# Лекция 6. Основни методи за нейерархична и йерархична клъстъризация

## *6.1. Клъстъризация чрез разделяне (partitioning)*

При зададена база данни от $n$ обекта и желания брой $k$ на клъстери, които трябва да бъдат създадени от тези данни, един разделящ клъстерен алгоритъм разчленява данни на $k$ непресичащи се раздела $(k \leq n)$, всеки от които представя един клъстер. Клъстерите се формират така, че да оптимизират определен критерий за разделяне, наричан често функцията за сходство, така че обектите в един клъстер са "сходни", докато обектите от различни клъстери са "различни" в термините на атрибутите от база данни.

Тази задача може да се разглежда като една от форми за комбинаторна оптимизация, тъй като ние търсим разпределянето на $n$ обекта по $k$ клъстера, което минимизира (или максимизира) избраният критерий. Броят на възможни разпределения е приблизително $k^n$. Например, има $2^{100} \approx 10^{10}$ възможни разпределения на 100 обекта в два клъстера. Очевидно е, че използването на изчерпващите методи за търсене на глобален екстремум в тази задача е неизгодно. По тази причина, повечето разделящи методи използват евристичната техника за итеративно подобряване на решението, водеща до намиране на локалния екстремум.

### 6.1.1. Използване на центроиди: метод k-Means

*Алгоритъм k-Means* (k–Средни) с входния параметър $k$, разделя множеството от $n$ обекта на $k$ клъстера по такъв начин, че вътре-клъстерното сходство става голямо, а между-клъстерното сходство – малко. Клъстерното сходство се измерва по отношение на средната стойност на обекти в клъстера – изкуствено създаден обект – *центроид*, представляващ център на клъстера, който може да се разглежда като център на тежестта на клъстера. Работата на алгоритъма се състои в следното (виж фиг. 6.1).

Алгоритъмът започва работата си със случаен избор на $k$ обекта като центрове на търсените клъстери. След това всеки от останалите обекти се разпределя към най-близкия клъстер, като разстоянието до клъстера се изчислява като разстояние от обекта до центъра на клъстера. След като всички обекти са разпределени, изчисляват се нови центрове на клъстери, които се представят като центроиди – обекти, в които стойността на всеки атрибут е средно аритметично от стойностите на съответния атрибут по всички обекти от дадения клъстер. След това описаният процес на разпределяне на обекти по клъстери се повтаря отново. Цикълът: формирането на клъстери – уточняване на клъстерните центрове се повтаря до

тогава, докато не се получи сходимостта на избрания критерий за качеството на клъстеризация. Най-често се използва критерият за минимизация на средно-квадратичната грешка.

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} d^2(\mathbf{p}, \mathbf{m_i}) \quad (6.1)$$

където $E$ е сума от квадрати на грешките на всички обекти от база данни, **p** е точка в пространство, представляваща дадения обект, а $\mathbf{m_i}$ е център на клъстер $C_i$ (**p** и $\mathbf{m_i}$ са многомерни точки - вектори); $d$ – е Евклидовото разстояние. Описаният критерий опитва да направи клъстери по-компактни и по-раздалечени един от друг.

---

**Алгоритъм k-Means.**

***Вход:*** *k – желания брой на клъстери;*
*n – брой на обекти в база данни*

***Изход:*** *Множество от k клъстери, минимизиращо критерий за средно-квадратичната грешка.*

***Описание:***

1. Избери произволно *k* обекта като начални центрове на клъстери
2. **Повтори**
3. Формирай клъстери:
   **от** *i = 1* **до** *n* направи:
   Изчисли разстояние от обекта до центъра на всеки от клъстери и разпредели обектът към клъстер с най-близкия център;
4. Изчисли новите центрове на клъстери като вектор от средните стойности на всички обекти в клъстера
5. **Докато** няма да има промени в избрания критерий

---

*Фиг. 6.1. Описание на алгоритъма k-Means*

Изчислителната сложност на алгоритъма е $O(knI)$, където $I$ е броят на итерации. Обикновено $k << n$ и $I << n$.

Съществуват различни вариации на алгоритъма, свързани с избора на начални центрове на клъстери и момента на тяхното преизчисляване. Например, при един от вариантите центровете се преизчисляват след всяка точка, която премина от един клъстер към друг, докато не спрат преместванията.

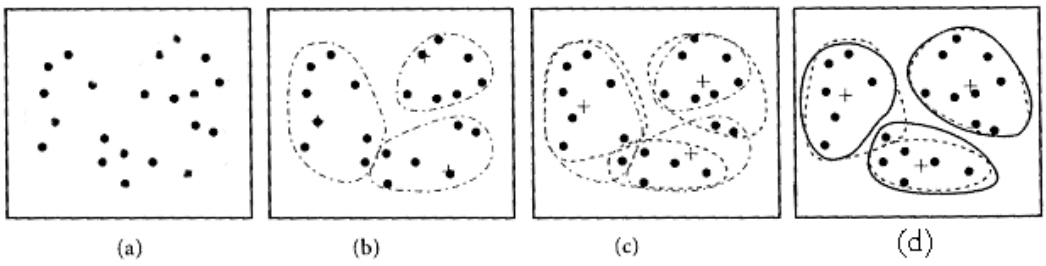
Търсенето в k-Means е ограничено до една малка част от цялото пространство на възможните разделяния. По тази причина е възможно да бъде изпуснато едно добро решение, тъй като алгоритмът намери някой локален минимум на използваният критерий. За да подобри качеството на решения често се прилагат метод на многократно повтаряне на алгоритъма с различни начални точки, случайно избрани в качеството на центрове на клъстери. Окончателното решение е това, което има най-малката стойност на избрания критерий.

Обаче k-Means може да бъде приложен само към обекти, описвани с *непрекъснати атрибути*, тъй като само за тях е определено понятие средната стойност. Алгоритмът работи добре с клъстери със сферичната форма, относително еднакви по размер и добре разделени в пространството. Той не е подходящ за намиране на клъстери с произволна форма и много различаващи се по размер. Освен това, той е чувствителен към шума и крайностите, тъй като дори малкото количество от такива обекти съществено променят координати на центроида.

