计算机科学
COMPUTER SCIENCE

基于顶点粒 k 步搜索和粗糙集的强连通分量挖掘算法

程富豪，徐泰华，陈建军，宋晶晶，杨习贝

引用本文

程富豪，徐泰华，陈建军，宋晶晶，杨习贝. 基于顶点粒 k 步搜索和粗糙集的强连通分量挖掘算法[J]. 计算机科学，2022, 49(8): 97-107.

CHENG Fu-hao, XU Tai-hua, CHEN Jian-jun, SONG Jing-jing, YANG Xi-bei. Strongly Connected Components Mining Algorithm Based on k-step Search of Vertex Granule and Rough Set Theory[J]. Computer Science, 2022, 49(8): 97-107.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210700202

基于顶点粒 k 步搜索和粗糙集的强连通分量挖掘算法

程富豪[1]　徐泰华[1]　陈建军[1]　宋晶晶[1,2]　杨习贝[1]
1 江苏科技大学计算机学院　江苏 镇江 212000
2 数据科学与智能应用福建省高校重点实验室　福建 漳州 363000
(cfh_vip@163.com)

摘　要　强连通分量挖掘是图论中的经典问题之一,如何设计更高效率的串行强连通分量挖掘算法具有现实需求。GRSCC 算法利用 k 步上近似和 k 步 R 相关集这两个粗糙集算子所构成的 SUB-RSCC 函数,可实现简单有向图中的强连通分量挖掘,而 SUB-RSCC 函数的调用次数决定了挖掘效率。根据挖掘强连通分量时顶点间存在的相关性,GRSCC 算法引入了粒化策略,减少了 SUB-RSCC 函数的调用次数,提高了挖掘效率。在 GRSCC 算法的基础上,分析发现了顶点间的另外两种强连通分量相关性,由此设计了一种新的顶点粒化策略,进而提出了一种顶点粒 k 步搜索方法,可更大程度地减少 SUB-RSCC 函数的调用次数。最后,提出了一种基于顶点粒 k 步搜索和粗糙集的强连通分量挖掘算法 KGRSCC。实验结果表明,相比 RSCC 算法、GRSCC 算法和 Tarjan 算法,KGRSCC 算法具有更好的性能。

关键词:强连通分量;粗糙集;图论;粒化策略;顶点粒 k 步搜索
中图法分类号　TP181

Strongly Connected Components Mining Algorithm Based on k-step Search of Vertex Granule and Rough Set Theory

CHENG Fu-hao[1],XU Tai-hua[1],CHEN Jian-jun[1],SONG Jing-jing[1,2] and YANG Xi-bei[1]
1 School of Computer,Jiangsu University of Science and Technology,Zhenjiang,Jiangsu 212000,China
2 Key Laboratory of Data Science and Intelligent Application,Fujian Province University,Zhangzhou,Fujian 363000,China

Abstract　Strong connected components (SCCs) mining is one of the classic problems in graph theory. It has practical requirements to design a serial SCCs mining algorithm with high efficiency. GRSCC algorithm can use SUB-RSCC function to discover SCCs of simple digraphs. SUB-RSCC function is formed by two operators of rough set theory (RST),k-step upper approximation set and k-step R-related,which are the main contributors to time consumption. Then the invocation times of SUB-RSCC decide the efficiency of GRSCC algorithm. Based on the SCCs correlations among vertices,GRSCC algorithm introduces granulation strategy to reduce the invocation times of SUB-RSCC function,then improve the mining efficiency. Two new SCCs correlations are found by analysis of SCCs in the framework of RST,then a new vertex granulation strategy is designed to granulate the vertex set of target digraphs. In order to reduce the invocation times of SUB-RSCC function to a greater extent,a method called k-step search of vertex granule is proposed. Finally,combining with GRSCC algorithm,an algorithm called KGRSCC for mining SCCs based on k-step search of vertex granule and RST is proposed. Experimental results show that,compared with RSCC,GRSCC and Tarjan algorithms,the proposed KGRSCC algorithm has better performance.

Keywords　Strongly connected components,Rough set,Graph theory,Granulation strategy,k-step search of vertex granule

1　引言

强连通分量是图论中的一种经典概念,其中的任意两个顶点既互为对方的祖先,又互为对方的后代。特别地,包含一个顶点(至少两个顶点)的强连通分量被称为无价值(有价值)强连通分量。强连通分量涉及很多研究领域,包括程序安全

到稿日期:2021-07-20　返修日期:2021-10-23
基金项目:国家自然科学基金(62006099,62076111,61906078);江苏省高等学校自然科学基金(20KJB520010);镇江市重点研发计划——社会发展(SH2018005)
This work was supported by the National Natural Science Foundation of China(62006099,62076111,61906078),Natural Science Foundation of Jiangsu Provincial Colleges and Universities(20KJB520010) and Key Research and Development Plan of Zhenjiang City—Social Development (SH2018005).
通信作者:徐泰华(xth19890410@163.com)

分析[1-2]、个性化推荐[3]、小世界网络[4]、传递闭包[5]等。作为图论中的一种经典问题，已有许多挖掘强连通分量的串行算法[6-10]和并行算法[11-15]被提出，这些算法多是基于宽度优先搜索(Breadth-First Search，BFS)或深度优先搜索(Depth-First Search，DFS)两种图搜索策略设计的。相比而言，并行算法对硬件的现实需求较高，不如串行算法应用简便，因此设计高效率的串行算法仍然具有现实需求。

粗糙集[16]作为一种处理不确定、不完备信息的有效工具，已被广泛运用在数据挖掘[17-18]、知识发现[19]、机器学习[20-21]、决策分析[22-24]等领域。文献[25]提出了一种基于粗糙集模型的串行强连通分量挖掘算法(Algorithm for Finding SCCs of Simple Digraphs Based on Generalized RST，RSCC)。与 Tarjan 算法[8](最为常用的串行强连通分量挖掘算法)相比，RSCC 算法可提速 10.4 倍之多。RSCC 的算法核心为 SUB-RSCC 函数，对于调用顶点集 T 中的每个顶点，调用 SUB-RSCC 函数即可得到全部有价值强连通分量。SUB-RSCC 函数的主要思想是：针对目标顶点 x，利用两个粗糙集算子(k 步上近似和 k 步 R 相关集)计算出 x 的 k 步上近似的并集和 k 步 R 相关集的并集这两个集合，二者的交集即为 x 对应的强连通分量，同时将该强连通分量中的顶点从调用顶点集 T 中删除。SUB-RSCC 函数的被调用次数决定了 RSCC 算法的效率。但是，经过分析发现，若顶点 x 的上近似集或 R 相关集为空，则针对 x 调用 SUB-RSCC 函数仅能得到一个无价值强连通分量。因此，RSCC 算法还使用了快速剔除策略，即将此类顶点从 SUB-RSCC 函数的调用顶点集 T 中快速剔除，避免浪费计算资源。

基于 RSCC 算法，文献[26]提出了一种基于粗糙集和粒化策略的强连通分量挖掘算法(The Algorithm for Computing SCCs of Simple Digraphs Based on RST and Granulation Strategy，GRSCC)。与 RSCC 算法相比，GRSCC 算法可提速 5.5 倍之多。文献[26]经过分析得到了 4 种顶点间的有价值强连通分量相关性，若顶点 x 的上近似集(或 R 相关集)只包含一个顶点 y：1)当 x 属于一个有价值强连通分量时，y 必然属于该有价值强连通分量；2)当 y 属于一个有价值强连通分量时，x 必然不会属于另一个有价值强连通分量。若针对情况 1)中的 y 和情况 2)中的 x 调用 SUB-RSCC 函数，不仅不能计算出新的有价值强连通分量，反而会造成计算资源的浪费。基于以上分析，GRSCC 算法不仅使用了快速剔除策略，还根据上述分析得到的顶点间的相关性，设计了一种粒化策略，将满足相关性的顶点 x 和 y 互相加入对方的顶点粒中，优化了 SUB-RSCC 函数的调用方式，即当针对 T 中顶点调用 SUB-RSCC 函数计算得到一个有价值强连通分量时，需同时从调用顶点集 T 中删除有价值强连通分量中的顶点，以及每个顶点所对应顶点粒中的顶点。这种调用方式可提高 GRSCC 算法的效率。

GRSCC 算法将粒化策略与 RSCC 算法结合，减少了 SUB-RSCC 函数的调用次数，提高了强连通分量的挖掘效率。然而，GRSCC 算法仅考虑了顶点间存在的有价值强连通分量关联，却忽略了顶点间存在的无价值强连通分量关联，本文对强连通分量再次进行深入分析后，得到两种顶点间的无价值强连通分量相关性，即若顶点 x 的上近似集(或 R 相关集)只包含一个顶点 y，当 y 单独构成一个无价值强连通分量时，x 必然也单独构成一个无价值强连通分量。然后，结合文献[26]中的情况 2)所描述的有价值强连通分量相关性，本文设计了一种新的顶点集粒化策略，对于每个顶点 y，若顶点 x 与 y 满足 4 种强连通分量相关性，则将 x 加入 y 对应的顶点粒。粒化策略执行结束后，会获得每个顶点对应的顶点粒，然后利用顶点粒来优化 SUB-RSCC 被调用的方式。

