论 文

基于置换图的便笺存储器分配

汪黎*, 杨学军, 戴华东

国防科技大学计算机学院, 长沙 410073

* 通信作者. E-mail: liwang@nudt.edu.cn

收稿日期: 2012-09-13; 接受日期: 2012-12-15

国家自然科学基金 (批准号: 61003081) 和国家创新研究群体科学基金 (批准号: 60921062) 资助项目

摘要 在当今的嵌入式系统中, 广泛地将片上存储器组织为软件管理的便笺存储器 (SPM). Li 等研究发现, 对于很多嵌入式应用, 其相干图中的数组生存期满足包含性. 他们证明了满足生存期包含性的数组相干图为超完美图, 并提出了一个基于超完美图的 SPM 分配算法. 他们的算法在面向嵌入式应用的 SPM 分配上获得了当前最好的性能. 本文进一步证明满足生存期包含性的数组相干图为置换图. 置换图是超完美图的一个子类. 在现有技术的情况下, 置换图在判定及区间着色方面比超完美图有优势, 如存在线性时间的识别算法, 存在线性时间的最优区间着色算法. 基于此理论结果, 我们将 Li 等的算法在保留原算法逻辑的基础上, 改进为基于置换图. 实验表明, 改进后的算法在很多不满足生存期包含性的相干图上仍能取得最优 SPM 分配, 获得比基于超完美图的分配算法更好的分配结果.

关键词 便笺存储器 SPM 分配 区间着色 超完美图 置换图

1 引言

在现代的嵌入式系统中, 广泛地将片上存储器组织为软件管理的便笺存储器 (scratchpad memory, SPM). 跟 cache 相比, SPM 在面积、功耗、访问速度等方面均具有优势^[1]. 基于 SPM 的系统, 需要软件显式地插入传输语句来控制数据进入 SPM 的时机和位置. 通常 SPM 的大小远小于待运行程序的数据集, 如何有效地利用有限的 SPM 空间, 将尽可能多的经常访问的数据分配到 SPM 中, 使存储碎片尽量少, 是提高基于 SPM 的系统上程序性能的关键.

SPM 分配问题, 就是把程序中访问的数组 (表示为生存期, live ranges) 分配到 SPM 中, 并且相干 (interfering, 即同时存活) 的两个生存期必须分配到互不交叠的 SPM 空间. 生存期之间的相干性通过对程序进行生存期分析 (liveness analysis) 获得, 表示为相干图 (interference graph, IG). 相干图为一个带权无向图, 图中每个结点对应一个待分配到 SPM 的数组生存期, 结点的权值等于对应数组的大小. 如果两个结点相邻 (有边相连), 表示对应的两个数组生存期相干. SPM 分配问题, 即为相干图中的每个结点分配一个 SPM 空间, 满足相邻的结点所分配的 SPM 空间不交叠. 寻找最优的 SPM 分配, 就是使得所需 SPM 空间最少.

给定一个带权图 $G = (V, E)$ 和一个顶点权值映射函数 $w : V \rightarrow \mathbb{N}$, $\mathbb{N}$ 为自然数集, 区间着色^[2] (interval coloring) 即为 V 中每个顶点 x 指派数轴上一个区间 I_x , 且满足以下条件: (1) $\forall x \in V$,

引用格式: 汪黎, 杨学军, 戴华东. 基于置换图的便笺存储器分配. 中国科学: 信息科学, 2013, 43: 932–946

图 1 数组生存期之间的关系

Figure 1 The relationship between two array live ranges. (a) Disjoint-non-interfering; (b) intersecting-interfering; (c) strictly contained-interfering; (d) equal-interfering

$|I_x| = w(x)$; (2) $\forall (x, y) \in E, I_x \cap I_y = \emptyset$. 色数 (常记作 $\chi(\mathcal{G}; w)$) 是对带权无向图 $(\mathcal{G}; w)$ 进行区间着色所需的区间跨度总和 $|\bigcup_{x \in V} I_x|$ 的最小值. 带权无向图 $(\mathcal{G}; w)$ 的最优区间着色, 就是寻找 $(\mathcal{G}; w)$ 的色数的问题. 对于一般的图, 其最优区间着色问题被证明是 NP 完全问题 [3].

SPM 分配问题可以自然地建模为图论中对带权图的区间着色问题: 为数组生存期分配 SPM 空间对应到为相干图中的结点分配区间; 最小化所需的 SPM 空间, 对应到寻求相干图的最优区间着色. 对于一个特定的区间着色方案 α , 定义其色数 (记作 $\chi_\alpha(\mathcal{G}; w)$) 为在该区间着色方案下所需的区间跨度总和. SPM 分配即寻找对于相干图的一个区间着色方案 α , 使得 $\chi_\alpha(\mathcal{G}; w) \rightarrow \chi(\mathcal{G}; w)$. 如果有 $\chi_\alpha(\mathcal{G}; w) = \chi(\mathcal{G}; w)$, 该方案是最优的.

定义 $\mathcal{L}(A)$ 表示数组 A 的生存期. 程序中任意两个数组生存期 $\mathcal{L}(A)$ 和 $\mathcal{L}(B)$ 之间的关系只可能为如下几种情况之一: (1) 如果 $\mathcal{L}(A) \cap \mathcal{L}(B) = \emptyset$, 则称 A 和 B 的生存期分离 (disjoint); (2) 如果 $\mathcal{L}(A) \not\subseteq \mathcal{L}(B)$, $\mathcal{L}(B) \not\subseteq \mathcal{L}(A)$, 且 $\mathcal{L}(A) \cap \mathcal{L}(B) \neq \emptyset$, 则称 A 和 B 的生存期交叉 (intersecting); (3) 如果 $\mathcal{L}(A) \subset \mathcal{L}(B)$, 或者 $\mathcal{L}(B) \subset \mathcal{L}(A)$, 则称 A 和 B 的生存期严格包含 (strictly contained); (4) 如果 $\mathcal{L}(A) = \mathcal{L}(B)$, 则称 A 和 B 的生存期相等 (equal). 对于(3)和(4)两种情况, 统称数组 A 和 B 的生存期包含 (相等的两个生存期可以认为是相互包含). 直观地, 两个数组生存期之间的关系如图 1 所示. 其中相干时的关系有两种: 交叉和包含, 其中包含又可细分为严格包含和相等.