*Пример 6.1.* Да предположим, че едно множество от двумерни обекти е разположено в пространство както е показано на фиг. 6.2(a). Нека k = 3, т.е. искаме да разделим обектите на 3 клъстера. Съгласно алгоритъма k-Means избираме произволно три обекта като начални центрове на клъстери (те са отбелязани със знак "+"). Всеки от останалите обекти се разпределя към един от трите клъстера в зависимост от разстояние до най-близкия център на клъстера. Резултат от това разпределение е показан на фиг 6.2(b), като всеки клъстер е отделен с пунктираната линия.

Полученото групиране определя нов център на всеки клъстер. Новата средна стойност на всеки клъстер се изчислява на базата на обекти, принадлежащи само към съответния клъстер. Всички обекти се преразпределят по клъстери съгласно изчислени нови центрове. Получените нови клъстери и техните центрове са показана на фиг. 6.2(c).

Описаният процес се повтаря и води до ситуацията, показана на фиг. 6.2(d). Тъй като вече няма никакво преместване на обекти от един клъстер в друг, то процесът спира, връщайки като резултат клъстерите, показани на последната рисунка с непрекъсната линия.




*Фиг.6.2. Клъстеризация на обекти на 3 клъстера чрез алгоритъм k-Means. (Център на всеки клъстер е обозначен с "+").*

## 6.1.2. Разполовяващ k-Means

Разполовяващият (bisecting) k-Means алгоритъм е едно разширение на традиционния k-Means, реализиращ следната проста идея: за да получим *k* клъстери, да разделим отначало цялото множество от точки на два клъстера, след това от тях да изберем един клъстер, който пак да разделим на два клъстера и т.н. докато не получим нужното количество клъстери. По-детайлно, разполовяващият k-Means алгоритъм е представен на Фиг. 6.2.

**Алгоритъм Разполовяващ k-Means.**
***Вход:*** *k – желания брой на клъстери;*
*n – брой на обекти в база данни*
l – брой на прилагания на k-Means на всяка стъпка по разделяне
***Изход:*** *Множество от k клъстери, минимизиращо критерий за средно-квадратичната грешка.*
***Описание:***

1. Инициализирай списък на клъстери само с един клъстер, съдържащ всички точки
2. **Повтори**
3. Избери един клъстер и го изтрий от списъка на клъстери
4. {Направи няколко опита по разделяне на избрания клъстер на 2 клъстера}
5. **for** *i = 1* **to** *l* **направи**:
6. Раздели избрания клъстер на 2 чрез прилагане на k-Means
7. **end for**
8. От *l* възможни двойки клъстери избери двойката с най-малката стойност на обща средно-квадратична грешка
8. Добави тези два клъстера в списък на клъстери
6. **Докато** списък от клъстери не съдържа k клъстери

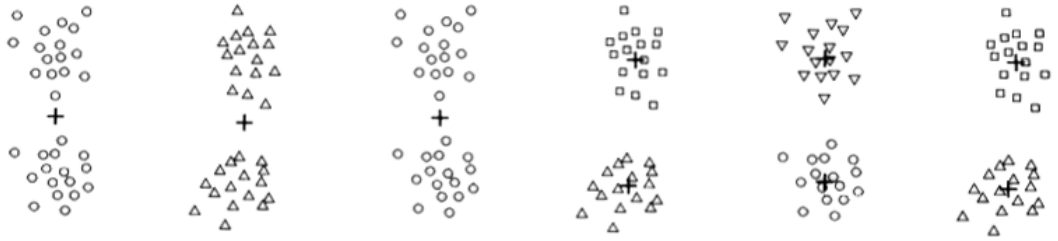
*Фиг. 6.2. Описание на алгоритъма „Разполовяващ”k-Means*

Съществуват различни способи за избор на клъстера, подлежащ на разделяне. Можем да изберем най-големия от клъстери на всяка стъпка, или да изберем този с най-голяма средно-квадратична грешка, или да използваме критерий, комбинираш предишните два.

Много често, клъстерите, получени в резултат от прилагането на разполовяващия k-Means допълнително се уточняват, използвайки центроидите на получените „чрез разполовяване” клъстери като начални центроиди за традиционния k-Means. Тази стъпка е необходима, тъй като макар k-Means гарантира намиране на клъстеризацията, който има локален минимум по отношение на средно-квадратичната грешка, при разполовавящия вариант на алгоритъма самият k-Means се използва „локално”, т.е. за разделяне на две части на отделните клъстери. По тази причина крайното множество от клъстери няма да представлява множество от клъстери с локален минимум на общата средно-квадратична грешка.

*Пример 6.2.* За да покажем, че разполовяващият k-Means е по-малко зависи от началната инициализация на центроиди, да разгледаме как той намира четири клъстера в данни (Фиг. 6.3). На първата итерация се намира една двойка клъстери; на втората – разположените вдясно данни се разбиват на два отделни клъстери, и на третата – данните разположени отляво се разбиват на два клъстера. Разполовяващият k-Means има по-малко проблеми с инициализацията, тъй като на всяка стъпка той прави няколко опита по двоичното разделяне на клъстера и

избира двойката с най-малката средно-квадратична грешка. Освен това на всяка стъпка има само по два центроида.



*a) Итерация 1* *b) Итерация 2* *c) Итерация 3*

*Фиг. 6.3. Илюстрация на работата на двоичния k-Means в примера с четири клъстера*

## 6.1.3. Алгоритми k-Modes и k-Prototypes

Вариантът на k-Means, позволяващ работата с номинални атрибути, се нарича *k-Modes.* Основната идея е да бъде заменено Евклидовото разстояние с друга мярка за различие, дефинирана върху номинални атрибути, и да бъде дефинирано понятие на средната стойност за такива атрибути. В качеството на мярката за различието на два обекта ***x*** и ***y***, описани само с номинални атрибути, е използвана вече познатата ни мярка за просто съвпадение:

$$d_1(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{p} \delta(x_i, y_i), \text{където } \delta(x_i, y_i) = \begin{cases} 0, \text{ако } x_i = y_i \\ 1, \text{ако } x_i \neq y_i \end{cases} \quad (6.2)$$

Използваният критерий за качеството (6.1) се трансформира в:

$$E_1 = \sum_{i=1}^{k} \sum_{p \in C_i} d_1(\mathbf{p}, \mathbf{m_i}) \quad (6.3)$$

Доказана е теоремата, че критерият има минимум, ако в качеството на центрове на клъстери $\mathbf{m_i}$ се избират *моди* на обекти, формиращи съответните клъстери. Модата на един клъстер е обект (вектор в *l*-мерното пространство), който има атрибутните стойности, най-често срещани между обектите от същия клъстер.