GRSCC 算法对顶点粒的利用方式是：从调用顶点集 T 中删除有价值强连通分量中的顶点后，删除了每个顶点对应顶点粒中的顶点。与 GRSCC 算法中的顶点粒相比，本文得到的顶点粒虽然是其子集，但是由于其粒化策略不同，顶点粒的利用方式也会不同。每个顶点 y 都有对应的顶点粒，而顶点粒中的顶点也有对应的顶点粒，依次类推可遍历到很多顶点。1)若 y 属于一个有价值强连通分量，根据有价值强连通分量的相关性，y 对应的顶点粒中的每个顶点 x 必然不会属于一个新的有价值强连通分量，进而每个 x 对应的顶点粒中的顶点也不会属于一个新的有价值强连通分量。依次类推，从一个顶点 y 出发，可以引导出一系列顶点，而这些顶点都必然不属于另一个新的有价值强连通分量，即不需要针对它们浪费计算资源来调用 SUB-RSCC 函数。2)若 y 属于一个无价值强连通分量，根据无价值强连通分量的相关性，y 对应的顶点粒中的每个顶点 x 必然也是一个无价值强连通分量，进而每个 x 对应的顶点粒中的顶点必然也是一个无价值强连通分量。与 1)类似，一个顶点 y 可以引导出一系列必然属于无价值强连通分量的顶点，自然也不需要调用 SUB-RSCC 函数。

基于上述分析，本文定义了一个顶点粒 k 步搜索(k-step Search of Vertex Granule，KSVG)函数来引导出一系列不需调用 SUB-RSCC 函数的顶点，同时提出了一种顶点标记方法来保证 KSVG 函数的收敛性，对 KSVG 函数遍历到的未被标记的顶点加以标记。后续计算中，若是遍历到已被标记的顶点，则不再遍历该顶点所对应的顶点粒，当遍历到的顶点都被标记，KSVG 函数终止。无论是利用快速剔除策略得到的所有无价值强连通分量，还是调用 SUB-RSCC 函数计算出某个有价值强连通分量或者无价值强连通分量，都可以利用 KSVG 函数计算获得一个对应的不需调用 SUB-RSCC 函数的顶点集，然后从调用顶点集 T 中删除该集合。

总的来说，本文根据分析得到的强连通分量相关性，设计了一种新的顶点粒化策略，进而提出了一种顶点粒 k 步搜索(KSVG)函数，使得粒化所得顶点粒可以得到更大程度的利用。最后，本文提出了一种基于顶点粒 k 步搜索和粗糙集的强连通分量挖掘算法(Algorithm for Finding SCCs of Simple Digraphs Based on k-step Search of Vertex Granule and Rough Set Theory，KGRSCC)。

本文第 2 节介绍了强连通分量和粗糙集的一些基础知识；第 3 节讨论了 GRSCC 算法及其不足；第 4 节提出了一种基于顶点粒 k 步搜索和粗糙集的强连通分量挖掘算法 KGRSCC；第 5 节对本文提出的 KGRSCC 方法进行了实验测试和算法比较；最后总结全文。

2　基本知识

2.1　强连通分量

图在计算机科学领域起着重要作用，许多应用问题都可以用图来描述。根据顶点之间的边是有向的还是无向的，图可分为有向图和无向图，本文聚焦于有向图。

定义 1[27]　一个有向图可以用 $D=\{U,E\}$ 表示，其中 U 表示顶点集，E 表示有向边集。U 的大小记为 $|U|$ 或者 n，E 的边数记为 $|E|$ 或者 m。

自环是一条始点与终点相同的有向边。若一个有向图既不包含自环，也不包含多重有向边，则称之为简单有向图。本文提到的有向图都是简单有向图。

定义 2[8]　$D=\{U,E\}$ 是一个有向图，$\forall x,y\in U$，若从顶点 x 到顶点 y 存在一个顶点与边的交替序列，则该序列被称为 x 到 y 的通路，记做 $p:x\xrightarrow{*}y$。其中 $\xrightarrow{*}$ 是 U 上的一个二元关系，称为可达关系。x 称为 y 的祖先，y 称为 x 的后代。特别地，每个顶点都是自身的祖先和后代。

强连通性是有向图的基本性质之一，是图论中重要的研究课题。在实际应用中，强连通图起着重要作用，讨论强连通图的结构特征也一直是图论研究的前沿课题之一。

定义 3[25]　$D=\{U,E\}$ 是一个有向图，$\forall x,y\in U$，都存在两条通路 $p_1:x\xrightarrow{*}y$ 和 $p_2:y\xrightarrow{*}x$，则称 D 是强连通图。

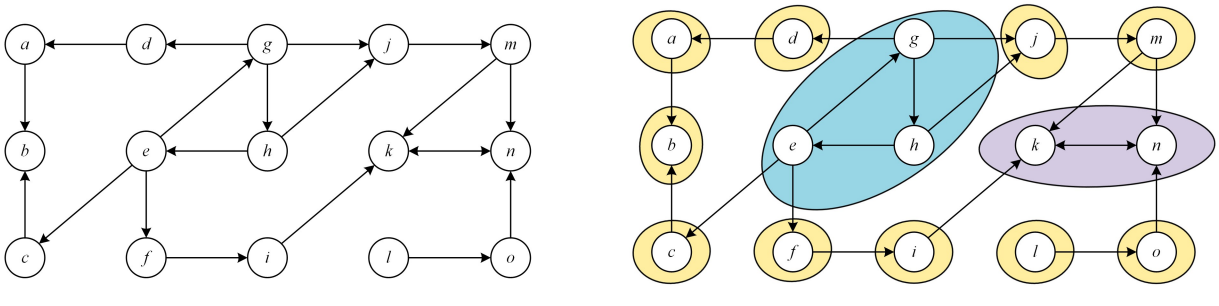
换言之，若 D 中任意两个顶点之间都是互相可达的，即 $x\xrightarrow{*}y$ 且 $y\xrightarrow{*}x$，则该图是强连通的。若有向图不是强连通的，其极大强连通子图称为强连通分量。

定义 4[8]　设 $D=\{U,E\}$ 是一个有向图，可以在 U 上定义一种等价关系：$\forall x,y\in U$，若 $x\xrightarrow{*}y$ 且 $y\xrightarrow{*}x$，则称 x 和 y 是等价的。这种等价关系对 U 进行划分会形成不同等价类：U_i，$1\leqslant i\leqslant|U|$。令 $D=\{U_i,E_i\}$，其中 $E_i=\{(x,y)\in E\mid x,y\in U_i\}$。满足以下描述的子图 D_i 就是 D 中的强连通分量：

(1)每个 D_i 都是强连通的；

(2)任何 D_i 都不是另一个属于 D 的强连通子图的真子图。

例 1　设 $D=\{U,E\}$ 是一个有向图，如图 1(a)所示。其中，$U=\{a,b,c,d,e,f,g,h,i,j,k,l,m,n,o\}$，$E=\{(a,b),(c,b),(d,a),(e,c),(e,f),(e,g),(f,i),(g,d),(g,h),(g,j),(h,e),(h,j),(i,k),(j,m),(k,n),(l,o),(m,k),(m,n),(n,k),(o,n)\}$。$D$ 中的强连通分量分别为 $D_{\{a\}}$，$D_{\{b\}}$，$D_{\{c\}}$，$D_{\{d\}}$，$D_{\{e,g,h\}}$，$D_{\{f\}}$，$D_{\{i\}}$，$D_{\{j\}}$，$D_{\{k,n\}}$，$D_{\{l\}}$，$D_{\{m\}}$，$D_{\{o\}}$。如图 1(b)所示，相应地，D 中强连通分量的数量为 12，其中有价值强连通分量(蓝色区域、紫色区域)的数量为 2，无价值强连通分量(黄色顶点)的数量为 10。



(a)有向图 D　　(b)D 中的 12 个强连通分量

图 1　有向图 D 及 D 中的 12 个强连通分量(电子版为彩图)

Fig. 1　Digraph D and 12 SCCs in D

2.2　粗糙集

粗糙集是一种处理不确定、不完备信息的有效数学工具，它用两个确定性的概念即上近似集和下近似集去描述不确定信息，而二元关系[28-30]在构建上下近似集时起着至关重要的作用。

定义 5[25]　设 U 是一个非空有限集合，$U\times U$ 为 U 和 U 的笛卡尔乘积，任一 $U\times U$ 的子集 R 被称为 U 上的一个二元关系，若 $(x,y)\in R$，则称 x 和 y 有关系 R，记作 xRy。

设 R 是论域 U 上的任意一个二元关系，则 R 可以是：

(1)串行的，若 $\forall x\in U,\exists y\in U$ 使得 xRy；

(2)自反的，若 $\forall x\in U$，都有 xRx；

(3)反自反的，若 $\forall x\in U$，都有 $(x,x)\notin R$；

(4)对称的，若 $\forall x\in U,xRy$ 蕴涵 yRx；

(5)传递的，若 $\forall x\in U,xRy$ 和 yRz 蕴涵 xRz。

在文献[28]中，Yao 用 (U,R) 表示一个广义近似空间，其中 R 是论域 U 上的任意二元关系。对于 U 中的任一元素 x，Yao 定义了 x 的 R 相关集，记作 $r(x)$，其中包含了与 x 有关系 R 的所有元素：

$$r(x)=\{y\mid xRy\}\tag{1}$$

经典粗糙集中的等价关系满足自反性、对称性和传递性。为了将粗糙集应用到更广的领域，国内外众多学者对粗糙集进行了改进和扩充。例如，Chen 等[31]提出了一种基于 SP 关系的广义粗糙集模型，将粗糙集应用于无向图领域；Xu 等[25]提出了一种基于 IR 关系的广义粗糙集模型，将粗糙集应用于有向图领域。

定义 6[31]　设 R 是论域 U 上的二元关系，若 R 是串行的，且是反自反的、对称的，则称 R 是 SP 关系，由 R 引导出的近似空间被称为 SP 近似空间。

