具有从众行为, 趋向于靠近群体粒子聚集的安全地点, 以避免被捕食者袭击 [32]. 安全地点有很多个体并且很难被捕食者袭击, 通过粒子的从众行为, 群体能保持在一起, 共同应对捕食者的袭击. 在从众粒子群算法中, 本文模拟这种行为.

定义1 (安全地点) 在粒子群中安全地点是有很多粒子, 并且很难被捕食者袭击的地方.

为了不失一般性, 在从众粒子群算法中, 本文给出如下的定义, 如图 1 所示.

定义2 (安全范围) 在粒子群中, 粒子数目占群体数目的百分比大于 μ 的区域为安全范围. 安全范围定义为 SR, 如图 1 所示, 三角形是粒子, 以 C_j 为中心的圆圈内的区域是安全范围 SR.

定义3 (安全位置) 安全位置是安全范围 SR 的中心点, 定义为 $\mathrm{SP} = (\mathrm{SP}_1, \mathrm{SP}_2, \ldots, \mathrm{SP}_n)$, 如图 1 所示, C_j 是 SP.

定义4 (最大安全距离) 安全范围的最大安全距离是 SR 的半径, 定义为 MSD, 如图 1 所示, 以 C_j 为起点的实线是 MSD.

若 X_i 在安全范围 SR 内, 则

$$||X_i - \mathrm{SP}|| \leqslant \mathrm{MSD}. \tag{10}$$

安全位置 SP 的计算步骤如下:

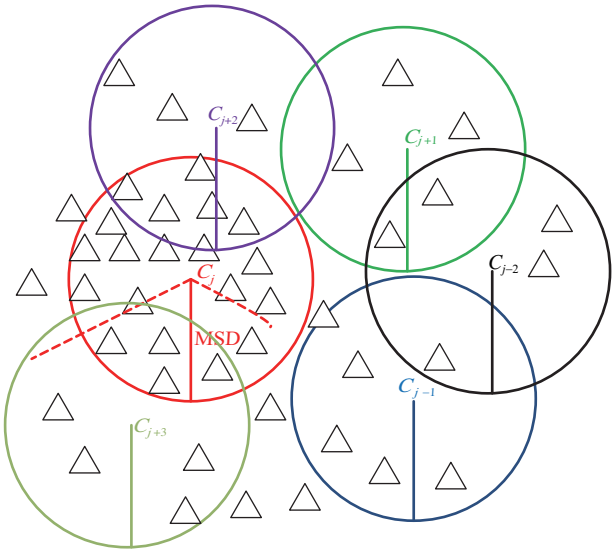

图 1 (网络版彩图) 群体中的安全位置

Figure 1 (Color online) The security positions in the population

步骤 1. 初始化一个位置, 将该位置设置为安全位置 SP.

步骤 2. 通过式 (10) 计算在安全范围 SR 内的粒子数目, 该数目定义为 num.

步骤 3. 如果 num/xSize $> \mu$, 则初始位置为安全位置 SP; 否则, 转步骤 1.

为了减少算法的计算量, 本文将群体中所有粒子位置的平均值作为安全位置, 则

$$\mathrm{SP}_d = \frac{\sum_{i=1}^{\mathrm{xSize}} X_{id}}{\mathrm{xSize}}. \tag{11}$$

当前全局最优位置和上一次全局最优位置几乎相同时, 算法可能陷入了局部最优. 此时在从众粒子群算法中, 每个粒子向当前全局最优的位置、自身经历的最优位置和安全位置靠近, 所以粒子的速度和位置更新的方程如下:

$$V_{id}^{k+1} = wV_{id}^k + c_1r_1(\mathrm{pbest}_{id}^k - X_{id}^k) + c_2r_2(\mathrm{gbest}_d^k - X_{id}^k) + c_3r_3(\mathrm{SP}_d^k - X_{id}^k), \tag{12}$$

$$X_{id}^{k+1} = X_{id}^k + V_{id}^{k+1}, \tag{13}$$

其中 c_3 为从众因子, r_3 为在区间 [0, 1] 内均匀分布的随机数. 式 (12) 中的第 4 部分为从众部分, 它能对粒子群的分布进行扰动, 使群体变得更加多样化. 因此, 从众粒子群算法能拓宽搜索区域和改变搜索方向, 以避免早熟收敛和陷入局部最优, 有利于逼近全局最优点.

3.2 从众粒子群算法的收敛性分析

本文对从众粒子群算法的收敛性进行分析, 由式 (12) 可知, 速度方程可改写为

$$V_i^{k+1} = wV_i^k + c_1r_1\mathrm{pbest}_i^k + c_2r_2\mathrm{gbest}^k + c_3r_3\mathrm{SP}^k - (c_1r_1 + c_2r_2 + c_3r_3)X_i^k. \tag{14}$$

当 $k \to \infty$ 时, V_i^k=0 且 V_i^{k+1}=0, 由式 (14) 可得

$$\lim_{k\to\infty} X_i = \frac{c_1r_1\mathrm{pbest} + c_2r_2\mathrm{gbest} + c_3r_3\mathrm{SP}}{c_1r_1 + c_2r_2 + c_3r_3}. \tag{15}$$

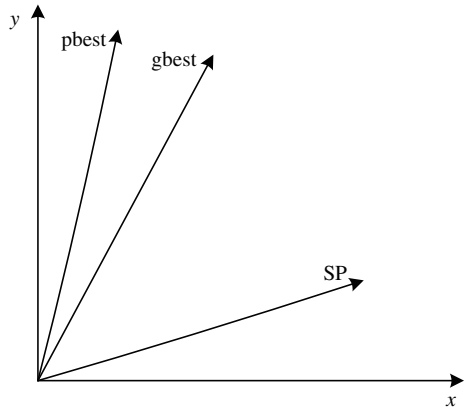

图 2 CPSO 中粒子的搜索方向

Figure 2 The search directions of particles in CPSO

由于 r_1, r_2, r_3 服从均匀分布, 求 X_i 的期望值可得

$$\lim_{k\to\infty} \mathrm{E}(X_i) = \mathrm{E}\left(\frac{c_1 r_1 \mathrm{pbest} + c_2 r_2 \mathrm{gbest} + c_3 r_3 \mathrm{SP}}{c_1 r_1 + c_2 r_2 + c_3 r_3}\right) = \frac{c_1 \mathrm{pbest} + c_2 \mathrm{gbest} + c_3 \mathrm{SP}}{c_1 + c_2 + c_3}$$
$$= (1-\alpha-\beta)\mathrm{pbest} + \alpha \mathrm{gbest} + \beta \mathrm{SP}. \tag{16}$$

式 (16) 中, $\alpha = \frac{c_2}{c_1+c_2+c_3}, \beta = \frac{c_3}{c_1+c_2+c_3}$. 同理可得, 标准粒子群算法中 X_i 的期望为

$$\lim_{k\to\infty} \mathrm{E}(X_i) = \mathrm{E}\left(\frac{c_1 r_1 \mathrm{pbest} + c_2 r_2 \mathrm{gbest}}{c_1 r_1 + c_2 r_2}\right) = \frac{c_1 \mathrm{pbest} + c_2 \mathrm{gbest}}{c_1 + c_2} = (1-\delta)\mathrm{pbest} + \delta \mathrm{gbest}. \tag{17}$$

式 (17) 中, $\delta = \frac{c_2}{c_1+c_2}$. 由式 (17) 可知, PSO 在优化前期种群的多样性好、收敛速度快. 但在优化后期粒子逐渐从 pbest 向 gbest 聚集, 种群的多样性下降, 算法容易陷入局部最优 [33]. 由式 (16) 和 (17) 可知, CPSO 和 PSO 的收敛值不一样, 当 PSO 陷入局部最优时, 通过 CPSO 能避免早熟收敛, 有效地跳出局部最优位置, 如图 2 所示.

