文章编号:1001 - 9081(2010)11 - 2967 - 03

嵌入式系统动态数据结构优化的并行进化算法

王晓升

(山东女子学院 信息技术学院, 济南 250300)

(wxs_infor@163.com)

摘　要:为了更好地解决现代多媒体嵌入式系统动态数据结构优化问题,结合 NSGA-II 和 SPEA2 两个多目标进化算法,引入岛屿模型和多线程机制,提出了一种并行多目标进化算法——PMOEA-NS。基于多核计算机系统,使用 PMOEA-NS 具体的 3 个不同并行算法和串行 NSGA-II、SPEA2,对一个实际动态嵌入式应用程序进行优化实验和计算,结果表明:与串行算法 NSGA-II 和 SPEA2 相比,并行算法不但提高了优化过程的速度,而且改善了解的质量和多样性。

关键词:嵌入式系统;动态数据结构;多目标;优化;并行进化算法

中图分类号: TP301.6;TP311.13　　**文献标志码**:A

Parallel evolutionary algorithm for dynamic data structures optimization in embedded system

WANG Xiao-sheng

(*Information Technology School, Shandong Women's University, Jinan Shandong* 250300, *China*)

Abstract: In order to better solve dynamic data structures optimization in embedded system, this paper combined NSGA-II and SPEA2, and adopted island model and multi-thread technique to describe a parallel multi-objective evolutionary algorithm. Using its specific three parallel algorithms and sequential NSGA-II and SPEA2, one embedded application on multi-core architecture was optimized in experiment. The results show that not only the speed of optimization process is enhanced, but also the quality and the variety of the solutions was improved.

Key words: embedded system; dynamic data structures; multi-objective; optimization; parallel evolutionary algorithm

0　引言

随着多媒体嵌入式设备的需求不断上升,设备所需的处理能力越来越强,内存需求及其动态性不断增加,例如:以前存储并运行在功能强大的台式计算机上的多媒体应用程序(如 3D 游戏、视频播放器等),依靠动态数据结构(如向量、链表等)存放数据,现在已移植到嵌入式系统上运行。但是由于嵌入式系统硬件资源高度精简,使得嵌入式开发者面临如何将来自台式机的大规模应用程序在资源受限的嵌入式系统上集成的问题。解决这个问题的首要任务是动态存储器子系统的优化。因此,嵌入式开发者应根据目标系统的一些实际限制和典型的嵌入式设计度量(如存储器访问、存储器使用和能量消耗[1]),在多个可能被分配的动态数据结构的实现(如动态数组、链表等)之中为应用程序的每个变量选择一种数据结构。显然这是一个多目标优化问题。为了解决这个问题,已开发了一些比较有效的启发式算法[2-3],但是它们并未搞清对应用程序中的每个变量怎样更好地为其选择动态数据结构,因为这是 NP 完全性问题,并且也不可能圆满地搜索到。本文基于 NSGA-II 和 SPEA2 两个多目标进化算法,提出了一种并行多目标进化算法,更好地实现了嵌入式系统中动态数据结构的优化。

1　动态数据结构的搜索及优化流程

嵌入式多媒体应用程序运用动态数据结构存取和处理数据,表 1 是文献[4]针对嵌入式多媒体应用程序变量的优化而开发的一个动态数据结构库(问题的决策变量空间),并结合实验实例对其进行了编号排序。

表 1　动态数据结构库 S

编号(代码)	动态数据结构	说　明
1	AR	数组
2	AR_P	指针数组
3	SLL	单链链表
4	DLL	双链链表
5	SLL_O	带有游历指针的单链链表
6	DLL_O	带有游历指针的双链链表
7	SLL_AR	单链链表数组
8	DLL_AR	双链链表数组
9	SLL_ARO	数组和游历指针的单链链表
10	DLL_ARO	数组和游历指针的双链链表

动态数据结构搜索举例如下。

若某段原始应用程序中有两个变量 v1、v2,创建 v1 和 v2 的代码为:

```
list<T1> * v1 = new list<T1>();
vector<T2> * v2 = new vector<T1>();
…
```

经搜索过程之后,得到一个候选解:

推荐 v1、v2 分别被实例化为 S 中的 DLL_AR 和 SLL,即优化后的代码为:

```
DLL_AR<T1> * v1 = new DLL_AR<T1>();
SLL<T2> * v2 = new SLL<T2>();
…
```