Алгоритъм, интегриращ k-Means и k-Modes, е предложен от Z. Huang (1998) и носи название *k-prototypes* (k–прототипи). Той позволява клъстеризацията на обекти, описвани със смесени (непрекъснати и номинални) атрибути, като в качеството на функция за различието се използва смесената функция:

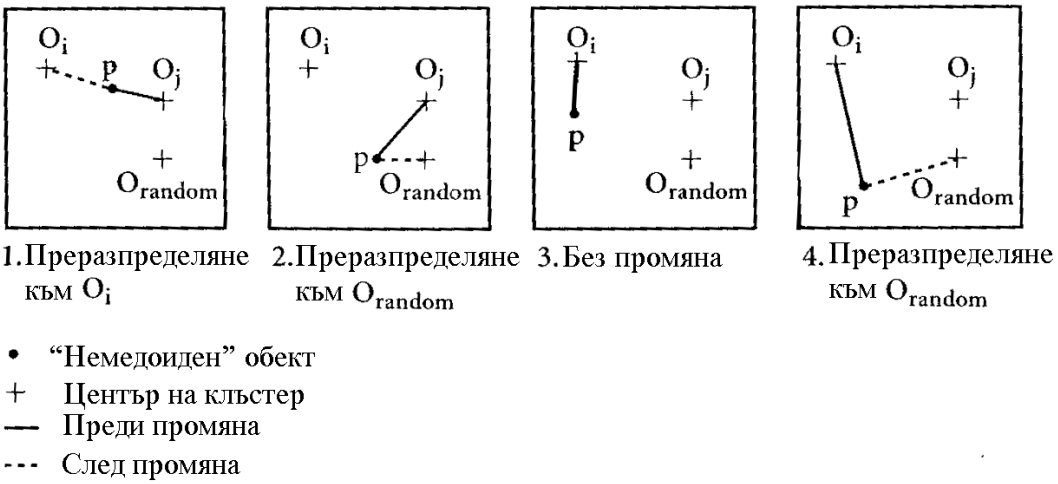
$$d(\mathbf{x},\mathbf{y}) = \sum_{i \in A_c} (x_i - y_i)^2 + \gamma \sum_{i \in A_n} \delta(x_i, y_i) \quad (6.4),$$

където $A_c$ и $A_n$ са, съответно, множества от непрекъснати и номинални атрибути, а $\gamma$ е задавана от потребителя константа, балансираща влияние на номинални атрибути.

### 6.1.4. Използване на представителни обекти: метод k-Medoids

Основният недостатък на алгоритъма k-Means е неговата чувствителност към екстремни обекти, тъй като един обект с много големи стойности може съществено да изкриви разпределение на данни. Този недостатък може да бъде преодолян, ако вместо центроида, като представителя на клъстер, ще се използва най-централно разположения в клъстера обект, наречен *медоид*. Клъстеризацията пак ще се извършва чрез разделянето на обекти по клъстери на базата на минимизацията на сумата от различията между всеки обект и съответният медоид. Както и при k-Means, работата на алгоритъма започва със случаен избор на *k* обекта, представящи клъстери. След това останалите обекти се разпределят по клъстери в зависимост от най-близкия представителен обект – в дадения случай медоид. След като всички обекти са преразпределени, за всеки клъстер се намира нов медоид и се извършва новата итерация по преразпределяне на обектите. Този цикъл продължава, докато не се спре подобряване на качеството на клъстеризация.

За реализацията на подобна схема трябва да дефинираме метод за намиране на нов медоид, т.е. да опишем случаи, кога един медоид трябва да бъде сменен с друг "немедоиден" обект. Очевидният критерий за този избор е оценъчната функция, показваща как се променя качеството на дадения клъстер (т.е. средното различие между обектите в клъстера и техния медоид) при подобна замяна. За да определим, дали някой немедоиден обект $O_{random}$ е една добра замяна на текущ медоид $O_j$, за всеки немедоиден обект $P$ се проверяват следните четири случая (виж фиг. 6.4) :




*Фиг. 6.4. Четири случая за използване на оценъчната функция за клъстеризация чрез k-Medoids*

- *Случай 1*. Нека $P$ принадлежи към клъстера с медоид $O_j$. Ако след замяна на $O_j$ с $O_{random}$ като медоид, $P$ става най-близо до един от другите медоиди $O_i$ $i \neq j$, то $P$ се преразпределя към този най-близък медоид $O_i$.

- *Случай 2*. Нека $P$ принадлежи към медоида $O_j$. Ако след замяна на $O_j$ с $O_{random}$ като медоид, $P$ става най-близо до $O_{random}$, то $P$ се преразпределя към $O_{random}$.

- *Случай 3*. Нека $P$ принадлежи към медоида $O_{i,}$: $i \neq j$. Ако след замяна на $O_j$ с $O_{random}$ като медоид, $P$ остава най-близо до $O_i$, то преразпределянето на $P$ не се извършва.

- *Случай 4*. Нека $P$ принадлежи към медоида $O_{i,}$ $i \neq j$. Ако след замяна на $O_j$ с $O_{random}$ като медоид, $P$ става най-близо до $O_{random}$, то $P$ се преразпределя към $O_{random}$.

Всеки път, когато се извършва преразпределянето, се изчислява разликата в квадрати на грешката, причинена от текущото преместване. Общата цена на замяна на текущ медоид с друг немедоиден обект се изчислява като разликата в квадрати на грешката за всички немедоидни обекти. Ако получената оценка е отрицателна (т.е. величината на критерия за качеството на клъстеризация стана по-малка), то текущия медоид $O_j$ се заменя с нов - $O_{random}$. Ако тази разлика е положителна, то текущ медоид се смята за приемлив (добър) и на тази итерация не се променя.

Един типичен k-Medoids алгоритъм е представен на фиг. 6.5.

