

АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА CLUSTERING

Както участниците може би са забелязали (а би трябвало и да са се досетили от релативното оценяване), за задачата няма полиномиално решение (тя е NP-hard). Задачата за разпределянето на дадени обекти в групи е известна, особено в насоки като статистиката и извличането на данни (data mining). Тъй като задачата е NP-hard, единственото, което участниците можеше да направят, е да напишат умна апроксимация. Дадения full feedback би трябвало да помогне значително на участниците в това.

Може би най-лесното решение, което очаквам повечето състезатели да направят, е да генерират рандом позиции на K-те точки (докато имат време) и да вземат най-доброто от тях. Това решение е много лесно за писане и също толкова неефективно. Все пак то би хванало някакви точки (около 20).

Очевидна оптимизация е да приложим горното решение, и за всяка итерация (тоест генерирани рандом позиции на K-те точки) да го оптимизираме докато можем – в случая докато стигнем до някакъв локален оптимум. Най-лесната оптимизация, която може да се приложи, е да се направи hill-climbing. Пробваме да „мръднем“ всяка от точките някакво разстояние нагоре, надолу, наляво или надясно, като гледаме в коя посока отговорът се подобрява. Докато се подобрява, мърдаме някоя от точките в някоя от посоките. Ако преместването на никоя от точките в никоя от посоките не даде по-добър резултат, намаляваме разстоянието, с което мърдаме (правим по-финни „настройки“ на координатите). Когато и това спре да дава по-добри резултати (или достигнем достатъчно малки стойности на преместване) спираме и генерираме нов набор от точки, които да оптимизираме, докато ни стига времето. В тази насока може много да се оптимизира, което предполагам и ще направят по-добрите състезатели. Предвиден е full feedback, който да позволи на състезателите да не се престарават с оптимизациите (тоест ако стигнат 100 точки да спрат).

Авторското решение, изненадващо, ползва дори по-прост алгоритъм, който дава (в повечето случаи) по-добри резултати. Решението е базирано на K-Means method (http://en.wikipedia.org/wiki/K-means_clustering), който е лесно приложим в дадената задача. В началото избираме позициите на K-те точки по случаен начин (да кажем взимаме произволни K от дадените ни N точки и в техните координати слагаме първоначално новите точки). Това гарантира, че никое куче няма да бъде „неизползвано“ в началото, тъй като точката, в която е сложено, ще е на разстояние нула. Всяка от старите точки „си харесва“ по една от новите (най-близката до нея), и всяка от новите точки се грижи за една или повече от старите. Това първоначално разпределение по нищо не се различава от предложените по-горе алгоритми. Въпросът е как можем да оптимизираме това разпределение по-ефективно. Бихме постигнали по-добър резултат ако преместим новите точки в средното аритметично на точките, за които се грижат. Правим това, но след преместванията е възможно някоя от старите точки да е по-близо до друга нова точка (тоест да принадлежи на клъстер, различен от текущо й разпределение). Така преизчисляваме коя точка към кой клъстер принадлежи и правим отново средно аритметично, като преместваме кучетата пазачи, оптимизирайки отговора отново. Този процес на

преизчисляване и местене рано или късно завършва, когато след преместване на новите точки никоя от старите не попада в нов клъстер (тоест алгоритъмът е сходящ към локален оптимум). За съжаление, също като при горните решения, този оптимум е локален, а не глобален, съответно решението не е оптимално. Все пак намерените оптимуми са по-добри от намираните с hill-climbing. Сложността на предложеното решение е силно зависима от първоначалния избор на позиции на точките (което определя колко итерации ще са нужни за достигане до оптимума). Все пак можем да използваме таймер за да измерваме времето и да приключим изпълнението на програмата, когато преминем някаква граница (да кажем 0.9 секунди). До тогава са се изпълнили достатъчен брой пълни итерации на алгоритъма за да сме намерили задоволително добро решение.

Автор: Александър Георгиев