定义 7[25]　设 R 是论域 U 上的二元关系，若 R 是反自反的，则称 R 是 IR 关系，由 R 引导出的近似空间被称为 IR 近似空间。

文献[25]针对 IR 近似空间中的 R 相关集、下近似集、上近似集等概念给出了如下定义。

定义 8[25]　设 (U,R) 是一个 IR 近似空间，$\forall x\in U$ 和 $\forall X\subseteq U$：

$$r(X)=\bigcup\{r(x):x\in X\} \tag{2}$$

$$\underline{R}X=\{x\in U\mid \varnothing\neq r(x)\subseteq X\} \tag{3}$$

$$\overline{R}X=\{x\in U\mid r(x)\cap X\neq\varnothing\} \tag{4}$$

为了利用粗糙集方法挖掘子图结构，在 SP 近似空间和 IR 近似空间下，Xu 等[25]和 Chen 等[31]分别定义了 k 步上近似和 k 步 R 相关集这两种粗糙集算子。

定义 9[31] 设(U,R)是一个 SP 近似空间，k 是一个正整数，$\forall X\subseteq U$：

$$\overline{R}^{(1)}X=\overline{R}X \tag{5}$$

$$\overline{R}^{(k)}X=\underbrace{\overline{R}(\overline{R}(\cdots(\overline{R}X)))}_{k\ times} \tag{6}$$

定义 10[25] 设(U,R)是一个 IR 近似空间，k 是一个正整数，$\forall X\subseteq U$：

$$r^{(1)}(X)=r(X) \tag{7}$$

$$r^{(k)}(X)=\underbrace{r(r(\cdots(r(X))))}_{k\ times} \tag{8}$$

进一步，文献[25]定义了 k 步上近似的并集 $\mathscr{A}(X)$和 k 步 R 相关集的并集 $\mathscr{D}(X)$：

$$\mathscr{A}(X)=\bigcup\{\overline{R}^{(k)}X:k\geqslant 1\} \tag{9}$$

$$\mathscr{D}(X)=\bigcup\{r^{(k)}(X):k\geqslant 1\} \tag{10}$$

简单起见，若 X 为单元素集合，即 $X=\{x\}$，则把 $\mathscr{A}(X)$，$\mathscr{D}(X)$简写为 $\mathscr{A}(x)$，$\mathscr{D}(x)$。

利用 k 步上近似和 k 步 R 相关集这两种粗糙集算子来挖掘强连通分量这种图结构，其理论基础是粗糙集和图论之间的对应关系[25]，如图 2 所示。此外，还需在粗糙集理论框架中对强连通分量进行数学描述以及性质分析。

粗糙集		图论
$\overline{R}\{x\}$	=	前继顶点集
$r(x)$	=	后继顶点集
$\mathscr{A}(x)$	=	祖先顶点集
$\mathscr{D}(x)$	=	后代顶点集

图 2 粗糙集和图论的对应关系[25]

Fig. 2 Relationship between RST and graph theory[25]

2.3 强连通分量的粗糙描述及分析

根据粗糙集和图论之间的对应关系，对于强连通分量这种子图结构，文献[25]给出了粗糙集理论中的等价描述。

定理 1[25] 设(U,R)是一个 IR 近似空间，$D=\{U,E\}$是对应的有向图，$\forall x\in U$，若 $\overline{R}\{x\}=\varnothing$ 或 $r(x)=\varnothing$，则 $D_{\{x\}}$ 为 D 中的无价值强连通分量。

定理 2[25] 设(U,R)是一个 IR 近似空间，$D=\{U,E\}$是对应的有向图，$\forall x\in U$，令 $K_x=\mathscr{A}(x)\cap\mathscr{D}(x)$和 $U_x=\{x\}\cup K_x$，则 D_{U_x} 为 D 中的强连通分量。

定理 3[25] 设(U,R)是一个 IR 近似空间，$D=\{U,E\}$是对应的有向图，$\forall x\in U$，若 $\mathscr{A}(x)\cap\mathscr{D}(x)=\varnothing$，则 $D_{\{x\}}$ 为 D 中的无价值强连通分量。

文献[26]利用上近似集和 R 相关集这两个粗糙集概念，对强连通分量进行了如下分析。

定理 4[26] 设(U,R)是一个 IR 近似空间，$D=\{U,E\}$是对应的有向图，$\forall x\in U$，若 $\overline{R}\{x\}=\{y\}$，x 和 y 同属于一个有价值强连通分量当且仅当 x 属于一个有价值强连通分量。

定理 5[26] 设(U,R)是一个 IR 近似空间，$D=\{U,E\}$是对应的有向图，$\forall x\in U$，若 $r(x)=\{y\}$，x 和 y 同属于一个有价值强连通分量当且仅当 x 属于一个有价值强连通分量。

3 GRSCC 算法分析

强连通分量是有向图中的重要子图结构，文献[26]提出了一种基于粒化策略和粗糙集的强连通分量挖掘算法 GRSCC，用于从一个简单有向图中发现其中的有价值强连通分量，如算法 1 所示。

算法 1 GRSCC

输入：有向图 D={U,E}

输出：D 中的有价值强连通分量

```
1. SCCset←Ø; T←U; granu←Ø;
2. for x∈U do
3.    if |r(x)|=1 then
4.       add x into granu(r(x));
5.    add r(x) into granu(x);
6.    end if
7.    if |R̄{x}|=1 then
8.       add x into granu(R̄{x});
9.       add R̄{x} into granu(x);
10.   end if
11.   if |r(x)|=Ø or |R̄{x}|=Ø then
12.      T←T\{x};
13.   end if
14. end for
15. while T≠Ø do
16.   SUB-RSCC(T(1));
17. end while
18. output SCCset;
19. function SUB-RSCC(x)
20.   Compute Ux=x∪{𝒜(x)∩𝒟(x)};
21.   if 𝒜(x)∩𝒟(x)=Ø then
22.      T←T\{x}; return;
23.   end if
24.   SCCset←{SCCset, Ux};
25.   T←T\Ux;
26.   T←T\ ⋃ granu(y);
                y∈Ux
27. end function
```

3.1 算法描述

GRSCC 算法中，根据定理 2，设计了其算法核心 SUB-RSCC 函数(见算法 1 中的第 19－27 行)，其作用就是求解 k 步上近似的并集和 k 步 R 相关集的并集这两个集合的交集。调用一次 SUB-RSCC 函数最多需 $2(n+m)$次运算，其中 n 是顶点数，m 是边数。

GRSCC 算法中顶点调用 SUB-RSCC 函数的次数决定了强连通分量的挖掘效率。而 SUB-RSCC 函数如何被调用则是根据算法中所设置的调用顶点集 T(初始化状态为目标有向图中的所有顶点)。针对 T 中的每个顶点 x，调用 SUB-RSCC 函数就可得到包含 x 的强连通分量。若该强连通分量为有价值的，那么其中必然还包含除 x 之外的其他顶点，若针对这些顶点来调用 SUB-RSCC 函数，则不可能得到新的有价值

强连通分量，对计算资源造成了浪费，因此需要将这些顶点也从 T 中删除（见算法 1 中的第 25 行）。

根据定理 1，可以快速剔除 T 中一部分必然会单独构成无价值强连通分量的顶点（见算法 1 中的第 11－13 行）。此外，根据定理 4、定理 5 得到的两种有价值强连通分量相关性，以及定理 4、定理 5 分析得到的另外两种有价值强连通分量的相关性（本文给出了证明，见定理 6、定理 7），满足相关性的顶点 x 和 y 最多引导出一种有价值强连通分量。因此，若已得知 x 或 y 中任一顶点所属的有价值强连通分量，则针对另一顶点调用 SUB-RSCC 函数都不会得到一个新的有价值强连通分量。根据这 4 种有价值强连通分量的相关性，GRSCC 算法使用了一种粒化策略（见算法 1 中的第 3－10 行），对目标有向图的顶点集进行粒化，确定某个顶点属于有价值强连通分量后，将该顶点对应顶点粒中的顶点从 T 中一并删除（见算法 1 中的第 26 行），来降低 SUB-RSCC 函数被调用的次数。

定理 6 设 (U,R) 是一个 IR 近似空间，$D=\{U,E\}$ 是对应的有向图，$\forall x\in U$，若 $\overline{R}\{x\}=\{y\}$，当 y 属于一个有价值强连通分量时，x 和 y 属于同一个有价值强连通分量，或者单独构成一个无价值强连通分量。

证明：由 $\overline{R}\{x\}=\{y\}$，可知 $x\neq y$，$\mathscr{A}(x)=\{y\}\cup\mathscr{A}(y)$，$x\in\mathscr{D}(y)$。当 y 属于一个有价值强连通分量时，不妨令该有价值强连通分量为 U_y，则 $\mathscr{A}(y)\cap\mathscr{D}(y)=U_y$，$y\in\mathscr{A}(y)$，那么 $\mathscr{A}(x)=\mathscr{A}(y)$。若 $x\in U_y$，显然 x 和 y 属于同一个有价值强连通分量；若 $x\notin U_y$，由于 $x\in\mathscr{D}(y)$，则 $x\notin\mathscr{A}(y)$，即 $x\notin\mathscr{A}(x)$，那么对于 $\forall z\in\mathscr{A}(x)$，显然有 $z\notin\mathscr{D}(x)$，故 $A(x)\cap\mathscr{D}(x)=\varnothing$，$x$ 单独构成一个无价值强连通分量。综上所述，定理得证。

定理 7 设 (U,R) 是一个 IR 近似空间，$D=\{U,E\}$ 是对应的有向图，$\forall x\in U$，若 $r(x)=\{y\}$，当 y 属于一个有价值强连通分量时，x 和 y 属于同一个有价值强连通分量，或者单独构成一个无价值强连通分量。

