Если для некоторой обобщенной формационной σ-функции f имеет место $\mathfrak{F} = BLF_\sigma(f)$, то класс $\mathfrak{F}$ называется *бэровским σ-локальным*, а f *– обобщенным σ-локальным заданием* класса $\mathfrak{F}$ (см. [5]).

Обобщенная формационная σ-функция f называется *внутренней*, если $f(a) \subseteq BLF_\sigma(f)$ для всех $a \in \sigma \cup \{\emptyset\}$.

Доказана следующая

Теорема 1. *Если $\mathfrak{F} = \cap_{j\in J}\mathfrak{F}_j$ и $\mathfrak{F}_j = BLF_\sigma(f_j)$ для всех $j \in J$, то $\mathfrak{F} = BLF_\sigma(f)$, где $f(\emptyset) = \cap_{j\in J} f_j(\emptyset)$ и $f(\sigma_i) = \cap_{j\in J} f_j(\sigma_i)$ для всех $\sigma_i \in \sigma^+(\mathfrak{F}) = \cap_{j\in J}\sigma^+(\mathfrak{F}_j)$ и $f(\sigma_i) = \varnothing$ для всех $\sigma_i \in \sigma \setminus \sigma^+(\mathfrak{F})$. Кроме того, если f_j – внутреннее обобщенное σ-локальное задание для всех $j \in J$, то f также является внутренним обобщенным σ-локальным заданием.*

Пусть $\{f_j \mid j \in J\}$ – набор всех обобщенных σ-локальных заданий формации $\mathfrak{F}$. В силу теоремы 1 $f = \cap_{j\in J} f_j$ – обобщенное σ-локальное задание формации F, называемое *наименьшим*, т.е. $f(a) = \cap_{j\in J} f_j(a)$ *для всех* $a \in \sigma_i \cup \{\emptyset\}$ по всем i.

Символом form($\mathfrak{X}$) обозначается пересечение всех формаций, содержащих совокупность групп $\mathfrak{X}$. Символом $c_\sigma$form($\mathfrak{X}$) обозначается пересечение всех бэровских σ-локальных формаций, содержащих совокупность групп $\mathfrak{X}$. Следующее утверждение дает способ построения наименьшего обобщенного σ-локального задания формации $\mathfrak{F} = c_\sigma$form($\mathfrak{X}$).

Теорема 2. *Пусть $\mathfrak{X}$ – некоторая непустая совокупность групп, $\mathfrak{F} = c_\sigma$form($\mathfrak{X}$) = $BLF_\sigma(f)$, где f – наименьшее обобщенное σ-локальное задание формации $\mathfrak{F}$. Тогда справедливы следующие утверждения*:

1) $\sigma^+(\mathfrak{X}) = \sigma^+(\mathfrak{F})$.

2) $f(\emptyset) = \text{form}(G/R_\sigma(G) \mid G \in \mathfrak{X})$.

3) $f(\sigma_i) = \text{form}(G/F_{\{g\sigma_i\}}(G) \mid G \in \mathfrak{X}) = \text{form}(G/F_{\{g\sigma_i\}}(G) \mid G \in \mathfrak{F})$ *для всех* $\sigma_i \in \sigma^+(\mathfrak{X})$ *и* $f(\sigma_i) = \varnothing$ *для всех* $\sigma_i \in \sigma \setminus \sigma^+(\mathfrak{X})$.

4) *Если h – произвольное обобщенное σ-локальное задание формации $\mathfrak{F}$, то для всех* $\sigma_i \in \sigma^+(\mathfrak{X})$ *имеет место*

$f(\sigma_i) = \text{form}(G \mid G \in \mathfrak{F} \cap h(\sigma_i), O_{\sigma_i}(G) = 1)$ *и*

$f(\emptyset) = \text{form}(G \mid G \in \mathfrak{F} \cap h(\emptyset), R_\sigma(G) = 1)$.

1. Скиба, А.Н. Алгебра формаций / А.Н. Скиба. – Минск: Беларуская навука, 1997. – 240 с.

2. Скиба, А.Н. Кратно $\mathfrak{L}$-композиционные формации конечных групп / А.Н. Скиба, Л.А. Шеметков // Украинский матем. журн. – 2000. – Т. 52, № 6. – С. 783–797.