由图 2 可知, 当 PSO 陷入局部最优时, SP 和 pbest, gbest 的位置差别较大, 通过 CPSO 更新速度和位置, 将使群体向安全位置附近具有启发信息的方向寻优, 所以 CPSO 能增加种群的多样性, 避免陷入局部最优, 提高全局搜索能力.

3.3 和声搜索算法更新全局最优位置

和声搜索算法是由 Geem 等提出来的一种新的元启发式算法, 该算法模拟音乐创作中乐师们凭借自己的记忆, 通过反复调整乐队中各乐器的音调, 最终达到一个美妙和声状态的过程 [34,35]. 与传统的优化算法相比较, 和声搜索算法概念简单、可调参数少和容易实现. 因此, 和声搜索算法已成功应用于旅行商路径寻优、管道系统设计和函数优化等问题 [36].

由于和声搜索算法计算简单, 在计算量增加不大的情况下, 能增强粒子群算法搜索的集中性, 在当前全局最优解附近提高算法的搜索精度, 有利于提升解的质量. 因此, 本文通过改进 HS 算法的初始化策略, 将 HS 集成到 CPSO-HS 中. 和声搜索算法具体的计算步骤如下.

步骤 1. 初始化和声搜索的基本参数. 这些参数包括: 和声记忆库大小 (HMS), 和声记忆库保留概率 (HMCR), 基音调整概率 (PAR) 和最大迭代次数等.

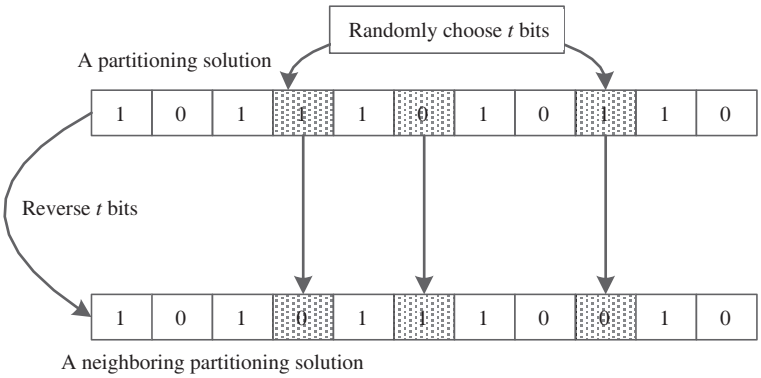

图 3 扰动后的软硬件划分解

Figure 3 The disturbed HW/SW partitioning solution

步骤 2. 初始化和声记忆库. 和声记忆库由随机产生的解向量组成.

步骤 3. 从和声记忆库产生新解. 新解通过以下 3 种机理产生: (1) 保留和声记忆库中的分量; (2) 随机选择音调; (3) 音调微调. 对解的各个分量分别以概率 HMCR 在和声记忆库内搜索, 以概率 1 – HMCR 在记忆库外搜索. 当在记忆库内进行搜索时, 对随机搜索产生的某一分量以概率 PAR 进行扰动 [37].

步骤 4. 更新和声记忆库. 若新解优于记忆库中的最差解, 则用新解替换最差解, 得到新的记忆库.

步骤 5. 判断是否满足终止条件. 若满足, 则停止迭代, 输出和声记忆库中的最优解; 否则, 重复步骤 3 和 4.

在步骤 3 中, 经过扰动后的软硬件划分解通过二进制取反得到 [13], 如图 3 所示.

由图 3 可知, 软硬件划分解 $\boldsymbol{x}$ 取反的过程是将其 t 个分量 x_i=1 (x_i=0) 转换为 x_i=0 (x_i=1), 所以经过扰动后的软硬件划分解由 $\boldsymbol{x}$ 中 t 个二进制取反得到. 为了提升解的质量, 对和声搜索算法的初始化策略进行改进, HS 的初始解向量由粒子自身经历的最优位置 pbest 组成.

通过改进和声搜索算法的初始化策略, 既能保证群体的多样性, 又能在当前全局最优解的附近搜索更优值. 在 CPSO-HS 中, 每次迭代时通过 HS 搜索更优的位置, 若搜索的位置优于当前全局最优位置, 则用该位置替代当前全局最优位置. 因此, HS 能在当前全局最优解的邻域进一步充分地搜索, 以提高算法的搜索精度, 有利于提升当前全局最优解的质量.

4 基于从众和声粒子群算法的并行软硬件划分方法

4.1 基于从众和声粒子群算法的硬件划分方法

本文中软硬件划分问题被定义为一类扩展的 0/1 背包问题, 所以软硬件划分方法采用离散的 CPSO-HS. 在离散 CPSO-HS 中, 每次迭代时粒子的位置按如下公式进行离散化:

$$X_{id}^k = \begin{cases} 1, & \text{if } r_4 < \text{sig}(V_{id}^k), \\ 0, & \text{else}, \end{cases} \tag{18}$$

其中 $\text{sig}(V_{id}^k) = \frac{1}{1+\exp(-V_{id}^k)}$, r_4 为在区间 [0,1] 内均匀分布的随机数. 当 r_4 的值小于 $\text{sig}(V_{id}^k)$ 时, X_{id}^k 等于 1; 否则, X_{id}^k 等于 0. 离散的 CPSO-HS 用于优化式 (1) 中的问题 P, 本文将基于 CPSO-HS 的

算法 1 CPSO-HS

Require: Communication graph G and the constraint R.

Ensure: The global best partition solution gbest and its hardware cost fgbest.