证明：由 $r(x)=\{y\}$，可知 $x\neq y$，$\mathscr{D}(x)=\{y\}\cup\mathscr{D}(y)$，$x\in\mathscr{A}(y)$。当 y 属于一个有价值强连通分量时，不妨令该有价值强连通分量为 U_y，则 $\mathscr{A}(y)\cap\mathscr{D}(y)=U_y$，$y\in\mathscr{D}(y)$，那么 $\mathscr{D}(x)=\mathscr{D}(y)$。若 $x\in U_y$，显然 x 和 y 属于同一个有价值强连通分量；若 $x\notin U_y$，由于 $x\in\mathscr{A}(y)$，则 $x\notin\mathscr{D}(y)$，即 $x\notin\mathscr{D}(x)$，那么对于 $\forall z\in\mathscr{D}(x)$，显然有 $z\notin\mathscr{A}(x)$，故 $\mathscr{A}(x)\cap\mathscr{D}(x)=\varnothing$，$x$ 单独构成一个无价值强连通分量。综上所述，定理得证。

例 2（续例 1） 利用 GRSCC 算法求解有向图 D 中的有价值强连通分量：令 T 为需要调用 SUB-RSCC 的顶点集，初始化 $T=\{a,b,c,d,e,f,g,h,i,j,k,l,m,n,o\}$。首先，根据定理 4－定理 7 驱动粒化过程构建顶点粒，其中每个顶点对应的顶点粒如表 1 所列。

表 1 GRSCC 算法中每个顶点对应的顶点粒

Table 1 Vertex granules corresponding to each vertex in GRSCC

顶点	a	b	c	d	e
顶点粒	$\{b,d\}$	$\{a,c\}$	$\{b,e\}$	$\{a,g\}$	$\{c,f,g,h\}$
顶点	f	g	h	i	j
顶点粒	$\{e,i\}$	$\{d,e,h\}$	$\{e,g\}$	$\{f,k\}$	$\{m\}$
顶点	k	l	m	n	o
顶点粒	$\{i,n\}$	$\{o\}$	$\{j\}$	$\{k,o\}$	$\{l,n\}$

根据定理 1，$r(b)=\varnothing$，$\overline{R}\{l\}=\varnothing$，可知 $D_{\{b\}}$ 和 $D_{\{l\}}$ 是两个无价值强连通分量，从 T 中删除顶点 b 和 l，得 $T=\{a,c,d,e,f,g,h,i,j,k,m,n,o\}$。

$T\neq\varnothing$，针对 T 中第一个顶点 a 调用 SUB-RSCC 函数，$U_a=a\cup\{\mathscr{A}(a)\cap\mathscr{D}(a)\}=\{a\}$，则 $D_{\{a\}}$ 是一个无价值强连通分量，从 T 中删除 U_a，得 $T=\{c,d,e,f,g,h,i,j,k,m,n,o\}$。

$T\neq\varnothing$，针对 T 中第一个顶点 c 调用 SUB-RSCC 函数，$U_c=c\cup\{\mathscr{A}(c)\cap\mathscr{D}(c)\}=\{c\}$，则 $D_{\{c\}}$ 是一个无价值强连通分量，从 T 中删除 U_b，得 $T=\{d,e,f,g,h,i,j,k,m,n,o\}$。

$T\neq\varnothing$，针对 T 中第一个顶点 d 调用 SUB-RSCC 函数，$U_d=d\cup\{\mathscr{A}(d)\cap\mathscr{D}(d)\}=\{d\}$，则 $D_{\{d\}}$ 是一个无价值强连通分量，从 T 中删除 U_d，得 $T=\{e,f,g,h,i,j,k,m,n,o\}$。

$T\neq\varnothing$，针对 T 中第一个顶点 e 调用 SUB-RSCC 函数，$U_e=e\cup\{\mathscr{A}(e)\cap\mathscr{D}(e)\}=\{e,g,h\}$，则 $D_{\{e,g,h\}}$ 是一个有价值强连通分量，从 T 中删除 U_e 以及 U_e 中每个顶点对应顶点粒中的顶点，得 $T=\{i,j,k,m,n,o\}$。

$T\neq\varnothing$，针对 T 中第一个顶点 i 调用 SUB-RSCC 函数，$U_i=i\cup\{\mathscr{A}(i)\cap\mathscr{D}(i)\}=\{i\}$，则 $D_{\{i\}}$ 是一个无价值强连通分量，从 T 中删除 U_i，得 $T=\{j,k,m,n,o\}$。

$T\neq\varnothing$，针对 T 中第一个顶点 j 调用 SUB-RSCC 函数，$U_j=j\cup\{\mathscr{A}(j)\cap\mathscr{D}(j)\}=\{j\}$，则 $D_{\{j\}}$ 是一个无价值强连通分量，从 T 中删除 U_j，得 $T=\{k,m,n,o\}$。

$T\neq\varnothing$，针对 T 中第一个顶点 k 调用 SUB-RSCC 函数，$U_k=k\cup\{\mathscr{A}(k)\cap\mathscr{D}(k)\}=\{k,n\}$，则 $D_{\{k,n\}}$ 是一个有价值强连通分量，从 T 中删除 U_k 以及 U_k 中每个顶点对应顶点粒中的顶点，得 $T=\{m\}$。

$T\neq\varnothing$，针对 T 中第一个顶点 m 调用 SUB-RSCC 函数，$U_m=m\cup\{\mathscr{A}(m)\cap\mathscr{D}(m)\}=\{m\}$，则 $D_{\{m\}}$ 是一个无价值强连通分量，从 T 中删除 U_m，得 $T=\varnothing$，算法结束。

经统计发现，GRSCC 算法挖掘出了有向图 D 中的全部 12 个强连通分量，其中 2 个有价值强连通分量为 $D_{\{e,g,h\}}$ 和 $D_{\{k,l\}}$，一共调用了 8 次 SUB-RSCC 函数。

3.2 局限性分析

根据定理 4－定理 7 所描述的 4 种有价值强连通分量相关性，GRSCC 算法将粒化策略引入到强连通分量的挖掘中，降低了 SUB-RSCC 函数被调用的次数，进而提高了算法的挖掘效率。在定理 4－定理 7 的基础上，利用上近似集和 R 相关集，对强连通分量再次进行深入分析后，得到两种无价值强连通分量相关性，见定理 8、定理 9。

定理 8 设 (U,R) 是一个 IR 近似空间，$D=\{U,E\}$ 是对应的有向图，$\forall x\in U$，若 $\overline{R}\{x\}=\{y\}$，当 y 单独构成一个无价值强连通分量时，x 必然也单独构成一个无价值强连通分量。

证明：由 $\overline{R}\{x\}=\{y\}$，可知 $x\neq y$，$\mathscr{A}(x)=\{y\}\cup\mathscr{A}(y)$，$\mathscr{D}(x)\subseteq\mathscr{D}(y)$。当 y 单独构成一个无价值强连通分量时，因 $y\in\mathscr{A}(x)$，故 $y\notin\mathscr{D}(x)$，根据定理 3，可知 $\mathscr{A}(y)\cap\mathscr{D}(y)=\varnothing$，因 $\mathscr{D}(x)\subseteq\mathscr{D}(y)$，故 $\mathscr{A}(y)\cap\mathscr{D}(x)=\varnothing$，则 $\mathscr{A}(x)\cap\mathscr{D}(x)=(\{y\}\cup\mathscr{A}(y))\cap\mathscr{D}(x)=(\{y\}\cap\mathscr{D}(x))\cup(\mathscr{A}(y)\cap\mathscr{D}(x))=\varnothing$。因此，$x$ 单独构成一个无价值强连通分量，定理得证。

定理 9 设 (U,R) 是一个 IR 近似空间，$D=\{U,E\}$ 是对应的有向图，$\forall x\in U$，若 $r(x)=\{y\}$，当 y 单独构成一个无价值

强连通分量时，x 必然也单独构成一个无价值强连通分量。

证明：由 $r(x)=\{y\}$，可知 $x\neq y$，$\mathcal{D}(x)=\{y\}\cup\mathcal{D}(y)$，$\mathcal{A}(x)\subseteq\mathcal{A}(y)$。当 y 单独构成一个无价值强连通分量时，因 $y\in\mathcal{D}(x)$，故 $y\notin\mathcal{A}(x)$，根据定理 3，可知 $\mathcal{A}(y)\cap\mathcal{D}(y)=\varnothing$，因 $\mathcal{A}(x)\subseteq\mathcal{A}(y)$，故 $\mathcal{A}(x)\cap\mathcal{D}(y)=\varnothing$，则 $\mathcal{A}(x)\cap\mathcal{D}(x)=\mathcal{A}(x)\cap(\{y\}\cup\mathcal{D}(y))=(\mathcal{A}(x)\cap\{y\})\cup(\mathcal{A}(x)\cap\mathcal{D}(y))=\varnothing$。因此，$x$ 单独构成一个无价值强连通分量，定理得证。

在 GRSCC 算法的强连通分量挖掘过程中，由于只关注有价值强连通分量的挖掘，忽略了针对无价值强连通分量的分析，进而也忽略了无价值强连通分量的快速识别对提高有价值强连通分量挖掘效率的作用。由定理 8 和定理 9 所论述的两种顶点间的无价值强连通分量相关性可知，GRSCC 对 SUB-RSCC 函数的一部分调用是不必要的。比如，当根据定理 1 发现某顶点 y 单独构成一个无价值强连通分量，或者针对顶点 y 调用 SUB-RSCC 函数挖掘出一个 y 单独构成的无价值强连通分量时，若某个顶点 x 的上近似集或 R 相关集只包含 y，那么针对 x 调用 SUB-RSCC 函数，必然会挖掘到一个由 x 单独构成的无价值强连通分量。