3. Чи, Чжан О $\sum_t^\sigma$-замкнутых классах конечных групп / Чжан Чи, А.Н. Скиба // Украинский матем. журн. – 2018. – Т. 70, № 2. – С. 1707–1715.

4. Chi, Zhang On n-multiply σ-local formations of finite groups / Zhang Chi, V.G. Safonov, A.N. Skiba // Comm. Algebra – 2019. – Vol. 47, № 3. – P. 957–968.

5. Safonov, V.G. On Baer-σ-local formations of finite groups / V.G. Safonov, I.N. Safonova, A.N. Skiba // Comm. Algebra – 2020. – Vol. 48, № 9. – P. 4002–4012.

ПРИМЕНЕНИЕ ОБОБЩЕНИЙ В РАЗРАБОТКЕ БИБЛИОТЕКИ ДЛЯ ИЕРАРХИЧЕСКОЙ АГЛОМЕРАТИВНОЙ КЛАСТЕРИЗАЦИИ

С.А. Ермоченко
Витебск, ВГУ имени П.М. Машерова

В рамках данного исследования разрабатывались библиотеки классов для языков программирования C++ и Java, которые позволяют выполнить иерархическую агломеративную кластеризацию произвольных объектов с использованием различных метрик. Эта работа выполнялась в рамках научно-исследовательской работы «Методы искусственного интеллекта для оптимизации образовательного процесса» ГПНИ «Цифровые и космические технологии, безопасность человека, общества и государства» на 2021–2025 годы, подпрограммы «Цифровые технологии и космическая информатика», задания «Информационные технологии повышения качества образовательного процесса».

В настоящее время анализ данных – это очень актуальное направление в области информационных технологий, востребованное в самых разных сферах. Наиболее популярными

языками программирования, которые применяются в анализе данных, являются языки Python и R. Но часто возникает ситуация, когда в существующем приложении, написанном на C++ или на Java, необходимо со временем добавить анализ накопившихся данных. В таком случае разработчики либо решают проблемы интеграции отдельных модулей, написанных на разных языках программирования, при этом менеджеры проектов подбирают команду разработчиков таким образом, чтобы было достаточное количество специалистов по каждому применяемому языку программирования. Либо анализ данных реализуют средствами того языка программирования, на котором написан уже существующий проект. Такой подход зачастую оказывается дешевле. В последнем случае разработчики используют существующие библиотеки для применяемого языка программирования. В данной работе рассматриваются языки программирования C++ и Java как одни из самых популярных и востребованных на рынке для разработки как Desktop-приложений, так и Web-приложений, и Mobile-приложений.

Целью данной работы является разработка архитектуры библиотек для языков программирования C++ и Java для выполнения иерархической агломеративной кластеризации на основе применения механизма шаблонов в C++ и механизма обобщений в Java.

Материал и методы. Материалом в данной работе послужили:

– описание алгоритмов иерархической агломеративной кластеризации с применением различных метрик [1-2];

– описание концепции шаблонов и обобщений для языков программирования C++ [3] и Java [4].

На основании указанных материалов составлялись математические модели, проектировалась архитектура и реализовывались необходимые классы.

Методы исследования: метод математического моделирования, метод объектно-ориентированного анализа и проектирования, нисходящее проектирование программного обеспечения, а также общенаучный метод анализа и обобщения.

Результаты и их обсуждение. Для агломеративного кластерного анализа сейчас для языков программирования C++ и Java существует большое количество библиотек. Но основной особенностью этих библиотек является способы абстрагирования от предметной области (от природы рассматриваемых объектов). Применяется два подхода:

1. Обработка данных в числовом виде без привязки к природе этих данных. В библиотеках, построенных по такому признаку, на вход алгоритму кластеризации подаётся матрица расстояний между объектами. Такой подход имеет несколько существенных минусов.

Во-первых, при обработке большого количества объектов приходится задействовать значительные объёмы памяти для хранения матрицы расстояний, что затрудняет распараллеливание обработки данных.

Во-вторых, не все метрики вычисления расстояния между кластерами можно использовать при таком подходе.

2. Все объекты приводятся к некоторому единому типу. Например, в Java рассматриваются ссылки типа Object, а в C++ – указатели типа void*. При таком подходе теряется информация о типе и программисту приходится постоянно выполнять преобразования типов.