**Алгоритъм k-Medoids**

***Вход:*** *k – желания брой на клъстери;*
*n – брой на обекти в база данни*

***Изход:*** *Множество от k клъстери, минимизиращо критерий за средно-квадратичната грешка от всеки обект до най-близкия медоид*.

***Описание:***

1. **Избери** произволно k обекта като начални медоиди на клъстери
2. **Повтори**
3. Формирай клъстери:
   **За всеки** *немедоиден обект* направи:
   Изчисли разстояние от обекта до медоида на всеки от клъстери и разпредели обект към клъстер с най-близкия медоид;
4. Изчисли новите медоиди:
   **За всеки** *немедоиден обект* направи:
   Избери по случаен начин някой немедоиден обект
   Изчисли общата стойност $S$ на замяна $O_j$ с $O_{randon}$;
   Ако $S < 0$ то замени $O_j$ с $O_{randon}$;
   Формирай множество от новите медоиди
5. **Докато** няма да има промени в избрания критерий

*Фиг. 6.5. Описание на алгоритъма k-Medoids*

Един от първите алгоритми, реализиращ метода на k-Medoids е *PAM (Partitioning Around Medoids)*. След случайното избиране на *k* медоида, алгоритмът итеративно опитва да подобри избора на медоиди. Той анализира всички възможни двойки обекти, от които един обект се разглежда като медоид, а друг – не. Качеството на получената клъстеризация се изчислява за всяка такава комбинация. Обектът $O_j$ се заменяше с този обект, който предизвика най-голямото намаляване в квадратичната грешка. Множеството от най-добрите обекти за всеки клъстер на една итерация формира медоидите за следващата итерация. Както се вижда, за големи стойности на *n* и *k* подобният алгоритъм изисква много големи изчисления.

Методът на k-Medoids е по устойчив на наличието на шума и екстремните обекти, тъй като един медоид по-малко се влияе от екстремните стойности отколко един центроид. Освен това, тъй като методът може да ползва произволна мярка за различие, той може да бъде прилаган и за обекти, описвани и с *номинални атрибути*. Обаче, от изчислителната гледна точка методът е доста по-скъп от k-Means. И двата методи изискват от потребителя задаването на параметъра *k*.

### 6.1.5. Клъстеризация чрез разделяне в големи бази данни

**От класическия k-Means към еднопасови k-Means алгоритми**

Големи бази данни, които са основния предмет на ИЗД, не могат да се поберат изцяло в основната памет. Неколкократното сканирането на такава база, съхранявана на някое външно запаметяващо устройство, изисква много време, обаче, за да получи едно добро решение класическият алгоритъм k-Means се нуждае от няколко итерации върху цялата база данни, като на всяка итерация е необходим достъпът до всеки неин елемент. По тази причина, съвременните изследванията се фокусират на създаване на клъстерните алгоритми, които правят *само едно сканиране* (пас) на цялата база данни. Тези методи подразбират, че само една част от базата данни може да бъде натоварена в основната памет (буфер) и се нуждаят само от едно минаване през всички данни.

В основата на подобни методи е намирането в данни на области с различни свойства: области от данни, които могат да бъдат компресирани, области от данни, които трябва да бъдат поддържани в основната памет и области от данни, които могат да бъдат пренебрегнати (изхвърлени). Един обект може да бъде *изхвърлен*, ако неговата принадлежност към окончателния клъстер е напълно установена. Обектът може да бъде компресиран, ако той не може да бъде изхвърлен, но принадлежи към някой тесен подклъстер. За съхраняване на информация за изхвърлени или компресирани данни се използва една структура на данни, наричана *клъстеризиращ признак (clustering feature)*. Клъстеризиращият признак съдържа така наречени *достатъчни статистики*, състоящи се от три величини – сума на стойностите, сума от квадрати на стойностите и броя на изхвърлените или компресирани обекти за всеки клъстер. Обектът, който нито може да бъде изхвърлен, нито може да бъде компресиран, трябва да бъде оставен в основната

памет. Еднопасовите алгоритми работят върху обекти, запазени в основната памет, заменяйки компресираните или изхвърлените обекти с техните клъстеризиращи признаци.

Един такъв *прост еднопасов k-Means алгоритъм* е предложен от Farnstrom et al (2000). Той работи по следния начин:

1. Инициализирай по случаен начин центрове на клъстери. Нека със всеки клъстер е свързано определено клъстеризиращо множество (признак), което пази в буфера достатъчните статистики за всички изхвърлени обекти на предишните итерации.
2. Запълни буфер с нови обекти.
3. Изпълни необходимите итерации на k-Means върху точките от буфера и данните от клъстеризиращите признаци, докато не бъде намерено решението. При тази клъстеризация всеки обект от клъстеризиращото множество се третира като един нормален обект, поместен в средата на това множество, но имащ тегло, равно на броя на обекти в множеството. (С други думи, множеството е представено като един "тежък" центроид и разстоянието от него до текущ център на клъстера се изчислява като обикновено Евклидово разстояние, претеглено с това тегло).
4. За всеки клъстер обнови достатъчните статистики в клъстеризиращото множество с данни за нови обекти, назначени за дадения клъстер.
5. Ако цялото множество от данни е прегледано - край. Иначе, изхвърли всички обекти (освен центроиди) от буфера и повтори цикъла, започвайки със стъпка 2.

Авторите съобщават, че предложения метод е няколко пъти по-бърз от класическия k-Means и постига едно сравнимо с него качество при експерименти, проведени върху синтетични и реални бази от данни с размер до 400 Мегабайта.

**От k-Medoids към CLARANS**

Алгоритми, базирани на метода k-Medoids, не могат директно да бъдат приложени към големи бази от данни. Един от начините да бъде избегнат този недостатък е чрез използване на метода наречен CLARA (Clustering LARge Applications), който се базира на използването на извадки.