1: X_nr:= the partition solution abstained by NodeRank;
2: $X = (X\_nr, X\_nr, \ldots, X\_nr)$, $V = \text{rand}(\text{xSize}, n)$; /*$X$ and V are matrixes with xSize columns and n rows. */
3: $\text{pbest} = (X\_nr, X\_nr, \ldots, X\_nr), \text{fpbest} = (0, 0, \ldots, 0), \text{gbest} = (0, 0, \ldots, 0), \text{fgbest} = 0, \text{fgbestold} = 1, \text{iter} = 0$;
4: Initialize other parameters, such as w, c_1, c_2, c_3 and epsx, the individual number xSize and the maximum iteration time iter_num;
5: **repeat**
6: iter:= iter+1;
7: X is discretized by (18);
8: $sx = S(X), hx = H(X)$; /*compute the software and hardware cost*/
9: **for** $i = 1$ to xSize **do**
10: **if** $sx(i) + C(X_i) > R$ **then**
11: $X_i = X\_nr, hx(i) = H(X_i)$; /*$X_i$ is replaced by X_nr.*/
12: **end if**
13: **end for**
14: Update pbest, gbest, fpbest, fgbest and w;
15: **for** $j = 1$ to n **do**
16: The security point SP is computed according to Eq. (11);
17: **end for**
18: **if** |fgbestold − fgbest| > epsx **then**
19: The velocity and position of particle are updated by Eqs. (8) and (9), respectively;
20: **else**
21: The velocity and position of particle are updated by Eqs. (12) and (13), respectively;
22: **end if**
23: gbest = Adjust_In(gbest), fgbest =H(gbest);
24: The global best position gbest is updated by HS;
25: fgbestold = fgbest;
26: **until** (iter > iter_num)

软硬件划分方法命名为 CPSO-HS. 将软硬件划分问题作为扩展的 0/1 背包问题进行求解, 目前算法 NodeRank 的性能最佳, 所以本文采用算法 NodeRank 对 CPSO-HS 的种群进行初始化. 为了减少计算量, 当两次迭代的全局最优位置几乎相同时, 本文判断粒子群算法陷入局部最优, 则算法 CPSO-HS 的描述如算法 1 所示.

在 CPSO-HS 中, 第 2 行的解向量由算法 NodeRank 获取的解进行初始化. xSize 是种群中粒子的数目, n 是任务结点的数目. 在第 3 行中, fpbest 是 pbest 的适应度, fgbest 是 gbest 的适应度. 在第 18~22 行中, 当前全局最优位置和上一次全局最优位置几乎相同时, 算法可能陷入了局部最优, 此时每个粒子的速度和位置按式 (12) 和 (13) 进行更新; 否则, 速度和位置按式 (8) 和 (9) 进行更新. 在第 23 行中, 文献 [13] 中的算法 Adjust_In 用于搜索更优的值. 在第 24 行中, 当前全局最优解通过 HS 更新, 算法 HS 的描述如算法 2 所示.

在 CPSO-HS 中, 第 10 行软硬件通讯代价 $C(\boldsymbol{x})$ 的计算过程如下:

$$C(\boldsymbol{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}|x_i - x_j|, \tag{19}$$

其中 c_{ij} 代表结点 v_i 和 v_j 之间的通讯代价.

算法 2 HS

Require: The personal best position pbest.

Ensure: The updated global best position gbest.

```
Initialize HMCR, PAR, iter_num, k=0;
repeat
  if rand < HMCR then
    x_old is a position chosen from pbest randomly;
    if rand < PAR then
      x_new is the disturbed solution of x_old;
    end if
  else
    x_new is generated randomly;
  end if
  if x_new is a feasible solution of Q and x_new is better than gbest then
    gbest is replaced by x_new;
  end if
  k=k+1;
until k > iter_num
```

定理1 软硬件通讯代价 $C(\boldsymbol{x})$ 的时间复杂度为 $O(n^2)$, 其中 n 为任务结点的数目.

证明 在式 (19) 中, 有两层循环计算, 每一层循环执行的时间为 $O(n)$. 所以, 软硬件通讯代价 $C(\boldsymbol{x})$ 的时间复杂度为 $O(n^2)$.

定理2 算法 CPSO-HS 的时间复杂度为 $O(\text{iter\_num}\cdot\text{xSize}\cdot n^2)$, 其中 iter_num 为最大迭代次数, xSize 为种群中粒子的数目, n 为任务结点的数目.

证明 在 CPSO-HS 中, 第 1 行算法 NodeRank 的时间复杂度为 $O(\text{iter\_num}(n+m)\log n)$ [21]. 在第 9∼13 行中, 当解不满足约束条件时, 由算法 NodeRank 获取的初始解进行替换, 所需的时间复杂度为 $O(\text{xSize}\cdot n^2)$. 在第 14 行中, 更新 pbest, gbest, fpbest 和 fgbest 的时间复杂度为 $O(\text{xSize}\cdot n)$. 在第 23 行中, Adjust_In 能在 $O(\log n(n+m))$ [13] 内完成. HS 是一种简单的优化算法, 所以我们可以将 HS 的时间复杂度设置为低于 $O(\text{xSize}\cdot n^2)$. 相比较 $\log n(n+m)$, $\text{xSize}\cdot n^2$ 的值更大. CPSO-HS 的最大迭代次数为 iter_num, 所以算法 CPSO-HS 的时间复杂度为 $O(\text{iter\_num}\cdot\text{xSize}\cdot n^2)$.

本文将基于标准粒子群算法的软硬件划分方法命名为 PSO, PSO 和 CPSO-HS 的计算步骤基本一样. 但在 PSO 中, 每个粒子的速度和位置按式 (8) 和 (9) 进行更新, 当前全局最优的位置不通过 HS 进行更新.

定理3 算法 CPSO-HS 的时间复杂度和 PSO 的时间复杂度相同.

证明 在 CPSO-HS 中, 第 19 行和 21 行分别是两种不同情况下更新粒子速度和位置的过程, 两者的时间复杂度相同. 在第 9∼13 行中, 计算通讯代价的时间复杂度为 $O(\text{xSize}\cdot n^2)$, HS 更新当前全局最优位置的时间复杂度低于 $O(\text{xSize}\cdot n^2)$. 所以, 算法 CPSO-HS 的时间复杂度和 PSO 的时间复杂度相同.

定理4 算法 CPSO-HS 的空间复杂度为 $O(\text{xSize}\cdot n)$, 和 PSO 的空间复杂度相同, 其中 xSize 为种群中粒子的数目, n 为任务结点的数目.

