轩瑞,陈磊,石海鹤. 图类算法可重用设计及其实现[J]. 江西师范大学学报(自然科学版),2023,47(1):52-60.
XUAN Rui,CHEN Lei,SHI Haihe. The reusable design and implementation of graph algorithms family[J]. Journal of Jiangxi Normal University(Natural Science),2023,47(1):52-60.

文章编号:1000-5862(2023)01-0052-09

图类算法可重用设计及其实现

轩 瑞,陈 磊,石海鹤*
(江西师范大学计算机信息工程学院,江西 南昌 330022)

摘要: 为了提高图算法生成效率和可靠性,该文提出一种将领域特征模型与构件组装技术相结合的可重用的图类算法开发方法. 首先,通过对一族图算法的深入分析,揭示出图类算法领域的共性特征和可变特征,建立领域特征模型;然后,分析特征之间的交互过程,设计图类算法的可重用构件,并对构件依赖关系做出描述;最后,借助高可靠平台对算法构件进行开发,建立高可靠可重用构件库,进一步由构件组装出多种图算法,提高了图算法的开发效率和可靠性. 实验表明开发出的图算法可重用构件库具有一定的实用性.

关键词: 图算法生成;特征模型;可重用设计;构件
中图分类号: TP 311 **文献标志码:** A **DOI:** 10.16357/j.cnki.issn1000-5862.2023.01.07

0 引言

图算法是利用节点之间的关系来推断复杂系统的结构和变化,是图分析的工具之一,也是分析相互关联数据的有效的方法之一[1]. 图算法在社交网络、城市交通网络、生物信息网络等网络领域中扮演着重要的角色[2-4],除此之外图算法在防止欺诈、优化呼叫路由以及预测流感的传播等领域中还存在巨大的应用潜力. 2010 年,美国航空旅行系统发生了 2 起涉及多个拥挤机场的严重事件,网络科学家使用图算法确认事件起因,并使用这些信息提出纠正建议[5];2018 年,斗鱼采用图算法来解决流量作弊问题[6]. 图算法还可以用来帮助理解相互关联的数据,从蛋白质相互作用到社交网络、从通信网络到电网、从零售体验到火星任务规划等都有图算法的应用实例[1].

面对图算法在各个领域中的广泛应用,众多学者从不同角度对其展开了研究[7-11],这些研究大多致力于特定问题求解及具体步骤优化,较少面向整个问题领域,且无法保证算法开发的可靠性. 当在一个领域中的信息量超过一定限度时,就会变得难以管理和使用[12].

本文提出了一种将特征模型与构件组装技术相结合的可重用的图类算法开发方法. 在面向重用的开发阶段,探索了图类算法领域的共性和可变性,建立了领域特征模型,并进一步基于高可靠平台设计和实现了图类算法的可重用构件库;在利用重用的开发阶段,通过若干构件组装生成了 Dijkstra 算法、Prim 算法和拓扑排序算法等,提高了图类算法的开发效率和可靠性.

1 相关研究工作

许多学者对图算法问题进行了研究,文献[10]利用 OpenCL 在高性能计算领域中的优势来进行加速计算大脑皮层曲率,实现了 Dijkstra 算法的并行编程. 文献[11]通过量化权重指标的选择,使用 Prim 算法求解公路网布局重要度最大树. 文献[13]在已有 BFS 算法优化的基础上提出了 2 种面向高通量计算机的优化算法以提高访存效率. 文献[14]提出了一种新的优化算法来寻找不同的拓扑排序. 针对在道路网中在线最短路径查询所面临的计算成本高、查询速度慢等问题,文献[15]提出了一种新

收稿日期:2022-10-25
基金项目:国家自然科学基金(62062039)和江西省自然科学基金(20212BAB202017)资助项目.
通信作者:石海鹤(1979—),女,江西乐平人,教授,博士,主要从事形式化方法、软件工程和生物信息学的研究. E-mail:haiheshi@jxnu.edu.cn

的基于缓存的时变最短路径查询算法. 以上研究主要关注算法层面的应用和优化,较少面向整个问题域.

在通用图算法框架研究方面,文献[16]建立了一个基于优势关系的广度优先搜索理论,并展示了如何使用该理论推导单源最短路径问题和最小生成树问题. 文献[17]采用程序转换步骤系统地推导图算法,从而揭示了算法的具体设计步骤及其基本原理,为从简单组件到创建算法提供了可靠的方法. 文献[18]研究了一个适合处理图算法的形式化语言代数,推导出了宽度优先遍历的一般方法,并表明该方法也可以应用于可达性和最短路径问题. 文献[19]提出了一种求解有向图路径问题的通用方法: 首先在图中构造一个表示路径集的正则表达式集合,然后根据正则表达式到给定问题域的自然映射给出了寻找最短路径所需的映射图; 该方法为求解任意路径问题提供了一种通用算法. 然而上述研究均没有提供相应的平台来验证所提出理论的可行性,且图算法的推导过程完全依赖于人工,推导效率较低. 文献[20]提出了一个用于最优路径查询的通用框架: 首先使用特定领域的语言来描述最佳路径查询问题,再由最优路径查询系统将其转换为 C++程序,进而查找出最优路径; 不过该框架仅研究了最优路径查询问题. 文献[21]开发了一个用于解决一般路径问题的通用算法程序,并实例化生成了一些典型的路径算法,如 Floyd 最短路径算法、最大容量路算法.