Основната идея на CLARA е следната: вместо разглеждането на цялото множество от данни, се избира само една малка част от тях. От тази извадка се избират медоидите чрез прилагане на алгоритъма PAM. Останалите данни се преразпределят по клъстери чрез измерването на тяхното сходство със всеки от създадените медоиди. Ако извадката е избрана по един действително случаен начин, тя трябва достатъчно точно да представя оригиналното множество от данни. При това положение е доста вероятно, че избраните представителни обекти (медоиди) ще приличат (бъдат близки до) на тези, които би били избрани при

използване на цялата база данни. CLARA избира от база данни няколко извадки, прилага към всяка от тях алгоритъм PAM и връща като изход най-добрата клъстеризация. Ясно е, че CLARA може да работи с по-големи бази данни от PAM. Изчислителната сложност на CLARA за всяка итерация е $O(ks^2 + k(n - k))$, където $s$ е размерът на извадката, $k$ е броят на клъстери, а $n$ – общият брой на обекти в базата данни.

Ефективността на CLARA зависи от размера на извадката. Докато PAM търси най-добрите k медоида измежду всички данни, CLARA ги търси само измежду данните от избраната извадка. По тази причина CLARA не може да намери най-добрите медоиди, ако те не са в извадката. От тук следва, че една добра клъстеризация, базирана върху извадката, не винаги е добра за цялата база данни.

*Алгоритъм CLARANS* (Clustering Large Application based upon RANdomized Search - Ng & Han 1994) е един опит да подобри CLARA. В отличието от CLARA, който на всеки етап от търсенето на медоиди използва някоя фиксирана извадка, CLARANS при търсенето избира всеки път някоя нова извадка, направена по случаен начин. Този процес на клъстеризацията може да бъде представен като търсене в един граф, чиито възли са потенциални решения, т.е. различни множества от k медоида. Клъстеризацията, получена след замяна само на един медоид с друг, се нарича *съсед* на текущата клъстеризация. Броят на съседи, които могат по случаен начин да бъдат тествани, се задава от параметъра, дефиниран от потребителя. Ако при това търсене в графа се намери по-добър съсед (т.е. имащ по-малка стойност на квадратичната грешка), то CLARANS се предвижва в този възел и процесът на търсене продължава с нова итерация (общият брой на итерации – е друг параметър на алгоритъма); в противен случай текущият възел се разглежда като едно локално оптимално решение. При това положение, ако допустимият брой на итерации не е надхвърлен, алгоритмът опитва да намери новия локален оптимум, започвайки работата си с нови случайно избрани възли.

Беше експериментално доказано, че CLARANS превъзхожда по ефективност и PAM и CLARA. Обаче, изчислителната сложност на CLARANS е около $O(n^2)$, а качеството на клъстеризация силно зависи от използвания метод за пораждане на извадките.

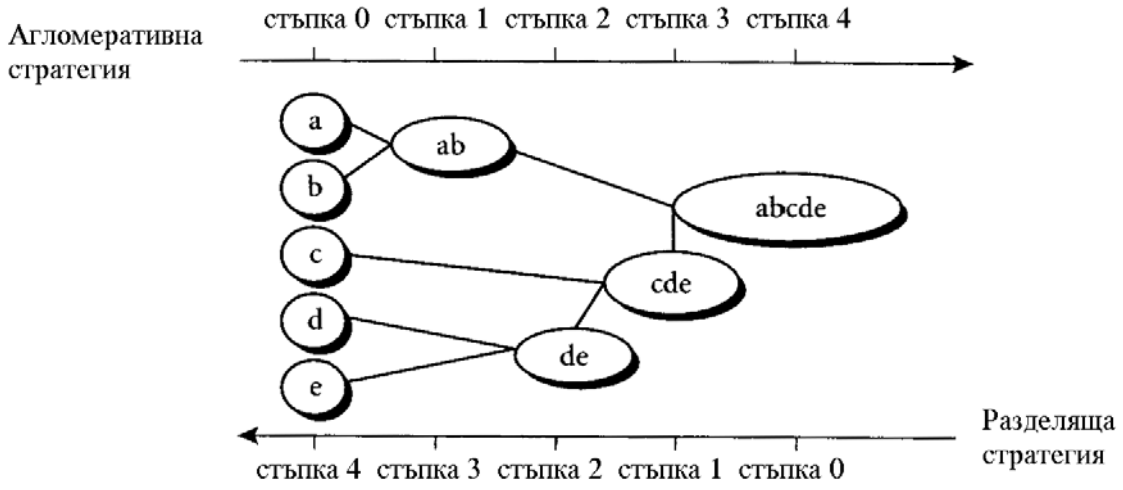
## 6.2. Методи за йерархична клъстеризация

Един метод за йерархична клъстеризация работи чрез групиране на обекти в дърво от клъстери. Такива методи могат да бъдат разбити на два класа – *агломеративни* (“слепващи”) и *разделящи* (divisive) в зависимост от това, дали йерархичната декомпозиция се прави отдолу – нагоре или отгоре – надолу. Качеството на един чисто йерархичен метод за клъстеризация се влошава от неговата неспособност да променя вече взетите решения за слепване или разбиване на междинни клъстери. Съвременните изследвания в тази област наблягат на интеграцията на

йерархичната агломерация с методи за итеративното преразпределяне на обекти по клъстери.

### 6.2.1. Агломеративна и разделяща йерархична клъстеризация

*Методите за агломеративна йерархична клъстеризация* започват работата си с поместването на всеки обект в един отделен клъстер, след което тези атомарни клъстери последователно се слепват във все по-големи и по-големи клъстери, докато или всички обекти не бъдат поместени в един единствен клъстер, или не бъде изпълнен някой предварително зададен критерий за спиране. Методите от тази група се различават по дефиницията на функцията на сходство между клъстери.




*Фиг. 6.6. Йерархична клъстеризация на множеството обекти {a,b,c,d,e}*

*Методите за разделяща йерархична клъстеризация* работят в посока, обратна на агломеративната клъстеризация – те започват работата с поместване на всички обекти в един единствен клъстер. След това те итеративно разбиват клъстерът на все по-малки и по-малки парчета, докато или не бъдат получени атомарни клъстери (т.е. съдържащи само по един обект), или не бъде изпълнен някой предварително зададен критерий за спиране, като, например, определен брой на клъстери или праг за най-голямото разстояние между два най-близки клъстера. Фиг. 6.6. илюстрира агломеративната и разделящата стратегии за йерархичната клъстеризация.