证明 在 CPSO-HS 中, 每个粒子向当前全局最优的位置、自身经历的最优位置和安全位置靠近. 当前全局最优位置和安全位置的空间复杂度都为 $O(n)$, 每个粒子自身经历最优位置的空间复杂度为 $O(\text{xSize}\cdot n)$. 所以, 算法 CPSO-HS 的空间复杂度为 $O(\text{xSize}\cdot n)$. 在 PSO 中, 每个粒子向当前全局最

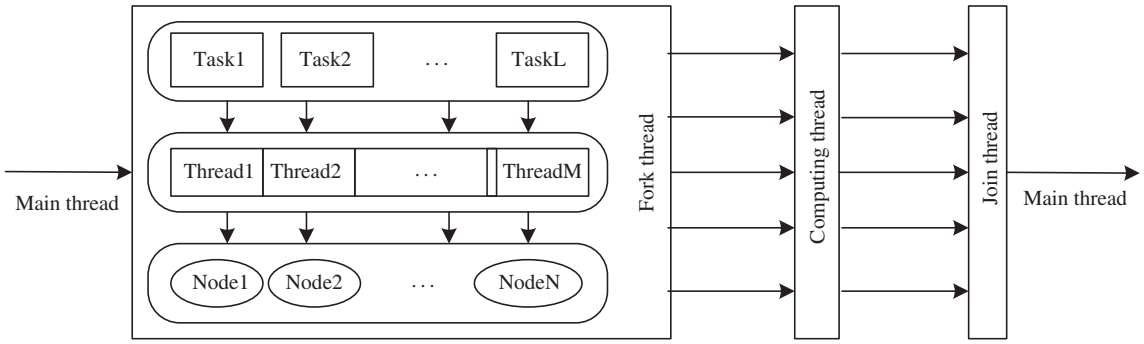

图 4　多核并行计算通讯代价的机制

Figure 4　The multi-core parallel mechanism for communication cost computing

优的位置和自身经历的最优位置靠近, 所以算法 PSO 的空间复杂度为 $O(\text{xSize}\cdot n)$, 和 CPSO-HS 的空间复杂度相同.

4.2　多核并行加速 CPSO-HS

并行计算的思想如下: 将一个复杂的问题划分为一个子问题集合, 这些子问题能同时在多个独立的计算节点上执行. 在 CPSO-HS 中, 软硬件通讯代价 $C(\boldsymbol{x})$ 的时间复杂度为 $O(n^2)$, 每次迭代都考虑了软硬件通讯代价, 因此通讯代价的计算量非常大. 为了减少 CPSO-HS 的整体运行时间, 本文采用多核并行对软硬件通讯代价 $C(\boldsymbol{x})$ 的计算过程进行加速, 多核并行的任务是计算 $C(\boldsymbol{x})$, 计算节点是处理器中的多核, 并行计算 $C(\boldsymbol{x})$ 的机制如图 4 所示 [38].

由图 4 可知, 计算通讯代价的任务由线程执行, 多核并行将执行任务的线程分配给计算节点, 每个节点并行循环地处理任务 [39], 然后将所有线程的计算结果进行合并, 得到通讯代价最终的计算结果. 在 CPSO-HS 中, xSize 个粒子的软硬件通讯代价计算过程在第 9∼13 行, 该过程采用多核并行计算, 从而能有效地减少 CPSO-HS 的运行时间.

5　实验研究及性能分析

为了验证本文软硬件划分方法的性能, 将该方法用于求解软硬件划分的基准任务. 本文算法的开发工具为 MATLAB R2014b, 计算机处理器的型号为 Intel(R) Core(TM) i7-4770, 主频为 3.4 GHz, 内存大小为 16 GB.

本文采用文献 [11] 中的参数设置, 软件代价 s_i 在 [1,100] 中随机产生并服从均匀分布, 硬件代价 h_i 由数学期望值为 ks_i 的正态分布随机产生. 通讯代价为区间 $[0, 2\rho s_{\max}]$ 中的随机数, 其中 $s_{\max}$ 为软件代价的最大值. 根据通信代价与软件计算代价之比 (CCR) 定义了 3 种不同类型的任务图: 计算密集型、中间型和通信密集型, 对应 CCR 取值分别为 0.1, 1, 10. 考虑两种不同约束条件 R: 强实时约束 R 在 $[0, 0.5\sum s_i]$ 中随机产生, 记为 R=low; 弱实时约束 R 在 $[0.5\sum s_i, \sum s_i]$ 随机产生, 记为 R=high. 所以, 对于不同值的 CCR 和 R 有 6 种不同的情况, 如表 1 所示. 表 2 为文献 [11] 中使用的全部基准任务, n 和 m 分别为任务结点和边的数目.

在 CPSO-HS 中, 作为重要的调节参数, 学习因子 c_1, c_2 和从众因子 c_3 对算法的性能有很大的影响. 从表 2 中选择典型的任务 random2, 本文通过 c_1, c_2 和 c_3 取不同值时, CPSO-HS 求解任务

表 1 CCR 和 R 取不同值的情况

Table 1 The cases for different values of CCR and R

Case	CCR	R
case1	0.1	Low
case2	0.1	High
case3	1	Low
case4	1	High
case5	10	Low
case6	10	High

表 2 文献 [11] 采用的基准任务

Table 2 Summary of used benchmarks, cited from [11]

Name	n	m	Size	Number
crc32	25	34	152	1
patricia	21	50	192	2
dijkstra	26	71	265	3
clustering	150	333	1299	4
rc6	329	448	2002	5
random1	1000	1000	5000	6
random2	1000	2000	8000	8
random3	1000	3000	11000	10
random4	1500	1500	7500	7
random5	1500	3000	12000	11
random6	1500	45000	16500	13
random7	2000	2000	10000	9
random8	2000	4000	16000	12
random9	2000	6000	22000	14

random2 的比较实验, 确定 CPSO-HS 中学习因子和从众因子合适的取值. 为了平衡全局搜索和局部搜索, c_1, c_2 和 c_3 在区间 [0.5, 2.5] 内取值. 每个参数以 0.25 为间隔从 0.5 到 2.5 取不同的值, 当一个参数变化时, 其他参数的值保持不变 [40]. 由于篇幅的限制, 本文仅列出当 c_1=1.5, c_2=1.5, c_3 取不同值时, 不同情况下通过 CPSO-HS 求解任务 random2 的结果, 如图 5 所示, 横轴代表 c_3 的不同取值, 纵轴代表硬件代价.

由图 5 可知, 当 c_3=1 时, 6 种不同情况下 CPSO-HS 获取的平均硬件代价最小. 综合所有实验的结果, 当学习因子 c_1, c_2 和从众因子 c_3 取不同值时, 算法 CPSO-HS 得到的解相差较大, 当 CPSO-HS 获取最优解时, 学习因子和从众因子的取值如下: c_1=1.5, c_2=1.5, c_3=1.