如果数组相干图 $\mathcal{G} = (V, E)$ 满足: $\forall a, b \in V, \mathcal{L}(a)$ 和 $\mathcal{L}(b)$ 要么分离, 要么包含, 即不存在图 1(b) 的情况, 则称 $\mathcal{G}$ 满足生存期包含性. Li 等 [4] 研究发现, 很多嵌入式应用的相干图满足生存期包含性. 他们证明了如果相干图满足生存期包含性, 则相干图是超完美图 (superperfect graph), 并提出了一个基于超完美图的 SPM 分配算法. 对于面向嵌入式应用的 SPM 分配, 该算法获得了当前最好的性能.

本文中, 我们进一步证明满足生存期包含性的相干图为置换图 (permutation graph). 置换图是超完美图的子类, 相对超完美图, 置换图在识别和区间着色方面具有优势. 利用这一理论结果, 我们对基于超完美图的 SPM 分配算法进行改进, 使之基于置换图. 实验表明, 改进后的基于置换图的算法能在更多的相干图上取得最优着色, 获得比基于超完美图的分配算法更好的分配结果.

2 相关研究工作

目前国际上提出的 SPM 分配技术可以分为静态 [5~7] 和动态 [4,8~13] 两大类, 本质区别在于是否在运行时将数据在 SPM 和片外存储器之间移动. 早期提出的技术都是静态的, 在整个程序运行期间, 数据要么在片外存储器中, 要么在 SPM 中, 而不会在两者之间动态传输. 显然, 静态分配技术能带来的程序性能提高有限.

动态分配技术显式地在程序中插入传输语句, 将数据在 SPM 和片外存储器之间移动. 动态分配方法能够获得的程序性能提高空间更大. 动态分配方法又可分为 3 类, 基于软件 caching 的, 基于整数规划的 (ILP) 和基于启发式的. 软件 caching 技术^[8,9]是较早提出的动态分配技术, 其基本思想是用软件模拟硬件 cache 对 SPM 进行管理. 软件 caching 技术的开销较大^[14]. 基于整数规划的分配技术^[11]能够获得最优分配, 但算法的开销很大, 对于大规模的程序, 可能无法运行出结果. 最近提出的动态分配技术都是基于启发式的. 其中, 文献 [10] 的技术只能运用于访问模式规整 (访射下标函数) 的循环. 文献 [12] 的方法采用一系列启发式决定如何在 SPM 和片外存储器之间拷贝数据. 文献 [13] 通过将 SPM 进行多次划分形成一个存在别名的寄存器文件, 然后利用一般化的图着色寄存器分配算法进行分配. 该算法在划分 SPM 之前, 需要将待分配数组的大小进行归类和对齐, 这会引入 SPM 空间浪费. 另外, 如果所划分出的数组类的大小不能整除 SPM 的大小, 则在 SPM 的尾部将引入存储碎片. 此外, 由于划分产生的寄存器别名的存在也可能引入存储碎片. Li 等^[4]研究发现很多嵌入式应用的相干图为超完美图, 他们提出了一个基于超完美图的 SPM 分配算法. 对于嵌入式应用, 相对其他基于启发式的动态分配算法, 他们的算法获得了最好的性能.

3 区间着色和置换图

本节介绍图论中关于超完美图、置换图和可比图等概念和性质, 以便于表述我们的证明.

3.1 基本概念

如果图 G 的由顶点集 $A \subseteq V$ 导出的子图 G_A 是完全图, 即图中任意两个顶点均有边相连, 则称 G_A 是团 (clique). 对于一个团 G_A , 如果不存在团 G_B , 满足 $A \subset B$, 则称 G_A 是极大团 (maximal clique). 一个团的阶是该团中所有顶点的权值之和. 而团数 (clique number, 常记作 $\omega(G; w)$) 是图中所有团的阶的最大值. 对于一般的图 G , 总有以下公式成立: $\chi(G; w) \geq \omega(G; w)$.

3.2 超完美图

定义 1 如果一个图 G 满足: 对于任意一种非负顶点权值函数 w , 均有 $\chi(G; w) = \omega(G; w)$, 则称该图为超完美图 (superperfect graph).

直观地, 如果一个图对于任何结点权值指派, 均满足其色数等于团数, 则该图为超完美图.

3.3 置换图

定义 2 令 π 为数字 $1, 2, \dots, n$ 的一个置换 (permutation), 由此置换导出的图 $G[\pi] = (V, E)$ 定义如下: $V = \{1, 2, \dots, n\}$, $(i, j) \in E \Leftrightarrow (i - j)(\pi_i^{-1} - \pi_j^{-1}) < 0$. 对于无向图 G , 如果存在一个置换 π , 使得 $G \cong G[\pi]$ ($\cong$ 表示同构 (isomorphic) 关系), 则 G 是置换图 (permutation graph).

例如, 对于数字 $1, 2, 3$, 我们定义一个置换 $\pi[1, 2, 3] = 3, 1, 2$, 则由 π 导出的置换图如图 2 所示.

置换图是超完美图的一个子类, 是超完美图的真子集. 因此, 置换图一定是超完美图, 但超完美图不一定是置换图. 相对超完美图, 在现有技术条件下, 置换图在识别和区间着色方面具有优势. 假设图的顶点数和边数分别为 n 和 m , McConnell 等^[15]提出了复杂性为 $O(n + m)$ 的置换图识别算法. 他们还提出了复杂性同样为 $O(n + m)$ 的置换图的最优区间着色算法^[16] (准确地讲, 他们提出了对于可比

图 2 由置换 $\pi[1, 2, 3] = 3, 1, 2$ 导出的置换图Figure 2 The permutation graph induced by $\pi[1, 2, 3] = 3, 1, 2$ 图 3 一个带权无向图的两个无环定向 α 和 β Figure 3 Two acyclic orientations α and β to a weighted undirected graph. (a) A weighted undirected graph; (b) one acyclic orientation to the graph in (a); another acyclic orientation to the graph in (a)

图, 在线性时间内寻求其传递定向的算法. 而根据一个图的传递定向, 可以利用深度优先搜索在线性时间内获得其最优的区间着色^[2]. 置换图一定是可比图, 因此, 他们的算法实际上给出了线性时间复杂性的置换图的最优区间着色算法. 关于可比图、传递定向以及可比图与置换图的关系, 详见下面的介绍). 对于超完美图, 目前还没有多项式时间复杂性的识别算法, 也没有多项式时间复杂性的最优区间着色算法.

