Table of Contents

[**LCC Informal Description** 2](#_Toc60644387)

[**LCC Formal Description** 3](#_Toc60644388)

[**Constraint Optimization Problem (COP)** (Ferdinando, Enrico, & Yeoh, 2018) 3](#_Toc60644389)

[**Definition** 3](#_Toc60644390)

[**Implications** 3](#_Toc60644391)

[**Distributed Constraint Optimization Problem (DCOP)** (Ferdinando, Enrico, & Yeoh, 2018) 3](#_Toc60644392)

[**Definition** 3](#_Toc60644393)

[**Implications** 4](#_Toc60644394)

[**Asymmetric Distributed Constraint Optimization Problem (ADCOP)** (Ferdinando, Enrico, & Yeoh, 2018) 4](#_Toc60644395)

[**Definition** 4](#_Toc60644396)

[**Implications** 5](#_Toc60644397)

[**Multi-Objective Distributed Constraint Optimization Problem (MODCOP)** (Ferdinando, Enrico, & Yeoh, 2018) 5](#_Toc60644398)

[**Definition** 5](#_Toc60644399)

[**Implications** 6](#_Toc60644400)

[**Coalition Structure Generation Problem (CSG)** (Präntare & Heintz, 2020) 6](#_Toc60644401)

[**Definition** 6](#_Toc60644402)

[**Implications** 6](#_Toc60644403)

[**Projection** 7](#_Toc60644404)

[**Constrained KMeans Clustering Problem (CKMeans)** (Wagstaff, Cardie, Rogers, & Schroedl, 2001) 9](#_Toc60644405)

[**Definition** 9](#_Toc60644406)

[**Implications** 9](#_Toc60644407)

[**Projection** 10](#_Toc60644408)

[**Distributed Constraint KMeans Problem (DCKMeans)** (Cornuéjols, Wemmert, Gançarski, & Bennani, 2018) 12](#_Toc60644409)

[**Definition** 12](#_Toc60644410)

[**Implications** 13](#_Toc60644411)

[**References** 13](#_Toc60644412)

# **LCC Informal Description**

One of the main challenges in the Catalan language course of the First Reception Service is meeting the different learning speeds of the students (TCNs). To do that, the teachers usually divide the class into smaller **working groups**. A course consists of multiple consecutive lessons (classes), one per day in the course period.

In WELCOME, the language course coordination (LCC) scenario refers to a language course of, **initially**, **three** consecutive lessons (or classes). In each lesson, every student performs exercises of four types of Catalan language learning activities related to the **reading**, **writing**, **grammar**, and **vocabulary** of Catalan. Each student is assigned to one working group per lesson. **At the end of each lesson**, the progress made by the students is individually assessed through means of a specific set of language learning assessment exercises for each of the four types of learning activities. Language **speaking activities** are not considered in WELCOME. The grouping of students per lesson is usually made by the teacher upon various factors, both **course** and **individual constraints**, and depending on the overall **course progress level** of individual students.

In this context, to facilitate the intervention of the teacher, it would be very helpful if the developed WELCOME technology can support the teacher in her language course coordination. The **objective** of the agents of the students would be to coordinate among each other for finding an (approximately) optimal assignment of students to groups with similar level of course progress. In particular, the WELCOME platform should allow to assess the progress of each of the students and make grouping suggestions to the teacher at the end of each lesson based on the following constraints:

1. **Individual constraints** specified by the students (TCNs)
2. **Gender**: Gender can be a major constraint for the participation of women in the language learning course. Although this limitation can be considered less important in large groups (such as a class), when working in smaller groups, female TCNs, who do not want to interact with the other gender, are less likely to participate freely in the exercises of the lessons if they are grouped into a mixed-gender group. This can also happen to male-TCNs, although it is less likely to happen.

Therefore, the agent should respect the preference of its TCN about being assigned to a mixed-gender group or not. In fact, this individual constraint **must be satisfied**.