算法 Base, Heur, NodeRank, GA, PSO 和 CPSO-HS 求解表 2 中的基准任务. 在 CPSO-HS 和 PSO 中, 惯性因子 (w) 随着迭代次数在 0.9 和 0.7 之间线性减少, 种群数目为 40, 最大迭代次数为 50. CPSO-HS 中其他参数取值如下: c_1=1.5, c_2=1.5, c_3=1, epsx=1, HMCR=0.9, PAR=0.3, HS 的最大迭代次数为 30 [34]. PSO 中的学习因子 c_1 和 c_2 都设置为 2, 算法 NodeRank 的参数设置和文献 [11] 中的相同. 在 GA 中, 杂交概率为 0.9, 变异概率为 0.1, 种群数目为 40, 最大迭代次数为 50. 本文从解的

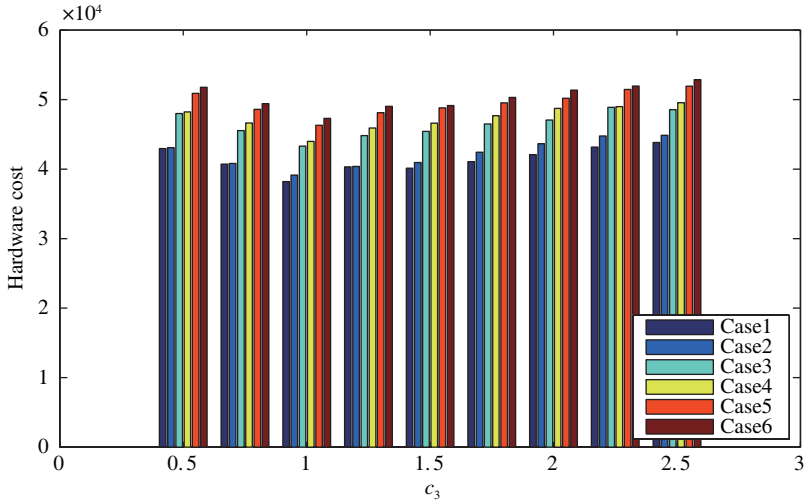

图 5 (网络版彩图) c_1=1.5, c_2=1.5, c_3 取不同值时, 不同情况下 CPSO-HS 求解任务 random2 的结果

Figure 5 (Color online) The solutions for random2 obtained by CPSO-HS on different cases, when c_1=1.5, c_2=1.5 and c_3 with different values

质量和运行时间两方面比较算法求解软硬件划分问题的性能, 为了不失一般性, 在实验中选择 100 个随机的实例进行统计分析.

5.1 解的质量

为了比较算法解的质量, 利用下式计算算法 A 对算法 B 的改进程度:

$$\text{imp}=\left(1-\frac{\text{hardware\_cost\_of\_A}}{\text{hardware\_cost\_of\_B}}\right)\times 100\%. \tag{20}$$

在式 (20) 中, imp > 0, imp = 0 和 imp < 0 分别表示算法 A 优于、同等于和差于算法 B. 图 6 为在不同情况下不同算法 100 次实例相对算法 Base 的平均改进程度, 横轴为任务规模的大小, 纵轴为算法相对于 Base 的改进程度.

由图 6 可知, 总体而言, CPSO-HS 得到的解优于或者相似于其他算法得到的解. 由于 GA 和 CPSO-HS 在优化机理上差别较大, 在极少数情况下, GA 得到的解优于 CPSO-HS, 此时软硬件划分问题更加适合采用 GA 进行求解. 对于通讯代价小的情况, 如图 6(a) 所示, 算法 NodeRank, GA, PSO 和 CPSO-HS 得到的解几乎相同. 这是因为较小的通讯代价对于软硬件划分问题没有太大的约束影响, 最优值也能通过其他算法得到. 但是随着通讯代价的增加, 软硬件划分问题和标准的背包问题很不一样, 求解软硬件划分优化问题的难度增加. 由于 CPSO-HS 的全局搜索能力更强, CPSO-HS 的改进程度更加明显, 如图 6(f) 所示.

5.2 运行时间

加速比是并行计算中重要的性能指标, 本文采用加速比 S_p 衡量多核并行计算通讯代价的加速效果:

$$S_p=\left(\frac{T_{\text{seq}}}{T_{\text{par}}}\right)\times 100\%, \tag{21}$$

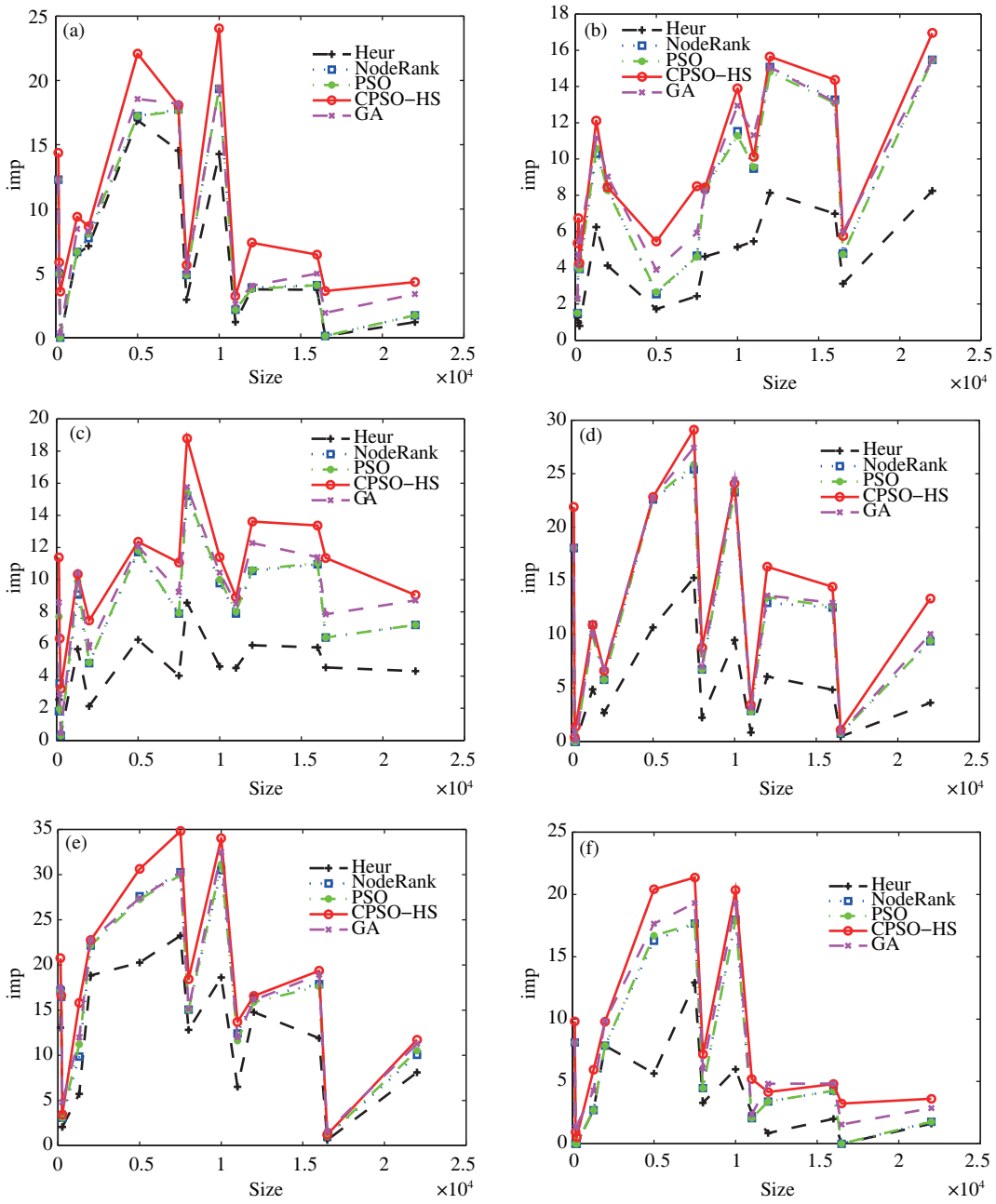

图 6 (网络版彩图) 100 次实例、不同情况下, 不同算法相对算法 Base 的平均改进程度

Figure 6 (Color online) Improvements of different algorithms over Base, averaged over 100 instances on different cases of (a) case1, (b) case2, (c) case3, (d) case4, (e) case5, and (f) case6