下面我们介绍置换图的一个判定定理, 该定理用于第 4 节中我们的证明. 在此之前, 我们首先介绍可比图 (comparability graph) 的概念和性质.

令 $\mathcal{G} = (V, E)$ 为一个无向图. 对 $\mathcal{G}$ 的一个定向 (orientation) 为一个函数 α , 对于 $\forall (x, y) \in E$, $\alpha(x, y) \in \{(x, y), (y, x)\}$, 即为每条边指定一个方向. 令 $\mathcal{G}_\alpha$ 为用弧 $\alpha(x, y)$ 替换每条边 (x, y) 得到的有向图, 如果 $\mathcal{G}_\alpha$ 不含有向环 (directed cycle), 则称定向 α 是无环的 (acyclic).

定义 3 对一个无向图 $\mathcal{G}$ 的定向 α , 如果满足当 $(x, y), (y, z) \in \mathcal{G}_\alpha$ 时, 总有 $(x, z) \in \mathcal{G}_\alpha$ 成立, 则称定向 α 是传递的 (transitive).

定义 4 一个无向图 $\mathcal{G}$, 如果存在一个传递定向, 则 $\mathcal{G}$ 是可比图 (comparability graph).

一个传递定向一定是无环定向, 但反之不一定. 例如, 对于图 3(a) 中的无向图, 图 3(b) 和 (c) 中分别给出两个无环定向 α 和 β . 其中, α 不是传递定向, 因为 $(x, b), (b, a) \in \mathcal{G}_\alpha$, 而 $(x, a) \notin \mathcal{G}_\alpha$. β 为传递定向, 从而 $\mathcal{G}$ 是可比图.

一个图 $\mathcal{G} = (V, E)$ 的补图 (complement graph) $\overline{\mathcal{G}} = (V, \overline{E})$ 定义为, $\overline{E} = \{(x, y) \in V \times V \mid x \neq y \text{ and } (x, y) \notin E\}$.

定理 1 一个图 $\mathcal{G}$ 是置换图 (permutation graph), 当且仅当 $\mathcal{G}$ 和 $\overline{\mathcal{G}}$ 均为可比图.

定义 5 令 $\mathcal{G}_0$ 为有 n 个顶点 $v_1, v_2, \dots, v_n$ 的图, $\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_n$ 为 n 个分离的子图. 这些图可以是有向或无向的. 定义组合图 $\mathcal{G} = \mathcal{G}_0[\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_n]$ 为: 首先用 $\mathcal{G}_i$ 替换 $\mathcal{G}_0$ 中的顶点 v_i . 然后, 对于所

图 4 组合图示例

Figure 4 Composition graph creation ($n = 3$)

有 $1 \leq i, j \leq n$, 如果在 G_0 中, 顶点 v_i 跟 v_j 相连, 则让 G_i 的每个顶点跟 G_j 的每个顶点相连. 形式地, 对于 $G_i = (V_i, E_i)$, 定义 $G = (V, E)$ 为

$$\begin{cases} V = \bigcup_{1 \leq i \leq n} V_i; \\ E = \bigcup_{1 \leq i \leq n} E_i \cup \{(x, y) \mid x \in V_i, y \in V_j \text{ and } (v_i, v_j) \in E_0\}. \end{cases}$$

例如, 图 4 给出了一个组合图的例子.

定理 2 令 $G = G_0[G_1, G_2, \dots, G_n]$, 其中所有 G_i 是分离的无向图. 则 G 是可比图当且仅当每个 $G_i (0 \leq i \leq n)$ 是可比图.

4 证明

本节我们证明满足生存期包含性的数组相干图 G 为置换图. 根据定理 1, 即证明 G 和 $\bar{G}$ 均为可比图. 我们首先介绍严格偏序集 (partially ordered set) 的定义及相关性质, 它们将用于本节的证明.

定义 6 如果集合 A 上的一个二元关系 “ $<$ ” 是反自反、反对称和传递的, 则称 “ $<$ ” 为 A 上的严格偏序 (strict partial order), $\langle A, < \rangle$ 为严格偏序集 (strictly partially ordered set).

定理 3 严格偏序集的子集仍为严格偏序集.

定义 7 $\langle A, < \rangle$ 为严格偏序集, 如果对于 $a \in A$, 不存在 $b \in A$, 使得 $b < a$, 则称 a 为 $\langle A, < \rangle$ 的极小元 (minimal element).

定理 4 非空有限严格偏序集总存在极小元.

接下来, 我们证明 G 是可比图.

定义 8 数组生存期集合 A 满足生存期包含性, 定义 A 上的一个二元关系 “ $\prec$ ”: $\forall a, b \in A$, (1) 如果 a 严格包含于 b 中, 令 $a \prec b$; (2) 如果 a 与 b 相等, 且 a 与 b 是不同的生存期, 则找到所有与 a 相等的生存期, 构成集合 $B = \{s_1, s_2, \dots, s_m\} \subseteq A$. 对于 $\forall s_i, s_j \in B, 1 \leq i < j \leq m$, 令 $s_i \prec s_j$.

定理 5 “ $\prec$ ” 为 A 上的严格偏序.

证明 根据定义 6, 即证明 “ $\prec$ ” 是反自反、反对称和传递的. 对于 $\forall a \in A$, a 与 a 是相等的, 因而不属于定义 8 中规定的第一种情况. 第二种情况要求相等的是两个不同的生存期, 因此也不满足, 所以 $a \not\prec a$, “ $\prec$ ” 是反自反的. 对于 $\forall a, b \in A$, 且 $a \prec b$, 有两种情况: (1) a 严格包含于 b 中, 则 b 不严格包含于 a 中, 也不与 a 相等, 所以 $b \not\prec a$; (2) a 与 b 相等, 则存在序列 $s_1, s_2, \dots, s_m$, 使得 $s_i = a$, $s_j = b$, $1 \leq i < j \leq m$. 由于 $j \neq i$, 因此 $s_j \neq s_i$, 即 $b \not\prec a$. “ $\prec$ ” 是反对称的. 对于任意 $a, b, c \in A$, $a \prec b, b \prec c$, 有以下情况: (1) a 严格包含于 b , b 与 c 相等, 则 a 严格包含于 c 中, 因此 $a \prec c$; (2) a 严

格包含于 b , b 严格包含于 c , 则 a 严格包含于 c 中, 因此 $a \prec c$; (3) a 与 b 相等, b 与 c 相等, 则必存在序列 $s_1, s_2, \dots, s_m$, 使得 $s_i = a, s_j = b, s_k = c, 1 \leq i < j < k \leq m$. 由于 $i < k$, 因此 $s_i \prec s_k$, 即 $a \prec c$; (4) a 与 b 相等, b 严格包含于 c 中, 则 a 严格包含于 c 中, 因此 $a \prec c$. “ $\prec$ ” 是传递的.