1. **Nationality**. Although many people do not have a problem when working with other nationalities, some of the TCNs might have. To ensure the progress of all members of a group in a class, the agent should respect the preference of its TCN about being assigned to a group with other nationalities than her own one. This constraint, however, **may be ignored** for sake of finding an overall grouping solution.
2. **Course constraints** to be satisfied by the coordinating agents of the TCNs
   1. Each class (lesson) has at least **two** and at most **twenty** students
   2. Each group in a lesson has at least **two** and at most **five** students
   3. Each group has students with **similar** course progress level
   4. Each group should have at most **one** student who **missed the preceding** lesson.

The **progress assessment scheme** to be used by the agents for their finding of an optimal grouping of students is as follows:

* The language learning skill of a student is measured at the end of each lesson of the course as the equally weighted average sum of percentages of correct answers for each language learning activity (reading, writing, grammar, vocabulary) given by the student in the lesson.
  + The *language learning assessment* score (LLA-X) for a given type X of language learning activity (X in {Reading, Writing, Grammar, Vocabulary}) denotes the percentage of correct answers for assessment exercises of this type by the student. An LLA-X score in the interval of 0-49, 50-59, 60-69, 70-79, 80-89, 90-100 is interpreted as “insufficient”, “sufficient”, “good”, “remarkable”, respectively, “outstanding” progress, respectively. Besides, the time a student spent to complete the assessment exercises of a given type of language learning activity in the interval of 0-5, 5-10, 10+ minutes is interpreted as „Student is doing OK“, „Student is facing difficulties“ and „Student is struggling“, respectively, in this regard.
  + The *language lesson* (LL) score of an individual student is defined as the equally weighted sum of the LLA-X scores she obtained for performing assessment exercises for the selected four types of language learning activities in multiple-choice form.

The *course progress level* (CPL) score for a student is determined as the averaged sum of the LL scores for all lessons she attended so far.

# **LCC Formal Description**

In this section, different kind of problems are shortly described to compare with LCC and to identify which one fits best.

## **Constraint Optimization Problem (COP)** (Ferdinando, Enrico, & Yeoh, 2018)

### **Definition**

COPs consist of variables, values and weighted constraints. COP solving algorithms assign values to variables such that the constraints are satisfied as much as possible. Since it is not always possible to satisfy all of the constraints, COP allows some constraints to be violated to some extent.

### **Implications**

COP time complexity is exponential which is not acceptable for LCC scenario. As well as COPs are solved in central manner. Distributed version of COP is in the next section.

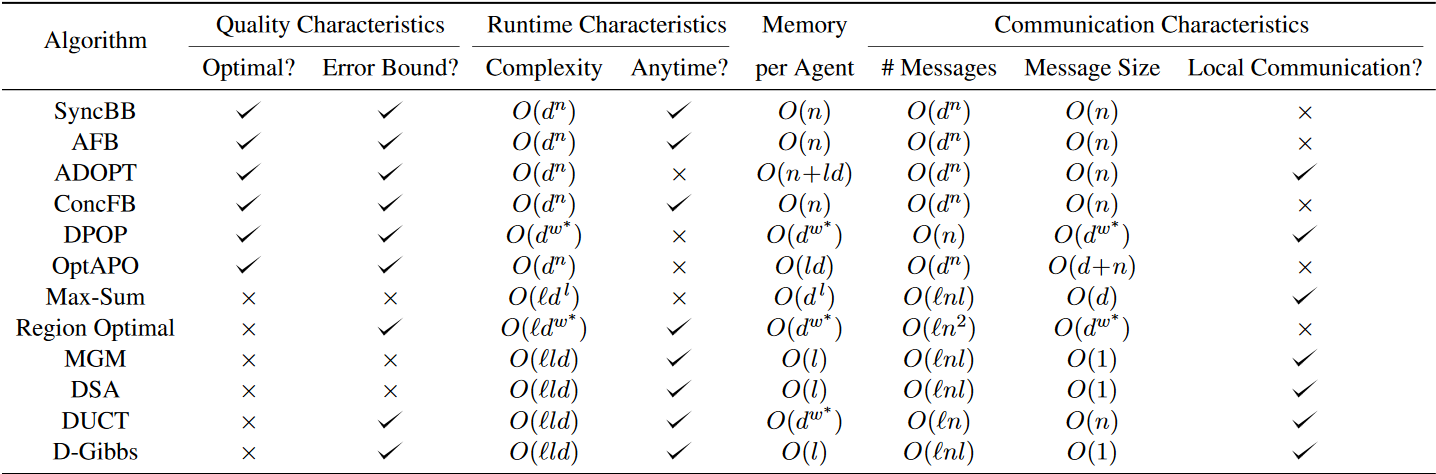
## **Distributed Constraint Optimization Problem (DCOP)** (Ferdinando, Enrico, & Yeoh, 2018)