其中 T_{seq} 为串行环境中计算通讯代价的时间, T_{par} 为并行环境中计算通讯代价的时间. 针对不同的基准任务, 在不同情况下随机选择 100 个软硬件划分的解, 计算通讯代价的平均加速比, 如图 7 所示, 横轴代表任务的序号, 纵轴代表 100 个软硬件通讯代价的平均加速比.

由图 7 可知, 总体而言, 随着任务规模和通讯代价增加, 加速比的值增大, 多核并行加速的效果更加明显. 当任务规模小于等于 2002 时, 加速比小于 1; 当任务规模大于 2002 时, 加速比大于 1. 这是

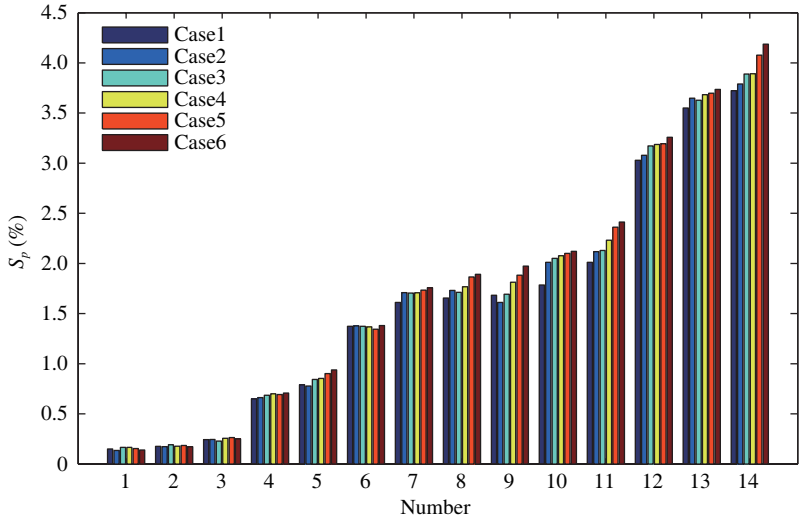

图 7 (网络版彩图) 不同情况下随机选择 100 个软硬件通讯代价的平均加速比

Figure 7 (Color online) The speedup of computing HW/SW communication cost, averaged over 100 instances on different cases

因为多核并行计算需要进行创建线程、创建进程、创建任务和任务间数据通讯等操作, 这些操作都会额外地耗费时间. 当任务规模较小时, 额外耗费的时间占据了通讯代价计算的大部分时间, 并行计算的优势不明显, 所以多核并行环境中计算通讯代价的时间长于串行环境; 当任务规模较大时, 通讯代价计算量大, 额外耗费的时间相比很小, 并行计算的优势明显, 所以多核并行环境中计算通讯代价的时间短于串行环境.

在 CPSO-HS 中通讯代价的计算过程在多核并行环境中运行, 其他算法中通讯代价的计算过程在串行环境中运行. 图 8 展示了不同情况下不同算法 100 次实例的平均运行时间, 横轴代表任务的序号, 纵轴代表不同算法的平均运行时间.

由图 8 可知, 随着基准任务规模的增大, 多核并行计算的优势变得更加明显. 由于遗传算法存在编解码、选择、杂交和变异等复杂运算, GA 的平均运行时间远长于其他算法的时间. 当任务规模较大时, 尽管 CPSO-HS 的计算步骤多于 PSO, 由于采用多核并行计算软硬件通讯代价, CPSO-HS 的整体运行时间短于 PSO 的时间. 当任务的规模小于 16000 时, CPSO-HS 的计算量大于 PSO, 但多核并行计算的优势不明显, 此时 CPSO-HS 的运行时间长于 PSO 的时间. 当任务规模大于等于 16000 时, 通讯代价计算量大, 采用多核并行的优势明显, 此时 CPSO-HS 的运行时间短于 PSO 的时间. 因此, 多核并行的 CPSO-HS 对于求解大规模的软硬件划分问题是一种非常有效的方法.

6 总结

本文提出的从众和声粒子群算法模拟粒子的从众行为, 个体粒子趋向于靠近群体粒子聚集的安全地点, 从而保持了搜索种群的多样性, 避免了早熟收敛和陷入局部最优. 通过改进和声搜索算法的初始化策略, 将 HS 集成到 CPSO-HS 中, 在当前全局最优解附近提高了算法的搜索精度, 提升了解的质量. 通过低成本的并行策略加速软硬件通讯代价的计算过程, 从而有效地减少了整体运行时间.