综合定理 6－定理 9 来看，无论顶点 y 属于一个有价值强连通分量还是一个无价值强连通分量，若某个顶点 x 的上近似集或 R 相关集只包含 y，那么针对 x 调用 SUB-RSCC 函数，必然不会挖掘出新的有价值强连通分量。因此，这些调用属于冗余调用，浪费了计算资源，降低了有价值强连通分量的挖掘效率。

接下来，将利用定理 6－定理 9 来设计一种新的顶点粒化策略，并提出了一种顶点粒 k 步搜索(KSVG)函数，使得粒化所得顶点粒可以得到最大程度的利用，以实现对 SUB-RSCC 函数调用方式的进一步优化，提高强连通分量的挖掘效率。

4 基于顶点粒 k 步搜索和粗糙集的强连通分量挖掘方法

在 GRSCC 算法的基础上，将粒化策略与顶点粒 k 步搜索方法相结合，提出了一种基于顶点粒 k 步搜索和粗糙集的强连通分量挖掘算法，如算法 2 所示。

算法 2 KGRSCC

输入：有向图 D={U,E}

输出：D 中的有价值强连通分量

```
1. SCCset←Ø;T←U;flag(T)←{0};T0←Ø;granu←Ø;
2. for x∈U do
3.    if R̄{x}=Ø or r(x)=Ø then
4.       T0← {T0,x};
5.    else
6.       if |R̄{x}|=1 then
7.          add x into granu(R̄{x});
8.       end if
9.       if |r(x)|=1 then
10.         add x into granu(r(x));
11.      end if
12.   end if
13. end for
14. Del=KSVG(T0);T←T \\ Del;
15. whileT ≠ Ø do
16.    SUB-RSCC(T(1));
17. end while
18. output SCCset;
19. function KSVG(X)
20.    k←0;gra0←X;Del←Ø;
21.    while grak≠ Ø do
22.       Del ← Del∪grak;flag(grak) ←{1};
23.       grak+1=∪{granu(y):y∈grak};
24.       grak+1←{y∈grak+1|flag(y)==0};
25.       k=k + 1;
27.    end while
28. end function
29. function SUB-RSCC(x)
30.    Compute Ux=x ∪{A(x)∩D(x)};
31.    if |Ux| >1 then
32.       SCCset←{SCCset,Ux};
33.    end if
34.    Delx← KSVG(Ux);T ← T\Delx;
35. end function
```

根据定理 6－定理 9 中阐述的 4 种挖掘强连通分量时顶点间的相关性，设计了一种新的粒化策略来对有向图的顶点集进行粒化以构建顶点粒(见算法 2 中的第 6－11 行)。具体的粒化方法为对每个上近似集和 R 相关集不为空的顶点 x 执行判断：1)判断其上近似集 $\overline{R}\{x\}$ 所包含的顶点数，若 $\overline{R}\{x\}$ 只包含一个顶点 y，则将顶点 x 加入顶点 y 所对应的顶点粒；2)判断其 R 相关集 $r(x)$ 所包含的顶点数，若 $r(x)$ 只包含一个顶点 y，则将顶点 x 加入顶点 y 所对应的顶点粒。

利用上述粒化策略对顶点集进行粒化后，形成了各个顶点所对应的顶点粒，对于单独构成一个无价值强连通分量的顶点，其对应顶点粒中的任一顶点也必然会单独构成一个无价值强连通分量；对于属于一个有价值强连通分量的顶点，其对应顶点粒中的任一顶点，要么属于当前有价值强连通分量，要么单独构成一个无价值强连通分量。这些对应顶点粒中的每个顶点对 SUB-RSCC 函数的调用都是冗余的，必然不会挖掘出一个新的有价值强连通分量，自然就不会影响有价值强连通分量的挖掘结果。

顶点粒的利用方式也会影响强连通分量的挖掘效率。考虑到每个顶点都有对应的顶点粒，那么顶点粒中的顶点必然也有其对应的顶点粒。对于一个确定属于某个强连通分量的顶点，可以确定该顶点对应顶点粒中的每个顶点必然不属于一个新的有价值强连通分量。那么，针对顶点粒中的每个顶点再次搜索顶点粒，对应顶点粒中的每个顶点必然也不属于一个新的有价值强连通分量。因此，KGRSCC 算法中定义了一种顶点粒 k 步搜索(k-step Search of Vertex Granule，KSVG)函数，当确定某个顶点所属的强连通分量是有价值或无价值之后，利用 KSVG 函数可以引导出一系列不需要调用 SUB-RSCC 函数的顶点。

为了保证 KSVG 函数计算结果的收敛性(否则这种顶点粒搜索过程可能会一直循环)，KGRSCC 算法还使用一种

顶点标记方法，对 KSVG 函数中每一步访问到的顶点加以标记，具体思路为：设置一个标志位 flag，flag 有 0 或 1 两个值，0 表示对应顶点未被标记，可能需调用 SUB-RSCC 函数，1 表示对应顶点已被标记，不需要调用 SUB-RSCC 函数，初始化所有顶点标志位为 0。

KSVG 函数的主要流程为：令 Del 为一个不需要调用 SUB-RSCC 函数的冗余顶点集，初始化 Del 为空集，若一个非空顶点集 X 中的每个顶点所属的强连通分量都是确定的，则将 X 中的每个顶点加入 Del 并对其进行标记(flag 置 1)，然后对 X 中的每个顶点同时进行 k 步的顶点粒搜索，其中 k 是一个非负整数，将每一步搜索到的顶点粒中未被标记(flag 为 0)的顶点加入 Del 并对其进行标记，直到某一步搜索到的顶点粒中不存在未被标记的顶点。当 KSVG 函数被调用时，随着对顶点粒的一步步搜索，被标记的顶点不断增加，最终顶点集 X 会引导出一个相应的非空顶点集 Del。

KGRSCC 算法中，SUB-RSCC 函数的调用方式如下：令 T 为需要调用 SUB-RSCC 的顶点集，初始化 T 为有向图顶点集 U。根据定理 1，对于 U 中的每个顶点 x，若 x 的上近似集或 R 相关集为空，则将 x 加入集合 T_0；否则根据定理 6－定理 9，驱动顶点集的粒化过程。然后，对集合 T_0 调用 KSVG 函数，计算得到一个冗余顶点集 Del，从 T 中删除 Del。针对 T 中的第一个顶点 x，调用 SUB-RSCC 函数，计算 x 所属的强连通分量顶点集 U_x，接着对 U_x 调用 KSVG 函数，计算得到冗余顶点集 Del_x，从 T 中删除 Del_x。如果 U_x 中的元素个数大于 1，U_x 引导出一个有价值强连通分量，将 U_x 加入到算法的输出结果集合。最后，直到 T 为空时，算法结束。

KGRSCC 和 GRSCC 算法的核心都是 SUB-RSCC 函数，两种算法的不同之处在于如何降低 SUB-RSCC 的调用次数：1)对于 SUB-RSCC 函数计算得到的有价值强连通分量，GRSCC 算法同时从 T 中删除有价值强连通中的顶点和每个顶点对应顶点粒中的顶点，而 KGRSCC 算法从 T 中删除了针对该有价值强连通分量进行顶点粒 k 步搜索后得到的所有冗余顶点(包含该有价值强连通分量中的顶点)；2)对于利用定理 1 或者 SUB-RSCC 函数得到的无价值强连通分量，GRSCC 算法仅仅从 T 中删除了无价值强连通分量中的顶点，而 KGRSCC 算法从 T 中删除了针对该无价值强连通分量进行顶点粒 k 步搜索后得到的所有冗余顶点(包含该无价值强连通分量中的顶点)。这种新的顶点删除方式无疑避免了更多的顶点对 SUB-RSCC 的冗余调用，同时也不会影响最终的挖掘结果。

例 3(续例 2)　利用 KGRSCC 算法求解有向图 D 中的有价值强连通分量：令 T 为需要调用 SUB-RSCC 的顶点集，初始化 $T=\{a,b,c,d,e,f,g,h,i,j,k,l,m,n,o\}$，$T_0=\varnothing$。首先，根据定理 1，$r(b)=\varnothing$，$\overline{R}\{l\}=\varnothing$，可知 $D_{\{b\}}$ 和 $D_{\{l\}}$ 是两个无价值强连通分量，将顶点 b 和 l 加入集合 T_0，$T_0=\{b,l\}$，然后对于同时存在上近似集和 R 相关集的顶点，根据定理 6－定理 9 驱动粒化过程构建顶点粒，其中每个顶点对应的顶点粒如表 2 所列。

表 2　KGRSCC 算法中每个顶点对应的顶点粒

Table 2　Vertex granules corresponding to each vertex in KGRSCC

顶点	a	b	c	d	e
顶点粒	$\{d\}$	$\{a,c\}$	$\varnothing$	$\{a\}$	$\{c,f,g\}$
顶点	f	g	h	i	j
顶点粒	$\{i\}$	$\{d,h\}$	$\{e\}$	$\{f\}$	$\{m\}$
顶点	k	l	m	n	o
顶点粒	$\{i,n\}$	$\{o\}$	$\{j\}$	$\{k,o\}$	$\{l\}$

接着，$Del=\mathrm{KSVG}(T_0)=\{a,b,c,d,l,o\}$，从 T 中删除 Del，得 $T=\{e,f,g,h,i,j,k,m,n\}$。