### **Definition**

DCOP is an extension of the COP framework to the multi-agent case, where agents control variables and constraints and need to coordinate the value assignment for the variables they control so as to optimize a global objective function. A DCOP is a tuple (A, X, D, F, α), where:

* A is a finite set of agents
* X is a finite set of variables
* D is a finite set of domain sets. Each domain set corresponds to one variable in X.
* F is a finite set of cost functions.
* α is a function which assigns the control of each variable to an agent.

Finding an optimal solution for DCOP is **NP-HARD problem**.



**Figure 1**. List of DCOP algorithms (Ferdinando, Enrico, & Yeoh, 2018)

* **n** refers to the number of variables
* **d** refers to the size of the largest domain
* **w\***refers to the induced with of the pseudo-tree
* **l** refers to the largest number of neighboring agents
* **i** refers to the number of iterations in incomplete algorithms

### **Implications**

Like COP, DCOP doesn’t formulate/generate clusters by nature. In DCOP, each agent assigns predefined values, to its variable/variables in a distributed/decentralized manner. Therefore, the groups which the TCNs can be assigned to are given as the domain for each TCN (variable).

Additionally, all agents controlling variables which are in the same cost function, incur the same cost as each other.

Figure 1 illustrates the list of DCOP algorithms with their properties such as optimality, anytime, computational and communication complexities etc.

In the worst case, all optimal DCOP algorithms will take exponential time: **O(dn)**.

So if there are 20 TCNs and each group can contain at least 2 TCNs, then in the worst case there will be 10 groups to cover all 20 TCNs. That means O(1020) runtime complexity for the optimal DCOP algorithms.

Therefore, DCOP doesn’t fit to LCC problem. Since ADCOP and MODCOP solving algorithms inherit DCOP algorithms, they don’t fit as well. Short information about ADCOP and MODCOP can be found in the next sections.

## **Asymmetric Distributed Constraint Optimization Problem (ADCOP)** (Ferdinando, Enrico, & Yeoh, 2018)

### **Definition**

ADCOP is a tuple (A, X, D, F, α) just like DCOP. In ADCOP, each agent can incur different costs from the same cost function. This is the most important difference between DCOP and ADCOP.

#### **ADCOP Algorithms:**

There are 2 approaches to solve ADCOPs:

* **Two-phase strategy**: where only one side of the constraint (i.e., the cost induced by one agent) is considered in the first phase. The other side(s) (i.e., the cost induced by the other agent(s)) is considered in the second phase once a complete assignment is produced. As a result, the costs of all agents are aggregated.
* **One-phase strategy**: is to systematically check both sides of constraints before reaching a full assignment.

DCOP algorithms are extended with the 2 strategies (one-phase and two-phase) to solve ADCOPs.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Extension of** | **Exact** | **Runtime** | **Memory per Agent** | **Number of Messages** | **Message size** |
| SyncABB-2ph | SyncBB | Yes | O(dn) | O(n) | O(dn) | O(n) |
| SyncABB-1ph | SyncBB | Yes | O(dn) | O(n) | O(dn) | O(n) |
| ATWB | AFB | Yes | O(dn) | O(n) | O(dn) | O(n) |
| ACLS | DSA | No | O(ild) | O(l) | O(inl) | O(l) |
| MCS-MGM | MGM | No | O(ild) | O(l) | O(inl) | O(l) |

**Figure 2**. List of ADCOP algorithms (Ferdinando, Enrico, & Yeoh, 2018)

### **Implications**

Since ADCOP solving algorithms are simply an extended version of DCOP algorithms, they grow exponentially as well. Therefore, LCC cannot be solved as ADCOP.