定义 9 定义严格偏序集 $\langle A, \prec \rangle$ 的导出图 $\mathcal{G}_{\langle A, \prec \rangle} = (V, E)$ 为: 对于 $\forall a \in A$, 引入一个顶点 $v_a \in V$. 对于 $\forall v_a, v_b \in V$, 引入边 (v_a, v_b) 当且仅当 $a \prec b$.

定理 6 $\mathcal{G}_{\langle A, \prec \rangle} \cong \mathcal{G}$, 其中 $\mathcal{G}$ 为 A 中元素构成的相干图.

证明 显然 $V(\mathcal{G}_{\langle A, \prec \rangle}) = V(\mathcal{G}) = A$. 由于 “ $\prec$ ” 是反自反、反对称的, 因此 $\mathcal{G}_{\langle A, \prec \rangle}$ 中不含自圈 (loops) 和多重边 (multiple edges). 对于 $\forall (a, b) \in E(\mathcal{G}_{\langle A, \prec \rangle})$, 则 $a \prec b$ 或 $b \prec a$ 成立, 根据 “ $\prec$ ” 的定义, a 与 b 要么严格包含, 要么相等, 此时均有 a 与 b 相干, 因此 $(a, b) \in E(\mathcal{G})$, 即 $E(\mathcal{G}_{\langle A, \prec \rangle}) \subseteq E(\mathcal{G})$. 对于 $\forall (a, b) \in E(\mathcal{G})$, 则 a 与 b 相干. 由于 A 满足生存期包含性, a 与 b 要么严格包含, 要么相等, 此时必有 $a \prec b$ 或 $b \prec a$ 成立, 因此 $(a, b) \in E(\mathcal{G}_{\langle A, \prec \rangle})$, 即 $E(\mathcal{G}) \subseteq E(\mathcal{G}_{\langle A, \prec \rangle})$. 因此, $E(\mathcal{G}) = E(\mathcal{G}_{\langle A, \prec \rangle})$. 则 $\mathcal{G}_{\langle A, \prec \rangle} \cong \mathcal{G}$.

接下来证明一系列引理:

引理 1 对于定义 8 中定义的严格偏序集 $\langle A, \prec \rangle$, 如果 $\forall a, b, c \in A$ 满足: $c \prec a$ 且 $c \prec b$, 则 $a \prec b$ 或 $b \prec a$ 成立.

证明 由于 $c \prec a$, 则 c 要么严格包含于 a 中, 要么与 a 相等. 类似地, c 要么严格包含于 b 中, 要么与 b 相等. 则 $c \subseteq (a \cap b)$, $a \cap b \neq \emptyset$, a 与 b 相干. 由于 A 满足生存期包含性, a 和 b 包含 (严格包含或相等), 根据 “ $\prec$ ” 的定义, $a \prec b$ 或 $b \prec a$ 成立.

定义 10 按以下方式从 A 构造图 $\mathcal{G}'$: 初始令 $A' = A$, 每步从 A' 中取出一个极小元 a , 引入 $\mathcal{G}'$ 中的一个新的顶点 v_a . 对于 $\forall v_b \in V(\mathcal{G}')$, 如果 $b \prec a$, 则引入边 (v_a, v_b) . 令 $A' = A' - \{a\}$, 重复上述步骤直到 $A' = \emptyset$.

根据定理 3 和 4, 定义 10 的构造过程是可行的.

引理 2 $\mathcal{G}' \cong \mathcal{G}_{\langle A, \prec \rangle}$.

证明 显然 $V(\mathcal{G}') = V(\mathcal{G}_{\langle A, \prec \rangle}) = A$, $E(\mathcal{G}') \subseteq E(\mathcal{G}_{\langle A, \prec \rangle})$. 对于 $\forall (a, b) \in E(\mathcal{G}_{\langle A, \prec \rangle})$, 不失一般性, 令 $a \prec b$, 则 b 必在 a 之后加入 $\mathcal{G}'$, 因此 $(a, b) \in E(\mathcal{G}')$, $E(\mathcal{G}_{\langle A, \prec \rangle}) \subseteq E(\mathcal{G}')$, 从而, $E(\mathcal{G}') = E(\mathcal{G}_{\langle A, \prec \rangle})$.

引理 3 在定义 10 的构造过程中, 对于 $\forall a \in V(\mathcal{G}')$, 假设 b 为 a 所在连通子图中最后加入的结点, 且 $a \neq b$, 则 $a \prec b$.

证明 由于 a 跟 b 在同一连通子图, 因而 a 与 b 间必存在一条最短路径 $\mathcal{P} = \{v_1, v_2, \dots, v_m\}$, 其中 $v_1 = a, v_m = b$. 假设 $m \geq 3$, 考虑子路径 $\{v_1, v_2, v_3\}$. 对于边 (v_1, v_2) , 假设 $v_2 \prec v_1$, 考虑边 (v_2, v_3) , 假设 $v_2 \prec v_3$, 根据引理 1, 则 $v_1 \prec v_3$ 或 $v_3 \prec v_1$ 成立, 因此 v_1 与 v_3 相邻, 这与 $\mathcal{P}$ 为最短路径矛盾; 假设 $v_3 \prec v_2$, 根据 “ $\prec$ ” 的传递性, $v_3 \prec v_1$, v_3 与 v_1 相邻, 与 $\mathcal{P}$ 为最短路径矛盾; 假设 $v_1 \prec v_2$, 如果 $v_2 \prec v_3$, 根据传递性, $v_1 \prec v_3$, v_1 与 v_3 相邻, 与 $\mathcal{P}$ 为最短路径矛盾; 因此, $v_3 \prec v_2$. 此时 v_2 应在 v_3 之后加入. 如果 $m = 3$, 则与 v_3 是最后加入的结点矛盾; 因此 $m > 3$, 考虑结点 v_4 , 如果 $v_4 \prec v_3$, 而根据前面有 $v_3 \prec v_2$, 根据传递性, 则 $v_4 \prec v_2$, v_4 与 v_2 相邻, 与 $\mathcal{P}$ 为最短路径矛盾; 如果 $v_3 \prec v_4$, 根据前面有 $v_3 \prec v_2$, 根据引理 1, 则 $v_4 \prec v_2$ 或 $v_2 \prec v_4$ 成立, v_2 与 v_4 相邻, 与 $\mathcal{P}$ 为最短路径矛盾. 综上, 应有 $m < 3$, 即 $m = 2$, 即存在边 (a, b) , 则 $a \prec b$ 或 $b \prec a$ 成立. 而 b 在 a 之后加入, 因此 $a \prec b$.

