Nondominated Local Coteries for Resource Allocation in Grids and Clouds

Jehn-Ruey Jiang
Department of Computer Science and Information Engineering
National Central University, Jhongli, 32001, Taiwan
E-mail: jrjiang@csie.ncu.edu.tw

Abstract

The resource allocation problem is a fundamental problem in Grid and Cloud computing environments. This paper focuses on constructing *nondominated* (*ND*) *local coteries* to solve the problem in a distributed way. Distributed algorithms using coteries usually incur low communication overhead and have high degree of fault-tolerance, and ND coteries are candidates for the algorithms to achieve the highest degree of fault-tolerance. A new type of coteries, called *p-coteries*, is defined to aid the construction of local coteries. Theorems about the nondomination of p-coteries are then developed, and an operation, called *pairwise-union* (*p-union*), is proposed to help generate ND p-coteries, which in turn can be used to generate ND local coteries for solving the resource allocation problem.

Keywords: *Mutual exclusion, resource allocation, distributed computing, coteries, quorums, nondomination*

1 Introduction

A Grid system is described as a computation infrastructure for resource sharing in dynamic virtual organizations [1]. A virtual organization is formed temporally for cooperation and resource sharing among several institutions to achieve some common computation goal. Also with the concept of virtualization, a Cloud system packs computing power and other resources as services to be leased to individuals or enterprise users [2]. Some resources, such as database objects, protected shared mutable storage, special data visualization displays and DAT tape drives, must be shared in a mutually exclusive way. Jobs or processes running in Grids or Clouds may require multiple resources simultaneously. Problems arise in designing efficient resource allocation algorithm allowing processes to simultaneously acquire multiple resources for execution. This paper proposes solving the resource allocation problem in a distributed manner with the help of local coteries constructed by the pairwise union (p-union) operation. It also shows that some local coteries are toward the optimality in the sense that they are not dominated by other local coteries.

In this paper, a Grid or a Cloud computing environment is regarded as a distributed system consisting of a set P of processes and a set R of shared resources each of which is of a different type and must be accessed in a mutually exclusive way. The processes can communicate with each other by exchanging messages, and from time to time, a process may request to enter the *critical section* (*CS*) to access some of the resources. A process p_i, $p_i \in P$, enters the *CS* after it acquires all the requested resources. Afterwards, p_i leaves the *CS* and releases all the acquired resources. Processes are assumed to leave the *CS* in finite time. The resource allocation problem is concerned with how to ensure that all resources are accessed in a mutually exclusive way and that all processes wishing to enter the *CS* can proceed in a finite time.

The process-accessing-resource relation in the resource allocation problem can be represented by a *resource allocation graph* (*RAG*). A *RAG* for the system with process set P and resource set R is a bipartite graph $G=(V, E)$, where $V=P \cup R$ is a set of vertices and E is a set of edges. There is an edge $e=(p, r) \in E$ if and only if process p requests to access resource r. Let $R_i = \{r \mid \text{process } p_i \text{ requests to access resource } r\}$ be the set of all the resources

that process p_i requests to access. If $R_i \cap R_j \neq \varnothing$, it means that process p_i and process p_j compete for the same resources.

Fig. 1 is an example of RAG for the system with $P=\{p_1, p_2, p_3\}$ and $R=\{r_1, r_2\}$. It is noted that we represent processes and resources as circles and squares, respectively. With respect to the RAG, $R_1=\{r_1\}$, $R_2=\{r_1, r_2\}$ and $R_3=\{r_2\}$, which means that process p_1 requests to access resource r_1, process p_2 requests to access resources r_1 and r_2, and process p_3 requests to access resource r_2.

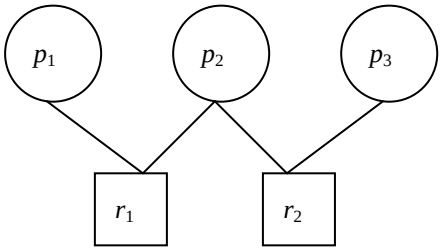

Figure 1. The resource allocation graph (RAG) for the system with $P=\{p_1, p_2, p_3\}$ and $R=\{r_1, r_2\}$

The mutual exclusion [3], the *k*-mutual exclusion [4], the *h*-out of-*k* mutual exclusion [5], the dining philosophers [6] and the drinking philosophers [7] problems are all special cases of the distributed resource allocation problem. The mutual exclusion problem deals with the mutually-exclusive sharing of a unique resource among all processes. The *k*-mutual exclusion problem deals with the sharing of *k* identical resources with the restriction that one process can access any one resource at a time. The *h*-out of-*k* mutual exclusion problem deals with the sharing of *k* identical resources with the restriction that one process can access any *h*, $h \leq k$, resources at a time. The dining philosophers problem and the drinking philosophers problem describe the resource sharing relation by the *conflict graph*, in which a vertex represents a process and an edge represents the resource shared by the two processes incident to the edge. It is noted that a conflict graph can be transformed to be a resource allocation graph by adding a resource node in the middle of an edge. In the dining philosophers problem, a process can enter the *CS* when it has acquired all the resources represented by the edges incident to it; while in the drinking philosophers problem, a process can enter the *CS* when it has acquired a subset of the resources.