收稿日期:2010 - 05 - 09;**修回日期**:2010 - 07 - 21。

作者简介:王晓升(1964 -),男,山东平度人,副教授,主要研究方向:计算智能、嵌入式系统。

一般地,假定要优化的应用程序包含要被实例化为库 S 里某一个动态数据结构的变量集合为 V,则优化过程的最终目标是获得一个二元集(变量,动态数据结构)$\{v_i \in V, s_j \in S\}$,以便最小化目标嵌入式系统的内存访问、内存使用和能量消耗,这是一个多目标优化问题。对应用程序的动态数据结构优化的整个流程如图 1 所示,主要包括 3 个阶段。

1) 记录应用程序的框架。为了从应用程序中获取实际的执行信息,在编译或执行时需要创建一个登记应用程序如下信息的框架报告:某元素的访问数量和地址、元素的增加、元素的移除、容器的清除、迭代器操作如预增加或废弃、构造函数、析构函数、拷贝构造函数和交换操作。

2) 参数评估。从应用程序的框架报告中提取如下信息或参数:候选变量的数量、存储在动态数据结构(最坏的情况下)中元素的数量、所存储元素数量的平均数、元素的大小(字节数)、指针的大小(字节数)、读访问的数量、写访问的数量以及 Cache 未命中的数量。利用这些参数在动态数据结构搜索中评估一个解(v_i, s_j)的质量[5]。

3) 执行多目标优化。利用第二阶段评估到的参数作为多目标进化算法的输入,最小化 3 种不同的目标:存储器访问、存储器使用和能量消耗。通过运行多目标进化算法,获得了动态数据结构的一个实例集$\{v_i \in V, s_i \in S\}$,应用程序中的变量由该实例集实例化即得到优化的程序代码。

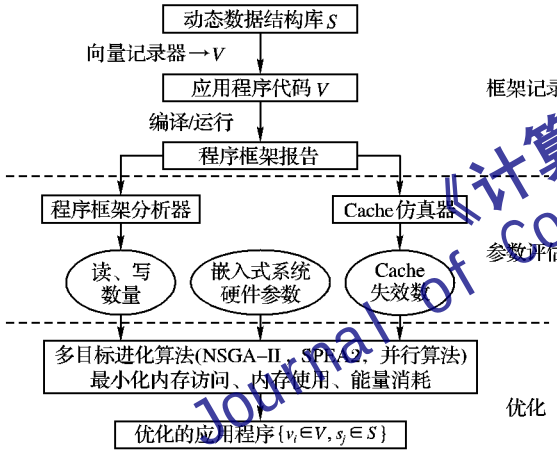

图 1 动态数据结构优化流程

2 进化算法

2.1 串行 NSGA-II 和 SPEA2 算法[6-7]

与单目标优化算法不同,要实现一种多目标优化的进化算法需要解决两个主要问题:1)怎样逼近 Pareto 最优前沿(Pareto Optimal Front, POF),它代表了相矛盾的多目标的折中解;2)怎样使搜索到的解具有良好的多样性。非支配解排序遗传算法 NSGA-II 和增强 Pareto 进化算法 SPEA2 是两个具有代表性的多目标优化的精英算法,其获取 Pareto 最优解的策略是采用精英保留操作,而保留精英个体的方式是外部存储器法,当算法结束后,将外部存储器中的解作为对问题的 Pareto 最优解集的近似。NSGA-II 和 SPEA2 相同的进化算法框架如下:

第 1 步 初始化群体 P;

第 2 步 从 P 中选择精英解到外部存储器 A 中;

第 3 步 从 P 和 A 两者或者两者之一中创建配对库;

第 4 步 基于配对库繁殖(复制),利用进化算子创建下一代 P;

第 5 步 A 与 P 结合;

第 6 步 如果外部条件不满足,则转第 2 步。

2.2 解的编码

为了运用多目标进化算法,需要定义一个基因表示法,表示全部可能的动态数据结构的实现的空间。一个个体编码的例子如下:

S	1	3	4	…	3
V	v_1	v_2	v_3	…	v_n

第一行为染色体(个体),其编码长度为应用程序中需优化的变量的个数,每个基因采用一个十进制整数表示,代表需要实例化相应变量的动态数据结构(来自表 1),例如,第二个变量 $v_2 \in V$ 将由 $s_3 \in S$(对应表 1 的 SLL)实例化。