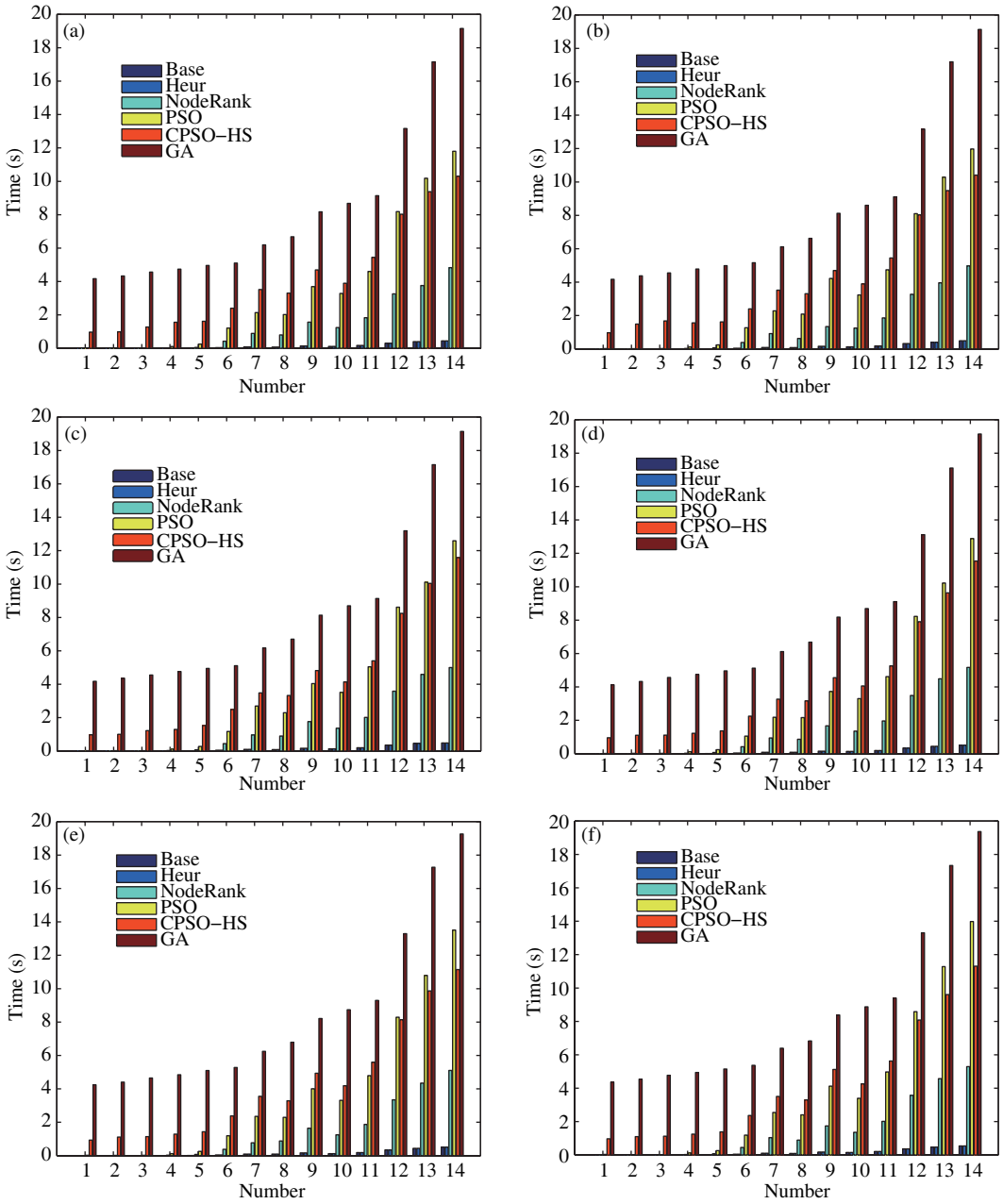

图 8 (网络版彩图) 100 次实例不同情况下, 不同算法的平均运行时间

Figure 8 (Color online) Runtime of different algorithms, averaged over 100 random instances on different cases of (a) case1, (b) case2, (c) case3, (d) case4, (e) case5, and (f) case6

在今后的工作中, 本文将引入其他改进粒子群算法的相关策略 [41, 42], 持续对 CPSO-HS 算法的性能进行改进. 考虑到优化算法已被广泛地应用于 CAD 与图形图像处理中 [43~45], CPSO-HS 算法也将扩展应用到这些领域.

参考文献

1 Zhang Y G, Luo W J, Zhang Z M, et al. A hardware/software partitioning algorithm based on artificial immune principles. Appl Soft Comput, 2008, 8: 383–391

2 Arato P, Mann Z A, Orban A. Algorithmic aspects of hardware/software partitioning. ACM Trans Des Automat El, 2005, 10: 136–156

3 Wu J G, Srikanthan T. Low-complex dynamic programming algorithm for hardware/software partitioning. Inform Process Lett, 2006, 98: 41–46

4 Madsen J. Lycos: the lyngby cosynthesis system. Des Autom Embed Syst, 1997, 2: 195–235

5 Chatha K S, Vemuri R. Hardware-software partitioning and pipelined scheduling of transformative applications. IEEE Trans VLSI Syst, 2002, 10: 193–208

6 Niemann R, Marwedel P. An algorithm for hardware/software partitioning using mixed integer linear programming. Des Autom Embed Syst, 1997, 2: 165–193

7 Wiangtong T, Cheung P Y K, Luk W. Comparing three heuristic search methods for functional partitioning in hardware-software codesign. Des Autom Embed Syst, 2002, 6: 425–449

8 Janakiraman N, Kumar P N. Multi-objective module partitioning design for dynamic and partial reconfigurable system-on-chip using genetic algorithm. J Syst Architect, 2014, 60: 119–139

9 Henkel J, Ernst R. An approach to automated hardware/software partitioning using a flexible granularity that is driven by high-level estimation techniques. IEEE Trans VLSI Syst, 2001, 9: 273–290

10 Peng L, Wu J G, Wang Y J. Hybrid algorithms for hardware/software partitioning and scheduling on reconfigurable devices. Math Comput Model, 2013, 58: 409–420

11 Wu J G, Srikanthan T, Chen G. Algorithmic aspects of hardware/software partitioning: 1D search algorithms. IEEE Trans Comput, 2010, 59: 532–544

12 Eles P, Peng Z, Kuchcinski K, et al. System level hardware/software partitioning based on simulated annealing and tabu search. Des Autom Embed Syst, 1997, 2: 5–32

13 Wu J G, Pu W, Lam S K, et al. Efficient heuristic and tabu search for hardware/software partitioning. J Supercomput, 2013, 66: 118–134

14 Xiong Z H, Li S K, Chen J H. Hardware/software partitioning based on ant optimization with initial pheromone. Comput Res Dev, 2005, 42: 2176–2183 [熊志辉, 李思昆, 陈吉华. 具有初始信息素的蚂蚁寻优软硬件划分算法. 计算机研究与发展, 2005, 42: 2176–2183]

15 Badawy W, Salem A. Hardware software partitioning using particle swarm optimization technique. In: Proceedings of the 6th International Workshop on System-on-Chip for Real-Time Applications. Alberta: IEEE Press, 2006. 189–194

16 Abdelhalim M B, Habib S E D. An integrated high-level hardware/software partitioning methodology. Des Autom Embed Syst, 2011, 15: 19–50

17 Lopez V M, Lopez J C. On the hardware-software partitioning problem: system modeling and partitioning techniques. ACM Trans Des Automat El, 2003, 8: 269–297

18 Bhattacharya A, Konar A, Das S, et al. Hardware software partitioning problem in embedded system design using particle swarm optimization algorithm. In: Proceedings of the 2nd International Conference on Complex, Intelligent and Software Intensive Systems. Barcelona: IEEE Press, 2008. 171–176

19 Eimuri T, Salehi S. Using DPSO and B&B algorithms for hardware/software partitioning in co-design. In: Proceedings of the 2nd International Conference on Computer Research and Development. Kuala Lumpur: IEEE Press, 2010. 416–420

20 Wu Y, Zhang H, Yang H B. Research on parallel HW/SW partitioning based on hybrid PSO algorithm. Lect Notes Comput Sci, 2009, 5574: 449–459