## **Multi-Objective Distributed Constraint Optimization Problem (MODCOP)** (Ferdinando, Enrico, & Yeoh, 2018)

### **Definition**

MODCOP is basically an extension of DCOP with multiple conflicting objectives where all objectives need to be optimized simultaneously. It is a tuple where A, X, D and are same as ADCOP.

* is a finite set of objective functions.

Therefore, the goal in MODCOP is to minimize the sum of all objective functions but it typically doesn’t achieve to optimize all of them. Therefore, there are tradeoffs between different objectives.

#### **Algorithms**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Extension of** | **Exact** | **Runtime** | **Memory per Agent** | **Number of Messages** | **Message size** |
| MO-SBB | SyncBB | Yes | O(dn) | O(np) | O(dn) | O(n) |
| Pseudo-tree based algorithm | ADOPT | Yes | O(dn) | O(np) | O(dn) | O(n) |
| B-MOMS | Bounded Max-Sum | No | O(idl) | O(pdn) | O(inl) | O(pdn) |
| DP-AOF | AOF | No | O(dw\*) | O(dw\*) | O(hn) | O(dw\*) |
| MO-DPOPLp | DPOP | No | O(dw\*) | O(dw\*) | O(hn) | O(dw\*) |
| DIPLS | Pareto Local Search | No | O(ikp) | O(np) |  | O(np) |

**Figure 3**. MODCOP algorithms (Ferdinando, Enrico, & Yeoh, 2018)

### **Implications**

Since MODCOP can be considered as multiple DCOPs and we said that LCC cannot be solved as DCOP, LCC cannot be solved as MODCOP as well.

## **Coalition Structure Generation Problem (CSG)** (Präntare & Heintz, 2020)

### **Definition**

Coalition Structure Generation (CSG) problem aims to create agent teams to solve problems where single agents cannot solve effectively or cannot solve at all. In CSG, there are three important terms: coalition, coalition value and coalition structure (CS).

1. A coalition refers to the team of agents. In other words, it is a set of agents. Additionally, a coalition can be empty.
2. Coalition value is generated by coalition value generation function. This value is reward and it represents how satisfied the agents are by being in the same coalition.
3. A CS is a set of coalitions. CSs differ based on the number and the content of coalitions they contain. Each CS contains all of the agents.

In order to solve a CSG problem, firstly algorithms produce all possible coalitions where each of them belongs to a CS. Then a reward value for each coalition is calculated. Afterwards, a value for each CS is calculated by summing the coalition values. Finally, since each CS has a value, the algorithm can compare them and produce a rank list of CS from best to worst.

There are exact and approximate CSG solving algorithms and they grow exponentially. For example,

**IP**- O(nn), (Rahwan, Michalak, Elkind, Wooldridge, & R.Jennings, 2009)

**ODP** - O(3n), (Rahwan, Michalak, Elkind, Wooldridge, & R.Jennings, 2009)

**ODP**-**IP** - O(3n), (Rahwan, Michalak, Elkind, Wooldridge, & R.Jennings, 2009)

**BOSS** - O(3n), (Boss)

**FACS** - O(3n). (FACS)

**BOSS** is an exact algorithm and according to authors (Boss), it finds an exact solution for 27 agents in approximately 111 minutes. (Boss)

**FACS** is an approximate algorithm which is much faster than BOSS. (FACS)

### **Implications**

Even though BOSS and FACS grow exponentially, they can be used to solve LCC scenario because of the constraint B.1 which indicates that 20 is the maximum number of TCNs to be in any classroom. In other words, in LCC scenario there will be maximum 20 agents and these two algorithms can find a solution for 20 agents.

Furthermore, there will be at least a day between 2 consecutive lessons in LCC scenario which is more than enough for BOSS to find a solution.

### **Projection**

Formal description of CSG\_LCC with evaluation functions

The application of CSG in the LCC case is divided into the four parts: Coalition Structure Generation, Filtering, Evaluation and Output results.

**Coalition Structure Generation**: Firstly, given the number of the TCN, we are able to permutate all the possible Coalition Structures by FACS or BOSS. Note that according to the course constraints, the number of TCNs should be between two and twenty.