2　基础知识

2.1　PAR 方法

薛锦云提出的 PAR[21-26]方法是一种统一的算法设计方法,涵盖了多种已知的算法设计技术,如动态规划法、贪心法、分治法、穷举法等,支持算法程序开发的全过程. PAR 方法由自定义算法设计语言 Radl(recurrence-based algorithm design language)、抽象程序设计语言 Apla(abstract programming language)和算法程序开发方法及程序生成系统(可生成 Java、C++等可执行程序)组成.

Radl 是基于递推关系的算法建模语言,是为描述算法规范、推导算法的转换规则和算法本身而设计的. 它提供了标准数据类型、自定义简单类型、预定义 ADT 和自定义 ADT 机制. 用户可以像使用标准数据类型一样使用预定义的 ADT,其常常用作 Radl→Apla 转换器的前端语言.

Apla 是基于功能抽象、数据抽象的抽象编程语言,其主要目的是描述抽象程序. 在 Apla 中的关键字、标识符、标准过程、标准函数、符号和类型系统与 Radl 的表达方式完全一致. 不同之处在于: Apla 增加了程序结构、语句过程和函数方面的内容,并且拥有敏捷的泛型机制,支持类型参数化、过程参数化、函数参数化. 这些优点使得 Apla 非常适合于形式化实现构件库.

使用 PAR 方法开发高可靠的执行程序有 2 种途径(见图 1).

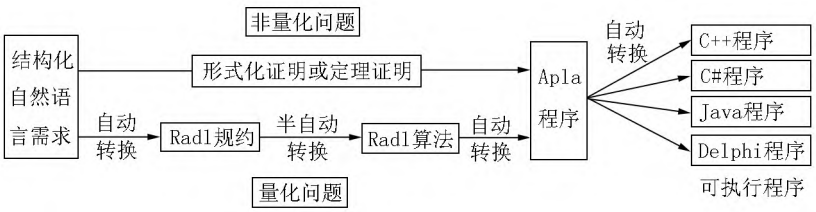

图 1　PAR 架构

2.2　特征模型

特征模型描述了在领域内实例的通用特征和可变特征以及它们之间的约束与依赖关系,是在特征建模过程中构建的,一般由描述特征层次关系的特征图和相关的文本描述组成. 特征建模是领域工程方法的核心活动,它清晰地刻画了特征模型在构建过程中所需的各类特征,是识别和捕捉可变性的关键技术. 文献[27]提出了一种面向特征的领域建模方法 FODM,它将在领域中的服务(service)、功能(function)、行为特点(behavior characteristic)作为重点分析对象,本文使用此方法进行建模工作.

特征建模阶段包含了特征选取、特征建立、特征关系描述以及特征模型建立等过程,在该阶段需要对被研究领域进行分析,充分了解其功能性特征,并找出特征之间的约束和依赖关系以及优先级等附加信息,进而获得该领域的主次特征,建立一种具有更高抽象层次的特征模型或者构件框架.

特征建模阶段主要包含 4 种特征[28]: 强制特征(mandatory)、可变特征(optional)、XOR 特征、OR 特征. 图 2(a)表示强制特征,被末端为实心圆的边所指向,强制特征表示在领域中的所有实例都必须包含的特征; 图 2(b)表示可变特征,被末端为空心圆的边所指向,可变特征表示领域中实例的一个可选

特征;图 2(c)表示 XOR 特征,被一组用空心弧连接的边所指向,XOR 特征表示在领域中的实例有且只能选择一个特征;图 2(d)表示 OR 特征,被一组用实心弧连接的边所指向,OR 特征表示在领域中的实例至少包含了在该特征中的一个特征.

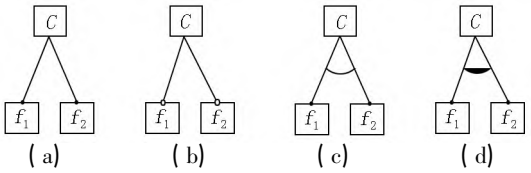

图 2 特征图

上述 4 种特征是特征建模中的主要属性,也是特征模型的重要组成部分,代表了在领域中的通用和可变属性,同时它们之间的约束以及依赖关系则需通过特征内在的层次结构关系、领域业务逻辑设计的约束关系以及在运行时的依赖关系来表示.

3 面向图领域的可重用构件设计

图是由顶点集合与边(顶点之间的关系)集合组成的数据结构,是刻画离散结构的一种有力工具.在运筹规划、网络研究和计算机程序流程分析中都存在图的实际应用.在生活中通常使用图来表达用文字难以描述的信息,如城市交通图、线路图、网络图等.

本部分通过提取不同图算法之间的共性和特性,将所有图的遍历问题概括并统一为最佳优先遍历问题,设计出了可重用的算法构件,并使用 PAR 方法和高可靠 PAR 平台对可重用构件进行形式化开发.

3.1 一族图算法问题分析

图算法是个庞大的家族,其中大部分成员的主体框架都可归结于图的遍历.图的遍历算法作为图算法的基础,应用十分广泛,一些比较经典的图算法都是在遍历算法的基础上加以改进的,如 Prim 算法、Dijkstra 算法以及拓扑排序算法等,将这类图算法的遍历过程定义为如下形式.