Quorum-based algorithms are promising for solving the distributed resource problem. Their basic idea is to as follows. Processes are first grouped as *quorums* to form a *quorum system* which is a family of sets (quorums) satisfying the *intersection property* that any two quorums overlap mutually. It is noted that if a quorum system also satisfies the *minimality property* that no quorum is a super set of others, then the quorum system is called a *coterie*. Many coterie-related structures are proposed to solve different cases of the distributed resource allocation problem. For example, the coterie is used to solve the mutual exclusion problem [8, 9], the *k*-coterie is used to solve the *k*-mutual exclusion problem [10, 11] and the *h*-out of-*k* mutual exclusion problem [12], and the local coterie [13, 14] is used to solve the general resource allocation problem.

The solutions using coterie structures usually incur low communication overhead and can tolerate process and/or communication link faults. Among coterie structures, *nondominated* (ND) coterie structures [8, 15] are candidates for the solutions to achieve the highest degree of fault-tolerance. Thus, we should always concentrate on ND coterie structures if fault-tolerance is significant. There are papers [8, 9, 10, 15] investigating ND coteries or ND *k*-coteries.

This paper investigates ND local coteries and aims at constructing ND local coteries to solve the resource allocation problem in Grids and Clouds in a distributed way. A new type of coteries, called *p-coteries*, is proposed to aid the construction of local coteries. Theorems about the nondomination of p-coteries are developed, and an operation, called *pairwise-union* (*p-union*), is proposed to help generate ND p-coteries from known ND

coteries, such as majority coteries, tree coteries, hierarchical coteries, composite coteries, level coteries, Lovasz coteries and cohorts coteries, all of which have been shown to be ND, as mentioned in [15]. As defined in [8], a coterie is family of mutually intersected, minimal subsets (or quorums) of a universal set. And as defined in [13], a local coterie is a list[1] of coteries. By the theorems and the operation, an ND local coterie $LC=(C_1,\ldots,C_{|P|})$ can be constructed to solve the resource allocation problem with process set P and resource set R as flows. For each resource r_j in R, an ND coterie Cr_j is constructed. Then for each process p_i in P, an ND p-coterie C_i is constructed for the following two cases. For the case that p_i accesses only one resource, say r_j, we set $C_i=Cr_j$. For the case that p_i accesses two or more resources, say $r_1,\ldots,r_m$, $m>1$, we set $C_i= Min(Cr_1\otimes\ldots\otimes Cr_m)$, where $\otimes$ stands for the p-union operation and $Min(Q)$ is a function to eliminate non-minimal quorums from a family Q of quorums. Process p_i needs to receive permissions from all members of a quorum of C_i to enter the CS; it releases all permissions after leaving the CS. Maekawa's permission-based algorithm [16] can be used for obtaining and releasing permissions to prevent deadlocks and livelocks. As will be shown, the p-coterie has the *inter-coterie intersection property*, which can be relied on to solve the resource allocation problem. Furthermore, since the constructed local coterie LC is ND, the solution is toward the highest degree of fault-tolerance.

The rest of this paper is organized as follows. Section 2 shows the definitions and examples of the coterie, local coterie and p-coterie. And Section 3 introduces theorems for checking their nondomination. It is also shown that the p-union operation can help generate ND p-coteries for the construction of ND local coteries. And finally, Section 4 concludes this paper.

2 The Coterie, Local Coterie and p-Coterie

In this section, we give the definitions and examples of the coterie, the local coterie and the p-coterie. We also introduce Kakugawa and Yamashita's algorithm [13] and Cheng et al.'s algorithm [14] for constructing local coteries to solve the resource allocation problem.

Definition 1. (Coterie) [8]

A *coterie* C under P is a family of subsets of P. Each member in C is called a *quorum* and should observe the following two properties:

Intersection Property: $\forall q_1, q_2 : q_1, q_2 \in C : q_1 \cap q_2 \neq \emptyset$

Minimality Property: $\forall q_1, q_2 : q_1, q_2 \in C : q_1 \not\subset q_2$

For example, $C = \{\{p_1, p_2\}, \{p_1, p_3\}, \{p_2, p_3\}\}$ is a coterie under $P=\{p_1, p_2, p_3\}$ because every pair of quorums (members) in C have a non-empty intersection, and no quorum is a super set of another quorum.