**Filtering**: These coalition structures will then be filtered by

1). Each group in a lesson has at least **two** and at most **five** TCNs;

2). Each group should have at most **one** TCN who **missed the preceding** lesson.

**Evaluation**: Each of the filtered coalition structure will be evaluated by coalition value function:

Where:

This Coalition Structure Value function accesses the quality of each Coalition Structure by taking the consideration of how well the individual preferences of each TCN in the same Coalition are satisfied, and how much the Language Lesson scores of TCNs in the same Coalition are different from each other.

Specifically, to access how well the individual preferences of each TCN in the same coalition are satisfied, the function calculates the times of the gender and nationality preferences have been met for each TCN when assigning to a Coalition. To access how much the Language Lesson scores of TCNs in the same Coalition are different from each other, the function calculates variance of the Language Lesson scores of all the TCNs in the last lesson, and then takes the multiplicative inverse. The value of the coalition is the product of these two parts. The Coalition Structure Value is the sum of all Coalition values.

An example is shown below. The Coalition Structure consists of two coalitions, each has three TCNs:

Coalition 1:

|  |  |  |  |
| --- | --- | --- | --- |
| TCN number | TCN1 | TCN2 | TCN3 |
| Gender preference | Same gender | Same gender | Different gender |
| Nationality preference | Same nationality | Same nationality | Different nationality |
| Gender | Male | Female | Male |
| Nationality | A | A | B |
| Language Lesson Score | 85 | 70 | 75 |

Coalition 2:

|  |  |  |  |
| --- | --- | --- | --- |
| TCN number | TCN4 | TCN5 | TCN6 |
| Gender preference | Different gender | Same gender | Same gender |
| Nationality preference | Different nationality | Different nationality | Different nationality |
| Gender | Female | Female | Male |
| Nationality | A | B | C |
| Language Lesson Score | 80 | 80 | 85 |

For Coalition 1, TCN 1 has the gender preference of same gender and the nationality preference of same nationality. Given the gender and nationality information of other TCNs in the same coalition, this TCN’s gender preference is satisfied once (TCN 3 is male as well), the nationality preference is satisfied once (TCN 2 has the same nationality). Then the product is: . This number means in total there are 2 times where TCN 1’s individual preference has been satisfied. Similarly, the number for TCN 2 is , the number for TCN 3 is . Hence the sum is .

The variance of the Language Lesson scores of all the TCNs from the last lesson is .

Hence the value for coalition 1 is: .

For Coalition 2, the times where TCN 4’s individual preference has been satisfied is ; the times where TCN 5’s individual preference has been satisfied is ; the times where TCN 6’s individual preference has been satisfied is . The sum is .

The variance of the Language Lesson scores of all the TCNs from the last lesson is .

Hence the value for coalition 2 is: .

Consequently, the value of this Coalition Structure is: .

**Output Results**: from the previous steps, we have the Coalition Structures and their corresponding values. Then we find the Coalition Structure with the largest score, this Coalition Structure is the result to output.