引理 4 在定义 10 的构造过程中, 任意一步加入一个结点 a 时, 如果图中存在结点 b , 满足 $b \prec a$. 假设 c 为 b 所在连通子图中任意一点, 则 $c \prec a$.

证明 假设 d 为 b 所在连通子图中最后加入的结点, 根据引理 3, $b \prec d$; 而 $b \prec a$, 根据引理 1,

图 5 G_0 和对它的传递定向

Figure 5 G_0 and its transitive orientation. (a) A decomposed graph; (b) the transitive orientation to the graph in (a)

图 6 解释定理 7 的证明的一个例子

Figure 6 An example illustrating the proof of Theorem 7. (a) An array live range set; (b) the derived comparability graph from live range set in (a); (c) the composition graph when inserting node 9

$a \prec d$ 或 $d \prec a$ 成立, 而 a 晚于 d 加入, 因此 $d \prec a$. 而根据引理 3, $c \prec d$, 根据 “ $\prec$ ” 的传递性, $c \prec a$.

直观地, 引理 4 说明, 对于当前加入的结点 a , 如果 a 与图 G' 中某一点 b 相邻, 则 a 必与 b 所在连通子图中每一点都相邻. 基于以上结果, 我们给出定理 7 的证明.

定理 7 G 为可比图.

证明 根据定理 6 和引理 2, 即证明 G' 为可比图. 我们证明, 根据定义 10 的构造方法, 每步构成的图 G' 均为可比图. 特别地, 当将 A 中结点全部加入时, 得到的图 $G' \cong G$ 为可比图. 简单起见, 下面的证明中用 G 表示 G' . 第一步, 加入一个结点, 此时 G 中只有一个结点, 没有边, 为可比图. 假设加入 k 个结点后, G 为可比图. 假设此时图中有 m 个连通子图, 则每个连通子图均为可比图. 当加入第 $k+1$ 个结点 a 时, 对于上一步得到的图中任意一个连通子图 G_i , 如果存在 $b \in V(G_i)$, 使得 $b \prec a$, 根据引理 4, 则该连通子图中任意一点 c 均应满足: $c \prec a$, 即 a 与 G_i 中每个结点均相邻. 不失一般性, 假设 a 与子图 $\{G_1, G_2, \dots, G_j\}$ 相邻, 根据定义 5 中图的组合操作, 如果我们让每个 G_i “塌缩” 为一个结点 v_i , 表示生成的 “降解图” 为 G_0 , 则加入结点 a 后的图为: $G = G_0[G_1, G_2, \dots, G_m, G_{m+1}]$. 其中 G_0 如图 5(a) 所示, 每个结点 v_i 相应于加入 a 之前图中的第 i 个连通子图 G_i , $i = 1, 2, \dots, m$. G_{m+1} 为只有一个顶点的图, 与 G_0 中的顶点 a 对应. 由于 G_i 均为可比图, 根据定理 2, 如果 G_0 为可比图, 则 G 为可比图. 要证明 G_0 为可比图, 根据定义 4, 只需要给出 G_0 的一个传递定向. 如图 5(b) 所示, G_0 总存在如下传递定向: 令 a 为源点, 所有与 a 相连的边的定向均为从 a 指向另一个顶点. 因此 G 为可比图.

下面, 我们用一个例子解释定义 10 的构造方式及定理 7 的证明. 假设数组生存期集合如图 6(a) 所示, 满足生存期包含性. 这些生存期之间的包含关系可直观看出. 其中, 结点 6 和 7 相等, 假设在关系 “ $\prec$ ” 下有结点 $7 \prec 6$. 根据定义 10 的构造方式, 同时规定在某步时, 如果集合中存在多个极小元, 选择编号最小的一个, 则顶点加入图的顺序依次为: $[2, 1, 5, 4, 3, 8, 7, 6, 9]$, 按照定理 7 的证明给出的定向方式, 其生成的图如图 6(b) 所示, 显然它是可比图. 当插入结点 9 时, 组合图 $\mathcal{G} = \mathcal{G}_0[\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathcal{G}_4]$ 如图 6(c) 所示.

接着我们证明 $\bar{\mathcal{G}}$ 是可比图. $\bar{\mathcal{G}} = (V, \bar{E})$ 的构造方法为: 对于满足生存期包含性的数组生存期集合 A , 对于 $\forall a \in A$, 引入顶点 $v_a \in V$. 对于 $\forall v_a, v_b \in V$, 引入边 $(v_a, v_b) \in \bar{E}$ 当且仅当 a 与 b 分离, 即 a 与 b 不相干. 下面我们根据定义 4 证明 $\bar{\mathcal{G}}$ 是可比图.

定理 8 $\bar{\mathcal{G}} = (V, \bar{E})$ 是可比图.

证明 对于 $\forall (x, y) \in \bar{E}$, 定义 (x, y) 的定向为: 如果生存期 x 结束存活的时间早于 y 开始存活的时间, 则令定向为 $x \rightarrow y$; 否则 $y \rightarrow x$. 由于 $(x, y) \in \bar{E}$ 表明 x 与 y 的生存期不相干, 因此必有 a 结束存活的时间早于 b 开始存活的时间, 或反之成立. 因此以上定义将对每条边给定一个唯一的定向. 该定向显然是无环的. 该定向的传递性证明如下, 如果有边 $(x, y), (y, z) \in \bar{E}$ 满足定向为: $x \rightarrow y, y \rightarrow z$, 则必有 x 结束存活的时间早于 z 开始存活的时间, 此时必有 x 与 z 不相干, 则 $(x, z) \in \bar{E}$, 且定向为 $x \rightarrow z$. 因而, 该定向是传递定向. 根据定义 4, $\bar{\mathcal{G}}$ 为可比图.

推论 1 $\mathcal{G}$ 为置换图.

根据定理 7 和 8, 以及定理 1, 推论 1 成立.