$T\neq\varnothing$，针对 T 中第一个顶点 e 调用 SUB-RSCC 函数，$U_e=e\cup\{\mathscr{N}(e)\cap\mathscr{D}(e)\}=\{e,g,h\}$，则 $D_{\{e,g,h\}}$ 是一个有价值强连通分量，$Del_e=\mathrm{KSVG}(U_e)=\{e,f,g,h,i\}$，从 T 中删除 Del_e，得 $T=\{j,k,m,n\}$。

$T\neq\varnothing$，针对 T 中第一个顶点 j 调用 SUB-RSCC 函数，$U_j=j\cup\{\mathscr{N}(j)\cap\mathscr{D}(j)\}=\{j\}$，则 $D_{\{j\}}$ 是一个无价值强连通分量，$Del_j=\mathrm{KSVG}(U_j)=\{j,m\}$，从 T 中删除 Del_j，得 $T=\{k,n\}$。

$T\neq\varnothing$，针对 T 中第一个顶点 k 调用 SUB-RSCC 函数，$U_k=k\cup\{\mathscr{N}(k)\cap\mathscr{D}(k)\}=\{k,n\}$，则 $D_{\{k,n\}}$ 是一个有价值强连通分量，$Del_k=\mathrm{KSVG}(U_k)=\{k,n\}$，从 T 中删除 Del_k，得 $T=\varnothing$，算法结束。

经统计发现，KGRSCC 算法成功挖掘出有向图 D 中的全部 12 个强连通分量，其中两个有价值强连通分量 $D_{\{e,g,h\}}$ 和 $D_{\{k,n\}}$，一共调用了 3 次 SUB-RSCC 函数。

结合例 2、例 3，GRSCC、KGRSCC 算法运行时 T 的变化以及 SUB-RSCC 的调用情况如图 3 所示。

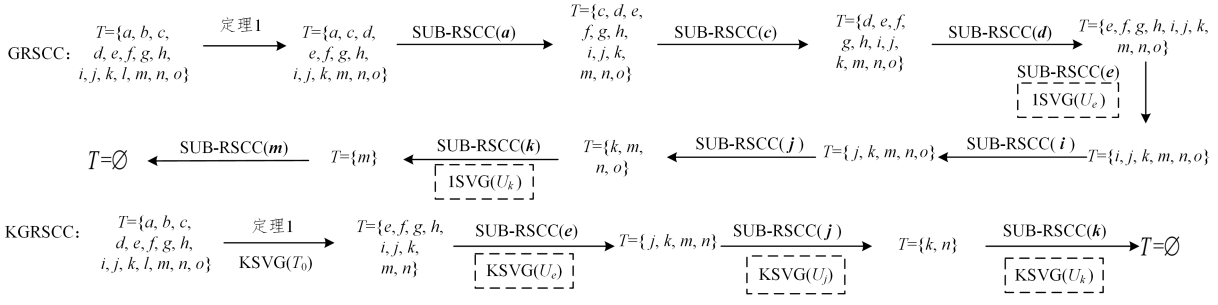


图 3　两种算法在有向图 D 上运行时 T 的变化过程以及对 SUB-RSCC 的调用情况

Fig. 3　Change process of T and invocation times of SUB-RSCC when two algorithms run on digraph D

可以看到，GRSCC 算法对 SUB-RSCC(e)和 SUB-RSCC(k)计算得到的有价值强连通分量的顶点集 U_e 和 U_k 搜索了一次顶点粒(与 KSVG 函数相对应，称这两个搜索过程为 1SVG(U_e)和 1SVG (U_k))，利用顶点粒避免了顶点 f 和 o

对 SUB-RSCC 的冗余调用，对 a,c,d,e,i,j,k,m 这 8 个顶点共调用了 8 次 SUB-RSCC 函数，而 KGRSCC 算法通过对上近似集或 R 相关集为空的顶点集合 T_0、SUB-RSCC 函数中计算得到的有价值强连通分量 U_e 和 U_k 以及无价值强连通分量 U_j 调用 KSVG 函数，减少了更多 SUB-RSCC 被调用的次数，只对顶点 e,j,k 调用了 3 次 SUB-RSCC 就挖掘出了全部有价值强连通分量。这意味着对于同一个有向图(见图 1)，相比 GRSCC 算法，使用了新的粒化策略以及顶点粒 k 步搜索方法的 KGRSCC 算法，SUB-RSCC 函数被调用的次数更少，强连通分量的挖掘效率更高。

事实上，GRSCC 算法也使用了顶点粒 k 步搜索方法，在针对无价值强连通分量进行 k 步搜索时，$k=0$ 就停止搜索，在针对有价值强连通分量进行 k 步搜索时，$k=1$ 就停止搜索。

图 4—图 6 分别给出了 3 种情形下 GRSCC 和 KGRSCC 算法对顶点粒的搜索过程，第一种情形(见图 4)针对的是上近似集或 R 相关集为空的顶点集合，第二种情形(见图 5)针对的是 SUB-RSCC 函数中计算得到的有价值强连通分量，第三种情形(见图 6)针对的是 SUB-RSCC 函数中计算得到的无价值强连通分量。

根据图 3，$r(b)=\emptyset$，$\overline{R}\{l\}=\emptyset$，根据定理 1，GRSCC 算法对顶点 b 和 l 的操作如图 4(a)所示：$k=0$，$Del=\{b,l\}$，然后从 T 中删除 Del。

根据定理 8 和定理 9，KGRSCC 算法对顶点 b 和 l 的操作如图 4(b)所示：1)$k=0$ 时，针对 T_0 中的顶点 b 和 l，先将它们标记为红色，再搜索对应的顶点粒，访问到顶点 a,c 和 o，发现它们都未被标记；2)$k=1$ 时，针对顶点 a,c 和 o，先将它们标记为红色，再搜索对应的顶点粒，发现 c 对应的顶点粒为空集，o 对应的顶点粒中唯一顶点 l 已被标记，只有 a 对应顶点粒中的顶点 d 未被标记；3)$k=2$ 时，针对顶点 d，先将它标记为红色，再搜索对应的顶点粒，发现 d 对应顶点粒中的唯一顶点 a 已被标记；4)$k=3$ 时，无法访问到未被标记的顶点，搜索结束，这个过程中所有被标记的顶点构成集合 $Del=\{a,b,c,d,l,o\}$，然后将 Del 从调用顶点集 T 中删除。

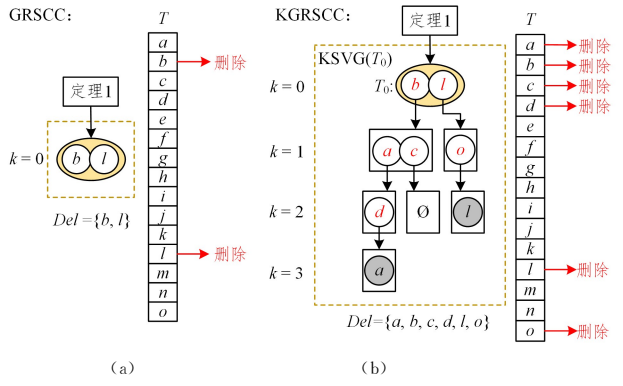


图 4 第一种情形下 GRSCC 和 KGRSCC 算法对顶点粒的搜索过程(电子版为彩图)

Fig. 4 Process of GRSCC and KGRSCC algorithms searching for vertex granules in the first case

可以看到，针对相同的顶点集 $T_0=\{b,l\}$，两种算法的处理方式不同，与 GRSCC 算法相比，KGRSCC 算法不仅从调用顶点集 T 中删除了顶点 b 和 l，还删除了顶点粒 k 步搜索后所访问到的 a,c,d,o 这 4 个顶点，结合图 3 中 GRSCC 算法对 SUB-RSCC 的调用情况，明显看出，KGRSCC 算法减少了 3 次 SUB-RSCC 调用，分别是 SUB-RSCC(a)，SUB-RSCC(c)和 SUB-RSCC(d)。

根据图 3，顶点 e 调用 SUB-RSCC 函数后，计算得到一个有价值强连通分量顶点集 $U_e=\{e,g,h\}$，根据定理 4—定理 7，GRSCC 算法对其操作如图 5(a)所示：1)$k=0$ 时，对 U_e 中的每个顶点搜索顶点粒；2)$k=1$ 时，对所有顶点粒中的顶点求并集，$Del_e=\{e,g,h\}\cup\{c,f,g,h\}\cup\{d,e,h\}\cup\{e,g\}=\{e,f,g,h\}$，然后从 T 中删除 Del_e。

根据定理 6—定理 9，KGRSCC 算法对其操作如图 5(b)所示：1)$k=0$ 时，针对 U_e 中顶点 e,g 和 h，先将它们标记为红色，然后搜索对应的顶点粒，注意到只有 e 对应顶点粒中的顶点 f 未被标记；2)$k=1$ 时，针对顶点 f，先将它标记为红色，然后搜索对应的顶点粒，发现 f 对应顶点粒中的顶点 i 也未被标记；3)$k=2$ 时，针对顶点 i，先将它标记为红色，然后搜索对应的顶点粒，发现 i 对应顶点粒中的唯一顶点 f 已被标记；4)$k=3$ 时，无法访问到未被标记的顶点，搜索结束，这个过程中所有被标记的顶点构成集合 $Del_e=\{e,f,g,h,i\}$，然后将 Del_e 从调用顶点集 T 中删除。

可以看到，针对相同的顶点集 $U_e=\{e,g,h\}$，两种算法的处理方式不同，与 GRSCC 算法相比，KGRSCC 算法不仅从调用顶点集 T 中删除了顶点 e,f,g,h，还删除了顶点粒 k 步搜索后所访问到的顶点 i，结合图 3 中 GRSCC 算法对 SUB-RSCC 的调用情况，明显可以看出，KGRSCC 算法减少了一次 SUB-RSCC 调用，即 SUB-RSCC(i)。