## **Constrained KMeans Clustering Problem (CKMeans)** (Wagstaff, Cardie, Rogers, & Schroedl, 2001)

### **Definition**

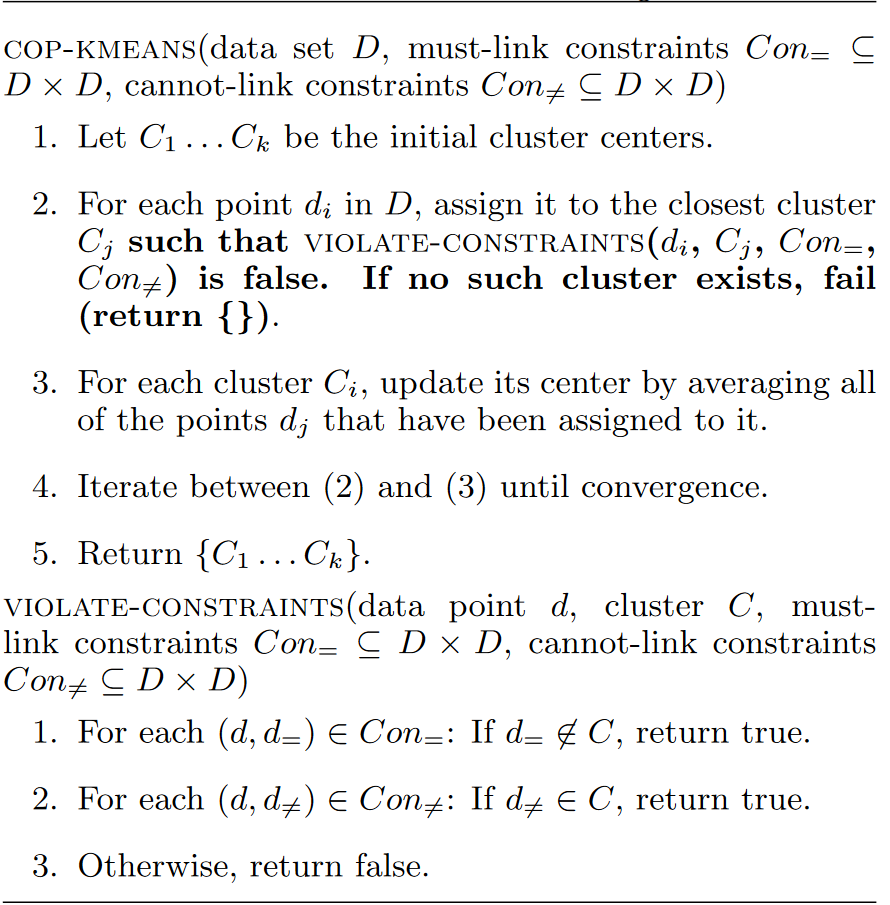
Clustering is an unsupervised method for data analysis. In clustering, the data points are grouped based on a similarity unit. Consequently, the data points are similar intra-group and dissimilar inter-group.

There are some clustering methods such as hierarchical clustering, kmeans clustering and etc. In KMeans clustering, the number of groups is required to be known beforehand which is represented by K. So in KMeans clustering, algorithm divides the data points into k clusters.

Constrained clustering is a method which takes into account the constraints between data points. These constraints are represented either as must-link or cannot-link.

So the definition of constrained kmeans clustering is “it divides the data points into k clusters while ensuring that the constraints (must-link and cannot-link) are not violated”. Additionally, it is a polynomial time algorithm unlike the ones we talked about so far.

CKMeans is also called as COP-KMeans and the algorithm is described in **figure 4**.



**Figure 4**. Constrained K-means algorithm (Wagstaff, Cardie, Rogers, & Schroedl, 2001)

### **Implications**

The definition of the CKMeans algorithm describes what we want to achieve in LCC scenario which is dividing TCNs into small groups while satisfying their personal preferences. Therefore, LCC can be solved with CKMeans clustering algorithm. Since the algorithm is polynomial, the solution of LCC might be found faster compared to exponential algorithms (BOSS&FACS).

### **Projection**

CKMeans algorithm for LCC scenario is described below:

----------------------------------------------------------------------------------

**COP-Kmeans** (TCN **data set** D, **must-link constraints** , **cannot-link constraints** )

1. Calculate **K** interval:
2. If are initial cluster centers and **clusterinterval** is cluster interval, then the cluster center value for cluster **i** can be calculated as following:

For example, if k=4, then clusterinterval = 25. Therefore, C1 = 12.5, C2 = 37.5, C3 = 62.5, C4 = 87.5. Consequently, C1, C2, C3 and C4 covers 0-25, 26-50, 51-75, and 76-100 respectively.

1. For each point , assign it to the closest cluster (constraint **B.3**) such that is **false**. If **no** such cluster **exist**, increment by one and check whether is not greater than . If incremented is not greater than , then **break** iteration and start again from **(2)**. If it is greater, then **fail** (return {}).
2. For each cluster , update its center by **averaging** all of the points that have been assigned to it.
3. Iterate between **(3)** and **(4)** until **convergence**.
4. Return
5. **Cardinality constraint (**constraint **B.2)**:

**If |C| > 4**, return **true**.

Meaning that if there are maximum 4 elements in C, then return true such that when d is assigned to C, there will be 5 elements in C.