И двете стратегии за йерархичната клъстеризация се базират, в общия случай, на метода за определяне на разстояние между клъстери. Съществуват различни дефиниции на това разстояние, но всички те се базиран на начина за изчисляване на разстояния между двойки обекти от различни клъстери. Една от най-ранните и най-важните мерки за разстояние е *минималното разстояние*, което още се нарича *разстояние до най-близкия съсед* или *метод на единичното свързване (single link)*. То дефинира разстояние между двата клъстера като разстояние между двойката от най-близки обекти от тези клъстери:

$$d_{\min}(C_i, C_j) = \min_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y}) \quad (6.5)$$

където $C_i$ и $C_j$ са два различни клъстера, а $d(\mathbf{x}, \mathbf{y})$ е разстояние между обектите **x** и **y**. Използването на тази мярка за разстояние между клъстери води до така наречен *"верижен ефект"*, при който дългите верижки от близо намиращи се точки (обекти) се назначават в един и същ клъстер. Това означава, че описаният метод на единичната връзка има ограничена стойност за сегментиране. Освен това, той е чувствителен към малки промени в данни и към наличието на екстремните обекти. Методът на единичната връзка има свойство (уникално в сравнение със всички останали мерки за разстояние между клъстери), че ако две двойки клъстери се намират на едно и също разстояние помежду си, то няма никакво значение, в кой ред те ще се сливат (или разделят) – общия резултат ще бъде един и същ.

Другата, пак екстремна, мярка за разстояние между клъстери е *максималното разстояние*, което още се нарича *разстояние до най-далечния съсед* или *метод на пълното свързване (complete link)*. То дефинира разстояние между двата клъстера като разстояние между двойката от най-отдалечени един от друг обекта от тези клъстери:

$$d_{\max}(C_i, C_j) = \max_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y}) \quad (6.6)$$

Използването на тази мярка за разстояние води до създаване на клъстери с еднакъв размер в термините на обема на заеманото от тях пространство (не на броя на обектите в тях).

По средата между тези две екстремни мерки се намират *разстояние между центроиди* и *средното разстояние*:

$$d_{mean}(C_i, C_j) = d(\mathbf{m}_i, \mathbf{m}_j) \quad (6.7)$$

и

$$d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{\mathbf{x} \in C_i} \sum_{\mathbf{y} \in C_j} d(\mathbf{x}, \mathbf{y}) \quad (6.8)$$

където $\mathbf{m}_i$ и $\mathbf{m}_j$ са централни обекти (центроиди) на съответните клъстери, а $n_i$ и $n_j$ са броя на обекти в тези клъстери.

В качеството на алгоритъма за разделяща йерархична клъстеризация може да се използва разполовяващият k-Means. В този случай желаният брой на клъстери $k$ трябва да е равен на общия брой на точки, а за получаване на йерархичното подреждане на клъстери (дендрограмата) трябва на всяка итерация да записваме, коя двойка клъстери е наследник на какъв клъстер.

Качеството на клъстеризация може да бъде оценено с помощта на различни мерки. По-детайлно, проблеми, свързани с оценяване на качеството на клъстеризация (както йерархична, така и нейерархична) ще бъдат разгледани в една от следващите лекции.

Основните проблеми на методи за йерархичната клъстеризация са свързани с избора на точки за сливане или разделяне на клъстери, тъй като след извършване на избраната операция всички последващите стъпки ще се извършват върху получените на тази стъпка клъстери. Няма нито връщане назад за промяна на вече взети решения, нито дори размяна на обекти между вече създадените клъстери. По този начин едно ненай-доброто решението за сливане или разделяне на клъстери, взето на някоя от стъпките, може до доведе до низкото качество на клъстеризацията като цяло.

Едно обещаващо направление за подобряване на качеството на йерархичните методи е интегриране на йерархичната клъстеризация с други клъстеризационни техники за получаване на клъстеризацията на няколко етапа. Някои от такива методи ще бъдат разгледани в следващите раздели. Първият метод, наречен BIRCH, започва йерархичната клъстеризация на обекти чрез използване на специални дървовидни структури данни, а след това прилага други клъстеризационни техники за доуточняване на клъстери. Вторият, наречен CURE, представя всеки клъстер чрез определен брой представителни обекти, след това ги придърпва в посоката център на клъстера в определена пропорция.

### 6.2.2. BIRCH: Балансирано итеративно намаляване и клъстъризация с използване на йерархии

BIRCH (Balanced Iterative Reducing and Clustering Using Hierarchies – Zhand et al. 1996) е един интегриран метод за йерархична клъстеризация. Той въвежда две понятия – *клъстеризиращ признак и дърво от клъстеризиращи признаци (CF – дърво)*, които се използват за компресирано представяне на клъстери. Един *клъстеризиращ признак (CF*) е триплет, обобщаващ информацията за подклъстери от обекти. Ако $N$ е броят на $p$-мерни обекти $\{O_i\}$ в подклъстера, то CF на този подклъстер се дефинира като:

$$CF = (N, \overrightarrow{LS}, SS) \quad (6.11)$$

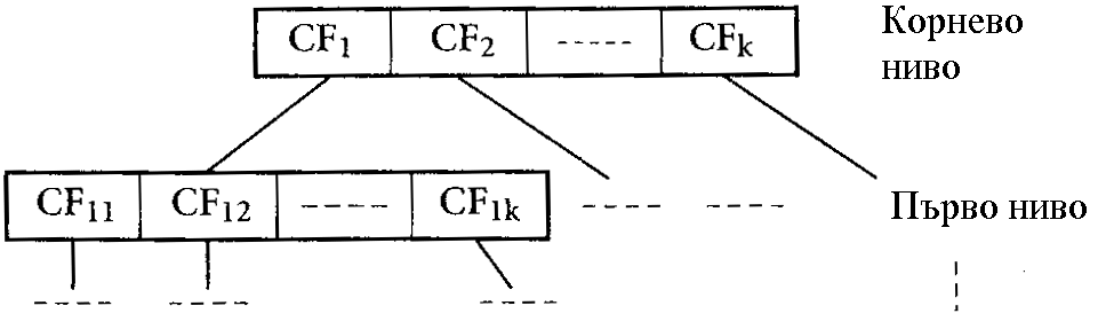
където $\overrightarrow{LS} = \sum_{i=1}^{N} \vec{O}_i, \; SS = \sum_{i=1}^{N} \vec{O}_i^2 \quad (6.12)$

На практика, CF съдържа статистики, достатъчни за абсолютно точно изчисляване на избраната мярка за разстояние между клъстери и на критерия за качеството на клъстеризацията. Той представя подклъстерът като един "тежък" обект в p-мерното пространство с тегло $N$ и координати на центроида на подклъстера, изчислявани чрез тези статистики.