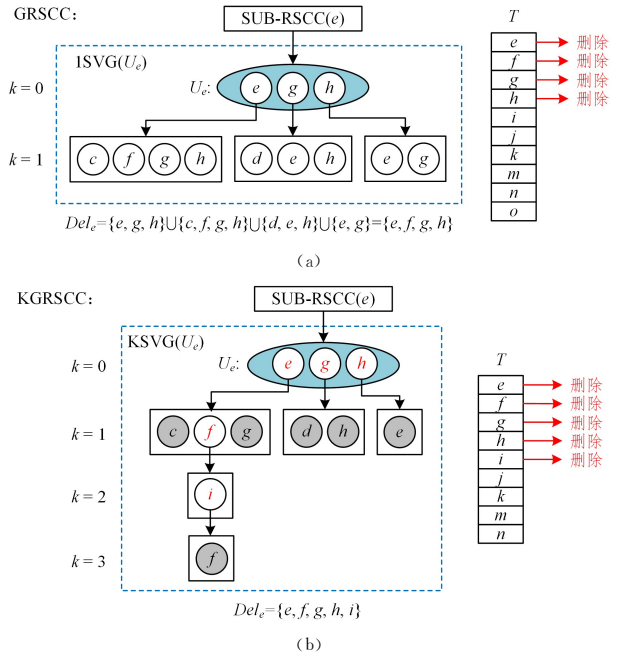


图 5 第二种情形下 GRSCC 和 KGRSCC 算法对顶点粒的搜索过程(电子版为彩图)

Fig. 5 Process of GRSCC and KGRSCC algorithms searching for vertex granules in the second case

根据图 3，顶点 j 调用 SUB-RSCC 函数后，计算得到一个无价值强连通分量顶点集 $U_j=\{j\}$，GRSCC 算法对其操作如图 6(a)所示：$k=0$ 时，$Del_j=\{j\}$，然后从 T 中删除 Del_j。

根据定理 8、定理 9，KGRSCC 算法对其操作如图 6(b)所示：1) $k=0$ 时，针对 U_j 中的顶点 j，先将它标记为红色，然后搜索对应的顶点粒，发现 j 对应顶点粒中的顶点 m 未被标记；2)$k=1$ 时，针对顶点 m，先将它标记为红色，然后搜索对应的顶点粒，发现 m 对应顶点粒中的顶点 j 已被标记；3)$k=2$ 时，无法访问到未被标记的顶点，搜索结束，这个过程所有被标记的顶点构成集合 $Del_j=\{j,m\}$，然后将 Del_j 从调用顶点集 T 中删除。

可以看到，针对相同的顶点集 $U_j=\{j\}$，两种算法的处理方式不同，与 GRSCC 算法相比，KGRSCC 算法不仅从调用顶点集 T 中删除了顶点 j，还删除了顶点粒 k 步搜索后所访问到的顶点 m，结合图 3 中 GRSCC 算法对 SUB-RSCC 的调用情况可以看出，KGRSCC 算法减少了一次 SUB-RSCC 调用，即 SUB-RSCC(m)。

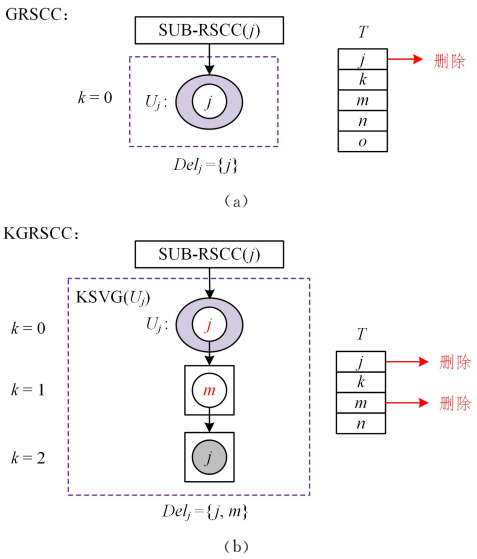


图 6 第三种情形下 GRSCC 和 KGRSCC 算法对顶点粒的搜索过程(电子版为彩图)

Fig. 6 Process of GRSCC and KGRSCC algorithms searching for vertex granules in the third case

利用 KGRSCC 算法挖掘一个有向图中的有价值强连通分量时，第一种情形只会发生一次，而第二、三种情形出现的次数则取决于图的结构。

与 GRSCC 算法相同，KGRSCC 算法的时间消耗也取决于顶点调用 SUB-RSCC 的次数，时间复杂度同为 $O(c(n+m))$。在本例中，KGRSCC 算法确实进一步减少了 SUB-RSCC 的调用次数，大大降低了算法的时间消耗。但 KGRSCC 算法相对于 GRSCC 算法的优势取决于目标有向图的结构特征，若目标有向图中不存在无价值强连通分量，顶点粒 k 步搜索方法就没有优化挖掘效率的空间，KGRSCC 算法的优势也就不会体现出来。

5 仿真实验

本实验使用 Matlab2017(b)编程，配置处理器为四核 Intel(R) Core (TM) i5-8265 @ 1.6GHz，内存 8GB，选取经典的 Tarjan 算法、RSCC 算法以及 GRSCC 算法作为比较算法。为了测试 KGRSCC 算法和 3 种比较算法的性能，实验使用 UFSM 数据库[32]中的 10 个图数据集进行测试，每个数据集上运行 10 次，实验结果为计算 10 次运行记录的平均值。数据集信息如表 3 所列，其中 n 表示顶点数，m 表示有向边数，c 表示有向图中有价值的强连通分量的数目。

表 3 数据集信息

Table 3 Datasets information

编号	数据集	n	m	c
1	celegansneural	297	2345	3
2	DK01R	903	10863	3
3	CollegeMsg	1899	20296	6
4	zenios	2873	1314	45
5	poli	4008	4180	38
6	EPA	4772	8965	26
7	shermanACd	6136	47193	2
8	EVA	8497	6724	10
9	FA	10617	72172	9
10	foldoc	13356	120238	6

实验结果表明，4 种算法的挖掘准确率是一致的，它们都能准确挖掘出所采用数据集中的全部有价值强连通分量，不需赘述。本文主要关注它们的效率比较，表 4 列出了 4 种算法挖掘效率的比较结果，其中 Tarjan，RSCC，GRSCC，KGRSCC 这 4 种挖掘算法的时间消耗 t 分别对应表 4 中第 2，3，6，9 列(最小值加粗)，单位为 s；N_{SUB} 的值表示 RSCC，GRSCC，KGRSCC 算法调用 SUB-RSCC 的次数，分别对应表 4 中第 4，7，10 列(最小值加粗)；S_T 的值表示 RSCC，GRSCC，KGRSCC 相比 Tarjan 算法的加速比，计算的是 Tarjan 算法对 3 种算法时间消耗的比值，分别对应表 4 中第 5，8，11 列(最大值加粗)；S_R 和 S_G 的值分别表示 KGRSCC 相比 RSCC 和 GRSCC 算法的加速比，计算的是两种算法对 KGRSCC 算法时间消耗的比值，对应表 4 中第 12，13 列。

表 4 Tarjan，RSCC，GRSCC 和 KGRSCC 算法的挖掘效率比较

Table 4 Comparison of mining efficiency of Tarjan，RSCC，GRSCC and KGRSCC algorithms

数据集	Tarjan	RSCC			GRSCC			KGRSCC				
	t	t	N_{SUB}	S_T	t	N_{SUB}	S_T	t	N_{SUB}	S_T	S_R	S_G
celegansneural	0.0168	0.0068	27	2.46	0.0066	25	2.53	**0.0029**	**3**	**5.82**	2.36	2.30
DK01R	0.0569	0.0077	5	7.41	0.0079	5	7.21	**0.0063**	**3**	**9.01**	1.22	1.25
CollegeMsg	0.1346	0.0280	15	4.81	0.0273	12	4.93	**0.0242**	**6**	**5.57**	1.16	1.13
zenios	0.1652	0.0305	**45**	5.42	**0.0300**	**45**	**5.50**	0.0308	**45**	5.37	0.99	0.98
poli	0.2544	0.0667	529	3.82	0.0611	480	4.17	**0.0277**	**103**	**9.18**	2.41	2.20
EPA	0.3466	0.1177	717	2.95	0.1129	670	3.07	**0.0559**	**199**	**6.20**	2.10	2.02
shermanACd	0.2999	0.0723	16	4.15	0.0736	16	4.07	**0.0250**	**2**	**12.01**	2.90	2.95
EVA	0.7637	0.0982	399	7.78	0.0973	392	7.85	**0.0435**	**76**	**17.57**	2.26	2.24
FA	0.7635	0.1157	16	6.60	0.1126	15	6.78	**0.0839**	**9**	**9.10**	1.38	1.34
foldoc	0.7758	0.2491	24	3.11	0.2201	21	3.52	**0.1372**	**11**	**5.66**	1.82	1.60