Definition 2. (Local Coterie) [13]

Given a RAG of the system with a set P of processes and a set R of resources, a local coterie $LC=(C_1,\ldots,C_{|P|})$ is a list of coteries under P. There is a coterie C_i associated with each process $p_i \in P$, $1 \le i \le |P|$ and all of the following conditions should hold:

***Non-emptiness Property*:** $\forall p_i : p_i \in P : C_i \neq \emptyset$

***Intersection Property*:** If $R_i \cap R_j \neq \emptyset$, then $\forall q_1, q_2$: $q_1 \in C_i$, $q_2 \in C_j$: $q_1 \cap q_2 \neq \emptyset$, where $R_i = \{r \mid$ process p_i requests to access resource $r\}$.

Minimality Property: $\forall p_i, q_1, q_2 : p_i \in P, q_1, q_2 \in C_i : q_1 \not\subset q_2$

[1] A local coterie is defined to be a "*set*" of coteries in paper [13]. Since the order of the coteries makes sense, we modify the definition of the local coterie to be a "*list*" of coteries.

For example, LC=({{p_1}}, {{p_1, p_3}}, {{p_3}}) is a local coterie for the RAG in Fig. 1. The reader can check that there is a coterie associated with every process (for example, C_1={{p_1}} for process p_1, C_2={{p_1, p_3}} for process p_2 and C_3={{p_3}} for process p_3) and every quorum in C_2 intersects with every quorum in C_1 (resp. C_3) because p_2 and p_1 (resp. p_3) compete for the same resource r_1 (resp. r_2).

The local coterie can be used to develop algorithms solving the resource allocation problem. To enter the critical section, a process is required to form a quorum, that is, to receive the permissions from all the processes of some quorum of its associated coterie. If we restrict that every process can grant its permission to only one process at a time, then the mutually exclusive access of resources is guaranteed because any two quorums q_1 and q_2, $q_1 \in C_i$ and $q_2 \in C_j$, must intersect when p_i and p_j compete for the same resources. It is noted that the minimality property is not necessary for the correctness of resource allocation but is used to enhance efficiency.

After obtaining permissions from all members of a quorum in the associated coterie, a process can enter the CS. It is assumed to leave the CS in finite time, and to release all obtained permissions after leaving the CS. To prevent deadlocks and livelocks, Maekawa's algorithm [16] should be used to obtain and release permissions.

Kakugawa and Yamashita proposed an algorithm [13] to construct local coteries. In the algorithm, for each process p_i, its associated coterie C_i is {q_i}, where q_i={$p_j | R_i \cap R_j \neq \emptyset$} (i.e., C_i has only one quorum containing all the processes competing resources with p_i). For example, with respect to the RAG in Fig. 1, a local coterie constructed by the Kakugawa and Yamashita's algorithm is ({{p_1, p_2}}, {{p_1, p_2, p_3}}, {{p_2, p_3}}). Cheng et al. proposed another algorithm [14] to construct local coteries, described below. For each resource r_j, the algorithm first finds out P_j={p| process p accesses resource r_j}, the set of all processes that access resource r_j. Then, for each resource r_j, the algorithm constructs a coterie Cr_j under P_j (note that in this paper, we use the term "*the coterie for resource* r_j" to refer to Cr_j). Afterwards, for each process p_i, a set Q_i of quorums is derived, where Q_i={q| $q=\bigcup_{j=1}^{m} q_j$, $q_j \in Cr_j$ and $r_j \in R_i$}. To be more precise, if process p_i accesses resources $r_1,\ldots,r_m$, m>1, then each member q of Q_i is of the form $q=q_1 \cup \ldots \cup q_m$, where $q_1 \in Cr_1,\ldots, q_m \in Cr_m$. Finally, the coterie C_i associated with p_i is derived by removing every non-minimal quorum of Q_i (note that a quorum is non-minimal if it is a superset of another quorum).

Below is a local coterie construction via the majority coterie for the RAG in Fig. 2 of a grid computing environment by the Cheng et al.'s algorithm.