给定一个图 G,起点为 v_1,找到一组满足性质 P 的顶点集合 X,即

$$\text{Specification } X = X \uparrow [v_1]\ P \uparrow [S \uparrow G.\text{out}(v_1)].$$

其中称运算符 $\uparrow$ 为插入运算符,其定义为

$$[a_0,a_1,\cdots,a_n] \uparrow a = [a_0,a_1,\cdots,a_n,a];$$

S 被定义为栈,表示等待被访问的顶点集;$G.\text{out}(v)$ 表示顶点 v 射出边的终点集;属性 P 表示顶点的某种选取方式,它被定义为

$$\begin{cases} P[S] = [], & S = [], \\ P[S] = [S.v] \uparrow P[S[t+1..]] \uparrow G.\text{out}(S,v)], & S \neq [], \end{cases}$$

$S.v$ 表示在栈 S 中前 v 个顶点集,t 表示栈尾.

根据上述定义,使用 P 来描述一些图算法的问题属性,如使用 P_{BFS}、P_{Prim}、$P_{\text{Topological sorting}}$ 分别表示 BFS、Prim 和拓扑排序问题的属性,它们的描述如下:P_{BFS} 为优先访问在图中距离起始顶点最近的点;P_{Prim} 为优先选取与当前顶点集最近的一个顶点;$P_{\text{Topological sorting}}$ 为优先选取在图中入度为 0 的顶点.

根据属性描述可以发现当上述 3 种图算法在图中遍历遇到满足问题属性的新顶点时,其做法有着某种共性特征,这一共性特征可被总结为最佳优先遍历(best first traversal,BFT)框架.为了更直观地描述该特征,将顶点的优先级定义为该类问题的新顶点选取指标,根据每个问题的属性设计出相应的优先级更新方法(priority update method,PUM),随着算法的推进,PUM 不断调整各个顶点的优先级值.BFT 框架的具体搜索过程如下:首先,从某个节点 v_0 开始搜索,处理完该节点本身的信息;然后,算法通过与该点相连接的所有边,遍历到它的所有邻接点,同时使用相应的 PUM 更新这些顶点的优先级值,在其后的每一步迭代中都选取当前优先级最高的顶点,并将该点与其父节点的联边加入遍历树中;最后,重复上述过程,直至所有顶点都被访问.

图 3 为该框架的算法流程图,其中 priority 函数用来存储顶点优先级值,visited 为布尔型数组,用来保存顶点状态.这里顶点的 priority 值越大说明其优先级越低,初始阶段将所有的顶点的 priority 值统一设置为最大.

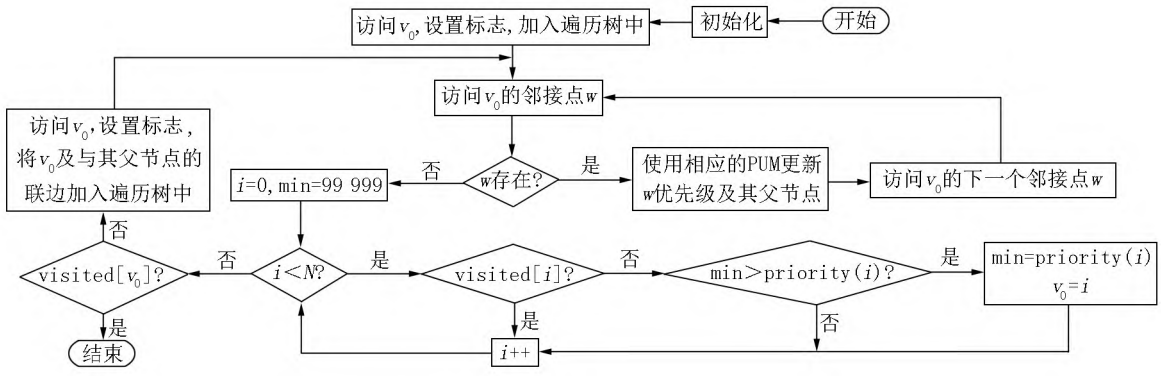

图 3 BFT 框架流程图

下面以求解单源最短路径的 Dijkstra 算法为例,深入介绍这一框架的具体应用.

设 G 为一个有 n 个顶点的无向非负权图,V 表示在图 G 中所有顶点集合,E 为在图 G 中所有边的集合,则 $G=(V,E)$,令 $V=\{v_1,v_2,\cdots,v_n\}$,(v_1,v_2) 表示顶点 v_1 和顶点 v_2 之间的边. G 的最短路径树(shortest path tree,SPT)表示为 $T=(U;T_E)$,其中 U 为 T 的顶点集,T_E 为 T 的边集,U 和 T_E 的初始值为空,$u\in V-U$. Dijkstra 算法的基本思想是逐步求解,即每次从顶点集 V 中选出一个距离起点 v_0 最近的顶点 v_t 加入 U 中,并将其与父节点之间的边加入 T_E 中,依次重复,直到所有的顶点都并入集合 U 中,由此得到最短路径树 T.