1. **Gender constraint (**constraint **A.1)**:

In order to easily implement this constraint, each cluster is assumed to have an attribute called “*gender*”. The attribute “*gender*” can take 3 values: {0, 1, and 2}.

0: represents *mixed*

1: represents *male*

2: represents *female*

Function *g* maps TCN *d* into his/her preferred gender. *0, 1, 2* represents *mixed*, *male* and *female* respectively.

Function *G* maps cluster *C* into its gender attribute *(0, 1, 2).*

So , return **true**.

Meaning that if gender preference of TCN *d* is not same as the gender of cluster *C*, then constraint is violated and therefore return **true**.

1. **Attendance constraint (**constraint **B.4)**:

In order to easily implement this constraint, each cluster is assumed to have an attribute called “*missed*”. This attribute represents the number of students in the cluster who missed the previous lesson. It can take an **integer** value such as 0, 1, 3, 5 and etc.

According to constraint B.4, there should be maximum 1 student who missed the previous lesson.

Function *m* maps TCN d into 0 or 1.

0: if TCN didn’t missed the previous lesson

1: if TCN missed the previous lesson

Function M maps cluster C into an integer number which represents the amount of students in C who missed the previous lesson.

So **,** return **true.**

Meaning that if TCN d missed the previous lesson and there is already someone who missed the previous lesson in the cluster, then the constraint is violated.

1. **Nationality constraint (**constraint **A.2)**:

In order to easily implement this constraint, each cluster is assumed to have an attribute called “**nationality**”. This attribute represents the nationality of the cluster members. It takes a string value either “*mixed*” or a *nationality* such as Syrian, Arabic, Greek and etc.

According to constraint A.2, this constraint can be violated. Therefore, we introduce a variable **v** which represents the violation rate regarding TCNs’ nationality preferences. The variable v takes values between 0 and 100. For example, if v=45 and there are 4 TCNs in the cluster, then less than (4\*45)/100 = 1.8 violations are allowed.

However, in order for the violation rate to be useful, we need to know the amount of violations in the cluster C. Therefore, each cluster is assigned with an attribute called “***nationalityviolations***” which represents the number of nationality violations in the cluster.

Function o maps TCN d into his/her preferred nationality/origin either *mixed* or his/her own *nationality.*

Function O maps cluster C into its nationality/origin attribute.

So **,** return **true**

Meaning that if the nationality preference of TCN d is not satisfied and the number of violations exceed the violation rate, then the constraint is violated.

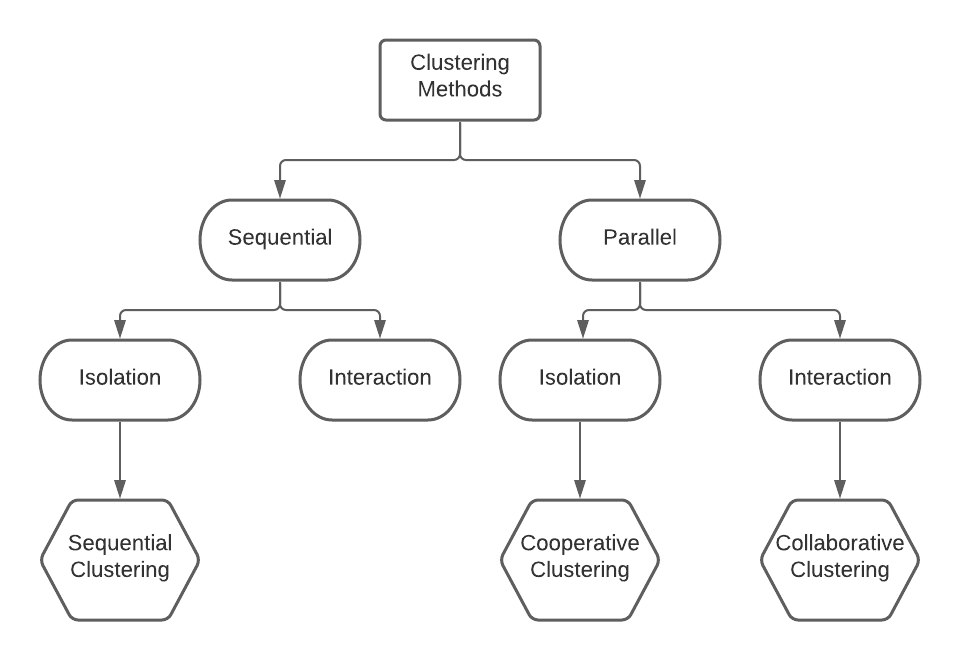
1. If none of the above conditions hold, then return **false**.