P={p_1, p_2, p_3, p_4, p_5}

R={r_1, r_2}

R_1={r_1, r_2}

R_2= R_4={r_1}

R_3= R_5={r_2}

P_1={p_1, p_2, p_4}

P_2={p_1, p_3, p_5}

Cr_1={{p_1, p_2}, {p_1, p_4}, {p_2, p_4}}

Cr_2={{p_1, p_3}, {p_1, p_5}, {p_3, p_5}}

Q_1={q|$q=q_1 \cup q_2$, for every $q_1 \in Cr_1$ and every $q_2 \in Cr_2$}={{p_1, p_2, p_3}, {p_1, p_3, p_4}, {p_1, p_2, p_3, p_4}, {p_1, p_2, p_5}, {p_1, p_4, p_5}, {p_1, p_2, p_4, p_5}, {p_1, p_2, p_3, p_5}, {p_1, p_3, p_4, p_5}, {p_2, p_3, p_4, p_5}}

Q_2=Q_4= Cr_1

Q_3=Q_5=Cr_2

By removing all non-minimal sets from Q_1, Q_2, …, and Q_5, we have

C_1={{p_1, p_2, p_3}, {p_1, p_3, p_4}, {p_1, p_2, p_5}, {p_1, p_4, p_5}, {p_2, p_3, p_4, p_5}}

C_2=C_4 =Cr_1

C_3=C_5 =Cr_2

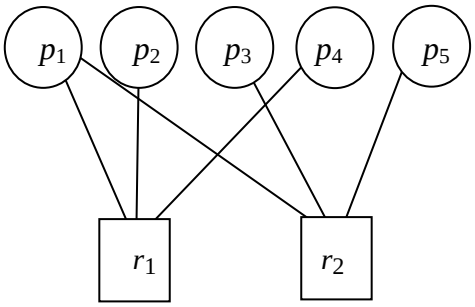

Figure 2. The RAG for the system with $P=\{p_1, p_2,\ldots, p_5\}$ and $R=\{r_1, r_2\}$

We have observed that when a process accesses only one resource, the coterie associated with the process is actually a coterie as defined in [8]. However, when a process accesses more than one resource, the coterie associated with the process has inter-coterie quorum intersection relationship with other coteries. Below, we define a new type of coteries, called *p-coteries*, to capture the inter-coterie quorum intersection relationship.

Definition 3. (p-coterie)
Given m (m>1) and coteries $Cr_1,\ldots,Cr_m$, a *p-coterie* C for $Cr_1,\ldots,Cr_m$ is defined to be a coterie satisfying the following three properties:
Intersection Property: $\forall q_1, q_2 : q_1, q_2 \in C : q_1 \cap q_2 \neq \varnothing$
Inter-coterie Intersection Property: $\forall q, q', j$: $q \in C$, $q' \in Cr_j$, $1 \leq j \leq m$: $q \cap q' \neq \varnothing$.
Minimality Property: $\forall q_1, q_2 : q_1, q_2 \in C : q_1 \not\subset q_2$

We can easily check that a p-coterie for $Cr_1,\ldots,Cr_m$ can be used as a coterie associated with the process accessing resources $r_1,\ldots,r_m$ in Cheng et al.'s algorithm [14], if coteries $Cr_1,\ldots,Cr_m$ are selected respectively to be the coteries for resources $r_1,\ldots,r_m$.

3 Nondomination of Local Coteries

In this section, we focus on the nondomination property of local coteries. A coterie is always better than the coterie it dominates in the sense that if a quorum can be formed in the dominated one then a quorum can be formed in the dominating one [8]. Thus, we should always concentrate on the nondominated (ND) coteries that no coterie can dominate. The local coteries constructed by the algorithms in [13] and [14] are not ND, though.

We thus develop theorems about the nondomination of p-coteries, and propose an operation, called *pairwise-union* (*p-union*), to help generate ND p-coteries. ND p-coteries can then be used to generate ND local coteries for solving the resource allocation problem in Grids and Clouds.

Below we first give the definition of coterie domination.
Definition 4. (coterie domination) [8]
Let C and D be two distinct coteries. C is said to *dominate* D iff $\forall q$, $\exists q'$: $q \in D$, $q' \in C$: $q' \subseteq q$. (We say that q' is the set that dominates q.)

For example, coterie $C=\{\{p_1, p_2\}, \{p_1, p_3\}, \{p_1, p_4\}, \{p_2, p_3, p_4\}\}$ dominates coterie $D=\{\{p_1, p_2, p_3\}, \{p_1, p_2, p_4\}, \{p_1, p_3, p_4\}, \{p_2, p_3, p_4\}\}$ because for every quorum q in D we can find a quorum q' in C such that q is a super set of q'. A dominating coterie, such as C, is always better than a dominated coterie, such as D, since if a quorum can

be formed in the dominated one then a quorum can be formed in the dominating one. A coterie is *nondominated* (ND) if no other coterie can dominate it. ND coteries are candidates to achieve the highest availability, which is the probability that a quorum can be formed in an error-prone environment. Thus, we should always concentrate on ND coteries if fault-tolerance is one of the main concerns. Some classes of coteries, such as the majority, the tree, the hierarchical, the composite, the level, the Lovasz, and the cohorts coteries, have been shown to be ND.