在这个问题中,将 $v_i(1\leqslant i\leqslant n)$ 与源点 s 之间的距离作为各顶点的优先级数; 每次将一个新的顶点 v_i 加入 U 后,更新在集合 $V-U$ 之中且与 U 之间有关联的顶点的优先级数,即在每次扩充 U 时,可将集合 $V-U$ 中的各个顶点 u 到 v_i 的距离看作 u 的优先级数(若 u 到 v_i 之间没有联边,则优先级设为 $+\infty$); 从而每条最短路径所对应的 v_i 都会因拥有最高优先级而被选中. 图 4 给出了 Dijkstra 算法的完整执行过程.

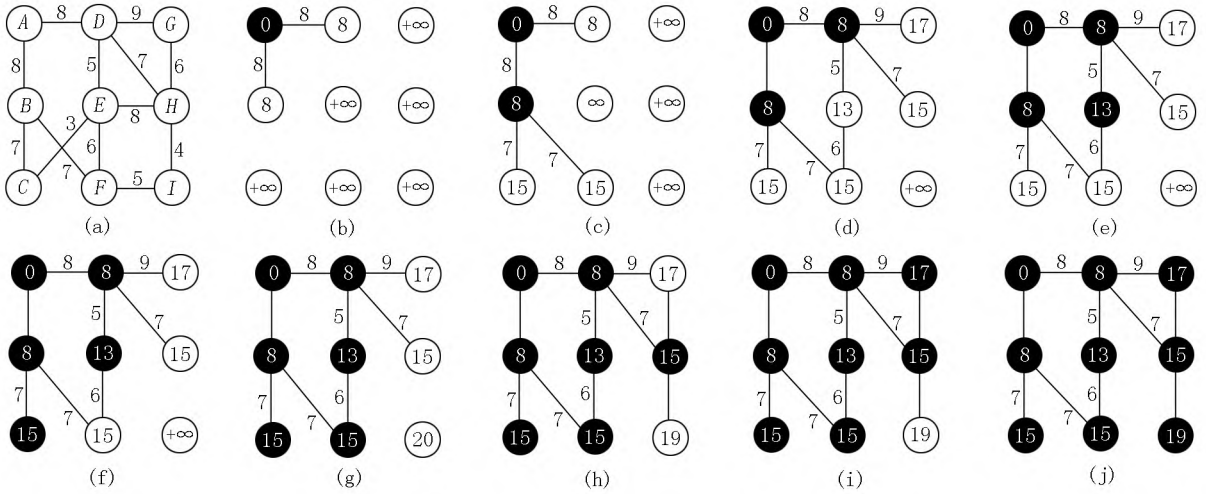

图 4　Dijkstra 算法示例

图 4(a)为一个含有 9 个顶点和 13 条边的无向图. 先任选一个顶点 A 作为源点(见图 4(b)),$T=(\{A\};\varnothing)$; 再从 A 到其顶点的边中选择一条权值最小的边(A,B)(见图 4(c)),扩充 T,$T=(\{A,B\};\{(A,B)\})$,此时能够到达 B 的点有 D、B、C 和 F; 然后从 A 到这 4 个点的边中选取一条权值最小的边(A,D)(见如图 4(d)),$T=(\{A,B,D\};\{(A,B),(A,D)\})$; 如此反复,直至最后如图 4(j)所示,得到最短路径树 T:

$T=(\{A,B,D,E,C,F,H,G,I\};\{(A,B),(A,D),(D,E),(B,C),(B,F),(D,H),(D,G),(H,I)\})$.

3.2　BFT 特征模型

基于对上述一族图算法问题的分析,利用 FODM 建模方法对在 BFT 领域中的服务 S(service)、功能 F(function)和行为特点 B(behavior characteristic)进行特征建模. 根据以上分析,动态更新顶点优先级操作是该领域问题的可变特征,而遍历顶点、选择顶点、输出等操作是该领域的共性特征. 通过分析在 BFT 中的一些主要执行步骤,可以将其进一步划分为初始化图信息、获取用户需求、选择优先级更新方法、算法组装和结果输出等功能,其中顶点初始化、动态更新优先级方法选择、构件组装为必选功能. 在上述分析的基础上,顶点优先级的更新方式是该领域的显著行为特点,本文主要选取了 Prim、DFS、Topological Sorting、BFS 和 Dijkstra 等 5 种行为特点. BFT 的特征模型图如图 5 所示.

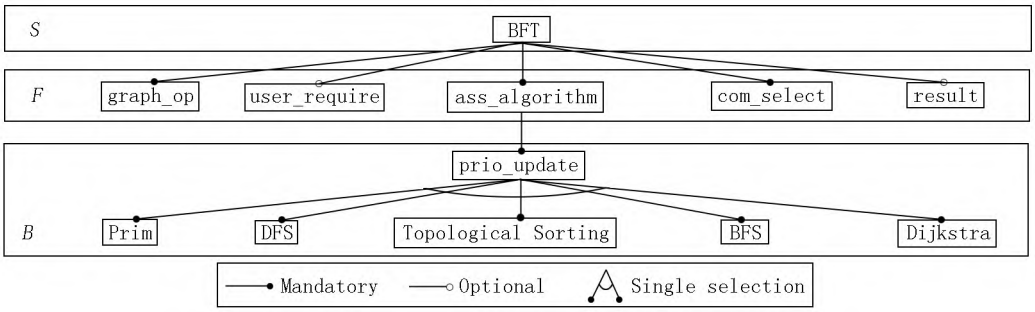

图 5　BFT 框架特征模型图

3.3 图构件形式化设计和实现

通过上述对一族图算法的问题分析以及对 BFT 框架的特征建模,本节把在 BFT 特征模型中的功能和特征抽象作为构件,对特征之间的约束以及依赖关系进行设计分析,建立算法构件之间的依赖关系图.