5 基于置换图的便笺存储器分配算法

我们的基于置换图的便笺存储器分配算法框架如算法 1 所示. 算法以程序的数组相干图作为输入. 第 4 行, 算法检查相干图 $\mathcal{G}$ 是否是置换图, 如果不是置换图, 则调用第 5 行的过程 `ExtendLiveRange` 延长某些数组的生存期, 使其满足生存期包含性. 算法第 7 行计算当前相干图 (置换图) $\mathcal{G}$ 的最优区间着色. 如果着色数大于 SPM 的大小 (第 8 行), 则算法找到 $\mathcal{G}$ 中所有阶数大于 SPM 大小的极大团, 然后从这些极大团中选择数组进行生存期分割 (第 9 行). 在生存期分割的过程中, 始终保持相干图满足生存期包含性, 直到当前相干图的着色数小于等于 SPM 的大小. 然后返回其对应的最优区间着色, 对应到一个 SPM 分配方案 (第 12 行).

与初始基于超完美图的算法^[4]相比, 我们的算法框架基本相同, 这也说明我们的改进可以非常容易地容纳入他们的算法框架, 从而保留原算法框架的有效性. 我们的算法的不同之处在于, 第 4 行, 我们检查输入相干图是否是置换图, 利用文献 [15] 提出的识别算法, 其可以在线性时间 $O(n + m)$ 内完成, n 和 m 分别为图的顶点数 (即生存期个数) 和边数. 相比之下, 原算法检查当前相干图是否满足生存期包含性, 这需要对所有生存期中的任意两个进行比较, 从而其复杂度为 $O(n^2)$ (C_n^2). 更重要的是, 在嵌入式应用中, 可能存在一些相干图, 图中绝大多数生存期满足包含性, 但有少量生存期与别的生存期存在交叉 (intersecting), 此时, 这些相干图不满足生存期包含性. 由于超完美图不存在多项式时间的识别算法, 当相干图不满足生存期包含性时, 原算法调用 `ExtendLiveRange` 过程延长某些生存期使之满足生存期包含性. 众所周知, 生存期延长会引入更多的相干性, 很可能增大 SPM 压力, 导致溢出 (spills), 从而降低性能. 然而, 对于这些相干图, 如第 6 节所验证, 很大一部分仍然是置换图, 此时我们的算法仍然可以获得最优区间着色, 即最优 SPM 分配, 无需进行生存期延长. 因此, 我们的算

法从理论上确保获得相等或更优的性能. 另一个改进来自于过程 `OptimalIntervalColoring` 的实现, 采用文献 [16] 提出的面向置换图的最优区间着色算法, 该过程可以在线性时间 $O(n + m)$ 内完成. 而原算法需要构造所有生存期的一个全序 (total order), 当前最快的排序算法的复杂度为 $O(n \log n)$.

Algorithm 1 Permutation graph-based SPM allocation

```

1: procedure PGC_SPM_Allocation
2: Input: IG:  $\mathcal{G} = (V, E)$ 
3: Output: an interval coloring  $\alpha$ 
4: if Is_Permutation_Graph( $\mathcal{G}$ )==false then
5:   Extend_Live_Range( $\mathcal{G}$ )
6: end if
7:  $\alpha$ =OptimalIntervalColoring( $\mathcal{G}$ )
8: while  $\chi_{\alpha}(\mathcal{G}) > \text{SPM\_SIZE}$  do
9:   Live_Range_Splitting( $\mathcal{G}$ )
10:   $\alpha$ =OptimalIntervalColoring( $\mathcal{G}$ )
11: end while
12: Return  $\alpha$ 
13: end procedure
  
```

6 实验

为了证明我们的算法能够在更多的数组相干图上获得最优着色, 我们随机地产生数组相干图, 使得相干图中的绝大多数结点满足生存期包含性, 只有少量的结点不满足. 此时, 这些相干图不满足生存期包含性. 如第 5 节所述, 对于这样的相干图, 基于超完美图的算法将需要首先对这些不满足生存期包含性的结点进行生存期延长, 使之满足生存期包含性. 生存期延长很可能会在相干图中引入更多的边, 从而降低图的可着色性, 即需要更大的 SPM 空间. 相比之下, 由于置换图具有线性时间的识别算法, 我们可以首先判断这些图 (即使它们不满足生存期包含性) 是否是置换图. 如果仍然是置换图, 则我们的算法仍然能够获得最优着色, 而无需进行生存期延长.

随机产生相干图的算法如算法 2 所示. 算法随机地产生数轴上的区间, 表示数组的生存期. 指定数轴的总宽度为 $[0, \text{RIGHT_END}]$, 表示程序的总运行时间. 算法第 1 行的循环共产生 `GRAPH_NUM` 个图. 第 2 行产生介于 $[\text{MIN_NODE_NUM}, \text{MAX_NODE_NUM}]$ 之间的一个随机数 `total_node_num`, 表示图中的顶点个数. 算法中的所有随机数服从离散均匀分布 (discrete uniform distribution), 用 Matlab 的 `unidrnd` 函数产生. 第 3 行产生介于 $[\text{MIN_OTHER_RATIO}, \text{MAX_OTHER_RATIO}]$ 之间的一个随机数 `other_node_ratio`, 表示不满足生存期包含性的生存期占有所有生存期个数的比例. 第 4 行计算满足生存期包含性的生存期个数. 第 5 行计算不满足生存期包含性的生存期个数. 第 6 行的循环产生所有的生存期. 对于一个特定的生存期, 第 7 行产生其长度, 为 $[1, \text{MAX_LIVE_RANGE_LENGTH}]$ 之间的一个随机数. 实验中 `MAX_LIVE_RANGE_LENGTH=50`, 即合理地假设长度超过 50 的生存期都应该进行分割. 第 8 行产生生存期开始的时间. 第 9 行产生该生存期对应数组的大小, 为 $[1, \text{MAX_LIVE_RANGE_WEIGHT}]$ 之间的一个随机数, 实验中 `MAX_LIVE_RANGE_WEIGHT=6`. 第 10 行的循环检查当前产生的生存期的合法性. 如果要产生的是一个满足生存期包含性的生存期 (nor-

mal_node), 则检查当前产生的生存期是否与所有以前产生的生存期中的任意一个存在交叉情况 (第 12 行), 如果存在, 则转第 7 行重新产生; 如果要产生的是一个不满足生存期包含性的生存期 (other_node), 则当与所有以前产生的生存期都不存在交叉情况时 (第 14 行), 需要重新产生.