Theorem 1 in the following is developed by Garcia-Molina and Barbara in [8]. This theorem is useful to check if a coterie is dominated or not.

Theorem 1. Let C be a coterie under P. Then, C is dominated iff there exists a set $x \subseteq P$ such that
L1. $\forall q : q \in C: q \not\subseteq x$.
L2. $\forall q : q \in C: q \cap x \neq \emptyset$.

Following the definition of coterie domination, we give the definition of local coterie domination below.

Definition 5. (local coterie domination)
Let $C=(C_1,\ldots,C_n)$ and $D=(C_1',\ldots,C_n')$ be two distinct local coteries. C is said to *dominate* D iff $C_i=C_i'$ or C_i dominates C_i', for $1 \le i \le n$ (i.e., every coterie in C equals or dominates its corresponding coterie in D).

For example, let local coterie C be ($\{\{p_1\}\}$, $\{\{p_1, p_3\}\}$, $\{\{p_3\}\}$) and local coterie D be ($\{\{p_1, p_2\}\}$, $\{\{p_1, p_2, p_3\}\}$, $\{\{p_3\}\}$). We can see that C dominates D since $\{\{p_1\}\}$ dominates $\{\{p_1, p_2\}\}$, $\{\{p_1, p_3\}\}$ dominates $\{\{p_1, p_2, p_3\}\}$, and $\{\{p_3\}\}$ equals $\{\{p_3\}\}$.

By Definition 5, the domination of two distinct local coteries is based on the domination (or equality) of each pair of corresponding coteries. When a process accesses only one resource, we can apply Theorem 1 to check the domination of the coterie associated with the process since the coterie is exactly the same as defined in [8]. When a process accesses more than one resource, the coterie associated with the process is a p-coterie. Below we give the definition of p-coterie domination, which is similar to that of coterie domination.

Definition 6. (p-coterie domination)
Let C and D be two distinct p-coteries for m ($m>0$) given coteries $Cr_1,\ldots,Cr_m$. C is said to *dominate* D iff $\forall q, \exists q'$: $q \in D$, $q' \in C$: $q' \subseteq q$.

We have the following theorem for checking the domination of a p-coterie. Note that by definition a p-coterie is also a coterie, which is a fact used in the proof of Theorem 2.

Theorem 2. Let C be a p-coterie for coteries $Cr_1,\ldots,Cr_m$, $m>1$. C is dominated if and only if there exists a set x such that
L1. $\forall q$: $q \in C$: $q \not\subseteq x$
L2. $\forall q$: $q \in C$: $q \cap x \neq \emptyset$
L3. $\forall q \forall j : q \in Cr_j$, $1 \le j \le m$: $q \cap x \neq \emptyset$
Proof :
(if part)

We first show that L1, L2 and L3 imply C is dominated. There are two cases to consider. Case 1: If there are one or more $q_1,\ldots,q_l \in C$ such that $x \subset q_1,\ldots,x \subset q_l$, then construct set $S=(C-q_1-\ldots-q_l) \cup \{x\}$. It is easy to see that S is a

p-coterie for $Cr_1,\ldots,Cr_m$ and S dominates C. Case 2: If there are no supersets of x in C, then $S=C\cup\{x\}$ is a p-coterie for $Cr_1,\ldots,Cr_m$ and S dominates C.

(only if part)

Now, assume that C is dominated by D, we show that conditions L1, L2 and L3 hold by considering two cases. Case 1: $C\subset D$. Let x be one of the elements in $D-C$. Set x must satisfy conditions L1, L2 and L3 or else D would not be a valid p-coterie for $Cr_1,\ldots,Cr_m$. Case 2: $C\not\subset D$. In such a case, there must be a set $q\in C$ and a set $x\in D$ such that $x\subset q$ (see Definition 4). If condition L1 is false for x, then $q'\subseteq x$ for some $q'\in C$ and C is not a coterie because $q' \subseteq x \subset q$. Similarly, if condition L2 doesn't hold for x, then D would not be a coterie because $\neg$L2 implies $\exists q'$: $q'\in C$: $q'\cap x=\emptyset$, which in turn implies $x\cap x'=\emptyset$, where x' is the set in D that dominates q'. If condition L3 doesn't hold for x, then D would not be a p-coterie for $Cr_1,\ldots,Cr_m$ because $\neg$L3 implies $\exists q'$: $q'\in Cr_j$: $q'\cap x=\emptyset$ for some Cr_j, $1\le j\le m$, which in turn implies $x\cap x'=\emptyset$, where x' is the set in D that dominates q'. We can see that either in case 1 or in case 2, the conditions L1, L2 and L3 should hold. ■