根据以上分析,将在 BFT 特征模型中的 4 个主要算法功能作为主要构件,其他特征作为辅助构件. 如图 6 所示,图中实线连接的节点为在 BFT 领域中必须的基本构件,实线箭头表示执行顺序,即操作构件(graph_op)、根据用户需求选择优先级更新方法构件(com_select)、算法组装构件(ass_algorithm)和结果输出构件(result)为主要构件;虚线箭头代表在算法执行过程中所需的数据、结构以及关联操作等,如在算法装配时,ass_algorithm 构件需要使用 graph_op 构件和 prio_update 构件提供的操作和数据来进行算法组装;点划线箭头表示在算法执行过程中 2 个构件之间的交互关系,如在 com_select 构件中,需要根据用户需求选择相应的 prio_update 构件中的顶点更新方式.

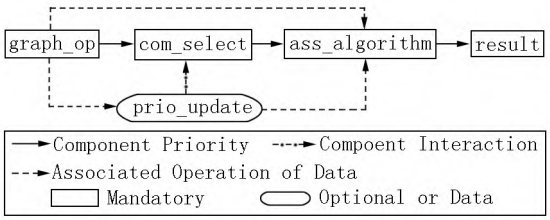

图 6 算法构件依赖图

基于 Apla 的敏捷泛型机制对得到的构件进行形式化实现,进而提高算法的抽象性. 首先使用 Radl 对算法构件进行功能规约的描述,对其进行形式化推导;然后根据构件之间的依赖关系,将同一数据依赖源的构件封装为一个 ADT,并使用 Apla 语言形式化实现所有的 ADT. 限于篇幅,这里仅给出了 shortest_path 构件的功能规约:

| [n n: integer;

c [0: $n-1$] [0: $n-1$]: array [array [integer]];

out dist [0: $n-1$] [0: $n-1$]: array [array [integer]] |

AQ: $n \geqslant 0 \wedge c \neq$ I

AR: ($\forall i,j: 0 \leqslant i < n, 0 \leqslant j < n$: dist [$i$] [$j$] ≡ (MIN $p: p \in$ path [i,j]: cost(p))),

cost(p) ≡ ($\sum p: p \in$ path [i,j]: c [i] [j]). 其中 in 和 out 分别表示输入、输出变量;AQ 和 AR 分别表示前置断言和后置断言;c 是一个 2 维数组,用来保存图的信息;dist 数组刻画了终止状态,为最终输出结果;在后置断言中 path(i,j) 表示顶点 i 和顶点 j 之间的所有路径;cost(p) 使用了求和量词 $\sum$ 来表示顶点 i 和顶点 j 之间各个路径的权值和.

根据 BFT 框架的特征模型以及构件之间的关系,使用 Apla 语言实现该领域构件. 下面给出了主要的泛型算法构件定义以及参数的具体释义.

1) graph_op 类型构件. 该构件被定义为一个 ADT,内部包含了一些图算法的基本操作,如判断图类型、查找邻接点、设置权重、记录顶点入度、顶点出度、记录顶点父节点和设置边的状态等.

```
define ADT graph_op (sometype elem);
  type graph_op = private;
  function create( ): graph_op;
  function judge( G: graph_op): String;
  function nextNbr( i: elem): elem;
  function weight( i: elem; j: elem): integer;
  function priority( i: elem): integer;
  function indegree( i: elem): integer;
  function outdegree( i: elem): integer;
  function parent( i: elem): elem;
  function Etype( i: elem; j: elem): String;
  procedure status( i: elem);
  …
enddef;
```

其中 sometype 为 Apla 中的关键字,表示类型变量;type 用于说明 ADT 的访问控制,这里将它设为私有方式.

2) prio_update 构件. 该构件被定义为一个泛型 ADT,内部包含一系列泛型子程序,用来实现不同的优先级更新方式. 在每个泛型子程序中内部都包含 1 个 graph_op 类型的变量 op 和 2 个 elem 类型的变量 k 和 v.

```
define ADT prio_update (sometype elem);
  typeprio_update = private;
  procedure MST (op: graph_op; k,v: elem);
  procedure shortest_path (op: graph_op; k,v: elem);
  procedure tra_BFS (op: graph_op; k,v: elem);
  procedure tra_DFS (op: graph_op; k,v: elem);
  procedure topsort (op: graph_op; k,v: elem);
enddef.
```