Algorithm 2 The algorithm for generating IGs

```

1: for count=1: GRAPH_NUM do
2:   total_node_num=random(MIN_NODE_NUM, MAX_NODE_NUM)
3:   other_node_ratio=random(MIN_OTHER_RATIO, MAX_OTHER_RATIO)
4:   normal_node_num=total_node_num*(1-other_node_ratio)
5:   other_node_num=total_node_num-normal_node_num
6:   for i = 1 : total_node_num
7:     live_range_length(i)=random(1, MAX_LIVE_RANGE_LENGTH)
8:     live_range_start(i)=random(1, RIGHT_END-live_range_length(i))
9:     live_range_weight(i)=random(1, MAX_LIVE_RANGE_WEIGHT)
10:    for j = 1 : i - 1 do
11:      if i ≤ normal_node_num then
12:        Check if i intersects with some j, if so, goto 7
13:      else
14:        Check if i does not intersect with any j, if so, goto 7
15:      end if
16:    end for
17:  end for
18: end for
19: end procedure

```

为了与基于超完美图的分配算法进行对比, 我们对每个图中不满足生存期包含性的结点运行算法 3 进行生存期延长, 使之满足生存期包含性. 然后用基于超完美图的分配算法^[4]进行分配. 比较两种算法得到的着色数.

Algorithm 3 The algorithm for live range extension

```

1: for i=normal_node_num+1:total_node_num do
2:   for j = 1 : i - 1 do
3:     Check if i intersects with some j, if so, extend i to contain j
4:   end for
5: end for
6: end procedure

```

我们进行了 3 组实验, 总共测试了 1600 个相干图. 在第一组实验中, 我们令 MIN_OTHER_RATIO=1%, MAX_OTHER_RATIO=5%, 固定区间总宽度 RIGHT_END=150. 改变图的规模, 进行了 6 个分组的实验, 每个分组产生 100 个相干图, 总共 600 个相干图. 各分组中图的统计情况如表 1 所示, 其中的数据均为各分组 100 个图数据的平均值, “其他结点数”表示不满足生存期包含性的结点数. 我们实现了文献 [15] 中的置换图识别算法, 可以在线性时间内判断输入相干图是否是置换图. 实验得到的各

表 1 第一组实验的数据统计

Table 1 The statistics for the first group of IGs

Group number	Node count	Other node count	Edge count
1	29.08	1.56	90.45
2	58.00	2.57	335.79
3	78.78	3.35	631.57
4	98.01	3.80	980.02
5	130.33	5.19	1767.03
6	173.91	6.47	3002.68

图 7 第一组实验的结果

Figure 7 The experimental results of the first group of IGs

表 2 第二组实验的数据统计

Table 2 The statistics for the second group of IGs

Group number	Node count	Other node count	Edge count
1	29.08	1.56	90.45
2	55.55	2.44	234.30
3	77.43	3.20	368.35
4	104.25	4.11	547.02
5	130.80	5.07	754.60
6	145.10	5.35	846.49

个分组的结果如图 7, 其中 “#Permutation graph” 表示该分组的 100 个图中置换图的个数; “#Better allocation” 表示该分组的 100 个图中我们的算法获得的结果更优 (即着色数更小) 的图的个数. 从图中可见, 当图的规模不是特别大时, 即使有少量不满足生存期包含性的结点, 图仍有很大可能是置换图. 如第 1 分组的 100 个图, 平均结点数为 29.08(很多嵌入式应用的相干图为该规模), 此时置换图个数达到 98 个. 此外, 在该组实验的 600 个图中, 我们的算法在 190 个图 (32%) 上得到的分配结果优于基于超完美图的分配算法.

第一组实验是在固定程序总执行时间的情况下进行的, 即固定区间总大小. 此时增大图的规模, 意味着生存期更密集, 相干的可能性更大, 这从表 1 中边数的增长比结点数的增长快可以看出. 第二组实验在图的规模增大的同时, 增加区间总大小, 以尽量使相干图的“稀疏度”不变. 我们以第一组实验中的第一分组数据为基准, 进行了 5 个分组的实验, 产生 500 个相干图, 每个分组 100 个. 在增大图

图 8 第二组实验的结果

Figure 8 The experimental results of the second group of IGs

表 3 第三组实验的数据统计

Table 3 The experimental results of the third group of IGs

Group number	Node count	Other node count	Edge count
1	29.08	1.56	90.45
2	26.27	2.07	81.69
3	24.52	2.56	76.67
4	29.32	3.55	102.40
5	30.72	4.60	123.52
6	30.60	5.46	126.94

图 9 第三组实验的结果

Figure 9 The experimental results of the third group of IGs

的规模的同时,也增大总区间的大小.每个分组相对上一个分组,将总区间的大小增加 50.各分组实验的数据统计如表 2 所示,实验结果如图 8 所示.实验表明,“稀疏”的图有更大的可能为置换图.该组实验的 600 个图中,我们的算法在 207 个图 (34.5%) 上获得的分配结果优于基于超完美图的分配算法.

与前面两组实验不同,第三组实验在固定图的规模情况下,提高不满足生存期包含性的结点所占的比例,进行了 5 个分组共 500 个相关图的实验,仍然以第一组实验中的第一分组数据为比较基准,每个分组仍然产生 100 个图,各分组实验的数据统计如表 3,实验结果如图 9.该实验表明,当不满足生存期包含性的结点比例达到最高 30%,对于较小规模的图,仍有很多是置换图.该组实验的 600 个图中,我们的算法在 223 个图 (37%) 上得到的分配结果优于基于超完美图的分配算法.

以一个简单例子来说明为什么我们的算法优于基于超完美图的分配算法.如图 10(a) 所示,图中有 4 个生存期 A, B, C 和 D ,其对应的相关图如图 10(b) 所示.由于生存期 C 与 D 交叉,因此该相关图不满足生存期包含性.基于超完美图的分配算法需要进行生存期延长,有两种延长方法:延长 C 或延长 D .延长 C 后的生存期如图 10(c) 所示,其对应的相关图如图 10(d),该相关图满足生存期包含

图 10 一个生存期延长的例子

Figure 10 An example to explain the advantage of permutation graph-based algorithm. (a) Original live ranges: not satisfy live range containment; (b) original IG: the chromatic number is 2 (assume each array has unit size); (c) the live ranges after extending live range C ; (d) the IG after extending C : the chromatic number is 3 (assume each array has unit size); (e) the live ranges after extending live range D ; (f) the IG after extending D : the chromatic number is 3 (assume each array has unit size)

性. 对图 10(d) 运行基于超完美图的分配算法, 可以得到着色数为 3 (假设每个数组的大小为 1), 也就是说, 需要 SPM 空间至少为 3. 另一种方法是延长 D , 延长 D 后的生存期如图 10(e) 所示, 其对应的相干图如图 10(f), 该相干图也满足生存期包含性, 用基于超完美图的分配算法得到的着色数也为 3. 实际上, 对图 10(b) 运行置换图识别算法可以知道, 尽管该相干图不满足生存期包含性, 其仍然是置换图, 从而可以在线性时间内获得其最优着色为 2. 可见, 我们的算法得到的结果优于基于超完美图的分配算法.