----------------------------------------------------------------------------------

## **Distributed Constraint KMeans Problem (DCKMeans)** (Cornuéjols, Wemmert, Gançarski, & Bennani, 2018)

### **Definition**

DCKMeans is used to combine clusterings to get a better overall clustering solution. There are 3 types: 1) **Sequential** clustering, 2) **Cooperative** clustering and 3) **Collaborative** clustering. These methods are used when there is one algorithm and different inputs or there is one input but different algorithms.



**Figure 5**. Classification of clustering methods (Cornuéjols, Wemmert, Gançarski, & Bennani, 2018)

1. **Sequential Clustering**: each algorithm (agent) in turn may use information provided by the previous clustering algorithm (agent) while possibly exploiting new additional data. The convergence point needs to be determined.
2. **Cooperative Clustering**: each algorithm runs independently of others and when all algorithms find a local solution, a master algorithm (master agent) takes control and combines the local results. The challenge for master algorithm (master agent) is to combine clusterings when they differ in the number of clusters and etc.
3. **Collaborative Clustering**: algorithms share information to find a better consensus clustering. In other words, they work together. Each collaborator might have same algorithm and different input, same algorithm and same input, different algorithm and same input, or different algorithm and different input. It is difficult to ensure that collaboration can bring improvement locally and it is also difficult to control the process.

* Higher number of algorithm executions can bring better results if they collaborate.
* If each algorithm had different input, then more data could result in a better local clustering solution.
* There is no guarantee that collaboration between two agents will improve both of their local clusterings.

### **Implications**

CKMeans algorithm is a central solution for LCC scenario. However, a distributed solution is necessary since agents work autonomously and they aren’t supposed to share the preferences of their TCN. In other words, each agent has the information about all other agents such as age, gender, nationality and etc. But agents don’t know the preferences of other agents. Therefore, each agent will find the best clustering for itself based on its preferences. Since *Cooperative* or *Collaborative* clustering methods are able to combine clusterings, they should be fine to solve the LCC in a distributed manner in multi-agent system.

# **References**

Boss. (n.d.). BOSS: A Bi-directional Search Technique for Optimal Coalition StructureGeneration with Minimal Overlapping. 8.

Cornuéjols, A., Wemmert, C., Gançarski, P., & Bennani, Y. (2018). Collaborative Clustering: Why, When, What and How. *formation Fusion, Elsevier*, 39.

FACS. (n.d.). FACS: Fast code-based Algorithm for Coalition Structure Generation. *AAAI21*.

Ferdinando, F., Enrico, P., & Yeoh, W. (2018). Distributed Constraint Optimization Problemsand Applications: A Survey. *Artificial Intelligence Research 61*, 76.

Grinshpoun, T., Grubshtein, A., Zivan, R., Netzer, A., & Meisels, A. (2013). Asymmetric Distributed Constraint Optimization Problems. *Journal of Artificial Intelligence Research 47* , 613-647.

Präntare, F., & Heintz, F. (2020). An anytime algorithm for optimal simultaneous coalitionstructure generation and assignment. *Autonomous Agents and Multi-Agent Systems*.

Rahwan, T., Michalak, T., Elkind, E., Wooldridge, M., & R.Jennings, N. (2009). An Exact Algorithm for Coalition Structure Generation andComplete Set Partitioning.

Wagstaff, K., Cardie, C., Rogers, S., & Schroedl, S. (2001). Constrained K-means Clustering with Background Knowledge. *Proceedings of the Eighteenth International Conference on Machine Learning,* , 577-584.