21 Chen Z, Wu J G, Song G Z, et al. NodeRank: an efficient algorithm for hardware/software partitioning. Chin J Comput, 2013, 36: 2033–2040 [陈志, 武继刚, 宋国治, 等. NodeRank: 一种高效软硬件划分算法. 计算机学报, 2013, 36: 2033–2040]

22 Farmahini F A, Mehdi K, Mehdi S J. Parallel-genetic-algorithm-based HW/SW partitioning. In: Proceedings of the International Symposium on Parallel Computing in Electrical Engineering. Bialystok: IEEE Press, 2006. 337–342

23 Bordoloi U D, Chakraborty S. GPU-based acceleration of system-level design tasks. Int J Parallel Prog, 2010, 38: 225–253

24 Bergh F, Engelbrecht A P. A cooperative approach to particle swarm optimization. IEEE Trans Evolut Comput, 2004, 8: 225–239

25 Coello C A C, Pulido G T, Lechuga M S. Handling multiple objectives with particle swarm optimization. IEEE Trans Evolut Comput, 2004, 8: 256–279

26 Zhang D J, He F Z, Wu Y Q. Singular feature interoperability of heterogeneous CAD model based on directed mutation particle swarm optimization. Sci Sin Inform, 2015, 45: 634–649 [张德军, 何发智, 吴亦奇. 一种基于定向变异粒子群算法的异构 CAD 模型奇异特征互操作方法. 中国科学: 信息科学, 2015, 45: 634–649]

27 Hei Y Q, Li W T, Li X H. Novel scheduling strategy for downlink multiuser MIMO system: particle swarm optimization. Sci Sin Inform, 2011, 41: 1463–1473 [黑永强, 李文涛, 李晓辉. 基于粒子群优化的多用户 MIMO 下行链路调度算法. 中国科学: 信息科学, 2011, 41: 1463–1473]

28 Liu C A, Yan X H, Liu C Y. Dynamic path planning for mobile robot based on improved genetic algorithm. Chinese J Electron, 2010, 19: 245–248

29 Ratnaweera A, Halgamuge S K, Watson H C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans Evolut Comput, 2004, 8: 240–255

30 Liang J J, Qin A K, Suganthan P N, et al. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evolut Comput, 2006, 10: 281–295

31 Veissier I, Boissy A, Nowak R, et al. Ontogeny of social awareness in domestic herbivores. Appl Anim Behav Sci, 1998, 57: 233–245

32 Herzenstein M, Dholakia U M, Andrews R L. Strategic herding behavior in peer-to-peer loan auctions. J Interact Mark, 2011, 25: 27–36

33 Clerc M, Kennedy J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evolut Comput, 2002, 6: 58–73

34 Geem Z W, Kim J H, Loganathan G V. A new heuristic optimization algorithm: harmony search. Simulation, 2001, 76: 60–68

35 Lee K S, Geem Z W. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput Method Appl M, 2005, 194: 3902–3933

36 Mahdavi M, Fesanghary M, Damangir E. An improved harmony search algorithm for solving optimization problems. Appl Math Comput, 2007, 188: 1567–1579

37 Kaveh A, Talatahari S. Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput Struct, 2009, 87: 267–283

38 Gordon M I, Thies W, Amarasinghe S. Exploiting coarse-grained task, data, and pipeline parallelism in stream programs. In: Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems. California: ACM, 2006. 151–162

39 Gao W J, Qian K M. Parallel computing in experimental mechanics and optical measurement: a review. Opt Laser Eng, 2012, 50: 608–617

40 Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agent. IEEE Syst Man Cybern, 1996, 26: 29–41

41 Yan X H, He F Z, Chen Y L, et al. An efficient improved particle swarm optimization based on prey behavior of fish schooling. J Adv Mech Des Syst, 2015, 9: 1–10

42 Zhan Z H, Zhang J, Li Y, et al. Adaptive particle swarm optimization. IEEE Syst Man Cybern, 2009, 39: 1362–1381

43 He F Z, Han S H. A method and tool for human-human interaction and instant collaboration in cscw-based CAD. Comput Ind, 2006, 57: 740–751

44 Zhang D J, He F Z, Han S H, et al. Quantitative optimization of interoperability during feature-based data exchange. Integr Comput-Aid E, 2016, 23: 31–50

45 Jing S X, He F Z, Han S H, et al. A method for topological entity correspondence in a replicated collaborative CAD system. Comput Ind, 2009, 60: 467–475

A parallel hardware/software partitioning method based on conformity particle-swarm optimization with harmony search

Xiaohu YAN[1,2,3], Fazhi HE[1,2*] & Yilin CHEN[1,2]

1 *State Key Laboratory of Software Engineering, Wuhan University, Wuhan* 430072*, China*;
2 *School of Computer Science and Technology, Wuhan University, Wuhan* 430072*, China*;
3 *Wuhan NARI Limited Company of State Grid Electric Power Research Institute, Wuhan* 430074*, China*
*E-mail: fzhe@whu.edu.cn

Abstract Hardware/software (HW/SW) partitioning is a key step in HW/SW codesign. With the increasing design complexity of embedded systems, HW/SW partitioning has become a challenging optimization problem. A parallel HW/SW partitioning method based on Conformity Particle-Swarm Optimization with Harmony Search (CPSO-HS) is presented in this paper. Firstly, the particles act as psychological conformists and tend to move towards a security point, with many particles and a lower possibility of being attacked by a predator. By simulating the conformist mentality in CPSO-HS, the searching population can remain varied and the algorithm avoids local optima. Secondly, to improve the initialization strategy, the Harmony Search (HS) is integrated to search for better positions, with which the global best position is updated. Hence, the searching precision and solution quality can be enhanced. The searching diversification and intensification in CPSO-HS can be improved through the two methods. Thirdly, since the time to compute the HW/SW communication cost is the most time-consuming process for the HW/SW partitioning problem, we adopt a multi-core parallel technique to accelerate the computing. Thus, the CPSO-HS runtime for large-scale HW/SW problems can be reduced efficiently in an ordinary PC platform. Finally, a number of experiments on benchmarks from state-of-the-art publications demonstrate that the proposed approach can achieve higher performance than other previous methods.

Keywords hardware/software partitioning, particle swarm optimization, harmony search, knapsack problem, communication cost, parallel computing

Xiaohu YAN was born in 1986. He received his B.S. degree from Huazhong Agricultural University, in 2008, and his M.S. degree from North China Electric Power University, in 2010. Currently, he is a Ph.D. candidate in the School of Computer Science and Technology in Wuhan University. His research interests include hardware/software partitioning and optimization algorithms.

Fazhi HE was born in 1968. He received his B.S., M.S., and Ph.D. degrees from Wuhan University of Technology, China. He was a post-doctor in Zhejiang University, a visiting researcher in the Korea Institute of Science and Technology, and a visiting faculty member in the University of North Carolina at Chapel Hill. Currently, he is a professor in the School of Computer Science and Technology, Wuhan University. His research interests are hardware/software partitioning, computer-aided design, computer graphics, and collaborative computation.