观察表 4，比较 RSCC，GRSCC，KGRSCC 这 3 种算法的 N_{SUB} 值（见表 4 中的第 4，7，10 列）；除了数据集 zenios 上三者是相等的，即不存在 SUB-RSCC 的冗余调用，其他 9 个数据集上 KGRSCC 算法都比 RSCC 算法、GRSCC 算法的 N_{SUB} 值小，这说明采用了新粒化策略以及顶点粒 k 步搜索的 KGRSCC 算法确实能够降低 SUB-RSCC 的调用次数，而且避免了更多顶点对 SUB-RSCC 的冗余调用。相应地，KGRSCC 算法中 SUB-RSCC 调用次数上的优势在算法时间消耗上也得到了体现，比较 RSCC，GRSCC，KGRSCC 这 3 种算法的 t 值（见表 4 中的第 3，6，9 列），除了数据集 zenios 上三者相近，其他数据上 KGRSCC 算法的 t 值都远小于 RSCC，GRSCC 算法。另外，对 Tarjan 算法的时间消耗的加速比 S_T 是评价一个新的强连通分量挖掘算法的重要指标。在选取的 10 个数据集上，KGRSCC 算法对 Tarjan 算法的加速比 S_T（见表 4 中的地第 1 列）为 5.37～17.57，特别地，EVA 数据集上加速比达到了 17.57，与 RSCC 算法、GRSCC 算法的 S_T 值（见表 4 中的第 4，7 列）相比，由于时间消耗上的优势，在加速比这一指标上，KGRSCC 算法也体现出了优势。

为了更直观地对比 Tarjan，RSCC，GRSCC，KGRSCC 算法在全部 10 个数据集上的时间消耗，以及观察 4 种算法时间消耗随顶点数的线性变化趋势，图 7 给出了不同数据集下 4 种算法的时间消耗折线图。由图 7 可以看出：1) KGRSCC 算法的时间消耗比 Tarjan 算法低很多；2) 除了在 zenios 数据集上 KGRSCC 算法的时间消耗与 RSCC 算法、GRSCC 算法相当，在其他 9 个数据集上 KGRSCC 算法的时间消耗更低。

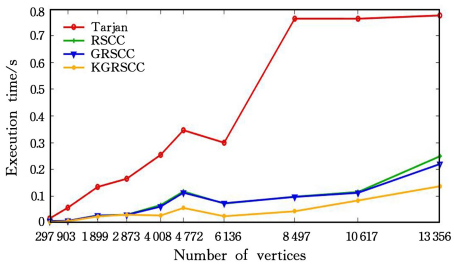


图 7 Tarjan，RSCC，GRSCC，KGRSCC 算法在 10 个数据集上的时间消耗

Fig. 7 Time consumption of Tarjan，RSCC，GRSCC and KGRSCC algorithms on 10 datasets

结束语 强连通分量是有向图中的重要结构，是图论的重要研究内容。之前的工作中，一系列基于粗糙集的强连通分量挖掘算法[25-26]被提出，其算法核心都是 SUB-RSCC 函数（由 k 步上近似和 k 步 R 相关集这两个粗糙集算子构成）。SUB-RSCC 函数的调用次数决定了以 SUB-RSCC 函数为核心的该类算法的计算效率。其中，GRSCC 算法利用顶点间存在的 4 种有价值强连通分量相关性，设计了一种粒化策略，成功减少了 SUB-RSCC 函数的调用次数。

本文利用上近似集与 R 相关集这两种粗糙集概念对强连通分量进行了深入分析，发现了顶点间的另外两种无价值强连通分量的相关性。进而，在 GRSCC 算法的基础上，设计了一种新的顶点粒化策略，并提出了一种顶点粒 k 步搜索（KSVG）函数，更大程度地利用了顶点粒。通过 KSVG 函数，可以引导出一系列不需调用 SUB-RSCC 函数的顶点，将它们从 SUB-RSCC 函数的调用顶点集 T 中删除，避免了更多顶点对 SUB-RSCC 函数的冗余调用，节省了计算资源。最后，本文提出了一种基于顶点粒 k 步搜索和粗糙集的强连通分量挖掘算法 KGRSCC。实验结果表明，KGRSCC 算法在挖掘强连通分量时表现出了更好的性能，其效率可达到 Tarjan，RSCC，GRSCC 算法的 17.57，2.90，2.95。本文设计了更加优化的粒化策略，将其应用到强连通分量挖掘问题上，进而提出了一种基于 KSVG 函数的顶点粒利用方法，成功实现了挖掘效率的进一步提升，为未来将粒化策略应用于其他相关图结构挖掘问题提供了借鉴与指导。

参 考 文 献

[1] PEARCE D J，KELLY P H，HANKIN C. Efficient field-sensitive pointer analysis of C[J]. ACM Transactions on Programming Languages and Systems，2007，30(1)：4-14.

[2] BURKE M. An interval-based approach to exhaustive and incremental interprocedural data-flow analysis[J]. ACM Transactions on Programming Languages and Systems，1990，12(3)：341-395.

[3] YANG H G，SHEN D R，KOU Y，et al. Strongly connected components based efficient computation of page rank[J]. Frontiers of Computer Science，2018，12(6)：1208-1219.

[4] ADAMIC L A. The Small World Web [C]// Proceedings of International Conference on Theory and Practice of Digital Libraries. Springer，1999：443-452.

[5] IOANNIDIS Y，RAMAKRISHNAN R，WINGER L. Transitive closure algorithms based on graph traversal[J]. ACM Transactions on Database Systems，1993，18(3)：512-576.

[6] NUUTILA E，SOISALON-SOININEN E. On finding the strongly connected components in a directed graph[J]. Information Processing Letters，1994，49(1)：9-14.

[7] PEARCE D J. A space-efficient algorithm for finding strongly connected components [J]. Information Processing Letters，2016，116(1)：47-52.

[8] TARJAN R. Depth-First Search and Linear Graph Algorithms [J]. SIAM Journal on Computing，1972，1(2)：146-160.

[9] SHARIR M. A strong-connectivity algorithm and its applications in data flow analysis[J]. Computers& Mathematics with Applications，1981，7(1)：67-72.

[10] GABOW H N. Path-based depth-first search for strong and biconnected components [J]. Information Processing Letters，2000，74(3)：107-114.

[11] GAZIT H，MILLER G L. An improved parallel algorithm that computes the BFS numbering of a directed graph[J]. Information Processing Letters，1988，28(2)：61-65.

[12] FLEISCHER L K，HENDRICKSON B，PINAR A. OnIdentifying Strongly Connected Components in Parallel[J]. Lecture Notes in Computer Science，2000，1800(4)：505-511.

[13] MCLENDON III W，HENDRICKSON B，PLIMPTON S J，et al. Finding strongly connected components in distributed

graphs[J]. Journal of Parallel and Distributed Computing,2005,65(8):901-910.

[14] BARNAT J,BAUCH P,BRIM L,et al. Computing Strongly Connected Components in Parallel on CUDA [C]// Proceedings of International Parallel and Distributed Processing Symposium. IEEE,2011:544-555.

[15] LOWE G. Concurrent depth-first search algorithms based on Tarjan's Algorithm[J]. International Journal on Software Tools for Technology Transfer,2016,18(2):129-147.

[16] PAWLAK Z. Rough sets:Theoretical aspects of reasoning about data[M]. Kluwer Academic Publishers,1992.

[17] YAO Y Y,ZHANG X Y. Class-specific attribute reducts in rough set theory[J]. Information Sciences,2017,418-419:601-618.

[18] QIAN Y H,LIANG X Y,WANG Q,et al. Local rough set:A solution to rough data analysis in big data[J]. International Journal of Approximate Reasoning,2018,97:38-63.

[19] WANG G Y,MA X A,YU H. Monotonic uncertainty measures for attribute reduction in probabilistic rough set model[J]. International Journal of Approximate Reasoning,2015,59:41-67.

[20] YANG X B,ZHANG M,DOU H L,et al. Neighborhood systems-based rough sets in incomplete information system[J]. Knowledge-Based Systems,2011,24(6):858-867.

[21] YANG X B,YAO Y Y. Ensemble selector for attribute reduction[J]. Applied Soft Computing,2018,70:1-11.

[22] YANG X B,LIANG S C,YU H L,et al. Pseudo-label neighborhood rough set:measures and attribute reductions[J]. International Journal of Approximate Reasoning,2019,105:112-129.

[23] SONG J J,TSANG E,CHEN D G,et al. Minimal decision cost reduct in fuzzy decision-theoretic rough set model[J]. Knowledge-Based Systems,2017,126:104-112.

[24] LIU K Y,YANG X B,YU H L,et al. Rough set based semi-supervised feature selection via ensemble selector[J]. Knowledge-Based Systems,2019,165:282-296.

[25] XU T H,WANG G Y. Finding strongly connected components of simple digraphs based on generalized rough sets theory[J]. Knowledge Based Systems,2018,149:88-98.

[26] XU T H,WANG G Y,YANG J. Finding strongly connected components of simple digraphs based on granulation strategy [J]. International Journal of Approximate Reasoning,2020,118:64-78.

[27] BANG-JENSEN J,GUTIN G Z. Digraphs Theory,Algorithms and Applications,Second Edition[M]. Berlin:Springer Science & Business Media,2009.

[28] YAO Y Y. Two views of the theory of rough sets in finite universes[J]. International Journal of Approximate Reasoning,1996,15(4):291-317.

[29] JARVINEN J. Lattice theory for rough sets[J]. Transactions on Rough Sets,2007,6:400-498.

[30] FAN T F. Rough set analysis of relational structures[J]. Information Sciences,2013,221:230-244.

[31] CHEN J K,LI J J,LIN Y J. Computing connected components of simple undirected graphs based on generalized rough sets[J]. Knowledge-Based Systems,2013,37:80-85.

[32] DAVIS T A,HU Y F. The university of Florida sparse matrix collection[J]. ACM Transactions on Mathematical Software,2011,38(1):1-25.

CHENG Fu-hao, born in 1994, postgraduate. His main research interests include granular computing and graph theory.

XU Tai-hua, born in 1989, Ph. D, lecture, master supervisor, is a member of China Computer Federation. His main research interests include intelligent information processing, granular computing and graph theory.

(责任编辑:喻藜)