По определение, всеки междинен възел има "деца". Междинните възли съхраняват сумите от CF на своите деца, обобщавайки по този начин и информация за техните деца. CF-дървото има два параметъра: *фактор на разклоняване B* и *праг T.* Факторът на разклоняване определя максимален брой на деца за един междинен

(нетерминален) възел, а праговият параметър определя максималния диаметър на подклъстери, съхранявани в терминалните възли (листата) на дървото. Тези два параметъра влияят на размер на CF–дървото.

CF-дървото е балансираното по височина дърво, което съхранява клъстеризиращи признаци, използвани за йерархична клъстеризация (виж примерът от фиг. 6.7).




*Фиг. 6.7. Пример на структурата на едно CF-дърво*

Работата на BIRCH се разделя на две фази:

- *Фаза 1*: BIRCH сканира цялата база данни, за да построи началното CF-дърво, което ще се пази в основната памет. Този етап може да се разглежда като многонивовата компресия на данни, опитваща да запази съществуващата в данни клъстерна структура.
- *Фаза 2*: BIRCH прилага избрания клъстерен алгоритъм за клъстеризация на терминалните възли на CF-дървото.

Построяването на CF-дърво на първата фаза става динамично чрез вмъкването на обект след обект, т.е. методът е инкрементален.

1. За всеки нов обект се намира най-близкото листо (подклъстер).
2. Ако диаметърът на този подклъстер след добавянето в него на новия обект не надминава зададения праг *T*, то обектът се добавя в подклъстера.
3. Иначе подклъстерът (и евентуално и други междинни възли над него) се разбива и обектът образува нов подклъстер.
4. След добавянето на всеки нов обект информацията за него се предава нагоре по дървото до самият корен.
5. Ако дървото става прекалено голямо, за да се побере в основната памет, то се компресира чрез сливането на най-близките листа.

Размерът на CF-дърво може да се промени чрез модифициране на праговия параметър, така че в случаи, когато дървото не се помества в основната памет, този параметър се намалява и дървото се модифицира. Този процес на престрояване на дървото се осъществява чрез построяване на нови дървета от листата на старото

дърво – без необходимостта от повторното прочитане на обекти. По този начин BIRCH прави само един пас на цялата база данни.

След построяване на CF-дърво, на фаза 2, BIRCH прилага към неговите листа някой от традиционни йерархични алгоритми за да получи окончателната клъстеризация.

И така, BIRCH опитва да направи най-добрите клъстери от наличните ресурси. При зададения размер на основната памет, важно е да бъде намалено време за входно-изходните операции. За тази цел алгоритъмът прилага многонивовото клъстеризиране – първото сканиране на данни създава едно добро клъстеризиране, което може да бъде (допълнително) подобрено чрез още едно или няколко сканирания. Изчислителната сложност на BIRCH е $O(n)$ – линейна по отношение на броя на обекти, подлежащи на клъстеризация.

Експериментите с BIRCH показали доброто качество на получаваната клъстеризация и ефективност при работа с големи бази данни. Обаче, тъй като размер на CF-дърво е ограничен, един възел в CF-дърво не винаги отговаря на това, което потребител разглежда като един естествен клъстер. Освен това, алгоритъмът е приложим само към обекти, описвани с непрекъснати атрибути, и не работи добре когато клъстери не са сферични по форма.

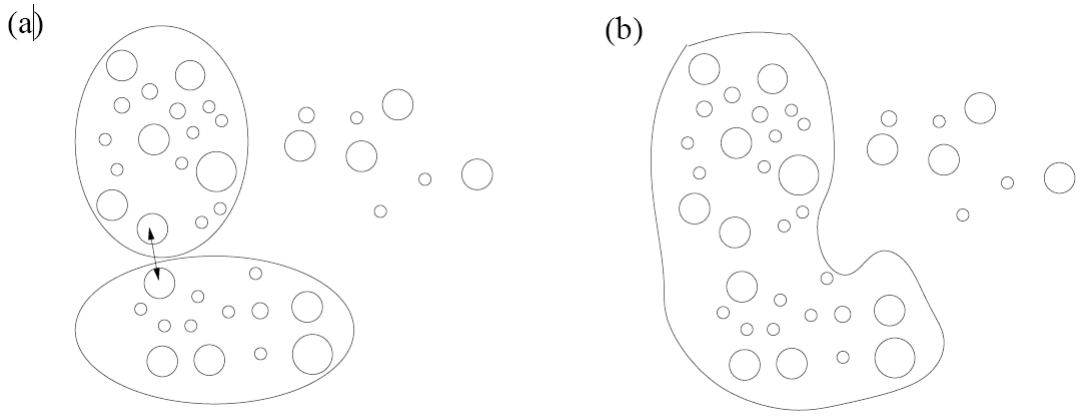
### 6.2.3. CURE: Кластеризация чрез използване на представители

Алгоритъм CURE (Clustering Using Representatives) е оригиналният алгоритъм за йерархична клъстеризация, който заема междинното място между подходите, базирани на центроиди, и подходите, базирани на представителни обекти. Вместо да използва само един центроид или представителен обект за представяне на клъстера, алгоритъмът избира за тази цел *фиксиран брой представителни обекти* (точки в пространството). Тези представителни за клъстер точки се генерират на два етапа: първо, се избират обектите, които са добре разпределени по целия клъстер - това означава, че всяка точка на клъстера става близка до един от тези обекти. На вторият етап избраните обекти се “стягат” (придвижват) към центъра на клъстера, чрез свиването на тяхното разстояние до центъра в определена пропорция (*фактор на свиване*). Самата клъстеризация се осъществява постъпково чрез сливането на всяка стъпка на два клъстера с най-близко разположени двойки от представителни обекти (принадлежащи на различни клъстери) в един нов клъстер.