2.3 解的评估

完成解的编码后,需要评估由个体表示的解。在此考虑一个基本存储层次,它由一个主共享存储器和一个 L1 数据 Cache 组成,可按照每个解对存储器访问的能量值作为解的评估值。因动态数据结构的生命周期短,则能量值采用共享方式按下式计算:

$$E = (N_1 \times E_1) + (N_2 \times E_2) + (M \times E_3)$$

其中:N_1 是数据 Cache 未命中的数量,N_2 是读/写 L1 数据 Cache 的数量,E_1 是每次访问主存的能量消耗,E_2 是每次访问 Cache 的能量消耗,M 为存储器踪迹(以字节数表示),E_3 是由主存产生的静态能量消耗。其他形式的存储层次可对照修改公式即可。以上参数可由实际应用程序的框架报告通过计算得到。

2.4 并行算法

本节结合两个串行版的多目标进化算法 SPEA2 和 NSGA-II,提出一种粗粒度的并行多目标进化算法——PMOEA-NS,在一个并行环境中完成嵌入式应用程序中动态数据结构实现的搜索。算法的基本思想是:通过使用一种岛屿模型,将种群划分为若干子种群,将各子种群分别分配给多个线程,并且使每个子种群分别独立地执行 NSGA-II 或 SPEA2 算法,实现并行求解。个体的数量、适应度函数、操作算子和终止条件皆与串行算法中的相同。算法的一般框架如下:

```
{
  创建 N 个子种群;
  for (每个子种群)
  {
    为子种群指定算法∈{SPEA2, NSGA-II};
    指定子种群的邻居;
    在一个独立线程中 Run(sub_population);
  }
}
Run(subpop)                                    //运行线程函数定义
{
  施加遗传算子;
  if (current_Generation mod 100 ==0)          //假设每进化 100 代
  {
    选择最好的个体;
    Send(subpop, individual, neighbour);
                                    //把最好的个体发送到邻居
  }
}
Send(subp, indi, neighb)                       //发送函数定义
{
  while(subp.current_Generation > neighb.current_Generation)
  {等待;}
  neighb.receive(indi);                        //相邻子种群接收个体
}
```

该算法是基于多线程设计的,适合在多核处理器的系统上运行。岛屿数量的配置,一般视种群规模和并行环境的并行程度而定。

3　实验结果

对一个实际的嵌入式应用程序——3D 游戏引擎[8]进行了优化评估,对该应用程序,登记了 3 128 个变量和 10 个可能的包含在表 1 的动态数据结构。

为了比较串行算法(NSGA-II 或 SPEA2,称为 SMOEA-NS)和并行算法 PMOEA-NS 的性能,本文 PMOEA-NS 具体采用了 4 个岛屿配置,此时相应有 3 个不同的 PMOEA-NS 算法:1)每个岛屿分别执行 NSGA-II、SPEA2、NSGA-II、SPEA2,该算法简记为 NS2;2)所有岛屿仅执行 SPEA2 算法,该算法简记为 S4;3)所有岛屿仅执行 NSGA-II 算法,该算法简记为 N4。所有的参数设置成与在串行算法中相同:外部存储器的大小(存储非支配解)等于初始群体规模,对每个岛屿群体规模设置为 200,代的数目设置为 2 000,交叉概率为 0.80 和变异的概率为 0.01 与在串行算法中相同。所有算法使用C++ 实现,其中 PMOEA-NS 是基于多处理器的多线程的C++ 应用程序。在 Intel P4(双核 CPU 2.4 GHz、2 GB DDR 内存)机器上,分别使用 SPEA2、NSGA-II、NS2、S4、N4 共 5 个算法进行动态数据结构的搜索。通过 100 次测试,分别计算出 Pareto 最优解的收敛性测度(C 测度值)和多样性测度(SP 测度值)[9-10]的平均值。

表 2 给出了分别执行 5 个算法所获得的非支配个体数量、执行时间、SP 测度和 C 测度的结果。结果表明:1)NSGA-II 提供了与 SPEA2 相同数量的非支配个体,NS2 提供了比 NSGA-II 和 SPEA2 多 62%、比 S4 多 28%,以及比 N4 多 26% 的最优解;2)在执行时间上,并行算法比串行算法在寻求最优解上所用的时间少,最快的是在两处理器上执行 N4 算法,相对于 SPEA2 及一个处理器的情况,速度提高了 86%;3)并行算法比串行算法获得较低的 SP 测度值,因此并行算法在解的多样性上优于串行算法;4)NS2 算法的 C 测度值最好、收敛性最优。因此,比较其他算法,在实现嵌入式应用程序中动态数据结构优化上,并行算法中的 NS2 算法是更佳的解决方案。