Особенностью подхода при разработке библиотек в рамках данной работы является применение шаблонных классов и функций (для языка программирования C++) и обобщений (для языка программирования Java). И хоть способ реализации данных подходов в этих двух языках отличается, общая идея с точки зрения создания высокоуровневого кода у них одинакова. В дальнейшем будем использовать термин обобщение как более широкий и будем считать шаблонные классы и функции в C++ одной из реализаций механизма обобщений.

При разработке библиотеки иерархической кластеризации необходимо начать с понятия кластера. Кластер – это некоторый набор родственных (близких между собой) объектов. Имеет смысл рассматривать кластер как набор объектов, который сравнивается с другим кластером. Для этого вводится понятие расстояния между объектами, помещаемыми в кластер. Данное расстояние может вычисляться самыми разными способами. Расстояние между двумя объекта, помещаемыми в разные кластеры, должно быть больше любых расстояний между каждым из этих объектов и любым другим объектом того же кластера, что и рассматриваемый объект. На начальном этапе кластеризации каждый объект помещается в отдельный кластер. Далее алгоритм кластеризации находит наиболее близкие кластеры и объединяет их в один кластер.

На последнем этапе получается один кластер, объединяющий всю совокупность рассматриваемых объектов.

При этом для программного моделирования кластера необходимо различать кластер, состоящий из одного только исходного объекта (листовой кластер) и кластер, состоящий из других кластеров. Приведём реализации обобщённого кластера на языках C++ и Java:

Язык C++

```
template <class T> class cluster {
    T* object;
    vector< cluster< T > > clusters;
    double distance;
public:
    cluster(T* object) {
        this->object = object;
    }
    cluster(vector< cluster< T > >
                                clusters,
            double distance) {
        this->object = NULL;
        this->clusters = clusters;
        this->distance = distance;
    }
    T* getObject() {
        return object;
    }
    vector< cluster< T > >
                         subclusters() {
        return clusters;
    }
    double getDistance() {
        return distance;
    }
    bool isSimple() {
        return object;
    }
};
```

Язык Java

```
public class Cluster<T> {
    private T object;
    private List<Cluster<T>>
                                subclusters;
    private double distance;
    public Cluster(T object) {
        this.object = object;
    }
    public Cluster(List<Cluster<T>>
                                subclusters,
                   double distance) {
        this.subclusters = subclusters;
        this.distance = distance;
    }
    public T getObject() {
        return object;
    }
    public List<Cluster<T>>
                        getSubclusters() {
        return subclusters;
    }
    public double getDistance() {
        return distance;
    }
    public boolean isSimple() {
        return object != null;
    }
}
```

Как видно, данные классы могут быть параметризованы любым типом, что позволяет легко манипулировать такими объектами. Благодаря применению принципа инкапсуляции, классы могут порождать только immutable-экземпляры, являющиеся либо обычным кластером, либо листовым. Далее при разработке библиотеки аналогичным образом создавались обобщённые интерфейсы для расчёта расстояния между объектами и между не листовыми кластерами. Данные интерфейсы позволяют гибко менять применяемые метрики расстояний при кластеризации для решения той или иной практической задачи.

Заключение. В представленной работе разработана архитектура библиотеки иерархической агломеративной кластеризации, базирующаяся на принципах обобщённого программирования. В библиотеке реализован сам алгоритм кластеризации и несколько способов вычисления расстояний между объектами. Для проверки корректности работы библиотеки выполнена кластеризация студентов одной из академических групп по оценкам, полученным ими на экзаменах за весь период обучения.

1. Мандель, И.Д. Кластерный анализ. – Москва: Финансы и статистика, 1988. – 176 с.
2. Жамбю, М. Иерархический кластер-анализ и соответствия. – Москва: Финансы и статистика, 1988. – 345 с.
3. Stroustrup, B. The C++ Programming Language, 4th Edition. – Addison-Wesley Professional, 2013. – 1376 p.
4. Gosling, J. The Java® Language Specification. Java SE 17 Edition [Electronic Resource] / J. Gosling, B. Joy, G. Steele etc. – Oracle Inc., 2021. – Mode access: https://docs.oracle.com/javase/specs/jls/se17/html/index.html. – Date access: 23.01.2022