为了更详细地说明顶点优先级的更新方式,这里列出了 prio_update 的实现部分. 以泛型子程序

MST 为例,对它的执行过程做以下说明: 若顶点 v 未被访问,则它的优先级就被置为 priority(v) 和(weight(k,v)) 之间的最小值.

```
implement ADT prio_update( sometype elem) ;
  type prio_update = integer;
  procedure MST ( op: graph_op; k; v: elem) ;
begin
  if( ¬ op. status( v) ) →op. priority( v) : = min( op.
priority( v) ,op. weight( k,v) ) ;
  op. parent( v) : = k; fi;
end;
  procedure shortest_path ( op: graph_op,k; v:
elem) ;
begin
  if( ¬ op. status( v) ) →op. priority( v) : = min( op.
priority( v) ,op. priorit( k) + op. weight( k,v) ) ; op.
parent( v) : = k; fi;
end;
  procedure tra_BFS ( op: graph_op,k,v: elem) ;
begin
  if( ¬ op. status( v) ) →op. priority( v) : = min( op.
priority( v) ,op. priority( k) + 1) ; op. parent( v) : =
k; fi;
end;
  procedure tra_DFS ( op: graph_op;
  k,v: elem) ;
begin
  if( ¬ op. status( v) ) →op. priority( v) : = min( op.
priority( v) , op. priority( k) - 1) ;
  op. parent( v) : = k; return; fi;
end;
procedure topsort ( op: graph_op; k,v: elem) ;
begin
  if( op. indegree( k) = 0) →op. indegree( v) --;
if( op. indegree( v) = 0) → op. priority( v) : =
min ( op. priority( v) ,op. priority( k) + 1) ;
op. parent( v) : = k; fi; fi;
end;
endimp.
```

3) com_select 构件. 该构件通过用户需求来指定相应的顶点更新策略. 构件内部设置最短路径、深度优先搜索、广度优先搜索、拓扑排序等选项,结合图类型和用户需求返回相应的顶点更新方式. 该构件原型如下:

```
function com_select ( req: String;
pu: prio_update) : prio_update.
```

4) ass_algorithm 算法构件. ass_algorithm 构件为在 BFT 框架下的通用算法构件,根据 com_select 返回的顶点更新方式对算法进行组装,进而生成指定的算法程序.

```
procedure ass_algorithm ( sometype elem;
somefunc com_select ( req: String;
pu: prio_update) : prio_update;
var opt: graph_op; s: integer) .
```

5) result 构件. 该构件被定义为一个泛型子程序,根据在 com_select 构件中所选顶点的更新方式来输出采用的遍历算法,同时输出遍历树.

```
procedure result( someproc
ass_algorithm( sometype elem;
somefunc com_select( req: String; pu: prio_update) :
prio_update; var opt: graph_op; s: integer) ) .
```

4　利用重用的图算法生成

本部分包括特定图算法生成和实验 2 个部分. 在图算法构件实现后,借助 PAR 平台将 Apla 构件进行组装并转换,自动生成了 Prim、Dijkstra、DFS、BFS、Topological Sorting 等算法. 生成的算法符合预期,验证了本文提出的图算法可重用开发方法的有效性.

4.1　特定图算法的生成

在得到 Apla 描述的构件后,使用 Apla 提供的泛型实例化机制组合这些构件,自动生成图领域的问题求解算法程序. 组装构件生成相应图算法的装配代码如下:

```
program BFT;
const MIN = 99 999; n = 9;
  /* input user's requirement* /
var
  s,w,i, j: integer;
  userReq: String; // 存储用户需求
  G: array [0…n, [array [0…n,integer]] ];
// 存储用户输入的图信息
procedure ass_algorithm ( sometype elem;
someproc pri_manner ( op: graph_op; k,v: elem) ;
s: integer) ; //①
var w,i: integer;
begin
```

```
        op.priority(s) := 0;
        op.status(s) := ture;
        op.parent(s) := -1;
        w := op.firstNbr(s);
        do(i < n) → if(¬ op.status(v)) → pri_manner(s,w); w := op.nextNbr(s); fi; od; i := 0;
        do(n > i) → if(¬ op.status(i)) → if(MIN > op.priority(i)) → min := op.priority(i); s := i; op.path(i); op.parent(i) := s; fi; fi; i++; od;
        if(op.status(s)) → break; fi;
        op.status(s) := true;
        type(op.parent(s),s) := TREE;
    end;
    sometype = {integer,char}; //②
    ADT
        operation: new graph_op(integer);
        priority: new prio_update(integer);
    var
        optn: operation;
        pu: priority;
    procedure select: new com_select(pu,optn);
    procedure rel: new result(select);
    procedure Prim: new ass_algorithm(integer,pu.MST); //③
    procedure Dijkstra: new ass_algorithm(integer,pu.shortest_path); //④
    procedure DFS: new ass_algorithm(integer,pu.tra_DFS); //⑤
    procedure BFS: new ass_algorithm(integer,pu.tra_BFS); //⑥
    procedure Topological sorting:
        new ass_algorithm(integer,pu.topsort); //⑦
    begin
        C := optn.create();
// 默认自动生成一个随机的无向带权图.
        i,j := 0;
        do(j < n) → if(¬ optn.status(i)) →
        Prim(C,0); //⑧
        fi; j++;
        od;
        write(rel()); //⑨
    end.
```

在以上程序中,根据用户需求可以生成相应的遍历算法,步骤 ③、④、⑤、⑥ 和 ⑦ 分别表示根据不同的用户需求实例化而成的遍历算法;过程 ① 为代码装配算法,它指明了在上述步骤实例化时对象内部要执行的关联操作;步骤 ② 表示 sometype 的实例化类型参数,为提高安全性,对类型参数设置了 region 安全机制,在实例化过程中 sometype 只能是 char 或 integer 型,否则会报错;步骤 ⑧ 表示在装配后的算法执行过程,这里仅执行了在步骤 ③ 实例化后的 Prim 算法,将 0 置为起点;步骤 ⑨ 输出最终结果.