7 结论

本文中, 我们证明了满足生存期包含性的数组相干图为置换图. 置换图具有线性时间的识别算法以及线性时间的最优区间着色算法. 基于此理论结果, 我们对文献 [4] 中提出的基于超完美图的 SPM 分配算法在保留原算法逻辑的基础上进行了改进, 使之基于置换图. 实验表明, 改进后的算法能够在很多不满足生存期包含性的相干图上获得最优着色, 获得比基于超完美图的算法更好的 SPM 分配结果.

参考文献

- 1 Banakar R, Steinke S, Lee B-S, et al. Scratchpad memory: design alternative for cache on-chip memory in embedded systems. In: CODES'02: Proceedings of the 10th International Symposium on Hardware/Software Codesign. Estes Park: ACM, 2002. 73–78
- 2 Golumbic M C. Algorithmic Graph Theory and Perfect Graphs. 2nd ed. Amsterdam: North-Holland Publishing Co., 2004. 105–148
- 3 Garey M R, Johnson D S. Computers and Intractability: A Guide to the Theory of NP-Completeness. New York: W. H. Freeman & Co., 1979. 190–205
- 4 Li L, Nguyen Q H, Xue J L. Scratchpad allocation for data aggregates in superperfect graphs. In: Proceedings of the 2007 ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems. San Diego: ACM, 2007. 207–216
- 5 Panda P R, Dutt N D, Nicolau A. Efficient utilization of scratch-pad memory in embedded processor applications. In: EDTC'97: Proceedings of the 1997 European Conference on Design and Test. Washington: IEEE Computer Society, 1997
- 6 Sjodin J, von Platen C. Storage allocation for embedded processors. In: CASES'01: Proceedings of the 2001 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems. Atlanta: ACM, 2001. 15–23

- 7 Avissar O, Barua R, Stewart D. An optimal memory allocation scheme for scratch-pad-based embedded systems. *ACM Trans Embed Comput Syst*, 2002, 1: 6–26
- 8 Hallnor E G, Reinhardt S K. A fully associative software-managed cache design. *SIGARCH Comput Archit New*, 2000, 28: 107–116
- 9 Moritz C A, Frank M, Amarasinghe S P. Flexcache: a framework for flexible compiler generated data caching. In: *IMS'00: Revised Papers from the 2nd International Workshop on Intelligent Memory Systems*. London: Springer-Verlag, 2001. 135–146
- 10 Kandemir M, Ramanujam J, Irwin J, et al. Dynamic management of scratch-pad memory space. In: *DAC'01: Proceedings of the 38th Conference on Design Automation*. Las Vegas: ACM, 2001. 690–695
- 11 Verma M, Wehmeyer L, Marwedel P. Dynamic overlay of scratchpad memory for energy minimization. In: *CODES+ISSS'04: Proceedings of the 2nd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*. Stockholm: ACM, 2004. 104–109
- 12 Udayakumaran S, Barua R. An integrated scratch-pad allocator for affine and non-affine code. In: *DATE'06: Proceedings of the Conference on Design, Automation and Test in Europe*. Munich: European Design and Automation Association, 2006. 925–930
- 13 Li L, Gao L, Xue J L. Memory coloring: a compiler approach for scratchpad memory management. In: *PACT'05: Proceedings of the 14th International Conference on Parallel Architectures and Compilation Techniques*. Washington: IEEE Computer Society, 2005. 329–338
- 14 Udayakumaran S, Barua R. Compiler-decided dynamic memory allocation for scratch-pad based embedded systems. In: *CASES'03: Proceedings of the 2003 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*. San Jose: ACM, 2003. 276–286
- 15 McConnell R M, Spinrad J P. Modular decomposition and transitive orientation. *Discrete Math*, 1999, 201: 189–241
- 16 McConnell R M, Spinrad J P. Linear-time transitive orientation. In: *SODA'97: Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*. New Orleans: ACM, 1997. 19–25

Scratchpad memory allocation for arrays in permutation graphs

WANG Li*, YANG XueJun & DAI HuaDong

School of Computer, National University of Defense Technology, Changsha 410073, China

*E-mail: liwang@nudt.edu.cn

Abstract In modern embedded systems, on-chip memory is generally organized as software-managed scratchpad memory (SPM). Li et al. discovered that in many embedded applications, the array interference graphs tend to be superperfect graphs. Based on this, they presented a superperfect graph-based SPM allocation algorithm, which is the best in the literature. In this paper, we make further progress in proving that the array interference graphs they focused on are permutation graphs; that is, a subclass of a superperfect graph. Compared to the latter, the permutation graph has certain advantages, such as linear time recognition and linear time optimal interval coloring. Based on our theoretical results, we improve their algorithm by transforming it into a permutation graph-based one by means of simple plug-in revisions. The experiments confirm that the improved algorithm achieves optimal coloring, and therefore, better allocation than the original superperfect graph-based algorithm for a broader scope of array interference graphs.

Keywords scratchpad memory, SPM allocation, interval coloring, superperfect graph, permutation graph

WANG Li was born in 1980. He received the Ph.D. degree in Computer Science and Technology from the National University of Defense Technology (NUDT), Changsha, in 2009. Currently, he is an assistant professor at NUDT. His research interests include operating system, memory management, and compiler optimization. He is a member of China Computer Federation.

DAI HuaDong was born in 1975. He received the Ph. D. degree in China from National University of Defence Technology, Changsha, in 2002. Currently, he is a professor at Computer School of National University of Defence Technology. His research interests include Operating System, Architecture and Parallel Processing. Dr. Dai Huadong is vice director of Domestic Infrastructure Software Research and Engineering Center of Ministry of Education of China.