Below, we propose an operation, denoted by ⊗ and called *pairwise-union* (*p-union*, for short), to generate p-coteries for a list of coteries. As will be shown later, we can apply p-union operation on ND coteries to generate ND p-coteries for the construction of ND local coteries.

Definition 7. (pairwise-union operation)

Let P_1 and P_2 be two non-empty sets of processes. Also let G be a coterie under P_1, and H be a coterie under P_2. The *pairwise-union* (*p-union*) operation ⊗ of G and H is defined to be

$G\otimes H=\{g\cup h \mid g\in G, h\in H\}$.

For example, let $G=\{\{p_1, p_2\},\{p_2, p_3\},\{p_1, p_3\}\}$ be a coterie under $P_1=\{p_1, p_2, p_3\}$ and $H=\{\{p_2, p_3\},\{p_3, p_4\},\{p_2, p_4\}\}$ be a coterie under $P_2=\{p_2, p_3, p_4\}$. Then $G\otimes H=\{ \{p_1, p_2, p_3\}, \{p_1, p_2, p_3, p_4\}, \{p_1, p_2, p_4\}, \{p_2, p_3\}, \{p_2, p_3, p_4\}, \{p_2, p_3, p_4\}, \{p_1, p_2, p_3\}, \{p_1, p_3, p_4\}, \{p_1, p_2, p_3, p_4\} \}$.

Let $F=Min(G\otimes H)$, where $Min(Q)$ is a function to eliminate non-minimal quorums from a family Q of quorums. The following Theorem 3, Theorem 4 and Theorem 5 are about properties of F.

Theorem 3. Let P_1 and P_2 be two non-empty sets of processes. If G is a coterie under P_1 and H is a coterie under P_2, then $F=Min(G\otimes H)$ is a p-coterie for G and H under $P_1\cup P_2$.

Proof:

The minimality property is satisfied after *Min*() function is applied. Thus, to prove the theorem, we only have to show (F1)$\forall f, \forall f'$: $f, f'\in F$: $f\cap f'\neq\emptyset$ (F2) $\forall f, \forall g$: $f\in F, g\in G$: $f\cap g\neq\emptyset$ (F3) $\forall f, \forall h$: $f\in F, h\in H$: $f\cap h\neq\emptyset$.

Let f and f' be two sets in F. We have $f=(g\cup h)$ for some $g\in G$ and some $h\in H$, and $f'=(g'\cup h')$ for some $g'\in G$ and some $h'\in H$. Assume $f\cap f'=\emptyset$. It follows that $(g\cup h)\cap(g'\cup h')=\emptyset$. And hence, we have $g\cap g'=\emptyset$ and $h\cap h'=\emptyset$, which contradicts the fact that G and H are coteries. So, the condition (F1) holds.

Let f be a set in F. We have $f=(g\cup h)$ for some $g\in G$ and some $h\in H$. Assume $f\cap g'=\emptyset$ for some $g'\in G$. It follows that $(g\cup h)\cap g'=\emptyset$. We have $g\cap g'=\emptyset$, which contradicts the fact that G is a coterie. Thus, the condition (F2) holds.

Let f be a set in F. We have $f=(g\cup h)$ for some $g\in G$ and some $h\in H$. Assume $f\cap h'=\emptyset$ for some $h'\in H$. It follows that $(g\cup h)\cap h'=\emptyset$. We have $h\cap h'=\emptyset$, which contradicts the fact that H is a coterie. Thus, the condition (F3) holds. ■

In Theorem 3, G and H are taken to be coteries. However, G and H in Theorem 3 can also be taken to be p-coteries because a p-coterie is also a coterie. Below, we apply Theorem 3 with G being a p-coterie to prove the following Theorem 4.

Theorem 4. Let $P_1,\ldots,P_m$, $m>1$, be non-empty sets of processes. If Cr_j is a coterie under P_j, $1\le j\le m$, then $Min(Cr_1\otimes\ldots\otimes Cr_m)$ is a p-coterie for $Cr_1,\ldots,Cr_m$ under $P_1\cup\ldots\cup P_m$.

Proof : (by induction on the value of m)

(1) Basis: (m=2)

By Theorem 3, the basis case holds.