**Базов алгоритъм за клъстеризация CURE**
***Параметри:*** $k$ – желания брой на клъстери
$c$ – брой на представителни обекти в клъстера
$\alpha$ – фактор на свиване

1. Инициализирай клъстерното множество (всяка точка – отделен клъстер)

2. За всеки клъстер $C_i$ намери най-близки клъстер $C_j$ (на този етап разстоянието между клъстери е просто разстояние между съответните точки от тези клъстери)
3. Направи сливането на два най-близки клъстера $C_k$ и $C_l$ в клъстер $C_m$ (както винаги $C_m$ е обединение от точки от клъстери $C_k$ и $C_l$ )
4. За клъстер $C_m$ избери $c$ представителни (добре разпределени в пространството) точки:
    i. Избери първата точка, най-отдалечена от центроида на $C_m$
    ii. Докато $c$ точки не бъдат избрани направи:
    Избери то(ката, която най-далечна по отношение на предишната
5 Премести избраните $c$ точки по направление към центроида на клъстера с фактор на свиване α: **p** = **p** + α (**mean - p**), където **p** е координати на представителната точка, **mean** - е координати на центроида.
6 Определи клъстер, който е най-близък към $C_m$, като разстоянието между клъстери се определя като разстояние между две *най-близки представителни точки* на съответните клъстери.
7 Направи сливане на най-близката двойка клъстери
8 Повтори стъпки 3-7 докато не се останат k клъстера.



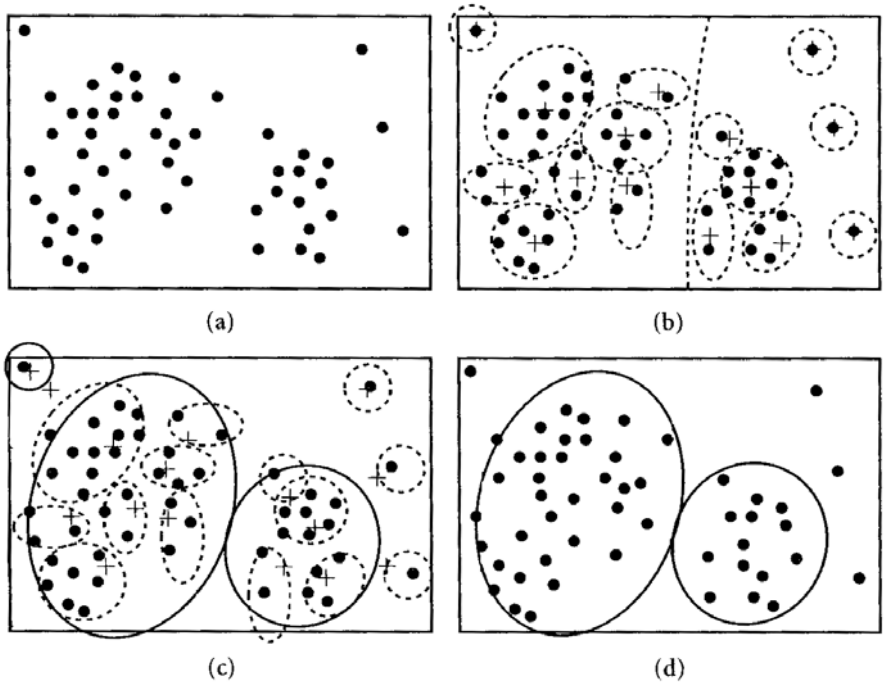

*Пример на клъстеризация с базов алгоритъм на CURE*

Използването на повече от една представителна точка за всеки клъстер позволява на CURE да се наглася добре към клъстери с несферична форма. Стягането или уплътняването на клъстери позволява да бъде намалена роля на екстремните обекти. По тази причина CURE е по-устойчив към наличието на екстремните обекти и идентифицира клъстери с несферична форма и голямото различие в размера. Той може да се прилага към големи бази данни без намаляване на качеството на клъстеризацията.

За работата с големи множества от данни CURE прилага техниката, комбинираща подбор на данни (извадки) с разделянето: една случайно направена извадка отначало се разделя на сегменти, като всеки сегмент се подлага на частична клъстеризация. След това тези предварителни клъстери се клъстеризират отново, за

да се получи желаната клъстеризация. Общата схема за работата на CURE може да се представи по следния начин:

1. От оригиналното множество от обекти се прави случайната извадка.
2. Извадката се разделя на определено множество от непресичащи се сегменти
3. Всеки сегмент се подлага на частичната клъстеризация
4. Екстремните обекти се премахват чрез използване на случайни извадки. Ако един клъстер нараства прекалено бавно, той се премахва.
5. Изпълнява се агломеративната клъстеризация на частични клъстери. Представителните точки, попаднали във всеки новосъздаден клъстер се стягат в посоката към центъра на клъстера със зададения фактор на свиване. Тези точки представят клъстер и определят неговата форма.
6. Данните (обекти) се маркират с етикети на съответните клъстери.




*Фиг. 6.8. Клъстеризация на множеството от точки чрез метода CURE*

*Пример 6.1.* Да предположим, че има определено множество от точки (обекти), разположени в една правоъгълна област. Случайната извадка от тези обекти е показана на фиг. 6.8(a). Обектите се разделят на два сегмента, които след това частично се клъстеризират на базата на минимално средно разстояние. Частичните клъстери са означени с пунктирани линии на фиг 6.8(b). Представителният обект за всеки клъстер е показан със знак "+". По-нататък, частичните клъстери отново се клъстеризират. Получените като резултат от този процес два клъстера са показани с плътни линии на фиг. 6.8(c). Представителните точки на всеки клъстер се получават от предвижването на представителните точки на частични клъстери към центъра на новия клъстер в пропорцията, определяна от фактора на свиване. Те определят формата на всеки клъстер. Така началните обекти се разделят на два клъстера, като екстремалните обекти се изключват, като това е показано на фиг. 6.8(d).

За клъстеризацията CURE се нуждае само от едно сканиране на цялата база данни. Изчислителната сложност на алгоритъма е *O(n)* – линейна, по отношения на броя обекти в базата. Анализът показа, че макар някои от параметрите на CURE (размер на извадката, броя на желаните клъстери, фактор на свиване) могат да варират в определени граници без да променят силно качеството на клъстеризацията, в общия случай те оказват значителното влияние на крайните резултати. Основният недостатък на CURE е невъзможност да работи с номинални атрибути.