4.2 实验

图 7 是随机生成的包含 9 个顶点、13 条边的测试图. 图 8 是选择不同构件进行组装生成的遍历算法的简要实验结果. 如图 8(a) 所示,该次实验使用 DFS 算法策略逐一更新其邻接点的优先级,选取优先级最高的顶点以及其与父节点的联边加入遍历树中,依次循环直至所有顶点均已加入. 组装生成的图算法得到的遍历结果与经典图算法得到的结果可能会存在不同之处,其原因在于:组装生成图算法的算法框架是通过抽取各种图算法的共同特征和可变特征所构成的. 虽然与经典图算法思想有所差别,但是它们所表现出的遍历过程大致相同,且能正确处理的图形类型也一致.

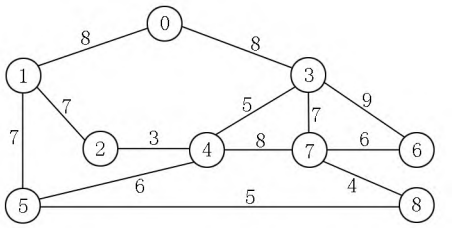

图 7 测试图

实现功能:深度优先遍历
使用 DFS 算法得到遍历树 T = ({顶点集};{边集})
T = ({0,1,2,4,3,6,7,8,5}; {(0,1),(1,2),(2,4),(4,3),(3,6),(6,7),(7,8),(8,5)})

(a)

实现功能:最小生成树
使用 Prim 算法得到遍历树 T = ({顶点集};{边集})
T = ({0,1,2,4,3,5,8,7,6}; {(0,1),(1,2),(2,4),(4,3),(4,5),(5,8),(8,7),(7,6)})

(b)

实现功能:单源最短路径
使用 Dijkstra 算法得到遍历树 T = ({顶点集};{边集})
T = ({0,1,3,4,2,5,7,6,8}; {(0,1),(0,3),(3,4),(1,2),(1,5),(3,7),(3,6),(7,8)})

(c)

图 8 实验结果

实验结果表明: 在底层可重用构件和 PAR 平台的支持下,采用本文提出的图类算法开发方法,可以高效生成可靠的特定图算法程序. 这验证了本文所提出方法的实用性. 由于构件是使用 PAR 平台实现的,构件组装是在 PAR 平台的支撑下完成的,所以生成结果的可靠性得到了有效保证.

5　总结

图算法是计算机科学中的一类特殊算法,兼具复杂性和多样性. 现有研究主要集中于某一特定问题的求解及具体步骤的优化,具有较高的专用性,较少从算法领域层次开展研究,使得开发过程存在可重用性较低、冗余性过高等问题.

本文通过分析图算法领域之间的结构关系,揭示出算法之间的共性和可变性,提高了算法开发的抽象性,使得领域算法信息变得有序. 同时根据特征模型开发出的图算法构件库,不仅能高抽象表示算法程序的功能特性,而且能直观地展示各算法构件之间的约束以及依赖关系. 这样既降低了算法程序的复杂性,又提高了构件装配产生算法的可复用性和可维护性,减少了形式化开发代价. 用户在使用该算法构件库时只需关注算法功能,而不用在意具体构件实现细节,从而提高了算法生成效率.

概括而言,本文的主要贡献包括以下 3 个方面:

1) 提出了一种结合特征模型与构件组装技术的图算法开发方法. 从领域问题形式化规约出发,通过领域分析找出一族图算法的共性特征,建立起特征依赖和特征模型,从而设计出领域可重用构件.

2) 形式化实现了一组高可靠的算法构件. 借助抽象泛型程序设计语言 Apla 对图类可重用构件进行形式化实现,得到了一组高可靠可重用的图领域构件. 利用 PAR 平台对构件进行组装并生成了 Prim、Dijkstra、拓扑排序等特定算法,显著提高了图算法开发效率和可靠性.

3) 在高可靠 PAR 平台的支持下,经可重用构件组装生成特定的求解算法,为其他领域算法的高可靠共性开发提供有价值的借鉴. 目前软件设计、开发与验证越来越需要形式化方法,而形式化建模与验证门槛较高,通常需要形式化专家来开展这项工作,投入较大、成本较高. 对于内在复杂的算法问题,可将开发结果、验证结论等做成验证库,有望降低复杂问题的验证代价.

下一步,将扩充本文所提出的方法在图算法领域中的应用范围,探索更优的通用算法框架.

6　参考文献

[1] NEEDHAM M, HODLER A E. Graph algorithms: practical examples in Apache Spark and Neo4j [M]. Sevastopol: O´Reilly Media Incorporated, 2019.

[2] HAVELIWALA T H. Topic-sensitive pagerank: a context-sensitive ranking algorithm for web search [J]. IEEE Transactions on Knowledge and Data Engineering, 2003, 15(4): 784-796.

[3] MISLOVE A, MARCON M, GUMMADI K P, et al. Measurements and analysis of online social networks [EB/OL]. [2022-08-11]. https://doi.org.10.1145/1298306.1298311.

[4] 王兆慧,沈华伟,曹婧,等. 图分类研究综述 [J]. 软件学报, 2022, 33(1): 171-192.