(2) Induction hypothesis:

Assume that if Cr_j is a coterie under P_j for $1\le j\le m$, then $G=Min(C_1\otimes\ldots\otimes C_m)$ is a p-coterie for $Cr_1,\ldots,Cr_m$ under $P_1\cup\ldots\cup P_m$.

(3) Induction step:

On the basis of the induction hypothesis, below we show that if Cr_j is a coterie under P_j for $1\le j\le m+1$, then $F=Min(Cr_1\otimes\ldots\otimes Cr_{m+1})$ is a p-coterie for $Cr_1,\ldots,Cr_{m+1}$ under $P_1\cup\ldots\cup P_{m+1}$.

Let G be $Min(Cr_1\otimes\ldots\otimes Cr_m)$. Then $F=Min(G\otimes Cr_{m+1})$. Since G is a p-coterie for $Cr_1,\ldots,Cr_m$ under $P_1\cup\ldots\cup P_m$ (by the induction hypothesis) and Cr_{m+1} is a coterie under P_{m+1}, we have F is a p-coterie for G and Cr_{m+1} under $P_1\cup\ldots\cup P_{m+1}$ by Theorem 3. Because G is a p-coterie for $Cr_1,\ldots,Cr_m$, each quorum in G intersects every quorum in $Cr_1,\ldots,Cr_m$. And because $F=Min(G\otimes Cr_{m+1})$, any quorum f in F must be of the form $f=g\cup q$, where $g\in G$ and $q\in Cr_{m+1}$. It follows that each quorum in F intersects every quorum in $Cr_1,\ldots,Cr_{m+1}$. Hence, we have that F is a p-coterie for $Cr_1,\ldots,Cr_{m+1}$ under $P_1\cup\ldots\cup P_{m+1}$.

Therefore, by the induction principle, we have $Min(C_1\otimes\ldots\otimes C_m)$ is a p-coterie for $Cr_1,\ldots,Cr_m$ under $P_1\cup\ldots\cup P_m$ for $m>1$. ■

The following Theorem 5 is about the nondomination of the p-coteries generated by the p-union operation.

Theorem 5. Let $P_1,\ldots,P_m$, $m>1$, be non-empty sets of processes. Also let Cr_j be a coterie under P_j for $1\le j\le m$, and $F=Min(Cr_1\otimes\ldots\otimes Cr_m)$ be a p-coterie for $Cr_1,\ldots,Cr_m$ under $P_1\cup\ldots\cup P_m$. Then, F is ND if $Cr_1,\ldots,Cr_m$ are all ND.

Proof:

Assume F is dominated, then by Theorem 2, there must exist a set $x\subseteq(P_1\cup\ldots\cup P_m)$ such that (L1) $\forall f: f\in F: f\not\subseteq x$, (L2) $\forall f: f\in F: f\cap x\neq\emptyset$, (L3) $\forall q\forall j: q\in Cr_j$, $1\le j\le m$: $q\cap x\neq\emptyset$.

Let $x_1=x\cap P_1$, $x_2=x\cap P_2$, …, and $x_m=x\cap P_m$. Then, we have $\forall q$: $q\in Cr_1$: $q\cap x_1\neq\emptyset$ because $q\cap x_1=q\cap x\cap P_1\neq\emptyset$ by (L3) and $(q\cap x)\subseteq P_1$. Similarly, we have $\forall q$: $q\in Cr_2$: $q\cap x_2\neq\emptyset$ because $q\cap x_2=q\cap x\cap P_2\neq\emptyset$ by (L3) and $(q\cap x)\subseteq P_2$. … And we have $\forall q$: $q\in Cr_m$: $q\cap x_m\neq\emptyset$ because $q\cap x_m=q\cap x\cap P_m\neq\emptyset$ by (L3) and $(q\cap x)\subseteq P_m$. To sum up, we have $\forall q\forall j$: $q\in Cr_j$, $1\le j\le m$: $q\cap x_j\neq\emptyset$.

Suppose $\forall q$: $q\in Cr_1$: $q\not\subseteq x_1$. Then, we have $Cr_1\cup\{x_1\}$ is a coterie dominating Cr_1, which contradicts the fact that Cr_1 is ND. It follows that $\exists q_1$: $q_1\in Cr_1$: $q_1\subseteq x_1$. We can proceed with the same inference to have $\exists q_2$: $q_2\in Cr_2$: $q_2\subseteq x_2$, …, and $\exists q_m$: $q_m\in Cr_m$: $q_m\subseteq x_m$. It follows that $(q_1\cup\ldots\cup q_m)\subseteq x$ since $(q_1\cup\ldots\cup q_m)\subseteq(x_1\cup\ldots\cup x_m)=(x\cap P_1)\cup\ldots\cup(x\cap P_m)\subseteq x$. Because $F=Min(Cr_1\otimes\ldots\otimes Cr_m)$, we have $\exists f$: $f\in F$: $f\subseteq(q_1\cup\ldots\cup q_m)$ by $\otimes$ operation definition. We then have $\exists f$: $f\in F$: $f\subseteq(q_1\cup\ldots\cup q_m)\subseteq x$, which contradicts (L1).