图 2 为应用程序的优化结果对比图。实验时,使用 AR、ARP、SLL 等实现动态数据结构。3 个目标(存储器访问、存储器使用、能量消耗)都被标准化成 AR 动态数据类型并用对数描述。为了综合比较这 5 个算法,该图也说明了通过图 1 描述的优化流程,达到了优化水平和最终的效益,而且该图也显示,在多目标之中 NS2 算法是最好的解决方案。

表 2　非支配解的数量、执行时间、SP 测度、C 测度比较

算法		非支配解的数量	执行时间/s		SP 测度	C 测度				
			单处理器(AMD Sempron)	两处理器(Intel P4 Core2)		NSGA-II	SPEA2	N4	S4	NS2
SMOEA-NS	NSGA-II	191	1899	709	0.002	—	0.046	0.072	0.031	0.085
	SPEA2	191	3027	1330	0.002	0.071	—	0.141	0.062	0.155
PMOEA-NS	N4	335	982	427	0.001	0.022	0.012	—	0.100	0.386
	S4	326	1698	710	0.001	0.025	0.019	0.271	—	0.518
	NS2	452	1201	560	0.001	0.018	0.003	0.156	0.230	—

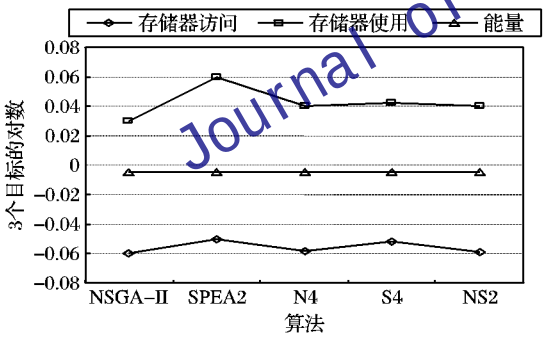

图 2　各种动态数据类型被标准化为 AR 实现的 3 目标优化的结果(对数)

4　结语

本文引入岛屿模型和迁移算子,结合了两个多目标进化算法 NSGA-II 和 SPEA2,提出了一种基于多线程机制的并行多目标进化算法——PMOEA-NS,以最小化内存访问、内存使用和能量消耗为目标,解决多媒体嵌入式系统中动态数据结构优化实现问题。分别使用 NSGA-II、SPEA2 串行算法和 N4、S4、NS2 并行算法进行实验,实验结果表明:并行实现不但提高了优化过程的速度,而且改善了解的质量和种类(多样性),对于庞大的种群(其中 NS2 算法优于其他算法)尤其如此。

参考文献:

[1] PANDA P R, CATTHOOR F, DUTT N D, *et al*. Data and memory optimization techniques for embedded systems [J]. ACM Transactions Design Automation Electronic System. 2001, 6(2): 149 - 206.

[2] BENINI L, MICHELI G. System level power optimization techniques and tools [J]. ACM Transactions Design Automation Electronic System. 2000, 5(2): 116 - 193.

[3] DAYLIGHT E G, ATIENZA D, VANDECAPPELLE A, *et al*. Memory-access-aware data structure transformations for embedded software with dynamic data accesses [J]. IEEE Transactions Very Large Scale Integrated (VLSI) System, 2004, 12(3): 270 - 281.

[4] ATIENZA D, BALOUKAS C, PAPADOPOULOS L, *et al*. Optimization of dynamic data structures in multimedia embedded systems using evolutionary computation [C]// SCOPES '07: Proceeding of the 10th International Workshop on Software & Compilers for Embedded Systems. New York: ACM Press, 2007: 32 - 41.

[5] EDLER J, HILL M D. Dinero IV trace-driven uniprocessor cache simulator [EB/OL]. [2008 - 10 - 16]. http://pages.cs.wisc.edu/~markhill/DineroIV.

[6] ZITZLER E, LAUMANNS M, THIELE L. SPEA2: Improving the strength pareto evolutionary algorithm [EB/OL]. [2009 - 12 - 12]. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.112.5073&rep=rep1&type=pdf.

[7] DEB K, PRATAP A, AGARWAL S, A fast and elitist multiobjective genetic algorithm: NSGA-II [J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182 - 196.

[8] KHAREVYCH L, KHAN R. 3D physics engine for elastic and deformable bodies [EB/OL]. [2008 - 08 - 25]. http://www.cs.umd.edu/Honors/reports/kharevych.html.

[9] 李密青, 郑金华, 罗彪, 等. 一种基于邻域的多目标进化算法[J]. 计算机应用, 2008, 28(6): 1570 - 1572.

[10] 申晓宁. 基于进化算法的多目标优化方法研究[D]. 南京: 南京理工大学, 2008.