[5] FLEURQUIN P. Systemic propagation of delays in the air-transportation network [EB/OL]. [2022-08-19]. http://digital.csic.es/bitstream/10261/134152/1/air-transportation_network_Fleurquin.pdf.

[6] DATAFUNTALK. 图算法在斗鱼反作弊中的实践 [EB/OL]. [2021-07-08]. https://www.infoq.cn/article/NhX7CKcIbT3cVmY08Nrr.

[7] VILLENEUVE D, DESAULNIERS G. The shortest path problem with forbidden paths [J]. European Journal of Operational Research, 2005, 165(1): 97-107.

[8] MAGZHAN K, JANI H M. A review and evaluations of shortest path algorithms [J]. International Journal of Scientific & Technology Research, 2013, 2(6): 99-104.

[9] 张天明,徐一恒,蔡鑫伟,等. 时态图最短路径查询方法 [J]. 计算机研究与发展, 2022, 59(2): 362-375.

[10] 杨保国. 基于 OpenCL 的最短路径图算法实现 [J]. 实验科学与技术, 2017, 15(1): 57-59, 76.

[11] 王佳敏,吴海燕. 基于 Prim 算法的农村公路布局重要度最大树的求解与应用 [J]. 公路, 2019, 64(6): 162-166.

[12] CLEOPHAS L, KOURIE D G, PIETERSE V, et al. Correctness-by-construction ∧ taxonomies⇒deep comprehension of algorithm families [EB/OL]. [2022-09-13]. https://wur.on.worldcat.org/discovery.

[13] 张承龙,曹华伟,王国波,等. 面向高通量计算机的图算法优化技术 [J]. 计算机研究与发展, 2020, 57(6): 1152-1163.

[14] BEIRANVAND A, CUFFE P. A topological sorting approach to identify coherent cut-sets within power grids [J]. IEEE Transactions on Power Systems, 2020, 35(1):

721-730.

[15] 黄阳,周旭,杨志邦,等. 基于缓存的时变道路网最短路径查询算法[J]. 计算机研究与发展,2022,59(2):376-389.

[16] NEDUNURI S,COOK W R,SMITH D R. Theory and techniques for synthesizing a family of graph algrithms [EB/OL]. [2022-06-17]. https://arxiv.org/abs/1207.0869v1.

[17] REIF J H,SCHERLIS W L. Deriving efficient graph algorithms [M]//DERSHOWITZ N. Verification: theory and practice. Berlin: Springer-Verlog,2003: 645-681.

[18] RUSSLING M. A general scheme for breadth-first graph traversal [M]//MÖLLER B. Mathematics of program construction. Berlin: Springer-Verlog,1995: 380-398.

[19] TARJAN R E. A unified approach to path problems [J]. Journal of the Association for Computing Machinery,1981,28(3):577-593.

[20] MORIHATA A,MATSUZAKI K,TAKEICHI M. Write it recursively: a generic framework for optimal path quires [C]//ICFP'08: Proceedings of the 13th ACM SIGPLAN International Conference on Functional Programming,September 20-28,2008,Victoria BC Canada. New York: Association for Computing Machinery,2008: 169-178.

[21] XUE Jinyun. Developing the generic path algorithmic program and its instantiations using PAR method [EB/OL]. [2022-08-17]. https://www.researchgate.net/publication/285841454_PAR_method_and_its_supporting_platform.

[22] XUE Jinyun. PAR method and its supporting platform [EB/OL]. [2022-08-17]. https://www.researchgate.net/publication/285841454_PAR_method_and_its_supporting_platform.

[23] 谢武平,薛锦云. Radl 算法到 Apla 程序的生成系统[J]. 计算机研究与发展,2014,51(4):856-864.

[24] XUE Jinyun. Two new strategies for developing loop invariants and their applications [J]. Journal of Computer Science and Technology,1993,8(2):147-154.

[25] 石海鹤,薛锦云. 基于 PAR 的排序算法自动生成研究[J]. 软件学报,2012,23(9):2248-2260.

[26] 薛锦云. 程序设计方法 [M]. 北京: 高等教育出版社,2001.

[27] 张伟,梅宏. 一种面向特征的领域模型及其建模过程[J]. 软件学报,2003,14(8):1345-1356.

[28] CZARNECKI K,EISENECKER U W. Generative programming: methods,tools,and applications [M]. Boston: Addision Wesley Publishing Co.,2000.

The Reusable Design and Implementation of Graph Algorithms Family

XUAN Rui,CHEN Lei,SHI Haihe*

(School of Computer and Information Engineering,Jiangxi Normal University,Nanchang Jiangxi 330022,China)

Abstract: In order to improve the efficiency and reliability of graph algorithms generation,the reusable development method of graph algorithms that combines domain feature model and component assembly technology is proposed. First,a family of graph algorithms are analyzed to reveal the common features and variable features in the domain of graph algorithms and a domain feature model is established. Then,the interaction process between features is analyzed,the reusable components of graph algorithms are descigned and the component dependencies are described. Finally,the algorithm components with the help of PAR platform are developed,a library of highly reliable reusable components is established,and further the various graph algorithms are assembled by the components,the development efficiency and reliability of these graph algorithms are significantly improved. The experiments show that the developed reusable component library of graph algorithms has certain practicality.

Key words: graph algorithms generation; feature model; reusable design; component

(责任编辑: 冉小晓)