The assumption that F is dominated cannot stand. Hence, the theorem holds. ■

Note that we do not know whether the "only if" part of Theorem 5 (i.e., F is ND only if $Cr_1,\ldots,Cr_m$ are all ND) is true or not; we leave it as an open problem. Fortunately, Theorem 5 itself is sufficient to guide us to derive ND p-coteries for the construction of ND local coteries.

4 Conclusion

This paper has defined a new type of coteries, called *p-coteries*, to aid the construction of local coteries. It has also described and developed theorems about the nondomination of p-coteries, and proposed an operation, called *pairwise-union* (*p-union*), to help generate ND p-coteries. By the theorems and the operation, ND local coteries can be constructed on the basis of known ND coteries, such as the majority coterie, tree coterie, hierarchical coterie, composite coterie, level coterie, Lovasz coterie or cohorts coterie, to solve the resource allocation problem in Grids and Clouds. Since the constructed local coterie *LC* is ND, it is toward the optimality in the sense that no other local coterie can dominate it.

References

[1] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations. International," *Journal Supercomputer Applications*, 15(3):200–222, 2001.

[2] F. Chang, J. Ren, and R. Viswanathan, "Optimal resource allocation in clouds," in *Proc. of IEEE 3rd International Conference on Cloud Computing (CLOUD)*, pp. 418-425, 2010.

[3] E. W. Dijkstra, "Solution to a problem in concurrent programming control," *CACM*, 8(9):569, 1965.

[4] M. Fisher, N. Lynch, J. Burns and A. Borondin, "Resource allocation with immunity to limited process failure," in *Proc. of the 20th IEEE annual symposium on foundations of Computer Science*, pp. 234–254, 1979.

[5] M. Raynal, "A distributed solution for the *k*-out of-*m* resources allocation problem," *Lecture Notes in Computer Sciences*, Springer Verlag, 497:599-609, 1991.

[6] E. W. Dijkstra, "Hierarchical ordering of sequential processes," *Acta Informatica*, 1:115-138, 1971.

[7] K. M. Chandy and J. Misra, "The drinking philosophers problem," *ACM Transactions on Programming Languages and Systems*, 6(4):632-646, 1984.

[8] H. Garcia-Molina and D. Barbara, "How to assign votes in a distributed system," *JACM*., 32(4):841-860, 1985.

[9] J.-R. Jiang, "Fault-tolerant distributed mutual exclusion with O(1) message overhead," in *Proc. of the 13th International Conference on Applied Informatics*, pp.228-231, 1995.

[10] J.-R. Jiang and S.-T. Huang, "Obtaining nondominated *k*-coteries for fault-tolerant distributed *k*-mutual exclusion," in *Proc. of 1994 IEEE International Conference on Parallel and Distributed Systems*, pp. 582–587, 1994.

[11] J.-R. Jiang, S.-T. Huang and Y.-C. Kuo, "Cohorts structures for fault-tolerant *k* entries to a critical section," *IEEE Transactions on Computers*, 48(2):222-228, 1997.

[12] J.-R. Jiang, "Distributed *h*-out of-*k* mutual exclusion using *k*-coteries," in *Proc. of the 3rd International Conference on Parallel and Distributed Computing, Application and Technologies* (*PDCAT'02*), pp. 218-226, 2002.

[13] H. Kakugawa and M. Yamashita, "Local coteries and a distributed resource allocation algorithm," *Transactions of Information Processing Society of Japan*, 37(8):1487-1496, 1996.

[14] Z. Cheng, Y. Wada, S. Hashimoto, A. He and T. Huang, "A new method for constructing efficient local coteries," in *Proc. of the 15th International Conference on Information Networking*, pp. 512 –517, 2001.

[15] Jehn-Ruey Jiang, "Quorum structures for fault-tolerant distributed mutual exclusion," *Ph.D. Dissertation*, Tsing-Hua University, 1995.

[16] M. Maekawa, "A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems," *ACM Transactions on Computer Systems*, 3(2): 145- 159, 1985.